Automatisierte Auswertung von Crashsimulationen unterschiedlicher Fahrzeug-Entwicklungsstände mit Methoden des maschinellen Lernens

Dissertation
zur Erlangung eines Doktorgrades

in der
Fakultät für Maschinenbau und Sicherheitstechnik
der
Bergischen Universität Wuppertal

vorgelegt von
David Kracker
aus Sindelfingen

Wuppertal 2023

Tag der mündlichen Prüfung: 12.09.2023

David Kracker

Automatisierte Auswertung von Crashsimulationen unterschiedlicher Fahrzeug-Entwicklungsstände mit Methoden des maschinellen Lernens

Dissertation, Bergische Universität Wuppertal,
Fakultät für Maschinenbau und Sicherheitstechnik,
Lehrstuhl für Optimierung mechanischer Strukturen, Januar 2023

Kurzfassung

Die verstärkt digitale Fahrzeugentwicklung ist ein wesentlicher Treiber für die stark ansteigende Anzahl an Crashsimulationen, die mittels der Finiten Elemente (FE) Methode abgebildet werden. Die gleichzeitig steigende Komplexität der Simulationsmodelle und das komplexe Crashverhalten eines Fahrzeugs sorgen dafür, dass die manuelle Auswertung der Simulationen aufwändig und unvollständig ist. Dadurch besteht das Risiko, dass ein unerwünschtes Crashverhalten übersehen wird und dadurch falsche konstruktive Maßnahmen am Fahrzeug getroffen werden. Eine automatisierte Auswertung des Crashverhaltens ist schwierig, da sich die FE-Netze der einzelnen Simulationen unterscheiden.

In dieser Dissertation wird durch ein Diskretisierungsverfahren die Vergleichbarkeit unterschiedlicher FE-Netze sichergestellt, wodurch die weitere maschinelle Analyse der Simulationen gewährleistet wird. Als neue Möglichkeit, um das Crashverhalten für sämtliche Bauteile, Zeitschritte und Auswertungsgrößen automatisiert zu analysieren, wird die Ausreißerdetektion vorgestellt. Dabei wird durch den Vergleich mit Simulationen aus der Historie auffälliges Crashverhalten identifiziert und dem Anwender im Post-Prozessor anschaulich dargestellt. Mit dem Crash-Verhalten-Detektor (CVD) wird der/die Ingenieur*in bei Wiederauftreten eines vorgegebenen Crashverhaltens in neuen Simulationen gewarnt. Dies ermöglicht eine individualisierte Auswertung gemäß den Vorgaben des Anwenders. Die Ausreißerdetektion und der CVD sorgen für eine vollständigere Auswertung der Crashsimulationen, indem sämtliche Informationen automatisiert analysiert werden und der Fokus gezielt auf die relevanten Bereiche einer Crashsimulation gelenkt wird.

Darüber hinaus werden Methoden der Dimensionsreduktion zur Visualisierung einer Schar von Simulationen betrachtet. Es wird untersucht, inwiefern nichtlineare Algorithmen Vorteile gegenüber linearen Verfahren liefern. Darüber hinaus werden unterschiedliche Möglichkeiten zur Berücksichtigung des zeitlichen Verhaltens betrachtet. Durch die Dimensionsreduktion ist es dem Anwender möglich, das Crashverhalten für eine Vielzahl an Simulationen für ausgewählte Bauteile übersichtlich in Streu- und Liniendiagrammen zu visualisieren. Damit können Simulationen aus der Historie, die ein ähnliches Crashverhalten zeigen, identifiziert werden. Diese neu gewonnenen Informationen helfen dem/der Ingenieur*in bei der Definition von neuen konstruktiven Maßnahmen, um das Crashverhalten zu verbessern.

Stichworte: Finite Elemente Methode, automatisierte Auswertung, datengestützte Auswertung, Ausreißerdetektion, Dimensionsreduktion, Klassifikation, Maschinelles Lernen

David Kracker

Automated evaluation of crash simulations with different stages in vehicle development using machine learning methods

Dissertation, University of Wuppertal,
School of Mechanical Engineering and Safety Engineering,
Chair of Optimization of Mechanical Structure, Januar 2023

Abstract

Increased digital vehicle development is a key driver of the sharp rise in the number of crash simulations that are performed using the finite element (FE) method. The simultaneous increase in the complexity of the simulation models and the complex crash behavior of a vehicle mean that manual evaluation of the simulations is time-consuming and incomplete. As a result, there is a risk that conspicuous crash behavior will be overlooked, leading to incorrect design measures being taken on the vehicle. Automated evaluation of the crash behavior is difficult because the FE meshes of the individual simulations differ.

In this dissertation, a discretization method is used to ensure the comparability of different FE meshes, which ensures further analysis of the simulations. Outlier detection is presented as a new method to automatically analyze crash behavior for all components, time steps and evaluation variables. In this process, conspicuous crash behavior is identified by comparison with simulations from the history and clearly presented to the user in the post-processor. With the crash behavior detector (CVD) the engineer is warned in case of recurrence of a given crash behavior in new simulations. This allows an individualized evaluation, according to the user's specifications. The outlier detection and the CVD ensure a more complete evaluation of the crash simulations by analyzing all information automatically and directing the focus specifically to the relevant areas of a crash simulation.

In addition, dimensionality reduction methods for visualizing a bunch of simulations are considered. The extent to which nonlinear algorithms provide advantages over linear methods is investigated. Furthermore, different possibilities are considered how the temporal behavior can be considered. Dimensionality reduction enables the user to clearly visualize the crash behavior for a large number of simulations for selected components in scatter and line diagrams. This allows the identification of simulations from the history that show similar crash behavior. This newly gained information helps the engineer to define new design measures to improve the crash behavior.

Keywords: Finite Element Method, Automated Evaluation, Data-Driven Evaluation, Outlier Detection, Dimensionality Reduction, Classification, Machine Learning

Danksagung

Als erstes möchte mich von ganzem Herzen bei meinem Doktorvater Prof. Dr.-Ing. Axel Schumacher für die hervorragende Betreuung und die wertvollen Impulse während meiner Dissertation bedanken. Seine fachliche Expertise, Hilfsbereitschaft und sein stets offenes Ohr haben maßgeblich dazu beigetragen, dass meine Arbeit einen erfolgreichen Abschluss gefunden hat.
Ebenso geht mein aufrichtiger Dank an Prof. Dr. Jochen Garcke, der mit seiner Unterstützung und seinen konstruktiven Ratschlägen einen wichtigen Beitrag zu meiner Forschung geleistet hat. Die kollegiale Zusammenarbeit habe ich als äußerst bereichernd empfunden.
Ein besonderer Dank geht auch an meine geschätzten Doktorandenkollegen an der Universität Wuppertal. Der regelmäßige Austausch bei Kolloquien und die konstruktiven Diskussionen haben meine Arbeit in vielerlei Hinsicht bereichert. Die gemeinsame Forschungsumgebung hat einen bedeutenden Beitrag zu meinem wissenschaftlichen Fortschritt geleistet.
Darüber hinaus möchte ich mich herzlich bei Dr.-Ing. Constantin Diez für seine wertvollen Beiträge und die inspirierenden Diskussionen bedanken. Das hat mir den Einstieg und die Bearbeitung meines Themas deutlich erleichtert.
Ein herzliches Dankeschön gilt auch meinen Kollegen bei der Dr. Ing. h.c. F. Porsche AG, insbesondere meinem Betreuer und Freund Dr.-Ing. Pit Schwanitz. Seine fachliche Kompetenz und seine stets unterstützende Haltung haben mir wertvolle Einblicke und Perspektiven eröffnet.
Ebenfalls möchte ich mich ganz herzlich bei meiner Mitdoktorandin, Kollegin und mittlerweile guter Freundin Dr.-Ing. Jana Büttner bedanken. Ihre motivierende Art und unsere gemeinsamen Diskussionen haben die Forschungsarbeit zu einem inspirierenden Prozess gemacht. Gerade in schwierigen Phasen der Doktorarbeit war unsere gegenseitige Unterstützung eine wichtige Hilfe.
Besonderer Erwähnung bedarf das Doktorandennetzwerk der Dr. Ing. h.c. F. Porsche AG, insbesondere der AG-B. Die Zusammenarbeit und der Erfahrungsaustausch innerhalb dieses Netzwerks haben die Qualität meiner Arbeit nachhaltig beeinflusst.
Darüber hinaus möchte ich mich herzlich bei Anas Azouni, Marco Antonio Peredo Tiburcio, Alexandra Otto, Marius Stach, Margarita Venediktova, Dominik Bodenstein und Revan Kumar Dhanasekaran für die wertvolle Unterstützung im Rahmen ihrer Abschlussarbeiten/Praktika bedanken.
Ein weiterer besonderer Dank geht an meine Familie, die mich stets unterstützt hat. Ihr Verständnis, ihre Ermutigung und ihre bedingungslose Unterstützung haben mir die notwendige Kraft und Zuversicht gegeben, um diese Herausforderung erfolgreich zu meistern.
Zuletzt, aber keinesfalls am wenigsten, möchte ich mich bei meiner Frau bedanken. Ihre Geduld, Liebe und Unterstützung waren meine ständige Quelle der Inspiration und Kraft. Ohne Sie wäre dieser Weg deutlich steiniger gewesen.

Gerlingen, im Dezember 2023 David Kracker

Inhaltsverzeichnis

Abkürzungs- und Symbolverzeichnis

Abkürzungen

AK	Ausreißerkennwert
AUC	Accuracy Under the Curve/Fläche unter der Kurve
BG	Batch-Größe
CAE	Computer Aided Engineering
CVD	Crash-Verhalten-Detektor
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DLV	Duplizieren Löschen Voxel
DLZ	Duplizieren Löschen Zeit
DLZ-RS	Duplizieren Löschen Zeit – Rauschen Skalierung
DLZ-SR	Duplizieren Löschen Zeit – Skalierung Rauschen
DPCA	Difference Principle Component Analysis
DS	Datenset
ED	Erreichbarkeits-Distanz
EuroNCAP	European New Car Assessment Programme
FE	Finite Elemente
FEM	Finite Elemente Methode
Isomap	Isometric Feature Mapping
KNN	k-nächste Nachbarn
KNND	k-nächste Nachbarn Distanz
LED	Lokale Erreichbarkeitsdistanz
LOF	Local Outlier Factor
LSTM	Long Short Term Memory
ML	Maschinelles Lernen
N_{NNE}	Anzahl von Nicht-Null Elementen
ODB	Offset Deformable Barrier
PCA	Hauptkomponentenanalyse/Principle Component Analysis
PCAD	PCA Distanz
SOS	Stochastic Outlier Selection
t-SNE	t-distributed Stochastic Neighbor Embedding
UMAP	Uniform Manifold Approximation and Projection
UAK	Unskalierter Ausreißerkennwert

Operatoren

x	Skalar
$\boldsymbol{x}$	Vektor
$\boldsymbol{X}$	Matrix
x_i	i-te Komponente von $\boldsymbol{x}$

Lateinische Zeichen

$\boldsymbol{A}$	Affinitätsmatrix
$\boldsymbol{B}$	Binding Probability Matrix
C	Schlupfvariable Support Vector Maschine
d	Dimension der Eingangsdaten
d_{red}	Zieldimension der Dimensionsreduktion
$\boldsymbol{D}$	Unähnlichkeitsmatrix
e	Anzahl zu modifizierender Zeilen/Spalten der Modifikation *Duplizieren&Löschen*
k	Anzahl zu berücksichtigender nächster Nachbarn, beziehungsweise der k-te Nachbar
l	Längeparameter
p	Perplexität
Q_{NX}	Qualitätskriterium
Q_{AVG}	Mittelwert von Q_{NX}
Q_{lokal}	lokale Qualität
Q_{global}	globale Qualität
r	Skalierungsfaktor der Modifikation *Rauschen*
$\boldsymbol{R}$	Matrix der niedrigdimensionalen Einbettung
s	Skalierungsfaktor für die Modifikation *Skalierung*
t	Skalierungsfaktor der Standardabweichung
$\boldsymbol{T}$	Transformationsmatrix
S	Schwellwert
$\boldsymbol{x}$	Merkmalsvektor
x_1	Merkmal 1
x_2	Merkmal 2
$\boldsymbol{X}$	Matrix der Eingangsdaten

Griechische Zeichen

φ	Azimutwinkel
ϑ	Polarwinkel
σ	Standardabweichung

1 Einleitung

1.1 Problemstellung und Motivation

Verkürzte Entwicklungszeiten, Kosteneinsparungen und strengere gesetzliche Anforderungen an die Fahrzeuge sorgen für einen verstärkt digitalen Entwicklungsprozess und eine steigende Anzahl an Simulationen. Im Kontext der Fahrzeugsicherheit wird das Crashverhalten anhand der Finite Elemente (FE) Methode simulativ erprobt. Aufgrund der Komplexität der Modelle ist die Auswertung aufwändig und zeitintensiv. In dieser Dissertation werden Ansätze entwickelt, die Teile des manuellen Auswertungsprozesses automatisieren, sodass der/die Ingenieur*in bei der Auswertung unterstützt wird und geeignete konstruktive Maßnahmen treffen kann, um ein bestimmtes Crashverhalten zu erwirken.

In mehreren Iterationen wird der Prozess der Datenanalyse und Modifikation der Konstruktion wiederholt, bis schlussendlich sowohl gesetzliche, Verbraucherschutz als auch Inhouse Anforderungen an das Crashverhalten der Fahrzeuge erfüllt sind. Im bisherigen Prozess erfolgt die Analyse der Crashsimulationen überwiegend manuell. Hierfür werden die Simulationen in einem Post-Prozessor wie beispielsweise dem *Animator* der Firma *gns mbH* visualisiert wobei sich verschiedene Herausforderungen ergeben, die im Folgenden erläutert werden.

Um eine hohe Präzision der Simulationsmodelle zu erzielen, ist die Komplexität der Modelle kontinuierlich im Laufe der Jahre angestiegen. Während beispielsweise bei der Entwicklung des Porsche 928 in den achtziger Jahren Simulationsmodelle ungefähr 10000 FE verwendet wurden, bestehen heutige digitale Prototypen aus bis zu 20 Millionen Schalenelementen.

Um die Dateigröße der Simulationsergebnisse zu kontrollieren, werden die Resultate nicht für jeden berechneten Zeitschritt abgespeichert, sondern nur alle 2ms. Für einen Crash mit 120ms Dauer liegen somit Informationen über 60 Zeitpunkte aus der Simulation vor. Der Begriff Zeitschritt wird im Folgenden für diese Zeitpunkte verwendet (Zeitschritt 1 bis Zeitschritt 60).

Aufgrund der Komplexität der Simulationsmodelle ergeben sich daher lange Ladezeiten für den Import der Ergebnisse, was die Auswertung verzögert. Darüber hinaus ist es nicht für jede einzelne Simulation möglich, die Ergebnisse vollumfänglich auszuwerten. Da sich ein Fahrzeug aus hunderten Bauteilen zusammensetzt und nur eine begrenzte Zeit für die Auswertung zur Verfügung steht, ist es nicht möglich jede Auswertungsgröße an allen geometrischen Orten im Detail zu analysieren. Der Prozess wird darüber hinaus dadurch erschwert, dass für jeden Knoten und jedes Element der FE-Daten verschiedene Auswertungsgrößen wie beispielsweise Verschiebungen, Geschwindigkeiten, Beschleunigungen oder plastische Dehnungen vorliegen und diese jeweils unterschiedliche Informationen über das Crashverhalten enthalten.

Die verstärkte digitale Fahrzeugentwicklung hat zur Folge, dass die Anzahl durchgeführter Simulationen jedes Jahr ansteigt. Beispielsweise werden bei der Dr. Ing. h.c. F. Porsche AG zurzeit etwa 20000 Gesamtfahrzeug-Crashsimulationen pro Jahr durchgeführt. Es wird deutlich, dass eine vollständige, interaktive Auswertung mit den bisherigen Methoden nicht für jede Simulation möglich ist. Dadurch besteht das Risiko, dass Bereiche im Fahrzeug mit unerwünschtem Crashverhalten übersehen werden. Auf Basis der aus der Analyse gewonnenen Erkenntnisse in Bezug auf die Änderungen an den Modellen werden neue konstruktive

Maßnahmen am Fahrzeug getroffen. Bei einer unvollständigen Analyse des Crashverhaltens, besteht deshalb das Risiko, dass diese überhaupt nicht alle tatsächlich relevanten Stellen im Fahrzeug betreffen. Das Resultat sind teure Entwicklungsschleifen und ein erhöhter Zeitbedarf für die Produktentwicklung.

Darüber hinaus sollte der Fokus auf der Weiterentwicklung der Fahrzeuge liegen und weniger auf der Durchführung der Auswertung und der Fehlersuche in den Crashsimulationen. Daher wird ein digitaler Assistent benötigt, der automatisiert sämtliche Crash-Simulationsdaten analysiert und dem Anwender die relevanten Informationen im Anschluss übersichtlich und schnell zur Verfügung stellt.

Eine Möglichkeit, die Auswertung zu automatisieren und auszuweiten, bietet eine sogenannte Standardauswertung. Dabei handelt es sich um einen standardisierten Bericht, der für jede Simulation automatisiert erstellt wird. Eine exemplarische Seite eines solchen Berichts ist in Abbildung 1 dargestellt. An vorab definierten Stellen des Fahrzeugs, die erfahrungsgemäß für den jeweiligen Lastfall relevant sind, werden Auswertungsgrößen wie die Verschiebung, Geschwindigkeit oder Beschleunigung einzelner Knoten, Elemente und Cross-Sections ausgewertet. Diese Werte werden über der Zeit aufgetragen und dem/der Ingenieur*in in Form von Liniendiagrammen dargestellt. Exemplarisch ist der Verlauf der Verschiebung eines Auswertungsknotens der Schottwand (rot eingefärbt) in x-Richtung abgebildet. Darüber hinaus können standardisierte Momentaufnahmen der deformierten Geometrien an vorher festgelegten Stellen im Fahrzeug erzeugt und in dem Bericht dargestellt werden.

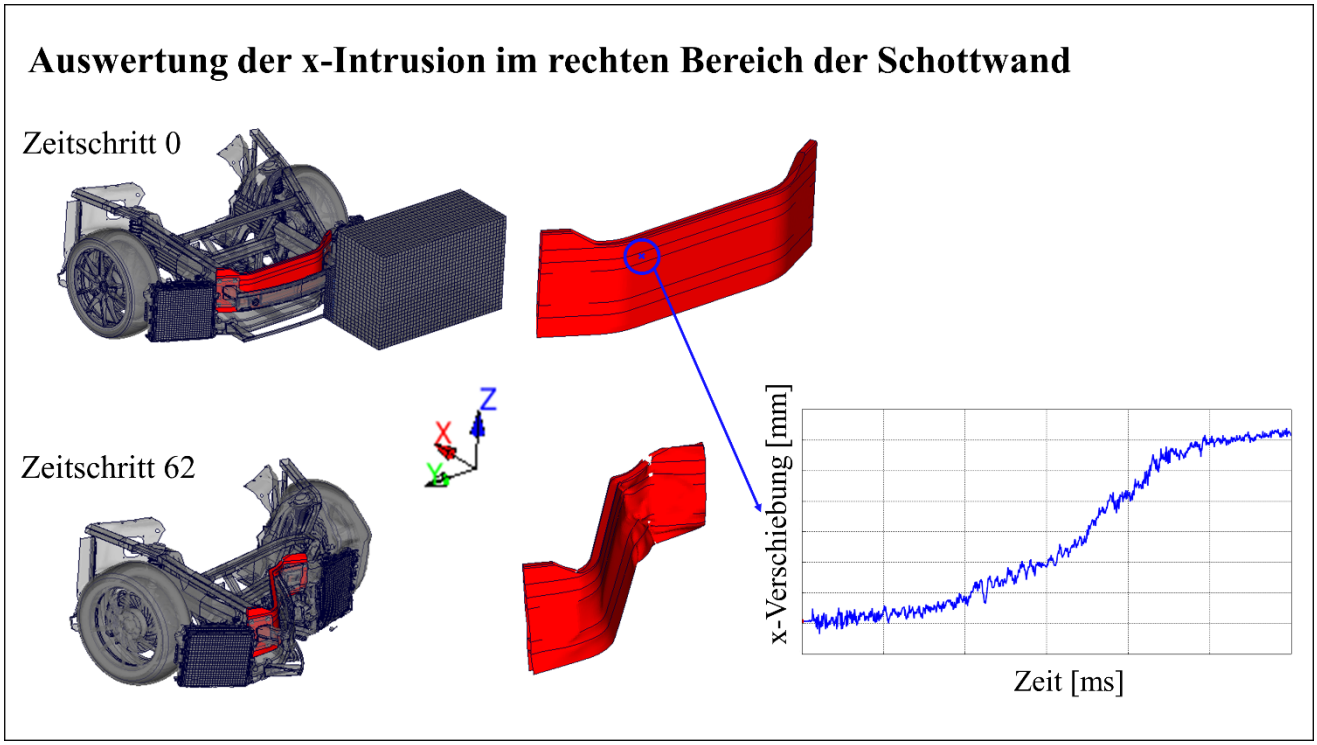


*Abbildung 1: Exemplarische Darstellung der Standardauswertung, auf der der zeitliche Verlauf der Verschiebung in x-Richtung eines Auswertungsknotens der Schottwand (rot) ausgewertet wird. Standardisierte Momentaufnahmen des Fahrzeugs (beispielsweise vor und nach dem Crash, hier: Zeitschritt 0 bzw. 62) helfen dem/der Ingenieur*in zusätzlich dabei, eine Übersicht über das Crashverhalten zu bekommen*

Diese Vorgehensweise ermöglicht zwar eine standardisierte Auswertung, diese ist jedoch in keinerlei Hinsicht vollumfänglich, da sie lediglich an vorab festgelegten Stellen für bestimmte Auswertungsgrößen erfolgt. Bereiche und Zeitschritte, die nicht explizit vorab ausgewählt wurden, werden demnach nicht berücksichtigt. Dadurch besteht erneut das Risiko, dass relevante Ereignisse übersehen werden.

Ein Problem für eine automatisierte Analyse der Ergebnisse aus Crashsimulationen unterschiedlicher Fahrzeug-Entwicklungsstände ergibt sich dadurch, dass sich die FE-Netze der einzelnen Simulationen unterscheiden. Dies ist sowohl auf geometrische Unterschiede als auch auf Neuvernetzungen der Geometrie zurückzuführen. Abbildung 2 zeigt ein Beispiel, in dem dieselbe Geometrie durch zwei unterschiedliche FE-Netze abgebildet wird. Diese heterogene Datenrepräsentation erschwert den automatisierten maschinellen Vergleich der einzelnen Varianten. Aus diesem Grund können in der bisher angewendeten Standardauswertung lediglich vorab definierte Knoten und Elemente (die in allen Simulationen vorhanden sind) ausgewertet werden, jedoch nicht das Deformationsverhalten ganzer Bauteile mit einbezogen werden.

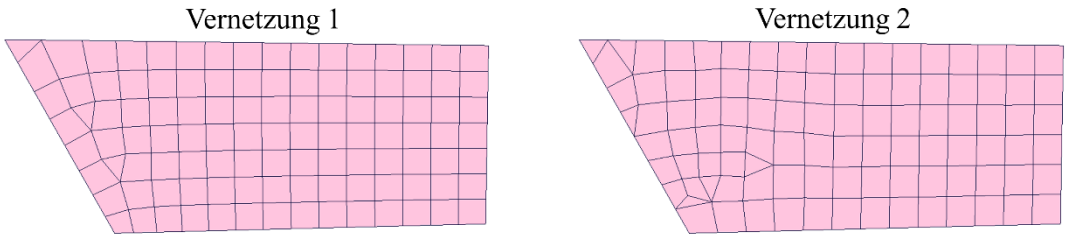


Abbildung 2: Darstellung zweier Bauteile mit derselben Geometrie, jedoch unterschiedlicher Vernetzung

Oftmals ist es hilfreich, vergangene Simulationen als Referenz heranzuziehen und auf Basis der Entwicklungshistorie konstruktive Verbesserungen zu definieren. Der Vergleich mit anderen Varianten ermöglicht dem/der Ingenieur*in die Auswirkungen der getroffenen konstruktiven Änderungen besser zu verstehen und neue Verbesserungen vorzunehmen. Darüber hinaus kann ein Crashverhalten, das bereits in der Vergangenheit durch eine Maßnahme abgestellt werden konnte, meistens erneut durch dieselbe oder eine ähnliche Modifikation verbessert werden (z.B. Einfügen einer Anfaltsicke).

Der/Die Ingenieur*in kennt jedoch meistens lediglich die letzten 10-20 Simulationen im Detail und weiß wie sich dort Modelländerungen auf das Crashverhalten und alle betrachteten Auswertungsgrößen ausgewirkt haben. Sollte die relevante Variante jedoch weiter zurück liegen und der/die Ingenieur*in sich demnach nicht mehr im Detail an die Simulation erinnern, gestaltet sich deren Suche aufwändig und zeitintensiv. Dies liegt daran, dass die Datenbank derzeit nicht nach einem bestimmtem Crashverhalten durchsucht werden kann (z.B. zeige alle Simulationen, in denen der *Längsträger* nach links knickt). Die Simulationsergebnisse sind nicht einheitlich und umfassend dokumentiert und können somit nicht nach deren Crashverhalten kategorisiert werden. Es obliegt jedem Anwender individuell, eine eigene Dokumentation der Simulationsergebnisse zu pflegen.

Wurde eine Variante von Interesse gefunden, kann diese gemeinsam mit anderen Simulationen in den *Animator* importiert werden. Neben der Komplexität der Daten limitieren auch zusätzlich begrenzte Hardware-Ressourcen (RAM – Random Access Memory) die Anzahl der

Simulationen, die gleichzeitig betrachtet werden können. Eine Simulation beansprucht in etwa einen Speicherplatz von 1-3 GB. Bei der Komplexität der Simulationsmodelle ist es üblicherweise möglich 5-10 Simulationen gleichzeitig zu importieren. Demnach ist ein vollständiger Vergleich mit der gesamten Entwicklungshistorie unter Verwendung von Gesamtfahrzeugsimulationen nicht möglich. Kausalitäten zwischen Modelländerungen und deren Auswirkung auf das Crashverhalten können demzufolge nur für wenige Gesamtfahrzeugsimulationen analysiert und für die Weiterentwicklung der Fahrzeuge verwendet werden.

1.2 Zielsetzung und Aufbau der Dissertation

Mit der vorliegenden Arbeit soll ein neuer Analyseansatz erarbeitet und bewertet werden, der dabei hilft, den dargestellten Problemen zu begegnen. Während in der bisherigen Standardauswertung lediglich einzelne FE-Knoten und Elemente ausgewertet werden, soll die mit dieser Arbeit vorgestellte Analyse des Crashverhaltens ganzer Bauteile dem Anwender ein vollständigeres Bild über das Crashverhalten des gesamten Fahrzeugs liefern.

Nach der Einleitung wird in Kapitel 2 der Stand der Technik reflektiert. Darüber hinaus werden die theoretischen Grundlagen für die in dieser Arbeit verwendeten Algorithmen erläutert.

Kapitel 3 beschreibt, wie der neue Analyseansatz in den bisherigen Prozess zur Auswertung von Crashsimulationen integriert werden kann.

Im darauffolgenden 4. Kapitel werden die verwendeten Crash-Simulationsdaten beschrieben. Neben Informationen zu dem Fahrzeugmodell und Lastfall werden einzelne in den Analysen verwendete Bauteile und deren Crashverhalten vorgestellt.

Der neue Analyseansatz gliedert sich in vier Bausteine, die das Crashverhalten jedes einzelnen Bauteils des Gesamtfahrzeugs separat analysieren (siehe Abbildung 3) und in den Kapiteln 5-8 detailliert beschrieben und untersucht werden.

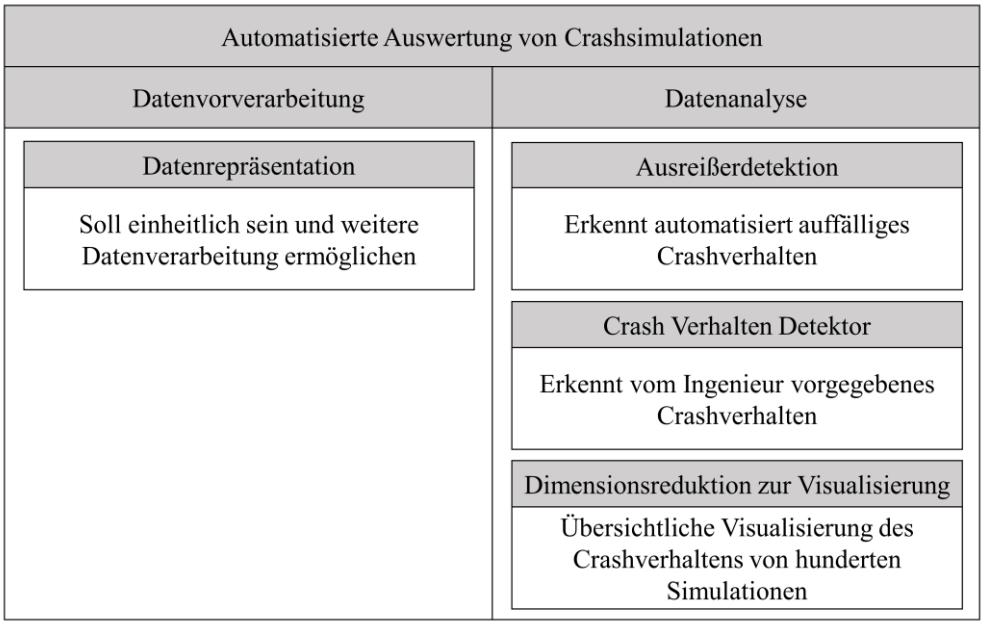


Abbildung 3: Darstellung der einzelnen Bestandteile der neuen automatisierten Auswertung von Crashsimulationen aufgeteilt in die Kategorien Datenvorverarbeitung und Datenanalyse

Das Ziel ist, Simulationen aus unterschiedlichen Fahrzeug-Entwicklungsständen automatisiert auszuwerten. Weil sich dadurch die FE-Netze unterscheiden, ist eine Datenvorverarbeitung erforderlich, die die Crash-Simulationsdaten zunächst in eine einheitliche Darstellung transformiert und damit die weitere maschinelle Weiterverarbeitung ermöglicht. Eine Anforderung an die neue Datenrepräsentation ist, dass die Informationen über das Crashverhalten unabhängig von den geometrischen Unterschieden zwischen den einzelnen Simulationen ausgewertet werden können. Darüber hinaus soll die Datenvorverarbeitung keine weiteren Interaktionen oder Informationen durch den Anwender erfordern und automatisiert in einem Post-Prozessing Skript ablaufen, das beispielsweise auch die Erzeugung der bisherigen Standardauswertung übernimmt. Darüber hinaus soll die neue Datenrepräsentation für den Anwender möglichst intuitiv interpretierbar sein, um die Akzeptanz für das neue Verfahren zu erhöhen und für eine bessere Nachvollziehbarkeit der Datenverarbeitung zu sorgen. In Kapitel 5 werden zwei aus der Literatur bekannte Diskretisierungsverfahren verglichen und mittels eines Qualitätskriteriums untersucht, wie gut die Informationen aus der Simulation in den diskretisierten Daten abgebildet werden. Darüber hinaus werden der benötigte Speicherplatz sowie die Berechnungszeit der Diskretisierung ausgewertet.

Nachdem eine einheitliche Datenrepräsentation sichergestellt ist, sollen anschließend die Analyseverfahren angewendet werden (Abbildung 3, rechte Spalte). Hierzu zählen neben der Ausreißerdetektion der Crash-Verhalten-Detektor (CVD) und die Visualisierung von Crashverhalten mittels Dimensionsreduktion.

Die Ausreißerdetektion soll automatisiert und ohne weitere Informationen durch den Anwender auffälliges Crashverhalten in neuen Simulationen detektieren. Hierzu sollen die Informationen aus den Vorgänger-Simulationen berücksichtigt werden. Die Ergebnisse der Ausreißerdetektion sollen in einem Post-Prozessor darstellbar sein, in welchem die Bauteile gemäß ihrer Auffälligkeit farblich hervorgehoben werden (beispielsweise Grün: unauffällig, Orange: leicht auffällig, Rot: sehr auffällig). Dadurch soll der Anwender fokussierter bei der Analyse des Crashverhaltens vorgehen können. Eine Möglichkeit, wie diese Darstellung aussehen könnte ist in Abbildung 4 gezeigt.

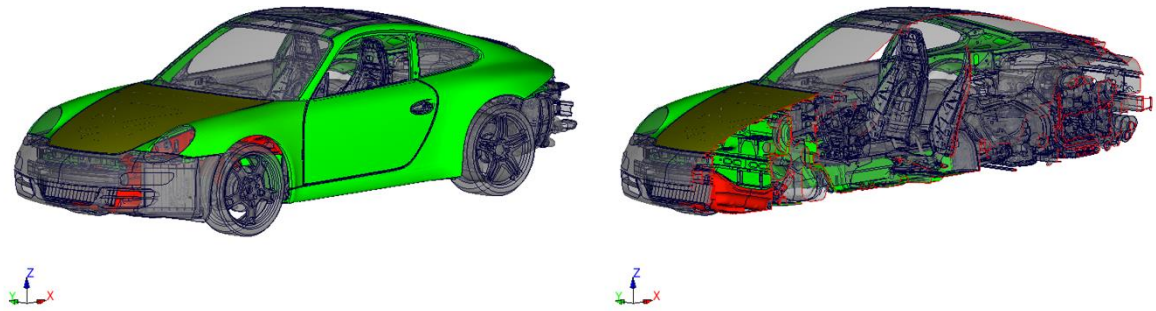


Abbildung 4: Exemplarische farbliche Darstellung im Post-Prozessor. Die Bauteile sind gemäß ihrer Auffälligkeit bzw. den Klassifikationsergebnissen des CVD im Kontext einer Vielzahl an Simulationen eingefärbt

Durch die farbliche Abstufung sollen auffällige Bereiche sofort sowohl räumlich als auch zeitabhängig identifiziert werden können. Damit soll verhindert werden, dass der/die Ingenieur*in wichtige Effekte im Crash übersieht, was zu teuren Entwicklungsschleifen und zu falschen Maßnahmen zur vermeintlichen Verbesserung des Crashverhaltens führen kann. In Kapitel 6 wird das neue Verfahren für die automatisierte Detektion von auffälligem Crashverhalten vorgestellt. Es werden aus der Literatur bekannte Algorithmen zur Ausreißerdetektion verglichen. Neben deren Leistungsfähigkeit, Ausreißer zu identifizieren, werden der Einfluss vorhandener Hyperparameter sowie der zeitliche Berechnungsaufwand bewertet.

Der CVD soll dem Anwender die Möglichkeit geben, ein für ihn interessantes Crashverhalten eines Bauteils (z.B. Längsträger knickt nach links) zu markieren und künftig eine Warnung bei einem Wiederauftreten dieses Verhaltens in neuen Simulationen zu erhalten. Die gewünschte Warnung soll im Post-Prozessor durch eine farbliche Hervorhebung des entsprechenden Bauteils erfolgen, sobald das erlernte Crashverhalten wiedererkannt wird. Die gesamte zeitliche Information aus der Simulation definiert das Crashverhalten eines Bauteils. Daher soll der CVD sämtliche Zeitschritte und nicht nur einen einzelnen berücksichtigen. Für die Ausreißerdetektion soll der Anwender keine weiteren Informationen bereitstellen müssen. Dahingegen ist es beim CVD erforderlich, dass der Anwender die Trainingsdaten für das Lernverfahren entsprechend dem jeweiligen Crashverhalten vorab kategorisiert. Mit dem trainierten Modell können anschließend neue Simulationen ausgewertet werden. Da das Kategorisieren der Simulationen von der Anzahl an zu analysierenden Bauteilen (bestimmt der Anwender selbst) abhängt, kann dieser Prozess für den Anwender zeitaufwändig sein. Daher soll der CVD aus möglichst wenig vorab kategorisierten Daten eine möglichst hohe Genauigkeit bei der Wiedererkennung von Crashverhalten erzielen und damit den Aufwand für die Kategorisierung der Trainingsdaten reduzieren. Im 7. Kapitel werden unterschiedliche Herangehensweisen für die Umsetzung des CVD verglichen. Klassische Algorithmen des maschinellen Lernens werden mit einem neuen Ansatz aus dem Bereiche des sogenannten selbstüberwachten Lernens vergleichen. Es wird untersucht, wie sich die Anzahl kategorisierter Trainingsdaten auf die Ergebnisqualität des CVD auswirkt. Darüber hinaus wird der Einfluss unterschiedlicher Hyperparameter analysiert.

Als weiteres Analyseverfahren soll die Dimensionsreduktion verwendet werden. Dabei ist das Ziel, das oftmals komplexe Crashverhalten eines Bauteils über eine Vielzahl von Simulationen hinweg gleichzeitig und übersichtlich im niedrigdimensionalen Raum darzustellen. Dabei soll ähnliches Crashverhalten in ähnlichen Bereichen und unterschiedliches Crashverhalten in entfernten Bereichen dargestellt werden, wodurch beispielsweise schnell Bifurkationen identifiziert werden können. Da Crashsimulationen nichtlinear sind, soll insbesondere untersucht werden ob nichtlineare Verfahren der Dimensionsreduktion Vorteile gegenüber linearen Ansätzen bei der Visualisierung des Crashverhaltens bieten. Damit interaktiv und möglichst ohne lange Wartezeiten aufgrund der Dimensionsreduktion mit den Crash-Simulationsdaten gearbeitet werden kann, soll die Dimensionsreduktion eine möglichst geringe Berechnungszeit aufweisen (wenige Sekunden). Neben den geometrischen Informationen soll die gesamte zeitliche Information aus der Crashsimulation berücksichtigt werden.

Die niedrigdimensionale Darstellung des Crashverhaltens eines Bauteils aus hunderten Simulationen liefert Erkenntnisse über die Entwicklungshistorie und kann mit Metainformationen wie beispielsweise einem Screenshot von dem zu betrachtenden deformierten Bauteil und weiteren Parametern, die eine Simulation charakterisieren (z.B. Wandstärke, Material, etc.), angereichert werden. Dadurch sollen die Einflüsse von konstruktiven Maßnahmen identifiziert werden können, was bei künftigen Entscheidungen über neue konstruktive Verbesserungen unterstützen kann. In Kapitel 8 werden unterschiedliche Varianten untersucht, das zeitliche Verhalten der Crashsimulation in der Dimensionsreduktion zu berücksichtigen. Darüber hinaus werden lineare und nichtlineare Verfahren aus der Literatur miteinander verglichen. Die Ergebnisse der Dimensionsreduktion werden sowohl subjektiv als auch objektiv mittels eines Qualitätskriteriums bewertet. Zudem wird der zeitliche Berechnungsaufwand in der Bewertung der Algorithmen berücksichtigt.

Im 9. Kapitel wird der neue Analyseansatz an einem Gesamtfahrzeugbeispiel demonstriert. Kapitel 10 fasst die Ergebnisse der Arbeit abschließend zusammen. Dabei wird auf die Potentiale und Einschränkungen des neuen Analyseansatzes eingegangen. Darüber hinaus wird ein Ausblick gegeben sowie auch eine Beschreibung möglicher Erweiterungen.

Die mit dieser Arbeit vorgestellten neuen Methoden sollen den/die Ingenieur*in bei der Auswertung der Crashsimulationen im Hinblick auf die Zeit als auch den Umfang der Erkenntnismöglichkeiten effizient unterstützen. Die gewonnene Zeit und auch die Erkenntnisse aus der Datenanalyse, sollen in die Weiterentwicklung der Fahrzeuge investiert werden. Darüber hinaus sollen die Robustheit und Vollständigkeit der Auswertung gesteigert werden, indem alle Informationen aus der Simulation automatisiert ausgewertet werden. Der/Die Ingenieur*in soll gleichzeitig bei der Definition von neuen konstruktiven Verbesserungen unterstützt werden, indem die dafür relevanten Informationen von dutzenden oder gar hunderten Simulationen berücksichtigt werden. Der zu dieser Arbeit entwickelte Analyseansatz soll dazu beitragen, zeitintensive oder gar unnötige Entwicklungsschleifen zu vermeiden, die Sicherheit der Fahrzeuge zu erhöhen und den Prozess der Crashauslegung zu beschleunigen.

2 Stand der Technik und theoretische Grundlagen

2.1 Finite Elemente Grundlagen

Bei der Finite Elemente Methode (FEM) handelt es sich um ein numerisches Verfahren, um komplexe physikalische Problemstellungen zu analysieren (Liu und Quek 2013). Sie findet dort Anwendung, wo eine analytische Lösung nicht möglich ist und diese numerisch approximiert werden muss. Ein bekanntes Anwendungsbeispiel stellen Crashsimulationen dar (Solanki et al. 2004). Hierbei kollidiert ein Fahrzeug bestehend aus hunderten Bauteilen aus unterschiedlichen Materialien mit einem Hindernis. Dabei wird die kinetische Energie in Wärme und plastische Verformung umgewandelt.

Um dies simulativ abzubilden, muss die vorliegende physikalische Problemstellung entsprechend modelliert werden. Nach Roth et al. (2011) sind hierfür vier Schritte notwendig: Approximieren der Geometrie, Vernetzung, Hinzufügen von Materialeigenschaften als auch die Definition von Randbedingungen (gemäß Lastfall).

Insbesondere die Vernetzung ist für den in dieser Arbeit vorgestellten neuen Analyseansatz von Crash-Simulationsdaten von besonderer Bedeutung, da sie festlegt, wie die Simulationsergebnisse vorverarbeitet werden müssen, um sie im Anschluss maschinell analysieren zu können. Die Geometrie des Fahrzeugs und je nach Lastfall die der Barriere wird mittels der sogenannten finiten Elemente (FE) approximiert (siehe Abbildung 2, Seite 3).

Dabei wird abhängig von der Geometrie eines Bauteils insbesondere zwischen 2D und 3D Elementen unterschieden. Während 2D Elemente Dreiecke oder Vierecke sind, handelt es sich bei 3D Elementen um Tetraeder und Hexaeder. Dünnwandige Strukturen, wie Profile oder Bleche werden mit 2D Elementen modelliert, während Volumenkörper, wie beispielsweise Gussbauteile oder Verbindungstechnik (z.B. Kleber, Schrauben, Schweißpunkte/-nähte) mittels 3D Elementen abgebildet werden. Um das physikalische Verhalten des Fahrzeugcrashs genauer zu approximieren, kann die Größe der FE verkleinert werden, wodurch sich jedoch deren Anzahl und damit der Berechnungsaufwand der Simulation erhöht. Heutige Simulationsmodelle der *Dr. Ing. h.c. F. Porsche AG* bestehen aus bis zu 20 Millionen FE.

Im Anschluss werden jedem FE Materialeigenschaften zugewiesen. Dies sind beispielsweise Eigenschaften, die den Zusammenhang zwischen Spannung und Dehnung beschreiben und in einer sogenannten Fließkurve dargestellt werden können. Neben der Dehnung ist die Dehnrate eine weitere Eigenschaft, die den Spannungszustand eines FE definiert. Darüber hinaus kann zwischen linearem und nichtlinearem Materialverhalten unterschieden werden. Aufgrund der hohen Geschwindigkeiten und großen Verformungen im Fahrzeugcrash werden nichtlineare Materialmodelle verwendet.

Neben der Definition der Materialien, ist die Kontaktmodellierung ein wichtiger Bestandteil der FEM. Sie ist notwendig, um Kontakt und Reibung zwischen dem Fahrzeug und der Barriere sowie der einzelnen Bauteile untereinander abzubilden. Gerade beim Crash berühren sich viele Bauteile aufgrund der großen Deformationen. Wird vom modellierten Kontakt eine Berührung mit einem anderen Bauteil erkannt, werden entsprechende Gegenkräfte aufgebaut, um unphysikalische Durchdringungen zu verhindern.

Ein weiterer Bestandteil der FEM ist die Definition von Randbedingungen (z.B. Rotations- und Translationsfreiheitsgrade) und Lastfalleigenschaften (z.B. Geschwindigkeit).

Im Anschluss werden die physikalischen Zusammenhänge mathematisch formuliert, sodass ein komplexes System aus Differentialgleichungen entsteht, das anschließend gelöst werden muss. Die explizite Zeitintegration liefert als Ergebnis Informationen über den zeitlichen Verlauf des Fahrzeugcrashs (Solanki et al. 2004). Neben den zeitlichen Knotenverschiebungen liegen weitere Informationen über das physikalische Verhalten des Fahrzeugs und der einzelnen FE wie beispielsweise Beschleunigungen oder plastische Dehnungen vor. Im Folgenden wird beschrieben, wie diese Informationen bei der bisherigen Crashauslegung ausgewertet werden.

2.2 Bisheriger Auswertungsprozess

Die Auswertung der Simulationen ist deshalb von besonderer Bedeutung, da auf deren Basis neue konstruktive Maßnahmen getroffen werden, um die Crashauslegung schrittweise zu verbessern. In diesem Abschnitt wird aufgezeigt, wie die manuelle Auswertung bisher mithilfe von Post-Prozessing Software abläuft. Darüber hinaus werden die automatisierte Standardauswertung sowie die Software *Diffcrash* der *SIDACT GmbH* und *Modelcompare* des *Fraunhofer SCAI* erläutert.

Zunächst wird die manuelle Auswertung beschrieben. Vorhandene Berechnungsergebnisse werden mittels entsprechender Post-Prozessing Software interaktiv visualisiert. Anhand dieser Darstellung kann die manuelle Analyse erfolgen. Der/Die Ingenieur*in geht dabei hauptsächlich erfahrungsbasiert vor, da er üblicherweise schon das Crashverhalten eines Fahrzeugs in einem bestimmten Lastfall kennt und dadurch weiß, welche Bereiche im Fahrzeug detailliert analysiert werden sollten. Neben unterschiedlichen Fahrzeugbereichen werden zudem unterschiedliche Auswertungsgrößen betrachtet (z.B. plastische Dehnungen, Verschiebungen, Geschwindigkeiten, Beschleunigungen, Kräfte, etc.). Durch diese Vorgehensweise kann unerwünschtes Crashverhalten entdeckt werden, sodass im nächsten Schritt entsprechende Maßnahmen abgeleitet werden, um ein gewünschtes Crashverhalten zu erwirken. Dazu werden umliegende Bereiche eines Bauteils näher betrachtet und die Modellunterschiede zwischen unterschiedlichen Varianten analysiert. Dadurch können Kausalitäten zwischen den Modelländerungen und dem resultierenden Crashverhalten abgeleitet werden. Daraufhin werden konstruktive Änderungen, Wandstärkenänderungen, Materialänderungen, etc. vorgenommen. Infolgedessen wird ein neues Modell mit den ausgearbeiteten Maßnahmen aufgebaut, eine neue Simulation hierzu berechnet und die Auswertung des Crashverhaltens des abgeänderten Modells beginnt von vorne.

Um diese sehr zeitaufwändige manuelle Analyse zu unterstützen und zu automatisieren, kommt zudem die in der Einleitung erläuterte Standardauswertung zum Einsatz. Diese ermöglicht zwar eine standardisierte Auswertung der Simulationen, ist jedoch auf die vom Anwender vorab definierten Darstellungen und Analysen. Darüber hinaus wird lediglich eine einzelne Simulation ausgewertet und keine vergleichende Analyse einer Vielzahl von Varianten von Simulationen, aus denen weitere Erkenntnisse gewonnen werden könnten.

Um die genannten Einschränkungen zu umgehen, wurde in der Vergangenheit entsprechende Software entwickelt, die den/die Ingenieur*in sowohl bei der Auswertung als auch bei der Definition von Maßnahmen unterstützen soll, um ein bestimmtes Crashverhalten erreichen zu können.

Eine solche Software ist *Diffcrash* (SIDACT GmbH 2021). Sie ermöglicht es, Streuungen zwischen Simulationen zu analysieren und auf dem FE-Netz farblich darzustellen. Dies hilft dabei, Bereiche im Fahrzeug mit großen Streuungen im Crashverhalten anschaulich und schnell zu identifizieren (Borsotto et al. 2015). Streuungen können verschiedene Ursachen zu Grunde liegen. Zum einen können sie durch numerische Instabilitäten aufgrund der Bauteilaufteilung für die parallele Berechnung einer Simulation hervorgerufen werden. Darüber hinaus tritt im Crash nichtlineares Verhalten auf (z.B. durch Materialien oder Kontakt). Durch sich plötzlich ändernde Kräfte an den Bauteilen, kann sich das Crashverhalten von den bisherigen Simulationen unterscheiden. Es können Bifurkationen und dadurch Instabilitäten im Lastpfad entstehen.

Neben der Identifikation der Bereiche im Fahrzeug mit besonders hohen Streuungen, ermöglicht *Diffcrash* zudem, Kausalitäten zwischen dem Crashverhalten einzelner Bauteile aufzudecken. Somit können beispielsweise Bauteile zum Zeitpunkt t_1 identifiziert werden, die für Streuungen in anderen Bauteilen zu einem späteren Zeitpunkt t_2 verantwortlich sind. Dadurch wird der Anwender gezielt auf die relevanten Stellen im Fahrzeug aufmerksam gemacht und bei der Weiterentwicklung des Fahrzeugs unterstützt.

Diffcrash liefert dem/der Ingenieur*in Informationen dazu, welche Bauteile sich gegenseitig beeinflussen und unterstützt ihn damit bei der Definition neuer Maßnahmen. In der Praxis werden zusätzlich Modelländerungen betrachtet, um Kausalitäten zwischen dem Simulationsinput und dem resultierenden Crashverhalten herstellen zu können. Die Änderungen an den Modellen sind jedoch des Öfteren schlecht dokumentiert, da diese üblicherweise vom Anwender durchgeführt werden und keine Systematik hierfür vorgegeben ist. Hierfür wurde *ModelCompare* entwickelt (Webseite ModelCompare; Garcke et al. 2017). Diese Software ist als PlugIn für den *Animator* verfügbar und kann Unterschiede zwischen zwei FE-Modellen erkennen und diese entsprechend auf der dreidimensionalen Geometrie des Fahrzeugs farblich visualisieren. Solche Änderungen an den Modellen können sehr vielseitig sein. Es kann sich beispielsweise um geometrische Unterschiede, Variationen in den Materialien und Wandstärken sowie Modifikationen von Schweißpunkten und Starrkörperelementen handeln. Die Analyse ist dabei unabhängig von Meta-Informationen über die einzelnen Bauteile möglich (Garcke et al. 2017). Weder Bauteil Identifikationsnummern noch Bauteilnamen müssen hierfür definiert werden. Den Autoren nach können sich dabei auch die FE-Netze unterscheiden, da effiziente Mapping Verfahren auf Basis der Knotenpositionen für eine geeignete Datenrepräsentation sorgen.

2.3 Finite Elemente Datenrepräsentation

Für die bauteilbasierte Analyse können Informationen der einzelnen FE als beschreibende Merkmale für das Crashverhalten verwendet werden. Beispielsweise können die plastischen Dehnungen betrachtet werden, wenn das Deformationsverhalten analysiert werden soll. Aufgrund unterschiedlicher FE-Netze tritt jedoch das Problem auf, dass unterschiedliche Simulationen über eine unterschiedliche Anzahl an FE verfügen, sodass ein direkter Vergleich nicht möglich ist. Selbst wenn gleich viele FE vorliegen, sorgen Permutationen in der Reihenfolge der FE dafür, dass nicht mehr dieselbe Information der einzelnen Varianten miteinander verglichen werden kann. In diesem Fall wird von unstrukturierte Daten gesprochen. Es gibt jedoch zwei Möglichkeiten, wie dennoch eine automatisierte Analyse von Simulationen mit unterschiedlichen FE-Netzen umgesetzt werden kann.

Zum einen besteht die Option von den Verfahren für die Datenanalyse lediglich jene zu berücksichtigen, die irreguläre Eingangsdaten, wie sie bei FE-Daten aus Crashsimulationen vorliegen, verarbeiten können. Zum anderen könnten die Daten in ein reguläres Format überführt werden, sodass eine wesentlich größere Auswahl von Algorithmen verfügbar wäre. Beide Vorgehensweisen werden im Folgenden beschrieben.

In den vergangenen Jahren wurden bereits einige Algorithmen vorgestellt, die auch irreguläre Daten verarbeiten können. So kann das FE-Netz grundsätzlich auch als Punktewolke betrachtet werden. Für die Verarbeitung von Daten in Form von dreidimensionalen Punktewolken kann beispielsweise PointNet (Qi et al. 2017a) verwendet werden. Durch seine Architektur ist es dazu in der Lage eine Permutationsinvarianz der einzelnen Punkte sicherzustellen und damit fähig, Informationen aus einer Vielzahl irregulären dreidimensionalen Daten zu extrahieren. In Qi et al. (2017b) wird die Erweiterung PointNet++ vorgestellt, durch die eine Verbesserung der Resultate erzielt werden kann, indem die relevanten Punkte der gesamten Wolke identifiziert werden und somit mehr Informationen ausgewertet werden können.

Neben der Interpretation als Punktewolke können die FE-Daten auch als mathematischer Graph betrachtet werden. In Estrach et al. (2014) und Kipf und Welling (2016) werden sogenannte *Graph Convolutional Neural Networks* vorgestellt, um Informationen aus Daten in Form mathematischer Graphen zu extrahieren. Dabei wird die Idee des Faltungsoperators aus dem Bereich der Bildverarbeitung auf Graphen übertragen.

Unabhängig davon ob die FE-Daten als Punktewolke oder mathematischer Graph für die weitere Analyse dargestellt werden, bleibt dabei jedoch das Problem bestehen, dass die Ergebnisse stets von den Trainingsdaten abhängen. Wenn also neue Simulationen hinzugekommen sind, muss in regelmäßigen Abständen stets auch der Trainingsprozess ergänzt werden. Dies sorgt jedoch dafür, dass sich die resultierenden Merkmalsvektoren dynamisch verändern und schränkt durch einen höheren Aufwand bei der Prozessintegration die Flexibilität in der praktischen Anwendung ein. Beispielsweise müssen Modelle, die später im Prozess eingesetzt werden und die als Dateninput diese Merkmalsvektoren erhalten, ebenfalls neu trainiert werden, was einen nicht unerheblichen prozessualen und zeitlichen Aufwand bedingt. Darüber hinaus ist es bei neuronalen Netzen nicht möglich, das Crashverhalten unabhängig von den Geometrieunterschieden zwischen den einzelnen Simulationen auszuwerten.

Ein weiterer Nachteil dieser Methoden ist, dass die Ergebnisse in Form von Merkmalsvektoren keine geometrische Interpretation zulassen, was hingegen bei der Überführung der FE-Daten in ein strukturiertes Datenformat nach wie vor gewährleistet ist.

Eine einfache Lösung das Crashverhalten eines Bauteils aus FE-Simulationen in strukturierter Form abzubilden, stellt eine Repräsentation als Bild dar. Mit Hilfe der dreidimensionalen Visualisierung der Simulationsergebnisse in einem Post-Prozessor können Screenshots des Crashverhaltens erzeugt und als Datenbasis für die weitere Analyse verwendet werden. Damit verschiebt sich die Auswertung von dreidimensionalen Daten zu einer Analyse einer Vielzahl von Bildern. Ist die spätere Datenanalyse beispielsweise die Klassifikation des Crashverhaltens, können etablierte Klassifikationsalgorithmen aus der Bildverarbeitung verwendet werden. In den vergangenen Jahren wurden große Fortschritte in der Verarbeitung von Bildern mittels neuronaler Netze erreicht. Insbesondere Deep Learning (DL) und sogenannte Convolutional Neural Networks (CNN) sorgen hier für Durchbrüche hinsichtlich der Klassifikationsgenauigkeiten (LeCun et al. 1989).

Anstatt das Crashverhalten lediglich anhand eines Bildes aus nur einem Blickwinkel zu erzeugen, schlagen die Autoren in Su et al. (2015) vor, eine Vielzahl von Blickwinkeln zu verwenden, um dreidimensionale Daten klassifizieren zu können, und stellen ein sogenanntes Mulit-View CNN vor. In Qi et al. (2016) zeigen die Autoren auf, dass die Darstellung von dreidimensionalen Geometrien in Form von Bildern aus verschiedenen Blickwinkeln durch CNN teilweise höhere Genauigkeiten erzielen als Algorithmen, die die gesamte Information der dreidimensionalen Daten verarbeiten.

Bei beiden Vorgehensweisen (Screenshots) besteht jedoch der Nachteil, dass Hinterschnitte von Bauteilen nicht berücksichtigt werden. Darüber hinaus muss eine Vielzahl an Bildern erzeugt werden: Zum einen aufgrund der Berücksichtigung verschiedener Blickwinkel, zum anderen, da verschiedene Auswertungsgrößen analysiert und dadurch je Blickwinkel mehrere Darstellungen mit entsprechenden farblichen Abbildungen der Auswertungsgrößen auf dem FE-Netz erzeugt werden müssen. Darüber hinaus muss dieses Vorgehen für jeden Zeitschritt wiederholt werden, um das gesamte Crashverhalten zu berücksichtigen. Dies verursacht einen hohen Speicherbedarf. Zudem ist trotz der Erlangung dieser verschiedenen Blickwinkel und Auswertungsgrößen nicht sichergestellt, dass sämtliche Informationen aus den originalen 3D-Daten in der neuen Datenrepräsentation enthalten sind. Daher wird diese Form der Darstellung in den Untersuchungen nicht betrachtet.

Eine weitere Möglichkeit, um unterschiedliche FE-Netze in ein strukturiertes Format zu überführen, stellen Mapping-Verfahren dar. In Garcke et al. (2017) werden diese verwendet, um Modellunterschiede zwischen zwei Simulationsvarianten zu identifizieren. Für das Mapping muss eine Simulation definiert werden, deren FE-Netz als Referenz verwendet wird, um daraufhin die FE-Netze der anderen Simulationen in dessen Struktur zu überführen. Der Vorteil des Mapping-Verfahrens liegt den Autoren nach darin, dass die Informationen des FE-Netzes im Gegensatz zu sogenannten Diskretisierungsansätzen in voller Auflösung vorliegen. Darüber hinaus ist es möglich, mittels der so gewonnenen Datenrepräsentation das Crashverhalten unabhängig von Geometrieunterschieden auszuwerten. Allerdings wirkt sich nachteilig aus, dass gegenüber Diskretisierungsverfahren mehr Speicherplatz benötigt wird. Darüber hinaus ist die Definition einer Referenzsimulation erforderlich, deren FE-Netz als Ausgangspunkt dient.

Für jedes Fahrzeugprojekt und Derivat (z.B. Coupe, Cabrio) müssen sorgfältig Referenzsimulationen bestimmt werden. Dabei kann eine „ungeeignete“ Referenz zu falschen und unvollständigen Resultaten führen. Wird beispielsweise als Referenz ein Cabriolet definiert obwohl in folgenden Simulationen auch Coupes betrachtet werden, führt dies dazu, dass nie das Dach der Coupes in der neuen Datenrepräsentation abgebildet wird. Aus diesen Gründen werden Mapping Verfahren in dieser Arbeit nicht betrachtet. Für entsprechende Anwendungen dieses Verfahrens im Kontext von Crash Simulationen wird auf (Garcke und Iza-Teran 2015; Bohn et al. 2013; Iza-Teran 2014) verwiesen.

Als weitere Möglichkeit der Datenrepräsentation eignen sich Diskretisierungsverfahren, um die originalen FE-Netze zu approximieren. Auch diese sind dazu in der Lage, die Informationen über die Geometrieunterschiede und das Crashverhalten getrennt voneinander darzustellen. Darüber hinaus erlauben die resultierenden Datenrepräsentationen nach wie vor eine geometrische Interpretation des ursprünglichen Bauteils. Es werden im Folgenden ein eindimensionaler, zweidimensionaler sowie dreidimensionaler Ansatz erläutert.

In Diez et al. (2016) und Diez (2019) wird ein Algorithmus vorgestellt, der profilförmige Bauteile durch eine eindimensionale Schwerpunktlinie abbildet, auf die Informationen des FE-Netzes projiziert werden. Es wird nachgewiesen, dass sich das Verfahren dazu eignet, das Crashverhalten profilförmiger Bauteile entsprechend abzubilden. Da in der vorliegenden Arbeit jedoch auch Bauteile mit komplexen Geometrien berücksichtigt werden sollen, wird dieser Ansatz nicht betrachtet.

Neben der eindimensionalen Abbildung existieren Verfahren, die die Daten durch eine zweidimensionale Repräsentation erneut in Form von Bildern darstellen. Diese Bilder stammen jedoch nicht aus Screenshots der Crashergebnisse, sondern aus der Projektion der dreidimensionalen FE-Daten auf eine zweidimensionale Kugeloberfläche. Spruegel und Wartzack (2014, 2015) verwenden diese Art der Datenrepräsentation, um Bauteile durch neuronale Netze zu identifizieren. In Spruegel et al. (2017; 2018; 2021) und Bickel et al. (2019) wird eine weitere Aufgabenstellung betrachtet, in der FE-Simulationen mittels eines Klassifikators auf Plausibilität geprüft werden können. Die genannten Veröffentlichungen haben das gemeinsame Ziel, die FE-Daten durch Bilder darzustellen und mittels neuronaler Netze weiter zu verarbeiten. Die sphärische Projektion der dreidimensionalen FE-Daten auf eine zweidimensionale Kugeloberfläche wird in der vorliegenden Arbeit verwendet und daher im Folgenden in Anlehnung an (Spruegel et al. 2018) näher erläutert.

Bevor das FE-Netz projiziert werden kann, wird zunächst eine Koordinatentransformation durch Anwendung einer Hauptkomponentenanalyse (Englisch: *Principal Component Analysis* - PCA) durchgeführt. Dadurch wird das Bauteil an neuen Achsen ausgerichtet und es lässt sich die Rotationsinvarianz bei der Bauteilerkennung sicherstellen.

Im Anschluss wird der geometrische Schwerpunkt des Bauteils ermittelt. Daraufhin wird jedes einzelne FE des Bauteils ausgehend vom Mittelpunkt entlang einer Geraden auf eine Kugeloberfläche projiziert. Das Ergebnis kann als Aufnahme eines Bildes mittels einer 360° Kamera verstanden werden und ist beispielhaft in Abbildung 5 dargestellt. Links ist ein *Längsträger* schematisch abgebildet. Die einzelnen Knoten der FE sind durch gerade Kanten miteinander verbunden, um den Ort der Knoten selbst und die Form der Geometrie

hervorzuheben. Um die sphärische Projektion intuitiv zu verstehen, kann man sich vorstellen, in der Mitte des Bauteils zu stehen und ein Bild mit einer 360 Grad Kamera aufzunehmen. Das Resultat für den Längsträger ist rechts in der Abbildung dargestellt. Die einzelnen Knoten des FE-Netzes treten nicht mehr in regelmäßigen Abständen auf. Stattdessen wird das Netz durch die sphärische Projektion auf die Kugeloberfläche und die anschließende zweidimensionale Darstellung verzerrt. Dies wird durch die vorher geraden Verbindungslinien zwischen den FE-Knoten und resultierenden gekrümmten Linien verdeutlicht. Das eine Ende des Längsträgers ist mittig im Bild dargestellt. Durch die Darstellung der Kugeloberfläche als zweidimensionales Bild sind die Ränder des Bildes periodisch, sodass das das andere Ende des Längsträgers im oberen sowie unteren Bildbereich wiederzufinden ist.

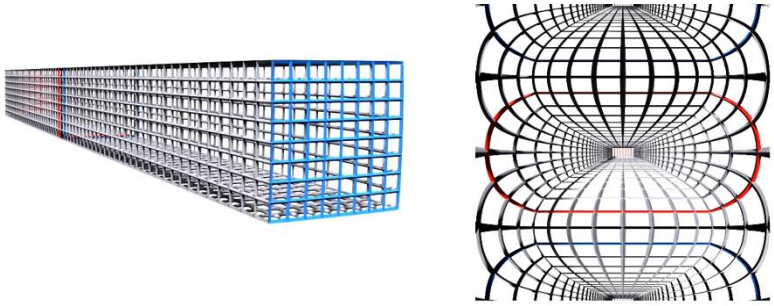


Abbildung 5: Diskretisierung durch Sphärische Projektion. Links: Schematische Darstellung des FE-Gitters eines Längsträgers; Rechts: Sphärische Projektion des Gitters (Filker 2020)

Nach der Projektion der FE wird die Kugeloberfläche diskretisiert und die FE werden in die neu entstehenden Flächen einsortiert. Diese Flächen können als Pixel interpretiert und das Ergebnis der Diskretisierung somit als zweidimensionales Bild dargestellt werden.

Bis zu diesem Schritt ist in den erzeugten Daten lediglich die Geometrieinformation enthalten. Im Anschluss werden Auswertungsgrößen aus der Simulation hinzugefügt, sodass die Informationen über das Crashverhalten abgebildet werden. Als Resultat liegen in Spruegel et al. (2021) für die sphärische Projektion der FE 20 zweidimensionale Matrizen vor, die jeweils unterschiedliche Auswertungsgrößen repräsentieren.

Die Autoren nennen als Nachteil dieser Vorgehensweise, dass durch die Projektion auf eine zweidimensionale Oberfläche Informationen über das zugrunde liegende Crashverhalten verloren gehen. Dabei verhindert das Projektionsverfahren, dass Hinterschnitte hinreichend abgebildet werden. Zudem sorgt die anschließende Diskretisierung dafür, dass kleine geometrische Merkmale nur schwierig hinreichend genau abzubilden sind.

Eine weitere Möglichkeit, die Daten zu diskretisieren, bietet das dreidimensionale Voxel-Verfahren. Voxel können als dreidimensionale Pixel verstanden werden. Somit ist es möglich, die dreidimensionale Struktur der originalen Daten im Gegensatz zum Sphären-Verfahren zu erhalten. Das Voxel-Verfahren ist in vielen Anwendungen bekannt und wird beispielsweise in der Optimierung mechanischer Strukturen verwendet Schumacher (2020), um die Topologie eines Bauteils zu approximieren. Darüber hinaus wird sie im Kontext der künstlichen Intelligenz eingesetzt, um beispielsweise dreidimensionale Eingangsdaten zu klassifizieren (Qi et al. 2016).

Die Autoren stellen in Wang et al. (2019) das sogenannte *NormalNet* vor, eine spezielle Art eines CNN. Diese verwendet die Normalenvektoren der FE-Oberfläche als beschreibende Merkmale und verarbeitet eine Voxel-Datenrepräsentation, um 3D-Daten zu klassifizieren.

Es existieren verschiedene Möglichkeiten, wie ein solches Voxel-Netz erzeugt werden kann. Unabhängig von der konkreten Vorgehensweise haben derartige Verfahren gemeinsam, dass im Anschluss die Informationen der FE-Daten in die entsprechenden Voxel übertragen werden. Als Resultat liegen für jede zu berücksichtigende Auswertungsgröße aus der Simulation dreidimensionale Matrizen vor. Abbildung 6 zeigt beispielhaft das Ergebnis eines durch Voxel diskretisierten *Längsträgers*. Das vordere Ende des Längsträgers im linken Bild ist abgeschrägt (rot markiert). Dies ist ebenfalls in den diskretisierten Voxel-Daten im rechten Bild zu erkennen. Rechts ist die Anzahl der FE, die in einen Voxel einsortiert werden, farblich hervorgehoben. Dass die Kanten entlang der Länge des Längsträgers abgeschrägt sind, kann daran erkannt werden, dass die Voxel entlang der Kanten des Bauteils einen helleren Farbwert aufweisen und damit weniger FE beinhalten als beispielsweise jene im mittleren Bereich der Außenflächen.

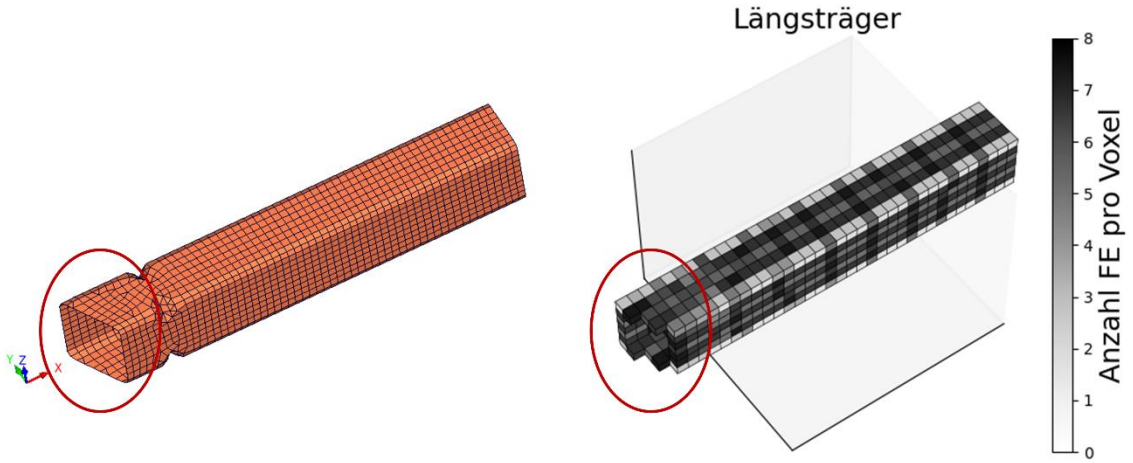


Abbildung 6: Diskretisierung durch Voxel-Verfahren: Links: Darstellung eines Längsträgers mit abgeschrägtem vorderen Ende (rot markiert); Rechts: Resultat der Voxel-Diskretisierung. Farblich hervorgehoben ist die Anzahl der FE pro Voxel

Der Vorteil der Voxel gegenüber der sphärischen Diskretisierung ist, dass die dreidimensionale Struktur der originalen Daten erhalten bleibt. Damit lassen sich die dreidimensionalen Daten besser approximieren. Es bleibt jedoch der Nachteil, dass aufgrund der Diskretisierung Informationen über lokale Eigenschaften der Daten schwierig abzubilden sind. Eine feinere Diskretisierung verschafft hier Abhilfe. Dabei muss jedoch beachtet werden, dass die Komplexität der resultierenden Ergebnisdaten kubisch skaliert, während bei dem Sphären-Verfahren eine quadratische Skalierung vorliegt.

2.4 Ausreißerdetektion

Die Definition eines Ausreißers beschreibt Hawkins (1980) wie folgt:

„Ein Ausreißer ist eine Beobachtung, die so sehr von den anderen Beobachtungen abweicht, dass der Verdacht besteht, dass sie durch einen anderen Mechanismus entstanden ist."

In einer neuen Crashsimulation können einzelne Bauteile durch ihr Crashverhalten hervorstechen. Weichen diese deutlich von dem der anderen Simulationen ab, wird dieses Verhalten als Ausreißer bezeichnet. Während sich das Fahrzeug und crashrelevante Strukturen im Entwicklungsprozess kontinuierlich weiterentwickeln, können einzelne Änderungen (Steffeslai et al. 2021) dazu führen, dass plötzlich ein bisher noch nie beobachtetes Crashverhalten auftritt. Knickte ein axial belasteter Längsträger in der Vergangenheit immer nach links aus, kann eine konstruktive Änderung an diesem oder einem anderen Bauteil beispielsweise dazu führen, dass der Träger plötzlich nach rechts ausknickt. Neben konstruktiven Änderungen sorgen numerische Effekte und nichtlineare Kontaktformulierungen für Unterschiede zwischen sonst gleichen Simulationsmodellen. Im CAE-Entwicklungsprozess ist es notwendig, diese daraus teils unerwarteten Effekte zu detektieren, um geeignete Maßnahmen zur Verbesserung der crashrelevanten Fahrzeugstrukturen zu treffen. Aus diesem Grund soll die Ausreißerdetektion in neuen Simulationen sämtliche Bauteile mit gegebenenfalls auch neuem auffälligem Crashverhalten identifizieren.

2.4.1 Grundlagen und Stand der Technik

In Anlehnung an Aggarwal (2017) werden im Folgenden die Grundlagen der Ausreißerdetektion zusammengefasst.

Unauffällige Datenpunkte in einem Datenset können mit einem mathematischen Modell beschrieben werden (z.B. lineares Regressionsmodell). Von einem Ausreißer wird gesprochen, wenn Abweichungen von diesem Modell auftreten.

Um durch einen Algorithmus automatisiert Ausreißer erkennen zu können, werden im Allgemeinen Annahmen über die zugrunde liegenden Daten getroffen und diese dementsprechend anhand eines geeigneten mathematischen Modells beschrieben. An dieser Stelle sei angemerkt, dass ein lineares Modell bei stark nichtlinearen Daten stets unzureichende Ergebnisse liefern wird. Ein Modell, das eine Gaußverteilung der Daten annimmt, diese jedoch nicht vorliegt, wird ebenfalls zu schlechten Resultaten führen. Dementsprechend muss stets ein geeignetes Verfahren für die zugrunde liegende Aufgabenstellung und die vorliegenden Daten verwendet werden. So können und sollten im Kontext der Crashsimulation im Vorhinein grundsätzlich keine solcher allgemeinen Annahmen getroffen werden, da das Crashverhalten äußerst komplex und teilweise insbesondere in frühen Entwicklungsphasen unvorhersehbar ist.

Unabhängig von derartigen Annahmen, die ein Algorithmus über die zugrunde liegenden Daten trifft, gibt es Unterscheidungen in der Art und Weise, wie die Resultate dem Anwender dargestellt werden. Zum einen gibt es Algorithmen, die in Form eines binären Werts bewerten, ob die einzelnen Dateninstanzen ein Ausreißerverhalten zeigen oder nicht.

Dadurch erhält der Anwender zwar eine klare Aussage, welche Instanzen auffällig sind. Nachteilig wirkt sich jedoch aus, dass er keine Einschätzung erhält, wie stark ein festgestellter Ausreißer ist.

Des Weiteren gibt es Algorithmen, die einen kontinuierlichen Ausreißerkennwert (AK) liefern, der dementsprechend die Stärke der Auffälligkeit anzeigt. Da ein Fahrzeug aus einer Vielzahl an Bauteilen besteht, die alle ein auffälliges Crashverhalten zeigen können, wird dem Anwender durch einen solchen kontinuierlichen und individuellen Kennwert erleichtert, die besonders kritischen Bereiche von den eher unauffälligen zu unterscheiden. Mit Hilfe eines kontinuierlichen Ausreißerkennwerts können die Bauteile entsprechend farblich im Post-Prozessor markiert werden, sodass der Anwender auf einen Blick eine Übersicht über alle auffälligen Bereiche im Fahrzeug in Bezug auf das Crashverhalten einzelner Bauteil erhält. Aus diesem Grund werden in der vorliegenden Arbeit für die Ausreißerdetektion auch lediglich Algorithmen betrachtet, die einen kontinuierlichen AK als Ergebnis liefern.

Die Autoren in Gao und Tan (2006) beschreiben allerdings, dass bei vielen Algorithmen zur Ausreißerdetektion zwar kontinuierliche Kennwerte als Resultat ausgegeben werden, diese jedoch unskaliert und damit auf keinen festen Wertebereich festgelegt sind. Damit könnten sich selbst innerhalb eines Datensets mit unterschiedlichen Hyperparametern die AK in völlig unterschiedlichen Wertebereichen ergeben. Darüber hinaus können sie sich zudem von Datenset zu Datenset unterscheiden (Kriegel et al. 2009). Da die AK für jedes Bauteil getrennt berechnet werden, liegt jeweils ein neues Datenset zugrunde, was die Vergleichbarkeit der AK der Bauteile erschwert. Auch dieser Umstand wird in der mit dieser Arbeit vorgestellten Methode zur Ausreißerdetektion berücksichtigt und damit die Vergleichbarkeit zwischen den AK unterschiedlicher Bauteile mit unterschiedlichem Crashverhalten sichergestellt.

Neben der Unterscheidung, in welcher Form die Ergebnisse dargestellt werden, kann darüber hinaus auch noch zwischen überwachten und unüberwachten Algorithmen zur Ausreißerdetektion unterschieden werden.

Bei den überwachten Verfahren stellt sich der hohe Aufwand durch das sogenannte Kategorisieren der Daten (in der Literatur auch Labeln genannt) nachteilig dar. Bevor ein Modell trainiert wird, muss der Anwender den Daten eine Kategorie zuweisen (z.B. Ausreißer oder kein Ausreißer). Diese Zuweisung müsste im Fall von Crashsimulationen für jedes Bauteil einzeln erfolgen. Somit müssten Daten von hunderten Bauteilen kategorisiert werden, was einen hohen manuellen Aufwand für den Anwender darstellen würde und in der praktischen Anwendung nicht gewünscht ist. Darüber hinaus ist die Bewertung, ob es sich um auffälliges Crashverhalten handelt oder nicht, stets vom betrachteten Kontext der anderen Simulationen, die für die Analyse herangezogen werden, abhängig. Eine Simulation, die im Kontext der letzten zehn Simulationen auffällig ist, muss dies beispielsweise nicht zwangsläufig in einem anderen Kontext sein (z.B. die letzten 200 Simulationen).

Das Streaming Emerging New Classes (SENC) Problem ist ein weiterer Punkt, der gegen die Verwendung überwachter Lernverfahren spricht. Es beschreibt nach Mu et al. (2017) den Zustand, dass zu Beginn des Modell-Trainings oft nicht sämtliche Kategorien, in die die einzelnen Instanzen eines Datensets eingeteilt werden könnten, bekannt sind. Zu Beginn einer Fahrzeugentwicklung stehen meist wenige Simulationen mit nur begrenzt unterschiedlichem und

auffälligem Crashverhalten zur Verfügung. In der weiteren Entwicklung treten stets neue Crashmodi auf, die von einem überwachten Lernverfahren nicht detektiert werden würden. Im Fahrzeugcrash sind unbegrenzt viele Deformationsmodi für die einzelnen Bauteile denkbar, die nicht alle im Vorhinein bekannt sein können.

Die vorangestellten Punkte zeigen, dass überwachte Lernverfahren für die Ausreißerdetektion nicht praktikabel sind. Aus diesem Grund werden in der vorliegenden Arbeit lediglich unüberwachte Lernverfahren verwendet. Bei diesen kann nach (Amer et al. 2013) zwar schwieriger zwischen einem Rauschen und echten Ausreißern unterschieden werden, da eine vom Anwender vorgegebene Entscheidungsgrenze hierfür fehlt, dafür ermöglichen sie jedoch die Anwendung im Fahrzeugcrash, da der Anwender keine Daten vorab kategorisieren muss. Nach (Aggarwal 2017) werden unüberwachte Verfahren oftmals in der Datenexploration verwendet, bei der die gefundenen Ausreißer dem Anwender aufgezeigt werden, um anwendungsabhängig deren Relevanz zu bewerten.

Darüber hinaus kann neben der Unterscheidung in überwachte und unüberwachte Verfahren, eine Unterteilung in modellbasierte und instanzbasierte Methoden vorgenommen werden. Gemäß der ersten Unterteilung wird vorab ein konkretes Modell trainiert, mit dem im Anschluss neue Daten ausgewertet werden. Bei instanzbasierten Verfahren, wird dagegen kein Modell erstellt. Stattdessen wird das Crashverhalten hinsichtlich Ausreißern für jede Analyse neu ausgewertet. Nach Aggarwal (2017) gehören die bekanntesten Algorithmen zur Ausreißerdetektion zu dieser Kategorie wie beispielsweise das Verfahren *k-nächste Nachbarn Distanz* (*KNND*) (Ramaswamy et al. 2000) und der *Local Outlier Factor* (*LOF*) (Breunig et al. 2000). Da sich der Kontext an Simulationen bei der Ausreißeranalyse von Crashsimulationen für einzelne Untersuchungen regelmäßig ändern kann, werden in dieser Arbeit daher allein instanzbasierte Methoden betrachtet.

Viele Algorithmen verfügen zudem über Hyperparameter (HP) mit denen die Ergebnisgüte der Modelle gesteuert werden kann. Gerade bei der Verfügbarkeit vieler HP ist es schwierig, die jeweils richtigen Werte einzustellen, um die Ergebnisgüte zu maximieren. Dies ist daher oftmals ein iterativer manueller und besonders zeitintensiver Prozess. Verfahren mit einer Vielzahl an solchen HP stellen beispielsweise neuronale Netze dar. Gerade wenn keine kategorisierten Daten zur Verfügung stehen, ist eine automatisierte Optimierung der HP aber ohnehin nicht möglich. Daher werden in dieser Arbeit allein jene betrachtet, die keine (z.B. PCA) beziehungsweise möglichst wenige HP aufweisen.

Gerade wenn mehrere HP-Kombinationen oder sogar unterschiedliche Algorithmen in Frage kommen, besteht die Möglichkeit deren jeweilige Einzelergebnisse in einem sogenannten Ensemble Modell zu kombinieren. Dadurch kann zum einen die Ergebnisgüte gesteigert werden. Zum anderen muss sich der Anwender nicht mit der manuellen Optimierung der HP-Werte beschäftigen, da mehrere HP-Kombinationen zu einem finalen Ergebnis gemittelt werden. Schlechte Ergebnisse von ungeeigneten HP können dabei von guten Resultaten guter Werte ausgeglichen werden. Dies kann zum einen verhindern, dass unauffällige Simulationen fälschlicherweise als Ausreißer gekennzeichnet werden, zum anderen, dass Ausreißer übersehen werden.

Es könnten noch weitere Unterteilungen der Algorithmen in verschiedenste Kategorien vorgenommen werden. Dies ginge jedoch über den Rahmen dieser Dissertation hinaus, weshalb für Details an dieser Stelle auf Aggarwal (2017) verwiesen wird.

In der Literatur existieren bereits erste Ansätze, um im Kontext von Crashsimulationen auffälliges Crashverhalten im Fahrzeug zu identifizieren.

Beispielsweise wird in Steffes-lai et al. (2021) ein Verfahren namens *SimCompare* vorgestellt, mittels welchem zwei Simulationen auf Gemeinsamkeiten und Unterschiede miteinander verglichen werden können. Dies erfolgt basierend auf einem einzelnen Zeitschritt, der vom Anwender festgelegt wird. Angenommen wird die Gleichheit der FE-Netze, wodurch ein Knoten beziehungsweise ein elementbasierter Vergleich ermöglicht wird. Dabei können beliebige Knoten- und Elementfunktionen (z.B. Verschiebungen beziehungsweise plastische Dehnungen) aus der Simulation für die Analyse verwendet werden. Für den Vergleich werden als Berechnungsmaß die L1- und L2-Norm vorgeschlagen. Das Ergebnis kann in einem Post-Prozessor wie dem *Animator* dargestellt werden und zeigt dem Anwender durch eine visuelle Hervorhebung die Bereiche der Abweichungen zwischen den Crashverhalten der beiden Simulationen. Ein Nachteil dieser Vorgehensweise stellt der Umstand dar, dass lediglich zwei Simulationen miteinander verglichen werden können.

2.4.2 Theorie der verwendeten Algorithmen

Nachdem die Grundlagen der Ausreißerdetektion und der Stand der Technik dargestellt wurden, folgt eine Beschreibung der Theorie der in dieser Arbeit verwendeten Algorithmen zur Ausreißerdetektion. Basierend auf den bisher beschriebenen Anforderungen wurden in Voruntersuchungen geeignete Algorithmen, die in der Python-Bibliothek PyOD (Zhao et al. 2019) implementiert sind, verglichen. Die aus diesem Vergleich resultierenden Algorithmen mit den besten Ergebnissen werden in der vorliegenden Arbeit detailliert betrachtet und daher deren Theorien im Folgenden beleuchtet.

2.4.2.1 Principal Component Analysis Distance

Die *Principle Component Analysis Distance* (*PCAD*) basiert auf der *Principle Component Analysis* (*PCA*), die ein bekanntes Verfahren aus der Dimensionsreduktion ist. Sie eignet sich darüber hinaus für die Detektion von Ausreißern in mehrdimensionalen Daten. Im Folgenden wird deren Funktionsweise in Anlehnung an Aggarwal (2017) erläutert.

Die originalen Daten werden durch die Matrix $\boldsymbol{X}$ mit der Form Nxd dargestellt. N entspricht dabei der Anzahl an Dateninstanzen, während d deren jeweilige Dimension abbildet. Das Ziel ist es eine Matrix $\boldsymbol{R}$ der Form Nxd_{red} aufzustellen, die in den d_{red} neuen Merkmalen möglichst viel Informationen über die originalen Daten enthält. Dabei soll $d_{red} << d$ gelten. Zunächst wird die Kovarianzmatrix berechnet. Liegen standardisierte Daten vor, entspricht die Kovarianzmatrix der Korrelationsmatrix. Im Anschluss erfolgt eine Eigenwertzerlegung. Die Eigenvektoren stellen dabei die Richtungen der *Hauptkomponenten* (HK) dar, während die Eigenwerte den jeweils dazugehörigen Varianzen entsprechen. Da das Ziel eine Maximierung der Varianz innerhalb der ersten Eigenvektoren ist, werden die Eigenwerte γ und zugehörigen Eigenvektoren $\boldsymbol{r}$ nach absteigender Varianz der Eigenwerte sortiert. Mit Hilfe der ersten d_{red} Eigenvektoren wird eine

Matrix $\boldsymbol{T}$ der Form dxd_{red} konstruiert. Diese Transformationsmatrix wird im Anschluss dazu verwendet, um mittels

$$\boldsymbol{R} = \boldsymbol{X} * \boldsymbol{T} \qquad \textbf{(2.1)}$$

die Eingangsdaten $\boldsymbol{X}$ in den neuen Unterraum $\boldsymbol{R}$ zu projizieren. Anschaulich betrachtet, findet bei der PCA eine Koordinatentransformation statt. Der Algorithmus beschreibt die Daten mit einem neuen Koordinatensystem, indem die Varianz entlang der neuen Hauptkomponenten jeweils maximiert wird. Die Achsen stehen dabei jeweils orthogonal zueinander. Abbildung 7 stellt diesen Vorgang der Anschaulichkeit halber beispielhaft in zwei Dimensionen dar. Die ursprünglichen Merkmale, die die Daten beschreiben sind dabei mit x_1 und x_2 bezeichnet. Die neuen aus der PCA resultierenden Merkmale heißen Hauptkomponente 1 und 2.

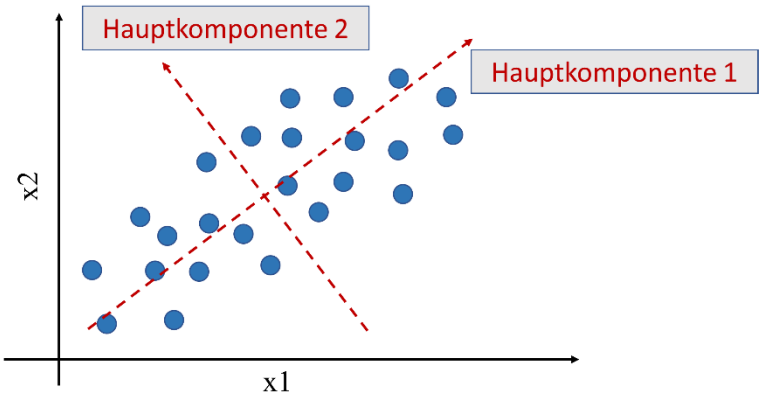


Abbildung 7: Schematische Darstellung der Funktionsweise der PCA

Da die PCA ein varianzbasiertes Verfahren ist, ist eine entsprechende Datenvorverarbeitung durch eine geeignete Skalierung der einzelnen Merkmale notwendig. Dies ist beispielsweise relevant, wenn die einzelnen Merkmale in unterschiedlichen Einheiten vorliegen. Wird das Merkmal x_1 in Gramm gemessen, das Merkmal x_2 dagegen in Kilogramm und beide variieren in einem ähnlichen Wertebereich (z.B. zwischen 0,5 kg und 10 kg) so ist die Varianz des Merkmals x_1 deutlich größer als die des Merkmals x_2. Folglich wird dem Merkmal x_1 bei der PCA wesentlich mehr Bedeutung zugewiesen, als es in der Realität der Fall ist. Als Skalierungsverfahren kann beispielsweise eine Zentrierung oder Standardisierung gewählt werden. Damit wird gleichzeitig sichergestellt, dass die transformierten Daten entlang der Hauptkomponenten ebenfalls zentriert sind.

Bis hierhin sind die Schritte der PCA, wie sie auch für die Dimensionsreduktion verwendet wird, erläutert. Auf diesen basiert die Berechnung der PCAD, die im Folgenden weiter beschrieben wird. Die ersten Hauptkomponenten weisen die größte Streuung auf, während kleinere Hauptkomponenten kleine Streuungen aufweisen. Dadurch, dass die Daten entlang der Hauptkomponenten zentriert sind, sind die Werte kleiner Hauptkomponenten für normale Datenpunkte annähernd Null. Ein Ausreißer ist dadurch charakterisiert, dass sein Wert dieser Hauptkomponenten signifikant von Null abweicht. Oft werden Ausreißer gerade durch die kleinen Hauptkomponenten charakterisiert. Für diese sind die Abstände zum Mittelpunkt der

Daten jedoch meist klein, verglichen mit denen der ersten Hauptkomponenten, bei denen sie selbst für normale Datenpunkte aufgrund der größeren Varianz groß sind. Daher werden die Hauptkomponenten zunächst standardisiert. Im Anschluss wird für jede Dateninstanz über sämtliche Hauptkomponenten hinweg der quadrierte euklidische Abstand zum Mittelpunkt μ der Daten als Ausreißerkennwert verwendet. Die Standardisierung der Hauptkomponenten kann mathematisch umgesetzt werden, indem die Hauptkomponenten durch deren Eigenwert γ dividiert werden, sodass als Ausreißerkennwert für ein Datenpunkt $\boldsymbol{x}$

$$\boldsymbol{PCAD}(\boldsymbol{x}) = \sum_{\boldsymbol{i=0}}^{\boldsymbol{d}} \frac{|(\boldsymbol{x} - \boldsymbol{\mu}) \cdot \boldsymbol{r_i}|^2}{\boldsymbol{\gamma_i}} \qquad \textbf{(2.2)}$$

folgt. Diese Vorgehensweise wird als *Soft-PCA* bezeichnet.

Nach Aggarwal (2017) gibt es neben der eben beschriebenen *Soft-PCA* eine weitere Variante zur Berechnung der Ausreißerkennwerte, die als *Hard-PCA* bezeichnet wird. Der Unterschied liegt darin, dass nicht sämtliche Hauptkomponenten für die Berechnung des Ausreißerkennwerts verwendet werden, sondern lediglich die *k* kleinsten Eigenvektoren. Damit wird ein vom Anwender festzulegender Parameter eingeführt, dessen optimaler Wert nicht intuitiv bestimmbar und darüber hinaus stets abhängig von den Eigenschaften der zugrunde liegenden Daten ist. Darüber hinaus ergibt sich das Problem, dass Ausreißer, die in den ersten Hauptkomponenten zum Vorschein treten, durch diese Vorgehensweise übersehen werden könnten. Der entscheidende Vorteil der *Soft-PCA* ist, dass kein Hyperparameter vom Anwender festgelegt werden muss. Darüber hinaus werden keine Annahmen über die Verteilung der Daten getroffen, über die in Crashsimulationen im Allgemeinen keine Informationen vorliegen. Ein Nachteil sowohl für die *Soft-* als auch die *Hard-PCA* tritt in Datensets mit einer großen Anzahl an Ausreißern zum Vorschein, da diese die Varianz beeinflussen, die für die Skalierung der Hauptkomponenten und die Berechnung der Ausreißerkennwerte verwendet wird. Dadurch besteht das Risiko, dass Ausreißer in den Daten nicht zuverlässig identifiziert werden können. Darüber hinaus sind die Ausreißerkennwerte nicht in einem festen Wertebereich skaliert. Dies bedeutet, dass der Vergleich zwischen den Ausreißerkennwerten verschiedener Bauteile erschwert ist.

In den Voruntersuchungen konnten keine weiteren parameterfreien Algorithmen die Ergebnisse der im Folgenden beschriebenen Algorithmen mit Hyperparametern übertreffen.

2.4.2.2 k-nächste Nachbarn Distanz

Ein bekanntes und mathematisch einfaches Verfahren zur Ausreißerdetektion basiert auf der Berechnung von Distanzen einzelner Datenpunkte zu deren *k*-ten Nachbarn (k-nächste Nachbarn Distanz – KNND). Der Ausreißerkennwert für einen Datenpunkt $\boldsymbol{x}$ ergibt sich damit zu

$$\boldsymbol{KNND}(\boldsymbol{x}) = \textbf{Distanz}(\boldsymbol{x}, \textbf{k-ter Nachbar}(\boldsymbol{x}))\,. \qquad \textbf{(2.3)}$$

Als Distanz kann eine beliebige Funktion wie beispielsweise der euklidische Abstand verwendet werden. Es wird angenommen, dass Ausreißer große und normale Datenpunkte kleine Distanzen zum k-ten Nachbarn aufweisen und dementsprechend hohe beziehungsweise niedrige Ausreißerkennwerte berechnet werden.

Neben dem eben beschriebenen sogenannten exakten KNND-Algorithmus existieren nach Aggarwal (2017) auch Abwandlungen zur Berechnung des Ausreißerkennwerts. Beispielsweise kann anstatt der Distanz zum *k*-ten Nachbarn der Mittelwert aus allen Distanzen bis zum *k*-ten Nachbarn gebildet werden. Neben dem Mittelwert sind weitere statistische Kennwerte denkbar wie beispielweise der Median oder der harmonische Mittelwert.

Der Vorteil von KNND-Verfahren ist neben der einfachen Implementierung auch deren Interpretierbarkeit. Im Vergleich zu anderen Verfahren, erlauben distanzbasierte Methoden darüber hinaus eine bessere Unterscheidung zwischen Rauschen und Ausreißern. Zudem ist es möglich Ausreißer auch in Formationen innerhalb kleinerer Cluster zu identifizieren (Aggarwal 2017; Amer et al. 2013).

Ein Nachteil ist, dass der Anwender einen geeigneten Wert für den Hyperparameter *k* festlegen muss, der im Allgemeinen in unüberwachten Anwendungsszenarien unbekannt ist. Je nach Größe der betrachteten Nachbarschaft können die Ausreißerkennwerte stark variieren, wodurch das Risiko besteht, dass normale Datenpunkte fälschlicherweise als Ausreißer identifiziert und Ausreißer gänzlich übersehen werden könnten. Hier bleibt dem Anwender meist keine andere Wahl, als in einer interaktiven Analyse verschiedene Werte für *k* auszuprobieren oder ein Ensemble Modell bestehend aus den gemittelten Ergebnissen einzelner Modelle mit unterschiedlichen Werten für den Hyperparameter *k* zu verwenden. Darüber hinaus besitzt dieses distanzbasierte Verfahren den Nachteil, dass es sich um eine globale Methode handelt, in der lokale Eigenschaften der Daten nicht berücksichtigt werden. Dies kann dazu führen, dass Ausreißer in Bereichen variierender Dichte nicht zuverlässig detektiert werden (siehe Amer et al. (2013)). Dichtebasierte Verfahren wie der Local Outlier Faktor (LOF), liefern hier bessere Resultate.

2.4.2.3 Local Outlier Factor

Im Gegensatz zum KNND-Algorithmus werden beim *Local Outlier Factor (LOF)* lokale Nachbarschaften bei der Berechnung des Ausreißerkennwerts berücksichtigt. Punkte, die in Bereichen geringer Dichte liegen erhalten dabei einen hohen Ausreißerkennwert, während Punkte in dichten Regionen normalen Datenobjekten entsprechen und einen niedrigen Wert afuweisen. Wie beim KNND-Algorithmus ist die Definition eines Hyperparameters *k* notwendig, der die Größe der zu berücksichtigenden Nachbarschaft festlegt. Im Folgenden wird die Funktionsweise des LOF Verfahrens in Anlehnungen an Breunig et al. (2000) zusammengefasst.

Die Grundlage für die Berechnung des LOF für einen Punkt p bildet die sogenannte Erreichbarkeits-Distanz (ED), die für seine k nächsten Nachbarn berechnet wird. Für zwei Punkte p und o ist sie definiert als

$$\boldsymbol{ED_k(p,o) = \max\{\text{k-te Distanz}(o), \text{Distanz}(p,o)\}} \quad . \qquad \textbf{(2.4)}$$

Die k-te Distanz eines Objekts o entspricht der Distanz von o zum am weitesten entfernten der k-nächsten Nachbarn. Als Distanzmaß kann eine beliebige Distanzfunktion verwendet werden. Üblicherweise wird der euklidische Abstand verwendet. Die Erreichbarkeits-Distanz nimmt den größeren Wert der k-ten Distanz des Objekts o und der euklidischen Distanz zwischen den Punkten p und o an. Für kleine k und einen Punkt p, der sich weit entfernt von Punkt o befindet,

entspricht die Erreichbarkeits-Distanz der euklidischen Distanz zwischen p und o. Für Punkte p, die sich nahe bei o befinden, nimmt die Erreichbarkeits-Distanz den Wert seiner k-ten Distanz an. Durch diese Vorgehensweise werden Fluktuationen in den Distanzen für nahe Punkte p geglättet. Der Parameter k gibt dabei dem Anwender die Kontrolle, wie stark diese Glättung erfolgen soll.

Auf Basis der Erreichbarkeits-Distanzen für die k-nächsten Nachbarn des Punkts p wird die lokale Erreichbarkeits-Dichte (LED) des Punkts p berechnet

$$\boldsymbol{LED_k(p) = 1/\left\{\frac{\sum_{o \in N_K(p)} ED_k(p,o)}{|N_k(p)|}\right\}} \quad . \qquad \textbf{(2.5)}$$

$N_K(p)$ entspricht der Anzahl der k-nächsten Nachbarn von p. Die lokale Erreichbarkeits-Dichte berechnet sich somit aus dem Kehrwert der mittleren Erreichbarkeits-Distanz für den Punkt p. Analog wird *LED(o)* berechnet. Damit ergibt sich der LOF des Punkts p zu

$$\boldsymbol{LOF_k(p) = \frac{\sum_{o \in N_K(p)} \frac{LED_k(o)}{LED_k(p)}}{|N_K(p)|}} \quad . \qquad \textbf{(2.6)}$$

Aus der Formel wird ersichtlich, dass der LOF(p) umso höher ist, desto kleiner die lokale Erreichbarkeits-Dichte von p und desto größer die lokale Erreichbarkeits-Dichte seiner k-nächsten Nachbarn ist. Damit resultieren für isolierte Punkte hohe und für Punkte in dichten Umgebungen niedrige Werte für deren LOF.

Für den Anwender stellt sich an dieser Stelle die Frage, welcher Wert für den Hyperparameter k gewählt werden sollte. In der Veröffentlichung zum LOF Verfahren (Breunig et al. 2000) werden Vorschläge für die Wahl von *k* genannt. Dabei wird eine Ensemble Methode empfohlen, indem eine obere sowie untere Grenze von *k* festgelegt, mehrere Iterationen mit verschiedenen *k* innerhalb der festgelegten Grenzen berechnet und die Ergebnisse der Modelle kombiniert werden. Auf Basis verschiedener Beispieldatensets wird für die untere Grenze empfohlen, Werte größer als 10 zu verwenden, um statistische Fluktuationen in den lokalen Dichten zu verhindern. Für die Definition der oberen Grenze wird angenommen, dass die normalen Datenpunkte in einem Cluster *C* liegen und die Ausreißer außerhalb dieses Clusters in Bereichen niedriger lokaler Dichte. In dieser Situation wird die obere Grenze als Kardinalität des Clusters *C* festgelegt. Für größere Werte werden die Ausreißerdatenpunkte in der Berechnung des LOF für Punkte im Cluster C berücksichtigt. Dadurch verringert sich der LOF der Ausreißer, was das Risiko erhöht diese zu übersehen.

2.4.2.4 Stochastic Outlier Selection

Auch bei dem *Stochastic Outlier Selection* (*SOS*) Algorithmus handelt es sich um ein unüberwachtes Lernverfahren zur Ausreißerdetektion. Dessen Funktionsweise wird in Anlehnung an Janssens (2013) beschrieben. Es wird das Konzept der Affinität verwendet, um zu bewerten, ob es sich um einen auffälligen Datenpunkt handelt. Im Gegensatz zu den bisherigen Algorithmen ist der resultierende Ausreißerkennwert zwischen Null und Eins skaliert. Null entspricht einem unauffälligen Punkt, während eins ein auffälliges Verhalten widerspiegelt.

Daher spricht der Autor in diesem Kontext von einer Ausreißer-Wahrscheinlichkeit, die die Interpretierbarkeit der Ergebnisse für den Anwender steigern soll, da Wahrscheinlichkeiten ein intuitiveres Verständnis liefern als AK, die über keinen festgelegten Wertebereich verfügen.

Ausgehend von den multidimensionalen Eingangsdaten $\boldsymbol{X}$ der Form *Nxd* wird zunächst eine Unähnlichkeitsmatrix $\boldsymbol{D}$ mit den Einträgen $\boldsymbol{d_{ij}}$ berechnet. In der Veröffentlichung zu diesem Algorithmus wird hierzu die euklidische Distanz verwendet, jedoch gleichzeitig auch darauf hingewiesen, dass sich hierfür ebenso beliebige andere Distanzmaße eignen. Im nächsten Schritt erfolgt die Berechnung der sogenannten Affinitätsmatrix $\boldsymbol{A}$ mit den Einträgen $\boldsymbol{a_{ij}}$, indem mittels einer Funktion die Distanzen in einen skalaren Wert zwischen null und eins überführt werden, der ein Maß für die Ähnlichkeit der Instanzen ist. Im SOS Algorithmus wird hierfür eine Gaußfunktion verwendet, sodass sich für die Einträge von $\boldsymbol{A}$ die folgenden Werte ergeben

$$\boldsymbol{a_{ij}} = \begin{cases} \mathbf{exp}\,(-\boldsymbol{d_{ij}^2}/\mathbf{2}\boldsymbol{\sigma_i^2}), & \boldsymbol{wenn\ i \neq j}. \\ \mathbf{0}, & \boldsymbol{wenn\ i = j}. \end{cases} \quad . \tag{2.7}$$

Punkte, die eine geringe Affinität gegenüber den benachbarten Instanzen aufweisen, werden als Ausreißer klassifiziert. Die Affinität eines Datenpunkts zu sich selbst beträgt null. Der Wert d_{ij} entspricht der Distanz zwischen den Punkten *p* und *o* und σ^2 der Varianz des Punkts p. Damit entspricht die Affinität der Wahrscheinlichkeitsdichtefunktion einer Gaußverteilung. Bei einer großen Varianz nimmt der Wert der Affinität mit steigender Distanz nur langsam ab. Desto kleiner die Varianz allerdings wird, desto schneller sinkt auch die Affinität mit steigender Distanz. Die Varianz eines Punkts p berechnet sich aus den Distanzen dessen lokaler Nachbarschaft. Für jeden Punkt wird dieselbe Anzahl nächster Nachbarn berücksichtigt. Diese Anzahl wird durch die sogenannte Perplexität definiert. Sie ist ein Hyperparameter des Algorithmus und mit dem Parameter *k* im KNND Verfahren vergleichbar, allerdings mit dem Unterschied, dass *k* lediglich ganzzahlige Werte annehmen kann, während *p* eine reelle Zahl sein kann. Der Wertebereich reicht dabei von *1* bis *N-1*. Punkte, die in dichten Bereichen liegen, erhalten eine kleine Varianz, sodass die Affinität mit steigenden Distanzen hinreichend schnell abfällt. Für Punkte in Bereichen niedriger Dichte verhält es sich umgekehrt. Desto niedriger die Dichte, desto größer die Varianz, sodass Punkte mit größeren Distanzen eine hohe Affinität aufweisen. Nach Janssens (2013) sorgt die adaptive Varianz dafür, dass Punkte in Bereichen niedriger Dichte nicht sofort als Ausreißer klassifiziert werden, sondern immer deren lokale Nachbarschaft in der Berechnung der Affinität berücksichtigt wird.

Im nächsten Schritt wird die Affinitätsmatrix $\boldsymbol{A}$ durch eine zeilenweise Normierung in die sogenannte *Binding Probability Matrix* $\boldsymbol{B}$ überführt

$$\boldsymbol{b_{ij}} = \frac{\boldsymbol{a_{ij}}}{\sum_{\boldsymbol{n=1}}^{\boldsymbol{N}} \boldsymbol{a_{in}}} \quad . \tag{2.8}$$

Die vorliegenden Daten können so als mathematischer Graph interpretiert werden, in dem die einzelnen Dateninstanzen dem Knoten des Graphen und deren Affinität dem Kantengewicht entsprechen. Im Gegensatz zu diskreten Graphen, in denen entweder eine Kante vorhanden ist oder nicht, wird durch diese Formulierung ein kontinuierlicher Wert zwischen null und eins

ausgegeben. Die *Binding Probability Matrix* $\boldsymbol{B}$ gibt demnach die Wahrscheinlichkeit an, dass zwei Knoten durch eine Kante miteinander verbunden sind.

Im Anschluss wird $\boldsymbol{B}$ verwendet, um die Ausreißerwahrscheinlichkeit für jeden Datenpunkt zu berechnen. Zusammenfassend lässt sich festhalten, dass die Ausreißerwahrscheinlichkeit für den Punkt N_i der Wahrscheinlichkeit entspricht, dass N_i von keinem der anderen Punkte als Nachbar gewählt wird. Diese Wahrscheinlichkeit ist gegeben durch

$$\boldsymbol{p(N_i) = \prod_{j \neq i} (1 - b_{ji})} \quad . \tag{2.9}$$

2.4.2.5 Median Absolute Deviation

Die bisher beschriebenen Algorithmen eignen sich insbesondere für die Ausreißerdetektion in mehrdimensionalen Daten. Deren Ergebnis ist ein Ausreißerkennwert, der die Auffälligkeit der Datenpunkte beschreibt. In manchen Anwendungen kann darüber hinaus eine binäre Aussage darüber hilfreich sein, ob ein Datenpunkt auffällig ist oder nicht. Hierfür existieren Algorithmen, die einen Schwellwert berechnen, um einen Datenpunkt als auffällig oder unauffällig zu klassifizieren. Unterschiedliche statistische Kennzahlen wie die Standardabweichung, der Interquartilsabstand oder die absolute mittlere Abweichung vom Median (Median Absolute Deviation - MAD), können hierfür verwendet werden.

Instanzen, deren Wert ein vom Anwender vorgegebenes Vielfaches der Standardabweichung beziehungsweise des Interquartilsabstandes überschreiten, werden somit als Ausreißer klassifiziert. Der Nachteil ist jedoch, dass die Standardabweichung und der Interquartilsabstand selbst wiederum durch Ausreißer beeinflusst werden. Ein statistischer Kennwert, der demgegenüber weitgehend unbeeinflusst von Ausreißern ist, ist der Median. Daher wird in Ley et al. (2013) empfohlen, die absolute Abweichung des Medians zu verwenden, da diese besonders robust für die Ausreißerdetektion in univariaten Daten ist. Die Funktionsweise wird im Folgenden in Anlehnung an Ley et al. (2013) beschrieben.

Hierfür wird zunächst der Median der Eingangsdaten $\boldsymbol{X}$ berechnet und von jedem einzelnen Wert im Datenset subtrahiert. Im Anschluss daran wird der Betrag dieser Differenzen gebildet und wiederum deren Median bestimmt

$$\boldsymbol{MAD = \mathbf{median}(|X - \mathbf{median}(X)|)} \quad . \tag{2.10}$$

Als Resultat liegt die absolute Abweichung eines jeden Datenpunktes vom Median vor (MAD), die als Maß für die Streuung in den Daten betrachtet werden kann.

Je nach Anwendungsfall und Verteilung der Daten, kann eine Analogie zur Standardabweichung hergestellt werden, indem der MAD mit einem entsprechenden Faktor *b* multipliziert wird

$$\boldsymbol{\sigma = b * MAD} \quad . \tag{2.11}$$

Für normalverteilte Daten wird hierbei b=1,48 angenommen (Ley et al. 2013). Analog zur Vorgehensweise für die Standardabweichung und den Interquartilsabstand, werden sämtliche Datenpunkte als Ausreißer identifiziert, deren Ausreißerkennwert einen Schwellwert

$$\boldsymbol{S = t * \sigma} \qquad \textbf{(2.12)}$$

überschreitet. Der Parameter *t* ist in Analogie zur Standardabweichung ein vom Anwender zu definierender Faktor.

2.5 Klassifikation

Im vorangegangenen Kapitel wird die Ausreißerdetektion beschrieben, um auffälliges Crashverhalten automatisiert in einer Vielzahl von Simulationen zu detektieren. Dabei kann es jedoch vorkommen, dass die daraus resultierenden Ausreißerkennwerte aus der Sicht des Ingenieurs zu niedrig erscheinen und nach seiner Ansicht das Risiko besteht, Bauteile mit auffälligem Crashverhalten zu übersehen. Um dies zu verhindern, ist eine Funktion hilfreich, die das vorab von dem/der Ingenieur*in definierte Crashverhalten in neuen Simulationen wiedererkennt und eine deutliche Warnung ausgibt.

Eine Möglichkeit diese Funktion umzusetzen, stellt beispielsweise die manuelle Merkmalsextraktion aus den hochdimensionalen Rohdaten und das Ableiten von Heuristiken dar, die darüber entscheiden, zu welcher Kategorie das vorliegende Crashverhalten eines Bauteils gehört. Bei Heuristiken handelt es sich um Kriterien, die vom Anwender vorgegeben werden, um Eigenschaften in den Daten auszuwerten und für die finale Prädiktion über die zugehörige Kategorie zu verwenden. Da das Crashverhalten jedoch sehr komplex ist (zeitliche und dreidimensionale räumliche Information über eine Vielzahl von FE) und sich von Bauteil zu Bauteil wesentlich unterscheiden kann, wäre das Auffinden der vorzugebenden Regeln sehr zeitaufwändig sowie die Robustheit und Übertragbarkeit der Ergebnisse für die verschiedensten Bauteile fraglich.

Eine weitere Möglichkeit, ein vorgegebenes Crashverhalten wiederzuerkennen, stellen Algorithmen aus dem maschinellen Lernen dar. Dabei wird dem Algorithmus die Merkmalsextraktion und Klassifikation selbst überlassen, wodurch der Aufwand seitens des Ingenieurs erheblich reduziert werden kann. Daher werden in dieser Arbeit maschinelle Lernverfahren betrachtet, um ein vorgegebenes Crashverhalten in neuen Simulationen wiedererkennen zu können.

2.5.1 Grundlagen und Stand der Technik

Wie bei der Ausreißerdetektion existieren auch bei der Klassifikation überwachte und unüberwachte Lernverfahren. Im Rahmen der Ausreißerdetektion werden überwachte Lernverfahren ausgeschlossen, da der Aufwand für das Kategorisieren sämtlicher Bauteile eines Gesamtfahrzeugs (mehrere Hundert) und aller dabei möglicherweise auftretenden Crashverhalten in der Praxis nicht umsetzbar ist. Für die Detektion von vorgegebenem Crashverhalten müssen jedoch nicht sämtliche Bauteile überwacht werden. Die Anwendung sieht vor, lediglich eine Auswahl an Bauteilen und Crashverhalten mit dem Algorithmus zu analysieren, die beispielsweise durch die automatisierte Ausreißerdetektion nicht hinreichend genau detektiert

werden (durch beispielsweise zu geringe Ausreißerkennwerte). Dadurch ist der Aufwand für das Kategorisieren der Daten an dieser Stelle wesentlich geringer und damit vertretbar.

Es werden dem Lernverfahren also nicht nur die Rohdaten aus der Crashsimulation zur Verfügung gestellt, sondern gleichzeitig auch die zugehörigen Kategorien der einzelnen Dateninstanzen. Auf Basis der Trainingsdaten wird ein Modell trainiert, das im Anschluss die zugehörige Kategorie neuer Daten über eine mathematische Funktion abbildet (siehe Abbildung 8).

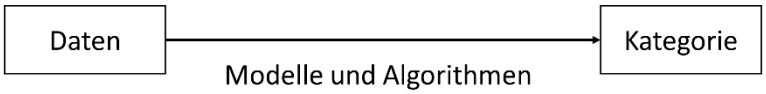


Abbildung 8: Abbildung der Daten über eine mathematische Funktion auf die entsprechende Kategorie (in Anlehnung an Suthaharan (2016))

Abbildung 9 stellt die allgemeine Vorgehensweise für das Finden eines geeigneten überwachten Lernverfahrens in Anlehnung an Kotsiantis et al. (2007) dar. Ausgehend von der vorliegenden Problemstellung werden zunächst die benötigten Daten identifiziert. Für die Detektion von vorgegebenem Crashverhalten wird beispielsweise festgelegt, welche Auswertungsgröße geeignet ist (z.B. plastische Dehnung der FE). Es folgt die Datenaufbereitung. Dazu dient beispielsweise die Voxel-Diskretisierung der FE-Daten. Im Anschluss wird das Datenset in Trainings- und Testdaten aufgeteilt. Manche Algorithmen erfordern eine Skalierung (z.B. Normierung oder Standardisierung) der Daten. Diese erfolgt gegebenenfalls nach der Aufteilung jeweils getrennt für das Trainings- und Testdatenset. Das Trainingsset wird für das Training der Modelle verwendet, während mittels der Testdaten die Ergebnisgüte der trainierten Modelle bewertet wird. Ein bekanntes Kriterium, um die Güte eines Modells zu bewerten ist die sogenannte *Genauigkeit*. Diese wird berechnet, indem die Anzahl der korrekten Prädiktionen $\boldsymbol{N_{korrekt}}$ durch die Gesamtanzahl an Prädiktionen $\boldsymbol{N}$ geteilt wird

$$\textbf{Genauigkeit} = \frac{\boldsymbol{N_{korrekt}}}{\boldsymbol{N}} \quad . \tag{2.13}$$

Das Verhältnis gibt damit den prozentualen Anteil der korrekten Prädiktionen an. Für den Fall, dass der ausgewählte Algorithmus über Hyperparameter verfügt, gibt es wie in Abbildung 9 dargestellt eine Rückkopplung zum Training des Modells, in der diese optimiert werden. Für den Fall, dass die Ergebnisse danach immer noch als unzureichend empfunden werden, können die vorangegangenen Prozessschritte angepasst werden, indem beispielweise andere Daten, andere Methoden zur Datenvorverarbeitung oder andere Algorithmen für die Klassifikation verwendet werden. Als Resultat liegt der Klassifikator mit der höchsten Genauigkeit vor, der in der produktiven Anwendung eingesetzt werden kann. Für weitere Details wird der interessierte Leser auf Suthaharan (2016) verwiesen.

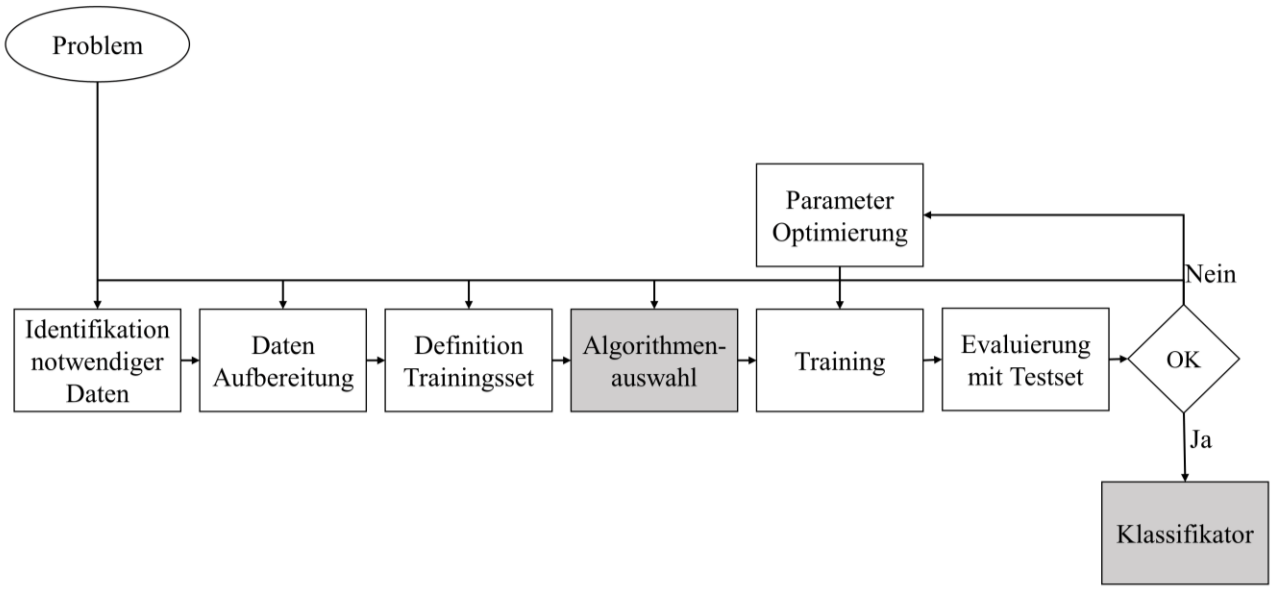


Abbildung 9: Ablauf, um einen überwachten Klassifikator zu trainieren (in Anlehnung an (Kotsiantis et al. 2007))

Überwachte Lernverfahren können in unterschiedliche Kategorien eingeteilt werden. Neben dem klassischen *Machine Learning* haben sich das *Deep Learning* und das *selbstüberwachte Lernen* als erfolgreiche Konzepte etabliert. Im Folgenden wird auf diese drei Lernverfahren eingegangen und entsprechende Literatur zum Stand der Technik zusammengefasst.

Das *Machine Learning* ist Teil der künstlichen Intelligenz und beschreibt im Allgemeinen Algorithmen, die auf Basis von vorhandenen Daten das Lösen einer bestimmten Aufgabe erlernen. Das Deep Learning ist ein Teilbereich des *Machine Learning* und beinhaltet Verfahren, die auf sogenannten tiefen neuronalen Netzen basieren. Wird das Deep Learning ausgeklammert, ist ein Vorteil des *Machine Learning* der, dass die Ansätze oft ressourcenschonender hinsichtlich der benötigten Anzahl an Trainingsdaten sind. Bereits aus einer geringen Anzahl an Daten können so hohe Genauigkeiten der Prädiktionen erzielt werden.

Ein bekanntes Datenset, um Klassifikationsalgorithmen miteinander zu vergleichen ist *MNIST* (Deng 2012). Es besteht aus 70000 schwarz-weiß Bildern handgeschriebener Ziffern zwischen 0 und 10 mit einer Auflösung von jeweils 28x28 Pixeln. LeCun et al. (2022) fassen die Ergebnisse einer Vielzahl von ML-Algorithmen für *MNIST* zusammen. Da es sich bei *MNIST* durch die kleine Auflösung der Bilder und geringe Komplexität der darin enthaltenen Informationen um ein relativ einfaches Datenset handelt, erzielen auch klassische *Machine Learning* Verfahren wie die sogenannten *k-nächste Nachbarn* (*KNN*) (5% Fehlerrate (LeCun et al. 1998)) oder *Support Vector Maschine* (1,1% Fehlerrate (LeCun et al. 1998)) sehr gute Klassifikationsergebnisse.

Ein Nachteil von klassischen ML-Verfahren ist jedoch, dass die Genauigkeit dieser Modelle mit steigender Komplexität der Daten (hochauflösende komplexe Bilder, Videos, Sprache) stark abnimmt.

In den vergangenen Jahren liefern insbesondere in der Bild- und Sprachverarbeitung Deep Learning Ansätze bessere Ergebnisse im Vergleich zu klassischen ML-Algorithmen. Das Deep Learning basiert auf tiefen neuronalen Netzen, die dazu in der Lage sind, komplexe Merkmale

aus den Daten zu extrahieren, ohne, dass diese im Detail vom Anwender vorab aufwändig aufbereitet werden müssen. Das Prinzip neuronaler Netze ist durch die Funktionsweise des menschlichen Gehirns motiviert und exemplarisch in Abbildung 10 dargestellt.

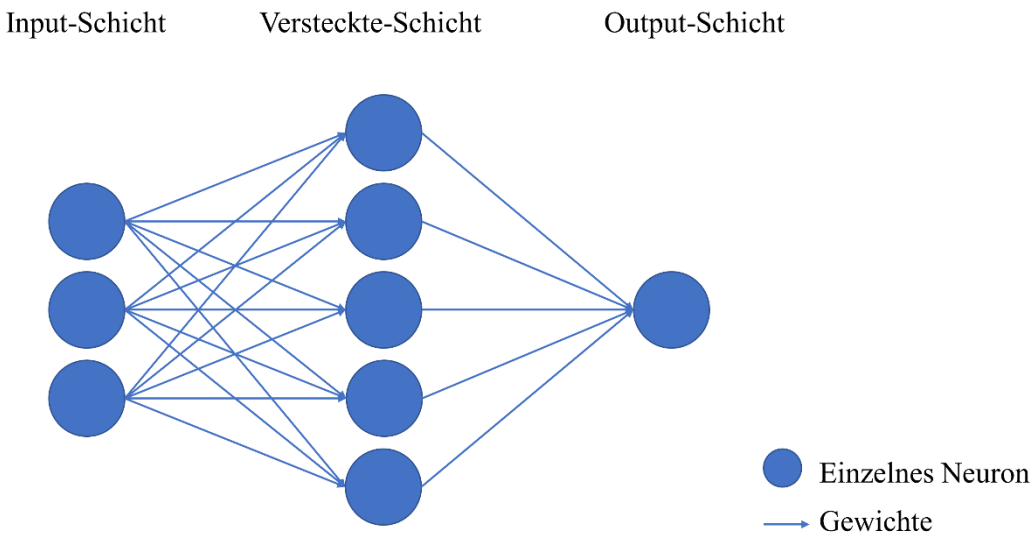


Abbildung 10: Exemplarischer Aufbau eines neuronalen Netzes bestehend aus einzelnen Neuronen, die durch Gewichte über verschiedene Schichten hinweg miteinander verbunden sind

Einzelne Neuronen (blaue Punkte) sind über verschiedene Schichten des Netzes hinweg miteinander verknüpft. Eine Input-Schicht verarbeitet die Eingangsdaten, extrahiert Informationen und leitet diese über die versteckte Schicht bis an die Output-Schicht weiter. An dieser findet dann die Prädiktion der zugehörigen Kategorie statt. Aktivierungsfunktionen werden an jedem Neuron verwendet, um Informationen unterschiedlich zu gewichten und an dahinterliegende Schichten zu übertragen.

Je nach Anwendungsfall eignen sich unterschiedliche Arten neuronaler Netze. Für die Verarbeitung von Daten, die in Form von Sequenzen (z.B. Zeitreihen) vorliegen, werden sogenannte Long-Short-Term-Memory (LSTM) Architekturen verwendet. In Graves und Schmidhuber (2005) werden beispielsweise LSTM eingesetzt, um Laute im Kontext der Spracherkennung zu klassifizieren.

Für die Klassifikation von Bildern werden CNN verwendet. Durch den Faltungsoperator, der auf verschiedenen Bereichen eines Bildes angewendet wird, ist das neuronale Netz dazu in der Lage komplexe Eigenschaften aus den Daten zu extrahieren. Unterschiedliche Faltungsoperatoren extrahieren dabei unterschiedliche Merkmale. Die ersten Schichten eines CNN lernen Eigenschaften wie z.B. den Verlauf von Kanten zu erkennen, die oft essenziell für die Prädiktion der zugehörigen Kategorie sind. In Krizhevsky et al. (2012) werden CNN verwendet, um Bilder aus dem „*ImageNet LSVRC-2010*“ Datenset zu klassifizieren. Dafür stehen 1,2 Millionen hoch aufgelöste Bilder aus 1000 verschiedenen Kategorien zur Verfügung, wobei durch das vorgestellte neuronale Netz Top-1 und Top-5 Error-Raten von 37,5% beziehungsweise 17,0% erzielt werden.

Während Bilder zweidimensionale Daten sind, liegen im Fall von Crashsimulationen jedoch räumliche 3D-Daten vor. In Su et al. (2015) wird ein Klassifikator trainiert, der die Form von 3D-Objekten klassifizieren kann. Hierfür wird ebenfalls auf CNN zurückgegriffen und die dreidimensionalen Eingangsdaten werden durch eine Verkettung mehrerer 2D-Bilder aus unterschiedlichen Perspektiven auf das zu klassifizierende Objekt dargestellt. Es wird aufgezeigt, dass dieser Ansatz im Vergleich zu Verfahren, die direkt 3D-Voxel-Daten oder Polygon-Netze verarbeiten, bessere Ergebnisse mit einem ressourcenschonenderen Modell erzielt.

Im Kontext von Crashsimulationen gibt es einen weiteren Ansatz, der zunächst ebenfalls die dreidimensionalen Eingangsdaten auf zwei Dimensionen reduziert, um diese im Anschluss mittels CNN zu klassifizieren (Spruegel et al. 2017; Spruegel et al. 2018; Bickel et al. 2019; Spruegel et al. 2021; Spruegel und Wartzack 2015). Hierbei werden jedoch nicht mehrere Bilder aus unterschiedlichen Perspektiven zur Verfügung gestellt, sondern die Bilder werden mittels des in Kapitel 2.3 vorgestellten Projektionsverfahrens auf eine Kugeloberfläche projiziert, sodass daraus ein zweidimensionales Bild resultiert, das von klassischen CNN weiterverarbeitet werden kann. Das Ziel dabei ist es nicht ein vorgegebenes Crashverhalten zu detektieren, sondern zum einen Bauteile zu erkennen sowie einen Plausibilitätscheck durchzuführen, um dadurch zu prüfen, ob eine Simulation in Ordnung oder nicht in Ordnung ist. Dabei äußern sich die Autoren allerdings nicht dazu, ob und wie das zeitliche Verhalten von Crashsimulationen berücksichtigt werden kann. Das zeitliche Verhalten ist jedoch eine zentrale Eigenschaft für ein Crashverhalten und dessen Abbildung wird in der in dieser Arbeit vorgestellten Methode berücksichtigt.

Der Erfolg von Verfahren des Deep Learning basiert auf einer Vielzahl an Dateninstanzen, die für das Training zur Verfügung stehen müssen. Desto mehr Daten vorliegen, desto bessere Ergebnisse erzielen die Modelle in der Regel. Dieser Umstand stellt jedoch eine starke Limitierung der Anwendbarkeit dieser Algorithmen dar. Insbesondere in Anwendungen, in denen das Akquirieren von Daten und im Falle einer Klassifikationsaufgabe zusätzlich der zugehörigen Kategorien aufwändig ist, werden Ansätze benötigt, die aus wenigen Daten robuste Prädiktionen erlernen. Insbesondere in der Anwendung für Crashsimulationen ist ein dateneffizienter Ansatz entscheidend für die Akzeptanz und den Erfolg des in dieser Arbeit vorgestellten Crash-Verhalten-Detektors, da die Zuweisung der Kategorien zu den Dateninstanzen aufwändig ist und zusätzliche Kapazität von dem/der Ingenieur*in abverlangt. Darüber hinaus liegen in der frühen Entwicklungsphase eines Fahrzeug-Projekts zunächst nur wenige Simulationen vor, die überhaupt für das Trainieren eines Modells verwendet werden können.

In der Literatur gibt es verschiedene Ansätze, um robuste Modelle aus wenigen aber komplexen kategorisierten Daten zu trainieren. Ein Ansatz ist das sogenannte *selbstüberwachte Lernen*. Dabei handelt es sich um ein zweistufiges Verfahren. Die Idee dahinter ist, im ersten Schritt eine Vielzahl an unkategorisierten Daten zu nutzen, um die darin enthaltenen Informationen zu extrahieren und in einer kompakten Repräsentation abzubilden. Wie schon der Name andeutet, soll dies in Form des überwachten Lernens erfolgen, jedoch ohne, dass der Anwender explizit kategorisierte Daten zur Verfügung stellen muss (Details hierzu folgen in Kapitel 2.5.2). Die erhaltene kompakte Repräsentation wird anschließend im zweiten Schritt verwendet, um einen Klassifikator mit möglichst wenigen kategorisierten Daten zu trainieren. Durch dieses Vorgehen wird der Aufwand des Anwenders zum Kategorisieren der Daten minimiert. Zu den Verfahren, die dem *selbstüberwachten Lernen* zuzuordnen sind, zählen beispielsweise die sogenannten

Siamese Networks (Bromley et al. 1993), *BYOL* (Grill et al. 2020) sowie *Simple Siamese Networks* (*SimSiam)* (Chen und He 2021).

Unabhängig davon, ob es sich um ein Verfahren des *Machine Learning*, *Deep Learning* oder *selbstüberwachten Lernens* handelt, kann nicht a priori festgelegt werden, welcher Algorithmus die höchste Prädiktionsgenauigkeit liefet. Nach Kotsiantis et al. (2007) sollten stets verschiedene Ansätze für die zugrunde liegende Aufgabenstellung verglichen werden. Daher werden in der vorliegenden Arbeit bekannte Vertreter des *Machine Learning* mit den *Simple Siamese Networks* verglichen. Im Folgenden werden die Grundlagen zu den verwendeten Algorithmen dargestellt.

2.5.2 Theorie der verwendeten Algorithmen

2.5.2.1 Support Vector Maschine

Die Idee der *Support Vector Maschine* geht auf Vapnik und Chervonenkis (1974) zurück. Dabei kann zwischen der sogenannten *linearen* und *Kernel Support Vector Maschine* unterschieden werden. Bei Aufgabenstellungen mit nicht linear separierbaren Daten sollte letztere verwendet werden. Im Folgenden wird die Funktionsweise der *linearen Support Vector Maschine* in Anlehnung an Raschka und Mirjalili (2019) beschrieben.

Der Algorithmus reduziert die Anzahl an Fehlklassifikationen, indem die Breite um die Entscheidungsgrenze maximiert wird, sodass ein möglichst großer Bereich entsteht, in dem keine Datenobjekte liegen. Daher werden *Support Vector Maschinen* auch als sogenannte *Large-Margin-Klassifikatoren* bezeichnet. Das Ziel dabei ist, ein möglichst verallgemeinerndes Modell zu erhalten. Im Fall von zu schmalen Bändern besteht das Risiko eines überangepassten Modells, das für die Trainingsdaten zwar gut funktioniert, bei der Klassifikation neuer Objekte allerdings scheitert. Die Entscheidungsgrenze kann im eindimensionalen Fall eine Linie oder im mehrdimensionalen Fall eine Hyperebene sein. Für die Berechnung der Hyperebene müssen jedoch nicht alle Datenobjekte betrachtet werden, sondern lediglich die, die ihr am nächsten liegen. Die Vektoren dieser Objekte zur Hyperebene werden daher als Stützvektoren bezeichnet. Dies macht den Algorithmus speichereffizient da lediglich ein Teil der Trainingsdatenmenge betrachtet werden muss.

Um nichtlineare Daten besser trennen zu können, wird die Schlupfvariable *C* eingeführt, mit der Fehlklassifikationen im Trainingsprozess gewichtet werden können. Große Werte von *C* erhöhen die Gewichtung von Fehlklassifikationen und bergen damit das Risiko einer Überanpassung des Modells an die Daten. Der Randbereich um die Kategorie Grenze kann somit sehr schmal werden. Dagegen verringern kleine C die Gewichtung von Fehlklassifikationen und sorgen für eine bessere Verallgemeinerungsfähigkeit des Modells. Hier besteht das Risiko einer Unteranpassung sowie des Eintretens von Fehlklassifikationen.

Für die *Support Vector Maschine* ist es wichtig, dass die Daten einen ähnlichen Wertebereich aufweisen, da ansonsten Merkmale mit großer Amplitude andere dominieren. Daher sollten sie vorab entsprechend skaliert werden (beispielsweise standardisiert).

2.5.2.2 Entscheidungbaum

Die Funktionsweise des *Entscheidungsbaum* Algorithmus wird in Anlehnung an Raschka und Mirjalili (2019) beschrieben. Entscheidungsbäume teilen die Daten durch Abfragen bestimmter Wertebereiche der einzelnen Merkmale in entsprechenden Kategorien ein. Dadurch entsteht ein Regelwerk, das verwendet werden kann, um neue Datenpunkte zu klassifizieren. Abbildung 11 stellt beispielhaft eine binäre Klassifikation von zweidimensionalen Daten dar, um das Prinzip des Verfahrens zu visualisieren. In dem sogenannten Wurzelknoten sind sämtliche Instanzen aller Kategorien enthalten – vier Instanzen von Kategorie *A* und sechs von *B*. Im nächsten Schritt wird ein Merkmal (hier x_1) sowie ein Schwellwert (hier exemplarisch 2) bestimmt. Der Schwellwert wird verwendet, um die Daten in zwei Intervalle aufzuteilen, in denen jeweils neue Verteilungen der zugehörigen Kategorien vorliegen. Dieses Vorgehen wird wiederholt, sodass bei jeder Iteration neue Kindknoten entstehen, die unterschiedliche Merkmale und Schwellwerte in den übrigen Daten abfragen. Sobald an einem Kindknoten nur noch Datenobjekte einer einzigen Kategorie vorliegen, kann keine weitere Aufteilung erfolgen. Die letzten Knoten werden daher auch als Blattknoten bezeichnet.

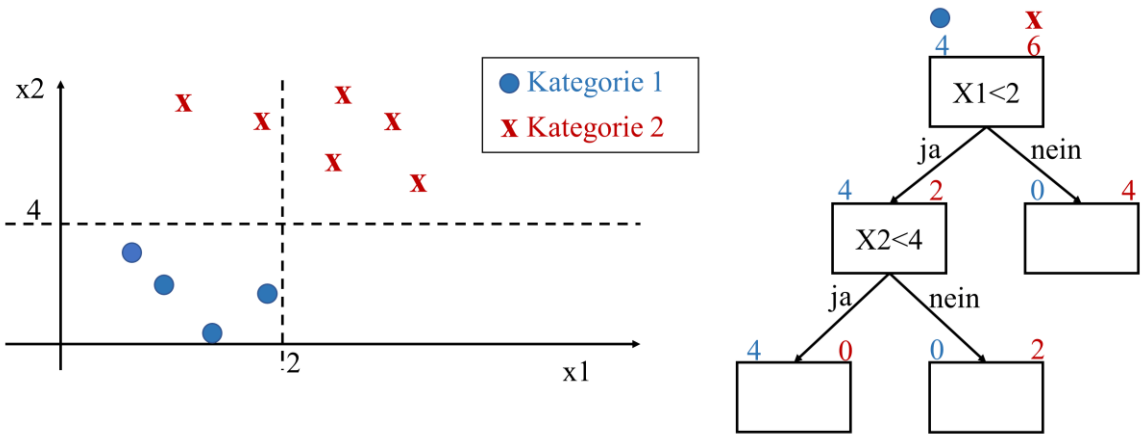


Abbildung 11: Links: Beispiel Datenset bestehend aus 2 Kategorien und den Merkmalen x_1 und x_2; Rechts: Schematischer Ablauf des Entscheidungsbaum Algorithmus

Die Aufteilung an jedem einzelnen Knoten wird gemäß des sogenannten Informationsgewinns vorgenommen. Desto besser sich die Kategorien nach einem Knoten aufteilen lassen, desto größer ist der sogenannte Informationsgewinn an diesem Knoten durch eine gute Aufteilung des Wertebereichs. Bei komplexen Daten kann eine Vielzahl an Aufteilungen der Wertebereiche einzelner Merkmale erfolgen. Dadurch entstehen komplexe Bäume, die für die Trainingsdaten gute Ergebnisse erzielen, jedoch aufgrund von Überanpassung bei neuen Daten scheitern. Aus diesem Grund gibt es die Möglichkeit die Tiefe des Entscheidungsbaums vorab zu definieren. Das Festlegen der maximalen Tiefe erfolgt durch die Definition eines Hyperparameters. Eine weitere Möglichkeit ist das sogenannte *Pruning*, bei dem für die Vorhersagegenauigkeit unwichtige Knoten aus dem Baum entfernt werden. Dadurch wird ebenfalls die Komplexität des Entscheidungsbaums und damit die Gefahr einer Überanpassung an die Trainingsdaten reduziert.

Ein Vorteil von Entscheidungsbäumen ist, dass die vom Modell prädizierten Ergebnisse nachvollzogen werden können, indem das entstandene Regelwerk analysiert wird. Für Details zum *Entscheidungsbaum* wird auf Breimann et al. (1984) verwiesen.

2.5.2.3 Random Forest

Werden mehrere Entscheidungsbäume zu einem Modell kombiniert, spricht man von einem *Random Forest*. Damit gehört er zur Kategorie der Ensemble Modelle. Dessen Funktionsweise wird im Folgenden in Anlehnung an Raschka und Mirjalili (2019) beschrieben. Die Idee dahinter ist, dass eine Vielzahl an Entscheidungsbäumen die Prädiktionsgenauigkeit eines einzelnen Modells übertreffen und zu robusteren Ergebnissen führen. Hierbei sind die einzelnen Entscheidungsbäume nicht identisch, sondern unterscheiden sich darin, mit welchem Subset der Trainingsdaten und der Merkmale sie erzeugt sind. Die Klassifizierung neuer Daten erfolgt, indem die Vorhersagen der einzelnen Bäume in eine Mehrheitsentscheidung einfließen.

Als Hyperparameter für einen *Random Forest* müssen die Anzahl der verwendeten Entscheidungsbäume sowie die maximale Anzahl zu verwendender Merkmale definiert werden. Darüber hinaus kann die Anzahl der Objekte für das Training bestimmt werden. Wird eine geringe Anzahl gewählt, unterscheiden sich die einzelnen Bäume stark voneinander. Dies kann die Wahrscheinlichkeit einer Überanpassung des *Random Forest* verringern. Da so insgesamt jedoch weniger Information aus der gesamten Trainingsdatenmenge verwendet wird, kann dies mit einer verringerten Klassifizierungsgenauigkeit einhergehen. Andersherum verhält es sich mit einer großen Anzahl zufällig gewählter Objekte aus den Trainingsdaten. Dadurch fällt der Unterschied der einzelnen Entscheidungsbäume zwar geringer aus, was einerseits die Genauigkeit des Modells für die Trainingsdaten erhöht, andererseits jedoch das Risiko der Überanpassung birgt. Details zum *Random Forest* können Breimann et al. (1984) entnommen werden.

2.5.2.4 Gauß-Prozess

Bei dem *Gauß-Prozess* handelt es sich um ein Verfahren, das sich sowohl für Regressions- als auch Klassifikationsaufgaben eignet. Dessen Funktionsweise wird im Folgenden in Anlehnung an (Seeger 2004) beschrieben. Während die Gauß-Wahrscheinlichkeitsfunktion die Verteilung von Zufallsvariablen beschreibt, modelliert der *Gauß-Prozess* die Eigenschaften von Funktionen und kann damit als eine Abstraktion der Gauß-Verteilungsfunktion betrachtet werden. Dabei wird von der Annahme ausgegangen, dass nah beieinander liegende Punkte ähnliche Funktionswerte aufweisen, bei denen es sich im Fall der Regression um kontinuierliche Werte handelt und bei Klassifikationsaufgaben um diskrete Werte der einzelnen Kategorien. Um dies abzubilden, ist die Definition eines sogenannten Kernels notwendig, der die Aufgabe der Kovarianzfunktion übernimmt. Dieser legt fest, wie die einzelnen Dateninstanzen in Relation zueinanderstehen. Die Wahl des Kernels ist dem Anwender überlassen und muss in Form eines Hyperparameters festgelegt werden. Ein weit verbreiteter Kernel ist die radiale Basisfunktion

$$\boldsymbol{rbf}(\boldsymbol{p},\boldsymbol{o}) = \mathbf{exp}\left(\frac{\mathbf{distanz}(\boldsymbol{p},\boldsymbol{o})^{\mathbf{2}}}{\mathbf{2}\boldsymbol{l}^{\mathbf{2}}}\right) \quad . \tag{2.14}$$

Dabei entspricht *l* einem Parameter, mit dem die Ähnlichkeit zwischen den beiden Punkten *p* und *o* skaliert wird. Die Parameter des Kernels werden während dem Modelltraining optimiert. Der *Gauß-Prozess* modelliert die Funktionswerte in Abhängigkeit der Daten $\boldsymbol{X}$. Um Klassifikationsaufgaben zu lösen, wird die logistische Funktion als Fehlerfunktion verwendet, um die modellinterne Datenrepräsentation in Wahrscheinlichkeiten zur Kategorie Zugehörigkeit zwischen 0 und 1 zu konvertieren.

Ein Vorteil des *Gauß-Prozess* ist, dass neben den Prädiktionen gleichzeitig Konfidenzen angegeben werden, sodass der Anwender eine Einschätzung darüber erhält, wie sicher die Prädiktionen sind. Damit erhält er gezielt Hinweise, welche weiteren Trainingsdaten benötigt werden, um die Konfidenzen zu erhöhen. Detailliertere Informationen zum *Gauß-Prozess* können Seeger (2004) entnommen werden.

2.5.2.5 Logistische Regression

Die Funktionsweise der *Logistischen Regression* wird am Beispiel der binären Klassifikation in Anlehnung an Raschka und Mirjalili (2019) erläutert. Die vorliegenden Daten können in zwei Kategorien unterteilt und anhand des Merkmals x_1 beschrieben werden. Dieses Verfahren lässt sich auch auf mehrdimensionale Daten übertragen (Komarek 2004). Der Wert des Merkmals ist entscheidend dabei, zu welcher Kategorie eine Dateninstanz gehört. Abbildung 12 veranschaulicht den Zusammenhang. Um die Kategoriezugehörigkeit in Abhängigkeit des beschreibenden Merkmals x_1 abzubilden, wird die logistische Funktion

$$\boldsymbol{f(x_1) = \frac{1}{1 + \exp(-x_1)}} \qquad \textbf{(2.15)}$$

verwendet.

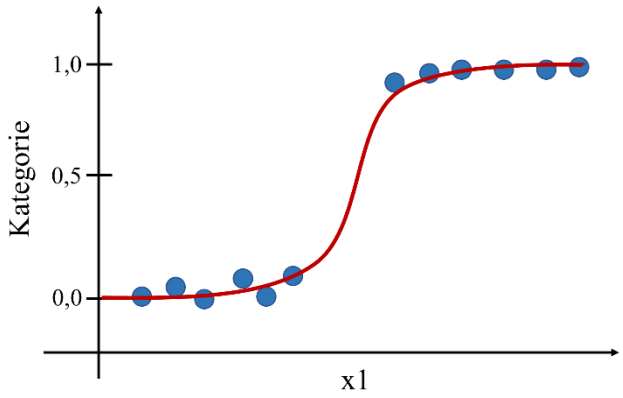


Abbildung 12: Exemplarischer Verlauf der Logistischen Regression für die binäre Klassifikation der Daten in Abhängigkeit des Merkmals x_1

Bis hierhin liegt, wie auch bereits der Name suggeriert, lediglich ein Regressionsverfahren vor. Aus diesem wird erst dann ein Klassifikator, sobald ein Schwellwert definiert wird, der entscheidet, ab welcher Wahrscheinlichkeit die entsprechende Kategorie vorliegt. Ein Vorteil der *Logistischen Regression* ist die Interpretierbarkeit der Ergebnisse, da neben der prädizierten

Kategorie gleichzeitig die Wahrscheinlichkeit angegeben wird, mit der ein Datenpunkt dieser angehört.

2.5.2.6 Simple Siamese Network SimSiam

Die bisher beschriebenen Algorithmen sind dem klassischen *Machine Learning* zuzuordnen. Darüber hinaus wird in dieser Arbeit ein *Deep Learning* Ansatz aus dem Gebiet des *selbstüberwachten Lernens* betrachtet. Dabei wird das Ziel verfolgt, aus möglichst wenig kategorisierten Daten eine möglichst hohe Prädiktionsgenauigkeit zu erreichen, um den Aufwand des Kategorisierens der Daten zu reduzieren. Daher wird im Folgenden die Theorie von *SimSiam* exemplarisch am Beispiel der Bildklassifikation in Anlehnung an Chen und He (2021) erläutert. *SimSiam* steht dabei für *Simple Siamese Networks* und deutet damit die Verwandtschaft des Algorithmus zu den sogenannten *Siamese Networks* (Bromley et al. 1993) an. Das Ziel ist es durch das Lernverfahren zunächst aus unkategorisierten Daten relevante Merkmale zu extrahieren, die in einem zweiten Schritt dazu verwendet werden, einen überwachten Klassifikator mittels weniger kategorisierter Daten zu trainieren. Im Folgenden wird beschrieben, wie der erste Schritt mittels *SimSiam* umgesetzt ist.

Da hierfür keine kategorisierten Daten zur Verfügung stehen, wird die Aufgabenstellung angepasst und es werden künstliche, überwachte Lernaufgaben geschaffen. Der prinzipielle Aufbau eines solchen Netzwerks ist in Abbildung 13 dargestellt.

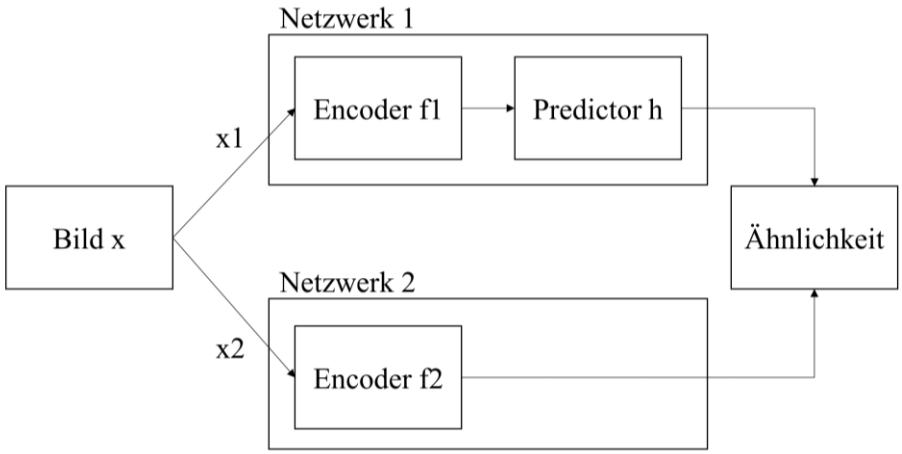


Abbildung 13: Allgemeiner Aufbau von SimSiam. Aus dem Eingangsbild x werden zwei Modifikationen (x_1 und x_2) an ein Encoder Netzwerk weitergegeben. Das Predictor Netzwerk im oberen Strang prädiziert den Output des Encoders f aus dem unteren Strang. Während dem Training wird dabei die Ähnlichkeit zwischen den beiden Vektoren maximiert.

Zunächst werden zwei Modifikationen x_1 und x_2 des originalen Bildes x erzeugt. Dies erfolgt, indem beispielsweise zur Erlangung der ersten Variante x_1 ein Gaußsches Rauschen zu den Farbwerten der einzelnen Pixel des originalen Bildes x hinzugefügt wird und als zweite Variante x_2 das Bild x verzerrt und rotiert wird. Diese Modifikationen werden genutzt, um die überwachte Aufgabenstellung umzusetzen.

x_1 ist der Input für den Encoder *f1* und x_2 der Input für den Encoder *f2*. Diese Encoder Netzwerke bestehen aus mehreren Faltungsschichten (CNN), die mit MaxPooling und Dropout Schichten

kombiniert werden. Deren Aufbau hängt von den Eigenschaften der zugrunde liegenden Daten ab. Durch die Encoder werden mittels des Faltungsoperators Merkmale wie beispielsweise Kantenverläufe in den Bildern detektiert. Das Resultat ist dabei ein eindimensionaler Vektor. Die Variante x_1 wird in dem „Predictor“ *h* weiterverarbeitet, wobei die Dimension des Vektors erhalten bleibt. Dieser ist ein einfaches neuronales Netz, das aus verschiedenen dichten Schichten besteht. Wie der Name schon andeutet, wird dieser Teil des Netzwerks dazu verwendet, den resultierenden Merkmalsvektor von x_2 durch den von x_1 zu prädizieren.

Der Unterschied zwischen den beiden Vektoren wird durch eine „Loss-Funktion“ berechnet. Hierfür kann beispielsweise der mittlere quadratische Fehler verwendet werden. Nachdem x_1 durch das *Netzwerk 1* und x_2 durch das *Netzwerk 2* verarbeitet wurden, wird im Anschluss daran der Prozess umgekehrt und x_1 ist der Input für *Netzwerk 2* sowie x_2 für „Netzwerk1“. Die dadurch resultierenden beiden Werte der „Loss-Funktion“ werden durch Bildung des Durchschnitts gemittelt.

Diese Vorgehensweise wird für sämtliche Instanzen der Trainingsdaten wiederholt. Dabei wird der Betrag der „Loss-Funktion“ während des Trainings minimiert, indem die Gewichte der einzelnen Neuronen von *Netzwerk 1* angepasst werden. *Netzwerk 2* wird nicht angepasst, sondern übernimmt unverändert die Gewichte aus *Netzwerk 1*. Als Resultat wird eine kompakte Repräsentation des originalen Bildes erlangt. Diese wird im Anschluss verwendet, um einen Klassifikator mit den kategorisierten Daten zu trainieren.

Die Idee hinter dieser Vorgehensweise besteht darin, dass leichte sowie zufällige Modifikationen des originalen Bildes immer noch dieselben Informationen über die zugehörige Kategorie enthalten. Indem also die Modifikation verwendet wird, um die andere zu prädizieren, werden die intrinsischen Informationen des originalen Bildes gelernt und in einer kompakten Darstellung abgebildet. Durch dieses Erzeugen unterschiedlicher Varianten aus nur einer einzigen Dateninstanz, können relevante Merkmale auch aus kleinen Datensets effizient extrahiert werden und es ist nicht wie in bisherigen Ansätzen das Vorhandensein tausender Trainingsdaten notwendig. Welche Modifikationen jeweils auf das originale Bild angewendet werden sollten, hängt von der zugrunde liegenden Aufgabenstellung ab und muss individuell festgelegt werden.

2.6 Dimensionsreduktion

2.6.1 Grundlagen

Die Dimensionsreduktion ist eine Methode, die bereits seit vielen Jahren in den unterschiedlichsten Anwendungsgebieten verwendet wird. Die Idee dahinter ist, die komplexen Eingangsdaten $\boldsymbol{X}$ mit N Datenpunkten der Dimension d in einen niedrigdimensionalen Raum der Dimension d_{red} zu überführen ($d_{red} \ll d$). Dabei werden basierend auf $\boldsymbol{X}$ und den originalen Dimensionen d neue Merkmale berechnet. Oft ist die intrinsische Dimension der vorliegenden Daten bedeutend kleiner, sodass diese mit einer geringeren Anzahl an Merkmalen beschrieben werden können.

Die Dimensionsreduktion findet bei verschiedensten Fragestellungen Anwendung, die im Folgenden näher erläutert werden. In Sible et al. (2020b), Mertler et al. (2015) sowie Steffes-lai und Clees (2011) wird sie beispielsweise dazu verwendet, die Ergebnisdateien aus der Simulation zu komprimieren, sodass geringere Dateigrößen vorliegen und damit der benötigte Festplattenspeicher reduziert werden kann. Aufgrund ihrer Komplexität liegen gerade bei Crashsimulationen große Ergebnisdateien (in Bezug auf ihre Dateigröße) vor, sodass die Datenkompression ein wichtiger Schritt im automatisierten Post-Prozessing der Ergebnisse ist.

Des Weiteren kann die Dimensionsreduktion als Schritt der Datenvorverarbeitung verwendet werden, um zum einen den Berechnungsaufwand nachfolgender Verfahren zu reduzieren, als auch die Qualität deren Ergebnisse zu verbessern (van der Maaten et al. 2009; Iza-Teran und Garcke 2014; Iza-Teran 2014). Gerade bei vielen Dimensionen muss der sogenannte Fluch der Dimensionalität berücksichtigt werden (Bellman 1961). Danach verlieren euklidische Distanzen zwischen Datenpunkten mit steigender Dimension an Bedeutung. Bei Verfahren (Clustering, Klassifikation, etc.), die auf der Berechnung von euklidischen Distanzen basieren, kann es demnach vorkommen, dass relevante Eigenschaften in den Daten nicht hinreichend abgebildet und dadurch unzureichende Ergebnisse erzielt werden. Die Dimensionsreduktion verbessert diese Resultate, indem Rauschen und andere für die Problemstellung unwichtige Eigenschaften der Daten vorab entfernt werden.

Eine weitere Anwendung stellt die Datenanalyse dar. In (Thole et al. 2010; Brown et al. 2012) werden die PCA und die davon abgeleitete sogenannte Difference PCA (DPCA) dazu verwendet, Kausalitäten zwischen Modelländerungen und dem daraus resultierenden Crashverhalten herzustellen. Eine weitere Möglichkeit, diese Kausalitäten herzuleiten, stellt eine geeignete Visualisierung einer Vielzahl von Simulationen und die Verknüpfung mit den Inputparametern dar. Im Allgemeinen ist die Visualisierung des Crashverhaltens für den interaktiven Vergleich einer Schar von Simulationen für den/die Ingenieur*in deshalb hilfreich, um eine Übersicht darüber zu erlangen, wie sich das Crashverhalten in der Vergangenheit in den einzelnen Varianten entwickelt hat. Dieser konkrete Anwendungsfall wird in der vorliegenden Arbeit betrachtet.

Interaktive Visualisierungen in dreidimensionalen Streu- und Liniendiagrammen eignen sich besonders für die Analyse der Ergebnisse. Eine Alternative stellen sogenannte *Parallel Coordinates Plots* dar (Inselberg und Dimsdale 1990). Diese können zwar im Vergleich zu Streu- und Liniendiagrammen mehr als drei Dimensionen darstellen, wodurch bei der Dimensionsreduktion ein geringerer Informationsverlust entsteht. Sie werden jedoch schnell unübersichtlich, wenn eine Vielzahl an Simulationen betrachtet werden soll. Damit die

Darstellung für den Anwender so einfach und intuitiv wie möglich ist, werden als Visualisierungsformat daher dreidimensionale Streu- und Liniendiagramme verwendet, welche die Zieldimension der Reduktionsverfahren auf drei festlegt.

Darüber hinaus muss beachtet werden, dass es deterministische und stochastische Verfahren zur Dimensionsreduktion gibt. Bei Letzteren ist besonders zu beachten, wie stark die Streuung der Ergebnisgüte aufgrund der stochastischen Eigenschaften ist. Sie sollte möglichst gering sein, da die Dimensionsreduktion sonst mehrere Male wiederholt werden muss, was wiederum den Aufwand und Zeitbedarf für den Anwender erhöht und das interaktive Arbeiten mit den Daten somit eingeschränkt.

Damit die Bedienung der Methode so einfach wie möglich ist, sollten darüber hinaus möglichst wenige Hyperparameter definiert werden müssen und deren Auswirkungen auf die Ergebnisse nachvollziehbar sein, sodass vom Anwender schnell geeignete Parameterwerte gefunden werden können.

Die Algorithmen zur Dimensionsreduktion können des Weiteren in lineare und nichtlineare Verfahren unterteilt werden. Für Details zu letzteren wird auf Lee und Verleysen (2007) verwiesen. Gerade bei hochgradig nicht linearen Anwendungen wie in Crashsimulationen, können nichtlineare Verfahren Vorteile gegenüber linearen Verfahren liefern (Mertler et al. 2015; Bohn et al. 2013).

In der Literatur werden unterschiedliche Verfahren zur Dimensionsreduktion und Visualisierung von Crashverhalten betrachtet. Im Folgenden wird ein Überblick über den Stand der Technik gegeben.

2.6.2 Stand der Technik

In Diez et al. (2016) wird eine geometriebasierte Dimensionsreduktion vorgestellt. Ausgehend von der Berechnung einer Ähnlichkeitsmatrix, wird diese im Anschluss dazu verwendet, mittels des *Multidimensional Scaling* (Borg und Groenen 2005) die Daten im niedrigdimensionalen Raum zu visualisieren.

In Iza-Teran (2014), Bohn et al. (2013), Iza-Teran und Garcke (2014), Garcke und Iza-Teran (2015) sowie Garcke et al. (2016) werden *Diffusion Maps* verwendet, um die Eigenschaften einer Vielzahl von Crashsimulationen durch neue Merkmalsvektoren zu beschreiben und entsprechend zu visualisieren. Das Ziel dabei ist, aus der gewonnenen Darstellung neue Erkenntnisse über das Crashverhalten zu erlangen.

Ein weiteres Verfahren stellen sogenannte *Sparse Grids* dar. Die Autoren von Bohn et al. (2016) sowie Feuersänger und Griebel (2009) vergleichen die Ergebnisse der PCA mit dem vorgestellten *Sparse Grids* basierten Ansatz. Da es sich hierbei um generative Methoden handelt, das bedeutet, dass aus den niedrigdimensionalen Daten eine Transformation in den originalen Raum durchgeführt werden kann, wird als Fehlermaß für die Dimensionsreduktion der Rekonstruktionsfehler betrachtet. Es wird dabei festgestellt, dass dieser bei den nichtlinearen *Sparse Grids* kleiner ist als bei der linearen PCA.

Darüber hinaus gibt es weitere Verfahren, die sich zur Dimensionsreduktion und demnach Visualisierung von unterschiedlichem Crashverhalten eigenen. In Garcke und Iza-Teran (2017), Iza-Teran und Garcke (2019), Sible et al. (2020b) und Sible (2020a) wird der *Laplace Beltrami*

Operator auf Basis des FE-Netzes konstruiert. Dabei wird durch die spektrale orthogonale Dekomposition die Eigenbasis geliefert, die anschließend dazu verwendet wird, Auswertungsgrößen aus der Simulation (z.B. Verschiebung in x,y,z) im dimensionsreduzierten Raum darzustellen. Die spektralen Koeffizienten stellen dabei die transformierten originalen Daten dar.

Mit fortschreitenden Erfolgen im Bereich des *Deep Learning* werden in jüngster Vergangenheit zunehmend *Deep Learning* basierte Verfahren für die Anwendung bei Crashsimulationen untersucht. Beispielsweise sind auch neuronale Netze in Form von Autoencodern dazu in der Lage, die relevanten Informationen der originalen Daten im latenten niedrigdimensionalen Raum darzustellen. In einer Voruntersuchung zu dieser Arbeit wird diese Fähigkeit bei „klassischen" und sogenannten *Variational Autoencodern* nachgewiesen.

In Abbasloo et al. (2019) werden sogenannte *Long Short-Term Memory* (LSTM) dazu verwendet, die relevanten Eigenschaften der Daten im latenten Raum zu repräsentieren. Für eine anschließende Visualisierung wird die Dimension des latenten Raums mittels des *t-distributed Stochastic Neighbor Embedding* (t-SNE) Algorithmus auf zwei Dimensionen reduziert.

Bei beiden Vorgehensweisen stellen sich neben starken stochastischen Einflüssen auch die hohe Anzahl an Hyperparametern sowie der hohe zeitliche Berechnungsaufwand für das Modell-Training als negativ dar, weshalb die Verwendung von neuronalen Netzen zur Dimensionsreduktion in der vorliegenden Arbeit ausgeschlossen wird.

In dieser Arbeit wird lineare PCA als Benchmark-Verfahren verwendet. Davon ausgehend wird untersucht, inwiefern nichtlineare Verfahren Vorteile bei der Visualisierung des Crashverhaltens bieten. Es wird insbesondere betrachtet, wie sich die Verwendung von geodätischen Abständen beim Isomap Algorithmus im Vergleich zu euklidischen Distanzen (PCA) auswirkt. Es gibt Verfahren mit dem speziellen Ziel der Visualisierung hochdimensionaler Daten. Daher werden zusätzlich t-SNE und UMAP darauf untersucht, inwiefern sie Vorteile bei der Visualisierung des Crashverhaltens liefern. Nachfolgend werden die theoretischen Grundlagen der verwendeten Algorithmen dargestellt.

2.6.3 Theorie der verwendeten Algorithmen

2.6.3.1 Principle Component Analysis

Die *Principle Component Analysis* (PCA) ist ein lineares Verfahren, um die Dimension der hochdimensionalen Eingangsdaten zu reduzieren. Sie wurde nach Lee und Verleysen (2007) von Pearson (1901) und Karhunen (1946) vorgestellt und hat sich über die Jahre in verschiedensten Anwendungsbereichen etabliert. In Kapitel 2.4.2.1 wird deren Funktionsweise erläutert.

Ein Vorteil der PCA ist, dass die in den projizierten Daten erhaltene Varianz und damit der gegenüber den originalen Daten verbliebene Informationsgehalt anhand der ersten d_{red} Eigenwerte berechnet werden kann. Diese Größe wird als erklärte Varianz bezeichnet. Wird sie in Relation zur gesamten in den originalen Daten $\boldsymbol{X}$ enthaltenen Varianz gesetzt, erhält man einen prozentualen Wert. Dieser kann als der noch in den reduzierten Daten enthaltene Informationsgehalt interpretiert werden. Je nach Anwendungsfall kann mit dessen Hilfe die Anzahl der benötigten Hauptkomponenten abgeschätzt werden. Ist das Ziel der Dimensionsreduktion eine Komprimierung der Daten, kann der Anwender beispielsweise

festlegen, dass mindesten 95% der gesamten Varianz erhalten bleiben soll und erhält damit die Information darüber, wie viele Hauptkomponenten zur Erreichung dieses Ziels benötigt werden. Ist die Anwendung dagegen eine Visualisierung, steht die Anzahl der Hauptkomponenten fest und beträgt üblicherweise 2 oder 3, da der Anwender somit die Daten anschaulich visualisieren kann. Ein weiterer Vorteil der PCA ist, dass der Algorithmus im Vergleich zu vielen nichtlinearen Verfahren keine Definition von Hyperparametern benötigt. Darüber hinaus sind die Ergebnisse deterministisch. Nachteilig wirkt sich dagegen die Linearität des Verfahrens aus. Bei nichtlinearen Daten können nichtlineare Algorithmen bessere Ergebnisse erzielen. Daher wird in dieser Arbeit auch der nichtlineare *Isometric Feature Mapping* (Isomap) Algorithmus untersucht.

2.6.3.2 Isometric Feature Mapping

Während lineare abstanderhaltende Verfahren wie die PCA auf euklidischen Distanzen basieren, werden beim *Isometric Feature Mapping* (Isomap) Algorithmus die geodätischen Distanzen zur Beschreibung der Ähnlichkeit zwischen den einzelnen Dateninstanzen verwendet. Im Folgenden werden die einzelnen Schritte dieser Berechnung in Anlehnung an Tenenbaum et al. (2000) dargestellt.

Zunächst wird für jeden einzelnen Datenpunkt im hochdimensionalen Raum dessen Nachbarschaft bestimmt. Der Anwender kann mit dem Hyperparameter *k* die Anzahl der zu berücksichtigenden benachbarten Datenpunkte festlegen. Zur Berechnung der *k* nächsten Nachbarn wird der euklidische Abstand eingesetzt. Dieser Vorgang wird für alle im Datenset vorhandenen Punkte wiederholt, woraufhin als Resultat ein mathematischer Graph vorliegt. In ihm ist die Information enthalten, welche Punkte jeweils miteinander verbunden sind (siehe Abbildung 14 Links). Im Folgenden wird er dazu verwendet, die geodätischen Distanzen zu approximieren. Ausgehend von dem vorliegenden Graphen wird nun für jeden einzelnen Datenpunkt der geodätische Abstand zu jedem anderen Punkt approximiert, indem jeweils dem kürzesten Pfad gefolgt wird. Als Ergebnis liegt eine Ähnlichkeitsmatrix basierend auf den geodätischen Distanzen vor (siehe Abbildung 14 Mitte). Mithilfe dieser wird im letzten Schritt die niedrigdimensionale Datenrepräsentation mittels des *Multidimensional Scaling* Verfahrens berechnet (siehe Abbildung 14 Rechts).

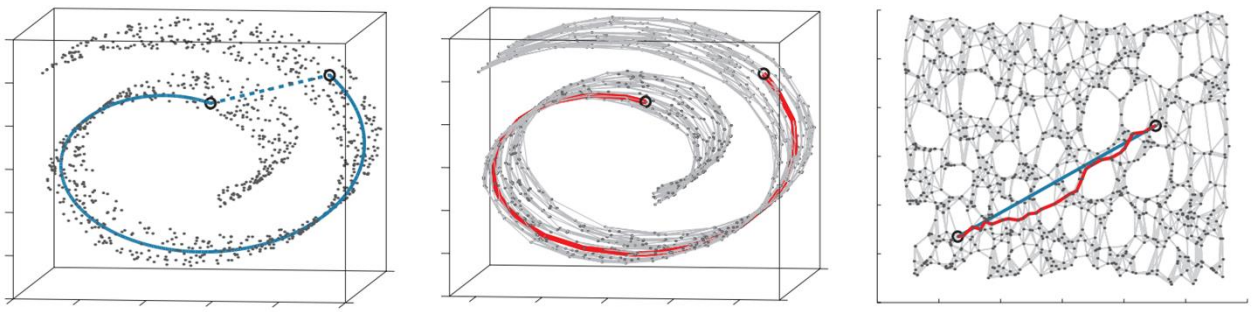


Abbildung 14: Funktionsweise des Isomap Algorithmus am Beispiel der „Swiss Role“ (Aus Tenenbaum et al. (2000)). Links sind exemplarisch Datenpunkte einer „Swiss Role“ dargestellt. Die blaue Linie stellt den geodätischen Abstand zweier Punkte dar. In der Mitte wird der geodätische Abstand der Punkte mittels eines mathematischen Graphen und der resultierenden roten Linie approximiert. Rechts ist das dimensionsreduzierte Ergebnis von Isomap abgebildet.

Der Vorteil dieses Algorithmus liegt darin, dass durch die Verwendung geodätischer Distanzen und lokaler Nachbarschaften die Eigenschaften der Mannigfaltigkeit besser berücksichtigt werden. Gerade bei Datensets wie der *Swiss Role* (Abbildung 14) ist dieses Vorgehen wichtig, um die Eigenschaften der hochdimensionalen Daten im niedrigdimensionalen Raum entsprechend abzubilden. Der Algorithmus trifft die Annahme, dass die vorhandenen Daten gleichverteilt aus der Mannigfaltigkeit gezogen sind. Wird diese Annahme verletzt, werden die Nachteile dieses Algorithmus deutlich. Lokale Nachbarschaften können nicht mehr korrekt durch die *k* nächsten Nachbarn repräsentiert werden. Es können sogenannte Kurzschlüsse entstehen, die die gesamte Ähnlichkeitsmatrix beeinflussen (Balasubramanian und Schwartz 2002). Als Folge wird die Mannigfaltigkeit nicht mehr hinreichend genau beschrieben und die niedrigdimensionale Repräsentation wird bedeutungslos. Dieses Risiko kann durch kleine Werte für *k* gemindert werden. Dadurch wird der Nachbarschaftsgraph jedoch dünn besetzt und die geodätischen Pfade können nicht mehr exakt nachgebildet werden. Dies resultiert darin, dass die Informationen über die globalen Eigenschaften der Daten verloren gehen und die Mannigfaltigkeit nicht mehr ausreichend genau beschrieben wird.

Diese Eigenschaften zeigen, dass eine sorgfältige Auswahl des Hyperparameters *k* getroffen werden muss. Es bleibt darüber hinaus festzuhalten, dass der Algorithmus durch die Verwendung von geodätischen Distanzen in Verbindung mit dem *Multidimensional Scaling* eher globale Eigenschaften erhält.

2.6.3.3 T-Distributed Stochastic Neighbor Embedding

Wie Isomap gehört auch *t-distributed Stochastic Neighbor Embedding* (t-SNE) zu den nichtlinearen Verfahren der Dimensionsreduktion. Der Fokus des Verfahrens liegt auf der Visualisierung hochdimensionaler Daten in zwei oder drei Dimensionen und wird in Anlehnung an van der Maaten und Hinton (2008) beschrieben. Ausgehend von den hochdimensionalen Daten wird als erstes die Ähnlichkeit der einzelnen Instanzen bestimmt. Dies erfolgt, indem euklidische Distanzen in konditionelle Wahrscheinlichkeiten umgewandelt werden. Für zwei Punkte *p* und *o* entspricht sie der Wahrscheinlichkeit, dass der Punkt *p* den Punkt *o* als Nachbarn auswählt. Für deren Ermittlung wird die Wahrscheinlichkeitsdichtefunktion einer Gaußverteilung verwendet (Gleichung 2.14).

Die Varianz in den Daten kann dabei in unterschiedlichen Bereichen variieren. In Gebieten mit hoher Dichte, sollte die Varianz kleiner sein als in Bereichen geringer Dichte. Daher wird die Varianz mittels der sogenannten *Perplexität* berechnet. Diese ist abhängig von den vorliegenden Daten und muss entsprechend vom Anwender in Form eines Hyperparameters festgelegt werden. Die resultierende Ähnlichkeitsmatrix ist nicht symmetrisch, da die jeweiligen Nachbarschaften der beiden Punkte unterschiedlich sind. Um lediglich einen Wert für jedes Punktepaar zu erhalten, werden die beiden Werte daher gemittelt.

Nachdem die Ähnlichkeitsmatrix für den hochdimensionalen Raum vorliegt, erfolgt die Berechnung für die niedrigdimensionalen Daten. Allerdings werden die Ähnlichkeiten zwischen den Punkten nicht mittels der oben beschriebenen Wahrscheinlichkeitsdichtefunktion einer Gaußverteilung, sondern anhand der sogenannten Studentschen t-Verteilung berechnet. Der Unterschied zwischen den beiden Verteilungen liegt darin, dass die Enden der Studentsche t-Verteilung höhere Werte im Vergleich zur Gaußverteilung besitzen (siehe Abbildung 15).

Dadurch wird erreicht, dass Punkten mit moderaten Distanzen im hochdimensionalen Raum größere Werte in der Distanzmatrix zugewiesen werden und somit verhindert, dass diese Punkte im niedrigdimensionalen Raum zu nah beieinander eingebettet werden.

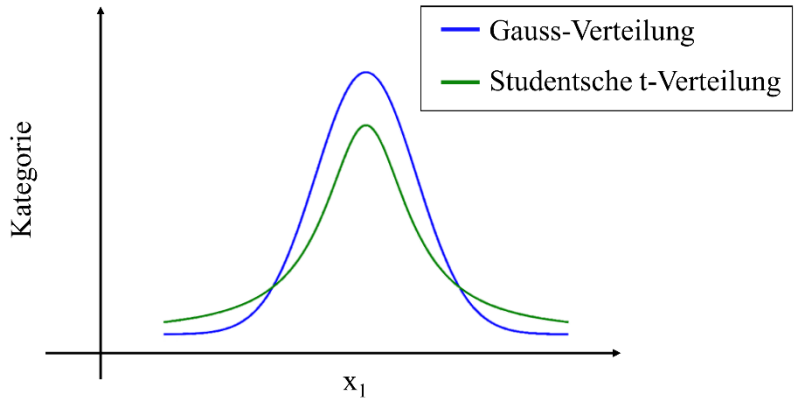


Abbildung 15: Exemplarische Darstellung einer Gauß- und Studentschen t-Verteilung

Nachdem die Werte für die beiden Verteilungen vorliegen, kann mithilfe der Kullback-Leibler Divergenz deren Ähnlichkeit berechnet werden. Mit Hilfe eines Gradientenabstiegsverfahrens wird die Kullback-Leibler-Divergenz minimiert. Die Initialisierung erfolgt mittels einer zufälligen Einbettung der Punkte durch eine isotrope Gaußverteilung, die um den Ursprung zentriert ist.

Der Vorteil des t-SNE Algorithmus ist, dass er gegenüber anderen Verfahren besser dazu in der Lage ist, Cluster in den Daten zu finden und entsprechend zu visualisieren. Darüber hinaus kann der Anwender mit Hilfe der einstellbaren Perplexität festlegen, ob der Algorithmus eher lokale oder globale Eigenschaften erhalten soll. Dadurch ist ein flexibles Explorieren der Daten möglich.

Im Vergleich zu Algorithmen wie Isomap liegt dem t-SNE Algorithmus jedoch eine nicht konvexe Optimierung zugrunde. Daher ist nicht gewährleistet, dass das globale Optimum gefunden wird. Stattdessen ist es möglich, dass der Algorithmus in einem lokalen Minimum stecken bleibt, was der Grund dafür ist, dass bei einer mehrmaligen Berechnung des Algorithmus mit denselben Hyperparametern unterschiedliche Einbettungen mit variierender Qualität gefunden werden (stochastisch).

Ein weiterer Nachteil ist, dass die Perplexität als Hyperparameter nicht intuitiv vom Benutzer einzustellen ist. Kleine Werte der Perplexität bevorzugen die Erhaltung lokaler Eigenschaften, während große Werte die Erhaltung der globalen Eigenschaften priorisieren. Insgesamt lässt sich festhalten, dass t-SNE auf Grund seines Designs globale Eigenschaften weniger stark erhält. Die Auswirkungen der Parameterwahl sind komplex und nur durch mehrere Versuche mit unterschiedlichen Perplexitäten kann eine für den konkreten Anwendungsfall geeignete Einbettung gefunden werden. In der ersten Veröffentlichung zu dem Algorithmus (van der Maaten und Hinton 2008) wird ein Wertebereich zwischen 5 und 50 vorgeschlagen. In jedem Fall sollte die Perplexität kleiner als die Größe des Datensets gewählt werden.

Bei der Anwendung von t-SNE muss beachtet werden, dass Clustergrößen und Clusterabstände nicht interpretiert werden können. Dies liegt an dem Design von t-SNE, wodurch die

hochdimensionalen Daten bei der Dimensionsreduktion verzerrt werden. Bereiche niedriger Dichte werden komprimiert, während Bereiche hoher Dichte auseinandergezogen werden. Im Gegensatz zur PCA und zu Isomap werden also die Abstände zwischen den Punkten im hoch- und niedrigdimensionalen Raum nicht erhalten. Darüber hinaus bleibt festzuhalten, dass dieser Algorithmus durch die nicht konvexe Optimierung eine vergleichsweise hohe Berechnungszeit aufweist.

Aus diesem Grund sind die Bestrebungen der Forschungen auf diesem Gebiet, die Dimensionsreduktion bei gleichbleibender Qualität zu beschleunigen, die Anfälligkeit in lokalen Optima der Optimierung stecken zu bleiben zu reduzieren, sowie neben den lokalen Eigenschaften gleichzeitig die globalen Eigenschaften zu erhalten. Ein Algorithmus, der diese Punkte adressiert, ist UMAP, dessen Funktionsweise im Folgenden erläutert wird (McInnes et al. 2018).

2.6.3.4 Uniform Manifold Approximation and Projection

Der Algorithmus *Uniform Manifold Approximation and Mapping* (UMAP) basiert ähnlich wie Isomap auf der Erzeugung eines Nachbarschaftsgraphen im hochdimensionalen Raum. Gleichzeitig wird jedoch auch für die niedrigdimensionalen Daten ein Nachbarschaftsgraph berechnet. Ähnlich wie bei t-SNE findet im Anschluss eine Optimierung statt, mit der versucht wird, die Eigenschaften der hochdimensionalen Daten so gut wie möglich in der niedrigdimensionalen Einbettung zu erhalten. Im Folgenden wird die Funktionsweise von UMAP erklärt. Für mathematische Beweise und Details zur Umsetzung wird auf McInnes et al. (2018) verwiesen, an deren Anlehnung die Funktionsweise im Folgenden beschrieben wird.

Zunächst wird dabei die Annahme getroffen, dass die hochdimensionalen Daten auf einer Mannigfaltigkeit eingebettet liegen und gleichverteilt von dieser gezogen sind. Darüber hinaus wird angenommen, dass die vorhandenen Daten lokal verbunden sind. Um nun den hochdimensionalen Graphen zu erzeugen, wird ein sogenannter Cech-Komplex konstruiert. Die Idee dahinter ist, die komplexe kontinuierliche Mannigfaltigkeit durch eine einfachere Kombinatorik anhand von sogenannten Simplexen anzunähern. Um die Topologie zu beschreiben, genügt es nicht zu wissen, welcher Punkt konkret mit welchem verbunden ist. Es müssen vielmehr Ähnlichkeiten zwischen den einzelnen Instanzen berechnet werden. Hierfür wird zunächst mittels eines Radius eine Umgebung für jeden Punkt definiert. Diesen Radius vorab zu bestimmen und festzulegen ist jedoch aufgrund der unterschiedlichen Beschaffenheit verschiedener Datensets nicht möglich. Wird ein zu kleiner Radius gewählt, werden die Punkte jeweils isoliert dargestellt, wird ein zu großer Umkreis bestimmt, sind alle Punkte miteinander verbunden. Daher wird ein flexibler Radius abhängig von der jeweiligen Umgebung eines Punktes p gewählt. Der Radius zu einem beliebigen Punkt p ist proportional zur Distanz zum k-ten Nachbarn von p. Entsprechend hat jeder Punkt im euklidischen Raum einen unterschiedlichen Radius. Hierbei entspricht k einem vom Anwender festzulegenden Hyperparameter, der die Anzahl zu berücksichtigender Nachbarn bezeichnet. Wird k zu klein gewählt, werden primär lokale Eigenschaften repräsentiert. Dabei geht allerdings die Information über die globalen Eigenschaften verloren. Große Werte für k legen dagegen den Fokus auf die globalen Eigenschaften der Topologie.

Es wird sichergestellt, dass durch diese Vorgehensweise keine isolierten Punkte entstehen. Jeder Punkt ist mit mindestens einem seiner nächsten Nachbarn verbunden. Damit nicht sämtliche Nachbarn innerhalb des *k*-ten Radius dieselbe Ähnlichkeit besitzen, fällt diese ausgehend vom ersten Nachbarn in Abhängigkeit von der Distanz zum aktuell betrachteten Punkt ab. Nun stehen die Ähnlichkeiten zwischen den einzelnen Punkten fest.

Ein ähnliches Vorgehen wird für die niedrigdimensionalen Daten wiederholt. Auch hier kann der Anwender durch einen Hyperparameter Einfluss auf die Resultate von UMAP nehmen. Der Parameter *min-dist* beeinflusst die Wahrscheinlichkeiten, dass zwei Punkte im niedrigdimensionalen Raum miteinander verbunden sind und legt damit die minimale Distanz zwischen zwei Punkten fest. Für kleine Werte werden die Punkte nah beieinander eingebettet, für große Werte dagegen weiter entfernt voneinander.

Nachdem die Ähnlichkeiten für die hoch- und niedrigdimensionalen Daten vorliegen, kann wie bei t-SNE die niedrigdimensionale Einbettung der Daten optimiert werden, sodass sie bestmöglich die Eigenschaften der originalen Daten repräsentiert. Die Ähnlichkeit der Verteilungen wird anhand der Kreuzentropie bestimmt, die entsprechend als Zielfunktion für die Optimierung verwendet wird. Als Optimierungsalgorithmus wird das stochastische Gradientenabstiegsverfahren verwendet.

Nach McInnes et al. (2018) ist ein wesentlicher Vorteil dieser Berechnung gegenüber t-SNE die Skalierbarkeit hinsichtlich des Berechnungsaufwands gegenüber einer steigenden Anzahl an Datenpunkten sowie der Dimension der Einbettung. Dies soll eine Echtzeitanwendung der Dimensionsreduktion ermöglichen. Darüber hinaus sollen hier neben den lokalen Eigenschaften, die auch t-SNE gut erhalten kann, gleichzeitig die globalen Eigenschaften besser erhalten werden.

Auch bei UMAP wird eine nicht konvexe Zielfunktion optimiert, sodass bei mehrmaliger Ausführung des Algorithmus mit denselben Hyperparametern unterschiedliche Einbettungen resultieren (stochastisch). Im Vergleich zu t-SNE sind laut den Autoren die Schwankungen der Qualität der Einbettung allerdings geringer.

Nachteilig wirkt sich im Vergleich zu den anderen für die Dimensionsreduktion verwendeten Algorithmen die Notwendigkeit der Definition von zwei Hyperparametern aus. Darüber hinaus sind ebenso wie bei t-SNE Clustergrößen und –abstände nicht zwingend interpretierbar. Allerdings werden nach McInnes et al. (2018) globale Eigenschaften besser abgebildet, sodass Clusterabstände bei UMAP eher eine Bedeutung im Vergleich zu t-SNE besitzen. Es muss zudem beachtet werden, dass gerade bei kleinen Datensets sowie bei vorhandenem Rauschen in den Daten das Risiko besteht, die Mannigfaltigkeit nur mangelhaft zu approximieren. Dies resultiert in einer schlechten Qualität der niedrigdimensionalen Repräsentation.

2.7 Qualitätskriterium zur Bewertung der Qualität einer neuen Datenrepräsentation

In den vergangenen Jahren wurde eine Vielzahl an Algorithmen für die nichtlineare Dimensionsreduktion vorgestellt (Lee und Verleysen 2007; van der Maaten et al. 2009). Einige davon verfügen über Hyperparameter, die vom Anwender eingestellt werden müssen. Hierbei den richtigen Wert zu finden, kann aufwändig und zeitintensiv sein. Darüber hinaus sind einige Algorithmen aufgrund nicht-konvexer Eigenschaften stochastisch, sodass aus mehreren Berechnungen der Dimensionsreduktion jeweils unterschiedliche Einbettungen resultieren und anschließend die beste manuell selektiert werden muss.

Um diese Herausforderungen zu lösen, ist ein Qualitätskriterium hilfreich, das die Qualität der Dimensionsreduktion durch einen skalaren Wert darstellt und damit eine Automatisierung der Hyperparameteroptimierung erlaubt. Darüber hinaus wird dadurch der quantitative Vergleich unterschiedlicher Dimensionsreduktions-Algorithmen ermöglicht.

In der Literatur existieren hierzu eine Vielzahl an Qualitätskriterien, von denen sich jedoch nicht jedes für sämtliche Aufgabenstellungen eignet (Zhang et al. 2012; Meng et al. 2011; Lee und Verleysen 2010). Eine Möglichkeit, die Qualität der Dimensionsreduktion zu bewerten, stellt beispielsweise die Berechnung des Rekonstruktionsfehlers dar (Verleysen und Lee 2013). Hierfür ist es jedoch notwendig, dass der Algorithmus zur Dimensionsreduktion eine Rücktransformation in den originalen Raum erlaubt, sodass beispielsweise der mittlere quadratische Fehler zwischen den originalen und den rekonstruierten Daten als Qualitätskriterium herangezogen werden kann. Da einige der in dieser Arbeit betrachteten Algorithmen nicht über die Möglichkeit der Rücktransformation verfügen, muss ein anderes Kriterium verwendet werden.

Eine weitere Möglichkeit stellt die Verwendung der Klassifikationsgenauigkeit dar (Verleysen und Lee 2013). Anhand der dimensionsreduzierten Repräsentationen wird ein Klassifikator trainiert und dessen Genauigkeit als Kriterium herangezogen. Dabei wird die Annahme getroffen, dass eine gute Dimensionsreduktion, gleichzeitig gute Ergebnisse bei der Klassifikation hervorruft. Dieses Kriterium eignet sich jedoch ebenfalls nur eingeschränkt, da für die Crashsimulationen üblicherweise keine kategorisierten Daten zur Verfügung stehen, mit denen ein Klassifikator trainiert und evaluiert werden könnte.

Eines der in der Literatur am häufigsten verwendeten Verfahren wird in Lee und Verleysen (2008b) durch die Einführung der Co-Ranking Matrix vorgestellt und im Folgenden in Anlehnung an die Veröffentlichung erläutert.

Das Ziel der Dimensionsreduktion ist es üblicherweise, Nachbarschaften zu erhalten, sodass nächste Nachbarn im hochdimensionalen auch nächste Nachbarn im niedrigdimensionalen Raum sind. Daher wird im ersten Schritt sowohl für die hochdimensionalen als auch die niedrigdimensionalen Daten jeweils eine Distanzmatrix berechnet, um die Nachbarschaften zu beschreiben. Bei einigen Verfahren der Dimensionsreduktion (z.B. t-SNE, UMAP) stimmen die euklidischen Distanzen zwischen den Punkten im hoch- und niedrigdimensionalen Raum jedoch nicht überein. Somit wird für die Bewertung der Qualität ein robusteres, von der Distanzmatrix abgeleitetes Maß für die Beschreibung der Nachbarschaften verwendet. Hierfür wird die Distanzmatrix in eine Rangmatrix überführt. In letzterer beschreibt ein Eintrag in der Matrix nicht die Distanz zwischen zwei Punkten, sondern der wievielte Nachbar, der eine Punkt zum anderen

ist. Die beiden Rangmatrizen stellen im Anschluss daran als gemeinsames zweidimensionales Histogramm die Co-Ranking Matrix dar. Deren schematischer Aufbau ist in Abbildung 16 dargestellt. Die beiden Achsen des Diagramms entsprechen den jeweiligen *k*-ten Nachbarschaften. Befinden sich alle Einträge der Co-Ranking Matrix exakt auf der Diagonalen, liegt eine perfekte Dimensionsreduktion vor, da sämtliche Nachbarschaftsbeziehungen im niedrigdimensionalen Raum mit denen im hochdimensionalen Raum übereinstimmen.

Einträge rechts neben der Diagonalen zeigen ein extrusives Verhalten. Dies bedeutet, dass Punkte, die im hochdimensionalen Raum enge Nachbarn sind, nach der Dimensionsreduktion weiter voneinander entfernt dargestellt werden. Einträge links der Diagonalen werden als Intrusionen bezeichnet. Punkte, die im hochdimensionalen Raum entfernte Nachbarn sind, werden im niedrigdimensionalen Raum näher beieinander eingebettet. Desto mehr und desto weiter entfernt von der Diagonalen Einträge vorhanden sind, desto schlechter werden Nachbarschaften erhalten und desto niedriger ist die Qualität der Dimensionsreduktion.

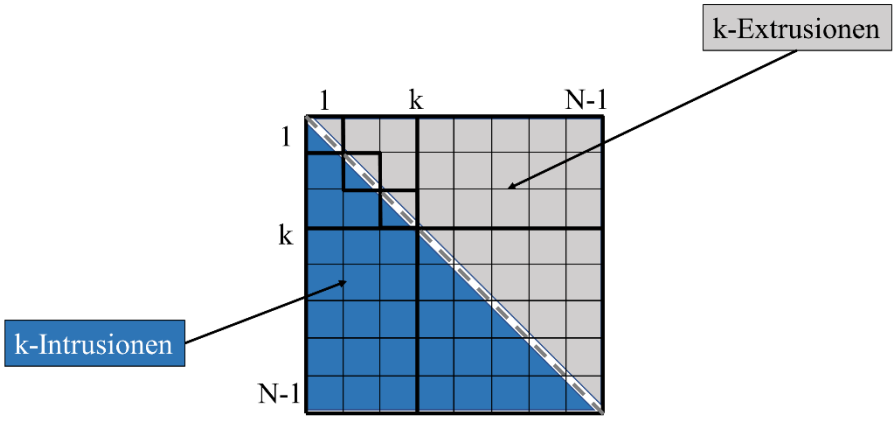


Abbildung 16: Schematische Darstellung der Co-Ranking Matrix mit farblicher Hervorhebung der K-Intrusionen und K-Extrusionen (in Anlehnung an Lee und Verleysen (2008a))

In Lee und Verleysen (2008a) wird gezeigt, dass die Co-Ranking Matrix in verschiedene Blöcke unterteilt werden kann, auf deren Basis wiederrum unterschiedliche Qualitätskriterien berechnet werden können. Verschiedene Blöcke und Dreiecke werden dabei gewichtet oder ungewichtet miteinander verrechnet und liefern dadurch unterschiedliche Qualitätskriterien. Die Autoren erklären hierzu, dass jede Art von Gewichtung beliebig ist und zeigen auf, dass auch ein ungewichteter Ansatz die Eigenschaften der Dimensionsreduktion über verschiedene Nachbarschaftsgrößen hinweg berücksichtigen kann.

Eines der bekanntesten Kriterien, welches auch in dieser Arbeit verwendet wird ist Q_{NX} (Griparis et al. 2016). Es liefert in Abhängigkeit der Größe der betrachteten Nachbarschaft K einen Wert, der aussagt, wie gut die Nachbarschaften im hoch- und niedrigdimensionalen Raum übereinstimmen und ermöglicht damit eine Aussage über die Qualität der Dimensionsreduktion. Das Kriterium Q_{NX} leitet sich aus der Co-Ranking Matrix ab

$$\boldsymbol{Q_{NX}(K) = \frac{1}{NK} \sum_{(k,l)\epsilon UL_K} q_{kl}} \quad , \tag{2.16}$$

wobei UL_K den Einträgen im oberen linken Quadranten der Co-Ranking Matrix entsprechen, k und l Laufvariablen für die entsprechenden Einträge q_{kl} der Co-Ranking Matrix sind und N die Anzahl an Dateninstanzen darstellt. Für jede mögliche Nachbarschaftsgröße K wird das entsprechende Q_{NX} berechnet, sodass als Resultat ein Kurvenverlauf vorliegt, der die Qualität der Dimensionsreduktion in Abhängigkeit von K darstellt. Q_{NX} beschreibt bei der Betrachtung kleiner beziehungsweise großer K, inwiefern lokale beziehungsweise globale Nachbarschaften aus dem hochdimensionalen im niedrigdimensionalen Raum erhalten bleiben.

In Griparis et al. (2016) werden verschiedene Algorithmen zur Dimensionsreduktion anhand von Q_{NX} verglichen, um den besten für die 3D Visualisierung einer Vielzahl von Bildern zu identifizieren. Neben dem Zweck, Verfahren zur Dimensionsreduktion zu vergleichen, wird Q_{NX} darüber hinaus dafür eingesetzt, die Hyperparameter eines Algorithmus zu optimieren. In Alaız et al. (2015) werden die Hyperparameter von Diffusion Maps durch eine Variante von Q_{NX} optimiert.

Um die Qualität der Dimensionsreduktion anhand eines einzelnen Skalars zu beschreiben, können weitere Qualitätskriterien aus dem Verlauf von Q_{NX} abgeleitet werden. Die einfachste Möglichkeit stellt beispielweise die Berechnung des Mittelwerts über sämtliche Nachbarschaften K dar. Dieses Kriterium wird im Folgenden mit Q_{AVG} bezeichnet (Lee und Verleysen 2010)

$$\boldsymbol{Q_{AVG} = \frac{1}{N-1} \sum_{K=1}^{N-1} Q_{NX}(K)} \quad . \tag{2.17}$$

Es gibt zahlreiche weitere Methoden, um die Qualität der Dimensionsreduktion zu quantifizieren. Für einen umfassenden Überblick wird auf Gracia et al. (2014) sowie Lee und Verleysen (2010) verwiesen, wo verschiedene Dimensionsreduktions-Algorithmen und Verfahren zu deren Qualitätsbewertung verglichen werden.

3 Prozessintegration

In diesem Kapitel wird die bisherige CAE-Prozesskette erläutert und es wird darauf eingegangen, wie die neuen Analyseverfahren in diese integriert werden können. Im Speziellen wird darauf eingegangen, welche Abläufe in der automatisierten und welche in der manuellen Auswertung ablaufen.

Abbildung 17 stellt den bisherigen Entwicklungsprozess der Crashauslegung vereinfacht dar. Nachdem eine Simulation am Hochleistungsrechner berechnet wurde, erfolgt als nächstes die automatisierte Auswertung. Ein Skript, in dem Standardauswertungen berechnet werden, wird automatisch gestartet, sodass dem Anwender die Ergebnisse in der manuellen Analyse sofort zur Verfügung stehen und keine Wartezeiten auftreten. Für die Standardauswertung werden zusätzlich automatisiert Screenshots aus den Crashsimulationen für festgelegt Bauteile und Zeitpunkte erzeugt.

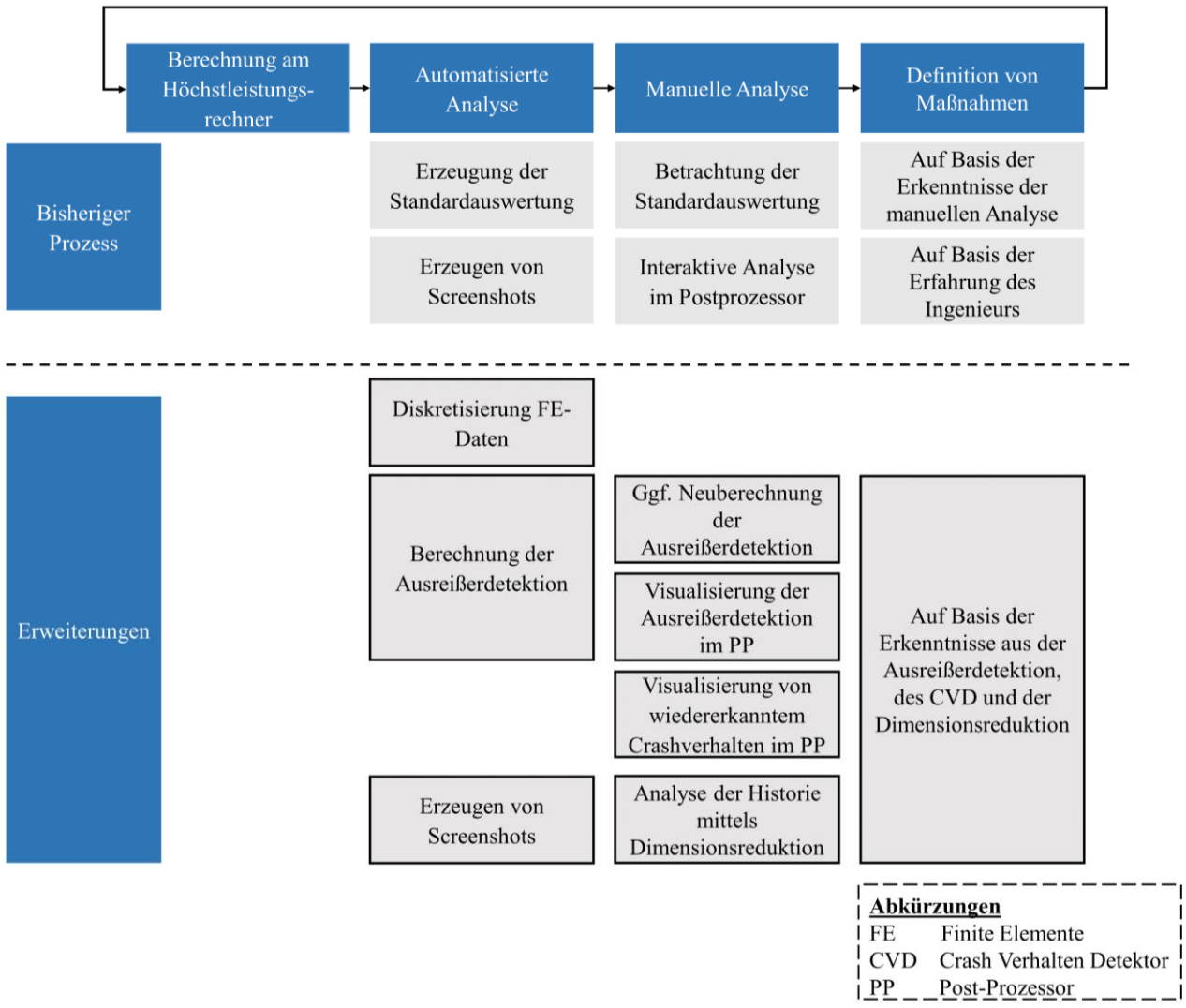


Abbildung 17: Integration der neuen Methode und deren einzelner Berechnungsschritte und Bestandteile in den bestehenden Auswertungsprozess

Im Anschluss daran folgt die manuelle Analyse. Zum einen werden die vorberechneten Standardauswertungen betrachtet. Darüber hinaus erfolgt die interaktive Analyse der Crashergebnisse in einem Post-Prozessor wie beispielsweise dem *Animator*. Interaktiv bedeutet an dieser Stelle, dass die dreidimensionalen Crashmodelle nach Belieben des auswertenden Ingenieurs von sämtlichen Seiten betrachtet werden, Schnitte durch das Fahrzeug gezogen werden können, um hinter der Karosserie liegende Bauteile zu analysieren sowie beliebige Auswertungsgrößen auf dem FE-Netz farblich dargestellt werden können (wie beispielsweise plastische Dehnungen), um besonders belastete Fahrzeugbereiche zu identifizieren. Gegebenenfalls erfolgen an dieser Stelle der Vergleich mit anderen Simulationen aus der Entwicklungshistorie und detaillierte Analysen mit Software wie Modelcompare oder Diffcrash. Anschließend werden Maßnahmen basierend auf den neu gewonnenen Erkenntnissen der Analysen und der Erfahrung des Ingenieurs definiert. Neue Crashmodelle werden aufgebaut, der Kreis schließt sich und eine neue Iteration beginnt.

Im Folgenden wird gezeigt, wie die vier Methoden des mit dieser Arbeit entwickelten neuen Analyseansatzes in den bisherigen Prozess integriert werden können und an welcher Stelle die einzelnen Berechnungen hierzu abzulaufen haben. Da sowohl das Einladen der Simulationsdatei in den Arbeitsspeicher (mittels effizienter Python-Bibliothek (LASSO GmbH 2022) in ca. 2 Minuten für ein Gesamtfahrzeug mit 10 Mio. FE) als auch die Berechnung nachfolgender Analyseverfahren zeitaufwändig ist, werden sämtliche Prozessschritte, die vorberechnet werden können, in den automatisierten Analyse-Teil integriert. Für manche Algorithmen ist eine Vorberechnung nicht möglich, da zunächst die Eingabe ergänzender Informationen durch den Anwender erforderlich ist. Deren Berechnung muss daher im manuellen Analyse-Teil erfolgen. Um dennoch eine interaktive Auswertung zu ermöglichen, wird an diese Verfahren die Anforderung einer geringen Berechnungszeit gestellt. Abbildung 17 stellt die Integration der neuen Methoden in den bestehenden Prozess überblicksweise dar. Die jeweils neuen Verfahrensschritte sind dabei im unteren Teil der Abbildung als „Erweiterungen“ dargestellt.

Sobald eine neue Simulation vom Hochleistungsrechner berechnet wurde, wird die automatisierte Analyse gestartet. Wie bisher auch wird zunächst eine Standardauswertung erstellt. Zusätzlich werden in diesen Schritt sämtliche Analysen integriert, die vorab durchgeführt werden können, um dadurch eine interaktive bzw. Auswertung in Echtzeit im manuellen Analyseteil zu ermöglichen.

Als erster neuer Schritt werden sodann die FE-Daten diskretisiert, um eine einheitliche Repräsentation zu erlangen und die weitere maschinelle Verarbeitung vorzubereiten. Als Informationsquelle hierfür dient die „d3plot“-Datei aus der LS-DYNA Simulation. Jedes Bauteil wird einzeln diskretisiert und die Ergebnisse daraus jeweils in einzelnen Dateien abgespeichert, in denen dann auch die Informationen über die jeweiligen Auswertungsgrößen vorhanden sind (z. B. zur plastischen Dehnung). Dieses Vorgehen ermöglicht später schnelle Zugriffszeiten auf die benötigten Informationen im Rahmen der interaktiven manuellen Analyse.

In einem zweiten Schritt erfolgt die Berechnung der Ausreißerdetektion. Hierbei ist zu beachten, dass stets ein Kontext definiert werden muss, innerhalb welchem eine neue Simulation hinsichtlich des auffälligen Crashverhaltens einzelner Bauteile bewertet wird. Diesen Kontext muss der Anwender demnach vorab festlegen (beispielsweise die letzten 20 Simulationen).

Damit bei der späteren Analyse der Dimensionsreduktionsergebnisse zusätzliche Metainformationen wie Bilder und Videos zu den einzelnen Simulationen (Punkte oder Linien in einem Diagramm) dargestellt werden können, ist es zudem notwendig, diese vorab zu erzeugen. Gerade bei einer Vielzahl von Simulationen kann dieser Schritt sehr zeitaufwändig sein, da sämtliche Simulationen zunächst in den *Animator* importiert werden müssen. Erst daraufhin erfolgt die Generierung der Bilder und Videos, für die ebenfalls einige Minuten pro Simulation benötigt werden. Daher wird dieser zeitintensive Schritt bereits in die automatisierte Analyse integriert, sodass die Ergebnisse im manuellen Analyseteil auf Abruf des Anwenders sofort zur Verfügung stehen.

Nachdem somit sämtliche erforderliche Daten vorliegen, kann die manuelle Auswertung durch den/die Ingenieur*in erfolgen. Nach wie vor ist hierbei die Standardauswertung zu betrachten und die Ergebnisse werden weiterhin im *Animator* visualisiert. An dieser Stelle gibt es nun jedoch Unterstützung durch die Ausreißerdetektion. So werden die vorberechneten Ergebnisse der Ausreißerdetektion nunmehr auf dem FE-Netz im *Animator* visualisiert, wobei die einzelnen Bauteile entsprechend ihrer Auffälligkeiten im Kontext der anderen Simulationen farblich hervorgehoben werden.

An dieser Stelle wird ergänzend angemerkt, dass die Ausreißerdetektion auf Wunsch auch während der manuellen Analyse neu berechnet werden kann. Dies kann für den Anwender beispielsweise dann interessant werden, wenn die betrachtete neue Simulation im Kontext anderer bzw. weiterer Simulationen auf auffälliges Crashverhalten untersucht werden soll als zunächst vom Anwender für die vorberechnete Ausreißerdetektion festgelegt (z. B. die letzten 20 Simulationen). Diese Vorgehensweise ist deshalb möglich, da die Ergebnisse durch die bauteilbasierte Diskretisierung bereits vorberechnet vorliegen. Dies ermöglicht zum einen schnelleren Datenzugriff auf die Informationen beliebiger Simulationen. Zum anderen sind parallele Berechnungen auf mehreren Prozessen (Bauteile werden parallel analysiert) möglich.

Der CVD ist ebenfalls dazu in der Lage, das Crashverhalten einzelner Bauteile während des manuellen Analyseschrittes auszuwerten. Für jedes Bauteil, das vom Anwender im Laufe der Entwicklung markiert wird, erfolgt das Training eines entsprechenden Modells. Das Training erfolgt in regelmäßigen Abständen, beziehungsweise immer schon dann automatisiert im Hintergrund, sobald neue kategorisierte Daten von dem/der Ingenieur*in zur Verfügung gestellt werden. Die Resultate können daher im *Animator* immer neu visualisiert werden, sodass der Anwender sofort sieht, bei welchen Bauteilen das markierte Crashverhalten erneut auftritt.

Dabei werden die Prädiktionen des CVD nicht in der automatisierten Auswertung berechnet, da im Laufe der Entwicklung die Ergebnisse älterer Simulationen ohnehin nicht vollständig ausgewertet werden würden. So gibt es zu Beginn eines Entwicklungsprojekts beispielsweise einen CVD für fünf Bauteile. Die Analyse nachfolgender Simulationen würde daher lediglich für diese Bauteile erfolgen. Im Zuge der Fahrzeugentwicklung werden sodann aber neue CVD für weitere Bauteile trainiert. Die CVD-Ergebnisse einer alten Simulation zum damaligen Zeitpunkt ihrer automatisierten Analyse, wären vom heutigen Standpunkt aus betrachtet immer unvollständig, da darin die neuen CVD nicht berücksichtigt sind. Es bestünde so erneut das Risiko, dass etwas übersehen wird. Daher erfolgt die Prädiktion des CVD im Rahmen der manuellen Analyse.

Die Dimensionsreduktion wird ebenfalls innerhalb der manuellen Analyse berechnet. Zum einen erfordert diese nur eine geringe Berechnungszeit, sodass eine Vorberechnung aus zeitlichen Gesichtspunkten nicht zwingend erforderlich ist. Zum anderen gibt es Verfahren für die Dimensionsreduktion, die die Definition von Hyperparametern erfordern und damit erst interaktiv während der Anwendung eingestellt werden können, was eine Vorberechnung ausschließt. Darüber hinaus muss zunächst ein Kontext an anderen Simulationen definiert sein, innerhalb dessen die Dimension der Daten reduziert werden soll. Dieser Kontext ist stark abhängig vom jeweilig vorliegenden Analyseziel und variiert damit ständig, was eine Vorberechnung der Dimensionsreduktion erneut ausschließt. Da die Berechnung der Dimensionsreduktion jedoch innerhalb weniger Sekunden erfolgt, stellt dies hinsichtlich der interaktiven Auswertung ohne lange Wartezeiten kein Problem dar. Schlussendlich kann der/die Ingenieur*in die Ergebnisse der Dimensionsreduktion in einer interaktiven mit Metadaten angereicherten Darstellung für die einzelnen Bauteile betrachten und so auch das Crashverhalten jedes einzelnen Bauteils einsehen.

4 Verwendete Crash-Simulationsdaten für den Entwurf des neuen Analyseansatzes

In diesem Kapitel werden die für den Entwurf des neuen Analyseansatzes verwendeten Bauteile sowie deren Crashverhalten vorgestellt. Die verwendeten Simulationen stammen aus einer Robustheitskampagne eines Vorderwagenabschnittsmodells (Andricevic 2016). Dieses ist beispielhaft in Abbildung 18 dargestellt. Es besteht überwiegend aus Aluminium-Strangpressprofilen mit elasto-plastischem Materialverhalten. Die Geometrien sind mit Schalenelementen und einer Kantenlänge von 5mm vernetzt.

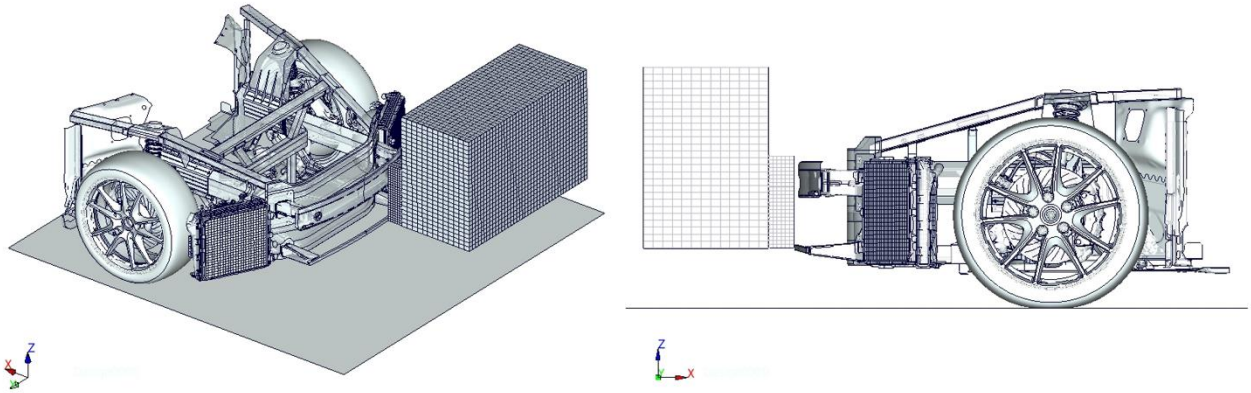


Abbildung 18: Vorderwagenabschnittsmodell im Offset Deformable Barrier (ODB) Lastfall

Bei dem betrachteten Lastfall handelt es sich um einen teilüberdeckten Frontcrash gemäß dem *EuroNCAP* (*European New Car Assessment Programme)* (carhs GmbH 2012), der im Folgenden als *Offset Deformable Barrier* (ODB) bezeichnet wird. Da in Andricevic (2016) Robustheitsanalysen durchgeführt wurden, wird dieser Lastfall aufgrund seiner Komplexität und der Neigung zu Bifurkationen im Lastpfad ausgewählt. Aus denselben Gründen eignen sich die durchgeführten Simulationen besonders für diese Arbeit, da unterschiedlichstes Crashverhalten mit den einzelnen Verfahren analysiert werden soll.

Die Simulationen wurden mit der Software LS-DYNA durchgeführt. Die Dauer des Crashs beträgt 124ms. Die Ergebnisse der Simulation werden alle 2ms in der d3plot-Datei abgespeichert. Dies bedeutet, dass die zeitliche Information von 62 Zeitschritten vorhanden ist.

Die einzelnen Varianten unterscheiden sich, indem die Wandstärken von 36 Bauteilen innerhalb verschiedener Wertebereiche variieren. In der Herstellung von Blechen für die Fahrzeugproduktion sind Streuungen in den Wandstärken unvermeidlich. Deren erlaubte Grenzen sind in einer Normung in Abhängigkeit der Bauteildimensionen definiert und in Tabelle 1 in Anlehnung an Andricevic (2016) abgebildet. Durch die Betrachtung des Crashverhaltens im Rahmen dieser Streuungen, ist es möglich die Robustheit einer Fahrzeugstruktur zu bewerten.

Insgesamt wurden 200 Simulationen durchgeführt, in denen die Wandstärken gemäß des *Advanced Latin Hypercube-Sampling*-Verfahrens (Huntington und Lyrintzis 1998) variieren. Die

200 Simulationen setzen sich aus vier Sätzen mit jeweils 50 Rechnungen zusammen. Die vier Sätze unterscheiden sich in den Streubereichen, die für die jeweiligen Wandstärken angenommen wurden. Dabei sind die Toleranzen in den ersten beiden Sätzen (DIN1, DIN2) mittels der einfachen sowie der doppelten DIN-Norm festgelegt. Für den dritten und vierten Satz sind die Grenzen auf ein Prozent (DIN01) beziehungsweise ein Promille (DIN001) der DIN-Norm definiert. Details zu dem Simulationsmodell und den gestreuten Parametern können (Andricevic 2016) entnommen werden.

Tabelle 1: Streubreiten der Wanddicken in Anlehnung an (Andricevic 2016)

Werkstoff	Bauweise	Wanddicke [mm]	Streubreite [mm]
Al-Legierung	Druckguss	≤10	±0,75
			±1
	Strangpressprofil	≤2	±0,2
			±0,3
		>2…3	±0,25
			±0,4
St-Legierung	Blech	≤2	±0,13

Das Ziel der in Kapitel 5 untersuchten Diskretisierungsverfahren ist die Überführung unterschiedlicher FE-Netze in ein einheitliches strukturiertes Format, um weitere maschinelle Analysen zu ermöglich. Die eben beschriebenen Simulationsdaten verfügen jedoch bereits über ein einheitliches Datenformat, da sich lediglich die Wandstärken der Varianten unterscheiden und damit deren FE-Netze identisch sind. Um den Einfluss der Diskretisierung auf den Informationsgehalt gegenüber den originalen FE-Daten quantifizieren zu können, wird das in Kapitel 2.7 vorgestellte Qualitätskriterium verwendet. Hierfür ist es erforderlich, dass Nachbarschaften der originalen FE-Daten berechnet werden können. Dies ist jedoch nur gegeben, wenn diese ebenfalls in einem einheitlichen strukturierten Format vorliegen. Würden sich die FE-Netze unterscheiden, wäre die Berechnung der Nachbarschaften nicht möglich. Aus diesem Grund werden die Daten aus der Robustheitskampagne mit gleichen FE-Netzen verwendet, um den Einfluss der Diskretisierungsverfahren quantifizieren zu können. An dieser Stelle wird explizit darauf hingewiesen, dass diese Vorgehensweise lediglich für die Quantifizierung des Informationsverlusts durch die Diskretisierung verwendet wird.

Neben der einheitlichen Datenrepräsentation ist es erforderlich, die Auswertungsgröße zu definieren, die für die Analysen verwendet werden soll. In den Ergebnissen der Crashsimulationen stehen neben Verschiebungen und Beschleunigungen beispielsweise die Information über die plastische Dehnung der einzelnen FE zur Verfügung. Letztere werden in

dieser Arbeit exemplarisch für die Auswertungen verwendet. Dies stellt keine Einschränkung der Methoden dar, da diese in der realen Anwendung mit jeder beliebigen Auswertungsgröße verwendet werden können.

In den Analysen verwendete Bauteile

Für die Analysen in Kapitel 5 werden exemplarisch 13 Bauteile aus dem Vorderwagen verwendet. Deren Einbauort und Lage im Fahrzeug ist in Abbildung 19 farblich hervorgehoben. Die Bauteile stammen demnach überwiegend aus der linken Hälfte des Fahrzeugs (das Fahrzeug vom Heck aus betrachtet), da hier aufgrund der Asymmetrie des Lastfalls die größten Deformationen entstehen und die Informationen ausgewertet werden sollen.

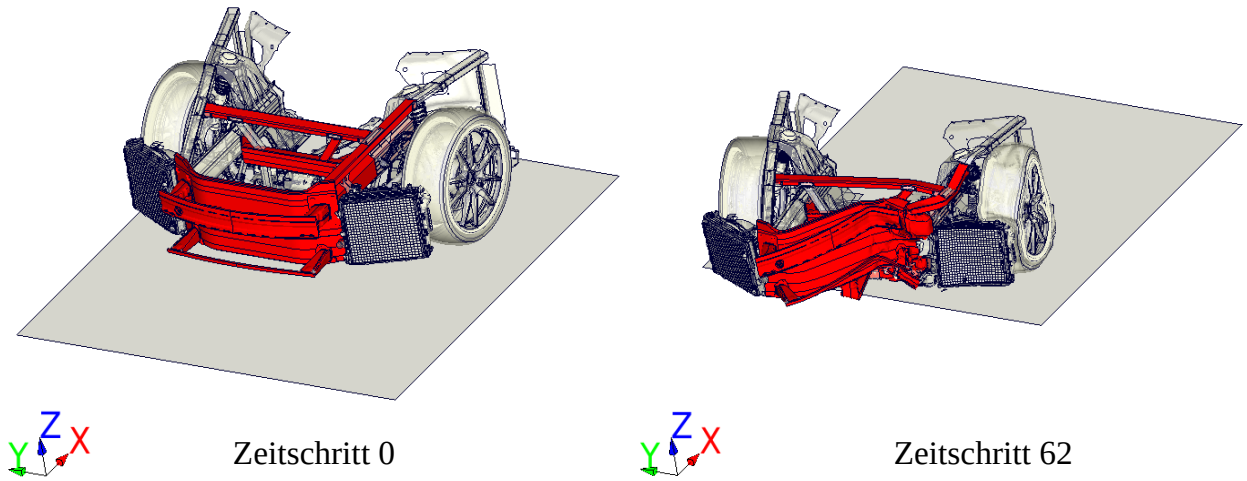
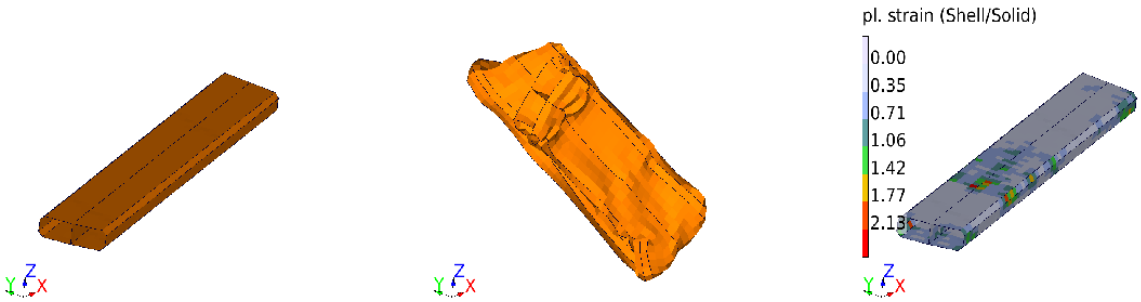


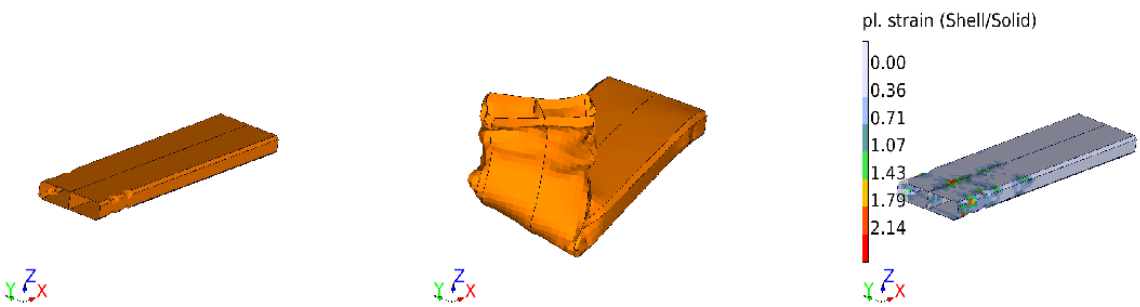
Abbildung 19: Vorderwagenabschnittmodell mit in den Analysen verwendeten eingefärbten Bauteilen. Links: Undeformierte Geometrie zum Zeitschritt 0. Rechts: Deformierter Zustand zum letzten Zeitschritt 62

In Abbildung 20 werden die Bauteile in einer detaillierten Ansicht dargestellt. Da der Vorderwagen überwiegend aus Aluminiumstrangpressprofilen besteht, sind die meisten abgebildeten Bauteile profilförmig. Dagegen stellt die *Schottwand* ein Beispiel für ein großes sowie flächiges Bauteil dar. Dadurch, dass die konkrete Lage der Bauteile im Fahrzeug variiert und sich das Fahrzeug in dieser Simulation in einem Frontlastfall befindet, werden manche axial, andere dagegen lateral belastet. Deren Deformationsmodi unterscheiden sich dementsprechend deutlich voneinander. In jeder Zeile der nachfolgenden Abbildung ist ein Bauteil dargestellt. In der ersten Spalte wird das undeformierte Bauteil zum initialen Zeitschritt des Crashs gezeigt, in der zweiten Spalte die finale Deformation zum Zeitschritt 62. In der dritten Spalte sind die plastischen Dehnungen Zeitschritt 62 auf der Oberfläche des Bauteils in % visualisiert. Die Verschiebungen sind auf den Wert 0,001 skaliert, sodass die undeformierte Geometrie mit den plastischen Dehnungen als Elementfunktion zu sehen ist.

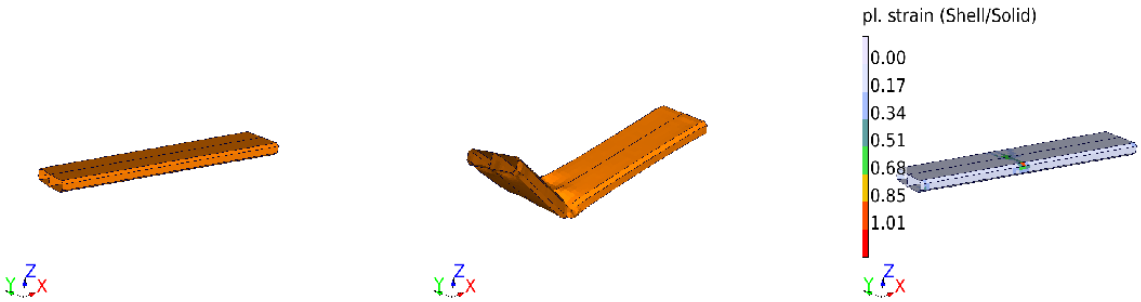
Strebe_Links



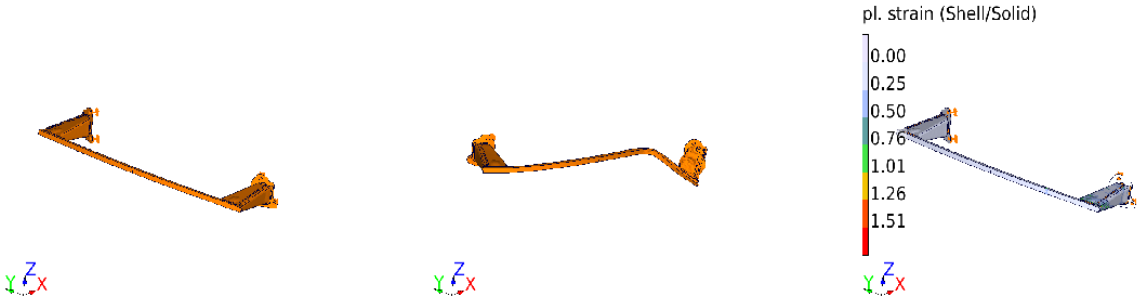
Strebe_Mitte



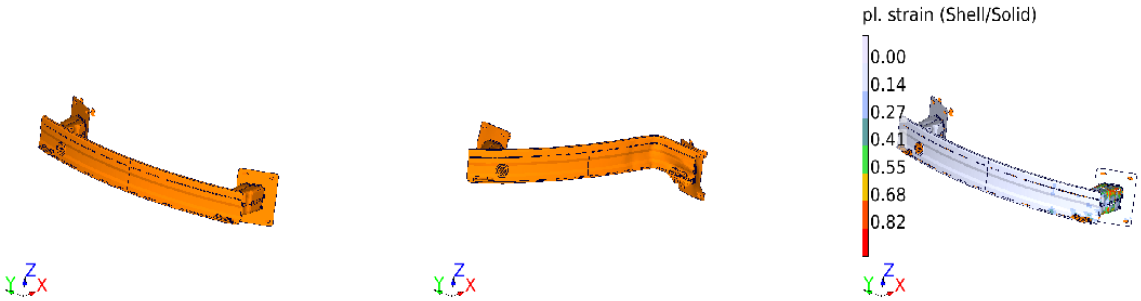
Strebe_Rechts



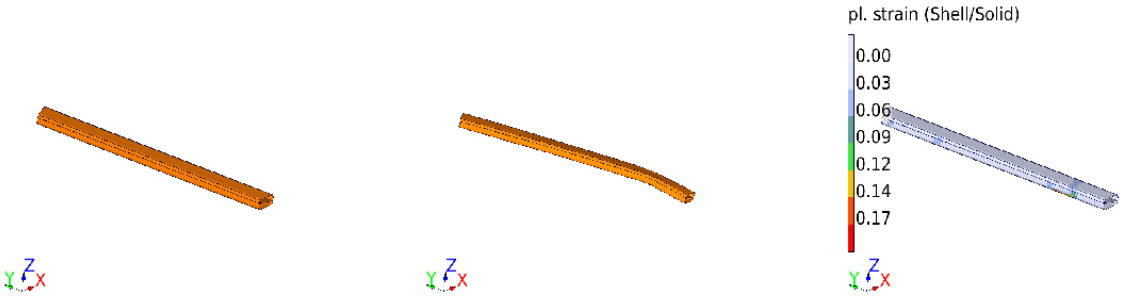
CMS_Unten



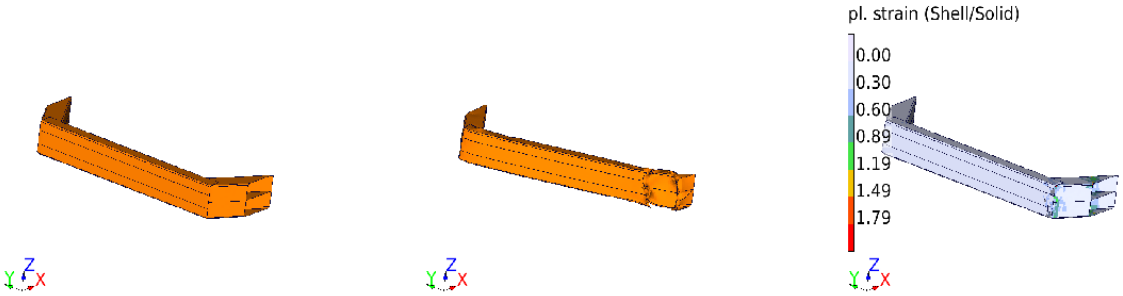
CMS_Oben



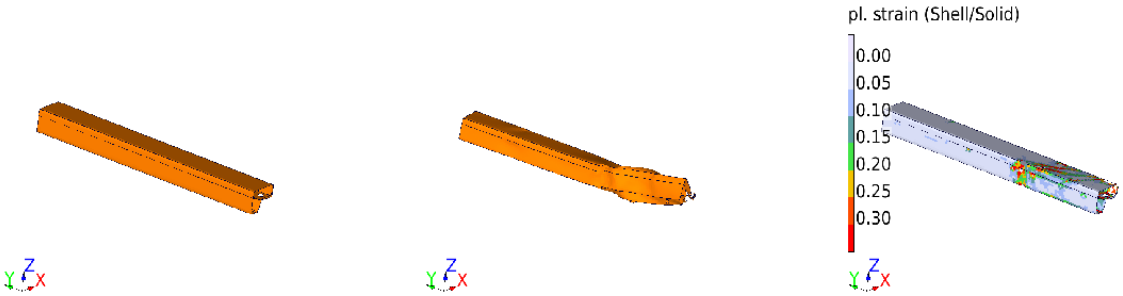
QT_Oben



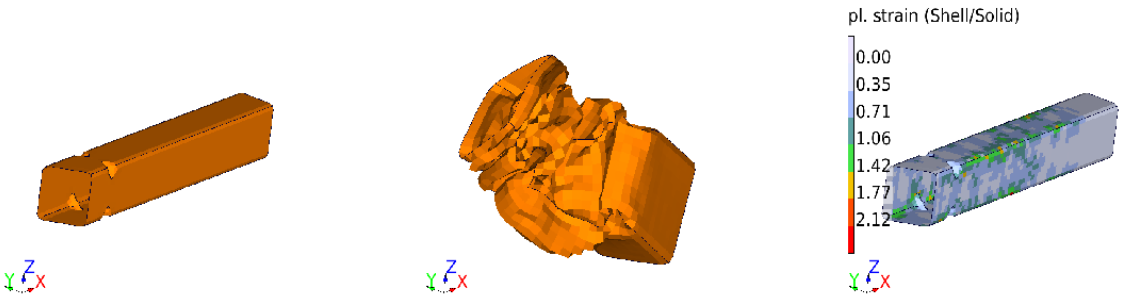
QT_Mitte



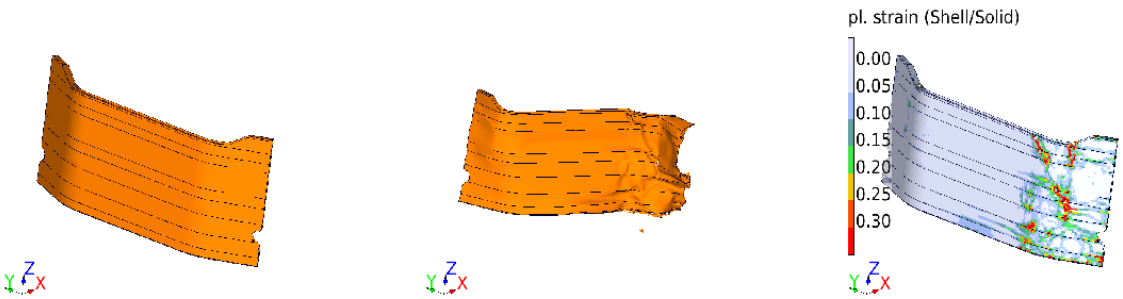
QT_Unten



LT_Unten_Links



SW



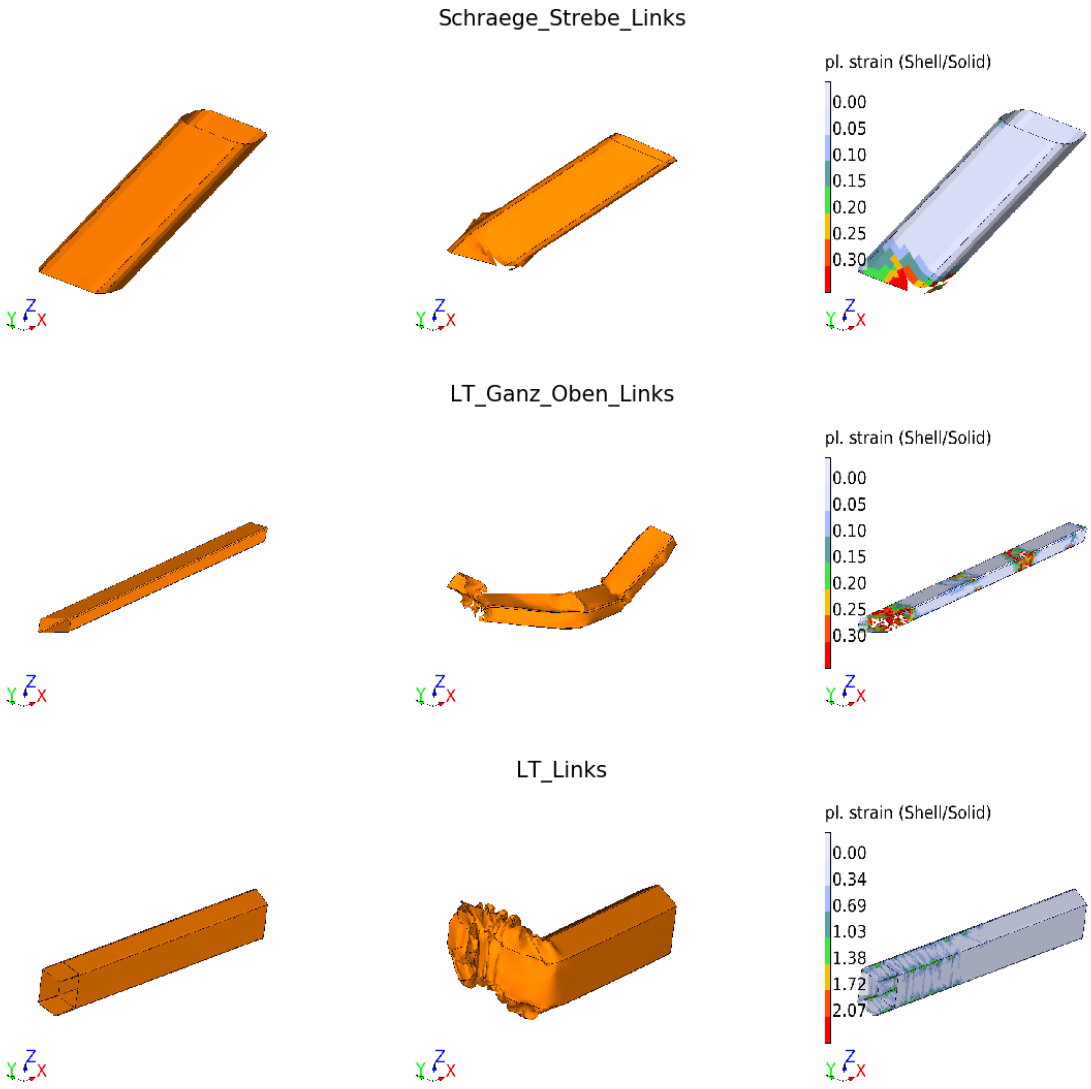


Abbildung 20: Darstellung der in den Analysen verwendeten Bauteile zum initialen und letzten Zeitschritt der Simulation (linke und mittlere Spalte) sowie der plastischen Dehnungen zum letzten Zeitschritt (rechte Spalte). Dabei sind die plastischen Dehnung auf dem undeformierten Bauteil visualisiert

Für die Untersuchungen in Kapitel 5 zu den unterschiedlichen Diskretisierungsverfahren werden alle in Abbildung 19 dargestellten Bauteile verwendet. Für die Analysen in den Kapiteln sechs bis acht wird der *Längsträger* in der unteren Lastebene auf der linken Fahrzeughälfte betrachtet, da dieser ein sehr charakteristisches Deformationsverhalten in den Simulationen zeigt.

Crashmodi des Längsträgers in der unteren Lastebene

Aufgrund der Parameterstreuungen treten in den einzelnen Simulationen unterschiedliche Crashverhalten auf. Besonders in der unteren Lastebene sind dabei Bifurkationen zu beobachten. Im Vergleich zu den anderen Bauteilen des Vorderwagens zeigt der *Längsträger* in der unteren Lastebene (siehe Abbildung 21) die größten Unterschiede im Crashverhalten zwischen den Simulationen. Damit eignet sich dieser besonders gut für die nachfolgenden Analysen. Für die anderen Bauteile sind Veränderungen oft nur im kleinsten Detail erkennbar und daher für den

Anwender auf den ersten Blick nur schwer auszumachen. Deshalb wird aus Gründen der Anschaulichkeit der oben beschriebene *Längsträger* in den Kapiteln sechs bis acht verwendet.

Dieser zeigt im Wesentlichen drei unterschiedliche Deformationsmodi auf, die in Abbildung 21 umfassend dargestellt werden. Im ersten Modus (Links) beginnt die Deformation des axial belasteten Bauteils in dessen vorderen Bereich und breitet sich im Verlauf der Simulation weiter nach hinten aus. Im zweiten (Mitte) verhält es sich umgekehrt, indem die initiale Deformation im hinteren Bereich stattfindet und sich im Verlauf nach vorne ausbreitet. Der dritte Deformationsmodus (Rechts) zeigt ein Mischverhalten aus den beiden vorherigen Verhalten, wobei die Deformation vorne und hinten gleichzeitig beginnt.

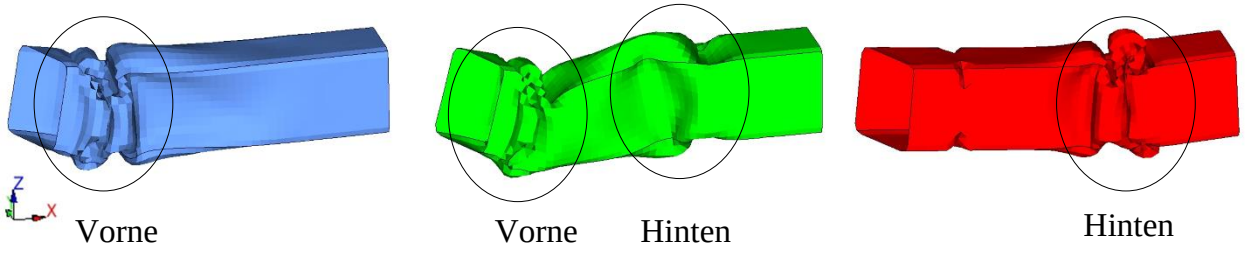


Abbildung 21: Darstellung drei verschiedener Deformationsmodi eines axial belasteten Längsträgers in der unteren Lastebene. Links: Faltet vorne. Mitte: Faltet vorne und hinten. Rechts: Faltet hinten

5 Datenrepräsentation

Um Simulationen mit unterschiedlichen FE-Netzen analysieren zu können, ist eine entsprechende Datenvorverarbeitung erforderlich, sodass sichergestellt wird, dass in der Datenanalyse stets die Information aus denselben Bereichen der Bauteile miteinander verglichen wird. In diesem Kapitel werden die beiden Diskretisierungsverfahren mittels der sphärischen Projektion und Voxel miteinander verglichen.

5.1 Anforderungen

Eine Anforderung an das Verfahren zur einheitlichen Datenrepräsentation ist, dass die Ergebnisse für den Anwender möglichst intuitiv interpretierbar sind.

Darüber hinaus sollen die neu generierten Daten möglichst wenig Speicherplatz auf der Festplatte beanspruchen. Damit ein schneller Datenzugriff für die späteren Analyse sichergestellt wird, soll die Datenaufbereitung automatisiert ablaufen, sobald eine neue Simulation berechnet ist. Die Dateigröße der einzelnen d3plot Ergebnisdateien beträgt bereits allein mehrere GB pro Simulation. Daher soll der benötigte Plattenplatz durch die neu generierten Daten so gering wie möglich sein.

Bei einigen Verfahren zur einheitlichen Datenrepräsentation gehen Informationen gegenüber den originalen Ergebnissen verloren. Beispielsweise hängt der verbleibende Informationsgehalt bei Diskretisierungsverfahren von der Granularität der Diskretisierung sowie dem Verfahren selbst ab. Damit für die anschließenden Datenanalysen die relevanten Informationen über das Crashverhalten aus der Simulation zur Verfügung stehen, ist es notwendig, dass der Informationsverlust der neuen Datenrepräsentation so gering wie möglich ist.

5.2 Implementierungen der Diskretisierungsverfahren

In diesem Kapitel werden die verwendeten Implementierungen des Sphären- sowie Voxel-Verfahrens vorgestellt. Für beide Methoden erfolgt die Zuweisung der FE zu den entsprechenden Pixeln beziehungsweise Voxeln zum initialen Zeitschritt, in dem die undeformierte Geometrie eines Bauteils vorliegt. Diese Zuordnung bleibt für alle nachfolgenden Zeitschritte bestehen, damit bei der Datenanalyse stets dieselben Bereiche des Bauteils miteinander verglichen werden.

Eine Neuzuweisung zu jedem Zeitschritt würde zu einer fehlerhaften Analyse führen, denn die Verschiebungen der FE unterscheiden sich von Simulation zu Simulation. Demzufolge würden die FE in jeder Simulation in unterschiedliche Voxel einsortiert werden. Es würden nicht mehr dieselben Bauteilbereiche in ähnlichen Bereichen der Voxel abgebildet werden und somit unterschiedliche Informationen in der späteren Analyse miteinander verglichen werden.

Für beide Verfahren muss daher bereits zu Beginn festgelegt werden, welche Auswertungsgröße den diskretisierten Elementen zugeordnet werden soll. Für den Fall, dass mehrere FE demselben diskretisierten Element zugeordnet werden, wird deren Mittelwert gebildet und dieser für die weitere Analyse verwendet.

Der Unterschied zwischen dem Sphären- und Voxel-Verfahren liegt in der Einsortierung der FE in die Pixel beziehungsweise Voxel. Dies wird im Folgenden in Anlehnung an eine Vorveröffentlichung des Autors (Kracker et al. 2020) für beide Vorgehensweisen beschrieben.

5.2.1 Sphären-Verfahren

Im ersten Schritt des Sphären-Verfahrens wird der geometrische Mittelpunkt des Bauteils zum initialen Zeitschritt bestimmt. Dies erfolgt, indem der Mittelwert über sämtliche FE jeweils für die x-, y- und z-Koordinate getrennt berechnet wird.

Im Anschluss daran wird jedes einzelne FE des Bauteils entlang einer Geraden ausgehend vom Mittelpunkt auf eine Kugeloberfläche projiziert (siehe Abbildung 5).

Die Projektion auf eine Kugeloberfläche wird durch die Transformation von kartesischen in Kugelkoordinaten umgesetzt. Dabei sind die beiden Winkel φ und ϑ (siehe Abbildung 22) ausschlaggebend für die Position der projizierten Elemente auf der Kugeloberfläche.

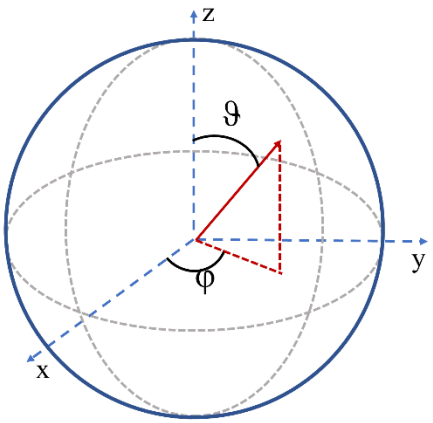


Abbildung 22: Visualisierung von Kugelkoordinaten und der beiden Azimut- und Polarwinkel φ und ϑ

Deren Wertebereiche reichen von 0° bis 360° beziehungsweise von 0° bis 180°. Der Radius spielt bei diesem Verfahren keine Rolle, da die Sortierung der FE in die einzelnen Pixel lediglich auf den beiden Winkeln basiert. Im Anschluss wird die Kugeloberfläche diskretisiert, indem die beiden Winkel in jeweils gleich viele Intervalle unterteilt werden. Dies wird vom Anwender durch den Parameter K_S gesteuert, der festlegt, wie viele Pixel das resultierende Bild der Diskretisierung in den beiden Dimensionen enthält. Der Winkel φ wird in äquidistante Intervalle der Größe

$$\boldsymbol{\varphi_i} = \begin{cases} \mathbf{0°}, & \boldsymbol{i = 1} \\ \boldsymbol{\varphi_{i-1}} + \dfrac{\mathbf{360°}}{\boldsymbol{K_S}}, & \boldsymbol{sonst} \end{cases} \tag{5.1}$$

zwischen 0° und 360° unterteilt.

Würde auch ϑ äquidistant unterteilt werden, wären die Kugeloberflächenelemente im Bereich von 0° und 180° sehr klein, während die im Bereich um 90° deutlich größer sind. Die einzelnen diskretisierten Bereiche auf der Kugeloberfläche würden unterschiedlich genau die FE-Daten der sphärischen Projektion abbilden. Um eine gleichmäßigere Verteilung der einsortierten FE auf der diskretisierten Kugeloberfläche zu erzielen, wird der Winkel ϑ so zwischen 0° und 180° unterteilt, dass die entstehenden Kugeloberlächenelemente jeweils den gleichen Flächeninhalt besitzen. Damit ergibt sich für den i-ten Winkelbereich des Winkels ϑ (i ist eine Laufvariable mit i = 1, …, K_S)

$$\vartheta_i = \begin{cases} 0°, & i = 0 \\ \cos^{-1}\left(\cos\left(\vartheta_{i-1} * \frac{\pi}{180°}\right) - \frac{2}{K_S}\right) * \frac{180°}{\pi}, & sonst. \end{cases} \qquad (5.2)$$

Die Intervalle von φ und ϑ definieren die Lage der einzelnen Pixel auf der Kugeloberfläche. Im Anschluss werden die FE in die zugehörigen Kugeloberflächenelemente einsortiert. Das Ergebnis dieser Vorgehensweise ist damit eine zweidimensionale Darstellung der ursprünglich dreidimensionalen FE-Daten. Diese können nun durch ein zweidimensionales Bild wie in Abbildung 23 rechts dargestellt, visualisiert werden.

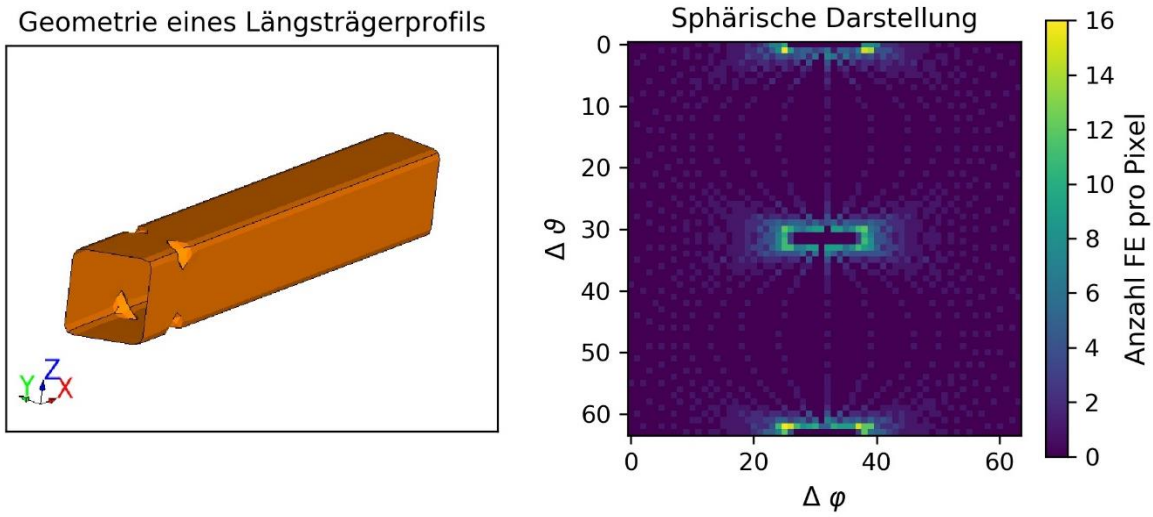


Abbildung 23: Links: Originale Geometrie eines Längsträgerprofils. Rechts: Resultierende Darstellung nach der sphärischen Projektion und Diskretisierung (K_S=64). Farblich dargestellt ist die Anzahl Finiter Elemente pro Pixel

In der linken Darstellung ist die Geometrie eines Längsträgerprofils abgebildet. Im rechten Bild ist farblich die Anzahl finiter Elemente pro diskretisiertem Kugeloberflächenelement dargestellt (siehe auch Abbildung 5). Die dunklen Pixel entsprechen Bereichen auf der Kugeloberfläche, in denen keine FE nach der Projektion vorhanden sind. Desto heller der Farbwert eines Pixels, desto mehr FE sind ihm im Zuge der Diskretisierung zugeordnet. Mit geringer Auflösung wird dieser

Fall öfter auftreten während mit größer werdenden K_S die Tendenz steigt, dass jedes FE einen eigenen Pixel erhält.

Bis hierhin sind lediglich die Geometrieinformationen über das Bauteil zum initialen Zeitschritt enthalten. Daher werden im Anschluss noch die Auswertungsgrößen hinzugefügt (hier die plastischen Dehnungen). Für jeden Zeitschritt aus der Simulation ergibt sich somit ein neues zweidimensionales Bild, das die entsprechenden Informationen über das Crashverhalten zu diesem Zeitschritt der Simulation enthält. Somit ergibt sich insgesamt als Resultat für jedes Bauteil ein dreidimensionaler Tensor. In den ersten beiden Dimensionen ist die projizierte Information der FE zu einem konkreten Zeitschritt enthalten, während die zeitliche Information in der dritten Dimension repräsentiert wird.

5.2.2 Voxel-Verfahren

Der Vorteil bei diesem Verfahren ist, dass im Gegensatz zur sphärischen Projektion die dreidimensionale Struktur der originalen Daten erhalten bleibt. Der Anwender definiert mit dem Diskretisierungsparameter K_V die Kantenlänge der Voxel, wobei diese für alle drei Raumrichtungen dieselbe ist. Damit bleibt bei dem Voxel-Verfahren im Gegensatz zu der soeben dargestellten Sphären-Diskretisierung die Intuition über den Einfluss des Diskretisierungsparameters erhalten. Der Anwender ist somit bei der Wahl von K_V dazu in der Lage abzuschätzen, wie viele FE in einem Voxel enthalten sein werden und wie gut ein Bauteil durch die gewählte Diskretisierung abgebildet wird.

Sobald die Positionen und Abmessungen der Voxel feststehen, werden die FE basierend auf ihren kartesischen Koordinaten in die entsprechenden Voxel einsortiert (siehe Abbildung 6).

Im Anschluss daran erfolgt die Übertragung der Informationen zu der ausgewählten Auswertungsgröße (z.B. plastische Dehnung) für jeden einzelnen Zeitschritt aus der Simulation.

Als Resultat dieses Verfahrens liegt ein vierdimensionaler Tensor für jedes Bauteil einer Simulation vor. In den ersten drei Dimensionen sind die projizierten Informationen der FE für einen einzelnen Zeitschritt enthalten. Durch die vierte Dimension wird das gesamte zeitliche Crashverhalten aus der Simulation repräsentiert.

5.2.3 Trennung von Informationen über Geometrieunterschiede und das Crashverhalten

Durch die beiden vorgestellten Diskretisierungsverfahren werden neben dem Crashverhalten auch Geometrieunterschiede zwischen den zu analysierenden Simulationen abgebildet. Für den Anwender ist es im Allgemeinen zwar auch interessant, die Geometrieunterschiede zwischen den Varianten zu identifizieren (siehe hierfür beispielsweise die Software ModelCompare (Garcke et al. 2017)), in dieser Arbeit steht jedoch die Analyse des Crashverhaltens im Vordergrund. Diese Thematik der erforderlichen Ausblendung von Geometrieunterschieden für das vorliegend zu untersuchende Crashverhalten bezieht sich sowohl auf die Ergebnisse des Sphären-, als auch auf die des Voxel-Verfahrens und soll an einem illustrativen Beispiel für die Anwendung der Dimensionsreduktion eines axial belasteten *Längsträgers* erläutert werden.

In 50 Simulationen befindet sich im vorderen Bereich des Bauteils ein Loch, während in weiteren 50 Simulationen keines vorhanden ist. In beiden geometrischen Varianten knickt der *Längsträger*

in jeweils der Hälfte der Fälle nach links beziehungsweise rechts aus. Das Ziel der Dimensionsreduktion ist, dieses Crashverhalten entsprechend darzustellen.

Bei diesem Beispiel sind demnach die Erwartungen des Anwenders darauf gerichtet, dass zwei Cluster mit jeweils 50 Simulationen identifiziert werden können, die das Crashverhalten in die beiden Crashmodi „knickt nach links“ und „knickt nach rechts“ aufteilen.

In der neuen Datenrepräsentation sind die Pixel beziehungsweise Voxel leer, in deren Bereich sich das Loch befindet, da hier keine FE vorliegen, die in die Pixel/Voxel einsortiert werden. In den Simulationen der anderen geometrischen Variante ohne Loch sind jedoch entsprechende Informationen in diesen Pixeln/Voxeln vorhanden. Bei einer Berücksichtigung sämtlicher Pixel/Voxel in der Analyse, würde die Information über das Crashverhalten und die geometrischen Unterschiede gemeinsam ausgewertet werden. Dementsprechend würde sich eine Aufteilung der 100 Simulationen in vier Cluster ergeben, die das Crashverhalten „knickt nach links“/„knickt nach rechts“ sowie die geometrische Varianten „mit Loch“ sowie „ohne Loch“ abbilden. Der Fokus dieser Arbeit liegt jedoch auf der reinen Analyse des Crashverhaltens der Bauteile. Damit eine von geometrischer Information getrennte Analyse erfolgen kann, ist es daher notwendig, dass lediglich jene Bereiche der Bauteile verglichen werden, in denen in sämtlichen zu betrachtenden Simulationen FE vorhanden sind.

Dies wird gelöst, indem einzelne Pixel/Voxel deaktiviert und aus der Analyse ausgeschlossen werden. Zunächst werden in jeder einzelnen Simulation jene Pixel/Voxel identifiziert, in denen FE existieren. Daraufhin wird für jeden dieser Pixel/Voxel überprüft ob in allen anderen Simulationen ebenfalls FE vorhanden sind. Falls dies der Fall ist, wird der Pixel/Voxel in der Auswertung berücksichtigt. Andernfalls liegen in diesem Bauteilbereich geometrische Unterschiede vor, sodass die Informationen dieser Pixel/Voxel nicht berücksichtigt werden. Dies erfolgt, indem deren Matrixeinträge über die gesamte Dauer des Crashs zu null gesetzt werden und damit nicht in die Analyse einfließen. Diese Vorgehensweise erlaubt die getrennte Auswertung von Unterschieden in der Geometrie und dem eigentlich zu untersuchenden Crashverhalten.

5.3 Vergleich des Sphären- und Voxel-Verfahrens

In den Anforderungen an die neue Datenrepräsentation wird definiert, dass das Diskretisierungsverfahren möglichst viele Informationen über die zugrunde liegenden FE Daten beinhalten und gleichzeitig möglichst wenig Speicherplatz beanspruchen soll. Im Folgenden werden diese beiden Aspekte unterersucht und es wird bewertet, welches Diskretisierungsverfahren sich diesbezüglich besser eignet. Die bessere Methode enthält mehr Informationen über die Crashdaten bei geringerem Speicherplatz. Dabei wird insbesondere der Einfluss durch den Diskretisierungsparameter K_S beziehungsweise K_V berücksichtigt.

5.3.1 Vorgehensweise

Damit für die Verfahren bewertet werden kann, wie viel Information gegenüber den originalen FE-Daten durch die Diskretisierung verloren geht, bedarf es einer Methode, die die nach der Diskretisierung enthaltenen Informationen mit den originalen Daten vergleicht und bewertet. Hierfür wird das in Kapitel 2.7 erläuterte Qualitätskriterium Q_{AVG} (Gleichung 2.17) verwendet, welches gemäß den Literaturangaben zur Bewertung der Qualität einer Dimensionsreduktion

eingesetzt wird. Das Ziel der Dimensionsreduktion ist, dass Nachbarschaften zwischen einzelnen Simulationen im hochdimensionalen Raum auch im niedrigdimensionalen Raum erhalten bleiben. Das Kriterium wertet diese Eigenschaften aus, wobei ein Wert von 1 einer vollständigen Übereinstimmung der Nachbarschaften und ein Wert von 0 einer zufälligen Einbettung entspricht.

Diese Vorgehensweise kann auf die Diskretisierung der FE-Daten übertragen werden. Dabei entsprechen die originalen FE-Netze aus der Simulation den hochdimensionalen und die Resultate der Diskretisierung den niedrigdimensionalen Daten. Damit die Nachbarschaften der originalen FE-Daten berechnet werden können, ist es notwendig, dass sich deren FE-Netze nicht unterscheiden. Deshalb werden in den vorgenommenen Analysen Simulationen aus einer Robustheitskampagne mit denselben FE-Netzen verwendet. An dieser Stelle wird betont, dass diese Vorgehensweise zur Bewertung unter Zuhilfenahme des Qualitätskriteriums notwendig ist, um damit den Informationsverlust durch die Diskretisierungsverfahren bestimmen zu können. Das Ziel der Diskretisierung ist daneben nach wie vor eine einheitliche Datenrepräsentation für den Fall zu schaffen, dass unterschiedliche FE-Netze in den zu untersuchenden Simulationen vorliegen. Nachdem auf diese Weise also die Nachbarschaften für die originalen FE-Netze und die der diskretisierten Daten berechnet sind, wird daraus das Qualitätskriterium Q_{AVG} abgeleitet, das den Informationsgehalt der diskretisierten Daten bewertet.

Um den benötigten Speicherplatz auszuwerten, wird die Dateigröße der diskretisierten Ergebnisse, die im CSR Format abgespeichert werden, herangezogen. In der vorliegenden Arbeit wird die CSR Implementierung aus Virtanen et al. (2020) verwendet.

Als Datengrundlage für die folgenden Untersuchungen dienen die in Kapitel 4 vorgestellten 13 Bauteile aus den 50 verschiedenen Simulationen aus Datenset DIN001. Für K_S beziehungsweise K_V werden exemplarisch die Werte aus Tabelle 2 verwendet. Durch diese Vorgehensweise können das Sphären- und Voxel-Verfahren bestmöglich auf den Erhalt der ursprünglichen Informationen hin verglichen werden.

Tabelle 2: In den Experimenten untersuchte Werte für die Diskretsierungsparameter K_S und K_V

Diskretisierungs-Verfahren	**Diskretisierung: grob → fein**
Sphäre K_S	16, 32, 64, 128, 256
Voxel K_V	50 mm, 40 mm, 30 mm, 20 mm, 10 mm

5.3.2 Einfluss der Diskretisierung auf den Informationsgehalt

Die diskretisierten Daten bestehen aus einzelnen Pixeln/Voxeln, die entweder Informationen über die plastischen Dehnungen der FE enthalten, falls in dem Bauteilbereich FE vorhanden sind, oder den Wert null aufweisen. Die Anzahl der Pixel/Voxel eines Bauteils, die Informationen über vorhandene FE beinhalten, wird im Folgenden mit N_{NNE} (Anzahl von Nicht-Null Elementen) bezeichnet.

Um den Informationsgehalt verschiedener Diskretisierungen zwischen dem Sphären- und Voxel-Verfahren miteinander zu vergleichen, wird der Verlauf der Qualität Q_{AVG} (Gleichung 2.17) in Abhängigkeit von N_{NNE} analysiert. Das Verfahren mit dem höheren Informationsgehalt für dieselbe N_{NNE} bildet dabei die Informationen über das Crashverhalten effizienter ab und eignet sich besser für die einheitliche Datenrepräsentation. Mit diesem Vorgehen wird es möglich, das Sphären- und Voxel-Verfahren sowie deren unterschiedliche Diskretisierungen in einem gemeinsamen Diagramm miteinander zu vergleichen.

Die Ergebnisse sind in Abbildung 24 exemplarisch für die *Schräge Strebe Links* sowie den *Längsträger* dargestellt. Für eine Übersicht weiterer Bauteil wird auf den Anhang A verwiesen.

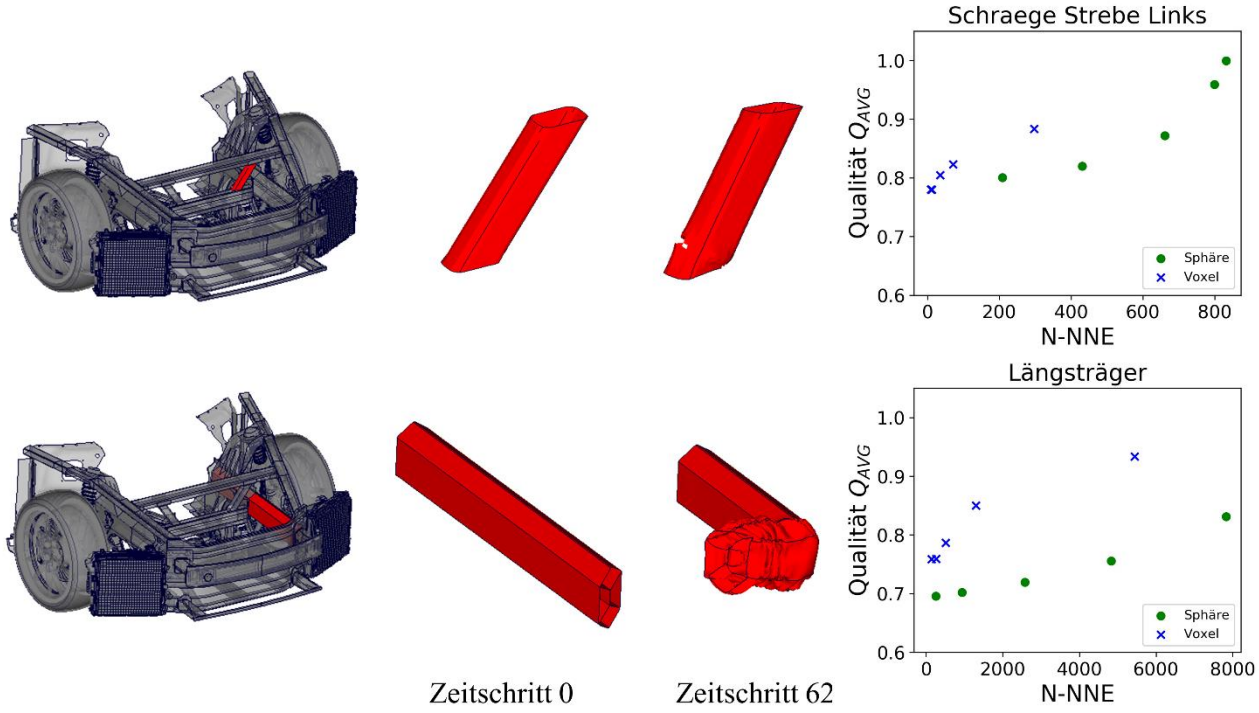


Abbildung 24: Darstellung der exemplarisch betrachteten Bauteile „Schräge Strebe Links“ und „Längsträger“ sowie des Qualitätskriteriums Q_{AVG} über der Anzahl Nicht-Null Elemente (N-NNE)

In den einzelnen Diagrammen ist jeweils das Qualitätskriterium Q_{AVG} über der Anzahl Nicht-Null Elemente N_{NNE} für die verschiedenen Diskretisierungen aus Tabelle 2 für das Sphären- und Voxel-Verfahren dargestellt. Daraus geht hervor, dass mit größeren Werten von N_{NNE}, der Informationsgehalt bei beiden Verfahren ansteigt. Die N_{NNE} steigen gleichzeitig mit größeren Werten von K_S beziehungsweise kleineren Werten von K_V. Es ist zu erkennen, dass die Qualität der Voxel-Daten beider Bauteile bei denselben Werten von N_{NNE} größer als die des Sphären-Verfahrens ist. Das Sphären-Verfahren erzielt jedoch für die Schräge Strebe Links für die großen Werte von K_S größere Werte für N_{NNE}, weshalb deren Qualität im Vergleich zu den untersuchten Diskretisierungen K_V des Voxel-Verfahrens höher ist.

Dass die Qualität pro N_{NNE} des Voxel-Verfahrens für die meisten Bauteile (vergleiche Anhang A) höher ist als beim Sphären-Verfahren liegt daran, dass das Sphären-Verfahren zu einer verzerrten Abbildung der Informationen führt. Die Stärke hängt dabei von der der Form des

Bauteils sowie dem vorliegenden Crashverhalten ab. Darüber hinaus hat die Größe des Bauteils einen Einfluss auf den abgebildeten Informationsgehalt beim Sphärenverfahren.

Um dies zu zeigen, werden die Ergebnisse der beiden Bauteile *Schräge Strebe Links* und *Längsträger* betrachtet. Der *Längsträger* wird ausgewählt, da hier die Überlegenheit der Voxel-Diskretisierung hinsichtlich deren Qualität deutlich zum Vorschein tritt, während bei der *Schrägen Strebe Links* die Sphären-Diskretisierung eine höhere Qualität erreicht.

In Abbildung 25 ist die Verteilung der FE auf die einzelnen Pixel der diskretisierten Kugeloberfläche exemplarisch für $K_S = 36$ dargestellt. Im linken Bild ist das Resultat für das Bauteil *Schräge Strebe Links* abgebildet, im rechten das des *Längsträgers*. Der Farbton stellt die Anzahl der FE pro Pixel dar.

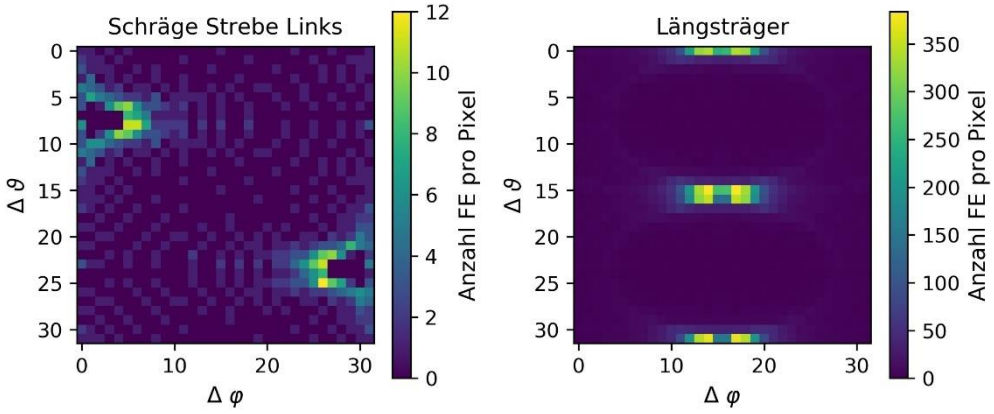


Abbildung 25: Darstellung der diskretisierten Ergebnisse des Sphären-Verfahrens exemplarisch für $K_S = 36$. Farblich hervorgehoben ist die Anzahl FE pro Pixel. Links ist die „Schräge Strebe Links", rechts der „Längsträger" dargestellt

Aus der Abbildung geht hervor, dass sich die beiden Enden des *Längsträgers* (Ort mit Häufungen von FE, im Bild die Bereiche mit hellen Pixeln) mittig oben und mittig unten im Bild befinden. Hier tritt die maximale Anzahl an FE in einem einzelnen Pixel auf und beträgt für den *Längsträger* 384. Da es sich um ein langes profilförmiges Bauteil handelt, tritt hier der Effekt durch die sphärische Verzerrung besonders deutlich auf. Stellt man sich vor, in der Mitte des *Längsträgers* bestehend aus einer Vielzahl an FE zu stehen und projiziert die einzelnen FE entlang der Geraden der Blickrichtung auf eine Kugeloberfläche, landen eine Vielzahl der FE der Enden des *Längsträgers* in sehr ähnlichen Bereichen auf der Kugeloberfläche. Die FE, die sich dagegen in der Mitte des Bauteils befinden werden in weniger dichte Regionen auf der Kugeloberfläche projiziert. Die anschließende Diskretisierung sorgt dafür, dass in den Bereichen der Enden des *Längsträgers* auf der Kugeloberfläche die Wahrscheinlichkeit höher ist, dass die Information mehrerer FE durch Bildung des Mittelwerts verschwimmt. In den mittleren Bereichen des Bauteils ist dies nicht der Fall. Dies ist demnach ein Effekt, der bei langen profilförmigen Bauteilen besonders stark zum Tragen kommt.

Bauteile deren Form einer Kugel ähneln, sind dagegen weniger von der Verzerrung betroffen, da hier die FE gleichmäßiger auf der Kugeloberfläche verteilt werden und dadurch weniger Bereiche mit Häufungen an FE entstehen. Dies bedeutet, dass ein Bauteil, mit ähnlicher Dimension in alle

drei Raumrichtungen wie beispielsweise die *Schräge Strebe Links*, beim Sphären-Verfahren besser abgebildet wird als ein profilförmiges. Es sind zwar auch Häufungen an den Enden des Bauteils vorhanden, diese fallen jedoch im Vergleich zum *Längsträger* schwächer aus. Die maximale Anzahl an FE pro Pixel liegt bei drei. Insgesamt werden die FE gleichmäßiger auf der Kugeloberfläche verteilt, sodass ein höherer Informationsgehalt abgebildet wird.

Neben der Form des Bauteils spielt der Ort der Deformation eine wichtige Rolle. An den Enden des *Längsträgers* befindet sich der Bereich des Bauteils, in dem die größten Verformungen in Folge des Crashs auftreten. Dies verstärkt den Effekt, dass der Informationsgehalt bezüglich des Crashverhaltens durch die Verzerrung reduziert wird. Gerade bei axial belasteten profilförmigen Bauteilen, die wie der *Längsträger* im vorderen oder hinteren Bereich deformieren, weist das Sphären-Verfahren demzufolge einen gravierenden Nachteil auf.

Das Voxel-Verfahren zeigt diese Nachteile nicht, da hier keine Verzerrung durch die Diskretisierung auftritt. Sämtliche Bauteilbereiche werden in gleichem Maße abgebildet. Dies ist in Abbildung 26 dargestellt. Daraus geht hervor, dass sowohl bei der *Schrägen Strebe Links* als auch beim *Längsträger* die FE gleichmäßiger in den einzelnen Voxeln verteilt sind. Insbesondere an den Enden der Bauteile treten nicht die extremen Häufungen auf, die beim Sphären-Verfahren durch die Verzerrung zu beobachten sind. Dies ist der Grund, weshalb der Informationsgehalt des Voxel-Verfahrens insbesondere für profilförmige Bauteile deutlich höher ist (siehe Abbildung 24).

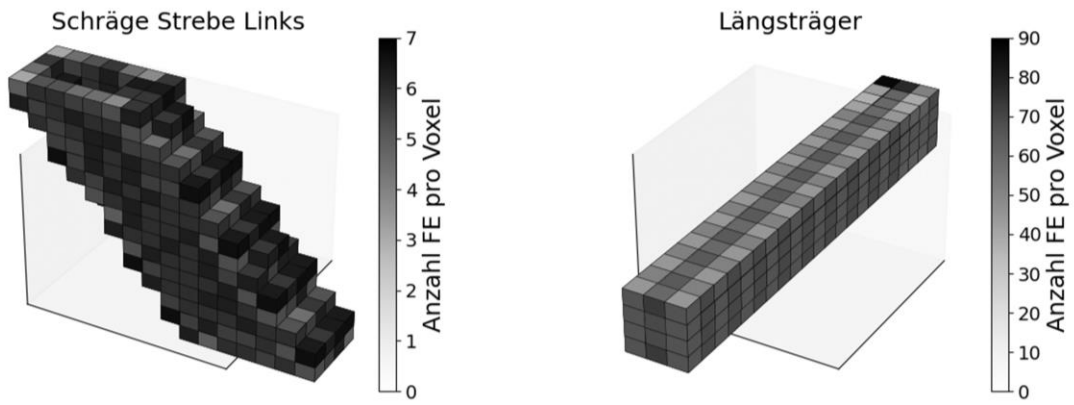


Abbildung 26: Ergebnisse der Voxel-Diskretisierung für die Bauteile „Schräge Strebe Links“ und „Längsträger“. In Graustufen dargestellt ist die Anzahl an FE pro Voxel

Neben dem Verzerrungseffekt durch die Bauteilform und den Ort der Deformation beeinflusst auch die Bauteilgröße die Ergebnisgüte des Sphären-Verfahrens. Dies ist beim Voxel-Verfahren nicht der Fall. Das Sphären-Verfahren bildet für ein konstantes K_S kleine Bauteile genauer ab als große, da weniger FE in denselben Pixeln landen und damit weniger Information über das Crashverhalten durch Bildung deren Mittelwerte verloren geht. Dies ist ebenfalls in Abbildung 25 zu sehen. Während beim *Längsträger*, der deutlich größer ist als die *Schräge Strebe Links*, bis zu 370 FE in einem Pixel landen, beträgt die maximale Anzahl bei der *Schrägen Strebe Links* 12. Somit müsste die Diskretisierung der Sphäre jeweils an die Größe des zu untersuchenden Bauteils

angepasst werden. Demgegenüber werden beim Voxel-Verfahren unterschiedlich große Bauteile bei demselben Wert für K_V mit demselben Detaillierungsgrad abgebildet, da die Anzahl der verwendeten Voxel entsprechend angepasst wird und damit der Informationsgehalt erhalten bleibt.

Zusammenfassend lässt sich festhalten, dass bei den meisten der untersuchten Bauteile das Voxel-Verfahren im Vergleich zum Sphären-Verfahren einen höheren Informationsgehalt pro N_{NNE} aufweist. Beim Sphären-Verfahren sorgt die Verzerrung, die von der Form des Bauteils und dem Ort der Deformation abhängt, dafür, dass verschiedene Bereiche des Bauteils unterschiedlich gut abgebildet werden. Darüber hinaus werden bei gleichen K_S kleine Bauteile besser repräsentiert als große. Daher sollte der Wert von K_S in Abhängigkeit der Bauteilgröße definiert werden, um sowohl kleine als auch große Bauteile hinreichend gut abzubilden. Das Voxel-Verfahren zeigt keinen Einfluss durch eine Verzerrung, da alle Bauteilbereiche gleichermaßen abgebildet werden. Damit ist der in den diskretisierten Daten enthaltene Informationsgehalt gleichzeitig unabhängig vom Bereich der Deformation. Darüber hinaus ist der Informationsgehalt bei konstantem K_V unabhängig von der Bauteilgröße. Kleine Bauteile werden durch weniger und große Bauteile durch mehr Voxel repräsentiert. Ein weiterer Vorteil ist, dass der Diskretisierungsparameter K_V vom Anwender intuitiv interpretiert werden kann. K_V legt die Größe der Voxel in der Einheit mm fest und ermöglicht dem Anwender damit abzuschätzen, wie viele FE pro Voxel ungefähr einsortiert werden. Eine solche intuitive Einschätzung über den Einfluss der Diskretisierung ist bei dem Sphären-Verfahren nicht gegeben.

5.3.3 Einfluss der Diskretisierung auf den Speicherplatz

Neben dem Informationsgehalt der diskretisierten Daten ist der benötigte Speicherplatz eine weitere Anforderung, die über die Eignung eines Verfahrens entscheidet. Da die Ergebnisse der Diskretisierung für jede Auswertungsgröße und für jedes Bauteil in separaten Dateien gespeichert werden, sollte der benötigte Speicherbedarf so gering wie möglich sein.

Um dem Leser eine bessere Einschätzung über die später aufgezeigten Speicherbedarfe der einzelnen Bauteile zu ermöglichen, folgt eine Überschlagsrechnung wie viel Speicherplatz ein Diskretisierungsverfahren für ein gesamtes Fahrzeug grundsätzlich benötigt.

Neben der plastischen Dehnung liegen weitere Auswertungsgrößen wie beispielsweise Verschiebungen (jeweils x,y,z Richtung), Beschleunigungen (jeweils x,y,z Richtung), Kräfte (jeweils x,y,z Richtung), Spannungen (jeweils x,y,z Richtung) und das Versagensverhalten der einzelnen FE (in diesem Fall insgesamt 14 verschiedene Auswertungsgrößen beziehungsweise Ergebnisdateien allein aus der Diskretisierung). Darüber hinaus wird angemerkt, dass die in den Untersuchungen exemplarisch verwendeten plastischen Dehnungen besonders effizient im CSR-Format abgespeichert werden können, da nicht jedes Element zu jedem Zeitschritt der Simulation plastisch verformt wird. Für die plastischen Dehnungen kann ein mittlerer Speicherbedarf pro Bauteil von 0,1 MB angenommen werden (siehe auch Abbildung 27). Dieser ist bei Größen wie beispielweise Verschiebungen deutlich höher, da diese für jedes Element zu jedem Zeitschritt der Simulation einen Wert aufweisen, sodass die Effizienz des CSR Formats sinkt und daher deutlich mehr Speicherplatz benötigt wird. Bei einem angenommenen mittleren Speicherbedarf pro Bauteil und pro Auswertungsgröße von 0,5 MB und der weiteren Annahme, dass das zu analysierende Fahrzeug aus etwa 500 Bauteilen besteht, ergibt sich demnach ein Speicherbedarf

für die diskretisierten Daten einer einzelnen Auswertungsgröße von 250 MB. Bei 14 Auswertungsgrößen beträgt der Speicherbedarf demnach 3,5GB. Die Ergebnisdateien aus der LS-Dyna Simulation weisen zwischen 2 und 3 GB auf, sodass der Speicherplatzbedarf durch die zusätzlichen diskretisierten Daten mehr als verdoppelt wird. Dies zieht folglich auch eine Verdopplung der Kosten für die abzuspeichernden Daten einer Simulation nach sich. Bei ca. 20000 Gesamtfahrzeugsimulationen pro Jahr ist dies ein nicht zu vernachlässigender Betrag.

Daher wird in diesem Abschnitt die Effizienz der beiden Verfahren hinsichtlich des Speicherbedarfs untersucht. Hierfür ist die Dateigröße in Abhängigkeit der Qualität der Ergebnisse in Abbildung 27 für die *Schottwand* und den *Längsträger* dargestellt. Für die Darstellung weiterer Bauteile wird auf den Anhang A verwiesen. So wird es möglich zu bewerten, welches Verfahren weniger Speicherplatz pro enthaltenem Informationsgehalt benötigt.

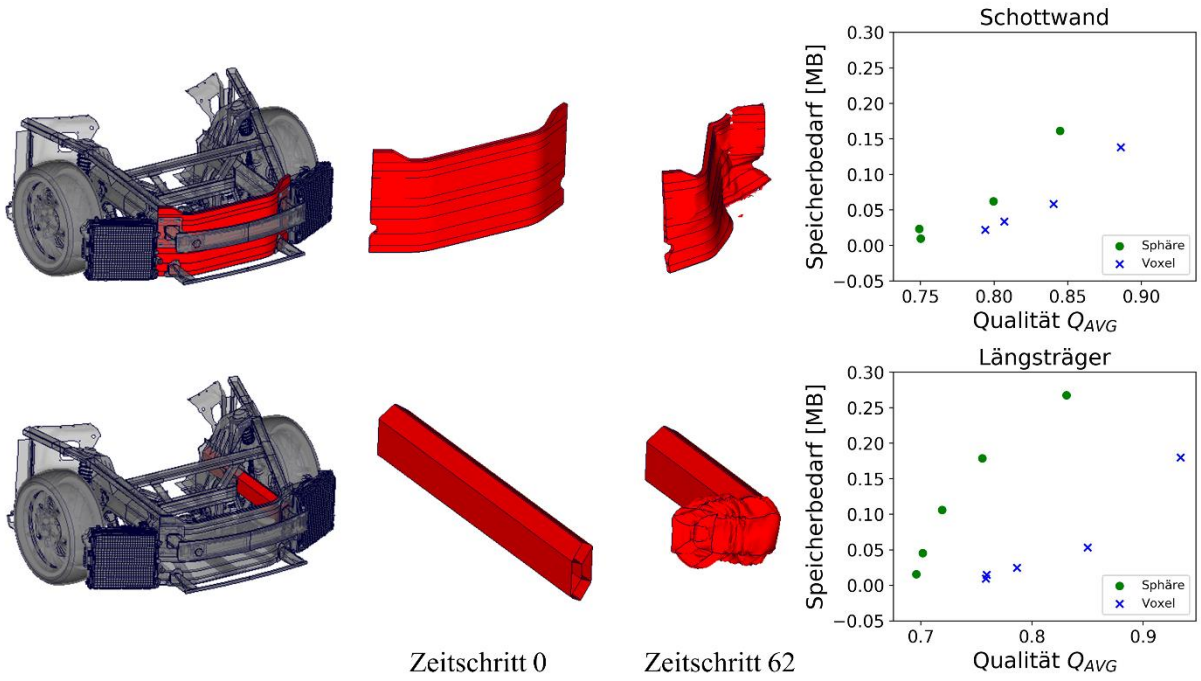


Abbildung 27: Darstellung des Speicherbedarfs in [MB] über dem in den diskretisierten Daten enthaltenen Informationsgehalt (repräsentiert durch die Qualität Q_{AVG}) exemplarische für die „Schottwand“ und den „Längsträger“

Bei den meisten Bauteilen ist der Speicherbedarf der Sphären-Daten höher als beim Voxel-Verfahren. Dabei ist der Unterschied beim *Längsträger* und der *Schottwand* am deutlichsten. Für den *Längsträger* ist der Speicherbedarf der Sphären-Daten bei einem Wert des Qualitätskriteriums von ca. 0,84 etwa fünfmal höher im Vergleich zu den Voxel-Daten. Bei der *Schottwand* und einer Qualität von 0,88 ergibt sich der Faktor 11 zwischen den beiden Verfahren.

Dies ist erneut auf die Verzerrung durch die sphärische Projektion zurückzuführen. Gerade bei langen profilförmigen Bauteile, bei denen das Crashverhalten insbesondere an den Enden auftritt, ist dieser Effekt am größten, da dort an den Enden die Häufungen der projizierten FE auftreten. Um denselben Informationsgehalt wie beim Voxel-Vefahren abzubilden, wäre eine sehr feine

Diskretisierung erforderlich, weshalb mehr Speicherplatz benötigt wird. Umgekehrt verhält es sich jedoch beim Voxel-Verfahren, da durch die gleichmäßigere Abbildung der originalen FE-Daten mehr Information pro Voxel enthalten ist. Dadurch werden weniger diskretisierte Elemente benötigt, was in einem geringeren Speicherbedarf resultiert. Aus diesem Grund ist auch hinsichtlich des Speicherbedarfs das Voxel-Verfahren besser geeignet.

5.4 Fazit

In den Analysen wird gezeigt, dass sich das Voxel-Verfahren für die einheitliche Datenrepräsentation besser eignet als das Sphären-Verfahren. Zum einen liegt ein höherer Informationsgehalt vor, da keine Verzerrungen der originalen FE-Daten auftreten. Zum anderen fällt der Speicherbedarf geringer aus, da bereits mehr Information je Voxel enthalten ist. Darüber hinaus sind die Ergebnisse für den Anwender intuitiver verständlich, da die dreidimensionale Struktur der Daten erhalten bleibt. Des Weiteren wird dem/der Ingenieur*in ermöglicht, die Auswirkungen des Diskretisierungsparameters K_V abzuschätzen, da dieser ebenfalls intuitiv interpretiert werden kann. Daher wird im Folgenden das Voxel-Verfahren als Methode für die Datenvorverarbeitung verwendet. Der Einfluss der Diskretisierung wird dabei weiterhin untersucht, indem die Analysen für unterschiedliche K_V durchgeführt werden.

6 Ausreißerdetektion

Nachdem die Daten durch die Diskretisierung in einem einheitlichen Format vorliegen, kann im Anschluss daran die automatisierte Ausreißerdetektion gestartet werden. Die Komplexität moderner Fahrzeuge und deren Crashverhalten erschwert eine vollumfängliche manuelle Analyse der Simulationsergebnisse. Der/Die Ingenieur*in kennt die für den jeweiligen Lastfall kritischen Bereiche im Fahrzeug und legt sein Augenmerk bei der Auswertung der Simulationen auf diese. Dabei besteht allerdings das Risiko, dass bedeutungsvolle Effekte in anderen Bereichen übersehen werden und demnach unzureichende Maßnahmen für die Verbesserung des Crashverhaltens getroffen werden. Zu solchen überraschend auftretenden und daher ggf. unentdeckten Effekten zählen beispielsweise ein Riss in einem Bauteil, in welchem bisher nie ein Riss aufgetreten ist oder ein axial belastetes Trägerprofil das ausknickt, während es bisher immer axial gestaucht wurde. Diese Situationen treten sehr vereinzelt auf, sodass ein solches Crashverhalten als Ausreißer bezeichnet wird. In diesem Kapitel wird eine Methode vorgestellt, mit deren Hilfe es möglich wird, auch derartig auffällige Bereiche im Fahrzeug vollautomatisiert zu identifizieren. Der/Die Ingenieur*in muss dafür lediglich eine Auswahl an Simulationen definieren in deren Kontext das Ausreißerverhalten analysiert werden soll. Dies können beispielsweise die letzten 20 sein.

6.1 Methode zur automatisierten Ausreißerdetektion in Crashsimulationen

6.1.1 Anforderungen an die Methode

Die bisherigen Anforderungen an die Ausreißerdetektion aus den Kapiteln 1.2 und 2.4 lassen sich wie folgt zusammenfassen:

- Ausreißerdetektion soll analysieren, wo und wann im Fahrzeug auffälliges Crashverhalten auftritt
- Als Ergebnis soll ein kontinuierlicher Ausreißerkennwert (AK) vorliegen
- Ergebnisse der Ausreißerdetektion sollen im Post-Prozessor darstellbar sein
- Anwender soll keine HP definieren müssen

Darüber hinaus ist der zeitliche Berechnungsaufwand der Ausreißerdetektion von Bedeutung. Die Ausreißerdetektion kann gemäß den Ausführungen in Kapitel 3 entweder in der automatisierten oder in der manuellen Auswertung durchgeführt werden. Letztere Möglichkeit erfordert, dass die Methode in Echtzeit funktioniert, damit der/die Ingenieur*in die Daten interaktiv analysieren kann und nicht auf die Ergebnisse warten muss. Echtzeit bedeutet im Kontext der beschriebenen Aufgabenstellung, dass beispielsweise während dem Einladen einer Simulation in eine Post-Prozessing Software wie den *Animator*, die Ausreißerkennwerte (AK) bereits berechnet werden können. Die Zeit für das Einladen eines Gesamtfahrzeug FE-Modells wird im Folgenden mit 120s angesetzt. Der Rohbau eines Fahrzeugs umfasst in etwa 500 Bauteile unterschiedlicher Abmessungen. Die folgende Überschlagsrechnung liefert die zeitliche Dauer, die ein Algorithmus zur Berechnung des AK für ein Bauteil durchschnittlicher Größe benötigen darf (Die

Untersuchungen in diesem Kapitel verwenden den *Längsträger* der unteren Lastebene aus Abbildung 21, Seite 58, dessen Größe als durchschnittlich bewertet wird.) Da die Ausreißerdetektion für die einzelnen Bauteile getrennt ausgeführt wird, werden die einzelnen AK parallelisiert mit mehreren Central Processing Units (CPU) berechnet. Für den Fall, dass beispielsweise wie in den vorliegenden Untersuchungen 16 CPU für die Berechnung der AK zur Verfügung stehen, sind

500 *Modulo* 16 = 32

Iterationen notwendig. Daraus resultiert ein maximaler Zeitbedarf von

120 s / 32 ≈ 4s

pro Bauteil, den der Algorithmus benötigen darf. Dieser Wert ist lediglich als grober Anhaltspunkt zu betrachten, da zum einen Streuungen in der Anzahl der Bauteile eines Fahrzeugs sowie in der Einladezeit in den Post-Prozessor auftreten. Zum anderen ist nicht bekannt, wie effizient die Implementierung der Algorithmen aus *PyOd* umgesetzt ist. Daher werden lediglich die Algorithmen in den folgenden Analysen berücksichtigt, deren Berechnungszeit unter einer Minute liegen. Auch mit zunehmender Größe des Datensets soll die Skalierbarkeit des Verfahrens noch gewährleistet sein, da in der praktischen Anwendung unterschiedlich große Datensets vorliegen können. Hierauf ist bei der Auswahl eines geeigneten Algorithmus ebenfalls zu achten.

Für den Fall, dass in den Daten ein einzelner starker Ausreißer auftritt, ist die Wahrscheinlichkeit hoch, dass dieser vom Algorithmus gefunden wird und einen hohen Ausreißerkennwert nahe eins erhält, da sich dessen Eigenschaften deutlich von denen der unauffälligen Simulationen unterscheiden. Steigt jedoch der Anteil der Ausreißer Simulationen im vorliegenden Datenset an, wird es wie auch für den Menschen bei einer manuellen Analyse, auch für den Algorithmus zunehmend schwieriger, diese als Ausreißer zu identifizieren. Ab einem gewissen Anteil, kann nicht mehr von einem Ausreißer, sondern muss womöglich von einem völlig anderen Crashverhalten gesprochen werden. Dass die Ausreißerkennwerte mit zunehmendem Ausreißeranteil abnehmen, ist daher eine geduldete Eigenschaft. Die Methode sollte jedoch robust gegenüber kleinen Schwankungen des Ausreißeranteils sein (Kracker et al. 2023).

6.1.2 Ablauf der neuen automatisierten Ausreißerdetektion

Abbildung 28 zeigt die einzelnen Schritte der Ausreißerdetektion. Zunächst definiert der Anwender, welche Simulationen im gemeinsamen Kontext analysiert werden sollen. Damit der/die Ingenieur*in von der Ausreißerdetektion Informationen dazu erhält, wo und wann im Fahrzeug ein auffälliges Crashverhalten auftritt, werden jedes Bauteil und jeder Zeitschritt getrennt auf potenzielles Ausreißerverhalten untersucht. Somit stehen detaillierte Informationen über die gesamte Dauer des Crashs zur Verfügung und Bifurkationspunkte können identifiziert werden. Ein weiterer Vorteil dieses Vorgehens ist die Möglichkeit, die Berechnungen parallelisieren zu können. Bei entsprechend vorhandener Hardware kann dadurch die Berechnungszeit zusätzlich massiv reduziert und ein interaktives Arbeiten mit den Daten ermöglicht werden.

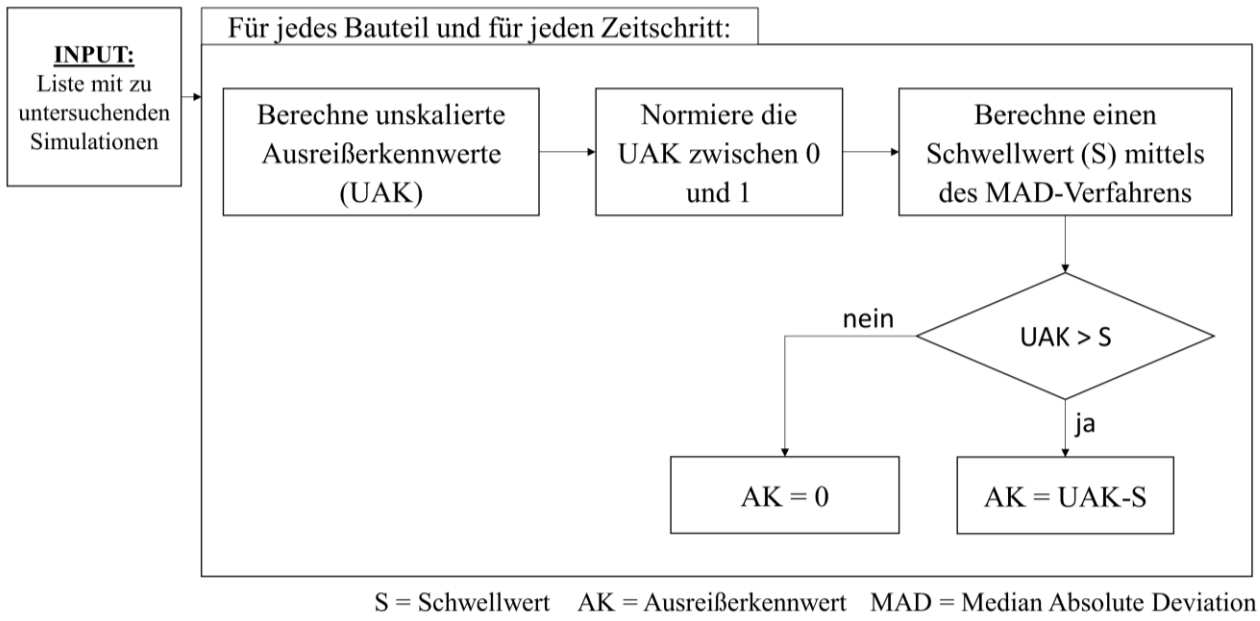


Abbildung 28: Schematischer Ablauf der Ausreißerdetektion und Berechnung der Ausreißerkennwerte (AK)

Die Berechnung des Ausreißerkennwerts (AK) läuft für jedes Bauteil und jeden Zeitschritt wie folgt ab:

Zunächst wird der *unskalierte Ausreißerkennwert* (UAK) berechnet, der die unskalierte Auffälligkeit eines Datenpunkts beschreibt. Dies erfolgt mittels unüberwachter Lernverfahren, da eine Anforderung ist, dass der Anwender keinen weiteren Input wie beispielsweise kategorisierte Daten zur Verfügung stellen muss. Die Formeln zur Berechnung des UAK sind abhängig vom jeweiligen Algorithmus. Für die in dieser Arbeit betrachteten Algorithmen wird auf die Formeln 2.2 für PCAD, 2.3 für KNND, 2.4 für LOF und 2.9 für SOS verwiesen.

Zudem existiert die Anforderung, dass Bauteile gemäß ihrer Auffälligkeit im Post-Prozessor eingefärbt werden können. Um dem Anwender diese Abstufung zu ermöglichen, soll ein kontinuierlicher Ausreißerkennwert berechnet werden, dessen Wertebereich zwischen 0 und 1 liegt. 0 bedeutet hierbei, dass keine Anzeichen eines abweichenden Verhaltens vorliegen, während der Wert 1 auf ein eindeutiges Ausreißerverhalten hindeutet.

Je nach Art des Algorithmus (statistisch, distanzbasiert, dichtebasiert siehe Kapitel 2.4) unterscheiden sich die Wertebereiche des resultierenden UAK teilweise deutlich und liegen in unterschiedlichen Größenordnungen. Zudem variieren die Wertebereiche des UAK für verschiedene Hyperparameter selbst innerhalb eines Algorithmus. Gerade bei distanzbasierten Verfahren wird dies besonders deutlich. Entspricht der Kennwerte eines Datenpunkts beispielsweise der Distanz zu dessen k-tem Nachbarn, unterscheiden sich die Ergebnisse in Abhängigkeit davon, ob der dritte oder beispielsweise der 30ste Nachbar betrachtet wird. Darüber hinaus sind die Wertebereiche selbst innerhalb eines Algorithmus und Hyperparameters von Bauteil zu Bauteil verschieden. Dies lässt sich erneut am Beispiel distanzbasierter Verfahren erläutern. Der Wert des UAK soll der Distanz zum k-ten Nachbarn entsprechen. Diese hängt stark

von der Größe und der Stärke der Deformation eines Bauteils ab. Große Bauteile mit starker Deformation werden höhere AK aufweisen als kleine Bauteile, die kaum deformieren.

Aus diesem Grund werden im nächsten Schritt die Werte des UAK zwischen 0 und 1 normiert. Dadurch wird sichergestellt, dass der AK unabhängig vom gewählten Algorithmus, dessen Hyperparametern, dem betrachteten Bauteil und dessen Größe, als auch Deformationsverhalten ist.

Durch diese Normierung ergibt sich jedoch das Problem, dass für jedes Bauteil und für jeden Zeitschritt mindestens eine Simulation den Wert eins aufweist und dies unabhängig davon, ob ein auffälliges Verhalten vorliegt oder nicht (siehe Abbildung 29 links). Daher besitzt das bis hierhin vorliegende Ergebnis noch keine Aussagekraft darüber, ob eine Simulation ein Ausreißerverhalten zeigt oder nicht. Die Problemformulierung hat sich demnach von der multivariaten zur univariaten Ausreißerdetektion verschoben.

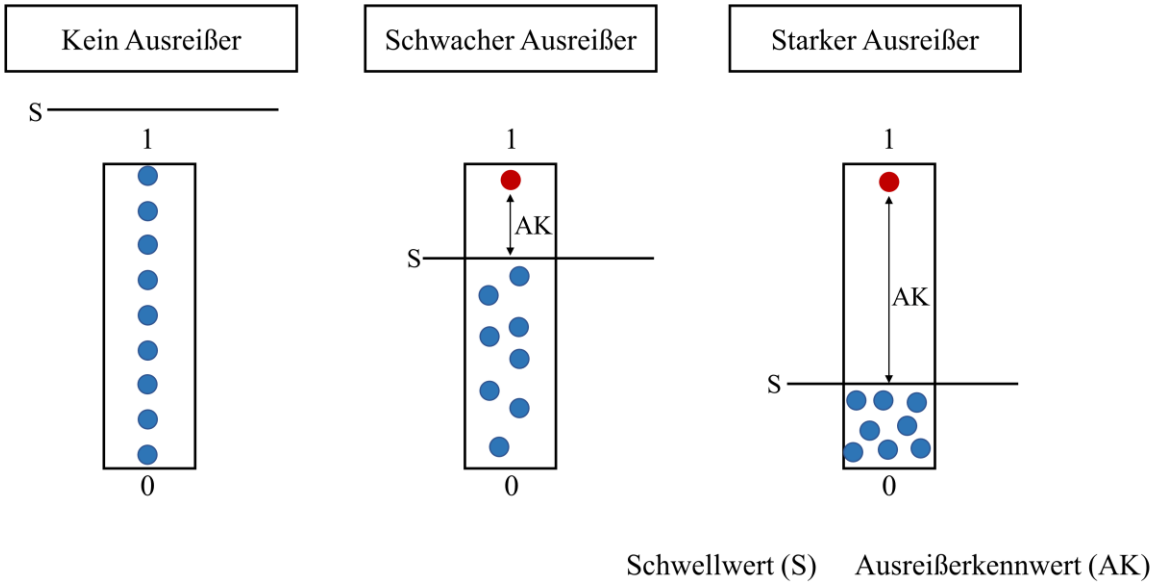


Abbildung 29: Beispielhafte Darstellung des Zusammenhangs zwischen den skalierten Ausreißerkennwerten (blaue bzw. rote Punkte), dem Schwellwert S und den finalen Ausreißerkennwerten

Mit statistischen Verfahren lässt sich jedoch aus der Verteilung des UAK ein Schwellwert berechnen, mittels welchem vorhandene Ausreißer in den univariaten Daten identifiziert werden können. Für die Berechnung dieses Schwellwerts wird das MAD-Verfahren verwendet (siehe Kapitel 2.4.2.5, Formel 2.12), da dieses auf der Berechnung des Medians basiert. Dieser wird im Vergleich zur Standardabweichung oder dem Interquartilsabstand kaum durch vorhandene Ausreißer beeinflusst und liefert somit robustere Ergebnisse. Ein Nachteil ist, dass der Hyperparameter *t* definiert werden muss, mit dem der Schwellwert ab dem Datenpunkte als Ausreißer klassifiziert werden beeinflusst wird. Zu große Werte von t sorgen dafür, dass Ausreißer aufgrund des resultierenden höheren Schwellwerts schwieriger identifiziert werden können. Bei zu kleinen Werten steigt das Risiko, dass unauffällige Simulationen

fälschlicherweise als Ausreißer klassifiziert werden. Im nächsten Kapitel wird daher der Einfluss des Hyperparameters *t* auf die Ergebnisse analysiert, da das Ziel der Analysen dieser Arbeit ist, einen robusten Wert für die Methode festzulegen und dem Anwender damit die Entscheidung hierüber abnehmen zu können.

Simulationen, bei denen der Wert des UAK kleiner ist als der berechnete Schwellwert weisen kein auffälliges Verhalten auf. Deren AK wird auf 0 gesetzt. Für den Fall, dass der UAK größer als der Schwellwert ausfällt, gibt dies einen Hinweis für einen vorhandenen Ausreißer in den Daten (Abbildung 29 Mitte und rechts). Würde an dieser Stelle der Wert des UAK als finaler AK verwendet werden, würde dies zu einer falschen Interpretation der Ergebnisse führen. Dies verdeutlicht das Beispiel in der Mitte der Abbildung 29. Dort ist in den Daten ein schwacher Ausreißer vorhanden. Dessen UAK beträgt allerdings aufgrund der vorgenommenen Normierung den Wert 1, weshalb der Anwender auch ein stark abweichendes Crashverhalten in der Simulation vermuten könnte. Die rechte Darstellung zeigt einen solchen starken Ausreißer, der denselben AK erhalten würde. Aus diesem Grund wird für Dateninstanzen, bei denen der UAK größer als der Schwellwert ist, jeweils die Differenz zwischen dem UAK und Schwellwert als finaler AK verwendet.

Damit berechnet sich der AK zu

$$\boldsymbol{AK} = \begin{cases} \mathbf{0}, & \boldsymbol{wenn\ UAK < S} \\ \boldsymbol{UAK - S}, & \boldsymbol{wenn\ UAK > S}\,. \end{cases} \qquad \textbf{(6.1)}$$

Der/Die Ingenieur*in soll keinen weiteren Input außer die zu analysierenden Simulationen zur Verfügung stellen müssen. Die meisten Algorithmen zur Berechnung des UAK erfordern jedoch die Definition von Hyperparametern.

In Kapitel 6.3.2 werden daher verschiedene Algorithmen und der Einfluss deren Hyperparameter auf die AK untersucht. Ziel ist es, Heuristiken abzuleiten, die die Wahl eines konkreten oder mehrerer Werte erlauben und dabei gleichzeitig robuste Ergebnisse liefern. Sollte es nicht möglich sein, einen konkreten Wert für die Hyperparameter festzulegen, bietet sich die Verwendung eines Ensemble Modells an. Die Idee dahinter ist, mehrere Modelle mit verschiedenen Hyperparametern zu trainieren und die resultierenden Ergebnisse zu einem einzelnen Wert zu kombinieren. Hierzu werden die Berechnungsschritte aus Abbildung 28 (Seite 73) für verschiedene Hyperparameter wiederholt und die resultierenden AK gemittelt. Durch dieses Vorgehen kann die Robustheit der Methode gesteigert und dem Anwender gleichzeitig die Definition eines Hyperparameters abgenommen werden.

Als Ergebnis liegt für jedes Bauteil und jeden Zeitschritt der Simulation ein Ausreißerkennwert (AK) vor, dessen Wertebereich zwischen null und eins liegt. Desto auffälliger ein Bauteil hervorsticht, desto näher liegt der AK am Wert eins. Unauffällige Bauteile erhalten niedrige Kennwerte nahe null. Die Ergebnisse können im *Animator* visualisiert werden, indem anhand des Wertes des AK eine entsprechende Einfärbung der Bauteile erfolgt (siehe Abbildung 4, Seite 5). Dadurch stechen auffällige Bereiche farblich sofort hervor und zeigen dem/der Ingenieur*in damit übersichtlich die kritischen Stellen im Fahrzeug an. Gleichzeitig ist es ihm möglich den konkreten Zeitpunkt des auffälligen Verhaltens zu bestimmen, da für jeden Zeitschritt ein AK vorliegt. Je nachdem welche Auswertungsgröße aus der Simulation ausgewählt wird, können

Ausreißer beispielsweise aufgrund deren Deformations- (plastische Dehnungen) oder Kinematikverhalten (Verschiebungen) identifiziert werden.

Dadurch, dass die Berechnung der AK vollautomatisiert abläuft, ist es nunmehr möglich sämtliche Bauteile und Auswertungsgrößen aus der Simulation auszuwerten. Somit wird die gesamte Auswertung gegenüber dem bisherigen Stand, bei dem bislang nur erfahrungsbasiert vorgegangen werden konnte, vollumfänglicher. Darüber hinaus wird verhindert, dass bestimmte Effekte innerhalb der Analyse übersehen werden, was letztlich eine robustere Auswertung ermöglicht. Da nunmehr der Fokus des Ingenieurs auf die relevanten Stellen im Fahrzeug gelegt wird, ist die Auswertung gleichzeitig fokussierter.

6.2 Durchführung der Ausreißerdetektion mit konkreten Daten

In diesem Kapitel wird der *Längsträger* der unteren Lastebene aus Abbildung 21 (Seite 58) verwendet, um die Einflüsse der verschiedenen Algorithmen auf den Ausreißerkennwert (AK) zu untersuchen. Es werden die in Kapitel 4 vorgestellten 50 Simulationen aus dem Datenset DIN001 verwendet. In 26 Simulationen beginnt die initiale Deformation des Bauteils im vorderen Bereich, während sie bei 12 weiteren Simulationen zuerst im hinteren Teil auftritt. Bei den übrigen 12 Simulationen wird ein Mischverhalten aus den beiden beschriebenen Deformationsmodi beobachtet (siehe Abbildung 21, Seite 58). Um die Interpretation der Ergebnisse zu vereinfachen, werden letztere Simulationen aus den Betrachtungen dieses Kapitels ausgeschlossen. Die Simulationen, in denen der vordere Bereich zuerst deformiert, zeigen das unauffällige Crashverhalten, während jene, bei denen die initiale Deformation im hinteren Bereich beginnt, die Ausreißer darstellen.

Desto höher der Ausreißeranteil in einem Datenset ist, desto schwieriger ist es sowohl für den/die Ingenieur*in als auch für einen Algorithmus, Auffälligkeiten zu identifizieren. Aus diesem Grund werden in den folgenden Analysen unterschiedlich schwierige Datensets verwendet, um die Eigenschaften der Verfahren unter den verschiedensten Szenarien, die in der praktischen Anwendung auftreten können, zu bewerten. Die ersten 26 Simulationen jeden Datensets entsprechen den unauffälligen, in denen die initiale Deformation im vorderen Bereich des Bauteils auftritt. Darüber hinaus ist im ersten Datenset eine Ausreißer Simulation vorhanden, was einem Ausreißeranteil von etwa 4% entspricht. Im zweiten und dritten Datenset sind zusätzlich je 5 bzw. 12 auffällige Simulationen enthalten. Dies gleicht einem Ausreißeranteil von circa 16 bzw. 32%. Das dritte Datenset beinhaltet somit die meisten Ausreißer und stellt damit die größte Herausforderung für die Methode zur Ausreißerdetektion dar. Die Zusammensetzung der drei Datensets ist in Tabelle 3 zusammenfassend dargestellt.

Tabelle 3: Zusammensetzung der drei in den Auswertungen verwendeten Datensets mit unterschiedlichem Anteil an Ausreißern (Kracker et al. 2023)

	Anzahl unauffälliger Simulationen	Anzahl Ausreißer Simulationen ($N_{Ausreißer}$)	Ausreißeranteil [%]
Datenset 1	26	1	3,70
Datenset 2	26	5	16,13
Datenset 3	26	12	31,58

Um den Einfluss der Diskretisierung zu bewerten, werden zunächst in den Kapiteln 6.3 und 6.4 die undiskretisierten FE-Daten und im Anschluss daran in Kapitel 6.5 die diskretisierten Daten verwendet.

6.3 Vergleich von Algorithmen zur Berechnung des unskalierten Ausreißerkennwerts

In diesem Abschnitt werden verschiedene Algorithmen zur Berechnung des UAK verglichen, um diejenigen zu identifizieren, die sich für die Ausreißerdetektion in Crashsimulationen besonders eignen. Neben der Fähigkeit Ausreißer zuverlässig zu identifizieren, werden der Hyperparametereinfluss und der zeitliche Berechnungsaufwand bewertet.

6.3.1 Vorgehensweise

In Abbildung 30 sind die AK aus dem LOF-Algorithmus exemplarisch für jeweils eine unauffällige und eine auffällige Simulation dargestellt. Für die unauffällige Simulation weist der AK über die gesamte Dauer des Crashs einen niedrigen Wert nahe null auf, der dem Anwender keinen Anlass bietet, diese Simulation als Ausreißer zu betrachten. Bei der auffälligen Simulation beträgt der AK für die ersten 21 Zeitschritte exakt null. In diesem Zeitraum treten zunächst keine wesentlichen Deformationen in dem betrachteten Bauteil auf, wodurch sich die einzelnen Simulationen auch nicht unterscheiden und demnach auch kein abweichendes Crashverhalten identifiziert wird. Zwischen Zeitschritt 22 und 27 wird der höchste Wert nahe eins erreicht. In diesem Zeitbereich ist die Auffälligkeit am größten, da sich das Anfaltverhalten am deutlichsten unterscheidet. Nach einem anfänglichen Absinken des AK steigt dieser zum Zeitschritt 30 wieder an und verhält sich bis zum Ende des Crashs annähernd konstant auf einem hohen Niveau.

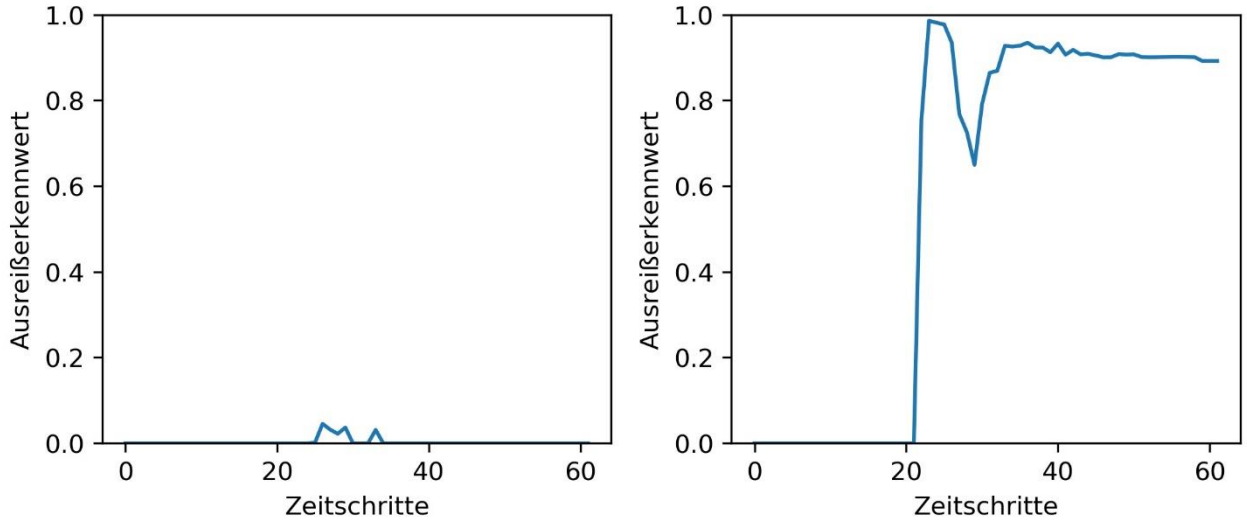


Abbildung 30: Darstellung des Ausreißerkennwerts exemplarisch aus dem LOF Algorithmus in Abhängigkeit der Zeit für eine unauffällige (links) und eine Ausreißer Simulation (rechts)

Jeden Zeitschritt in den nachfolgenden Analysen einzeln zu betrachten wäre für den Leser unübersichtlich. Gleichzeitig ist die Auswertung basierend auf lediglich einem einzigen Zeitschritt zu sehr eingeschränkt. Aus diesem Grund wird der Mittelwert aus den AK der einzelnen Zeitschritte berechnet. Daraus ergibt sich für jede Simulation ein einzelner Ausreißerkennwert (Im Beispiel aus Abbildung 30 beträgt er 0,002 für die unauffällige und 0,575 für die Ausreißer Simulation). Dadurch, dass die Auffälligkeit der Simulationen nun nur noch durch einen AK beschrieben wird, können Effekte, die beispielsweise durch Hyperparameter oder den Ausreißeranteil hervorgerufen werden, anschaulicher dargestellt werden.

In einer Vorabstudie wurden sämtliche Verfahren aus PyOD hinsichtlich deren Fähigkeit untersucht, die vorhandenen Ausreißer in den drei Beispieldatensets zu identifizieren. Lediglich jene Algorithmen, die hierzu in der Lage sind, werden in den weiteren Analysen berücksichtigt. Zu diesen zählen die aus Kapitel 2.4.2 bekannten

- Principle Component Analysis Distanz (PCAD)
- k-nächste Nachbarn Distanz (KNND)
- Local Outlier Factor (LOF)
- Stochastic Outlier Selection (SOS)

Diese werden in den folgenden Abschnitten auf verschiedene Eigenschaften hin untersucht. Zunächst wird der Einfluss durch eventuell vorhandene Hyperparameter analysiert. Dabei wird insbesondere untersucht, ob sich ein Wertebereich festlegen lässt, der im Allgemeinen zu robusten Ergebnissen führt. Im Anschluss daran erfolgt die Evaluierung der Abhängigkeit der AK vom Ausreißeranteil. Daraufhin wird die zeitliche Skalierbarkeit untersucht und bewertet inwiefern sich die Algorithmen für eine interaktive Analyse in Echtzeit eignen.

Für sämtliche Untersuchungen in den Kapiteln 6.3.2 - 6.3.4 wird der Parameter t des MAD-Verfahrens auf den Wert zwei festgelegt. Dessen Einfluss auf den AK wird anschließend in Kapitel 6.4 untersucht. In Kapitel 6.5 folgt die Analyse, inwiefern verschiedene Diskretisierungen K_V die Ergebnisse der Ausreißerdetektion beeinflussen.

6.3.2 Einfluss der Hyperparameter

Da der Anwender nicht gezwungen sein soll Werte für den HP vorgeben zu müssen, ist ein Algorithmus besonders geeignet, wenn er entweder keine HP aufweist oder deren Einfluss auf die Ausreißerdetektion klein ist, sodass ein Standardwert für den HP oder ein Ensemble Modell verwendet werden kann. Mit Ausnahme der PCAD, bei der es sich um ein parameterfreies Verfahren handelt, verfügen die drei anderen Algorithmen jeweils über einen Hyperparameter. Beim LOF- und KNND-Algorithmus wird die Größe der zu betrachtenden Nachbarschaft durch den HP *k* definiert. Beim SOS-Algorithmus wird die Perplexität *p* festgelegt. Die HP der unterschiedlichen Verfahren haben einen gemeinsamen Wertebereich, der zwischen 1 und N-1 liegt, wobei *N* der Anzahl der Simulation entspricht.

Der Einfluss der HP auf die Ausreißerdetektion wird im Folgenden bewertet, indem die Streuung des Ausreißerkennwerts für alle möglichen Werte des jeweiligen HP analysiert wird. Abbildung 31 stellt die Ergebnisse der Übersichtlichkeit halber für jeweils fünf unauffällige (linke Spalte) und fünf Ausreißer (rechte Spalte) Simulationen exemplarisch aus dem zweiten Datenset in Form von Verteilungs- und Boxplots dar.

Die unauffälligen Simulationen (linke Spalte) werden von sämtlichen Algorithmen mit einem niedrigen AK nahe null bewertet. Dabei fällt auf, dass die PCAD und KNND die geringsten Streuungen zeigen. Da es sich bei der PCAD um ein parameterfreies Verfahren handelt, existiert lediglich ein einziger Ausreißerkennwert. Die SOS und LOF Verfahren liefern dagegen größere Streuungen. Gerade für die Simulation drei liegen deren AK für einen ungünstig gewählten Hyperparameter nahe 0,3, was den Anwender fälschlicherweise dazu veranlassen könnte, von einem auffälligen Crashverhalten auszugehen. Dies kann bei der Auswertung hinderlich sein und von der Analyse der eigentlich relevanten Stellen in der Crashsimulation ablenken.

In der rechten Spalte sind die AK der fünf auffälligen Simulationen abgebildet. Da die PCAD ein parameterfreies Verfahren ist, liegt erneut für jede Simulation ein jeweils konstanter Wert vor. Dieser liegt bei den fünf Ausreißer Simulationen zwischen 0,2 und 0,3. Bei den übrigen drei Algorithmen sind erneut Streuungen zu beobachten, die im Vergleich zu den unauffälligen Simulationen größer ausfallen. Das SOS-Verfahren liefert insgesamt die niedrigsten Werte, während LOF die höchsten aufweist. Die PCAD und KNND liegen dazwischen. Darüber hinaus ist ersichtlich, dass der Hyperparametereinfluss bei LOF am größten ist, gefolgt von KNND und SOS. Einzelne Hyperparameter liefern dabei zu niedrige Ausreißerkennwerte bei auffälligen Simulationen, wodurch das Risiko besteht, Ausreißer zu übersehen.

Für auffällige Simulationen sollte ein möglichst hoher und für unauffällige ein möglichst niedriger AK vorliegen. Die Abbildung zeigt, dass die Hyperparameter hierauf einen direkten Einfluss haben und ungünstige Werte sich negativ auswirken können.

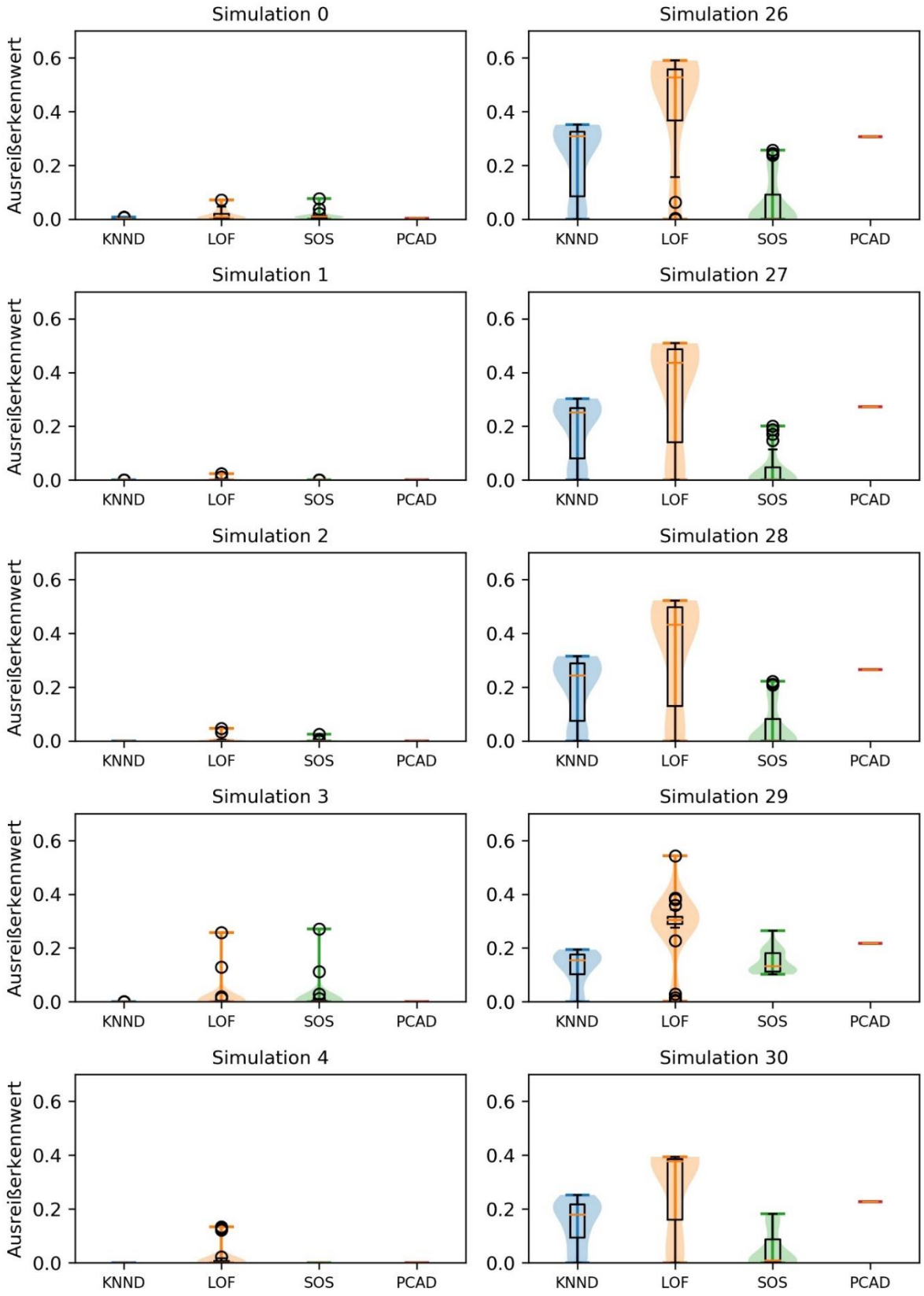


Abbildung 31: Farbliche Darstellung der Verteilung der Ausreißerkennwerte aus den Algorithmen KNND, LOF und SOS für verschiedene Hyperparameter (Größe der zu berücksichtigenden Nachbarschaft k bzw. Perplexität p im Wertebereich zwischen 1 und 30). Da die PCAD keine Hyperparameter enthält, liegt hier nur ein einzelner Datenpunkt vor. Der Boxplot zeigt den Median, das 1. und 3. Quartil. Einzelne Ausreißer sind als Kreisringe dargestellt. Links exemplarisch für die unauffälligen, rechts für die Ausreißer Simulationen.

Um den Hyperparametereinfluss im Folgenden zu quantifizieren, werden der Mittelwert und die Standardabweichung des AK für jede Simulation und jeden Algorithmus berechnet und in Abbildung 32 links beziehungsweise rechts exemplarisch am zweiten Datenset visualisiert. Auf der horizontalen Achse sind die einzelnen Simulationen dargestellt, während der Mittelwert (links) beziehungsweise die Standardabweichung (rechts) deren AK auf der vertikalen Achse aufgetragen ist. Aus Gründen der Darstellbarkeit werden die Punkte von jedem Algorithmus jeweils durch Linien miteinander verbunden.

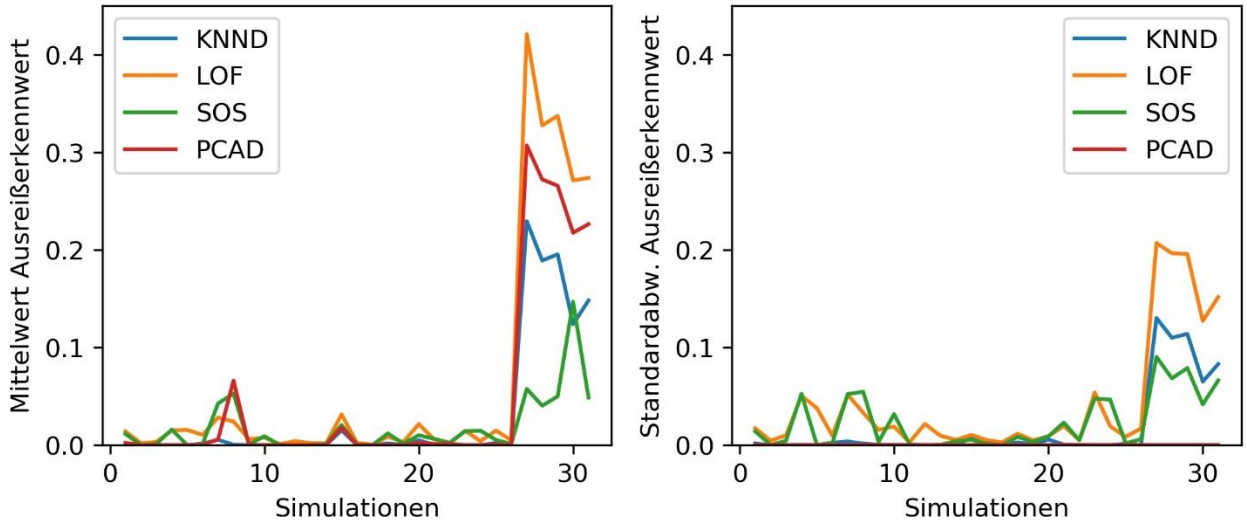


Abbildung 32: Mittelwert (links) beziehungsweise Standardabweichung (rechts) des Ausreißerkennwerts über sämtliche Hyperparameter des jeweiligen Algorithmus dargestellt für jede einzelne Simulation aus Datenset 2.

Die 26 unauffälligen Simulationen weisen stets einen niedrigen Mittelwert des AK auf, der keinen Anlass dazu gibt, ein auffälliges Crashverhalten zu vermuten. Für die letzten fünf Ausreißer werden bei jedem Algorithmus dagegen höhere Werte ausgegeben. LOF, PCAD und KNND liefern dabei deutliche Anzeichen für ein auffälliges Crashverhalten, indem die AK meist über 0,2 liegen, während die Ergebnisse von SOS durch niedrigere AK<0,2 deutlich schlechter ausfallen. Dadurch besteht das Risiko, dass der Anwender auffälliges Crashverhalten übersieht.

Bei der Betrachtung der Standardabweichung wird ersichtlich, dass diese für sämtliche unauffällige Simulationen bei KNND nahezu null beträgt. LOF und SOS weisen dagegen ähnliche Werte zwischen 0 und 0,06 auf. Bei den auffälligen Simulationen ist die Standardabweichung für LOF am größten, während KNND in der Mitte liegt und SOS die geringste Streuung zeigt. Für die PCA beträgt sie null, da kein Hyperparameter vorliegt. Insgesamt ist die Standardabweichung für alle drei Algorithmen mit Hyperparametern im Vergleich zu den jeweiligen Mittelwerten groß, sodass bei keinem Algorithmus der Einfluss dessen Hyperparameters vernachlässigt werden kann. Da sämtliche Ausreißer deutlich niedrigere AK als 1 aufweisen (obwohl sie sich deutlich unterscheiden), wären die Ergebnisse eines Ensemble Modells mit allen Hyperparametern aus dem Wertebereich zwischen 1 und N-1 unzureichend. Durch eine Eingrenzung des Wertebereichs könnte die Ergebnisgüte eines Ensemble Modells verbessert werden. Um also die Anforderung zu erfüllen, dass der Anwender keinen Hyperparameter definieren soll, wird im Folgenden untersucht, inwiefern ein sinnvoller Wert beziehungsweise Wertebereich für den jeweiligen Hyperparameter angegeben werden kann.

6.3.2.1 Einfluss des Hyperparameters *k* beim LOF-Algorithmus

Zunächst wird der Einfluss des Hyperparameters *k* für den *LOF*-Algorithmus untersucht. Der Übersichtlichkeit halber werden die AK in Abbildung 33 links für die Simulation 1 (unauffällig) und rechts für die Simulation 26 (Ausreißer) dargestellt.

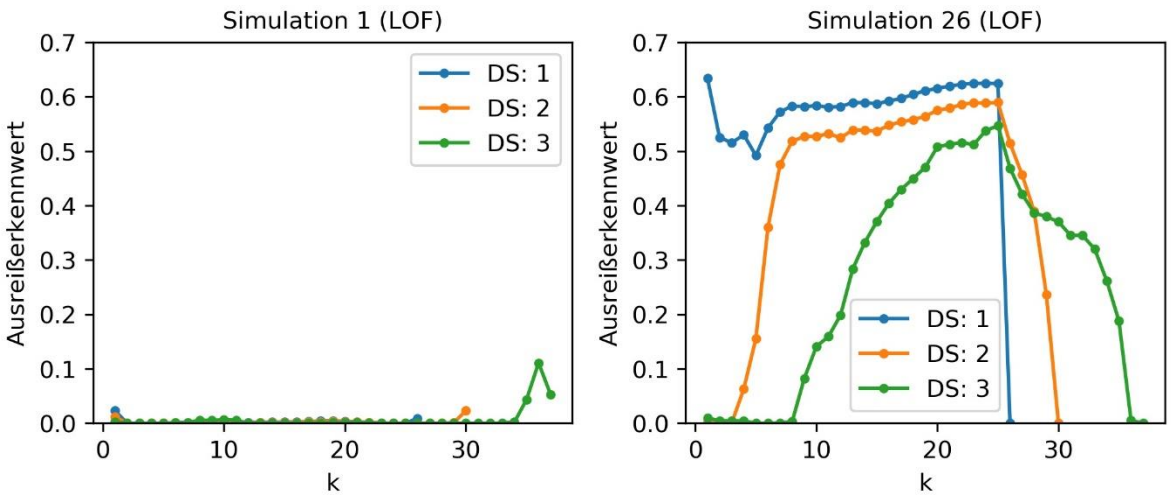


Abbildung 33: Verlauf des Ausreißerkennwerts für den LOF Algorithmus in Abhängigkeit des Hyperparameters k dargestellt für die drei Datensets (DS). Links für die unauffällige Simulation 1, rechts für die Ausreißer Simulation 26

Bei der unauffälligen Simulation weisen die AK für jedes der drei Datensets und über den gesamten Wertebereich von *k* hinweg niedrige Werte auf. Dies deckt sich mit den Erkenntnissen aus dem vorangegangenen Kapitel, wonach für die unauffälligen Simulationen geringe Streuungen im Zusammenhang mit variierenden Hyperparameterwerten einhergehen. Damit ist die Anforderung an die Methode erfüllt, dass unauffälliges Crashverhalten mit einem niedrigen AK bewertet werden soll.

Für die auffällige Simulation kann dagegen ein starker Einfluss von *k* über alle drei Datensets hinweg beobachtet werden. Für das Datenset 1 mit nur einer enthaltenen auffälligen Simulation treten die höchsten Werte des AK auf. Dies liegt daran, dass im Vergleich zu den anderen Datensets der geringste Ausreißeranteil vorliegt und damit das Crashverhalten der auffälligen Simulation am deutlichsten hervortritt. Der Einfluss des Ausreißeranteils wird im nächsten Kapitel genauer beleuchtet. Für steigende Werte von k fällt der AK zunächst auf den Wert 0,49 ab, bevor er ab k = 6 wieder ansteigt und für k = 25 den Wert 0,62 erreicht. Für k = 26 beträgt der AK null.

Die Berechnung des *Unskalierten Ausreißerkennwerts* (UAK) basiert beim *LOF*-Algorithmus auf lokalen Dichten in den zu analysierenden Daten. Nach Gleichung 2.4 wird hierfür die Erreichbarkeits-Distanz verwendet. Bei Datenset 1 weist sie deshalb für k=1 einen großen Wert auf, da die Ausreißer Simulation im Kontext der 26 unauffälligen Simulationen über eine große Distanz zu deren erstem Nachbarn verfügt. Dementsprechend liegt eine kleine lokale Dichte für die auffällige Simulation vor, was nach Gleichung 2.6 zu einem großen UAK führt. Gleichzeitig

sind die lokalen Dichten für die unauffälligen Simulationen größer, sodass deren UAK geringer ist. Dies führt dazu, dass der Schwellwert niedrig und dementsprechend die Differenz zwischen dem UAK für die Ausreißer Simulation und dem Schwellwert groß ist, was zu einem hohen AK führt.

Der AK fällt für k = 26 auf null ab, da in diesem Fall sowohl für die unauffälligen als auch für die Ausreißer Simulationen die lokalen Dichten in einem ähnlichen Wertebereich liegen. Für jede unauffällige Simulation entspricht der 26. Nachbar jeweils der auffälligen Simulation, was einen hohen Wert für die Erreichbarkeits-Distanz zur Folge hat. Andersherum ist der 26. Nachbar für die auffällige Simulation stets eine unauffällige. Dadurch sind die Werte der Erreichbarkeits-Distanz und damit des UAK stets ähnlich für alle Simulationen, was dazu führt, dass sämtliche UAK nach Gleichung 6.1 kleiner als der Schwellwert sind und damit der AK null beträgt. Für das zweite und dritte Datenset sind jeweils ähnliche Effekte jedoch mit größerer Wirkung festzustellen. Für alle Datensets gibt es Wertebereiche des Hyperparameters, die hinreichend hohe AK liefern, um auffälliges Crashverhalten zuverlässig zu identifizieren.

Für $k < N_{Ausreißer}$ liegen kleine AK vor. Dies liegt daran, dass innerhalb der entsprechend zu berücksichtigenden Nachbarschaften die Dichten für unauffällige und auffällige Simulationen jeweils ähnlich sind. Wird durch $k < N_{Ausreißer}$ jeweils nur der Kontext betrachtet, in dem gleiches Crashverhalten vorliegt, unterscheiden sich die UAK nicht, sodass der Schwellwert stets größer ist als der UAK und dementsprechend auch der AK null beträgt.

Sobald k weiter erhöht wird, verringern unauffällige Simulationen die lokale Dichte der auffälligen Simulationen. Gleichzeitig bleibt die lokale Dichte innerhalb der unauffälligen Simulationen nahezu konstant. Der UAK berechnet sich nach Gleichung 2.6 aus dem Verhältnis der lokalen Dichte der k-nächsten Nachbarn und der lokalen Dichte des betrachteten Punkts p. Für den UAK der auffälligen Simulation bedeutet dies, dass stets größere Werte als eins ausgegeben werden, während sich für unauffällige Simulationen Werte nahe eins ergeben. Dementsprechend unterscheiden sich die UAK zwischen auffälligen und unauffälligen Simulationen. Mittels des Schwellwerts, der zwischen den beiden UAKs liegt, ergibt sich der AK der unauffälligen Simulationen zu null und der der auffälligen zu > 0.

Wird der Wert von k über die Differenz $N - N_{Ausreißer}$ erhöht, beeinflussen die auffälligen Simulationen die lokale Dichte der unauffälligen. Dementsprechend gleichen sich die Werte des UAK zwischen den unauffälligen und auffälligen Simulationen zunehmend an, was zu einer Verringerung des AK führt bis dieser schließlich den Wert null erreicht.

Durch diese Ergebnisse lässt sich ableiten, dass für den unteren beziehungsweise oberen Grenzwert des Hyperparameters k stets ein Wert größer als $N_{Ausreißer}$ beziehungsweise kleiner als $N - N_{Ausreißer}$ gewählt werden sollte. Gleichzeitig sollte der Wert des unteren Grenzwerts gemäß den Verfassern der Veröffentlichung zum LOF-Algortihmus (Breunig et al. 2000) aufgrund statistischer Fluktuationen stets größer als 10 sein.

Innerhalb der beschriebenen oberen und unteren Grenzen gibt es für das Datenset 1 und Datenset 2 einen Bereich, in dem der AK nahezu konstant verläuft. Diese Eigenschaft sorgt aufgrund des geringen Hyperparametereinflusses für eine hohe Robustheit des Verfahrens in diesem Bereich. Je nach verfügbarer Berechnungskapazität der Hardware, kann die Anzahl an Stützstellen für das Ensemble Modell variabel eingestellt werden. Soll lediglich ein Wert berechnet werden, wird als

Startpunkt k = N/2 empfohlen, da dieser Wert in der Mitte des robusten Wertebereichs liegt. Für alle folgenden Stützstellen können nacheinander beliebige Werte innerhalb des Wertebereichs $N_{Ausreißer} < k < N - N_{Ausreißer}$ dazugenommen werden. Durch dieses Vorgehen ist es möglich, auffälliges Crashverhalten vollautomatisiert zu detektieren, ohne dass der Anwender Werte für Hyperparameter festlegen muss. Der definierte Wertebereich erfordert jedoch die Kenntnis über den Wert $N_{Ausreißer}$, der dem Anwender im Allgemeinen jedoch nicht bekannt ist. Um das Ensemble Modell mit guten Werten für den Parameter k zu trainieren, muss für diesen daher eine Annahme getroffen werden. Da in der Praxis auch höhere Ausreißeranteile zwischen 10 und 20% vorkommen können, wird in den folgenden Experimenten mit dem LOF-Algorithmus ein Ensemble Modell mit $N_{Ausreißer}$=5 verwendet (Ausreißeranteil Datenset 1 mit 27 Simulationen: 18,5%; Ausreißeranteil Datenset 2 mit 31 Simulationen: 16,1%; Ausreißeranteil Datenset 3 mit 38 Simulationen: 13,2%).

6.3.2.2 Einfluss des Hyperparameters *k* beim KNND-Algorithmus

Als nächstes wird der Einfluss von *k* auf die Ergebnisse des KNND-Algorithmus untersucht. Bei diesem Verfahren wird die Distanz zum *k*-ten Nachbarn als UAK verwendet. In Abbildung 34 sind die Verläufe des AK über *k* für die drei Datensets 1, 2 und 3 dargestellt.

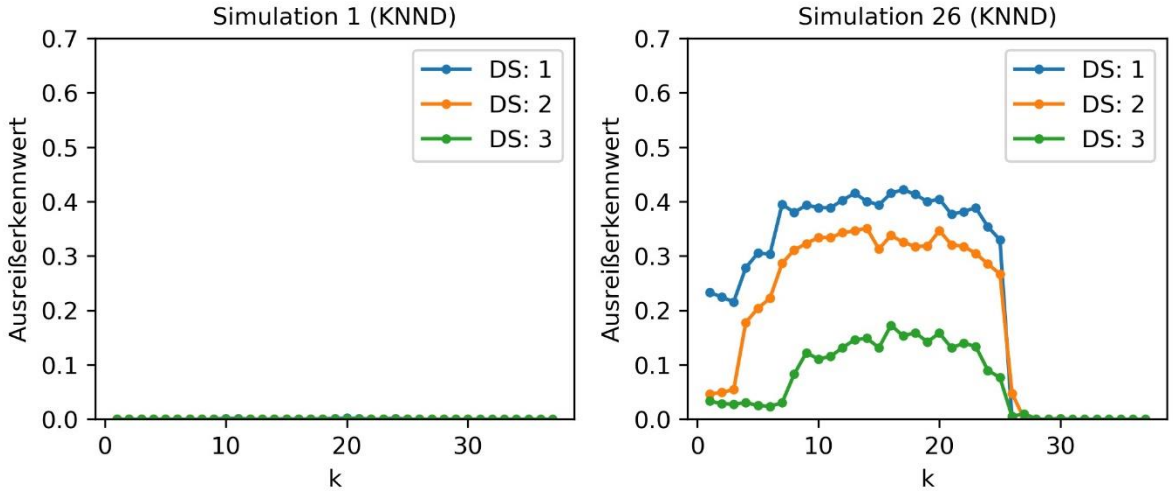


Abbildung 34: Verlauf des Ausreißerkennwerts für den KNND Algorithmus in Abhängigkeit des Hyperparameters k dargestellt für die drei Datensets (DS). Links für die unauffällige Simulation 1, rechts für die Ausreißer Simulation 26

Der AK der unauffälligen Simulationen beträgt für sämtliche Datensets und Hyperparameterwerte null. Für die auffälligen Simulationen gibt es sowohl beim Datenset 1 als auch beim Datenset 2 Wertebereiche für *k*, die über einen ausreichend hohen AK verfügen, um den Anwender so auf ein entsprechend abweichendes Crashverhalten hinzuweisen. Für Datenset 1 und 2 betragen die Maximalwerte 0,42 sowie 0,35. Da die AK jedoch über alle Zeitschritte gemittelt werden, erhält der Anwender für einzelne Zeitschritte dennoch Kennwerte nahe eins, sodass das betrachtete Bauteil hervorsticht. Der Maximalwert beträgt im dritten Datenset 0,17. Der Ausreißeranteil ist in diesem Datenset mit ca. 32% am größten, sodass die auffälligen Simulationen weniger stark hervorgehoben werden. Alle drei Kurven weisen jedoch einen ähnlichen Verlauf auf. Zunächst liegt ein geringer Wert für den AK vor. Mit steigendem Wert

des Hyperparameters *k* nimmt gleichzeitig der AK zu, bis ein Maximalwert erreicht wird und der AK im Folgenden auf null abfällt.

Für den Fall, dass wie im Datenset 1 lediglich ein Ausreißer in den Daten vorhanden ist, liefert bereits die Distanz zum ersten Nachbarn einen höheren UAK für die auffällige Simulation, da der erste Nachbar bereits eine unauffällige Simulation ist und damit eine große Distanz vorliegt. Dagegen sind die Abstände zum ersten Nachbarn für die unauffälligen Simulationen wesentlich kleiner. Dies führt dazu, dass sich der AK für die auffällige Simulation deutlich von null unterscheidet. Mit steigendem *k* erhöht sich sowohl für die unauffälligen als auch die Ausreißer Simulationen die Distanz zum *k*-ten Nachbarn und damit der UAK. Die Distanzen der unauffälligen Simulationen steigen zunächst jedoch langsamer an als die der Ausreißer. Dementsprechend steigt der AK mit steigendem k zunächst weiter an, bis ein Plateau erreicht wird. Ab dem Wert k = $N - N_{Ausreißer}$, fällt der AK auf null ab. Bisher war der k-te Nachbar einer unauffälligen Simulation eine andere unauffällige, wodurch stets kleinere Werte für die Distanzen vorlagen als bei einer auffälligen Simulation zu deren k-tem Nachbar der immer eine unauffällige Simulation war. Ab dem kritischen Wert ist jedoch der k-te Nachbar einer unauffälligen Simulation ein Ausreißer, was zu einem sprunghaften Anstieg der Distanz und damit des UAK führt. Damit liegen die UAK von unauffälligen und auffälligen Simulationen in ähnlichen Wertebereichen, sodass der Schwellwert größer als der UAK ist, was dazu führt, dass der AK stets null beträgt.

Bei den anderen beiden Datensets können ähnliche Effekte beobachtet werden, die auf dieselbe Begründung zurückzuführen sind. Es fällt jedoch auf, dass der AK schon für $k < N_{Ausreißer}$ ansteigt. Beispielweise gilt für DS2 $N_{Ausreißer} = 5$. Der AK steigt jedoch schon sprunghaft ab $N_{Ausreißer} = 4$ an. Dies kann darauf zurückgeführt werden, dass auch innerhalb der Ausreißer Simulationen ein unterschiedliches Crashverhalten auftritt, was größere Distanzen zum k-ten Nachbarn und damit ein frühzeitiges Ansteigen des AK verursacht. Dasselbe gilt für das Datenset 3 mit 12 Ausreißern, bei dem bereits ab k = 8 ein Anstieg des AK verzeichnet werden kann.

Damit ergibt sich für robuste Ausreißerkennwerte derselbe Wertebereich des Hyperparameters k wie beim LOF-Verfahren. Als untere Grenze kann $N_{Ausreißer}$ verwendet werden, während die obere Grenze auf $N - N_{Ausreißer}$ festgelegt wird. Erneut kann ein Ensemble Modell verwendet und damit die Robustheit der Methode gesteigert werden. Insgesamt liefert der KNND Algorithmus im Vergleich zum LOF Algorithmus geringere AK. Die Maximalwerte der drei Datensets betragen bei KNND 0,42, 0,35 und 0,17 verglichen mit denen des LOF in Höhe von 0,62, 0,59 und 0,54. Ein Grund dafür, dass LOF höhere AK liefert, ist der, dass zusätzlich lokale Dichten der einzelnen Dateninstanzen berücksichtigt werden. Dadurch fließt mehr Information über die umgebenden Dateninstanzen in die Berechnung der AK ein, was dazu führt, dass Ausreißer deutlicher identifiziert werden können. Die Resultate von KNND weisen den/die Ingenieur*in im ersten und zweiten Datenset auf ein auffälliges Crashverhalten hin. Bei den Ergebnissen des dritten Datensets besteht jedoch das Risiko, dass auffälliges Crashverhalten aufgrund der niedrigeren AK übersehen wird.

6.3.2.3 Einfluss des Hyperparameters p beim SOS Algorithmus

In Abbildung 35 sind die Ergebnisse für den SOS Algorithmus dargestellt.

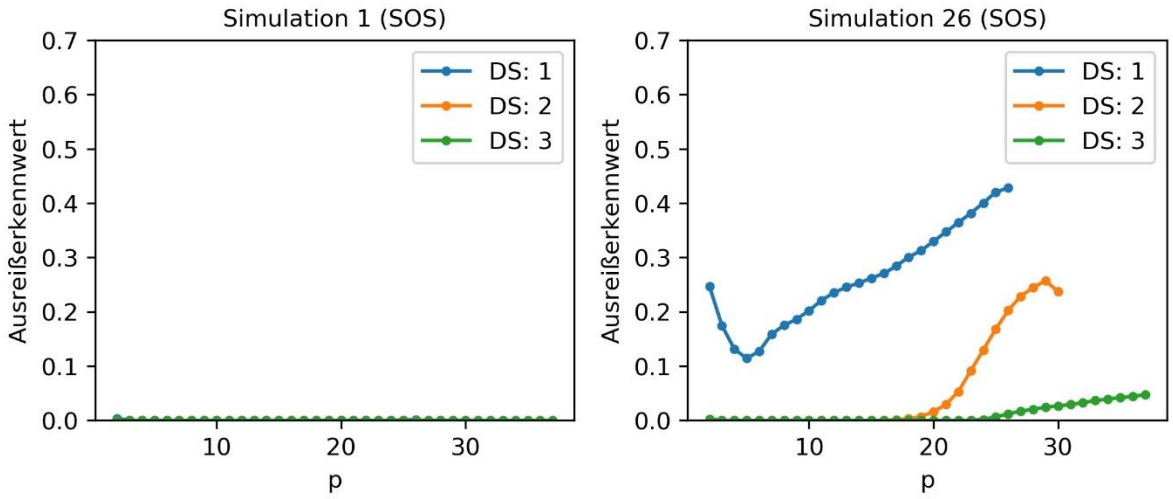


Abbildung 35: Verlauf des Ausreißerkennwerts für den SOS Algorithmus in Abhängigkeit des Hyperparameters p (Perplexität) dargestellt für die drei Datensets (DS). Links für die unauffällige Simulation 1, rechts für die Ausreißer Simulation 26

Die drei Kurven für die verschiedenen Datensets weisen für die auffällige Ausreißer Simulation 26 gegenüber KNN und LOF ein unterschiedliches Verhalten auf. Der Hyperparameter entspricht hier nicht der Anzahl der zu betrachtenden Nachbarschaften, sondern der Perplexität *p*, deren Einfluss in Kapitel 2 beschrieben wird. Es kann kein Wertebereich von *p* identifiziert werden, in dem ein konstanter Verlauf des AK auftritt und der sich so für ein Ensemble Modell eignen würde.

Für das erste Datenset weist der AK bei der Ausreißer Simulation 26 für p=1 zunächst einen Wert von 0,24 auf. Im weiteren Verlauf fällt der Wert zunächst ab, bevor er ab p=5 monoton auf seinen Maximalwert von 0,43 bei p=26 steigt. Für die beiden anderen Datensets liegt anfangs für den AK der Wert 0 vor. Im zweiten Datenset steigt der Wert zwischen $18 < p < 30$ auf 0,26 an, bevor er bei 31 auf den Wert 0,23 abfällt. Im dritten Datenset ist ab p=23 ein monoton steigender AK zu beobachten. Dessen Maximalwert beträgt 0,05 und liefert damit keine Hinweise auf ein auffälliges Crashverhalten. Bei jedem Datenset liefern größere Werte von *p* einen höheren Ausreißerkennwert. Jedoch kann für keines der drei Datensets ein Wertebereich für p abgeleitet werden, bei dem der AK konstant mit einem hohen Wert verläuft. Gleichzeitig fallen die Maxima der drei Kurven im Vergleich zu denen des LOF und KNND Algorithmus deutlich geringer aus. Die kleinen AK des dritten Datensets lassen den Anwender fälschlicherweise kein auffälliges Crashverhalten vermuten.

Wie in der Theorie zu SOS beschrieben wird, basiert der UAK auf der Berechnung sogenannter *Binding Probabilities*. Ein einzelner Ausreißer weist eine geringe *Binding Probability* mit anderen Datenpunkten auf, was sich wiederum in einem hohen UAK äußert. Für den Fall, dass jedoch zwei oder mehr Ausreißer mit einem ähnlichen Crashverhalten vorhanden sind, steigen

deren *Binding Probabilities* an, was dazu führt, dass ein geringerer UAK ausgegeben wird. Dieses Phänomen lässt sich auf jedes beliebige Ausreißer Crashverhalten übertragen, wenn dieses vielfach vorkommt. Dieser Effekt ist der Grund dafür, dass die AK im zweiten und dritten Datenset, in welchem die Ausreißer ein ähnliches Crashverhalten aufweisen, niedrig ausfallen. Bei den beiden anderen Algorithmen sinkt der AK zwar auch mit steigendem Ausreißeranteil in den drei Datensets, jedoch nicht in dem Maße wie bei SOS.

Da demnach weder sinnvolle Werte für den Hyperparameter *p* angegeben werden können noch die jeweiligen AK hoch genug sind, um zuverlässig Ausreißer zu identifizieren, ist der SOS-Algorithmus im Kontext der Ausreißerdetektion in Crashsimulationen ungeeignet.

6.3.3 Einfluss des Ausreißeranteils

In diesem Kapitel wird der Einfluss des Ausreißeranteils auf den AK untersucht. Wie in den Anforderungen für die Methode zur Ausreißerdetektion definiert, ist es wichtig, dass die Resultate des Algorithmus möglichst robust gegenüber einem steigenden Ausreißeranteil sein sollen. Dieses Verhalten wird daher im Folgenden detailliert analysiert.

Abbildung 36 zeigt den AK in Abhängigkeit der Anzahl der im Datenset enthaltenen Ausreißer für KNND, LOF, SOS und die PCAD. Die Basis bilden die aus den vorangegangenen Abschnitten bekannten 26 unauffälligen Simulationen. Um den Einfluss durch den Ausreißeranteil zu untersuchen, wird sukzessive jeweils ein weiterer Ausreißer in das Datenset eingefügt, bevor im Anschluss der entsprechende AK berechnet wird. Für KNND und LOF werden jeweils Ensemble Modelle mit den im vorangegangenen Abschnitt identifizierten Hyperparameter Werten verwendet. Erneut wird die zeitliche Information der 62 Zeitschritte gemittelt, sodass lediglich ein einzelner Wert pro Simulation betrachtet werden muss. Für eine übersichtlichere Darstellung werden jeweils die AK für die unauffälligen und die auffälligen Simulationen getrennt dargestellt, indem deren AK gemittelt werden.

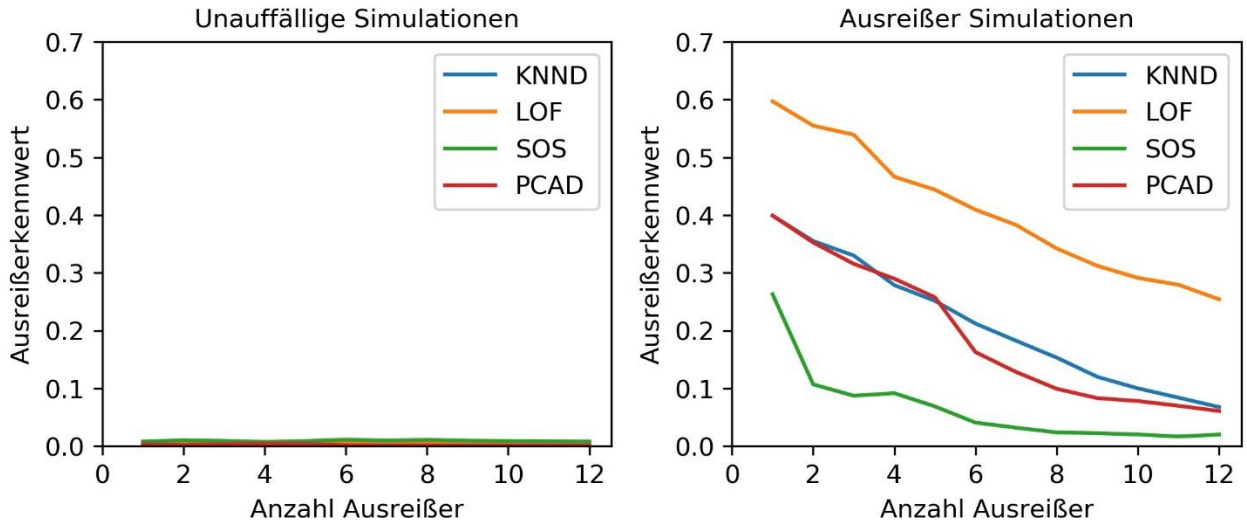


Abbildung 36: Ausreißerkennwert in Abhängigkeit der Anzahl vorhandener Ausreißer. Links für die unauffälligen Simulationen, rechts für die Ausreißer Simulationen

Aus der Darstellung geht hervor, dass unauffällige Simulationen unabhängig von den vorhandenen Ausreißern stets mit einem niedrigen AK nahe null bewertet werden. Damit ist sichergestellt, dass unauffälliges Crashverhalten nicht fälschlicherweise als Ausreißer identifiziert wird.

In der Abbildung der auffälligen Simulationen ist zu erkennen, dass der AK mit steigendem Ausreißeranteil für sämtliche betrachtete Algorithmen sinkt. LOF weist dabei die mit Abstand höchsten Werte auf. Bei nur veinem vorhandenen Ausreißer beträgt der AK 0,6. Mit steigendem Anteil der Ausreißer fällt er monoton ab bis er bei 12 den Wert 0,26 erreicht. Die AK von KNND und der PCAD folgen einem ähnlichen Verlauf auf niedrigerem Niveau. Sie betragen bei nur einem vorhandenen Ausreißer 0,4 und fallen bei 12 Ausreißern auf 0,06 ab. Für SOS liegen im Vergleich zu den anderen Algorithmen die niedrigsten AK von 0,26 bei nur einem bzw. 0,02 bei 12 Ausreißern vor.

Um zu bewerten, welcher Algorithmus stärker durch die vorhandenen Ausreißer beeinflusst wird, ist in Abbildung 37 für alle vier Algorithmen der relative Verlauf des AK in Bezug auf den AK mit jeweils nur einem vorhandenen Ausreißer dargestellt. Daraus geht hervor, dass der LOF Algorithmus die geringste Abhängigkeit gegenüber dem Ausreißeranteil aufweist, indem dieser bei 12 Ausreißern auf 42% abfällt. Die Ergebnisse von KNND und PCAD werden stärker beeinflusst, indem sich deren AK auf 17% bzw. 15% reduzieren. Am schlechtesten schneidet das SOS Verfahren bei 12 Ausreißern ab, indem sich der AK auf 7% verringert. Damit zeigt der LOF Algorithmus hinsichtlich seiner Robustheit in Abhängigkeit des Ausreißeranteils die besten Ergebnisse. Gleichzeitig dazu wird die Unterlegenheit des SOS Algorithmus gegenüber allen anderen untersuchten Algorithmen deutlich, da hier die größte Abhängigkeit hinsichtlich des Ausreißeranteils gegeben ist.

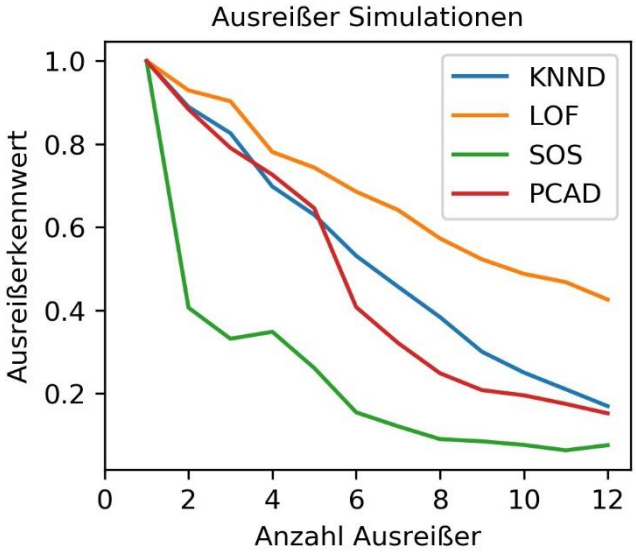


Abbildung 37: Ausreißerkennwert in Abhängigkeit der Anzahl vorhandener Ausreißer normiert auf den Ausreißerkennwert für nur einen vorhandenen Ausreißer

6.3.4 Skalierbarkeit

In diesem Abschnitt wird die Skalierbarkeit der Algorithmen hinsichtlich des zeitlichen Berechnungsaufwands in Abhängigkeit der Datenset-Größe untersucht. Dabei wird die Anzahl der zu analysierenden Simulationen zwischen 10 und 200 variiert. Hierfür werden alle Simulationen aus den Datensets DIN001, DIN01, DIN1 und DIN2 verwendet (siehe Kapitel 4). Die einzelnen Algorithmen werden mit den jeweiligen Standard-Parametern ausgeführt und deren Berechnungszeit aufgezeichnet. Es wird sichergestellt, dass auf dem verwendeten Computer keine weiteren Prozesse im Hintergrund ausgeführt werden. Darüber hinaus wird jede Messung 100-mal wiederholt und die Ergebnisse werden gemittelt, um deren Aussagekraft zu erhöhen. Abbildung 38 zeigt die Berechnungszeit in Abhängigkeit der Größe des Datensets für die vier Algorithmen.

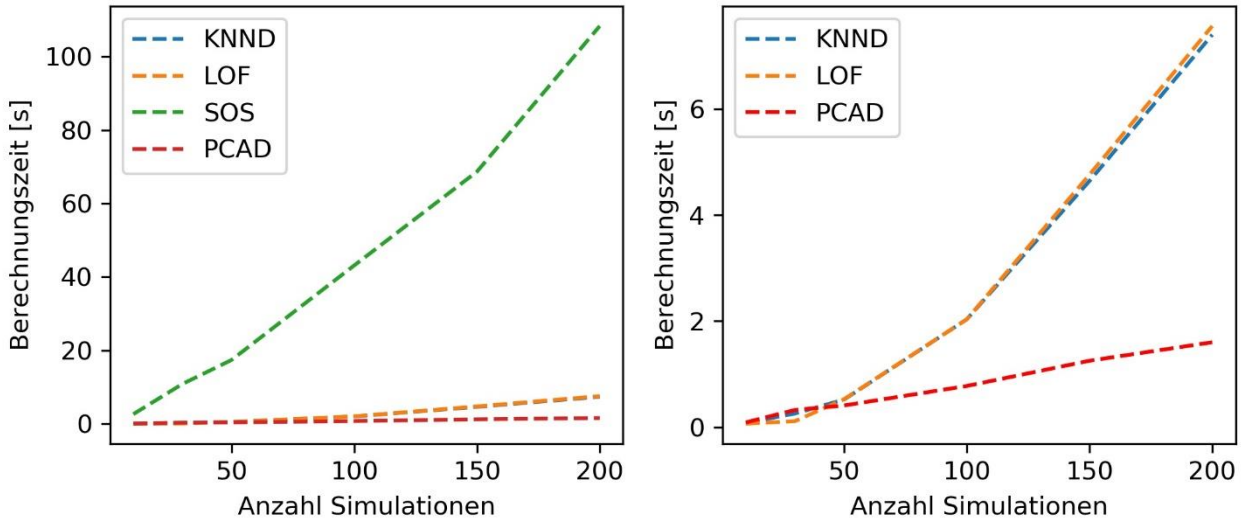


Abbildung 38: Berechnungszeit der vier Algorithmen in Abhängigkeit der Anzahl zu analysierender Simulationen. Links für alle vier Algorithmen, rechts der Übersichtlichkeit halber für KNN, LOF und PCAD

In der linken Darstellung sticht der SOS Algorithmus durch seine hohen Berechnungszeiten deutlich hervor. Bereits für kleine Datensets, in denen die anderen Algorithmen Bruchteile einer Sekunde benötigen, weist die Berechnungszeit von SOS bereits mehrere Sekunden auf. Mit steigender Größe des Datensets erhöht sich der Zeitbedarf weiter, bis er bei 200 Simulationen 120s beträgt. Im Vergleich zu den anderen Algorithmen, deren Berechnungszeit im Bereich weniger Sekunden liegt, ist sie bei SOS damit um ein Vielfaches größer. Damit ist dieser Algorithmus hinsichtlich des Berechnungsaufwands in der praktischen Anwendung am wenigsten geeignet. Für eine übersichtlichere Darstellung der Ergebnisse von PCAD, KNND und LOF, sind in Abbildung 38 rechts lediglich deren Verläufe visualisiert.

KNND und LOF weisen dabei über sämtliche Datensets hinweg einen ähnlichen Berechnungsaufwand auf. Für kleine Datensets bis 50 Simulationen beträgt dieser unter einer Sekunde und steigt für 200 Simulationen auf ein Vielfaches von 7,5s an. Mit steigender Anzahl an Simulationen skaliert die PCAD dagegen deutlich besser als die beiden anderen Verfahren. Während LOF und KNDD bei 200 Simulationen über 7s benötigen, beträgt der

Berechnungsaufwand für die PCAD lediglich 1,5s. Bereits bei der Analyse von nur einem Bauteil wird dieser Unterschied demnach deutlich. Für den Fall der Analyse eines Gesamtfahrzeugs mit mehreren Hundert Bauteilen vergrößert sich dieser Vorsprung der PCAD und verschafft ihr damit hinsichtlich des Berechnungsaufwands einen deutlichen Vorteil gegenüber den anderen untersuchten Verfahren. Es kann jedoch festgehalten werden, dass alle drei Algorithmen KNND, LOF und PCAD die in den Anforderungen definierte maximale Berechnungszeit von einer Minute erreichen und sich damit grundsätzlich zur Anwendung im Kontext der Ausreißerdetektion in Crashsimulationen eignen.

6.3.5 Zusammenfassung der Algorithmen

Eine Anforderung an den Algorithmus zur Ausreißerdetektion ist, dass falls dieser einen Hyperparameter aufweist, dessen Einfluss möglichst gering ausfallen soll. Als parameterfreies Verfahren schneidet die PCAD an dieser Stelle am besten ab, da keine Hyperparameter durch den Anwender festgelegt werden müssen. Für die übrigen drei Algorithmen liegt jeweils ein Hyperparameter vor. Für LOF und KNND ist es möglich einen Wertebereich für deren Hyperparameter *k* zu definieren, der robuste Ergebnisse liefert. Dadurch kann ein Ensemble Modell mit einer Vorauswahl geeigneter Parameterwerte berechnet werden, wodurch die Qualität der Ergebnisse gesteigert wird. Für das SOS Verfahren kann kein allgemeingültiger Wertebereich festgelegt werden.

Die durchgeführte Untersuchung des Einflusses durch den Ausreißeranteil ergibt, dass LOF zum einen die höchsten AK aufweist und zum anderen gleichzeitig den geringsten Einfluss gegenüber einer steigenden Anzahl an Ausreißern zeigt. Auch hier schneidet der SOS Algorithmus mit den niedrigsten Werten der AK und der größten Sensitivität gegenüber dem Ausreißeranteil am schlechtesten ab.

Bei der Analyse der Berechnungszeit liefert die PCAD die besten Resultate. Hier liegt der geringste Einfluss der Datenset-Größe vor. KNND und LOF zeigen ähnliche Verläufe, skalieren jedoch schlechter als die PCAD. Der SOS-Algorithmus eignet sich bereits bei kleinen Datensets durch seine hohen Berechnungszeiten nicht für die interaktive Analyse.

Zusammenfassend lässt sich festhalten, dass LOF am deutlichsten die Ausreißer in den Daten identifiziert. Da das Verfahren Hyperparameter aufweist, müssen mehrere Modelle trainiert und die Ergebnisse in einem Ensemble Modell kombiniert werden. Dies erhöht zwar einerseits den Berechnungsaufwand, liefert gleichzeitig jedoch robustere Ausreißerkennwerte.

Für den Fall, dass die Berechnungszeit für den Anwender das ausschlaggebende Kriterium ist, sollte die PCAD verwendet werden. Diese zeigt die beste Skalierbarkeit hinsichtlich der Anzahl zu analysierender Simulationen auf und ist gleichzeitig dazu in der Lage, Ausreißer durch einen hinreichend hohen AK zu identifizieren. Der Anwender sollte sich jedoch im Klaren darüber sein, dass gerade bei einem größeren Ausreißeranteil das Risiko besteht, diese durch zu niedrige AK zu übersehen.

Für die nachfolgenden Betrachtungen wird das SOS Verfahren aufgrund unzureichender Ergebnisse ausgeschlossen. Da die Resultate von KNND denen des LOF Algorithmus ähneln und keine weiteren Vorteile für KNND ersichtlich sind, wird auch dieses Verfahren nicht weiter berücksichtigt.

6.4 Einfluss des Hyperparameters *t* im *Median Absolute Deviation* Verfahren

Erneut werden die AK für jeden Zeitschritt getrennt berechnet. Da bei LOF der Hyperparameter *k* definiert werden muss, werden gemäß dem in Kapitel 6.3.2 gefundenen Wertebereich mehrere AK berechnet und deren Mittelwert als Ergebnis des Ensemble Modells verwendet. Dieses Vorgehen wird für sämtliche Zeitschritte wiederholt. Anschließend werden die AK über die Zeit gemittelt, sodass pro Bauteil ein einzelner AK vorliegt. Der Einfluss des Hyperparameters *t* wird für verschiedene Datensets mit variierendem Ausreißeranteil untersucht. Die AK werden jeweils für die unauffälligen Simulationen und die Ausreißer getrennt berechnet, indem jeweils über die entsprechenden Simulationen der beiden Kategorien der Mittelwert gebildet wird. Der Einfluss von *t* auf den AK für LOF ist in Abbildung 39 dargestellt.

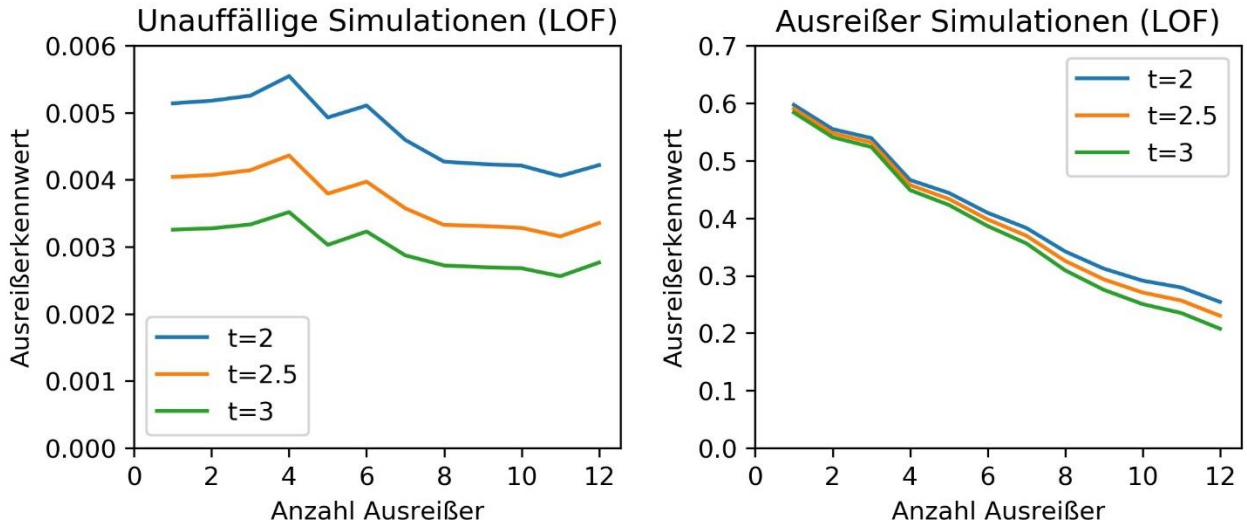


Abbildung 39: Darstellung des Ausreißerkennwerts aus dem LOF-Algorithmus in Abhängigkeit der Anzahl an Ausreißern für unterschiedliche Werte für t. Links für die unauffälligen Simulationen, rechts für die Ausreißer Simulationen

Aus der Literatur prominente Werte für t sind 2, 2,5 und 3 (siehe beispielsweise Miller (1991)). Diese werden auch in den folgenden Untersuchungen verwendet. Jede Kurve entspricht einem Wert für *t*. Die Anzahl der Ausreißer im jeweiligen Datenset ist in der Abbildung auf der horizontalen, der AK auf der vertikalen Achse dargestellt. Es zeigt sich, dass mit steigendem Wert von *t* der AK niedriger ausfällt. Dies gilt sowohl für die unauffälligen (links) als auch die auffälligen Simulationen (rechts) und hängt damit zusammen, dass der Schwellwert mit steigendem *t* größer wird. Damit verringert sich die Distanz zwischen dem UAK und dem Schwellwert, sodass der resultierende AK sinkt. Dieser Effekt verstärkt sich mit steigendem Ausreißeranteil. Bei der Betrachtung der Ergebnisse aus Abbildung 39 muss die Größenordnung der AK bei den unauffälligen und den Ausreißer Simulationen beachtet werden. Diese ist für erstere bedeutend kleiner, was zeigt, dass der Einfluss des Parameters *t* für den Anwender zumindest bei den unauffälligen Simulationen vernachlässigbar ist.

Für die PCAD werden in Abbildung 40 dieselben Effekte beobachtet mit dem Unterschied, dass der Einfluss von *t* mit steigendem Ausreißeranteil kleiner wird. Der Einfluss bewegt sich für beide Algorithmen in einer ähnlichen vernachlässigbaren Größenordnung.

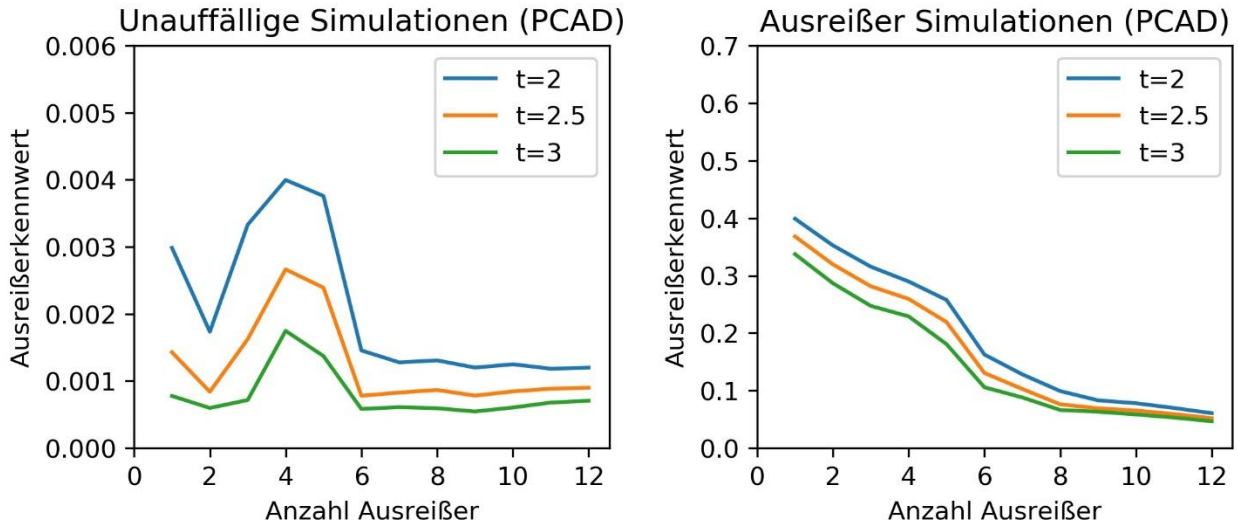


Abbildung 40: Darstellung des Ausreißerkennwerts aus dem PCAD-Algorithmus in Abhängigkeit der Anzahl an Ausreißern für unterschiedliche Werte für t. Links für die unauffälligen Simulationen, rechts für die Ausreißer Simulationen

Da sich durch kleine Werte von *t* höhere AK für die Ausreißer Simulationen erzielen lassen und gleichzeitig keine nennenswerten negativen Effekte bei den AK für die unauffälligen Simulationen auftreten, wird der Ansatz eines möglichst niedrigen Wertes empfohlen. Wie in den bisherigen Analysen wird für die weiteren Betrachtungen daher der Wert *t=2* verwendet.

6.5 Einfluss der Diskretisierung

In diesem Abschnitt wird der Einfluss der Voxel-Diskretisierung auf den AK analysiert. Für K_V werden die Werte 10mm, 20mm, 30mm und 50mm betrachtet und mit den Ergebnissen der originalen undiskretisierten Daten verglichen. Es wird untersucht, bis zu welcher Diskretisierung das auffällige Anfaltverhalten des *Längsträgers* zuverlässig detektiert wird und wie sich der Ausreißeranteil hierbei auf die Ergebnisse auswirkt. Darüber hinaus wird verglichen, welcher der beiden Algorithmen (LOF, PCAD) robuster gegenüber dem Einfluss der Diskretisierung ist.

Die Untersuchungen erfolgen erneut für unterschiedliche Ausreißeranteile. In sämtlichen Datensets sind stets 26 unauffällige Simulationen enthalten. Diese werden sukzessive um jeweils eine Ausreißer Simulation erweitert, woraufhin der AK für das neue Datenset berechnet wird. Der höchste Wert des Ausreißeranteils liegt im letzten Datenset vor, in welchem sämtliche 12 auffällige Simulationen enthalten sind. Abbildung 41 stellt die Ergebnisse der Analysen dar.

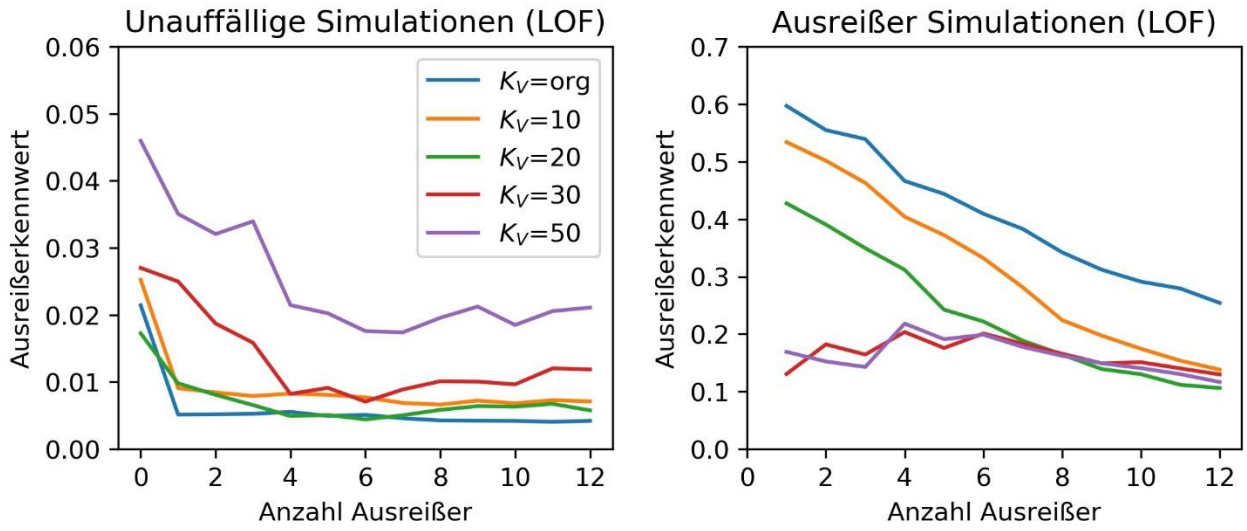


Abbildung 41: Darstellung des Ausreißerkennwerts aus dem LOF-Algorithmus in Abhängigkeit der Anzahl an Ausreißern für die originalen FE-Daten (org) und unterschiedliche Diskretisierungen K_V. Links für die unauffälligen, rechts für die Ausreißer Simulationen

Der AK wird in Abhängigkeit des Ausreißeranteils links für die unauffälligen und rechts für die Ausreißer Simulationen visualisiert. Die Resultate der originalen und diskretisierten Daten sind durch die einzelnen Linien abgebildet. Für die originalen Daten liegen für jedes Datenset die höchsten Werte vor. Bei einem Ausreißer beträgt der AK = 0,6 bis er im weiteren Verlauf für 12 Ausreißer auf den Wert 0,26 abfällt. Mit größer werdender Diskretisierung sinkt der AK, da zunehmend mehrere FE in einem diskretisierten Voxel auftreten und deren Informationen gemittelt werden. Mit gröberer Diskretisierung geht mehr Information über die originalen Daten verloren, was dazu führt, dass Simulationen mit auffälligem Crashverhalten von der Ausreißerdetektion schwieriger erkannt werden.

Eine Diskretisierung mit 10 bzw. 20 mm liefert für kleine Ausreißeranteile zunächst noch hohe Werte, die dem Anwender ein klares Indiz für ein auffälliges Crashverhalten liefern. Mit zunehmendem Ausreißeranteil fallen die AK weiter ab. Bei 12 Ausreißern wird mit den diskretisierten Daten ein AK zwischen 0,1 und 0,14 erzielt. Dem gegenüber stehen die originalen Daten, bei denen der AK 0,26 beträgt und damit deutlich höher liegt.

Für die Diskretisierung 30 und 50 mm beträgt der AK bereits bei den Datensets mit einem Ausreißer lediglich ca. 0,15. Der Ausreißeranteil zeigt gleichzeitig kaum mehr einen Einfluss. Diese Ergebnisse zeigen also, dass durch eine zu grobe Diskretisierung zu viel Information über das Crashverhalten verloren geht, was dazu führt, dass der Anwender durch niedrige AK nicht hinreichend auf auffälliges Crashverhalten hingewiesen wird. Gegenüber den originalen FE-Daten zeigt sich bei den diskretisierten Daten, dass sich der AK sensitiver gegenüber einem steigenden Ausreißeranteil verhält.

Bei den unauffälligen Simulationen verhält sich der Zusammenhang zwischen der Diskretisierung und dem Wert des AK andersherum. Während die originalen Daten die niedrigsten Werte für den AK zeigen, steigen sie mit gröberer Diskretisierung an. Der Vergleich der Größenordnungen der AK für die unauffälligen und die Ausreißer Simulationen zeigt jedoch,

dass dieser Effekt eine untergeordnete Rolle spielt.Die beim LOF-Algorithmus gesammelten Beobachtungen lassen sich auch auf die Ergebnisse der PCAD übertragen. Diese sind in Abbildung 42 visualisiert.

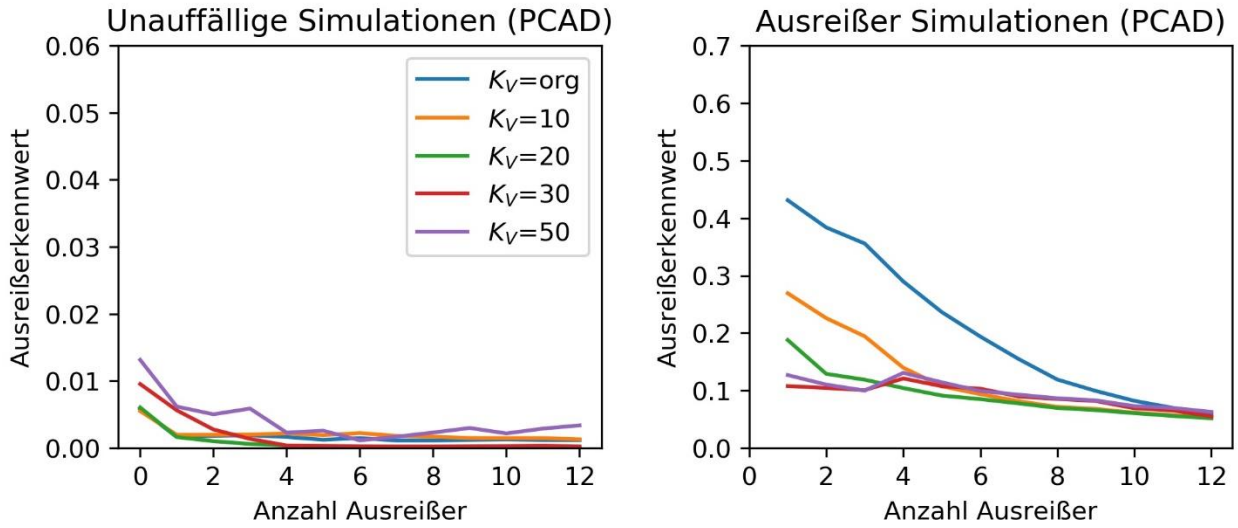


Abbildung 42: Darstellung des Ausreißerkennwerts aus dem PCAD-Algorithmus in Abhängigkeit der Anzahl an Ausreißern für unterschiedliche Diskretisierungen K_V. Links für die unauffälligen, Rechts für die Ausreißer Simulationen

Sowohl bei den unauffälligen als auch auffälligen Simulationen besteht, wie schon beim LOF, derselbe Zusammenhang zwischen der Diskretisierung und dem AK. Es werden jedoch insgesamt niedrigere Werte im Vergleich zum LOF beobachtet. Bei der PCAD ist es bereits für die Diskretisierungen 10 und 20 mm mit einem geringen Ausreißeranteil (kleiner als 4 Ausreißer) schwer, das auffällige Verhalten anhand der niedrigen AK zu identifizieren. Ab vier Ausreißern liegen durchweg niedrige Werte (<0,14) für alle Diskretisierungen vor. Hier zeigt LOF seine Überlegenheit gegenüber der PCAD, indem selbst bei diskretisierten Daten und einem hohen Ausreißeranteil auffällige Simulationen durch einen hinreichend hohen AK als solche identifiziert werden können.

6.6 Fazit

In diesem Kapitel wird eine Methode zur automatisierten Detektion von auffälligem Crashverhalten vorgestellt. Es werden die vier Algorithmen SOS, KNND, LOF und PCAD für die Ausreißerdetektion auf unterschiedliche Kriterien hin untersucht.

Die besten Ergebnisse erzielt die Anwendung des LOF-Algorithmus. Dieser erfordert zwar die Definition eines Hyperparameters. Wie im Rahmen der Analyse gezeigt wird, ist es jedoch möglich einen Hyperparameterbereich festzulegen und damit ein Ensemble Modell zu verwenden, das die Robustheit der Ergebnisse erhöht. Hierfür muss eine Annahme getroffen werden, bis zu welchem Ausreißeranteil zuverlässig Auffälligkeiten erkannt werden sollen. In dieser Arbeit wird exemplarisch $N_{Ausreißer}$=5 angenommen, was bei den betrachteten Datensets

einem Ausreißeranteil zwischen 13,1% und 18,5% entspricht. Bei noch höheren Anteilen wird davon ausgegangen, dass es sich nicht mehr um ein reines Ausreißerverhalten handelt, sondern schon von einem anderen Deformationsmodus gesprochen werden muss. Darüber hinaus zeigen die Untersuchungen, dass bei LOF die geringste Abhängigkeit gegenüber einem steigenden Ausreißeranteil vorliegt. Zudem kann auffälliges Crashverhalten, auch wenn die Daten vorab diskretisiert werden, zuverlässig detektiert werden.

Damit die Ergebnisse der Ausreißerdetektion stets zwischen null und eins liegen, werden die UAK skaliert und mittels des MAD-Verfahrens in den AK überführt. Hierfür ist die Definition des Parameters *t* erforderlich. In der Auswertung wird gezeigt, dass dessen Einfluss auf die AK gering ist. Desto kleiner dessen Wert, desto höhere AK werden ausgegeben. Da die AK der unauffälligen Simulationen gleichzeitig nur geringfügig beeinflusst werden, wird der Wert *t=2* empfohlen.

Die Untersuchungen diesen Kapitels zeigen, dass mittels der vorgestellten Methodik Ausreißer in Echtzeit vollautomatisiert und ohne zusätzlichen Input vom Anwender detektiert werden können. Als Resultat liegt für jedes Bauteil und für jeden Zeitschritt der Simulation ein AK zwischen null und eins vor. Dementsprechend können die Bauteile gemäß ihrem AK farblich im *Animator* visualisiert und damit sofort auffällige Bereiche im Fahrzeug identifiziert werden.

7 Crash-Verhalten-Detektor

Im vorangegangenen Kapitel wird gezeigt, dass sich mittels der Ausreißerdetektion auffälliges Crashverhalten automatisiert detektieren lässt. Es gibt jedoch Situationen, in denen die Ausreißerdetektion einen für den/die auswertende*n Ingenieur*in zu niedrigen Ausreißerkennwert (AK) liefert. Dies kann zum einen dadurch bedingt sein, dass ein diskretes Crashverhalten mit unterschiedlichen Deformationsmodi vorliegt (z.B. Knickt links, Knickt rechts) und jeder Modus ungefähr gleich oft vorkommt. Zum anderen können Transition im Crashverhalten vorliegen. Dies bedeutet, dass bei Betrachtung sämtlicher Simulationen ein kontinuierlicher Übergang zwischen einzelnen Deformationsmodi existiert. Im Beispiel des knickenden Trägerprofils, würde ein kontinuierlicher Übergang zwischen nach links knickenden, axial faltenden sowie auch nach rechts knickenden Varianten existieren. Da durch die Transitionen hinreichend viele Simulationen mit ähnlichem Crashverhalten vorliegen und dadurch der Ausreißer nicht eindeutig hervorsticht, kann es in diesem Fall vorkommen, dass für ein unerwünschtes Crashverhalten ein zu niedriger AK ausgewiesen wird. Zudem ist es oft eine individuelle Entscheidung des Ingenieurs, welches Crashverhalten kritisch eingestuft wird. Aus diesem Grund wird ein Verfahren benötigt, das dazu in der Lage ist, ein vom Anwender vorgegebenes Crashverhalten in neuen Simulationen wiederzuerkennen.

7.1 Methode zur Detektion von vorgegebenem Crashverhalten

7.1.1 Anforderungen an die neue Methode

Die in Kapitel 1.2 beschriebenen Anforderungen an den Crash-Verhalten-Detektor (CVD) können wie folgt zusammengefasst werden:

- Der CVD soll aus so wenig, wie möglich vorab kategorisierten Daten lernen, ein vorgegebenes Crashverhalten wiederzuerkennen
- Ergebnisse des CVD sollen im Animator darstellbar sein
- Gesamte zeitliche Information definiert ein Crashverhalten und soll daher berücksichtigt werden

7.1.2 Möglichkeiten zur Umsetzung des Crash-Verhalten-Detektors

Unter Berücksichtigung der dargelegten Anforderungen, gibt es verschiedene Möglichkeiten, ein vorgegebenes Crashverhalten wiederzuerkennen.

Option A: Maschinelles Lernen

Als Informationsgrundlage liegen die durch das Voxel-Verfahren diskretisierten Daten einer Simulation als vierdimensionaler Tensor vor. Entlang der ersten drei Dimensionen ist die räumliche Information über die FE enthalten, während als vierte Dimension das zeitliche Verhalten des Bauteils, während des Crashs abgebildet ist.

Dieser vierdimensionale Tensor kann wiederum in einen eindimensionalen Merkmalsvektor umgeformt werden, der aufgrund der Datenvorverarbeitung durch die Voxel-Diskretisierung

dieselbe Länge für alle zu betrachtenden Simulationen aufweist. Somit ergibt sich dessen Dimension als das Produkt der Anzahl an Voxeln und der Anzahl an Zeitschritten. Dieser Vektor kann anschließend verwendet werden, um einen Klassifikator aus dem Bereich des Maschinellen Lernens zu trainieren. Dafür eignen sich beispielsweise Verfahren wie *Random Forest, Support Vector Maschinen,* der *Gauß-Prozess* sowie die *Logistische Regression*, die in dieser Arbeit miteinander verglichen werden.

Ein wesentlicher Nachteil dieses Vorgehens ist jedoch, dass die so vorliegenden Daten aufgrund der großen Dimension sehr komplex sind. Dies wird insbesondere dann problematisch, wenn nur wenige Dateninstanzen für das Trainieren eines ML-Modells zur Verfügung stehen. Darüber hinaus steigt die für das Training benötigte Zeit mit der Dimension der Daten an. Aus diesen Gründen wird diese Vorgehensweise in der nachfolgenden Analyse nicht weiter betrachtet.

Option B: Dimensionsreduktion+Maschinelles Lernen

Viele Informationen aus dem eben beschriebenen hochdimensionalen Merkmalsvektor sind irrelevant für die Klassifikation des Crashverhaltens. Beispielsweise beinhalten die ersten Zeitschritte üblicherweise wenig relevante Information über das Crashverhalten, da erst eine geringe oder noch gar keine Deformation insbesondere auf der crashabgewandten Seite des Fahrzeugs vorliegt. Des Weiteren sind üblicherweise nur bestimmte Bereiche des Bauteils (bestimmte Voxel) betroffen. Die Dimension der zu Grunde liegenden Aufgabenstellung ist demnach bedeutend geringer als die des vorangehend vorgestellten Merkmalsvektors. Irrelevante Informationen können gerade bei kleinen Datensätzen zu Modellen mit niedriger Klassifikationsgenauigkeit (Formel 2.13) führen. Aus diesem Grund kann die Dimension vorab reduziert werden, um im Anschluss einen ML-Klassifikator auf Basis des reduzierten Merkmalsvektors zu trainieren. Dadurch, dass die Information über das Crashverhalten so in einem deutlich verkleinerten Merkmalsvektor enthalten ist, soll die Prädiktionsgenauigkeit bei einer geringen Anzahl verfügbarer Trainingsinstanzen erhöht werden. In der vom Autor betreuten Masterarbeit von A. Azouni wird die Eignung dieser Vorgehensweise für Daten aus der Crashsimulation bestätigt.

Option C: Selbstüberwachtes Lernen in Verbindung mit einem Klassifikator

Im Gegensatz zu klassischen ML-Algorithmen erzielen insbesondere bei der Klassifikation komplexer Daten wie beispielsweise Bildern *Deep Learning* Ansätze bessere Ergebnisse. Während die meisten *Deep Learning* Verfahren darauf angewiesen sind, dass eine große Datenmenge mit kategorisierten Daten zur Verfügung steht, liegt ein Fokus der Forschung derzeit auf der Entwicklung von Algorithmen, die die relevanten Merkmale aus einer geringen Anzahl an Trainingsdaten extrahieren können. Die so gewonnenen Merkmale, die die Informationen über die originalen Daten in einer kompakten Darstellung beinhalten, werden im Anschluss dazu verwendet, um einen einfachen Klassifikator zu trainieren.

In der vom Autor betreuten Masterarbeit von R. Dhanasekaran werden drei Verfahren aus dem Bereich des selbstüberwachten Lernens für die Klassifikation von Crashverhalten untersucht. Die Ergebnisse von *Siamese Networks* werden mit denen von *Byol* und *SimSiam* verglichen. In

Experimenten mit unterschiedlichen Bauteilen und Crashverhalten wird aufgezeigt, dass *SimSiam* die vergleichsweise besten Ergebnisse liefert. Aus diesem Grund wird dieses Verfahren im Detail untersucht und analysiert, inwiefern sich bessere Ergebnisse insbesondere hinsichtlich der Anzahl kategorisierter Trainingsdaten erzielen lassen gegenüber dem Ansatz basierend auf einer Dimensionsreduktion und klassischen ML-Verfahren.

Was *SimSiam* auszeichnet, ist die Art und Weise, wie Modifikationen aus den originalen Daten generiert und für das Training des neuronalen Netzwerks genutzt werden. Grundsätzlich könnten auch Modifikationen der originalen Daten verwendet werden, um die Trainingsdaten für „Option B: Dimensionsreduktion und Maschinelles Lernen“ anzureichern. Der Fokus dieser Arbeit liegt jedoch auf der Analyse von *SimSiam* und der Fragestellung, ob sich dieses Verfahren grundsätzlich für die Klassifikation von Crash-Simulationsdaten eignet. Für den Vergleich der Ergebnisse soll daher ein simples Benchmark-Verfahren verwendet werden und daher Option B ohne weitere Datenanreicherung durch Modifikationen.

7.1.3 Umsetzung von selbstüberwachtem Lernen

Im Folgenden werden die einzelnen Schritte von *SimSiam* beschrieben. Dazu gehören die verwendete Datenrepräsentation, die Modifikations-Techniken und deren Kombinationen sowie die Architektur des verwendeten Netzwerks.

Datenrepräsentation

Für 2D Datenformate wie Bilder gibt es viele Anwendungen, die von neuen Deep Learning Technologien profitieren. Darüber hinaus sind diese neuronalen Netzwerke meist weniger komplex und schneller zu trainieren, im Vergleich zu denen, die 3D Daten wie Voxel und Punktwolken verarbeiten. Daher werden neue Deep Learning Verfahren zunächst überwiegend für Anwendungen mit Bilddaten entwickelt, sodass hier eine größere Auswahl an Algorithmen (und Implementierungen) vorliegt. Dies ist auch der Fall bei selbstüberwachten Lernverfahren wie *SimSiam*. Um bereits vorhandene Implementierungen des Algorithmus nutzen zu können, werden auch in dieser Arbeit die Daten aus der Voxel-Diskretisierung zunächst in Bilder transformiert. Dies geschieht, indem die ersten drei Dimensionen über das räumliche Crahverhalten in einen 1D-Vektor überführt und als horizontale Achse des Bildes verwendet werden. Die zeitliche Information wird entlang der vertikalen Achse abgebildet. Als Resultat liegt ein zweidimensionales Bild wie in Abbildung 43 dargestellt vor, in dem sowohl die räumliche als auch die zeitliche Information über das Crashverhalten enthalten ist.

In der Abbildung ist die plastische Dehnung der einzelnen Voxel farblich dargestellt. Die oberen Zeilen des Bildes weisen einen Wert von 0 für sämtliche Voxel (Spalten) auf und stellen die ersten Zeitschritte des Crashs dar, in denen noch keine Deformation vorliegt. Der Wert der einzelnen Voxel erhöht sich für die Zeilen weiter unten im Bild, da dadurch die späteren Zeitschritte abgebildet sind, in denen plastische Dehnungen in Folge der Deformation des Bauteils auftreten. Einzelne Spalten, die den Voxeln des Bauteils entsprechen weisen den Wert null, andere einen größeren Wert auf. Dies liegt darin begründet, dass das Bauteil nicht in sämtlichen Bereichen plastisch deformiert wird. Einige Bereiche sind stärker betroffen als andere.

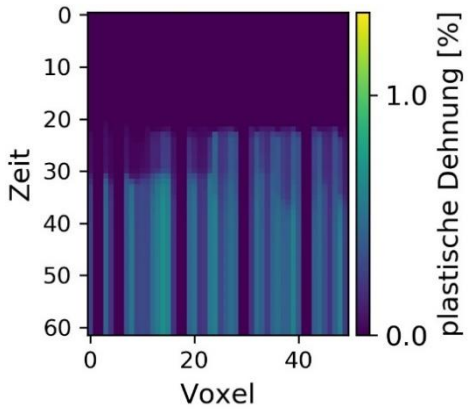


Abbildung 43: Darstellung des Crashverhaltens als zweidimensionales Bild auf Basis der voxelisierten FE-Daten. Entlang der Zeilen ist das zeitliche, entlang der Spalten das räumliche Verhalten der einzelnen Voxel abgebildet. Die plastischen Dehnungen sind farblich dargestellt

Es muss beachtet werden, dass durch die Darstellung des Crashverhaltens als Bild Informationen gegenüber der Voxel-Diskretisierung verloren gehen. Die Kontinuität entlang der Zeit bleibt in den Bildern zwar wie in Abbildung 43 dargestellt erhalten. Dafür wird sie jedoch für die räumlichen Informationen unterbrochen, indem die Voxel nun als eindimensionaler Vektor dargestellt werden. Das Ziel dieser Arbeit ist es, herauszufinden, ob SimSiam als selbstüberwachtes Lernverfahren grundsätzlich für die Klassifikation von Crashverhalten angewendet werden kann und ob es Vorteile gegenüber dem klassischen Ansatz PCA+ML gibt. Aus diesem Grund wird der Informationsverlust für diese Untersuchungen hingenommen und die Daten aus der Voxel-Diskretisierung werden in Bilder, wie beispielhaft in Abbildung 43 dargestellt, transformiert.

Aufbau des in dieser Arbeit verwendeten auf *SimSiam* basierenden Modells

Der grundlegende Aufbau von *SimSiam* nach Chen und He (2021) wird aus den Untersuchungen der vom Autor betreuten Masterarbeit von R. Dhanasekaran übernommen und ist in Abbildung 44 dargestellt. Lediglich die Architekturen und Hyperparameter des *Encoders f1*, *Encoders f2*, sowie des *Predictor h* werden für diese Arbeit modifiziert. Für die Funktionsweise von SimSiam wird auf Abschnitt 2.5.2.6 und Chen und He (2021) verwiesen.

Die Modifikationen x1 und x2 des originalen Bilds x sind die Eingangsdaten für die Encoder Netzwerke f1 und f2. Deren Aufbau ist identisch und besteht aus einem Convolutional Neural Network (CNN) sowie einem sogenannten Projection Network p. Das CNN ist aus einer Folge von sogenannten Convolution, Max Pooling und Dropout Schichten aufgebaut. Die Convolution ist eine mathematische Faltungsoperation, die für die Merkmalsextraktion (bspw. Kantendetektion) verwendet wird. Max Pooling wird eingesetzt, um unnötige Informationen der Ergebnisse der Convolution-Schicht herauszufiltern. So ist beispielsweise die exakte Lage einer Kante irrelevant für die Klassifikation des zugrunde liegenden Objekts. Um eine Überanpassung des trainierten Modells an die Trainingsdaten zu vermeiden, wird eine Dropout Schicht verwendet. Dafür wird ein als Hyperparameter zu definierender Anteil zufälliger Neuronen im Trainingsprozess deaktiviert.

Im Anschluss werden die Daten zum Projection Network p weitergeleitet. Dieses setzt sich aus einer Dense Schicht gefolgt von der sogenannten Batch Normalization, der Rectified Linear Unit (ReLu) als Aktivierungsfunktion und einer erneuten abschließenden Dense Schicht zusammen. In einer Dense Schicht sind alle Neuronen jeweils mit allen Neuronen der vorherigen Schicht verbunden. Dies erhöht den benötigten Berechnungsaufwand, weshalb sie oftmals eher am Ende eines komplexen neuronalen Netzwerks verwendet wird, wo bereits extrahierte Merkmale vorliegen. Dense Schichten sollen Zusammenhänge zwischen den Daten erfassen oder beispielsweise deren Dimension zu reduzieren. Die sogenannte Batch Normalization wird verwendet, um einen stabileren und schnelleren Trainingsprozess zu erreichen. Dabei werden die Eingangsdaten für die nächste Schicht zentriert und standardisiert. Die ReLu ist eine Aktivierungsfunktion und wird verwendet, um Nichtlinearitäten in den Eingangsdaten abzubilden. Negative Werte werden zu null gesetzt, positive unverändert weitergeleitet.

Im Anschluss folgt das Predictor Network h, das denselben Aufbau wie das Projection Network p aufweist. Die Hyperparameter der einzelnen Schichten sind in Tabelle 4 zusammengefasst. Im Folgenden wird auf die in dieser Arbeit verwendeten Modifikationen x_1 und x_2 des originalen Ursprungsbildes x eingegangen.

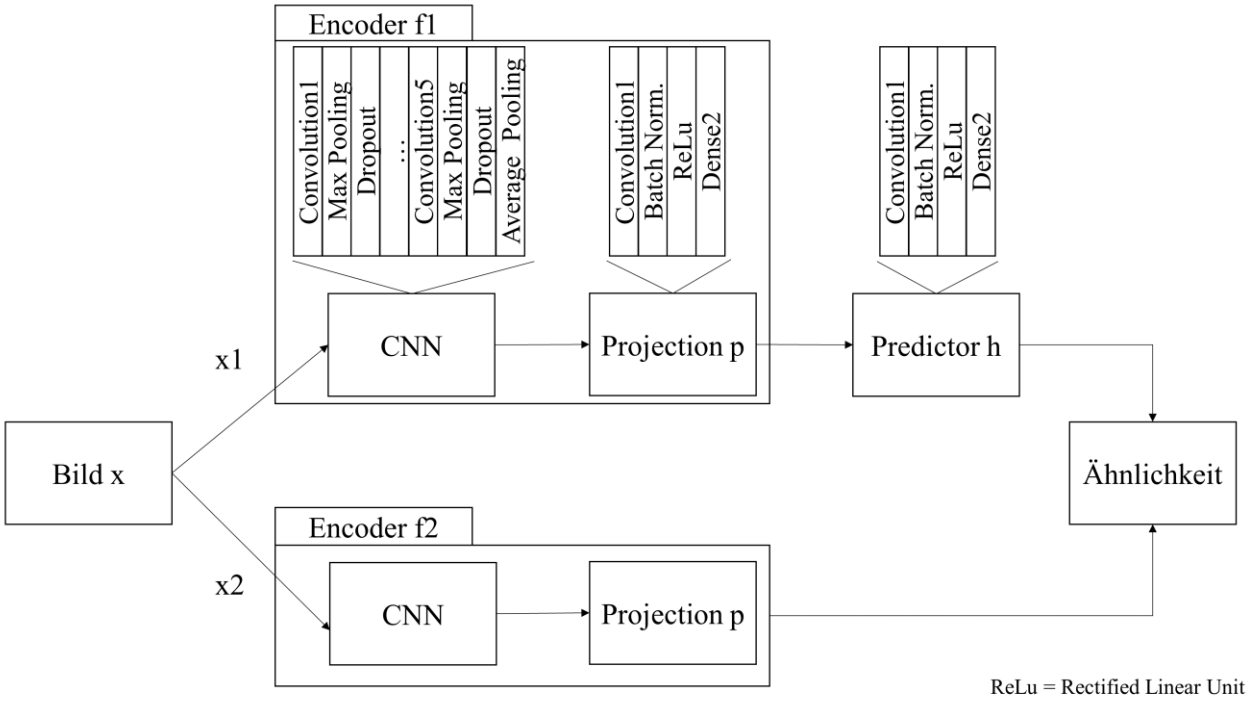


Abbildung 44: Architektur des verwendeten Netzwerks für SimSiam. Details zu den einzelnen Schichten des neuronalen Netzwerks sind Tabelle 4 zu entnehmen.

Tabelle 4: Hyperparameter der einzelnen Schichten von SimSiam

	Filteranzahl	Kernel-Größe	Strides	Padding	Aktivierungsfunktion
Convolution1	128	5,5	2	same	ReLU
Convolution2	256	3,3	2	same	ReLU
Convolution3	256	3,3	2	Same	ReLU
Convolution4	128	3,3	2	Same	ReLU
Convolution5	64	3,3	2	Same	ReLU
	Pool Größe	Strides			
Max Pooling	3	2			
	Rate [%]	Seed			
Dropout	0.3	55			
	Anzahl Neuronen				
Dense1	256				
Dense2	128				

Modifikations-Techniken

Die Idee hinter den Modifikationen ist es, zufällige Varianten aus den originalen Daten zu erzeugen. In der vom Autor betreuten Masterarbeit von R. Dhanasekaran werden drei Modifikations-Techniken angewendet (Rauschen, Skalierung, Duplizieren-Löschen), deren Funktionsweise im Folgenden beschrieben wird. Zur Anschauung der Modifikationen sind in Abbildung 45 die resultierenden Varianten des originalen Bildes dargestellt.

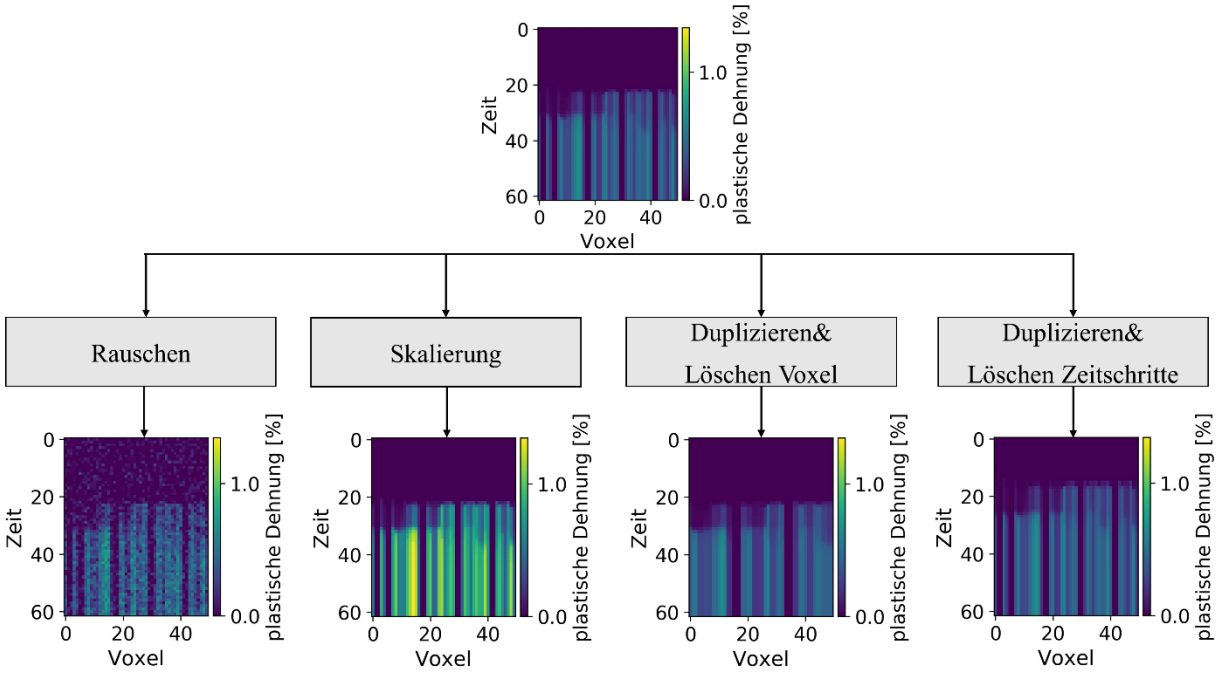


Abbildung 45: Exemplarische Resultate der vier betrachteten Modifikations-Techniken. Ausgehend vom originalen Bild werden das Rauschen (links), die Skalierung (zweites Bild von links), das Duplizieren&Löschen entlang der Voxel (drittes Bild von links) und entlang der Zeitschritte (rechts) betrachtet

Bei der Technik ***Rauschen*** wird jeder Pixel mit einem zufälligen Wert aus einer Gaußverteilung mit Mittelwert 0 und einer zu definierenden Standardabweichung addiert. Dieser Wert kann aufgrund der Symmetrie der Gaußverteilung um den Mittelwert 0 sowohl positiv als auch negativ sein. Mithilfe der Standardabweichung der Gaußverteilung kann die Stärke des Rauschens kontrolliert werden. Wie in Abbildung 45 unten links dargestellt, werden durch das Rauschen das Bild und die Struktur dahinter undeutlicher. Die Varianten, die durch dieses Verfahren entstehen, weisen bei einer Interpretation eines entsprechenden Crashverhaltens teilweise unphysikalisches Verhalten auf. Durch das hinzugefügte Rauschen treten nun an einzelnen Voxeln plastische Dehnungen auf, die in der Simulation nicht zu beobachten sind. Darüber hinaus steigen die plastischen Dehnungen eines Voxels entlang der Zeit nicht mehr kontinuierlich an, sondern können sich zwischenzeitlich durch das Rauschen auch wieder verringern. Ein solches Crashverhalten würde sich nicht mit einer Crashsimulation reproduzieren lassen. Die Frage ist, ob sich ein unphysikalisches Verhalten durch die Modifikationen negativ auf die Ergebnisse von SimSiam auswirkt.

SimSiam lernt charakteristische Strukturen in den Trainings-Bildern, um ein Crashverhalten zu identifizieren (z.B. Kanten, Helligkeitsverläufe, etc.). Daher ist in erster Linie relevant, was an dem Crashverhalten charakteristisch ist und wie sich hierauf die Modifikation Rauschen in den Bildern auswirkt. Dabei spielt beispielsweise eine Rolle ob es sich eher um lokale oder globale Effekte (sowohl zeitlich als auch räumlich) handelt und wie stark Deformationen sind. Bei einem globalen Effekt, der sich über einen großen Bereich des Bauteils und mehrere Zeitschritte erstreckt, sollte durch das Rauschen nach wie vor die charakteristische Information über das Crashverhalten in den modifizierten Bildern enthalten sein. Hier könnte ein vergleichsweise

stärkeres Rauschen verwendet werden. Bei einem lokalen Effekt dagegen, der sich nur auf einen kleinen Bereich des Bauteils und einen oder wenige Zeitschritte begrenzt, kann schon ein kleines Rauschen dazu führen, dass zu wenig verbleibende Informationen in den modifizierten Bildern stecken, um das Crashverhalten zuverlässig zu erkennen. Somit ist es für die Wiedererkennung des Crashverhaltens weniger relevant, ob die Modifikationen ein physikalisches Verhalten darstellen. Viel mehr spielt es eine Rolle, ob nach der Modifikation noch genügend Informationen über die Charakteristiken des Crashverhaltens vorhanden sind. Aus diesem Grund wird in diesem Kapitel untersucht, wie sich die Stärke des Rauschens auf die Ergebnisqualität von SimSiam auswirkt.

Die Idee hinter der ***Skalierung*** besteht darin, dass sämtliche Pixel des Bildes mittels eines konstanten Werts multipliziert werden. Dieser wird zufällig aus einem vorzugebenden Wertebereich ausgewählt. Dadurch bleibt die Struktur des Bildes erhalten und lediglich der Betrag der einzelnen Pixel wird skaliert. Dementsprechend liegen in der resultierenden Variante in Abbildung 45 hellere Farbwerte der einzelnen Pixel vor, die gemäß der Legende und einer physikalischen Interpretation für höhere plastische Dehnungen verglichen mit dem originalen Bild stehen. Dies könnte beispielsweise durch eine höhere Auftreffgeschwindigkeit verursacht werden. Auch hier kann das Ergebnis nach der Modifikation unphysikalisch sein, da die plastischen Dehnungen in der Realität nicht linear mit der Auftreffgeschwindigkeit skalieren. Aufgrund von Nichtlinearitäten beispielsweise in den Materialeigenschaften oder plötzlichem Versagen kann bei einer höheren Auftreffgeschwindigkeit ein völlig anderes Crashverhalten und damit andere plastischen Dehnungen auftreten. Es sei jedoch erneut angemerkt, dass es weniger darum geht physikalisch korrekte Modifikationen vorzunehmen, sondern stattdessen die charakteristischen Strukturen in den Bildern zu erlernen. Diese sind auch nach einer Skalierung weiterhin vorhanden.

Eine weitere Möglichkeit, Varianten eines originalen Bildes zu erzeugen ist das zufällige ***Duplizieren&Löschen*** einzelner Zeilen/Zeitschritte (*DLZ*) beziehungsweise Spalten/Voxel (*DLV*). Durch das Duplizieren zufälliger Spalten beziehungsweise Zeilen wird das Bild zunächst vergrößert. Durch das anschließende Löschen derselben Anzahl an Spalten bzw. Zeilen aus den Randbereichen wird die ursprüngliche Dimension des Bildes wiederhergestellt. Aus einem vorzugebenden Wertebereich wird ein Wert zufällig gezogen, der festlegt, wie viele Zeilen, bzw. Spalten dupliziert bzw. gelöscht werden sollen.

Bei der Vorgehensweise *DLV* werden zufällige Spalten dupliziert, wodurch sich die horizontale Position der Voxel, wie in Abbildung 45 (unten, drittes Bild von links) dargestellt, verschiebt. Wo vorher Voxel waren, in denen keine plastischen Dehnungen während des Crashs auftreten (dunkle Stellen) sind diese nun vereinzelt vorhanden. Dasselbe gilt auch für den umgekehrten Fall. Dadurch wird die Information, in welchen Bauteilbereichen plastische Dehnungen auftreten, manipuliert. Durch das anschließende Zuschneiden gehen Informationen über das Crashverhalten einzelner Bauteilbereiche verloren. Die zeitliche Information bleibt bei DLV dagegen unverändert. Im Vergleich zu DLZ werden bei DLV stärkere Manipulationen am zugrunde liegenden Crashverhalten vorgenommen. Auch hier gilt, dass zu starke Modifikationen des originalen Bildes dazu führen können, dass charakteristische Informationen über das Crashverhalten verloren gehen und *SimSiam* dadurch die zugrunde liegende Aufgabenstellung nicht mehr erlernen kann.

Bei *DLZ* werden zufällige Zeitschritte im Crash dupliziert und das Bild im Anschluss daran durch zufälliges Löschen der ersten und letzten Zeilen auf die ursprüngliche Dimension zurückgeführt. Abbildung 45 zeigt, dass gegenüber dem originalen Bild die Position der Spalten/Voxel unverändert ist, sich dagegen die Zeilen verändert haben. Zu einem früheren Zeitpunkt im Crash können bereits plastische Dehnungen beobachtet werden. Dies zeigt, dass für die Duplizierung der Zeilen überwiegend jene mit plastischen Dehnungen aus den späteren Zeitschritten ausgewählt wurden, statt denen zu den frühen Zeitschritten, wo noch keine plastischen Dehnungen auftreten. Gleichzeitig wurden eher die frühen Zeitschritte gelöscht. Wird dieses Ergebnis physikalisch interpretiert, so fängt die Deformation des Bauteils nun zu einem früheren Zeitschritt im Crash statt. Auch hier kann sich unphysikalisches Crahverhalten ergeben, wenn die Kontinuität in der Zeit aufgebrochen wird.

Das beschriebene Vorgehen zeigt, dass die Modifikationen auf den Bildern angewendet werden, die sich aus der Voxel-Datenrepräsentation ableiten lassen. An dieser Stelle sei angemerkt, dass es genauso möglich wäre, die Modifikationen im dreidimensionalen Voxel-Raum durchzuführen, bevor die Daten im Anschluss in eine Bildrepräsentation überführt werden. Für die im Rahmen der Analyse verwendeten Modifikations-Techniken macht dies jedoch keinen wesentlichen Unterschied und wird deshalb nicht weiter betrachtet.

Kombination der Modifikations-Techniken

Bei *SimSiam* wird ein Bild durch zwei unterschiedliche Sets an Modifikationen (S1 und S2) für das *Netzwerk 1* beziehungsweise *Netzwerk 2* modifiziert. Die Sets wiederum bestehen aus einer Aneinanderreihung unterschiedlicher Modifikationen und sind in Tabelle 5 dargestellt.

Tabelle 5: In den Experimenten untersuchte Sets an Modifikationen für Netzwerk 1 und Netzwerk 2 in SimSiam

	Erste Modifikation	**Zweite Modifikation**
Set 1	DLZ/DLV	Skalierung
Set 2	Skalierung	Rauschen

Klassifikator

Nach dem Training des selbstüberwachten Lernverfahrens stehen die neuen Merkmalsvektoren der Trainingsinstanzen zur Verfügung. Im nächsten Schritt werden diese gemeinsam mit den durch den Anwender vorgegebenen Kategorien dazu verwendet, einen Klassifikator zu trainieren. In dieser Arbeit wird hierzu ein einfaches neuronales Netz eingesetzt. Die Merkmalsvektoren stellen den Input für die Dense-Schicht mit 16 Neuronen dar. Diese sind mit einem Output Neuron verknüpft. Die logistische Funktion bildet die Ergebnisse der Prädiktion auf den Wertebereich zwischen 0 und 1 ab. In dieser Arbeit wird der Schwellwert 0,5 verwendet, um diese Werte in ein binäres Ergebnis umzuwandeln, das anzeigt, welches der beiden Crashverhalten vorliegt.

7.2 Anwendung des Crash-Verhalten-Detektors mit konkreten Daten

Für die Validierung überwachter Lernverfahren werden die verfügbaren Daten in ein Trainings- sowie Testset unterteilt. Die Simulationen im Trainingsset werden für das Training des Modells verwendet, während das Testset lediglich für die Validierung des trainierten Modells herangezogen wird. Es wird sichergestellt, dass die Datensets für das Training und Testen jeweils gleich viele Simulationen aus beiden Kategorien beinhalten.

Für das Training werden die 50 Simulationen aus dem Datenset *DIN001* (siehe Kapitel 4) herangezogen. In dieser Arbeit wird exemplarisch die binäre Klassifikation am Beispiel des unterschiedlich anfaltenden Längsträgers aus Abbildung 21 (Seite 58) betrachtet. Die Simulationen, in denen das Anfalten im vorderen Bereich beginnt, werden der Kategorie *A* zugeordnet, diejenigen, in denen die Deformation zuerst im hinteren Bereich auftritt, der Kategorie *B*. Grundsätzlich funktioniert die in dieser Arbeit gezeigte Methode jedoch auch für eine beliebige Anzahl an Kategorien, falls mehr als zwei Deformationsmodi eines Bauteils detektiert werden sollen.

Da ein balanciertes Training mit jeweils gleich vielen Simulationen aus beiden Kategorien durchgeführt werden soll und 26 Simulationen aus Kategorie *A* beziehungsweise 12 Simulationen aus Kategorie *B* vorliegen, werden aus Kategorie *A* lediglich 12 Simulationen betrachtet, sodass insgesamt 24 Simulationen zur Verfügung stehen.

Das Ziel des CVD ist, mit so wenigen kategorisierten Daten wie möglich ein Modell mit möglichst hoher Vorhersagegüte zu trainieren. Daher werden für die Analyse des Konvergenzverhaltens aus den vorhandenen 24 Simulationen kleinere Datensets gebildet und jeweils untersucht, welche Klassifikationsgenauigkeit (Formel 2.13) sich mit der jeweiligen Anzahl an Trainingsdaten erreichen lässt. Tabelle 6 fasst die Informationen der betrachteten Trainingsdatensets 1 - 4 zusammen.

Tabelle 6: Die in den Experimenten betrachteten vier Datensets mit der jeweils zugehörigen Anzahl an Trainingsinstanzen

Datenset	**Gesamtanzahl Trainingsdaten (N)**
1	2
2	6
3	10
4	20

Für das Testen der trainierten Modelle werden die Simulationen aus dem Datenset *DIN01* verwendet. Hier liegen 25 Simulationen aus Kategorie *A* und 9 Simulationen aus Kategorie *B* vor. Für ein balanciertes Training werden erneut gleich viele Simulationen aus beiden Kategorien verwendet, sodass insgesamt 18 Simulationen als Testset zur Verfügung stehen.

7.3 Ablauf der Experimente und Parametrierung der Modelle

7.3.1 Benchmark-Methode

Für den späteren Vergleich der Ergebnisse von *SimSiam* wird als Benchmark-Methode die Kombination aus Dimensionsreduktion und ML-Klassifikator verwendet. Zur Dimensionsreduktion wird die PCA genutzt. Damit kein Informationsverlust auftritt, wird die Zieldimension so gewählt, dass sie der Anzahl an Trainingsdaten entspricht (siehe Tabelle 6). Dadurch wird lediglich eine Koordinatentransformation durchgeführt, ohne Informationen über die originalen Daten zu verlieren. Dadurch, dass die ursprüngliche Dimension viel größer ist als die Anzahl an Trainingsdaten erlaubt dieses Vorgehen dennoch eine deutliche Reduktion der Komplexität der Daten.

Als ML-Klassifikatoren werden die in Tabelle 7 dargestellten Algorithmen aus dem Softwarepaket *Scikit-learn* (Bisong 2019) verwendet. Das Ziel der Experimente ist es, das Konvergenzverhalten der einzelnen Verfahren bei steigender Anzahl an Trainingsdaten zu untersuchen. Dabei wird angenommen, dass sich die Genauigkeit der Modelle mit steigender Anzahl an Trainingsinstanzen erhöht. Je nachdem welche *N* Dateninstanzen aus den 24 Trainingsdaten gezogen werden, kann die Qualität der Ergebnisse teilweise stark variieren. Dies kommt insbesondere bei kleinen Datensets negativ zum Tragen. Um ein allgemeineres Bild über die Eigenschaften eines Algorithmus zu erhalten, wird deshalb jedes Experiment 50-mal wiederholt. Für jedes *N* aus Tabelle 6 werden 50 unterschiedliche Datensets durch „Ziehen ohne Zurücklegen“ aus den 24 Trainingsinstanzen generiert. Damit ein fairer Vergleich zwischen den verschiedenen Algorithmen ermöglicht wird, werden die so erzeugten Subsets gespeichert und für sämtliche Klassifikatoren verwendet.

Für jedes der resultierenden Datensets wird ein Klassifikator trainiert. Mittels einer Rastersuche und dem Kreuzvalidierungsverfahren werden die jeweiligen Hyperparameter des Verfahrens optimiert. Tabelle 7 stellt die untersuchten Parameter der Rastersuche dar. Da für N = 2 keine Kreuzvalidierung möglich ist, findet keine Optimierung der Parameter statt. Stattdessen werden die Klassifikatoren bei diesen Datensets mit den jeweiligen Standardparametern durchgeführt, die in Tabelle 7 hervorgehoben sind. Für N = 6 kann die 3-fache Kreuzvalidierung verwendet werden, was zu einem Train-Test Split von 4/2 führt. Für N = 10 und N = 20 wird jeweils die 5-fache Kreuzvalidierung angewendet.

Tabelle 7: Für die Benchmark-Methode verwendete Algorithmen und untersuchte Hyperparameter (Implementierungen aus Scikit-learn *(Bisong 2019)). Die jeweiligen Standardparameter der Algorithmen sind schwarz hervorgehoben*

Klassifikator	**Hyperparameter**	**Suchbereich für Grid Search**
Random Forest	Criterion	**Gini**, Entropy
	Max Features	**Auto**, sqrt, log2
	Warm Start	True, **False**
	Oob_score	True, **False**
	N_estimators	50,**100**,150,200,250,300
Support Vector Maschine	Kernel	Linear, poly, **rbf**, sigmoid
	C	1e-5, 1e-4, 1e-3, 1e-2, 1e-1, **1** 10,100
Gauß Prozess	Warm start	True, **False**
	Kernel	RBF, WhiteKernel, Exponentiation(RationalQuadratic, exponent=2), ExpSineSquared, PairwiseKernel, **None**
Logistische Regression	C	1e-5, 1e-4, 1e-3, 1e-2, 1e-1, **1**, 10,100
	Fit_intercept	**True**, False
	Warm_start	True, **False**

7.3.2 SimSiam

Für die Analyse von *SimSiam* wird die Architektur aus Abschnitt 7.3 verwendet. Im Gegensatz zu den ML-Verfahren, bei denen oft aufgrund der geringeren Komplexität eine Rastersuche zur Hyperparameteroptimierung verwendet werden kann, ist gerade bei neuronalen Netzen die Optimierung der Hyperparameter schwierig. Dies liegt zum einen daran, dass neuronale Netze über eine Vielzahl an Parametern verfügen. Darüber hinaus ist die für das Training benötigte Zeit im Vergleich zu klassischen ML-Verfahren wesentlich höher. Im Fall von *SimSiam* kommt zu all dem hinzu, dass die Ergebnisse aufgrund der zufälligen Modifikationen stochastisch sind und daher für eine zuverlässige Aussage über eine konkrete Hyperparameterkonfiguration mehrere Wiederholungen notwendig sind. Daher ist eine vollumfängliche Optimierung der Hyperparameter nicht möglich. Insbesondere die Wiederholungen für die statistische Absicherung der Ergebnisse sorgen für lange Berechnungszeiten, die die Optimierung erschweren.

Aus diesem Grund wird ein Greedy-Ansatz verfolgt und der Einfluss relevanter Hyperparameter von *SimSiam* sequenziell untersucht. Es wird jeweils der beste Wert für die nächsten Untersuchungen ausgewählt. Der Fokus liegt dabei auf den Parametern, die die Trainingsphase im Wesentlichen beeinflussen. Neben der Analyse des Einflusses durch die Batch-Größe, werden unterschiedliche Modifikations-Techniken und der Einfluss deren Parameter untersucht. Die Architektur des neuronalen Netzes und dessen Hyperparameter aus Tabelle 4 bleiben dabei unverändert.

Zunächst wird der Einfluss der Batch-Größe analysiert. Dieser ist ein wesentlicher Parameter, der den Lernprozess des Modells beeinflusst. In Tabelle 8 sind die Werte abgebildet, die für die einzelnen Datensets verwendet werden.

Tabelle 8: In den Experimenten untersuchte Batch-Größen für die vier betrachteten Datensets

Datenset	**Gesamtanzahl Trainingsdaten (N)**	**Batch-Größe**
1	2	2
2	6	2,3,6
3	10	2,5,10
4	20	2,5,10,20

Ein wichtiger Bestandteil von *SimSiam* ist die Wahl der Modifikations-Techniken. Daher werden diese als nächstes untersucht. Zunächst wird dabei betrachtet, ob die Technik DLV gegenüber der DLZ Vorteile liefert. Darüber hinaus wird der Einfluss der Reihenfolge der Modifikationen betrachtet. In Set 1 (siehe Tabelle 5) für das *Netzwerk 1* werden die Modifikationen *DLV* und *Skalierung* angewendet. Deren Reihenfolge spielt keine Rolle. Die Vorgehensweise, zuerst einen Bildausschnitt auszuwählen und im Anschluss die Pixel mit einem konstanten Faktor zu multiplizieren liefert dieselben Resultate, wie das Bild zuerst zu skalieren und im Anschluss den Bildausschnitt auszuwählen. In Set 2 für *Netzwerk 2* wird das Bild verändert, indem die Pixel skaliert und auf das gesamte Bild ein Gaußches Rauschen addiert wird. In diesem Fall wirkt sich die Reihenfolge der beiden Methoden auf das Ergebnis aus und wird daher in den Untersuchungen betrachtet. Die Variante, in der zuerst die Skalierung und im Anschluss das Rauschen hinzugefügt wird, wird im Folgenden als *DLV-SR* bezeichnet, die andere als *DLV-RS*

Die Modifikations-Techniken verfügen außerdem über Hyperparameter. Daher wird im Anschluss der Einfluss des Skalierungsfaktors, des Rauschens und der zufälligen Duplizierung von Zeilen bzw. Spalten analysiert. Tabelle 9 fasst die untersuchten Parameter der einzelnen Experimente zusammen. Für die Skalierung wird der untere Grenzwert konstant bei 1,25 belassen. Dafür wird die obere Grenze mit den Werten aus der Tabelle in den einzelnen Experimenten variiert. Bei der Modifikation DLV wird die Anzahl zu duplizierender beziehungsweise zu löschender Spalten variiert. Für das Gaußsche Rauschen werden für die Standardabweichung die Werte aus der Tabelle verwendet. In jedem der Experimente zu den drei Modifikations-Techniken wird zudem eine Variante betrachtet, in der die Modifikation gänzlich weggelassen wird.

Tabelle 9: In den Experimenten betrachtete Werte für die Hyperparameter der einzelnen Modifikations-Techniken

Skalierung (obere Grenze für den Skalierungsfaktor *s*)	2,3,4,6,8
DLV/DLZ (Anzahl *e* zu duplizierender/löschender Spalten)	5,10,12,15,20,30
Rauschen (Standardabweichung *r*)	0,0001; 0,001; 0,01; 0,1

Jedes der Experimente wird erneut für die vier verschieden großen Datensets aus Tabelle 6 durchgeführt. Da die Modifikationen stochastisch sind, unterscheiden sich die Ergebnisse von *SimSiam* selbst bei Wiederholung der Experimente mit denselben Hyperparametern auf denselben Trainingsinstanzen. Um die Ergebnisse statistisch abzusichern, ist es daher notwendig, diese mehrere Male zu wiederholen (hier erneut 50-mal). Da der Fokus bei der Analyse von *SimSiam* auf dem Einfluss verschiedener Hyperparameter liegt, werden bei den 50 Wiederholungen im Gegensatz zu den Untersuchungen der Benchmark-Methode stets dieselben Dateninstanzen für das Training verwendet. Dadurch wird zusätzliche Varianz durch variierende Trainingsdatensets vermieden und die durch die Hyperparameter hervorgerufenen Effekte können besser untersucht werden.

7.3.3 Vergleich der Benchmark-Methode mit SimSiam

Nach der Untersuchung verschiedener Parameter bei *SimSiam*, erfolgt der Vergleich mit den Ergebnissen der Benchmark-Methode. Bei der Benchmark-Methode werden für jeden Klassifikator und jedes Datenset jeweils 50 Wiederholungen auf unterschiedliche Dateninstanzen durchgeführt. Die Untersuchungen von *SimSiam* erfolgen dagegen mit jeweils denselben Dateninstanzen, um die Varianz in den Ergebnissen zu reduzieren und damit die Effekte der Hyperparameter deutlicher analysieren zu können. Für den direkten Vergleich der beiden Methoden werden jeweils 50 Modelle mit je unterschiedlichen Dateninstanzen trainiert. Die Trainings- und Testdaten sind dabei jedoch für beide Methoden (Benchmark-Methode und *SimSiam)* jeweils die gleichen.

7.3.4 Bewertung des Diskretisierungseinflusses

Die Voxel-Diskretisierung beeinflusst die Datenrepräsentation und damit die Größe der Bilder für *SimSiam*. Da die Voxel auf der horizontalen Achse der Bilder dargestellt sind, unterscheidet sich jeweils deren Breite für die unterschiedlichen Diskretisierungen. Tabelle 10 zeigt die betrachteten Diskretisierungen und die daraus resultierenden Dimensionen der Bilder.

Tabelle 10: Betrachtete Diskretisierungen und daraus resultierende Dimensionen der Trainingsbilder

Diskretisierung K_V	**Anzahl Voxel**	**Dimension**
10	1225	62x1225
20	216	62x216
30	72	62x72
50	14	62x14

Abhängig von der gewählten Diskretisierung ist die auf den Bildern enthaltene Information über das Crashverhalten unterschiedlich komplex. Daher wird untersucht, inwiefern die Diskretisierung die Prädiktionsgenauigkeiten von *SimSiam* beeinflusst, um Rückschlüsse auf die Robustheit des Verfahrens bei unterschiedlich komplexen Trainingsdaten zu erhalten.

Für die Kapitel 7.4-7.6 werden zunächst exemplarisch die diskretisierten Daten mit K_V=20 verwendet. In Kapitel 7.7 wird der Einfluss der Diskretisierung auf die Ergebnisse des CVD untersucht, in dem die Resultate der Diskretisierungen 10mm, 20mm, 30mm sowie 50 mm miteinander verglichen werden. Erneut werden jeweils 50 Wiederholungen mit unterschiedlichen Dateninstanzen vorgenommen.

7.4 Ergebnisse der Benchmark-Methode

Im Folgenden erfolgt die Analyse der Benchmark-Methode. Als Auswertungsgröße wird die Genauigkeit (Formel 2.13) der Prädiktionen auf den Testdaten verwendet. Abbildung 46 links stellt diese in Abhängigkeit der Anzahl der verwendeten Trainingsinstanzen dar.

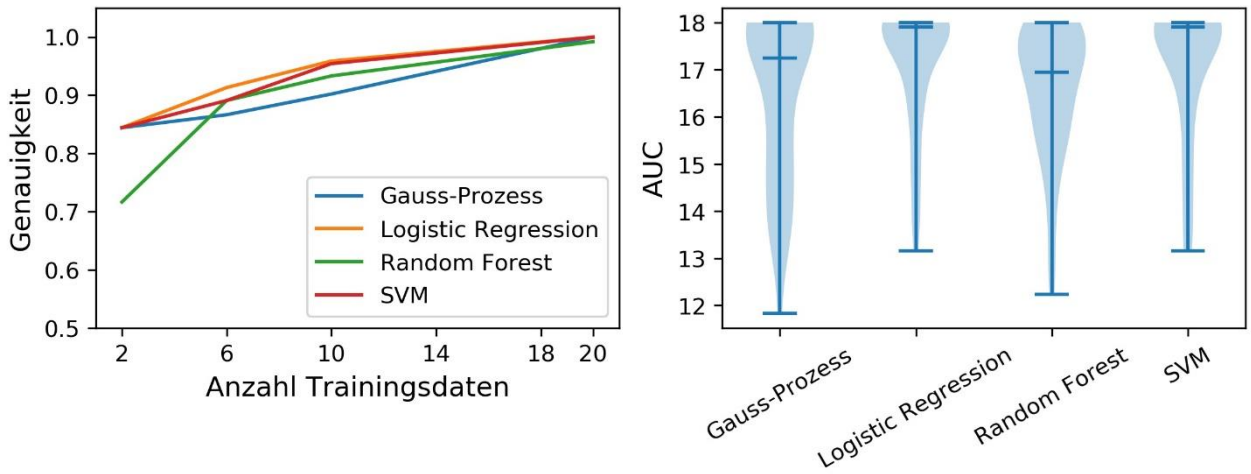


Abbildung 46: Links: Darstellung der gemittelten Genauigkeiten aus 50 Wiederholungen in Abhängigkeit der 4 Datensets mit 2, 6, 10 und 20 Trainingsinstanzen. Rechts: Darstellung der Area Under Curve (AUC)-Werte aus 50 Wiederholungen mit je unterschiedlichen Trainingsinstanzen

Da für jedes Trainingsdatenset 50 Wiederholungen durchgeführt werden, ist jeweils der Mittelwert der Genauigkeit als Liniendiagramm abgebildet. Jede Linie repräsentiert damit die durchschnittliche Genauigkeit der einzelnen Klassifikatoren.

Daraus geht hervor, dass die Genauigkeit der einzelnen Modelle mit steigender Anzahl an Trainingsdaten größer wird. Dieser Trend ist bei allen Klassifikatoren zu beobachten. Die durchweg höchsten Genauigkeiten werden von der *Logistischen Regression* erreicht. Obwohl für N = 2 bei keinem der Modelle Hyperparameter angepasst werden, wird von der *Logistischen Regression*, der *Support Vector Maschine* und dem *Gauß-Prozess* eine mittlere Genauigkeit von 0,85 erreicht. Für N = 20 beträgt die mittlere Genauigkeit aller Modelle ungefähr 1, sodass nahezu alle Testinstanzen der jeweils richtigen Kategorie zugeordnet werden. In einer produktiven Anwendung des CVD müsste der/die Ingenieur*in für zuverlässige Prädiktionen mit einer Genauigkeit über 0,9 dennoch zwischen 10 und 20 Simulationen manuell klassifizieren, was einen beträchtlichen Aufwand darstellen würde.

Die Kurven der gemittelten Genauigkeiten stellen das Konvergenzverhalten in Abhängigkeit der Anzahl verwendeter Trainingsdaten übersichtlich dar. Sie eignen sich jedoch nicht für die Analyse der Streuungen innerhalb der 50 einzelnen Wiederholungen mit den unterschiedlichen Trainingsinstanzen. Eine einzelne Kurve (ein trainiertes Modell mit bestimmter Konfiguration an Trainingsinstanzen) kann jedoch durch einen einzigen Wert beschrieben werden, indem das Integral unter ihr berechnet wird. Das Ergebnis wird als Area Under Curve (AUC) bezeichnet. Das Ziel ist demnach, einen möglichst hohen Wert des AUC zu erreichen. Ein einzelner Wert lässt sich übersichtlicher dazu verwenden, die Streuungen in den 50 Wiederholungen für jeden Algorithmus darzustellen. Aus diesem Grund werden in Abbildung 46 rechts ergänzend Verteilungsdiagramme des AUC betrachtet.

Hierzu werden zunächst die Genauigkeiten der 50 Wiederholungen für die verschiedenen Datensets der Größe nach sortiert. Jeweils die schlechtesten Ergebnisse der vier Datensets bilden eine Kurve. Durch dieses Vorgehen, werden auch die schlechtesten Kombinationen für die Konvergenzkurven und damit die geringsten AUC-Werte abgebildet. Im Anschluss daran kann für jede der Kurven das Integral berechnet und in einem Verteilungsdiagramm dargestellt werden. Als Ergebnis liegt für jeden Klassifikator ein Verteilungsdiagramm bestehend aus 50 Werten für den dazugehörigen AUC vor, woraus die durch die variierenden Trainingsinstanzen hervorgerufene Streuung ersichtlich wird. Für eine ideale Konvergenzkurve, die für sämtliche Datensetgrößen Genauigkeiten von 1 liefert, beträgt der AUC den Wert 18.

Aus Abbildung 46 rechts geht hervor, dass für jeden Klassifikator der einzelnen 50 Experimente AUC-Werte nahe 18 vorliegen. Das bedeutet, dass die Kombination aus PCA und ML-Klassifikator in bestimmten Trainingssetkonfigurationen dazu in der Lage ist, auch bereits mit 2 Trainingsinstanzen neue Simulationen zuverlässig zu klassifizieren. Andernfalls könnte kein AUC von 18 vorliegen, wenn nicht auch für N = 2 die Genauigkeit 1 beträgt. Gleichzeitig gibt es einige Fälle, in denen die Klassifikatoren ein sehr schlechtes Ergebnis liefern, indem niedrige AUCs von 12 oder weniger vorliegen. Diese Fälle treten bei allen Klassifikatoren auf. Der Median befindet sich bei sämtlichen Algorithmen in einem ähnlichen Wertebereich und schwankt zwischen 17 und 17,8. Die geringste Streuung bei den höchsten Werten des AUC liegt bei der *Logistischen Regression* vor.

Zusammenfassend lässt sich festhalten, dass die Genauigkeit der Modelle mit steigender Anzahl an Trainingsdaten zunimmt. Zwischen den einzelnen Klassifikatoren sind dabei keine wesentlichen Unterschiede erkennbar. In diesem einfachen Beispiel, in dem zwei diskrete Crashmodi vorliegen, sind 20 kategorisierte Dateninstanzen für eine hinreichend genaue Prädiktion notwendig. In komplizierteren Anwendungsszenarien mit komplexerem Crashverhalten ist mit einem flacheren Verlauf der Konvergenzkurven zu rechnen, was dazu führt, dass noch mehr Daten vom Anwender manuell kategorisiert werden müssten, um hinreichend hohe Prädiktionsgenauigkeiten zu erreichen.

7.5 Ergebnisse von Simsiam

Im Folgenden werden die Hyperparameter der Modifikations-Techniken für *SimSiam* untersucht. Diese werden schrittweise nacheinander betrachtet und verbessert. Zunächst wird der Einfluss der Batch-Größe untersucht.

7.5.1 Einfluss der Batch-Größe

Abbildung 47 zeigt die Resultate für die vier verschieden großen Datensets 1 - 4 in jeweils einzelnen Verteilungsdiagrammen. Links oben sind die Ergebnisse für zwei Trainingsinstanzen und die Batch-Größe 2 dargestellt. Das Verteilungsdiagramm aus den 50 Wiederholungen des Experiments zeigt, dass die höchstmögliche Genauigkeit 1 erreicht wird. Dies bedeutet, dass *SimSiam* dazu in der Lage ist, selbst bei lediglich 2 Trainingsinstanzen sämtliche 18 Simulationen aus den Testdaten richtig zu klassifizieren. Die meisten Ergebnisse liegen um den Median 0,5 verteilt. Eine Genauigkeit von 0,5 entspricht einem Modell, das keinerlei Struktur in den Daten gelernt hat und dessen Ergebnisse daher nicht besser als die einer zufälligen Schätzung sind.

Oben rechts sind die Ergebnisse für das zweite Datenset mit 6 Trainingsinstanzen abgebildet. Entsprechend werden die Batch-Größen 2, 3 sowie 6 betrachtet. Aus der Darstellung ist ersichtlich, dass für alle drei Batch-Größen Genauigkeiten von 1 erzielt werden. Bei der Betrachtung der unteren Grenze fällt auf, dass der niedrigste Wert 0,45 bei der Batch-Größe 3 auftritt. Die beiden anderen Parameter erzielen ca. 0,5. Bei der Betrachtung der Median Werte und der Häufigkeitsverteilungen fällt jedoch ein eindeutiger Trend auf. Während für die Batch-Größe 2 der Median bei 0,79 liegt, steigt er kontinuierlich mit größer werdender Batch-Größe an, sodass er bei einer Batch-Größe 3 den Wert 0,88 und bei einer Batch-Größe von 6 den Wert 0,95 erreicht. Auch die Verteilungen verschieben sich zugunsten höherer Genauigkeiten mit steigender Batch-Größe. Aus diesen Gründen ist die Batch-Größe 6 den anderen beiden Werten vorzuziehen und wird für die Experimente im nächsten Kapitel für das zweite Datenset verwendet.

Die Ergebnisse für das dritte Datenset mit 10 Trainingsinstanzen sind unten links abgebildet. Es werden die Batch-Größen 2, 5 und 10 untersucht. Erneut erreichen die besten Prädiktionen eine Genauigkeit von 1. Der Median liegt für alle drei Batch-Größen auf einem ähnlichen Niveau von 0,94. Anhand dieses Wertes kann keine Entscheidung getroffen werden, welche Batch-Größe die besten Ergebnisse liefert. Die Verteilungen geben allerdings Aufschluss darüber, dass bei einer Batch-Größe 10 mehr Experimente höhere Genauigkeiten erzielen als bei den anderen beiden Batch-Größen. Darüber hinaus treten die schlechtesten Ergebnisse mit einer Genauigkeit von 0,5 bei der Batch-Größe 2 und 5 auf. Die niedrigste Genauigkeit bei einer Batch-Größe von 10 beträgt

dagegen 0,6. Aus diesen Gründen wird im Folgenden für das dritte Datenset die Batch-Größe 10 verwendet.

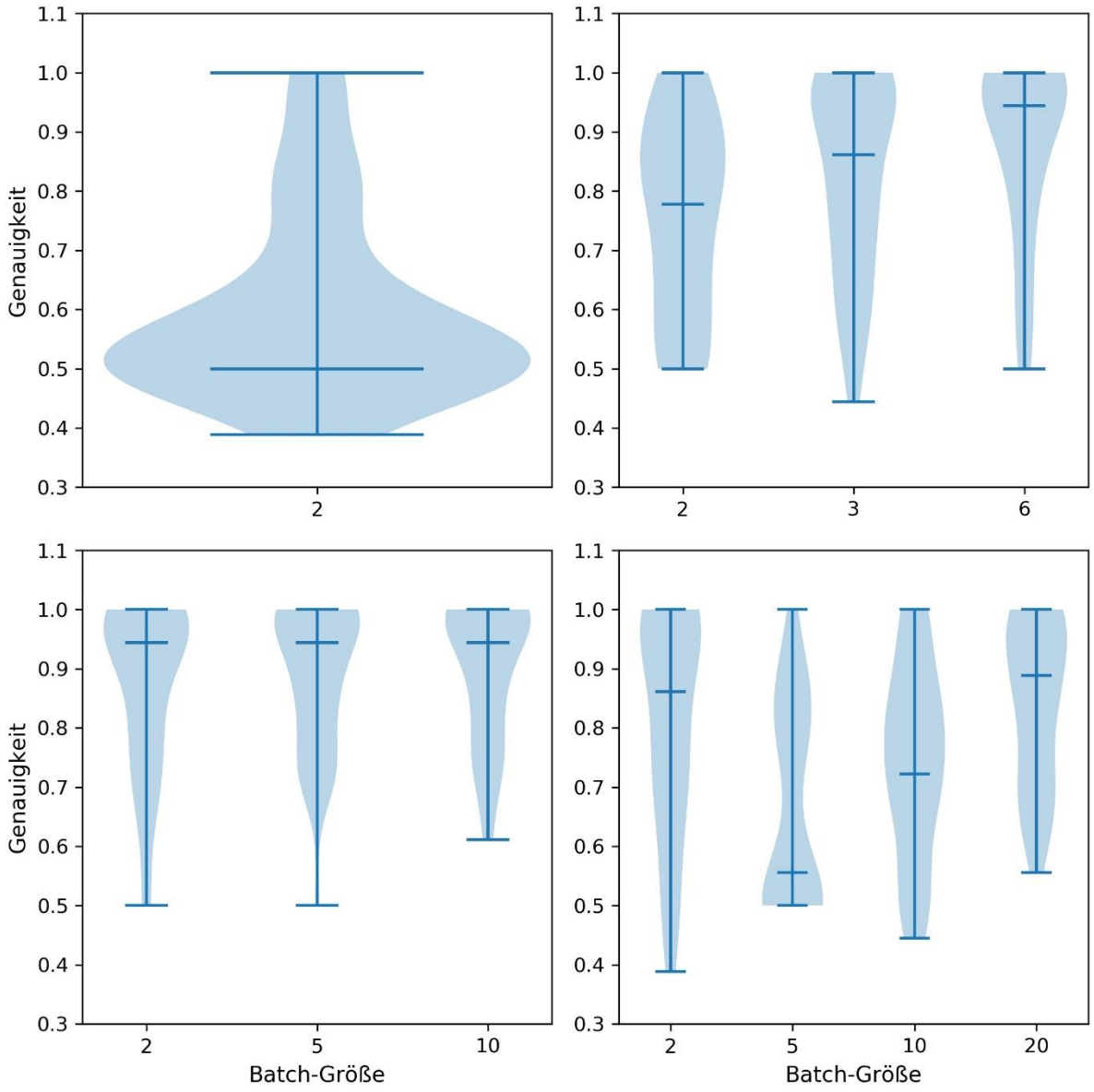


Abbildung 47: Darstellung der Genauigkeit für 50 Wiederholungen mit unterschiedlichen Instanzen aus dem Trainingsset für verschiedene Batch-Größen. Verteilungsdiagramme dargestellt mit kleinsten und größten Werten sowie dem Median. Oben links für Datenset 1 mit N=2, oben rechts für Datenset 2 mit N = 6, unten links für Datenset 3 mit N = 10 und unten rechts für Datenset 4 mit N = 20

Die Ergebnisse für das vierte Datenset mit 20 Trainingsinstanzen sind in Abbildung 47 unten rechts abgebildet. Es werden die Batch-Größen 2, 5, 10 und 20 untersucht. Erneut ist es möglich, die Genauigkeit 1 für sämtliche Batch-Größen zu erreichen. Bei der Betrachtung des Medians fällt auf, dass dieser bei der Batch-Größe 2 den Wert 0,87 aufweist, bei der Batch-Größe 5 sein Minimum 0,55 erreicht und anschließend mit größer werdender Batch-Größe wieder zunimmt und den Wert 0,7 bzw. das Maximum 0,9 für eine Batch-Größe 10 bzw. 20 aufweist. Daraus geht hervor, dass sich mit einer Batch-Größe 2 und 20 die besten Ergebnisse erzielen lassen. Jedoch

fällt die Streuung für eine Batch-Größe 20 geringer aus. Gleichzeitig liegen die schlechtesten Werte mit 0,52 immer noch oberhalb der schlechtesten Werte für eine Batch-Größe 2 mit 0,39. Dies zeigt, dass sich mit der Batch-Größe 20 die besten Ergebnisse erzielen lassen. Aus diesem Grund wird in den folgenden Experimenten für das vierte Datenset die Batch-Größe mit dem Wert 20 verwendet.

7.5.2 Einfluss der Reihenfolge der Modifikations-Techniken

Nachdem im vorangegangenen Abschnitt geeignete Batch-Größen für die vier verschiedenen Datensets identifiziert wurden, folgt in diesem Abschnitt die Untersuchung, wie sich die Modifikations-Techniken auf die Ergebnisse auswirken. Dabei wird zunächst die Modifikations-Technik *Duplizieren&Löschen entlang der Voxel* (*DLV*) als Variante von *Duplizieren&Löschen entlang der Zeit* (*DLZ)* analysiert. Im Anschluss daran wird der Einfluss der Reihenfolge der Modifikationen bewertet.

In Abbildung 48 sind die Ergebnisse des Ausgangsmodells *DLZ*, sowie dessen Varianten dargestellt. Um diese statistisch abzusichern, werden erneut für jede der drei Varianten 50 Wiederholungen durchgeführt, in denen sich jeweils die Trainingsinstanzen unterscheiden.

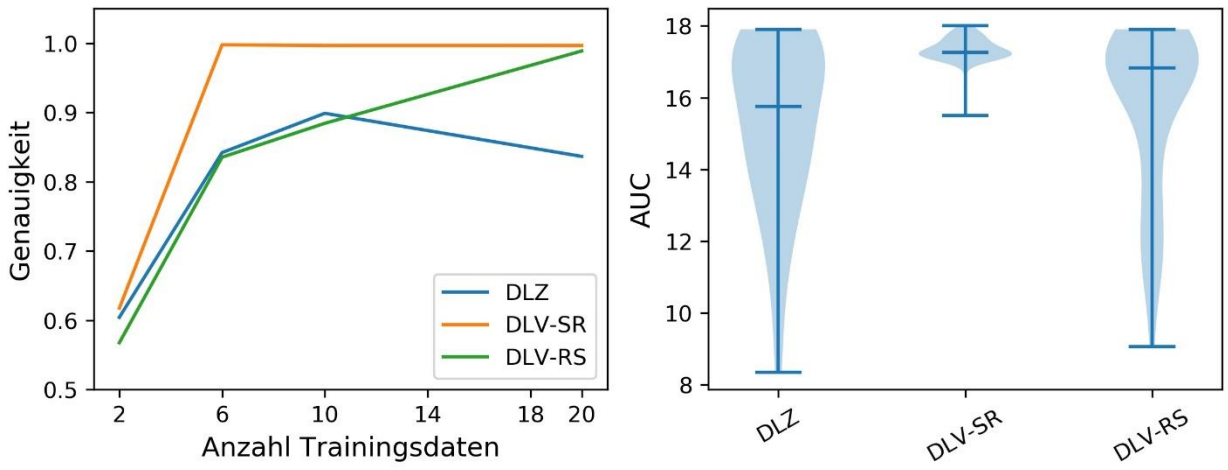


Abbildung 48: Einfluss der Reihenfolge der Modifikations-Techniken für DLZ, DLV-SR und DLV-RS. Darstellung der gemittelten Genauigkeiten aus 50 Wiederholungen in Abhängigkeit der 4 Datensets mit 2,6,10 und 20 Trainingsinstanzen. Rechts: Darstellung der Area Under Curve (AUC)-Werte aus 50 Wiederholungen mit je unterschiedlichen Trainingsinstanzen

In Abbildung 48 links sind die gemittelten Genauigkeiten für die vier Trainingsdatensets aufgetragen. Die Kurve von *DLZ* schneidet dabei am schlechtesten ab, da die Genauigkeit insbesondere für N = 20 deutlich niedriger als bei den anderen Varianten ausfällt. Darüber hinaus steigt die Genauigkeit bei den anderen Vorgehensweisen kontinuierlich an, während sie bei dem Ausgangsmodell *DLZ* nach N = 10 sogar abfällt. Bei *DLV* fällt auf, dass bereits ab N=6 die Genauigkeit 1 erreicht wird.

Erneut wird der AUC-Wert als weiteres Auswertungskriterium herangezogen, um die Streuung der 50 Wiederholungen mit unterschiedlichen Trainingsinstanzen zu bewerten. Die Ergebnisse sind in Abbildung 48 rechts in Form von Verteilungdiagrammen dargestellt. Zusätzlich sind der kleinste und größte Wert sowie der Median durch horizontale Balken markiert. Für die Ausgangsvariante *DLZ* liegen sowohl das niedrigste Minimum als auch der kleinste Median vor. Die Verteilung ist im Vergleich zu den anderen Varianten besonders im Bereich niedriger AUC-Werte bauchiger. Der höchste Median und die geringste Streuung zeigen sich bei *DLV-SR*. Für DLV-RS liegt ein geringfügig niedrigerer Median vor. Gleichzeitig ist die Streuung deutlich größer, da vermehrt niedrige AUC Werte auftreten.

Aus der Abbildung geht hervor, dass die Methode *DLV* besser abschneidet als *DLZ*. *SimSiam* extrahiert aus den Varianten der Bilder, die durch die Modifikation entlang der Voxel entstehen, mehr relevante Informationen über die Daten. Eine Veränderung entlang der Voxel entspricht einer modifizierten Verteilung der plastischen Dehnungen auf dem Bauteil, während bei *DLZ* zufällige Zeitschritte dupliziert werden. In dem betrachteten Beispiel spielt jedoch die Information, wo auf dem Bauteil Deformationen auftreten, eine für die zugrunde liegende Klassifikationsaufgabe weitaus wichtigere Rolle als der zeitliche Verlauf. Daher liegen für *DLV* die besseren Ergebnisse vor.

Bei dem Vergleich zwischen den Reihenfolgen der Modifikationen fällt auf, dass die Variante in der zuerst eine *Skalierung* durchgeführt und im Anschluss das *Rauschen* hinzugefügt wird höhere Genauigkeiten bei geringerer Streuung erzielt als die Variante, in der die Modifikationen andersherum erfolgen. Hier zeigt sich der Vorteil, das Eingangsbild zuerst zu skalieren und im Anschluss das Rauschen hinzuzufügen. Durch dieses Vorgehen wird die Stärke des Rauschens lediglich durch die für die Gaußverteilung festgelegte Varianz bestimmt. Bei der anderen Vorgehensweise wird dagegen das Rauschen zusätzlich mit dem Faktor aus der Skalierung multipliziert, was einen größeren Einfluss des Rauschens auf die Stärke der Modifikation des originalen Bildes verursacht. Bei der Betrachtung der Genauigkeits-Kurven aus Abbildung 48 links zeigt sich, dass die stärkere Modifikation durch *DLV-RS* überwiegend die Genauigkeit der kleineren Datensets betrifft. Während die Genauigkeit für N = 20 nahezu unbeeinflusst den Wert 1 beträgt, sinken die Genauigkeiten für kleinere Datensets ab. Insbesondere bei kleinen Datensets sollte daher darauf geachtet werden, dass keine zu großen Variationen des originalen Bildes durch die Modifikations-Techniken erzeugt werden. In einem späteren Abschnitt wird der Einfluss des Rauschens detaillierter untersucht, indem verschiedene Werte für dessen Varianz betrachtet werden. In den folgenden Analysen wird die *DLV*-Modifikation verwendet, in der zuerst die Skalierung und im Anschluss daran das Rauschen erfolgt.

7.5.3 Parametereinfluss Skalierung

In diesem Abschnitt wird der Einfluss durch die Stärke der Skalierung untersucht. Während bisher der Faktor 8 verwendet wurde, wird dieser in den folgenden Experimenten sukzessive bis auf den Wert 2 verkleinert (siehe Tabelle 9). Darüber hinaus wird eine Variante berechnet, in der die Skalierung als Modifikations-Technik gänzlich weggelassen wird, sodass in Set 1 (Tabelle 5) nur noch das *Duplizieren&Löschen* und in Set 2 nur noch das *Rauschen* angewendet werden. Dadurch soll ersichtlich werden, wie sich das Weglassen der Modifikations-Technik auswirkt. Die Ergebnisse sind in Abbildung 49 dargestellt.

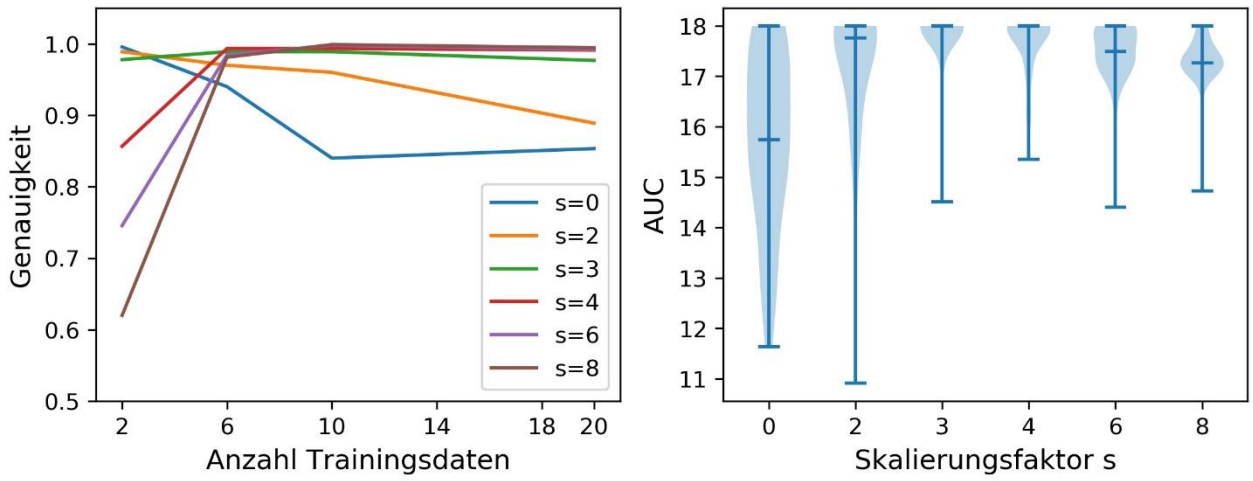


Abbildung 49: Einfluss der Stärke der Skalierung. Darstellung der gemittelten Genauigkeiten aus 50 Wiederholungen in Abhängigkeit der 4 Datensets mit 2, 6, 10 und 20 Trainingsinstanzen. Rechts: Darstellung der Area Under Curve (AUC)-Werte aus 50 Wiederholungen mit je unterschiedlichen Trainingsinstanzen

Für s = 8 zeigt das Liniendiagramm in Abbildung 49 links bei N = 2 eine Genauigkeit von 0,62. Ab einer Datensetgröße von N = 6 nimmt diese einen Wert von ungefähr 1 an. Desto kleiner der Skalierungsfaktor gewählt wird, desto besser wird die Genauigkeit für N = 2. Während in den meisten Fällen die Genauigkeit für die anderen Datensets auf einem sehr hohen Niveau nahe 1 bleibt, sinkt sie bei s = 2 mit größer werdendem Datenset kontinuierlich ab. Dieses Verhalten kann auch in der Variante, in der die Skalierung gänzlich weggelassen wird, beobachtet werden. Hier sinkt die Qualität noch drastischer ab als bei s = 2.

Ein ähnliches Bild zeigt sich bei der Betrachtung der Verteilungsdiagramme in Abbildung 49 rechts. Während der Median für die Variante ohne Skalierung am kleinsten ist, steigt er mit größer werdendem s bis auf den Wert 18 für s = 3 und s = 4 an. In diesen beiden Varianten sind die Streuungen gleichzeitig am geringsten und die mittleren Werte des AUC am höchsten. Mit größer werdender Skalierung sinken die AUC-Werte wieder ab und die Streuung in den Ergebnissen nimmt zu.

Zusammenfassend lässt sich festhalten, dass ein kleiner Skalierungsfaktor bei kleinen Datensets bessere Ergebnisse liefert als ein hoher. In Analogie an das *Signal-Rausch-Verhältnis* aus der Signalverarbeitung können die relevante Information in den Trainingsdaten als Signal und die Variationen durch die Modifikationen als Rauschen interpretiert werden. Bei großen Skalierungsfaktoren dominiert für kleine N das Rauschen das Signal, sodass im Extremfall das Risiko besteht, dass nach den Modifikationen zu wenig relevante Information aus den modifizierten Daten extrahiert werden kann und daraus schlechte Prädiktionen resultieren. Andersherum sieht es bei großen N aus, in denen bei gleichem Rauschen ein größeres Signal vorliegt, da es mehr Trainingsinstanzen gibt. Dadurch ist es dem Modell dennoch möglich durch die größere Anzahl an Trainingsdaten auch bei hohen Skalierungsfaktoren gute Repräsentationen aus den Bildern zu extrahieren.

Bei kleinen Skalierungsfaktoren verhält es sich andersherum. Für kleine Trainingsdatensets liegt ein größeres Verhältnis zwischen Signal und Rauschen vor. Im Extremfall, in dem die Skalierung als Modifikations-Technik weggelassen wird, liegen noch höhere Genauigkeiten vor als bei s = 2. An dieser Stelle sei angemerkt, dass dennoch Variationen auftreten, da noch die Modifikations-Techniken c und n angewendet werden. In diesem Beispiel scheinen die Variationen hierdurch auszureichen, um die relevanten Informationen aus den Trainingsdaten zu extrahieren. Für große Trainingsdatensets liegt ein noch viel größeres Verhältnis zwischen dem Signal und dem Rauschen vor. Hier scheinen die Variationen durch die Modifikations-Techniken für die Beispieldaten zu klein zu sein. Aus diesem Grund werden schlechtere Repräsentationen gelernt und die Klassifikationsgenauigkeit nimmt ab. Generell kann festgehalten werden, dass bei größeren Trainingsdatensets ein höherer Einfluss durch die Modifikationen sichergestellt werden sollte. Liegen dagegen nur wenige Trainingsdaten vor, sollten die Variationen kleiner gewählt werden.

Im Vergleich zu den anderen Skalierungsfaktoren ergeben sich für s = 3 durchschnittlich die höchsten Genauigkeiten sowie die geringsten Streuungen. Aus diesem Grund wird die Skalierung im Folgenden auf den Wert s = 3 festgelegt.

7.5.4 Parametereinfluss Duplizieren&Löschen

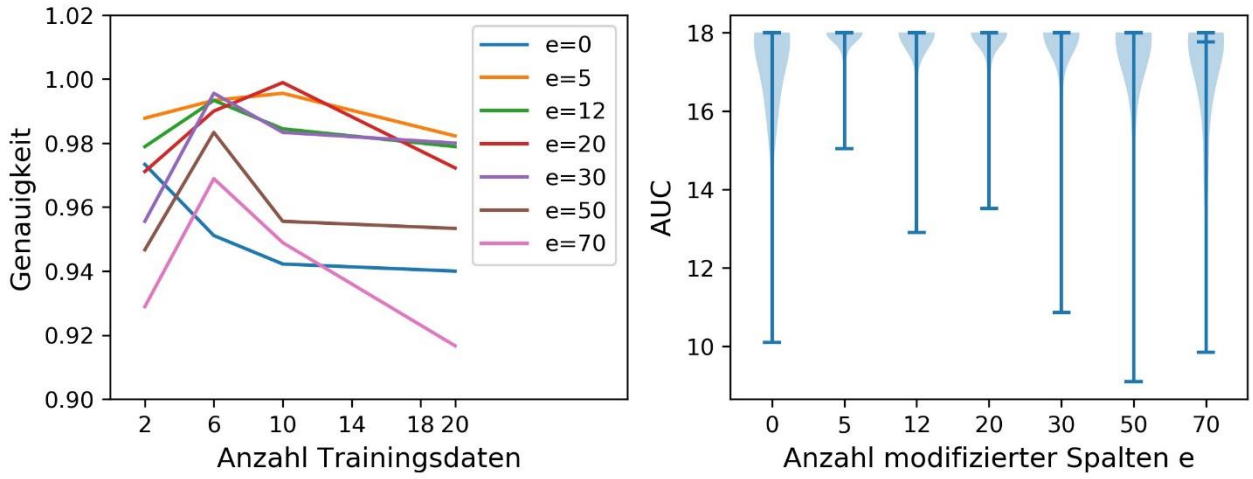


Abbildung 50: Einfluss der Anzahl zu modifizierender Voxel mittels der Modifikations-Technik Duplizieren&Löschen entlang der Voxel (DLV). Darstellung der gemittelten Genauigkeiten aus 50 Wiederholungen in Abhängigkeit der 4 Datensets mit 2,6,10 und 20 Trainingsinstanzen. Rechts: Darstellung der Area Under Curve (AUC)-Werte aus 50 Wiederholungen mit je unterschiedlichen Trainingsinstanzen

Im Folgenden wird der Einfluss des Parameters *e* für die Modifikation *Duplizieren&Löschen* untersucht. Ein niedriger Wert sorgt für geringe Modifikationen des originalen Bildes, während ein hoher Wert zu starken Veränderungen führt. Er kontrolliert dabei die Anzahl zu modifizierender Spalten des originalen Bildes. Die Ergebnisse aus den Experimenten sind in Abbildung 50 visualisiert.

Die mittleren Genauigkeiten sind in der linken Grafik dargestellt und zeigen, dass die Kurven mit hohen Werten für *e* die niedrigsten Genauigkeiten aufweisen. Im Fall von N = 2 zeigt sich eine

stetige Verbesserung der Genauigkeiten bis für e = 5 das Maximum 0,98 erreicht wird. Wird diese Modifikations-Technik dagegen weggelassen, verringert sich die gesamte Kurve über alle *N* hinweg und liefert deutlich schlechtere Genauigkeiten. Diese sinkt sogar mit steigender Anzahl verwendeter Trainingsdaten. Es sei jedoch angemerkt, dass die mittleren Genauigkeiten für sämtliche der Parameter über 0,9 liegen und damit einen hohen Wert gerade im Hinblick auf die Benchmark Ergebnisse darstellen.

Die Streuungen der einzelnen Experimente können in Abbildung 50 rechts abgelesen werden. Bei der Betrachtung der einzelnen Minima fällt auf, dass sich diese mit zunehmendem Wert von *e* verschlechtern. Bei e=20 und e=50 wird dieser Trend unterbrochen. Dies kann auf stochastische Einflüsse durch die Stichprobengröße (lediglich 50 Wiederholungen) zurückgeführt werden. Die gesamten Verteilungen zeigen dagegen den eindeutigen Effekt, dass die Streuung mit größer werdendem *N* zunimmt und die mittleren AUC-Werte abnehmen. Die besten Resultate zeigen sich auch hier bei e = 5, indem der größte Mittelwert, das höchste Minimum sowie die kleinste Streuung auftreten. Wird die Modifikation-Technik weggelassen, verschlechtern sich die Ergebnisse und die Verteilung verlagert sich in Richtung niedrigerer AUC-Werte.

Bei der Modifikation *Duplizieren&Löschen* wird das Bild durch Duplizieren und Löschen zufälliger Voxel verändert. Die physikalische Interpretation dieser Modifikation entspricht einem veränderten Deformationsverhalten des Bauteils. In einem Bereich wo vorher keine plastischen Dehnnugen waren, können nach der Modifikation plastische Deformationen durch das zufällige Duplizieren auftreten. Auch der umgekehrte Fall is möglich.

In dem ausgewählten Beispiel sind insgesamt 250 Voxel vorhanden. Von diesen bis zu 70 Spalten zu duplizieren stellt eine starke Modifikation der originalen Bildes dar. In dem gewählten Anwendungsfall sind diese Modifikationen zu stark, sodass niedrigere mittlere Genauigkeiten sowie stärkere Streuungen in den einzelnen Wiederholungen vorliegen, da das Modell die für die Klassifikation relevanten Eigenschaften nur eingeschränkt lernt.

Wird das Verfahren dagegen ganz weggelassen, sinkt auch hier die Ergebnisgüte. Es treten zu geringe Variationen in den Modifikationen auf, da nur noch die Skalierung und das Rauschen verwendet werden. Dementsprechend wird der Lernprozess negativ beeinflusst, sodass schlechtere Repräsentationen vorliegen, die es dem Klassifikator erschweren, die richtige Kategorie zu prädizieren. Da bei e = 5 die besten Ergebnisse mit den geringsten Streuungen vorliegen wird dieser Parameter in den nachfolgenden Analysen verwendet.

7.5.5 Parametereinfluss Rauschen

Nachdem in den vorherigen Abschnitten der Parameter e der Modifikation *Duplizieren&Löschen* untersucht wurde, erfolgt nun die Analyse, wie sich die Skalierung der Varianz um den Faktor r als Hyperparameter der Modifikation *Rauschen* auf die Resultate des CVD auswirkt. Die Ergebnisse der gemittelten Genauigkeit sind in Abbildung 51 links dargestellt.

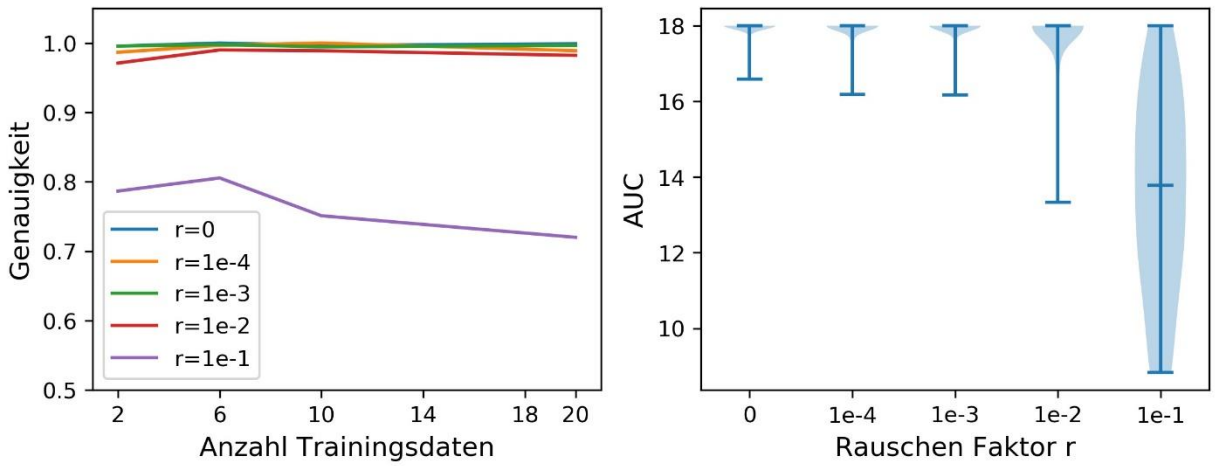


Abbildung 51: Links: Einfluss des Hyperparameters r für die Modifikations-Technik Rauschen. Darstellung der gemittelten Genauigkeiten aus 50 Wiederholungen in Abhängigkeit der 4 Datensets mit 2,6,10 und 20 Trainingsinstanzen. Rechts: Darstellung der Area Under Curve (AUC)-Werte aus 50 Wiederholungen mit je unterschiedlichen Trainingsinstanzen

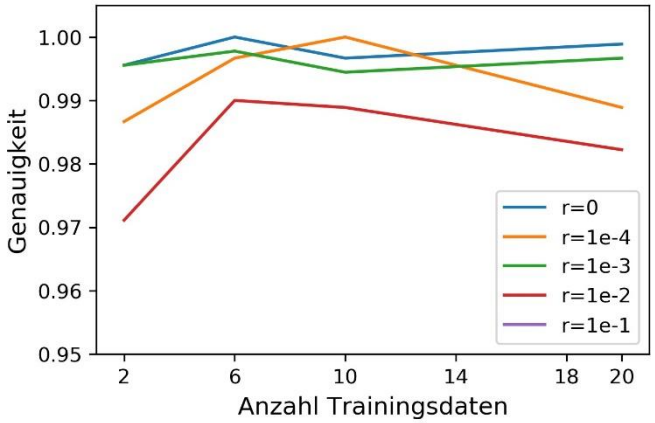


Abbildung 52: Einfluss des Hyperparameters r für die Modifikations-Technik Rauschen. Darstellung der gemittelten Genauigkeiten aus 50 Wiederholungen in Abhängigkeit der 4 Datensets mit 2, 6, 10 und 20 Trainingsinstanzen.

Der Parameter r = 0,1 liefert dabei die niedrigsten Genauigkeiten über sämtliche Datensets hinweg. Ausgehend von 0,78 sinkt die Genauigkeit mit steigender Anzahl an Trainingsdaten auf den Wert 0,73 ab. Bei einer Verkleinerung auf den Faktor s = 0.01 wird die mittlere Genauigkeit deutlich erhöht. Dieser Parameter wurde auch schon in den bisherigen Experimenten aus den vorangegangenen Kapiteln verwendet. Ausgehend von 0,97 steigt die Genauigkeit zunächst auf den Wert 0,99 und verringert sich leicht bei N = 20. Da sämtliche Genauigkeiten (außer für r=0,1) auf einem sehr hohen Niveau über 0,96 verlaufen, zeigt Abbildung 52 einen vergrößerten Ausschnitt der Genauigkeiten, um die Unterschiede zwischen den einzelnen Faktoren *r* zu verdeutlichen. Bei einer weiteren Verkleinerung des Faktors r, erhöht sich die Genauigkeit geringfügig weiter. Die Resultate für r = 1e-3 und die Variante, in der das Rauschen weggelassen wird, unterscheiden sich dabei kaum und verlaufen beide oberhalb einer Genauigkeit von 0,99. Die Experimente ohne Rauschen weisen geringfügig höhere Genauigkeiten auf.

Abbildung 51 rechts stellt die Verteilungsdiagramme der AUC-Werte für die unterschiedlichen Skalierungsfaktoren dar. Auch hier wird deutlich, dass sich die Ergebnisse durch einen kleineren

Skalierungsfaktor stetig verbessern lassen. Während bei s = 1e-1 die niedrigsten Minima der AUCs auftreten, liegt gleichzeitig der niedrigste Median mit 14 vor. Darüber hinaus ist die Streuung in diesem Fall am größten. Mit kleiner werdendem Faktor verschieben sich die Minima in Bereiche höherer AUC Werte. Die Streuung in den 50 Experimenten nimmt gleichzeitig ab. Die meisten Experimente weisen scharfe Verteilungen bei hohen AUC Werten auf. Lediglich einzelne Ausreißer treten dabei auf. Die geringste Streuung und höchsten Minima zeigen sich in den Experimenten, in denen auf das Rauschen als Modifikations-Technik gänzlich verzichtet wird. Es lässt sich jedoch feststellen, dass zwischen r = 1e-3, r = 1e.4 und dem Verzicht auf das Rauschen keine wesentlichen Unterschiede festzustellen sind.

Aus den bisherigen Darstellungen lässt sich zusammenfassend festhalten, dass das Rauschen bei einer zu großen Varianz die zugrundeliegende Information in den Daten dominiert. Dadurch ist *SimSiam* nicht mehr dazu in der Lage, die für die Unterscheidung zwischen den beiden Kategorien wichtigen Informationen zu extrahieren. Während bisher der Skalierungsfaktor der Varianz r = 1e-2 verwendet wurde, kann durch einen geringeren Wert die Streuung in der Ergebnissen weiter reduziert werden. Diese Verbesserung der Genauigkeiten liegt jedoch auf einem eher kleinen Niveau, da durch das sequentielle Vorgehen bei der Parameteroptimierung, das meiste Potential bereits durch die Skalierung und das Duplizieren gehoben wurde und damit die erst später betrachteten Parameter über einen geringeren Einfluss auf die Qualität der Ergebnisse verfügen. Abbildung 51 zeigt, dass in den Experimenten ohne Rauschen, weitere Verbesserungen erzielt werden. Die Idee hinter *SimSiam* ist jedoch, durch verschiedene Modifikations-Techniken bessere Datenrepräsentationen zu erlernen. Unterschiedliche Aufgabenstellungen erfordern unterschiedlich starke Modifikationen, um aus wenigen Trainingsdaten eine möglichst hohe Genauigkeit zu erzielen. Vor dem Hintergrund der Übertragbarkeit des Modells auf andere Aufgabenstellungen sowie zur Sicherstellung einer größeren Variation in den Modifikationen, wird das Rauschen dennoch beibehalten. Die Verteilungen und mittleren Genauigkeiten sind bei r = 1e-4 und r = 1e-3 sehr ähnlich. Da jedoch bei r = 1e-4 lediglich 41 Experimente einen AUC von 18 erreichen, bei r = 1e-3 dagegen 46, wird im Folgenden die Varianz für das Rauschen auf den Wert r = 1e-3 festgelegt.

7.5.6 Zusammenfassung Modifikations-Techniken

Abbildung 53 fasst die Ergebnisse der schrittweisen Verbesserung der Hyperparameter für *SimSiam* zusammen. Im linken Diagramm zeigt das Ausgangsmodell die niedrigsten Genauigkeiten für die einzelnen Datensets. Ausgehend von 0,61 bei N=2 steigt sie zunächst bis auf 0,89 bei N=10 an und sinkt für N=20 auf 0,85 ab. Die Variante, in der die Modifikation DLV anstelle von DLZ verwendet wird, sorgt für die größten Verbesserungen für N>6. Bei diesen wird die maximale Genauigkeit 1 erreicht. Für N=2 legt sie dagegen nur leicht auf 0,63 zu. Die Reduzierung des Skalierungsfaktors von 8 auf 3 sorgt bei N=2 für die größte Verbesserung. Für größere *N* nimmt die Genauigkeit geringfügig ab. Schon mit diesen Ergebnissen ist das Ziel, mit möglichst wenig Trainingsdaten möglichst gute Ergebnisse zu erzielen erreicht. Die Modifikationen *Duplizieren&Löschen* und *Rauschen* verbessern die Resultate nur geringfügig, sorgen jedoch dafür, dass die maximale Ergebnisgüte 1 nahezu für alle 50 Wiederholungen erzielt wird.

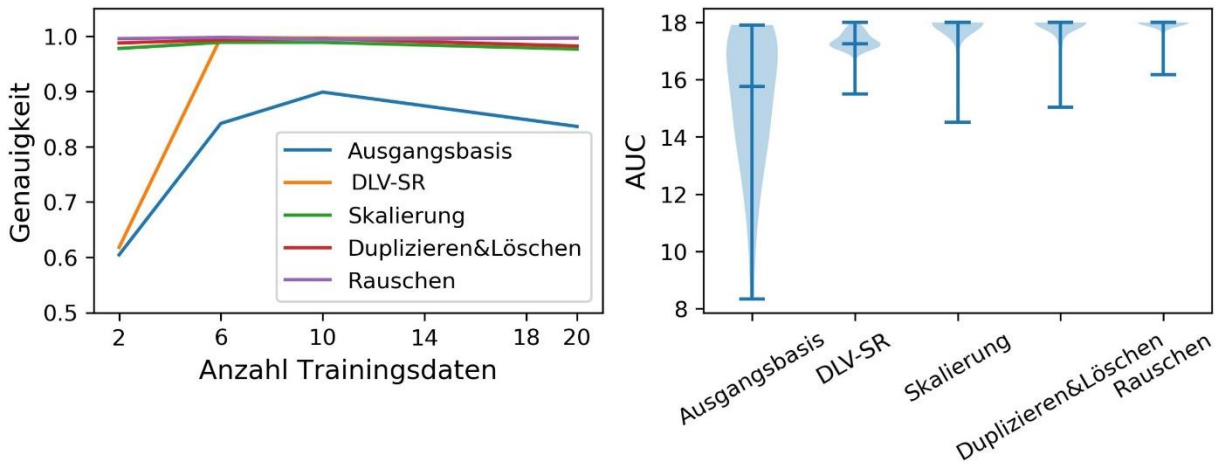


Abbildung 53: Links: Zusammenfassung der einzelnen Modifikations-Techniken. Darstellung der gemittelten Genauigkeiten aus 50 Wiederholungen in Abhängigkeit der 4 Datensets mit 2,6,10 und 20 Trainingsinstanzen. Rechts: Darstellung der Area Under Curve (AUC)-Werte aus 50 Wiederholungen mit je unterschiedlichen Trainingsinstanzen

Die Verteilungsdiagramme in Abbildung 53 rechts bestätigen diese Beobachtungen und geben gleichzeitig Aufschluss über die Beeinflussung der Streuung innerhalb der 50 Wiederholungen. Die einzelnen Anpassungen der Parameter für die Modifikationen verbessern die AUC-Werte sukzessive, indem sich die Streuungen zugunsten höherer AUC Werte verschieben und gleichzeitig verringern. Im Folgenden werden die Batch-Größe 20, sowie die *DLV*-Modifikation genutzt, in der zuerst die Skalierung und im Anschluss daran das Rauschen erfolgt. Für die Skalierung wird s=3, für die Anzahl zu löschender Spalten (DLV) e=5 und für das Rauschen r=1e-3 verwendet.

7.6 Vergleich der Benchmark-Methode mit *SimSiam*

Bei der Untersuchung der Benchmark-Methode wurden 50 Wiederholungen mit jeweils unterschiedlichen Trainingsinstanzen durchgeführt. Bei der bisherigen Analyse von *SimSiam* wurden dagegen 50 Wiederholungen mit jeweils denselben Trainingsinstanzen durchgeführt, um zusätzliche Streuungen durch variierende Trainingsdatensets zu verhindern und damit die durch die Hyperparameter hervorgerufenen Effekte besser analysieren zu können. Für den Vergleich mit der Benchmark-Methode werden jetzt auch für SimSiam 50 Wiederholungen mit jeweils unterschiedlichen Trainingsinstanzen berechnet. Die 50 unterschiedlichen Trainingsdatensets sind jedoch dieselben für beide Methoden, sodass ein fairer Vergleich sichergestellt ist. Mit diesem Vorgehen wird gleichzeitig erreicht, dass die Verfahren sowohl einfache als auch schwierige Klassifikationsaufgaben lösen müssen. Dadurch wird ein allgemeineres Bild im Hinblick auf deren Ergebnisgüte erreicht.

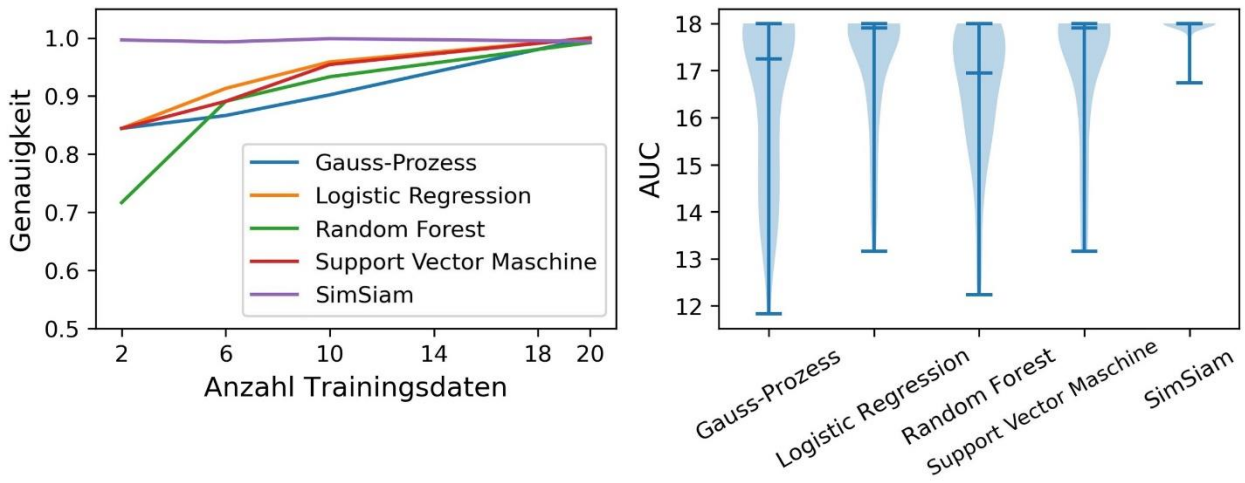


Abbildung 54: Links: Vergleich der Benchmark-Methode mit den Resultaten von SimSiam. Darstellung der gemittelten Genauigkeiten aus 50 Wiederholungen in Abhängigkeit der 4 Datensets mit 2,6,10 und 20 Trainingsinstanzen. Rechts: Darstellung der Area Under Curve (AUC)-Werte aus 50 Wiederholungen mit je unterschiedlichen Trainingsinstanzen

Abbildung 54 links zeigt die mittleren Genauigkeiten der einzelnen Modelle. Dabei ist der Unterschied zwischen der Benchmark-Methode und *SimSiam* deutlich zu erkennen. Beim Benchmark fallen die Genauigkeiten bei kleinen Datensets deutlich niedriger im Vergleich zu *SimSiam aus*. Bei N = 2 liegen sie bei Ersterem im Mittel bei 0,84 während sie bei Letzterem den Wert 1 aufweisen. Dies bedeutet, dass die Kategorie von den ungesehenen 18 Testinstanzen vom Benchmark in 13 bzw. von *SimSiam* in sämtlichen Fällen korrekt prädiziert wird. Damit ist für *SimSiam* die Eingangs formulierte Aufgabenstellung erreicht, aus so wenig wie möglich kategorisierten Daten möglichst hohe Prädiktionsgenauigkeiten zu erzielen. Die Genauigkeit steigt mit größer werdendem *N* bei den Benchmark Modellen kontinuierlich an, bis sie bei N = 20 den Wert 0,99 erreicht. Die Kurve für *SimSiam* weist dagegen durchweg ab N = 2 eine mittlere Genauigkeit von ca. 1 auf.

Ein ähnliches Bild zeigt sich bei der Analyse von Abbildung 54 rechts. Im Vergleich zu *SimSiam* treten bei den *Machine Learning* Modellen größere Streuungen, niedrigere Minima sowie kleinere Median Werte auf. Der geringste Wert von AUC beträgt für *SimSiam* 16,8. Lediglich in fünf von 50 Experimenten liegt ein niedrigerer AUC-Wert als der höchste mit 18 vor.

Die Analysen zeigen, dass sich mittels *SimSiam* deutlich bessere Resultate erzielen lassen als bei der gezeigten Benchmark-Methode, indem die PCA mit klassischen ML-Algorithmen kombiniert wird. Die Anzahl kategorisierter Daten für das Training des Modells kann reduziert werden, wobei dennoch hohe Genauigkeiten erzielt werden. Es wird aufgezeigt, dass *SimSiam* dazu in der Lage ist, durch unterschiedliche Modifikations-Techniken selbst im Falle von nur zwei Trainingsinstanzen eine Genauigkeit von 1 zu erzielen. Gleichzeitig wird gezeigt, dass im Vergleich zum Benchmark dabei die Streuung in den Ergebnissen bei Wiederholung der Experimente mit unterschiedlichen Subsets der Trainingsdaten drastisch reduziert wird. Damit ist das Eingangs definierte Ziel erreicht, aus möglichst wenig Trainingsdaten eine möglichst hohe Prädiktionsgenauigkeit zu erzielen.

7.7 Bewertung des Diskretisierungseinflusses

In den vorangegangenen Abschnitten werden unterschiedliche Datensetgrößen betrachtet und gezeigt, dass selbst für N = 2 Genauigkeiten von 1 erzielt werden können. In diesem Abschnitt wird der Einfluss der Diskretisierung für N = 2 analysiert. Hierfür werden erneut 50 Wiederholungen mit jeweils unterschiedlichen Trainingsinstanzen durchgeführt, um unterschiedlich schwierige Konfigurationen der Trainingsdaten abzuprüfen.

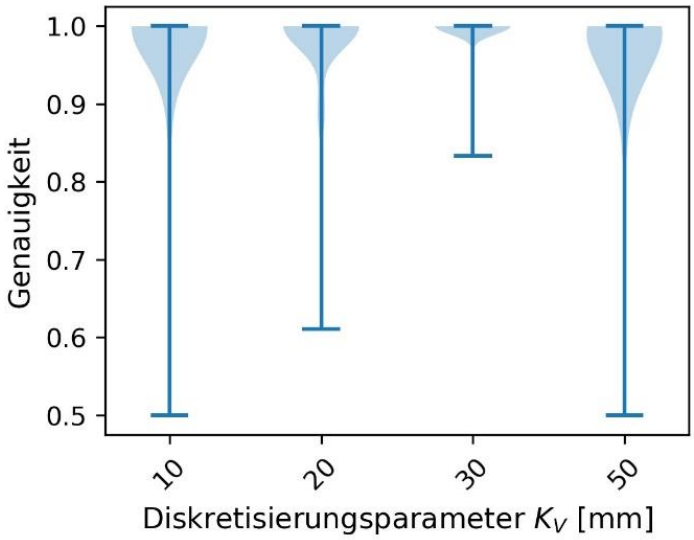


Abbildung 55: Darstellung der Genauigkeit in Form von Verteilungsdiagrammen in Abhängigkeit der verwendeten Voxel-Diskretisierung mit dem Parameter K_V in mm

Abbildung 55 zeigt die Genauigkeiten der jeweiligen Diskretisierungen in Form von Verteilungsdiagrammen. Daraus geht hervor, dass die Ergebnisse (ausgenommen K_V = 50) für kleinere Diskretisierungen eine größere Streuung und dabei gleichzeitig kleinere Minimalwerte aufweisen. Bei einer Diskretisierung von 10 mm beträgt die minimale Genauigkeit 0,5, was einem Modell entspricht, das die zugrundeliegende Aufgabenstellung nicht gelernt hat. Dagegen liefert das schlechteste Experiment für eine Diskretisierung von 30 mm eine wesentlich höhere minimale Genauigkeit von 0,83. Während die Median Werte für alle Diskretisierungen den Wert 1 aufweisen, steigt der Mittelwert mit steigender Diskretisierung von 0,97 für K_V = 10 mm über 0,98 für K_V = 20 mm bis auf 0,98 für K_V = 30 mm an. Für die Diskretisierung K_V = 50 mm sinkt der Mittelwert auf 0,96. Bei diesen Beobachtungen sei angemerkt, dass jeweils nur einzelne Experimente/Konfigurationen des Trainingssets für diese Unterschiede in den Verteilungsdiagrammen verantwortlich sind. Der überwiegende Teil der Ergebnisse liefert jedoch für sämtliche Diskretisierungen eine Genauigkeit von 1.

In den bisherigen Untersuchungen wurde eine Diskretisierung mit 20 mm verwendet. Wird diese auf beispielsweise 10 mm verkleinert, werden die Simulationsergebnisse genauer abgebildet, was gleichzeitig die Dimension der Voxelrepräsentation erhöht. Da diese auf der horizontalen Achse in den Bildern für *SimSiam* verwendet wird, steigt damit die Dimension und Komplexität der Aufgabenstellung an. Im Allgemeinen müssen beim Deep Learning für komplexere

Aufgabenstellungen (z.B. größere Bilder) komplexere Netzwerkarchitekturen gewählt werden (z.B. Anzahl Filter pro Faltungsschicht und Anzahl Faltungsschichten erhöhen). Da jedoch bei den unterschiedlichen Diskretisierungen keine Anpassungen an der Architektur der einzelnen Netzwerke vorgenommen werden, verschlechtern sich die Ergebnisse bei kleinerer Diskretisierung geringfügig.

Für gröbere Diskretisierungen verbessern sich dagegen die Genauigkeiten der Prädiktionen. Die Diskretisierung wirkt wie ein Filter, der die Informationen mehrerer FE aus der Umgebung zusammenfasst. Durch größere Diskretisierungen sinkt die Anzahl an Voxeln, mit denen ein Bauteil abgebildet wird. Dadurch liegen kleinere Bilder als Eingangsdaten für *SimSiam* vor. Da auch für die Diskretisierung 30 mm keine Anpassungen gegenüber dem Modell für K_V = 20 mm vorgenommen werden, verbessern sich in diesem Fall die Genauigkeiten, da dasselbe Modell eine weniger komplexe Aufgabenstellung lösen muss. Die Diskretisierung kann jedoch nicht beliebig groß gewählt werden, da sie sonst zu viele für die Klassifikation notwendige Informationen über das Crashverhalten des Bauteils herausfiltert. In diesem Fall sinken die Genauigkeiten wie bei K_V = 50mm zu sehen ist.

7.8 Zusammenfassung

In diesem Kapitel wird eine Methode vorgestellt, mit deren Hilfe ein vorgegebenes Crashverhalten automatisiert wiedererkannt werden kann. Nachdem der/die Ingenieur*in das Bauteilverhalten einzelner Simulationen kategorisiert hat, kann ein Modell trainiert werden, das ein vorgegebenes Crashverhalten in neuen Simulationen zuverlässig wiedererkennen kann.

In den Untersuchungen werden grundlegende Eigenschaften von *SimSiam* betrachtet. Einen wesentlichen Bestandteil dieses Verfahrens stellen die Modifikations-Techniken dar, deren Reihenfolge sowie Parameter analysiert wurden. Sind die dadurch erzeugten Variationen der originalen Eingangsdaten zu klein oder zu groß, werden niedrige Genauigkeiten erzielt. Da verschieden große Datensets betrachtet wurden, kann ein Zusammenhang zwischen der Stärke der Modifikation und der Anzahl an Trainingsinstanzen hergestellt werden. Dabei wurde festgestellt, dass bei kleinen Datensets auch eine entsprechend geringere Modifikation im Vergleich zu größeren Datensets vorgenommen werden sollte.

Das Ziel, dem/der Ingenieur*in so wenig Aufwand wie möglich bei dem Kategorisieren der Daten zu bereiten, ist erreicht, indem ein Modell entwickelt wurde, das dazu in der Lage ist, aus wenigen kategorisierten Daten eine möglichst hohe Klassifikationsgenauigkeit zu erzielen. Dabei wird sogar der optimale Fall abgedeckt, bei welchem nur eine Instanz pro zu prädizierender Kategorie zur Verfügung gestellt werden muss. Dies stellt den geringstmöglichen Aufwand für den Anwender dar. Damit übertreffen die Ergebnisse von *SimSiam* die des Benchmarks deutlich.

Darüber hinaus wurde der Einfluss der Diskretisierung untersucht. Kleine Diskretisierungen führen zu einer höheren Auflösung der originalen FE-Daten. Dies steigert jedoch die Komplexität der von *SimSiam* zu lösenden Aufgabenstellung. Dementsprechend sind komplexere Architekturen des neuronalen Netzes erforderlich. Wird die Diskretisierung dagegen zu groß gewählt, kann es passieren, dass die für die Klassifikationsaufgabe erforderliche Information herausgefiltert wird, was in geringeren Genauigkeiten der Prädiktionen resultiert.

Es sei angemerkt, dass Hyperparameter eines Algorithmus wie allgemein im *Machine Learning/Deep Learning* üblich, an die zugrunde liegende Aufgabenstellung angepasst werden müssen. Für *SimSiam* sind das insbesondere die Architekturen der neuronalen Netze und die Modifikations-Techniken sowie deren Hyperparameter. Die Analyse der Modifikations-Techniken innerhalb dieses Kapitels bietet dem Anwender der Methode eine Hilfestellung, bei unzureichenden Ergebnissen die richtigen Parameter zu verbessern.

8 Visualisierung des Crashverhaltens mittels Dimensionsreduktion

Mithilfe der Dimensionsreduktion können in einer übersichtlichen und interaktiven Visualisierung einer Vielzahl von Simulationen Informationen über Modelländerungen mit dem resultierenden Crashverhalten eines Bauteils verknüpft werden, sodass der/die Ingenieur*in bei der Ableitung konstruktiver Verbesserungen unterstützt wird. Hierbei werden die hochdimensionalen Daten der Crashsimulation in einen niedrigdimensionalen Raum projiziert. Dadurch ist es möglich, das Crashverhalten eines Bauteils aus hunderten Simulationen in einem einzelnen Streu- oder Liniendiagramm darzustellen (siehe Abbildung 56, Seite 128). Dabei liefert dieses Verfahren einen Überblick über das Crashverhalten und zeigt auf einen Blick, welche Simulationen ein ähnliches bzw. unähnliches Verhalten aufzeigen. Durch eine Erweiterung der Diagramme um interaktive Funktionalitäten, können die Ergebnisse dabei sogar mit weiteren Informationen verknüpft werden. Beispielsweise können Metadaten zur Simulation oder Bilder und Videos des betrachteten Bauteils interaktiv dargestellt werden. Dies unterstützt den/die Ingenieur*in bei der Definition von konstruktiven Änderungen zur Verbesserung des Crashverhaltens des Fahrzeugs. Durch die Visualisierung der Crashsimulationen ist es gleichzeitig möglich, die Datenbasis nach Simulationen mit ähnlichem Crashverhalten zu durchsuchen. Dadurch gelangt der Anwender schneller an relevante Informationen, die er für die Weiterentwicklung der Modelle benötigt. In diesem Kapitel werden verschiedene Algorithmen zur Dimensionsreduktion miteinander verglichen und dabei insbesondere untersucht, ob nichtlineare Verfahren gegenüber linearen Vorteile bieten.

8.1 Anforderungen an den Algorithmus zur Dimensionsreduktion

Die Anforderungen aus den Kapiteln 1.2 und 2.6 an die Dimensionsreduktion lassen sich wie folgt zusammenfassen:

- Ergebnisse müssen für den/die Ingenieur*in intuitiv verständlich sein (ähnliches Crashverhalten sollte in ähnlichen Bereichen der niedrigdimensionalen Darstellung abgebildet werden)
- Gesamtes zeitliches Verhalten soll berücksichtigt werden
- Möglichst geringer Informationsverlust gegenüber den originalen Daten
- Geringe Berechnungszeit für interaktive Datenanalyse
- Effekte eventuell vorhandener Hyperparameter sollen intuitiv und nachvollziehbar sein
- Resultate sollten möglichst deterministisch sein.

8.2 Vorgehensweise

Ein wichtiger Schritt der Dimensionsreduktion liegt zunächst in einer entsprechenden Datenvorverarbeitung. Es wird daher untersucht, inwiefern dabei das zeitliche Verhalten berücksichtigt werden kann. Hierbei werden das sogenannte *OPioS* und *OLioS* Verfahren miteinander verglichen (siehe auch die Vorveröffentlichung zu dieser Arbeit (Kracker et al. 2020)). Erläuterungen zur Funktionsweise dieser beiden Verfahren folgen später im Kapitel.

Darüber hinaus ist es für die meisten Algorithmen zur Dimensionsreduktion notwendig, die Daten vorab entsprechend zu skalieren. Daher werden in der Analyse auch unterschiedliche Methoden zur Datenskalierung betrachtet. Neben der Standardisierung werden die Zentrierung und Normierung untersucht.

Im Anschluss daran werden die Kriterien vorgestellt, mit denen die verschiedenen Algorithmen zur Dimensionsreduktion miteinander verglichen werden können. Neben der qualitativen Analyse mithilfe der Visualisierungen der eingebetteten Daten wird die Qualität der Dimensionsreduktion mittels des in Kapitel 2.7 vorgestellten Qualitätskriteriums Q_{NX} bewertet. Darüber hinaus wird die Zeitkomplexität der einzelnen Algorithmen analysiert. Im Anschluss erfolgt der Vergleich vier verschiedener Algorithmen zur Dimensionsreduktion. Die Ergebnisse der linearen PCA werden denen der nichtlinearen Verfahren Isomap, t-SNE und UMAP gegenübergestellt. Der Vergleich der Algorithmen erfolgt anhand der undiskretisierten Daten aus der Robustheitskampagne. Zuletzt wird der Einfluss der Voxel-Diskretisierung für ausgewählte Algorithmen untersucht.

Es werden für die PCA, Isomap sowie t-SNE die Implementierungen aus dem Paket *Scikit-learn* (Bisong 2019) verwendet. Für UMAP wird die Implementierung der Entwickler (McInnes et al. 2018) genutzt. Sämtliche Berechnungen erfolgen auf einem „*Intel® Xeon® Gold 6234“* Prozessor mit 3.30GHz. Sämtliche Algorithmen, die eine parallele Ausführung auf der CPU erlauben, werden mit 14 Kernen berechnet.

Für die nachfolgenden Analysen wird dabei beispielhaft der bereits in Kapitel 4 vorgestellte *Längsträger* verwendet. Dessen Crashverhalten eignet sich für die Auswertung der Dimensionsreduktion in besonderem Maße, da im Wesentlichen drei unterschiedliche Crashmodi vorliegen, die teilweise ineinander übergehen. Das Bauteil besteht aus 2286 FE, zu welchen die zeitliche Information von 62 Zeitschritten verfügbar ist. Es werden die Simulationen aus dem Datenset DIN001 betrachtet. Die Ergebnisse der Dimensionsreduktionsalgorithmen und die Auswirkungen der Hyperparameter werden zunächst anhand der originalen, anstatt anhand der diskretisierten Daten erläutert, um zunächst Effekte durch die Diskretisierung auszuschließen. Dies ist zulässig, da zum einen die grundlegenden Effekte der unterschiedlichen Algorithmen auf sowohl die originalen als auch die diskretisierten Daten übertragbar sind. Zum anderen bilden die Ergebnisse der originalen Daten die Referenz für die späteren Untersuchungen der Auswirkungen durch die Diskretisierung. Daher werden zunächst die Informationen sämtlicher 2286 FE für die Dimensionsreduktion verwendet.

8.3 Datenvorverarbeitung

Bevor die Dimensionsreduktion durchgeführt werden kann, müssen die vorliegenden Daten entsprechend vorverarbeitet werden. Hierzu zählt neben einer geeigneten Datenskalierung die Berücksichtigung des zeitlichen Verhaltens.

8.3.1 Berücksichtigung des zeitlichen Verhaltens

Die Analysen in diesem Kapitel erfolgen exemplarisch anhand der PCA, sind jedoch auf jeden anderen Algorithmus übertragbar. Die Dimension wird auf drei Hauptkomponenten reduziert, um die Daten in Streudiagrammen zu visualisieren und auszuwerten.

Bei *OPioS* wird das zeitliche Verhalten neben der Information der plastischen Dehnung für jedes FE als Merkmal der Daten verwendet. Dementsprechend bilden 50 Instanzen mit jeweils 2286*62 Merkmalen das zu untersuchende Datenset. In Abbildung 56 links sind die ersten drei Hauptkomponenten in einem Streudiagramm dargestellt. Jeder Punkt entspricht dabei einer Simulation (*OPioS*: One Point is one Simulation = Ein Punkt entspricht einer Simulation). Die Einfärbung der Punkte erfolgt anhand der ersten drei Hauptkomponenten, deren Werte für die RGB-Farbcodierung verwendet werden. Durch die farbliche Darstellung der einzelnen Simulationen fällt es dem Betrachter leichter, Gruppierungen von Punkten zu identifizieren, die zu einem bestimmten Crashverhalten gehören.

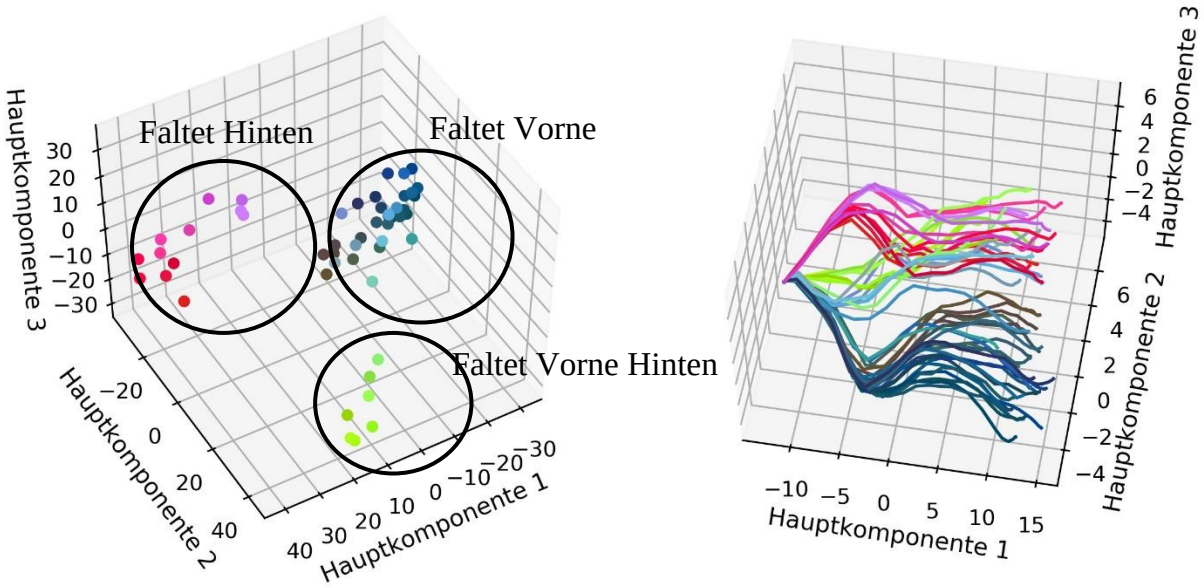


Abbildung 56: Dimensionsreduzierte Darstellung der 50 Simulationen mittels der PCA und der OPioS- (links) beziehungsweise OLioS- (rechts) Datenrepräsentation in drei Dimensionen

Die Punkte im hinteren rechten Bereich des Streudiagramms entsprechen den Simulationen, bei denen die initiale Deformation des *Längsträgers* im vorderen Bereich (crashzugewandte Seite) beginnt und sich im Verlauf des Crashs nach hinten (crashabgewandte Seite) ausbreitet. Üblicherweise wird ein solches Crashverhalten durch konstruktive Maßnahmen erwirkt, da dies die Robustheit im Lastpfad erhöht. Dieses Deformationsverhalten wird im Folgenden als *Faltet Vorne* bezeichnet.

Die Simulationen im linken Cluster zeigen das gegensätzliche Crashverhalten, bei dem die Deformation im hinteren Bereich des Bauteils beginnt und sich während des Crashs nach vorne ausbreitet. Dies stellt ein unerwünschtes Crashverhalten dar, weil es Instabilitäten in den Lastpfad bringt. Im Folgenden werden diese Simulationen als *Faltet Hinten* bezeichnet.

Im vorderen Bereich des Streudiagramms finden sich die Simulationen, bei denen ein Mischverhalten der beiden soeben beschriebenen Crashmodi auftritt. Die Deformation des Bauteils beginnt gleichzeitig im vorderen und hinteren Bereich. Im Verlauf des Crashs wird

dadurch das gesamte Bauteil gestaucht. Dieses Verhalten wird im Folgenden mit der Bezeichnung *Faltet Vorne&Hinten* beschrieben.

Aus Abbildung 56 geht hervor, dass die PCA dazu in der Lage ist, die unterschiedlichen Crashverhalten übersichtlich darzustellen. Der Anwender erhält eine schnelle Übersicht über die Verteilung der Daten und die auftretenden Crashmodi. Die farbliche Darstellung hebt dabei noch einmal die Einbettung der Simulationen in den unterschiedlichen Bereichen des Diagramms hervor.

Die PCA mit der *OPioS* Datenrepräsentation wird für die nachfolgenden Untersuchungen als Ausgangsbasis verwendet. Um die unterschiedlichen Datenvorverarbeitungsschritte und Algorithmen zur Dimensionsreduktion mit dieser vergleichen zu können, werden für alle noch folgenden Untersuchungen die Farbcodierung aus der PCA *OPioS* aus Abbildung 56 links verwendet. Dies soll dem Leser eine schnelle Einordnung darüber ermöglichen, wie das Crashverhalten der anderen Verfahren im Vergleich zur PCA *OPioS* im niedrigdimensionalen Raum eingebettet wird. Da er nun weiß, welche Farbe welches Crashverhalten darstellt, wird er so bei der Wiedererkennung der auftretenden Crashverhalten unterstützt.

Ein Nachteil der *OPioS* Datenrepräsentation ist, dass aus der niedrigdimensionalen Darstellung die Zeitschritte, zu denen Bifurkationen auftreten, nicht bestimmt werden können. Bei *OLioS* ist diese Möglichkeit dagegen gegeben. Deren Funktionsweise wird deshalb im Folgenden beschrieben.

Im Unterschied zum Vorgehen bei *OPioS* wird jeder Zeitschritt als einzelner Datenpunkt betrachtet. Dementsprechend steigt die Anzahl der Instanzen von 50 auf 50*62. Die Anzahl der Merkmale einer Simulation reduziert sich gleichzeitig von 2286*62 auf 2286. In der Dimensionsreduktion wird bei dieser Vorgehensweise lediglich die Information über die plastischen Dehnungen der FE reduziert. Die Information über das zeitliche Verhalten bleibt erhalten. Erneut wird die Dimension mittels der PCA auf drei Hauptkomponenten reduziert. Für jeden Zeitschritt liegt somit ein Datenpunkt im dreidimensionalen Raum vor. Da bekannt ist, welche Zeitschritte der Datenmatrix zu welcher Simulation gehören, können diese jeweils miteinander verbunden und in einem Liniendiagramm visualisiert werden. Das Ergebnis ist in Abbildung 56 rechts dargestellt. Eine Linie besteht demzufolge aus 62 Datenpunkten, deren Verlauf im 3D-Raum durch die ersten drei Hauptkomponenten aus der PCA festgelegt ist.

In Abbildung 56 rechts ist zu sehen, dass die 50 dargestellten Linien einen gemeinsamen Ursprung besitzen. In diesem sind die ersten fünf Zeitschritte eingebettet, da hier noch keine Deformationen an dem Bauteil vorliegen und sich die Simulationen damit auch noch nicht unterscheiden. Mit Einsetzen der Deformation, verändern sich die plastischen Dehnungen der FE, wodurch Unterschiede zwischen den Simulationen ersichtlich werden und sich die Linien vom Ursprung entfernen. Desto stärker das Bauteil deformiert wird, desto stärker streut das Crashverhalten in den betrachteten Simulationen und die einzelnen Linien verlaufen in unterschiedliche Richtungen. Ab dem Zeitschritt 22 ist eine klare Bifurkation im Crashverhalten sichtbar. Es bilden sich im weiteren Verlauf weitere Bifurkationspunkte und Trends der Linien aus. Anhand der Einfärbung wird ersichtlich, dass es sich bei den verschiedenen Strängen um die aus Abbildung 21 (Seite 58) bekannten drei Crashverhalten *Faltet Vorne*, *Faltet Hinten* und *Faltet Vorne&Hinten* handelt. Die 26 Linien mit dem Crashverhalten *Faltet Vorne* bewegen sich nach

unten rechts, während sich die 12 Linien von *Faltet Hinten* in die entgegengesetzte Richtung nach oben entwickeln. Die 13 Linien mit dem Mischverhalten *Faltet Vorne&Hinten* verlaufen in der Mitte der beiden anderen Stränge, was der intuitiven Einschätzung entspricht, dass sich dieses Verhalten als eine Mischung aus den beiden anderen darstellt. Demnach ist es auch bei *OLioS* möglich, die Bifurkationen ähnlich wie bei der Verwendung von *OPioS* zu visualisieren. Darüber hinaus erlaubt *OLioS* jedoch die Identifikation der Zeitschritte, zu denen Bifurkationen im Crashverhalten auftreten.

Um den nach der Dimensionsreduktion erhaltenen Informationsgehalt in den transformierten Daten zu quantifizieren, kann bei der PCA die kumulierte erklärte Varianz betrachtet werden. Deren Verlauf über der Anzahl der verwendeten Hauptkomponenten hinweg ist in Abbildung 57 für *OPioS* und *OLioS* dargestellt. Für beide Methoden ist eine horizontale Linie eingezeichnet, die die erklärte Varianz für drei Hauptkomponenten markiert. Sie beträgt bei *OPioS* ca. 0,75 und bei *OLioS* ca. 0,33. Da bei *OPioS* als Merkmal neben den plastischen Dehnungen gleichzeitig auch das zeitliche Verhalten zur Beschreibung der Daten verwendet wird, gehen bei der Dimensionsreduktion mehr Informationen als beim *OLioS*-Ansatz verloren. Entsprechend niedriger ist die erklärte Varianz. Dies ist demnach ein weiterer Vorteil der *OLioS* Methode.

Für beide Vorgehensweisen steigen die Kurven monoton an. Bei *OPioS* sind insgesamt 50 Datenobjekte vorhanden, daher können mit 50 Hauptkomponenten sämtliche Eigenschaften der Daten ohne Informationsverlust beschrieben werden (dabei beträgt die erklärte Varianz eins). Bei *OLioS* ist dies wiederum nicht möglich. Aus der Abbildung geht hervor, dass erst bei der Verwendung von 100 Hauptkomponenten 99,9 % der erklärten Varianz erfasst werden. Dies stellt jedoch keinen Nachteil dar, da die Daten mittels zwei oder drei Hauptkomponenten visualisiert werden sollen.

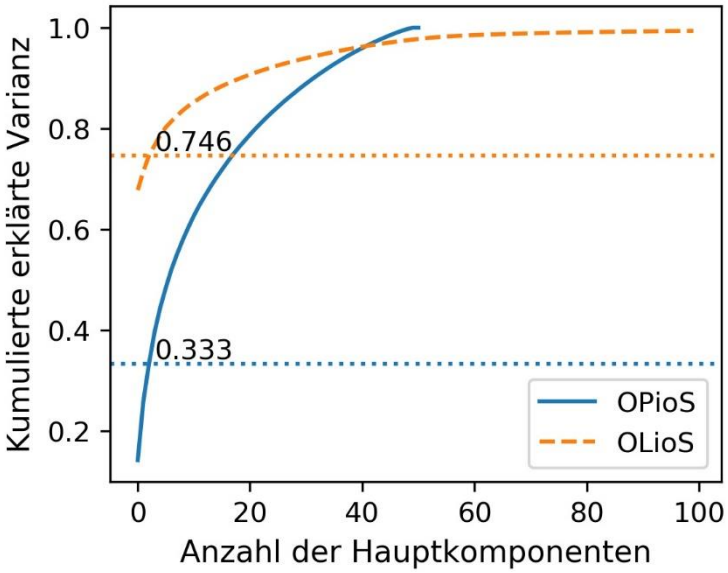


Abbildung 57: Darstellung der kumulierten erklärten Varianz in Abhängigkeit der Anzahl verwendeter Hauptkomponenten für die OPioS- und OLioS-Datenrepräsentation (Kracker et al. 2020)

Trotz der beschriebenen Nachteile des *OPioS*-Verfahrens wird diese Methode in den weiteren Analysen betrachtet. Dadurch, dass jede Simulation durch lediglich einen Punkt beschrieben wird, ist es mit OPioS möglich, einen sehr anschaulichen Überblick über die Verteilung der Crashverhalten im niedrigdimensionalen Raum zu erhalten.

Bisher wurde die Dimension der originalen Daten auf drei Hauptkomponenten reduziert. Für die interaktive Analyse mit entsprechender Visualisierungssoftware eignet sich die dreidimensionale Darstellung grundsätzlich, da durch die Betrachtung aus unterschiedlichen Blickrichtungen die Verteilung der Daten abgeschätzt werden kann. Dem Anwender fällt jedoch bei statischen Darstellungen (wie die Bilder in dieser Arbeit) die Einordnung schwer, wo sich die Punkte beziehungsweise Linien im dreidimensionalen Raum genau befinden. Im Verlauf des Kapitels sollen die Einbettungen von verschiedenen Skalierungsmethoden und Algorithmen verglichen. Da jedes Verfahren eine andere niedrigdimensionale Abbildung erzeugt und im Detail betrachtet werden muss, wäre die Visualisierung in drei Dimensionen für das mit dieser Arbeit zu untersuchende Crashverhalten unvorteilhaft. Daher wird im Folgenden die Dimension der originalen Daten auf zwei Hauptkomponenten reduziert, wodurch die visuelle Interpretation der Ergebnisse vereinfacht werden kann. Aus den Verläufen der erklärten Varianz aus der obigen Abbildung geht zwar hervor, dass dadurch mehr Informationen über die originalen Daten verloren gehen. Der verbleibende Informationsgehalt befindet sich jedoch in einem ähnlichen Wertebereich und auch das Deformationsverhalten des gewählten Beispiels kann in zwei Dimensionen, wie in Abbildung 58 gezeigt, anschaulich dargestellt werden. Die aus den Analysen diesen Kapitels gewonnenen Erkenntnisse lassen sich dennoch auch auf die Dimensionsreduktion mit drei Dimensionen übertragen. An dieser Stelle wird darauf hingewiesen, dass in der praktischen Anwendung jedoch stets eine interaktive dreidimensionale Darstellung verwendet werden sollte, da dadurch noch mehr Informationen über das zugrunde liegende Crashverhalten abgebildet werden können.

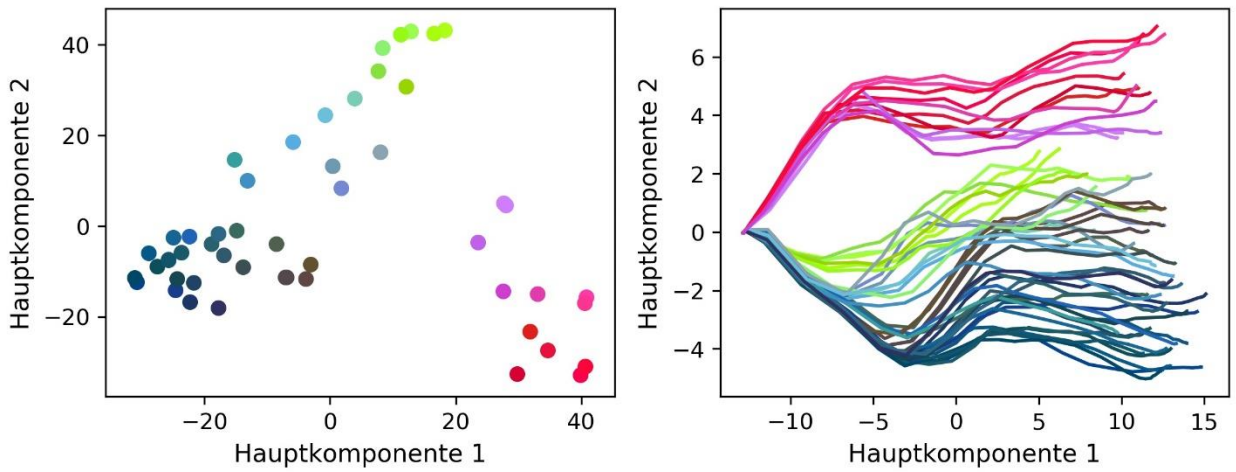


Abbildung 58: Dimensionsreduzierte Darstellung der 50 Simulationen mittels der PCA und der OPioS- (links) beziehungsweise OLioS- (rechts) Datenrepräsentation in zwei Dimensionen. Siehe auch (Kracker et al. 2020)

8.3.2 Skalierung der Daten

Neben der geeigneten Repräsentation der zeitlichen Information gehört zur erforderlichen Datenvorverarbeitung auch die adäquate Skalierung der Daten. In diesem Abschnitt werden daher verschiedene Möglichkeiten zur Datenskalierung vorgestellt.

Diese ist notwendig, damit die einzelnen Merkmale der hochdimensionalen Daten in gleichem Maße bei der Dimensionsreduktion berücksichtigt werden. Würden beispielsweise Merkmale mit unterschiedlichen Einheiten wie Gramm und Kilogramm verarbeitet werden, hätten die Merkmale ein unterschiedliches Gewicht bei der Berechnung der niedrigdimensionalen Darstellung. Das Merkmal, das in der kleineren Einheit gemessen wird, würde eine größere Varianz aufweisen und entsprechend mehr Einfluss auf die Berechnung der niedrigdimensionalen Datenrepräsentation nehmen als das Merkmal mit der größeren Einheit und dementsprechend kleineren Varianz. Aus diesem Grund ist es notwendig, insbesondere bei varianzbasierten Verfahren wie der PCA, die Daten vorab zu skalieren. In dem beschriebenen Beispiel der unterschiedlichen Gewichtseinheiten würde es genügen, die Merkmale in derselben Einheit abzubilden. Neben der Konvertierung der Einheiten existieren weitere Methoden wie beispielsweise die Zentrierung der Daten (Mittelwert wird abgezogen), die Standardisierung (Mittelwert wird abgezogen und das Ergebnis durch die Standardabweichung dividiert) als auch die Normierung (Daten werden auf einen vorgegebenen Wertebereich normiert), deren Einfluss im Folgenden analysiert wird.

Hierzu wird beispielhaft die *OLioS* Datenrepräsentation mit 50*62 Instanzen und 2286 Merkmalen betrachtet. Jedes der Merkmale beschreibt die plastische Dehnung eines FE des Bauteils und damit dieselbe Auswertungsgröße in derselben Einheit. Da dieselben Einheiten vorliegen, müssen diese nicht konvertiert werden. Es stellt sich dennoch die Frage, inwiefern eine Zentrierung beziehungsweise Standardisierung jedes einzelnen Merkmals die Ergebnisse der Dimensionsreduktion beeinflussen.

Bei der PCA Implementierung von *Scikit-learn* wird bereits jedes Merkmal für sich zentriert. Die **Zentrierung** liefert dabei dieselbe qualitative Visualisierung im Vergleich zu den unskalierten Daten mit dem Unterschied, dass die Mittelwerte der neuen Hauptkomponenten jeweils null betragen. Um die vier Algorithmen zur Dimensionsreduktion vergleichen zu können wird vorab eine PCA durchgeführt, um die Berechnungszeit zu optimieren. Daher müssen die Daten hier nicht in einem gesonderten Schritt vorab zentriert werden.

Die getrennte **Standardisierung** jedes einzelnen Merkmals sorgt im Gegensatz zur Zentrierung für eine Verzerrung der Daten und damit für eine andere Einbettung der Ergebnisse. Dies ist in Abbildung 59 links dargestellt. Erneut wird die Einfärbung der einzelnen Linien aus dem vorherigen Kapitel gemäß der *OPioS* Datenrepräsentation übernommen, damit die unterschiedlichen Crashverhalten in der niedrigdimensionalen Darstellung leichter identifiziert werden können.

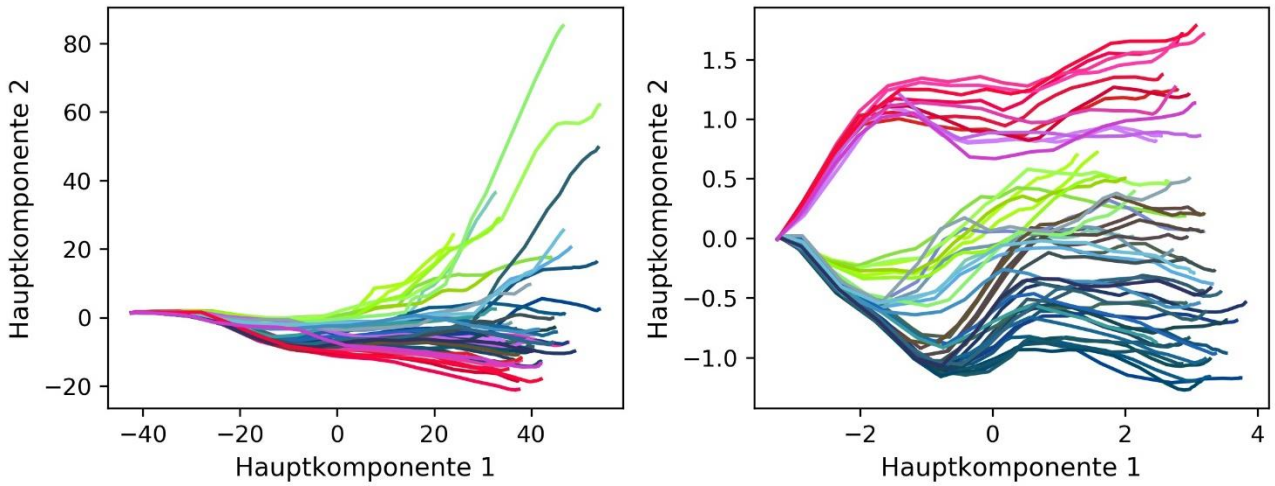


Abbildung 59: Dimensionsreduzierte Darstellung der 50 Simulationen mittels der PCA und der OLioS-Datenrepräsentation für unterschiedliche Skalierungsmethoden. Links: Standardisierung, Rechts: Normierung

Die Linien entspringen nach wie vor einem gemeinsamen Ursprung. Die Bifurkationen zum Zeitschritt 22 sind dagegen nicht mehr eindeutig zu erkennen. Gleichzeitig hebt sich das Verhalten *Faltet Hinten* nicht mehr so deutlich von den anderen Crashmodi ab. Stattdessen tritt eine starke Streuung zu den späteren Zeitschritten auf. Auch nach einer detaillierten Analyse der deformierten Bauteile sind jedoch keine wesentlichen Unterschiede zwischen den Simulationen in den letzten Zeitschritten zu identifizieren.

Dadurch, dass jedes FE getrennt für sich standardisiert wird, erhalten Bauteilbereiche mit kleinen Streuungen zwischen den Simulationen eine größere Bedeutung, während Bereiche mit großen Streuungen an Bedeutung verlieren. Dies sogt für eine Verzerrung der Daten und der für den/die Ingenieur*in relevanten Informationen. Diese veränderte Gewichtung der einzelnen Bauteilbereiche ist jedoch nicht gewünscht, da durch sie die relevanten Effekte mit den größten Streuungen verloren gehen. Demzufolge sollte nicht jedes Element für sich betrachtet standardisiert werden. Diese Beobachtungen und Erläuterungen lassen sich auch auf die *OPioS* Methode übertragen.

Eine andere Art der Datenskalierung ist die Normierung zwischen null und eins. Diese Art der Skalierung erfolgt nicht getrennt für jedes einzelne FE, sondern im Kontext sämtlicher FE und ändert nichts an der Verteilung der Daten. Dadurch kommt es bei dieser Methode wie auch bei der Zentrierung zu keiner Verzerrung im niedrigdimensionalen Raum. Die zentrierten Daten führen lediglich zu einem veränderten Wertebereich der Hauptkomponenten. In Abbildung 59 rechts sind die Ergebnisse der PCA mit den normierten Daten dargestellt. Der qualitative Verlauf der eingebetteten Linien unterscheidet sich nicht von dem aus Abbildung 58 (Seite 131), wo keine Skalierung der Daten vorgenommen wird. Bei der Betrachtung der Wertebereiche der neuen Merkmalsachsen ist festzustellen, dass sich diese um ca. den Faktor vier verkleinert haben. Dies liegt daran, dass die Daten durch die Normierung auf den Wertebereich zwischen null und eins abgebildet werden und sich folglich die Varianz in den Daten verringert. Die maximale plastische

Dehnung in den unskalierten Daten beträgt ca. 3,95%, was dem Faktor der Verringerung der Wertebereiche der Hauptkomponenten entspricht.

Für die nachfolgenden Analysen wird kein separater Schritt zur Datenskalierung vorgenommen. Da zum Erhalt einer kürzeren Berechnungszeit vorab die Dimension mit der *Scikit-learn* Implementierung der PCA (Bisong 2019) reduziert wird, sind die Eingangsdaten für die weiteren Algorithmen bereits zentriert.

8.4 Bewertung der Dimensionsreduktions-Algorithmen

In diesem Kapitel werden Kriterien vorgestellt, mit denen die verschiedenen Algorithmen zur Dimensionsreduktion ausgewertet und verglichen werden können. Neben der subjektiven Bewertung der Visualisierungen im latenten Raum erfolgt eine quantitative Bewertung der Ergebnisse mittels des Qualitätskriteriums Q_{NX}. Darüber hinaus wird der zeitliche Berechnungsaufwand analysiert.

Subjektive Bewertung der Ergebnisse anhand von Streu- und Liniendiagrammen

Das für den/die Ingenieur*in relevante Ergebnis der Dimensionsreduktion ist die Visualisierung der hochdimensionalen Daten im niedrigdimensionalen Raum in Form von Streu- (*OPioS*) beziehungsweise Liniendiagrammen (*OLioS*). Daher werden diese Darstellungen zunächst analog zu den vorangegangenen Abschnitten subjektiv untersucht. Simulationen mit ähnlichem Crashverhalten sollen in ähnlichen Regionen eingebettet werden, während Simulationen mit unterschiedlichem Crashverhalten weit voneinander entfernt dargestellt werden sollten. Die Einfärbung der einzelnen Simulationen gemäß Abbildung 56 (Seite 128) hilft bei dieser Art der Analyse, da dadurch die verschiedenen Crashverhalten den einzelnen Punkten beziehungsweise Linien schnell zugeordnet werden können.

Quantitative Bewertung anhand eines Qualitätskriteriums

Bei der PCA ist es möglich anhand der erklärten Varianz abzuschätzen wie viel der Information durch die Dimensionsreduktion verloren geht. Zudem ist es möglich, die originalen Daten aus den Hauptkomponenten mittels einer inversen Transformation zu rekonstruieren. Es könnte zusätzlich ein Rekonstruktionsfehler berechnet und als Bewertungsmaß für die in den ausgewählten Hauptkomponenten enthaltene Information verwendet werden. Die anderen zu betrachtenden Algorithmen bieten diese Möglichkeiten jedoch nicht. Sie haben keinen Parameter wie die erklärte Varianz und verfügen nicht über die Möglichkeit, die originalen Daten aus dem latenten Raum zu rekonstruieren. Aus diesem Grund wird ein Qualitätskriterium benötigt, das einen einheitlichen Vergleich aller betrachteten Algorithmen ermöglicht. Hierfür wird das in Kapitel 2.7 vorgestellte Kriterium Q_{NX} (siehe Formel 2.16) verwendet. Es folgt ein illustratives Beispiel, um das Qualitätskriterium Q_{NX} zu veranschaulichen. In Abbildung 60 oben links wird ein einfaches exemplarisches Beispieldatenset dargestellt. Es handelt sich um zweidimensionale Daten, die die größte Varianz entlang der horizontalen Achse aufweisen und die entlang der vertikalen Achse in zwei Cluster aufgeteilt werden. Die Werte der horizontalen Achse sind dabei für die beiden Cluster leicht versetzt. Die Dimension der Daten wird auf einen eindimensionalen Raum reduziert. Verglichen werden die Ergebnisse der linearen PCA (Abbildung 60 oben Mitte) und des nichtlinearen UMAP Algorithmus (Abbildung 60 oben rechts).

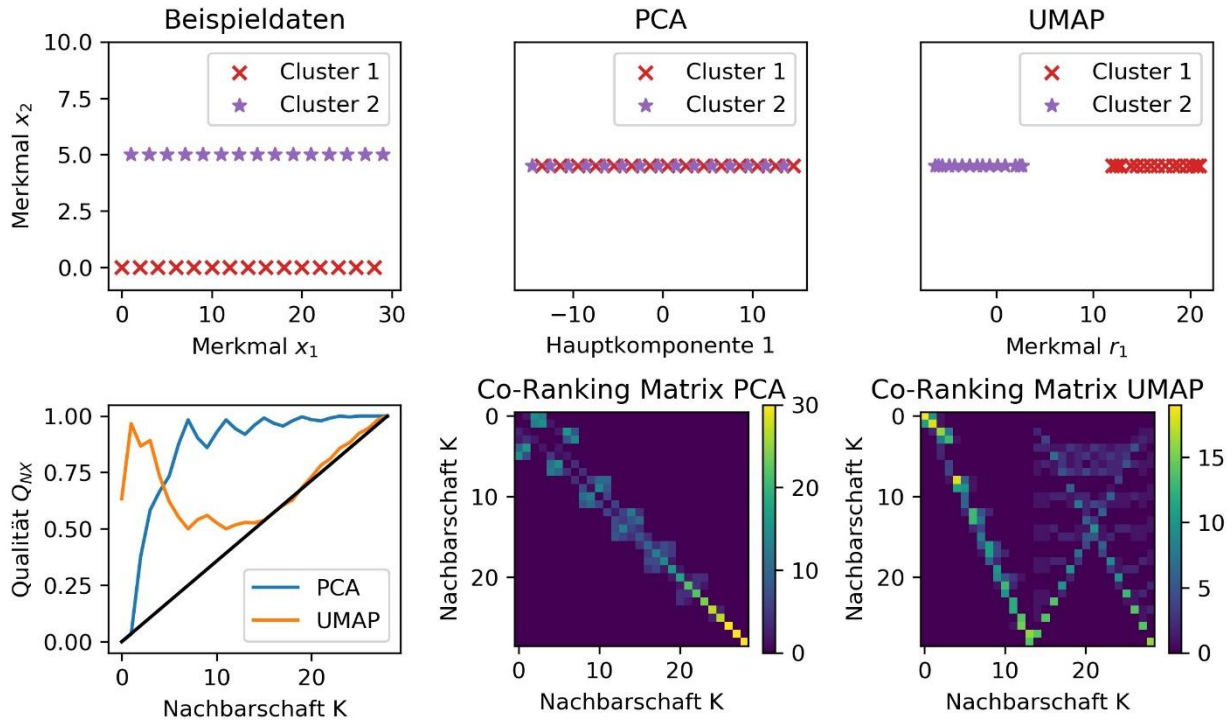


Abbildung 60: Darstellung des verwendeten Beispieldatensets bestehend aus zwei Kategorien, die durch 2 Merkmale beschrieben werden (oben links). Resultat der Dimensionsreduktion mittels der PCA (oben mitte) und UMAP (oben rechts). Aus den niedrigdimensionalen Darstellungen resultierende Co-Ranking Matrizen für die PCA (unten mitte) und UMAP (unten rechts). Farblich dargestellt ist die Anzahl der übereinstimmenden Nachbarschaften im hoch- und niedrigdimensionalen Raum. Aus den Co-Ranking Matrizen resultierende Verläufe von Q_{NX} in Abhängigkeit der Nachbarschaft K (unten links)

Die PCA ist ein varianzbasiertes Verfahren und bildet mit der neuen Hauptachse die Richtung der originalen Daten ab, entlang derer die maximale Varianz auftritt. Da sich diese entlang der horizontalen Achse befindet, entspricht die neue Hauptachse annäherungsweise diesem Merkmal. Vereinfacht kann davon ausgegangen werden, dass die Werte entlang der vertikalen Achse auf eine Dimension kollabieren. Dadurch sind die Cluster nicht mehr voneinander zu unterscheiden und überlappen sich vollständig. Dies hat zur Folge, dass kleine Nachbarschaften nicht erhalten bleiben. Vorher waren die nächsten Nachbarn in den beiden Clustern jeweils Instanzen aus demselben Cluster, während nach der Dimensionsreduktion jeweils eine Instanz aus dem anderen Cluster der nächste Nachbar wird. Größere Nachbarschaften bleiben dagegen weitestgehend erhalten. Weit entfernte Nachbarn im hochdimensionalen Raum sind auch nach der Dimensionsreduktion ferne Nachbarn.

Die Co-Ranking Matrix (Abbildung 60 unten Mitte) und das Qualitätskriterium Q_{NX} (Abbildung 60 unten rechts) verdeutlichen diesen Sachverhalt. Liegen alle Einträge der Co-Ranking Matrix exakt auf der Diagonalen, liegt eine Dimensionsreduktion ohne Informationsverlust vor, da sämtliche Nachbarschaftsbeziehungen im niedrigdimensionalen Raum mit denen im hochdimensionalen Raum übereinstimmen. Desto mehr und desto weiter entfernt die Einträge fernab der Diagonalen liegen, desto schlechter werden Nachbarschaften erhalten.

Für kleine Nachbarschaften <=8 liegen in der Abbildung der Co-Ranking Matrix der PCA kaum Elemente auf der Diagonalen. Dies zeigt, dass lokale Eigenschaften bei der Dimensionsreduktion nicht erhalten bleiben. Für größer werdende Nachbarschaften liegen immer mehr Elemente auf der Diagonalen, was wiederum zeigt, dass diese bei der Dimensionsreduktion gut erhalten bleiben. Dies spiegelt sich auch in dem aus der Co-Ranking Matrix abgeleiteten Q_{NX} Verlauf wider. Für kleine Nachbarschaften liegt eine niedrige Qualität vor, während sie für größer werdende Nachbarschaften immer weiter steigt und nahezu den Wert 1 annimmt, was einer vollständigen Erhaltung der globalen Eigenschaften entspricht.

Der UMAP Algorithmus stellt demgegenüber ein nichtlineares Verfahren dar und approximiert in einem ersten Schritt die Nachbarschaften im hochdimensionalen Raum, bevor im Anschluss daran die Dimensionsreduktion durchgeführt wird. Das Ergebnis wird in Abbildung 60 oben rechts visualisiert. Daraus wird ersichtlich, dass UMAP auch mit nur einer Dimension dazu in der Lage ist, die Cluster getrennt voneinander darzustellen. Dadurch, dass die Cluster nach wie vor separiert werden können, wird deutlich, dass kleine Nachbarschaften erhalten bleiben. Sämtliche nächste Nachbarn aus dem hochdimensionalen Raum sind auch in der niedrigdimensionalen Einbettung nächste Nachbarn. Gleichzeitig ist erkennbar, dass dies für größere Nachbarschaften nicht mehr der Fall ist.

Diese Beobachtungen sind erneut in der Co-Ranking Matrix (Abbildung 60 unten Mitte) und der Qualität Q_{NX} (Abbildung 60 unten rechts) von UMAP wiederzufinden. Für kleine Nachbarschaften (N<=4) liegen nahezu alle Werte auf der Diagonalen. Dies bedeutet, dass die Nachbarschaften im hoch- und niedrigdimensionalen Raum übereinstimmen. Entsprechend ist die lokale Qualität hoch und auch Q_{NX} weist für kleine Nachbarschaften einen hohen Wert nahe 1 auf. Für größer werdende Nachbarschaften liegen kaum Einträge auf der Diagonalen, was zeigt, dass die Nachbarschaften im hoch- und niedrigdimensionalen Raum nicht mehr übereinstimmen. Dies ist im fallenden Qualitätsverlauf von Q_{NX} mit größer werdender Nachbarschaft zu erkennen.

Zusammenfassend lässt sich festhalten, dass eine hohe **lokale Qualität** einem großen Wert von Q_{NX} für kleine Nachbarschaften entspricht (siehe Q_{NX} UMAP Abbildung 60 unten links). Dies bedeutet anschaulich, dass kleine Nachbarschaften im hoch- und niedrigdimensionalen Raum erhalten bleiben. Gleichzeitig bedeutet eine hohe **globale Qualität**, dass Q_{NX} große Werte für größere Nachbarschaften aufweist (siehe Qnx PCA Abbildung 60 unten links) und größere Nachbarschaften bei der Dimensionsreduktion erhalten bleiben.

Idealerweise sollte der Algorithmus für die vorliegend vorzunehmende Dimensionsreduktion aber sowohl **lokal** als auch **global** eine hohe Qualität erzielen. Dies würde im betrachteten Beispiel bedeuten, dass sowohl die Cluster als auch deren relative Positionierung zueinander bei der Dimensionsreduktion erhalten bleiben. Im nachfolgenden Kapitel wird aufgezeigt, dass dies gerade bei hochdimensionalen Daten meist unmöglich ist. Hierzu werden verschiedene Algorithmen zur Dimensionsreduktion auf deren Erhaltung von **lokaler und globaler Qualität** mittels des Q_{NX} Verlaufs bewertet.

Zeitlicher Berechnungsaufwand

Neben der qualitativen Bewertung der Dimensionsreduktion ist der zeitliche Berechnungsaufwand ein weiteres Kriterium für die Auswahl eines geeigneten Algorithmus, da die Dimensionsreduktion nicht in der automatisierten Analyse neuer Simulationen vorab erfolgt, sondern im manuellen Analyseteil vorzunehmen ist. Damit ein interaktives Arbeiten mit den Daten möglich ist, muss der Algorithmus dazu in der Lage sein, die Dimensionsreduktion in Echtzeit (wenige Sekunden) durchzuführen. Daher wird auch der zeitliche Berechnungsaufwand jedes einzelnen Algorithmus als weiteres Qualitätsmerkmal berücksichtigt. Algorithmen, die eine parallele Berechnung der Dimensionsreduktion auf mehreren CPUs ermöglichen, werden mit 14 CPUS ausgeführt, sodass ein fairer Vergleich mittels der zur Verfügung stehenden Hardware ermöglich wird.

Stochastisches Verhalten

Neben den bereits erläuterten Auswertungskriterien wird zusätzlich untersucht, inwiefern sich die Ergebnisse eines stochastischen Algorithmus bei mehrmaliger Ausführung mit denselben Parametern unterscheiden. Die resultierenden Einbettungen können stark variieren und von unterschiedlicher Qualität sein. Einige Ergebnisse eigenen sich dabei besser als andere für die Visualisierung des Crashverhaltens, weshalb die Dimensionsreduktion für stochastische Algorithmen mehrfach berechnet und die daraus resultierende beste Einbettung verwendet werden sollte. Da dieser Vorgang auch noch für verschiedene Hyperparameter wiederholt werden muss, ist dies hinsichtlich der Berechnungszeit ein wesentlicher Nachteil eines solchen Algorithmus. Ein Verfahren, das einen geringen stochastischen Einfluss aufzeigt, ist besser für die praktische Anwendung geeignet, da die Berechnungen weniger häufig wiederholt werden müssen und damit einen Vorteil in der Berechnungszeit liefern können.

8.5 Analyse der Dimensionsreduktions-Algorithmen für die *OPioS* Datenrepräsentation

Im Folgenden werden die Ergebnisse der einzelnen Algorithmen zur Dimensionsreduktion analysiert. Hierfür werden die originalen undiskretisierten Daten des *Längsträgers* verwendet. Entsprechend werden 50 Simulationen mit jeweils 2286*62 Merkmalen der *OPioS* Datenrepräsentation in ihrer Dimension reduziert. Um die Berechnungszeit insbesondere für den t-SNE und UMAP Algorithmus zu optimieren, wird die Dimension der Originaldaten vorab mithilfe der PCA auf 50 Merkmale reduziert. Da die PCA in diesem Fall lediglich einer Koordinatentransformation entspricht und die erklärte Varianz für 50 Merkmale gemäß Abbildung 57 (Seite 130) eins beträgt, geht dadurch keine Information gegenüber der originalen Daten verloren. Zunächst werden die Einbettungen der einzelnen Verfahren qualitativ untersucht. Dabei wird ein besonderes Augenmerk auf den Einfluss der jeweiligen Hyperparameter auf die dimensionsreduzierte Darstellung gelegt. Die Ergebnisse werden mit denen der PCA verglichen. Neben der qualitativen Beschreibung der Streudiagramme wird der qualitative Verlauf von Q_{NX} über verschiedene Nachbarschaftsgrößen hinweg untersucht. Im Anschluss daran erfolgt ein Vergleich der Algorithmen hinsichtlich deren jeweils benötigter Berechnungszeit. Für diejenigen Algorithmen, die hinsichtlich der Auswertungskriterien am besten abschneiden, wird im letzten Schritt der Einfluss der Voxel-Diskretisierung analysiert.

8.5.1 Principle Component Analysis

Die Ergebnisse der PCA bei Anwendung der *OPioS*-Methode wurden bereits in den vorangegangenen Abschnitten hinsichtlich deren Qualität diskutiert. Daher wird an dieser Stelle lediglich der Verlauf des Qualitätskriteriums erläutert. Dieser ist in Abbildung 61 dargestellt. Die schwarze Diagonale entspricht als Referenz der Qualität einer zufälligen Einbettung der hochdimensionalen Daten im latenten Raum. Die blaue gestrichelte Linie zeigt den Verlauf von Q_{NX} für die PCA über verschiedene Nachbarschaften hinweg. Für kleine Nachbarschaften liegt eine niedrige Qualität vor. Dies bedeutet, dass lokale Nachbarschaften durch die Dimensionsreduktion nur begrenzt erhalten bleiben. Mit größer werdender Nachbarschaft erhöht sich gleichzeitig der Wert von Q_{NX}. Dementsprechend ist in diesen Fällen die Übereinstimmung im hoch- und niedrigdimensionalen Raum größer. In Bezug auf die Darstellung im Streudiagramm aus Abbildung 58 (Seite 131) bedeutet dies, dass die relativen Abstände der drei Cluster für die drei Crashmodi *Faltet Vorne*, *Faltet Hinten* und *Faltet Vorne&Hinten* annähernd mit denen im hochdimensionalen Raum übereinstimmen, wodurch globale Nachbarschaften erhalten bleiben. Innerhalb der Cluster decken sich die Nachbarschaften dagegen nur begrenzt, was in der niedrigen lokalen Qualität resultiert.

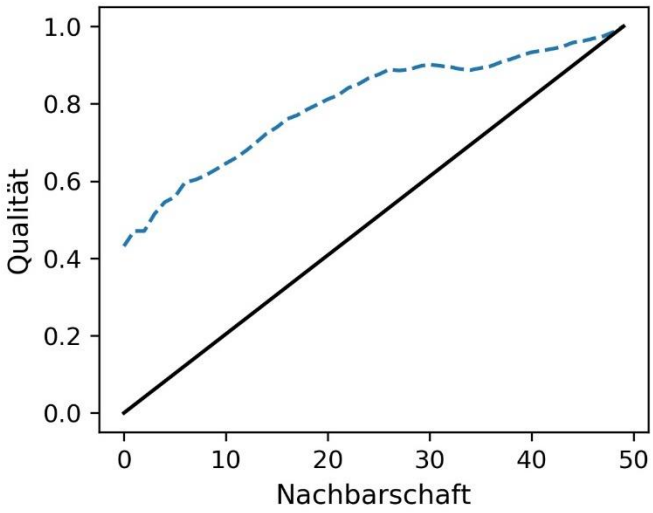


Abbildung 61: Verlauf von Q_{NX} in Abhängigkeit der betrachteten Nachbarschaftsgröße für die PCA

8.5.2 Isometric Feature Mapping

Im Folgenden Abschnitt werden die Ergebnisse und Auswirkungen des Hyperparameters *k* für den Isomap Algorithmus beschrieben. Mit *k* kann die Größe der Nachbarschaft festgelegt werden, die für die Konstruktion des hochdimensionalen Graphen berücksichtigt werden soll. Der Wert kann dabei zwischen eins und der Anzahl der vorhandenen Instanzen (in diesem Fall 50) variiert werden. Für kleine Werte von *k* werden lediglich kleine Nachbarschaften berücksichtigt. Es besteht das Risiko, dass die globalen Eigenschaften der Daten nicht hinreichend genau abgebildet werden. Dagegen können bei großen *k* sogenannte „Kurzschlüsse“ auftreten, die zu einer falschen Darstellung der originalen hochdimensionalen Daten führen. In den nachfolgenden Untersuchungen werden die in Abbildung 62 dargestellten Hyperparameter analysiert. Diese stammen exemplarisch aus dem möglichen Wertebereich zwischen eins und 50, um deren

Einfluss auf die Dimensionsreduktion darzustellen. Im Folgenden werden die Ergebnisse für k=3 und k=23 näher beschrieben.

Für k = 3 ist ein gemeinsamer Ursprung aller Simulationen erkennbar, von welchem aus drei Stränge sternförmig nach außen verlaufen. Im Ursprung und dem nach oben verlaufenden Strang sind die Simulationen aus dem Deformationsverhalten *Faltet Vorne* eingebettet. Der Strang, der nach unten rechts verläuft, beinhaltet die Simulationen, in denen die initiale Deformation im hinteren Bereich des Bauteils auftritt. Gleichzeitig ist hier ein kontinuierlicher Farbverlauf innerhalb der zusammenhängenden Simulationen zu erkennen. Dies deutet darauf hin, dass lokale Nachbarschaften erhalten bleiben. Die Simulationen, die das Mischverhalten *Faltet Vorne&Hinten* aufweisen sind unten links im Streudiagramm zu finden.

Diese Form der Einbettung zeigt, dass durch die kleine Nachbarschaft k=3 lediglich lokale Eigenschaften der Daten beibehalten werden. Simulationen, die jeweils zueinander ähnlich sind, sind auch in der niedrigdimensionalen Darstellung in ähnlichen Bereichen wiederzufinden. Die globalen Eigenschaften können dagegen nicht beibehalten werden. Dies ist an der sternförmigen Einbettung der Simulationen zu erkennen. Die drei deutlich unterschiedlichen Crashmodi werden nicht wie beispielsweise bei der global starken PCA in voneinander klar getrennten Clustern zusammengefasst. Besonders die roten Simulationen (*Faltet Hinten*), die sich wesentlich in ihrem Crashverhalten von den anderen Simulationen abheben, befinden sich in ähnlichen Bereichen wie die Simulationen mit dem konträren Crashverhalten *Faltet Vorne*, bei welchem die Deformation im vorderen Bauteilbereich beginnt. In der niedrigdimensionalen Darstellung wird somit ein kontinuierlicher Übergang zwischen den Simulationen dargestellt, obwohl sich deren Deformationsmodi deutlich unterscheiden.

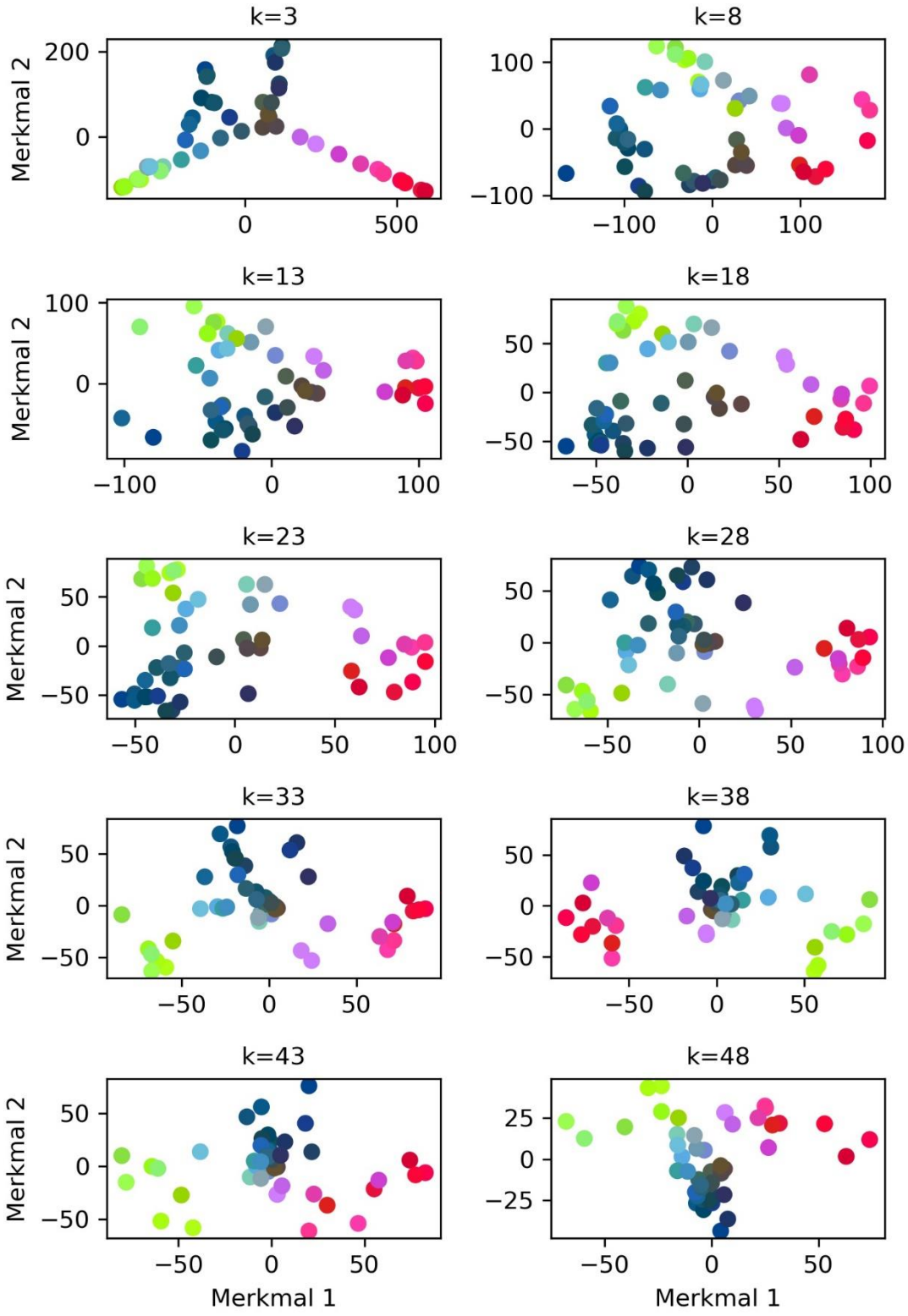


Abbildung 62: Dimensionsreduzierte Darstellung der 50 Simulationen mittels Isomap und der OPioS-Datenrepräsentation für verschiedene Werte des Hyperparameters k

Diese Beobachtungen werden von dem Verlauf des Qualitätskriteriums Q_{NX} in Abbildung 63 bestätigt. Für kleine Nachbarschaften weist die Qualität im Vergleich zur PCA deutlich höhere Werte auf. Mit größer werdender Nachbarschaft steigt sie dagegen bei der PCA auf höhere Werte

an als bei Isomap mit k=3. Ab einer Nachbarschaft von 15 wird deutlich, dass die PCA die globalen Nachbarschaften besser erhalten kann.

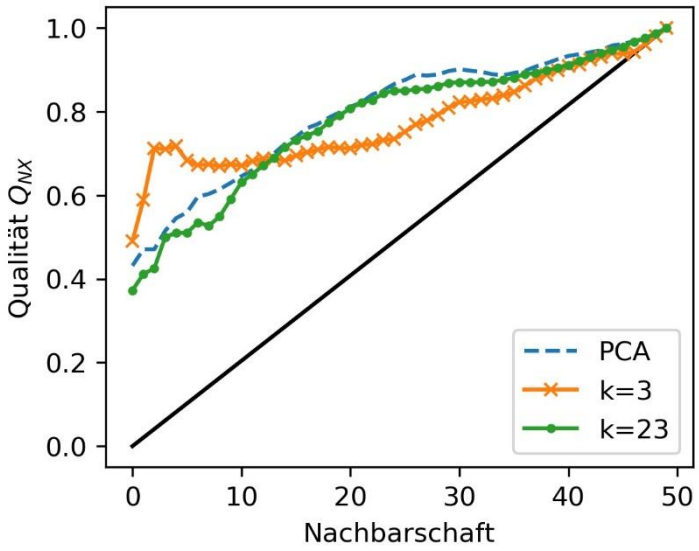


Abbildung 63: Verlauf von Q_{NX} in Abhängigkeit der betrachteten Nachbarschaftsgröße für die PCA, Isomap mit k=3 und k=23

In der Einbettung für k=23 ist das Cluster mit dem Verhaltensmuster *Faltet Hinten* im Vergleich zu k=3 deutlicher von den restlichen Simulationen separiert. In der Mitte sowie im unteren linken Bereich des Diagramms sind die Simulationen eingebettet, in denen die Deformation im vorderen Bereich beginnt. Die Simulationen mit dem Mischverhalten sind oben links zu finden. Insgesamt ist diese Einbettung dem Streudiagramm der PCA ähnlich, was sich sowohl in der Ausbildung von Clustern als auch der kontinuierlichen Farbverläufe innerhalb der Cluster widerspiegelt.

Besonders bei dem Vergleich des roten Clusters für k=3 und k=23 fällt auf, dass sich die lokalen Nachbarschaften in den beiden Darstellungen unterscheiden. Der Verlauf des Qualitätskriteriums in Abbildung 62 bestätigt diese Beobachtung. Während bei k=3 eine höhere lokale Qualität anhand von Q_{NX} verglichen mit der PCA festgestellt wird, ist bei k=23 eine niedrige Qualität zu verzeichnen. Darüber hinaus liegen für k=23 über alle Nachbarschaftsgrößen hinweg im Vergleich zur PCA niedrigere Werte von Q_{NX} vor.

Aus diesen beiden Beispielen wird deutlich, dass der Hyperparameter des Isomap-Algorithmus steuert, inwiefern lokale beziehungsweise globale Eigenschaften erhalten bleiben. Desto größer dessen Wert, desto stärker werden globale Beziehungen bei der Dimensionsreduktion berücksichtigt und desto ähnlicher sind die Ergebnisse denen der PCA. Gleichzeitig kann festgestellt werden, dass der Isomap-Algorithmus zwar hinsichtlich der lokalen Qualität für k=3 den Ergebnissen der PCA überlegen ist, die Visualisierung der Ergebnisse dem Anwender an dieser Stelle jedoch keine Vorteile liefert, da die globalen Nachbarschaften unzureichend erhalten bleiben.

8.5.3 t-Stochastic Neighbor Embedding

Als Nächstes werden die Ergebnisse des t-SNE-Algorithmus diskutiert. Für den Hyperparameter *Perplexität p* schlagen die Autoren einen Wertebereich zwischen 5 und 50 vor. In dieser Arbeit

werden Parameter zwischen 5 und 29 untersucht. Für das vorliegende Beispieldatenset treten keine nennenswerten Unterschiede in den Ergebnissen für größere Werte auf.

Bei t-SNE handelt es sich um eine stochastische Methode. Mehrmalige Berechnungen mit denselben Hyperparametern führen daher zu unterschiedlichen Ergebnissen. Es können dabei starke Schwankungen in der Qualität der dimensionsreduzierten Daten auftreten. In Abbildung 64 wird der t-SNE-Algorithmus 100-mal angewendet und für jedes Ergebnis die resultierende Qualität Q_{AVG}, die sich als Mittelwert von Q_{NX} über sämtliche Nachbarschaften hinweg ergibt, visualisiert. Dabei treten zwei deutliche Ausreißer mit einer niedrigen Qualität Q_{AVG} von ca. 0,75 auf. Der Mittelwert beträgt 0,79 während die Standardabweichung den Wert 0,011 annimmt. Diese Visualisierung verdeutlicht die Notwendigkeit, den Algorithmus stets mehrere Male mit den jeweils gleichen Hyperparametern auszuführen und das Resultat mit den höchsten Werten von Q_{AVG} zu verwenden, damit dem Anwender nicht zufällig ein schlechtes Ergebnis dargestellt wird.

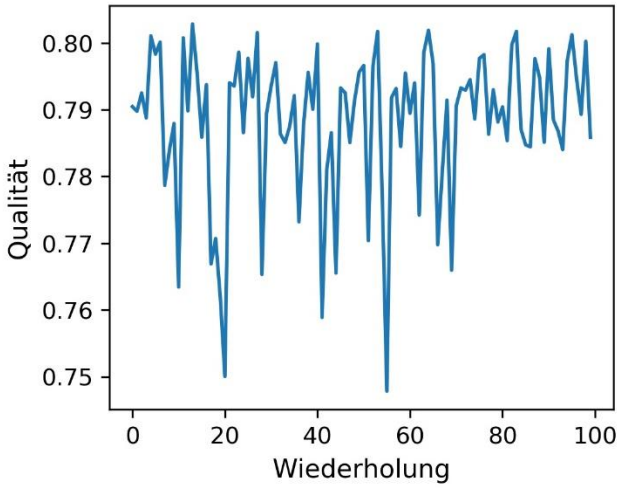


Abbildung 64: Darstellung der Qualität Q_{AVG} für 100 Wiederholungen des t-SNE Algorithmus

Daher wird für die nachfolgenden Untersuchungen der t-SNE-Algorithmus jeweils 100-mal angewendet und das hinsichtlich Q_{AVG} beste Resultat für die weitere Analyse ausgewählt. Die entsprechenden Ergebnisse sind in Abbildung 65 dargestellt. Exemplarisch werden die Einbettungen für p=5 und p=25 näher beschrieben.

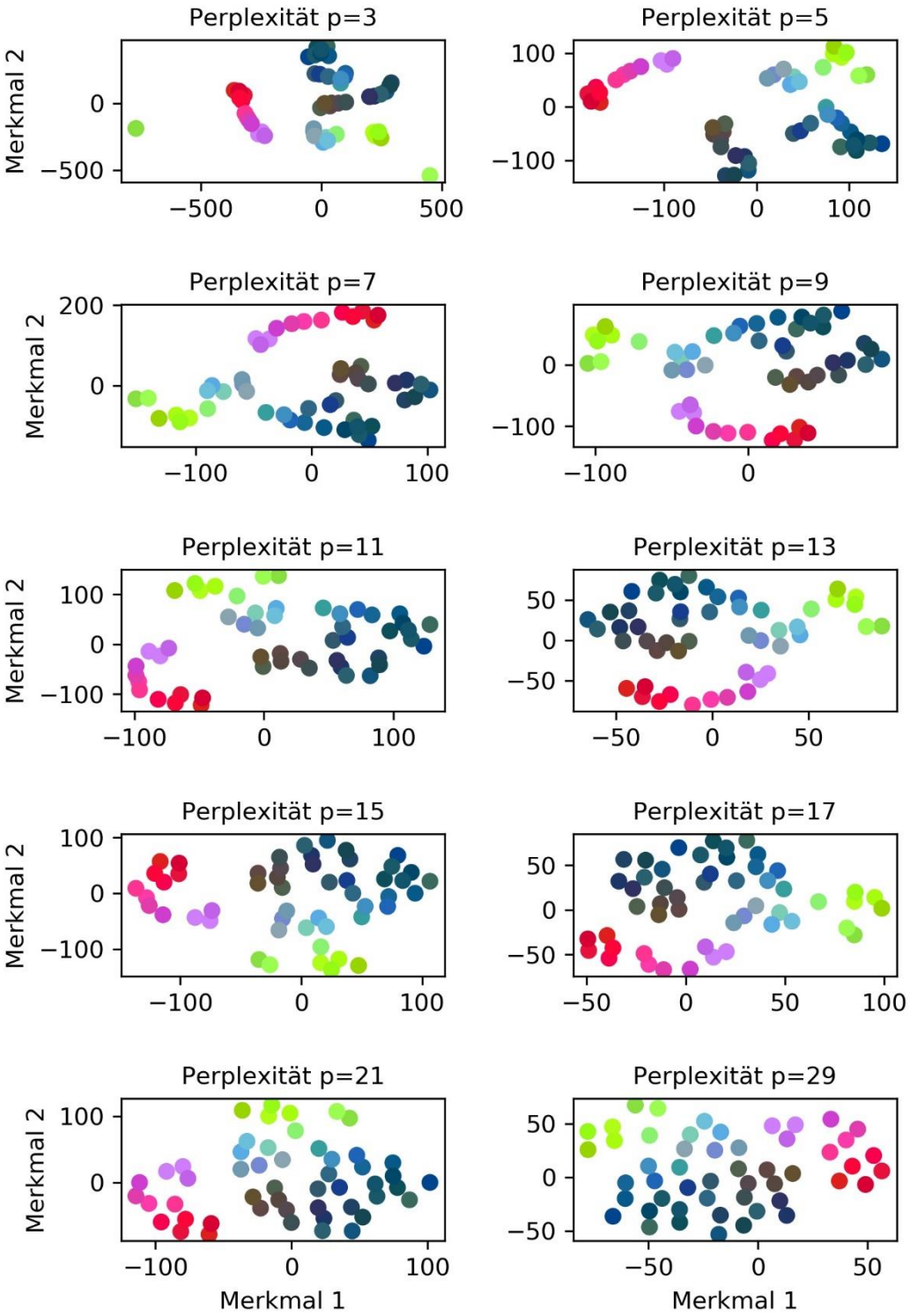


Abbildung 65: Dimensionsreduzierte Darstellung der 50 Simulationen mittels t-SNE und der OPioS-Datenrepräsentation für verschiedene Werte des Hyperparameters p

Für p=5 sind vier Cluster zu erkennen. Während die Simulationen mit dem Crashverhalten *Faltet Hinten* oben links eingebettet sind, finden sich die Simulationen mit dem Verhalten *Faltet Vorne* in den beiden Clustern im unteren rechten Bereich wieder. Die Simulationen mit *Faltet Vorne&Hinten* sind oben rechts abgebildet.

Zwischen den beiden Clustern, in denen die initiale Deformation im vorderen Bereich des Bauteils auftritt, kann auch nach einer tiefgreifenden Analyse aus Ingenieurssicht kein wesentlicher Unterschied im Crashverhalten festgestellt werden. Der Algorithmus scheint im Gegensatz zum/zur auswertenden Ingenieur*in dazu in der Lage zu sein, sehr feine Unterschiede in den Daten aufzulösen. Hier bleibt festzuhalten, dass eine zu feine Aufteilung solcher Unterschiede, dem/der Ingenieur*in aber auch ein vermeintlich deutlich unterschiedliches Crashverhalten suggerieren kann, wo jedoch nur geringe Unterschiede vorliegen. Im Allgemeinen können, wie bereits von den Autoren des Algorithmus beschrieben wird (van der Maaten und Hinton 2008) auch Ergebnissen auftreten, in denen Cluster gänzlich bedeutungslos sind.

Innerhalb der einzelnen Cluster treten kontinuierliche Farbverläufe auf. Dies spricht analog zum Isomap mit k=3 dafür, dass lokale Nachbarschaften erhalten bleiben. Die feine Aufteilung der Daten in mehrere Cluster führt jedoch zu einer negativen Beeinflussung der globalen Qualität.

Der Verlauf von Q_{NX} aus Abbildung 66 bestätigt diese Beobachtung. Für kleine Nachbarschaften liegt eine hohe Qualität vor. Gerade im Vergleich mit den Ergebnissen der PCA werden die Vor- und Nachteile von t-SNE mit p=5 deutlich. Ist der Anwender an dem Erhalt lokaler Nachbarschaften interessiert, liefert t- SNE die besseren Ergebnisse. Möchte er allerdings einen Überblick über sämtliche Nachbarschaften hinweg erhalten, schneidet die PCA hinsichtlich dem Erhalt globaler Nachbarschaften besser ab.

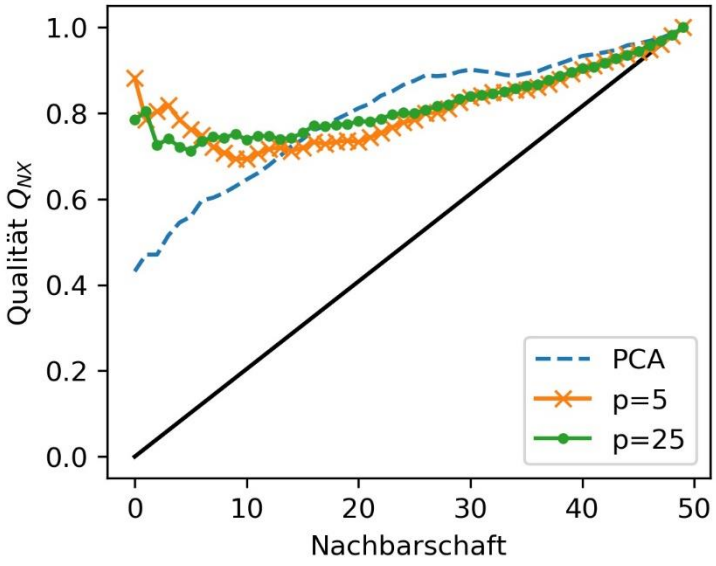


Abbildung 66: Verlauf von Q_{NX} in Abhängigkeit der betrachteten Nachbarschaftsgröße für die PCA und t-SNE mit p=3 und p=23

Mit größer werdender Perplexität geht die feine Aufteilung in Cluster zunehmend verloren. Für p=25 treten diese Gruppierungen kaum noch auf. Die drei Crashmodi werden dann zwar noch in jeweils unterschiedlichen Bereichen im latenten Raum dargestellt, eine klare Abgrenzung untereinander ist dagegen nicht mehr möglich. Dies lässt vermuten, dass im Vergleich zu p=5 eine höhere globale Qualität vorliegt. Nach dem Verlauf von Q_{NX} aus Abbildung 66 bestätigt sich diese Annahme jedoch nur eingeschränkt. Lediglich für Nachbarschaften zwischen 7 und 30 ist die Qualität Q_{NX} von p=25 besser. Für größere Nachbarschaften liegen keine nennenswerten

Unterschiede vor. Für kleinere Nachbarschaften zeigt sich dagegen, dass die Visualisierung mit p=25 schlechter abschneidet als mit p=5, die lokale Qualität jedoch immer noch höher ist als bei der PCA.

Die Analysen zeigen, dass die Perplexität ähnliche Effekte in der Visualisierung der niedrigdimensionalen Daten hervorruft, wie der Parameter *k* beim Isomap-Algorithmus. Kleine Werte sorgen für die Erhaltung lokaler Nachbarschaften und liefern Visualisierungen, in denen eine Aufteilung der Daten in mehrere Cluster erfolgt. Dagegen steigt mit größeren Werten von *p* der Erhalt globaler Nachbarschaften, wodurch die Anzahl der kleinen Cluster abnimmt. Anhand des farblichen Verlaufs ist ersichtlich, dass das Crashverhalten dennoch gemäß seiner Ähnlichkeit in jeweils unterschiedlichen Bereichen eingebettet wird.

8.5.4 Uniform Manifold Approximation and Projection

Der UMAP-Algorithmus weist zwei Hyperparameter auf. Der Parameter k legt die im hochdimensionalen Raum zu betrachtende Nachbarschaft fest, um die originalen Daten zu approximieren. Das Konzept ist dabei ähnlich zu dem bei Isomap und t-SNE. Dieser Parameter kann für die vorliegenden Daten einen Wert zwischen eins und 50 annehmen. In dieser Analyse werden exemplarisch Werte beginnend bei fünf bis zu einem maximalen Wert von 25 untersucht. Größere Parameterwerte zeigen in dem verwendeten Datenset keine weiteren Veränderungen in der Darstellung.

Der zweite Parameter *min-dist*, beeinflusst die Visualisierung der transformierten Daten im niedrigdimensionalen Raum. Kleine Werte sorgen für eine dichtere Einbettung der Simulationen, während größere Werte die einzelnen Punkte weiter voneinander entfernt darstellen. Der Parameterbereich liegt zwischen null und eins. Exemplarisch werden die Werte aus Abbildung 68 untersucht, um den Einfluss dieses Parameters abzubilden.

Wie t-SNE ist auch UMAP ein stochastischer Algorithmus. Mehrere Berechnungen mit denselben Hyperparametern sorgen demnach für unterschiedliche Ergebnisse. In Abbildung 67 sind für 100 Berechnungen von UMAP die Ergebnisse von Q_{AVG} visualisiert. Die Qualität variiert dabei deutlich und es tritt ein Ausreißer mit $Q_{AVG} = 0{,}71$ auf. Der Mittelwert beträgt 0,78 während die Standardabweichung den Wert 0,02 annimmt. Damit ist die Standardabweichung bei UMAP nahezu doppelt so hoch wie die bei t-SNE.

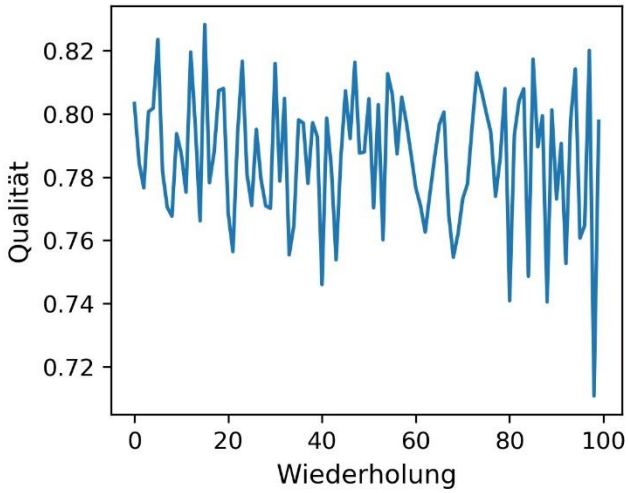


Abbildung 67: Darstellung der Qualität Q_{AVG} für 100 Wiederholungen von UMAP

Im Folgenden werden, wie bei t-SNE, jeweils 100 Einbettungen für jede Hyperparameterkombination berechnet und jeweils die Beste hinsichtlich Q_{AVG} ausgewählt. Die verwendeten Hyperparameter und resultierenden Ergebnisse sind in Abbildung 68 visualisiert.

Für *k*=5 und *min-dist*=0,01 sind vier Cluster, ähnlich wie bei t-SNE mit *p*=5, zu erkennen. Das rote Cluster (*Faltet Hinten*) weist dabei den größten Abstand zu den anderen Instanzen auf. Innerhalb der Cluster sind kontinuierliche Farbübergänge dargestellt, obwohl die Punkte sehr dicht gepackt sind. Diese Visualisierung erlaubt dem/der Ingenieur*in einen schnellen Einblick in die Datenbeschaffenheit. Er kann sofort erkennen, dass unterschiedliche Crashverhalten vorliegen. Bei gleichbleibendem Wert für *k*=5 und steigendem Wert für *min-dist* ist zu erkennen, dass die feine Unterteilung in Cluster zunehmend verloren geht. Für *k*=5 und *min-dist*=0,99 bildet sich ein Cluster mit dem Crashverhalten *Faltet Hinten* und eines mit den übrigen Crashmodi aus. Die Punkte sind dabei nicht so dicht gepackt wie bei *min-dist* =0,01. Um die drei wesentlichen Crashverhalten unterscheiden zu können, eignet sich die Darstellung mit höheren *min-dist*-Werten demnach weniger als im Bereich kleinerer *min-dist.*

Mit steigendem *k* prägen sich die Crashmodi schwächer in einzelnen Clustern aus. Gleichzeitig sind dadurch ähnliche Punkte weniger dicht verortet. Nach wie vor sind die unterschiedlichen Crashverhalten jeweils in ähnlichen Regionen des latenten Raums eingebettet. Es ist jedoch durch die fehlende Clusterbildung nicht mehr auf einen Blick erkennbar, dass sich unterschiedliche Crashmodi bezüglich des betrachteten Bauteils ergeben. Darüber hinaus ist anhand der Streudiagramme festzustellen, dass sich für steigende Werte von *k* der Einfluss von *min-dist* auf die Ergebnisse zunehmend verringert.

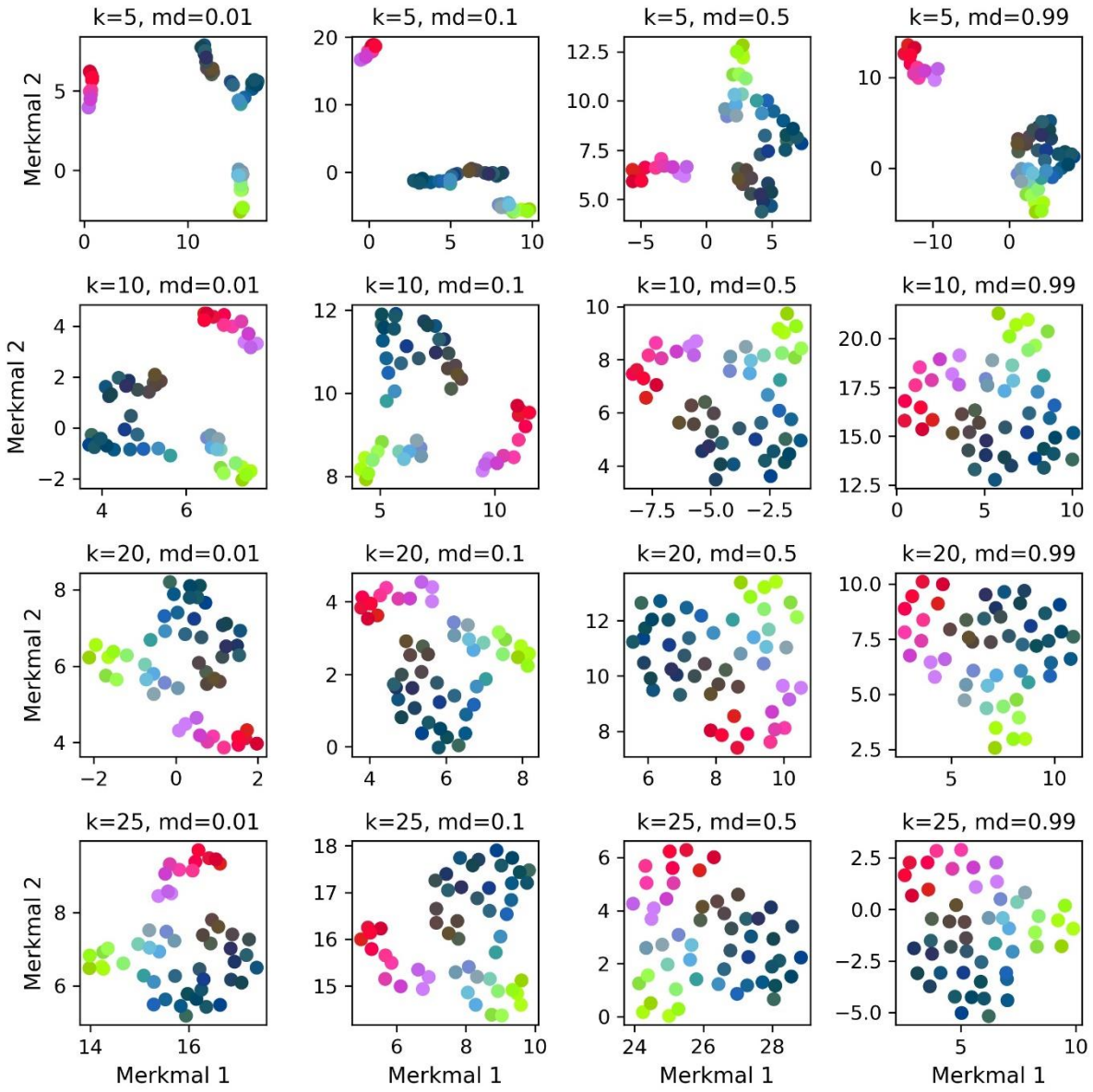


Abbildung 68: Dimensionsreduzierte Darstellung der 50 Simulationen mittels UMAP und der OPioS-Datenrepräsentation für verschiedene Werte der Hyperparameters k und min-dist (md)

In Abbildung 69 ist die Qualität Q_{NX} aus Gründen der Übersichtlichkeit lediglich für vier markante Einbettungen dargestellt. Für sämtliche Hyperparameterkombinationen ist die Qualität für lokale Nachbarschaften höher, für globale Nachbarschaften dagegen niedriger verglichen mit der PCA. Dies deckt sich mit den Ergebnissen der Analysen des *t-SNE*-Algorithmus, der eher lokale Nachbarschaften erhält. Für *k*=5 sind sich die Ergebnisse für *min-dist*=0,01 und *min-dist*=0,99 sehr ähnlich. Für Nachbarschaften größer als 23 ist die Qualität für *min-dist*=0,01 geringfügig höher. Globale Nachbarschaften werden hier besser erhalten.

Für *k*=25 liefert *min-dist*=0,01 höhere Werte von Q_{NX} verglichen mit den Ergebnissen von *min-dist*=0,99 für Nachbarschaften kleiner als 8. Für größere Nachbarschaften sind kaum Unterschiede beim Verlauf von Q_{NX} zu erkennen. Dies deckt sich mit den Beobachtungen aus Abbildung 68, wo der Einfluss von *min-dist* mit steigendem *k* abnimmt. Die Einbettungen sehen sich hierbei sehr ähnlich, was sich in den Verläufen von Q_{NX} widerspiegelt.

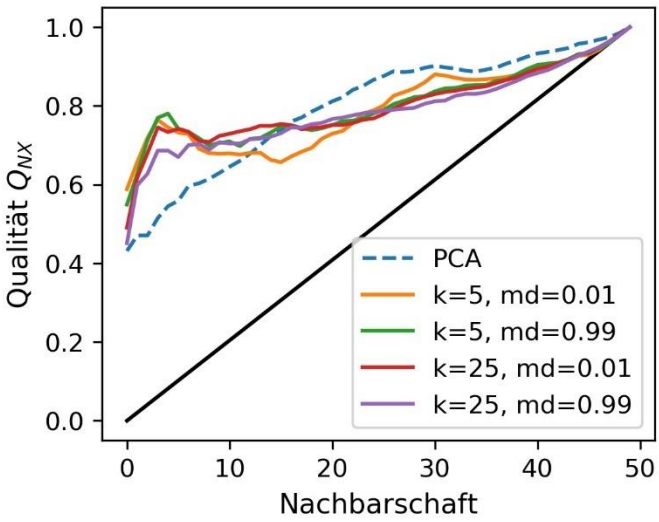


Abbildung 69: Verlauf von Q_{NX} in Abhängigkeit der betrachteten Nachbarschaftsgröße für die PCA und UMAP für verschiedene Hyperparameterwerte von k und min-dist (md)

8.5.5 Vergleich der Algorithmen hinsichtlich deren Qualität und Berechnungszeit

Nachdem in den vorangegangenen Abschnitten die Ergebnisse der einzelnen Algorithmen und die Auswirkungen der jeweiligen Hyperparameter im Hinblick auf deren Qualität ausgewertet wurden, wird nun in diesem Abschnitt ein gesamtheitlicher Vergleich vorgenommen. Neben der quantitativen Bewertung der Qualität wird dabei auch auf die benötigten Berechnungszeiten eingegangen.

Eine Darstellung des Q_{NX}-Verlaufs für sämtliche Algorithmen und deren Hyperparameter wäre unübersichtlich. Aus diesem Grund wird der Q_{NX}-Verlauf vorliegend durch die Definition des Parameters *kmax* in zwei Bereiche aufgeteilt, aus denen zwei Qualitätskriterien Q_{lokal} und Q_{global} abgeleitet werden. Diese können als beschreibende Merkmale eines Algorithmus für eine anschauliche zweidimensionale Abbildung verwendet werden. Der erste Bereich beinhaltet die kleinen Nachbarschaften und fasst die Qualität für diese in dem Parameter Q_{lokal} zusammen. Hierfür werden die Mittelwerte von Q_{NX} gebildet. Der zweite Bereich beinhaltet die großen Nachbarschaften und fasst deren Q_{NX}-Werte in dem Parameter Q_{global} zusammen. Somit werden zwei Qualitätskriterien erlangt, die den Erhalt der lokalen und der globalen Nachbarschaften eines Algorithmus und Hyperparameters bewerten.

Für den Vergleich der vier Algorithmen wird beispielhaft *kmax*=10 gewählt. Ähnliche Ergebnisse ergeben sich auch für andere Werte von *kmax*. Damit wird durch die ersten 10 Nachbarn die lokale Qualität definiert, während sich die globale Qualität aus den größeren Nachbarschaften berechnen lässt. Diese Vorgehensweise ermöglicht es, sämtliche Algorithmen mit unterschiedlichen Hyperparametern in einem einzigen Streudiagramm darzustellen (Abbildung 70). Auf der horizontalen Achse wird die lokale Qualität visualisiert, auf der vertikalen Achse dagegen die globale.

Aus der Darstellung geht hervor, dass die PCA die höchste globale Qualität aufweist, wohingegen t-SNE die beste lokale Qualität zeigt. Isomap verhält sich sowohl lokal als auch global schlechter als die PCA. UMAP verfügt zwar über eine höhere lokale Qualität als die PCA, verhält sich allerdings sowohl lokal als auch global schlechter im Vergleich zum t-SNE-Algorithmus.

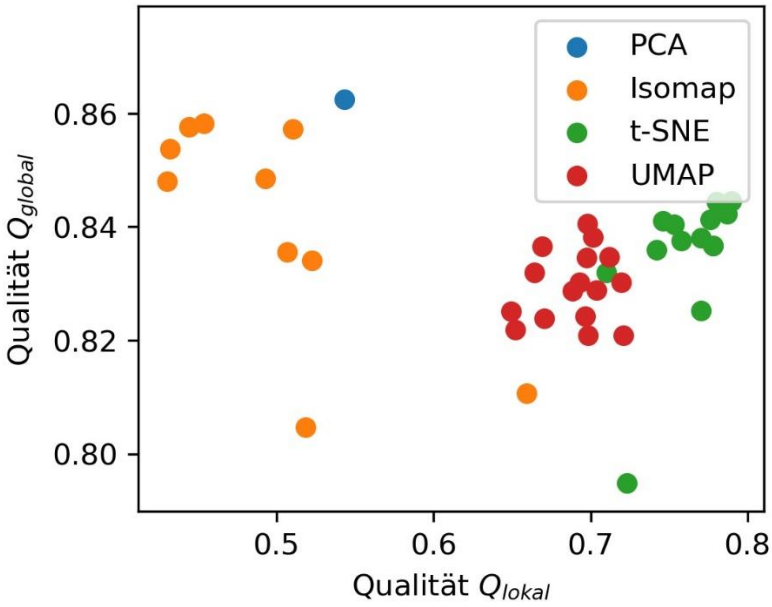


Abbildung 70: Darstellung der Qualitäten Q_{lokal} und Q_{global} für die vier Dimensionsreduktionsalgorithmen und die in diesem Kapitel exemplarisch betrachteten Hyperparameterwerte

Nachdem also die PCA und t-SNE-Algorithmen die besten Ergebnisse hinsichtlich der lokalen und globalen Qualität liefern, werden als nächstes deren Berechnungszeiten betrachtet. Diese sind in Tabelle 11 dargestellt. Bei den Verfahren mit mehreren untersuchten Hyperparameterwerten handelt es sich dabei um die durchschnittlich benötigte Zeit für die Dimensionsreduktion über alle Hyperparameter hinweg. Bei den stochastischen Algorithmen t-SNE und UMAP ist die Zeit angegeben, die für **eine** Berechnung der Dimensionsreduktion benötigt wird.

Tabelle 11: Darstellung des zeitlichen Berechnungsaufwands für die vier Dimensionsreduktionsalgorithmen

Algorithmus	**Zeit[s]**
PCA	0,16
Isomap	0,01
T-SNE	0,2
UMAP	1,5

Isomap weist demnach die geringste Berechnungszeit auf, gefolgt von der PCA und t-SNE. UMAP benötigt daneben am längsten für die Dimensionsreduktion. Wie im vorangegangenen Kapitel beschrieben, ist es für t-SNE und UMAP außerdem notwendig aufgrund deren stochastischem Verhalten, die Einbettung mehrere Male zu berechnen und anschließend die beste (mit dem höchsten Wert Q_{AVG}) hiervon zu verwenden. Dies führt dazu, dass die Dimensionsreduktion mit diesen beiden Algorithmen bei der interaktiven Anwendung noch ineffizienter ist. Es bleibt festzuhalten, dass t-SNE um nahezu eine Größenordnung schneller ist als UMAP.

8.5.6 Zusammenfassung aus Algorithmen-Vergleich

Nachdem die Algorithmen und deren Hyperparameter sowohl qualitativ anhand der resultierenden Streudiagramme, als auch quantitativ mittels Q_{NX}, Q_{lokal}, Q_{global} sowie der benötigten Berechnungszeit analysiert wurden, können die nachfolgenden Schlussfolgerungen gezogen werden.

Die PCA ist durch die schnelle Berechnung der Dimensionsreduktion für die interaktive Auswertung geeignet und weist zudem die höchste globale Qualität im Vergleich zu allen anderen betrachteten Algorithmen auf. Die Streudiagramme geben dem/der Ingenieur*in eine schnelle Übersicht über das Crashverhalten der Varianten. Dabei werden die Simulationen, die sich signifikant von den anderen unterscheiden, in separaten Clustern visualisiert.

Der Isomap-Algorithmus kann für die Anwendung zur Dimensionsreduktion und Visualisierung von Crashsimulationen ausgeschlossen werden. Er verfügt zwar über die geringste Berechnungszeit, liefert jedoch gleichzeitig sowohl lokal als auch global schlechtere Ergebnisse als die anderen betrachteten Verfahren. Hinsichtlich der Identifikation des unterschiedlichen Crashverhaltens liefern die resultierenden Visualisierungen dem Anwender keinen Vorteil gegenüber denen der PCA. Stattdessen muss der/die Ingenieur*in in einem manuellen, iterativen Prozess zusätzlich einen geeigneten Hyperparameter festlegen.

UMAP weist neben seinem stochastischen Verhalten gleichzeitig einen hohen Berechnungsaufwand auf, was eine interaktive Analyse der vorliegenden Daten erschwert. Zudem werden sowohl die lokale als auch die globale Qualität durch die Anwendung des t-SNE-Algorithmus übertroffen. Darüber hinaus ist UMAP nach den Untersuchungen im vorangegangenen Kapitel sensitiver in Bezug auf das stochastischen Verhalten. Die Standardabweichung von Q_{AVG} ist nahezu doppelt so hoch verglichen mit der von t-SNE.

Der t-SNE-Algorithmus ist zwar ebenfalls stochastisch und erfordert eine längere Berechnungszeit im Vergleich zur PCA. Gleichzeitig ist er jedoch dazu in der Lage, bei kleinen Werten für die Perplexität die lokalen Nachbarschaften sehr gut abzubilden. Dadurch werden unterschiedliche Crashverhalten für den Anwender deutlicher in separaten Clustern dargestellt.

Für die Anwendung der Dimensionsreduktion zu Visualisierungszwecken kann demnach abschließend zusammengefasst werden, dass sich sowohl die PCA als auch der t-SNE-Algorithmus eignen. Falls der Anwender einen Überblick über große Nachbarschaften hinweg erhalten möchte, sollte er die PCA zur Dimensionsreduktion wählen. Ist er dagegen an der Erhaltung lokaler Nachbarschaften interessiert, lohnt sich die Anwendung von t-SNE mit kleinen Werten für die Perplexität. Dabei sollte jedoch aufgrund der stochastischen Eigenschaften

beachtet werden, mehrere Einbettungen zu berechnen und die beste mit den höchsten Werten von Q_{AVG} für die Analysen auszuwählen.

8.5.7 Einfluss der Voxel-Diskretisierung für die *OPioS* Datenrepräsentation

In den vorangegangenen Vergleichen zeigen die PCA und der t-SNE-Algorithmus die besten Ergebnisse für die Visualisierung der hochdimensionalen Crashsimulationen. Im folgenden Abschnitt wird der Einfluss der Voxel-Diskretisierung auf die Ergebnisse der Dimensionsreduktion untersucht. Für t-SNE wird hierfür exemplarisch die Perplexität mit dem Wert fünf verwendet. Wie im vorangegangenen Abschnitt wird die Einbettung jeweils 100 mal berechnet und diejenige Einbettung mit dem besten Wert von Q_{AVG} verwendet. Es werden exemplarisch die Diskretisierungen zwischen 10mm und 50mm betrachtet. Die Ergebnisse werden mit denen der originalen undiskretisierten Daten verglichen.

In Abbildung 71 sind die Ergebnisse der PCA für die verschiedenen Diskretisierungen dargestellt.

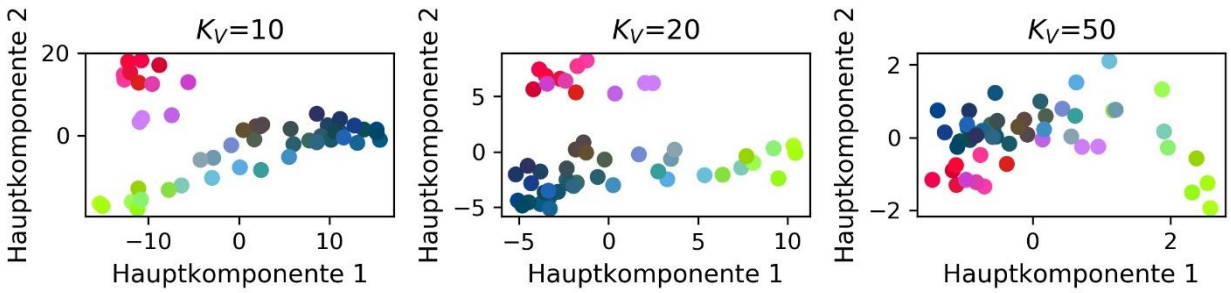


Abbildung 71: Dimensionsreduzierte Darstellung der 50 Simulationen mittels der PCA und der OPioS-Datenrepräsentation für verschiedene Diskretisierungsparameter K_V

Für K_V=10 und K_V =20 sind kaum Unterschiede zu den originalen Daten zu erkennen. Das rote Cluster hat ungefähr denselben Abstand zu dem grünen und blauen Cluster. Globale Nachbarschaften scheinen demnach erhalten zu bleiben. Innerhalb der Cluster sind erneut Farbverläufe zu erkennen. Mit steigender Diskretisierung verringert sich der Abstand zwischen dem roten und dem blauen Cluster. Für d=50 verschwimmen die Simulationen des Crashverhaltens *Faltet Hinten* mit denen von *Faltet Vorne*. Das rote Cluster hebt sich nicht mehr von den anderen Simulationen ab. Dagegen werden die grünen Simulationen von den restlichen Instanzen getrennt dargestellt. Sowohl lokale als auch globale Nachbarschaften scheinen durch die grobe Diskretisierung verloren zu gehen. Somit ist es dem/der Ingenieur*in nicht mehr möglich mittels der PCA die unterschiedlichen Crashverhalten getrennt voneinander darzustellen. Da es sich bei den unterschiedlichen Crashmodi um Effekte handelt, die sich über das gesamte Bauteil erstrecken und nicht nur in einem kleinen Bereich auftreten, ist es auch mit der groben Diskretisierung möglich, ähnliche Simulationen in ähnlichen Bereichen des latenten Raums einzubetten.

Nach den vorangegangenen Untersuchungen des t-SNE-Algorithmus ist zu erwarten, dass lokale Eigenschaften im Vergleich zur PCA besser erhalten werden und Cluster in den Daten identifiziert werden können. Die Streudiagramme sind in Abbildung 72 für p=5 in Abhängigkeit der verwendeten Diskretisierung dargestellt.

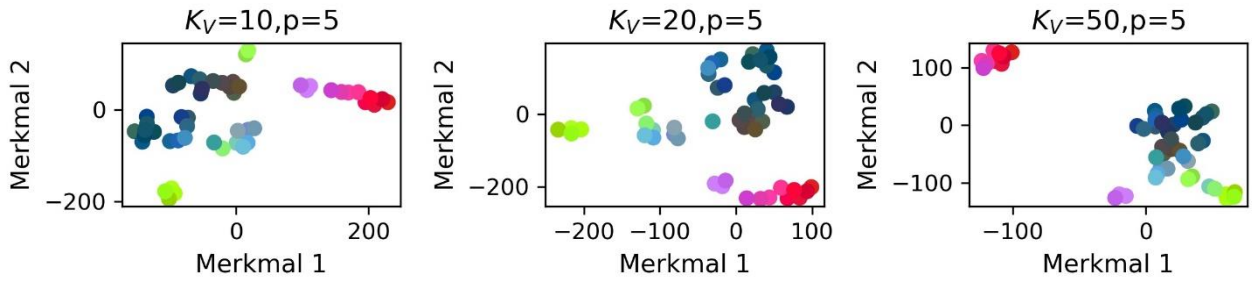


Abbildung 72: Dimensionsreduzierte Darstellung der 50 Simulationen mittels t-SNE mit p=5 und der OPioS-Datenrepräsentation für verschiedene Diskretisierungsparameter K_V

Die Ergebnisse von K_V=10 und K_V=20 sind erneut sehr ähnlich. Dabei sind die Verhalten *Faltet Vorne*, *Faltet Hinten* sowie *Faltet Vorne&Hinten* im latenten Raum deutlich voneinander zu unterscheiden. Während die PCA nicht dazu in der Lage ist, für K_V=50 die Simulationen mit dem Crashverhalten *Faltet Hinten* getrennt von den übrigen einzubetten, schafft dies der t-SNE Algorithmus mit kleiner Perplexität.

Das bestätigt die Vermutung aus dem vorangegangenen Abschnitt, dass ein globales Crashverhalten wie *Faltet Vorne*/*Faltet Hinten*, das sich über das gesamte Bauteil erstreckt, auch bei einer groben Diskretisierung in der Dimensionsreduktion unterscheidbar sein sollte. Aus der Darstellung geht hervor, dass sich mit steigender Diskretisierung das rote Cluster sogar deutlicher besser von den übrigen Simulationen trennen lässt. Das liegt daran, dass die Diskretisierung als Filter wirkt, durch den lokale Unterschiede in den Simulationen verloren gehen. Gleichzeitig werden die globalen Unterschiede erhalten, was dazu führt, dass die verschiedenen Crashmodi trotz grober Diskretisierung in unterschiedlichen Clustern dargestellt werden können.

8.6 Analyse der Dimensionsreduktions-Algorithmen für die *OLioS* Datenrepräsentation

Analog zu den Untersuchungen der *OPioS*-Datenrepräsentation werden im Folgenden die vier Algorithmen zur Dimensionsreduktion für die *OLioS*-Datenrepräsentation miteinander verglichen. Um die Berechnungszeit der einzelnen Algorithmen zu beschleunigen, werden erneut nicht die originalen undiskretisierten Daten der Form 50*62 x 2286 verwendet, sondern lediglich die ersten 100 Hauptkomponenten der PCA. Gemäß Abbildung 57 (Seite 130) werden dadurch 99% der gesamten Varianz der Daten abgedeckt, wodurch der Informationsgehalt der Originaldaten nahezu vollständig erhalten bleibt. Im Gegensatz zur *OPioS*-Datenrepräsentation, werden die Ergebnisse der Dimensionsreduktion nicht als Streudiagramme, sondern als Liniendiagramme visualisiert, wobei erneut die Ergebnisse der einzelnen Algorithmen sowie die Auswirkung deren Hyperparameter qualitativ untersucht werden. Im Anschluss daran erfolgt ein Vergleich der Algorithmen sowie die Analyse des Diskretisierungseinflusses auf die niedrigdimensionale Darstellung des Crashverhaltens.

8.6.1 Principle Component Analysis

Die aus der *OLioS*-Datenrepräsentation resultierenden Ergebnisse der PCA wurden bereits erläutert und sind aus Gründen der Anschaulichkeit erneut in Abbildung 73 links dargestellt. Daher wird im Folgenden der Verlauf von Q_{NX} aus Abbildung 73 rechts analysiert.

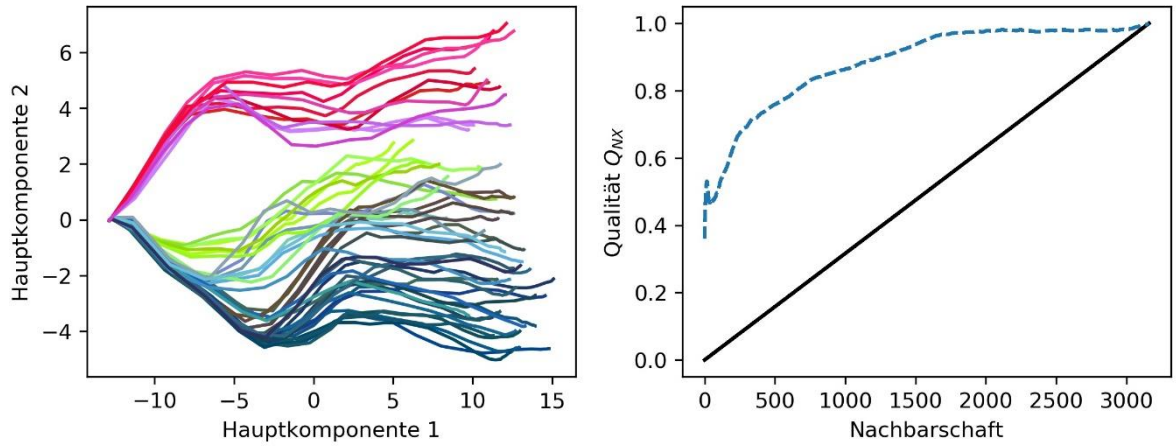


Abbildung 73: Links: Dimensionsreduzierte Darstellung der PCA für die OLioS Datenrepräsentation (übernommen aus Abbildung 58); Rechts: Verlauf von Q_{NX} in Abhängigkeit der betrachteten Nachbarschaftsgröße für die PCA

Für kleine Nachbarschaften liegen wie bei der *OLioS*-Datenrepräsentation geringe Werte von Q_{NX} vor. Lokale Nachbarschaften scheinen demnach nur in begrenztem Maße durch die Dimensionsreduktion erhalten zu sein. Für größere Nachbarschaften steigt auch bei *OLioS* die Qualität an und konvergiert ab circa N=1700 zu eins. Ab hier entsprechen die Nachbarschaften im niedrigdimensionalen nahezu vollständig denen im hochdimensionalen Raum. Die größten Nachbarschaften, die auftreten können, bestehen zwischen den ersten und den letzten Zeitschritten, da sich hier das Crashverhalten am deutlichsten unterscheidet. Abbildung 73 zeigt, dass dies bei der PCA in den Verläufen der einzelnen Linien berücksichtigt ist. Die größten

Distanzen existieren zwischen dem Beginn und dem Ende der Linien, was gleichzeitig die Erklärung dafür liefert, dass große Nachbarschaften nach dem Verlauf von Q_{NX} erhalten bleiben. Dadurch ist die PCA dazu in der Lage, dem/der Ingenieur*in einen schnellen Überblick über das zeitliche Verhalten der Simulationen zu verschafften und Bifurkationen sehr deutlich darzustellen. In den folgenden Abschnitten werden die anderen Algorithmen und deren Hyperparametereinfluss untersucht. Die Ergebnisse werden mit denen der PCA verglichen.

8.6.2 Isometric Feature Mapping

In Abbildung 74 sind die Ergebnisse von *Isomap* für unterschiedliche *k* dargestellt. In der *OLioS*-Datenrepräsentation sind insgesamt 50*62 Datenobjekte vorhanden. Daher wird der Wertebereich für die untersuchten Hyperparameter im Vergleich zur *OPioS*- Methode entsprechend vergrößert. Es werden die Ergebnisse für Werte zwischen 20 und 2000 exemplarisch dargestellt und erläutert.

Für kleine Nachbarschaften wie beispielsweise k=20 liefert der *Isomap*-Algorithmus eine charakteristische Ausprägung der niedrigdimensionalen Darstellung. Wie bei der PCA entspringen sämtliche Linien einem gemeinsamen Ursprung. Sobald ab dem mittleren Zeitschritt die Bifurkation im Crashverhalten auftritt, verlaufen die Linien für die folgenden Zeitschritte in jeweils unterschiedliche Richtungen. Die Simulationen mit dem Crashverhalten *Faltet Hinten* wandern nach oben rechts, während die mit *Faltet Vorne* nach oben links und die mit dem Mischverhaltenvnach unten verlaufen. Die letzten Zeitschritte sind an den Enden der jeweiligen Linien zu finden und in unterschiedlichen Bereichen des latenten Raums eingebettet.

Bei der PCA wird ein anderes Verhalten beobachtet. Nachdem die Linien während den mittleren Zeitschritten auseinanderlaufen, folgen sie für die späteren Zeitschritte einer gemeinsamen Richtung. Die Ähnlichkeiten zwischen den drei beobachteten Crashmodi nehmen mit fortschreitender Zeit im Crash wieder zu, da das gesamte Bauteil deformiert ist und die einzelnen Varianten damit wieder ähnliche plastische Dehnungen in allen Bereichen aufweisen. Damit liegen sämtliche Enden der Linien in einem ähnlichen Bereich. Das Resultat des Isomap-Algorithmus bettet die Enden jedoch in unterschiedlichen Bereichen in der zweidimensionalen Darstellung ein.

Dieses Verhalten liegt an der kleinen Nachbarschaft, die durch den Parameter *k* definiert wird. Die Nachbarschaft eines Punkts wird bei *OLioS* nicht nur durch die anderen Simulationen definiert, sondern auch durch jeweils benachbarte Zeitschritte derselben Simulation. Durch kleine Werte für *k* werden für die Abbildung der originalen Daten im hochdimensionalen Raum lediglich kleine Nachbarschaften berücksichtigt. Im Extremfall sind dies lediglich die anderen Zeitschritte derselben Simulation, ansonsten auch andere Zeitschritte anderer Simulationen. Dadurch bildet der mathematische Graph nicht sämtliche Eigenschaften der originalen Daten ab, was dazu führt, dass auch die geodätischen Distanzen zwischen den Instanzen ungenau berechnet werden. Die Nachbarschaftsbeziehungen bleiben somit lediglich für die jeweils nächsten Nachbarn erhalten. Im niedrigdimensionalen Raum kann damit zwar für die einzelnen Simulationen eine Kontinuität entlang der Zeitschritte sowie ein ähnlicher Verlauf der Linien ähnlicher Simulationen erreicht werden, gleichzeitig wird die globale Nachbarschaft jedoch nicht korrekt approximiert. Dies äußert sich darin, dass die späten Zeitschritte, bei denen wieder ein ähnlicheres Crashverhalten

zwischen sämtlichen Simulationen auftritt, in gänzlich unterschiedlichen Bereichen der Visualisierung eingebettet werden.

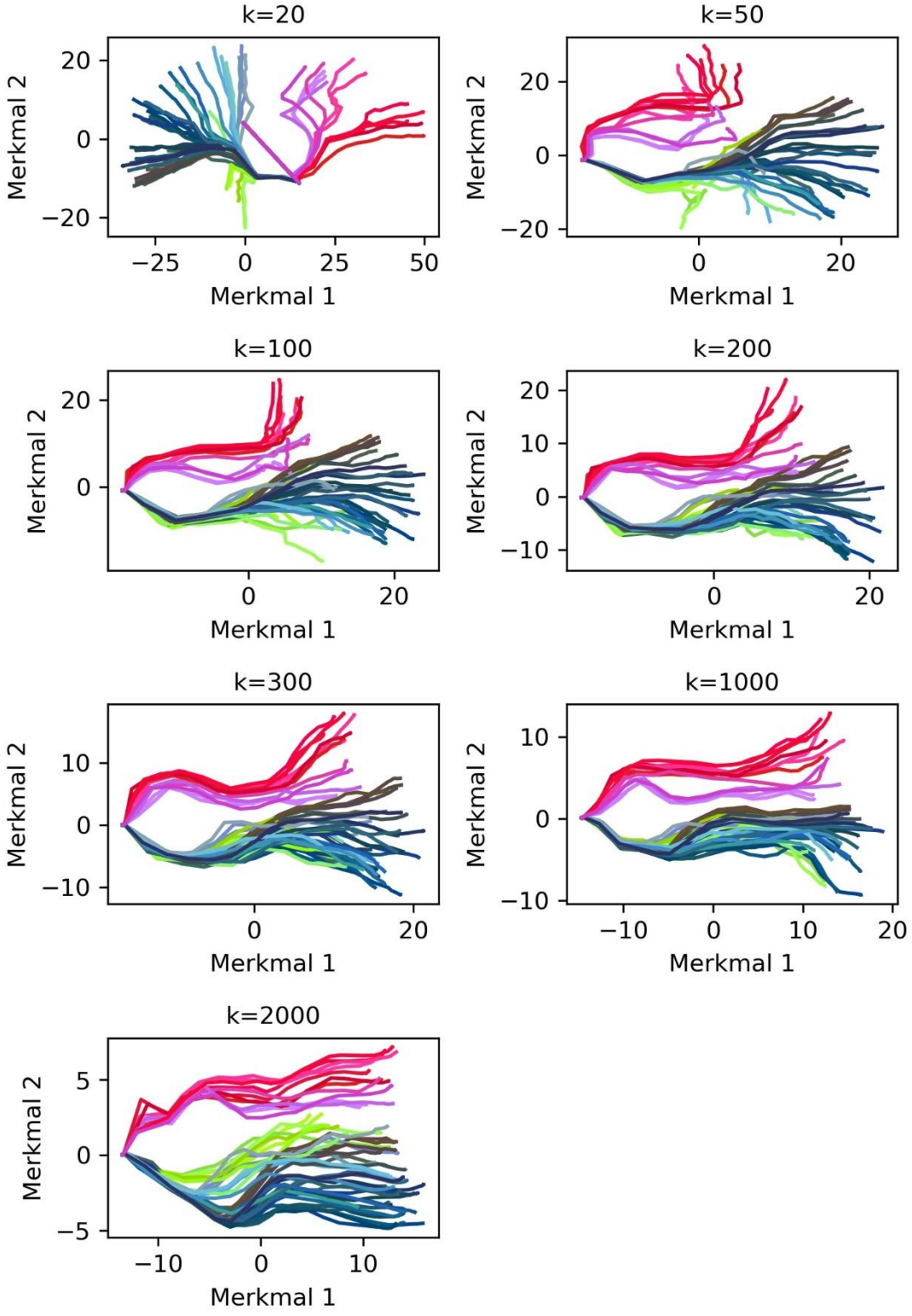


Abbildung 74: Dimensionsreduzierte Darstellung der 50 Simulationen mittels Isomap und der OLioS-Datenrepräsentation für verschiedene Hyperparameterwerte von k

Diese Beobachtungen werden durch den Verlauf von Q_{NX} in Abbildung 75 bestätigt.

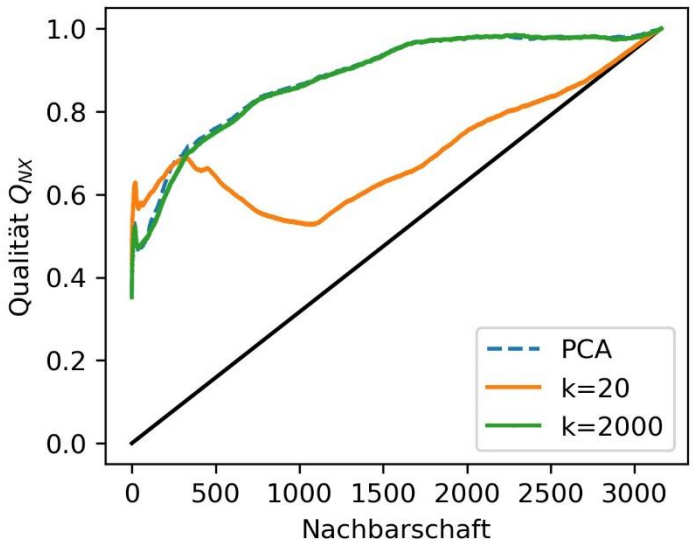


Abbildung 75: Verlauf von Q_{NX} in Abhängigkeit der betrachteten Nachbarschaftsgröße für die PCA und Isomap mit k=20 und k=2000

Für kleine Nachbarschaften (bis ca. 300) liegt für *k*=20 zunächst eine höhere Qualität gegenüber der PCA vor. Mit größer werdender Nachbarschaft sinkt diese jedoch zunehmend ab, bis sie sich schließlich kaum von der Qualität einer zufälligen Einbettung der Daten (nahe der Diagonalen) unterscheidet.

Mit steigenden Werten für *k* werden die globalen Eigenschaften der Daten besser approximiert. Aus den Liniendiagrammen aus Abbildung 74 geht hervor, dass sich die Ergebnisse von *Isomap* denen der PCA annähern. Bei *k*=200 ist das Crashverhalten *Faltet Hinten* deutlich von den anderen Simulationen getrennt dargestellt. Darüber hinaus verlaufen die Linien mit fortschreitender Zeit in ähnliche Richtungen im latenten Raum. Die beiden Crashmodi *Faltet Vorne&Hinten* (grün) und *Faltet Vorne (blau)* sind jedoch nicht mehr voneinander zu unterscheiden.

Erst ab *k*=2000 heben sich die grünen wieder von den blauen und braunen Linien ab und das Bifurkationsverhalten innerhalb dieser Simulationen wird erkennbar. Die Visualisierung erinnert an die Ergebnisse der PCA, bei denen Subcluster in den Daten identifiziert werden können. Dies liegt daran, dass nach der Konstruktion des mathematischen Graphen und der Berechnung der geodätischen Distanzen die Einbettung im latenten Raum anhand des *Multidimensional Scaling* Verfahrens erfolgt. Dieses entspricht für euklidische Distanzen der PCA, weshalb eine hohe Ähnlichkeit zwischen beiden Resultaten vorliegt.

Einige der Linien mit dem Verhalten *Faltet Hinten* weisen für die mittleren Zeitschritte (23-26) Knicke auf. Dies ist bei den Ergebnissen der PCA nicht zu beobachten. Eine mögliche Erklärung für dieses Phänomen sind die im Kapitel 2.6.3 beschriebenen Kurzschlüsse, die bei zu groß gewählten Werten für k bei der Konstruktion des Nachbarschaftsgraphen im hochdimensionalen Raum auftreten können. Dadurch werden Nachbarschaften teilweise falsch abgebildet und verursachen Verzerrungen in den geodätischen Distanzen. Dies kann zu fehlerhaften Einbettungen in der niedrigdimensionalen Darstellung führen.

Der Verlauf von Q_{NX} aus Abbildung 75 bestätigt diese Beobachtungen für k=2000, indem er sich mit dem der PCA nahezu vollständig über sämtliche Nachbarschaften hinweg deckt. Zwischen k=100 und k=400 ist die Qualität von *Isomap* niedriger als bei der PCA. Dies ist auf eventuell vorhandene Kurzschlüsse zurückzuführen, wodurch Nachbarschaften falsch abgebildet werden und dementsprechend Q_{NX} negativ beeinflusst wird.

8.6.3 t-Stochastic Neighbor Embedding

Im Folgenden wird der t-SNE-Algorithmus und der Einfluss dessen Hyperparameters Perplexität p auf die Ergebnisse der Dimensionsreduktion untersucht. Wie in Kapitel 2.6.3 beschrieben, ist nach den Autoren, die diesen Algorithmus vorstellen, ein Wertebereich für p zwischen 5 und 50 zu wählen. In den hier durchgeführten Untersuchungen können jedoch auch für größere Werte signifikante Unterschiede in der Einbettung festgestellt werden, weshalb derselbe Wertebereich wie bei *Isomap* zwischen 5 und 2000 betrachtet wird. Die jeweiligen Ergebnisse sind in Abbildung 76 dargestellt.

Dieser Abbildung ist zu entnehmen, dass die Linien für p=5 chaotisch im latenten Raum verlaufen. Es sind weder ein gemeinsamer Ursprung noch eine zeitliche Kontinuität in den einzelnen Linien zu erkennen. Durch die kleine Perplexität werden die Eigenschaften der hochdimensionalen Daten unzureichend abgebildet. Aus diesem Grund liefert der t-SNE-Algorithmus dem/der Ingenieur*in keinen Mehrwert bei der Analyse des Crashverhaltens, da die unterschiedlichen Crashmodi nicht identifiziert werden können.

Für p=30 und p=50 ähneln die Liniendiagramme denen des Isomap-Algorithmus für kleine Werte von k. Besonders ab dem Zeitschritt, zu dem die Bifurkationen auftreten, sind die Crashmodi *Faltet Hinten*, *Faltet Vorne*, *Faltet Vorne&Hinten* in unterschiedlichen Bereichen in den Diagrammen wiederzufinden. Die Linien verlaufen jedoch erneut chaotisch während der frühen Zeitschritte. Darüber hinaus kann kein gemeinsamer Ursprung zum initialen Zeitschritt identifiziert werden. Dies suggeriert dem Anwender, dass bereits in den ersten Zeitschritten ein stark unterschiedliches Crashverhalten zwischen den Simulationen auftritt, was in der Realität jedoch nicht beobachtet werden kann. Damit sind auch diese Hyperparameterwerte für die Visualisierung des hochdimensionalen Crashverhaltens ungeeignet.

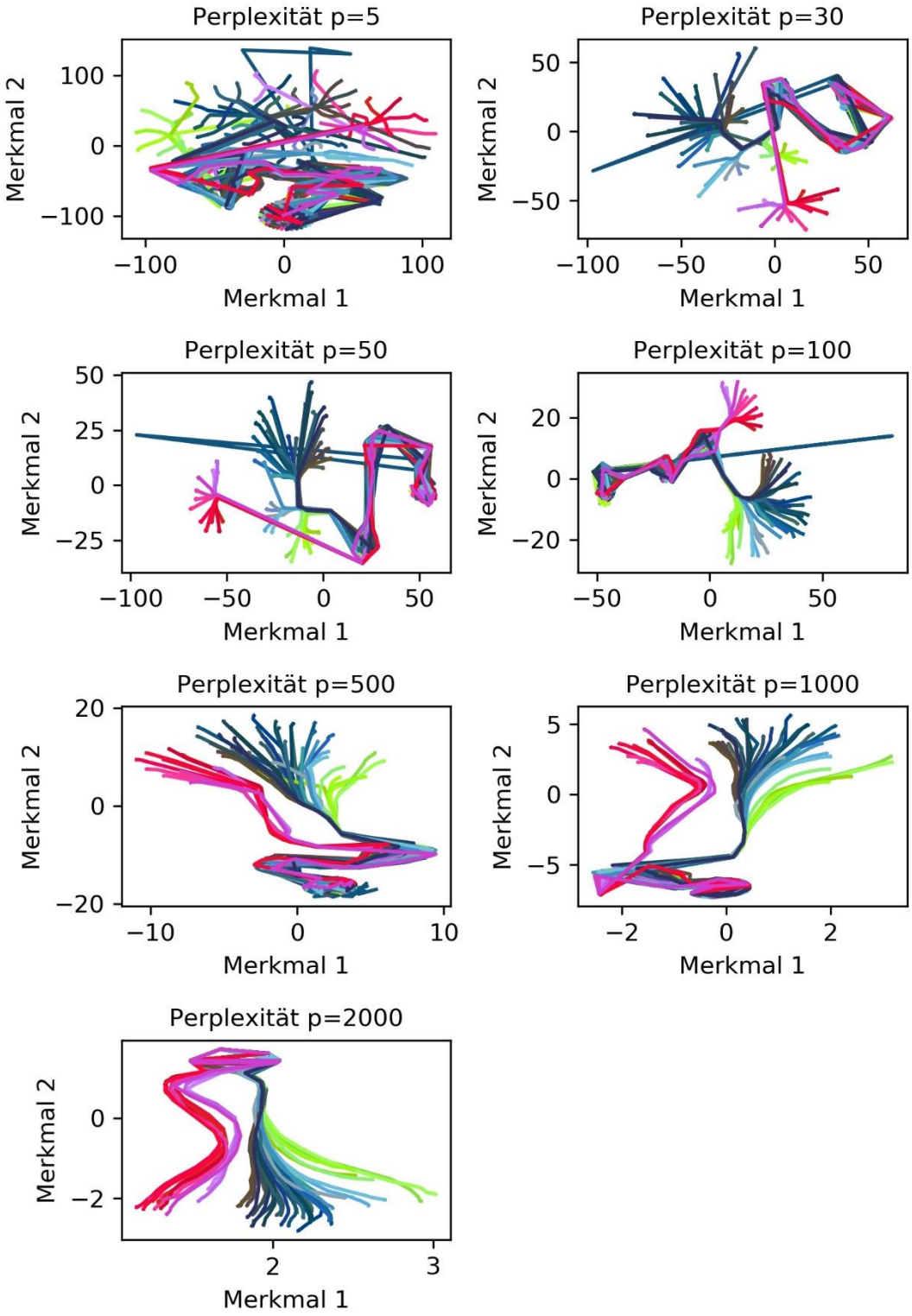


Abbildung 76: Dimensionsreduzierte Darstellung der 50 Simulationen mittels t-SNE und der OLioS-Datenrepräsentation für verschiedene Hyperparameterwerte von p

In Abbildung 77 ist beispielhaft der Verlauf von Q_{NX} für p=30 dargestellt. Wie bei Isomap ist die lokale Qualität für kleine Nachbarschaften zunächst hoch, bevor sie im folgenden Verlauf für größere Nachbarschaften deutlich abfällt. Für große Nachbarschaften ist die Qualität der Dimensionsreduktion auch bei *t-SNE* nur geringfügig besser als bei einer zufälligen Einbettung der hochdimensionalen Daten. Die hohe lokale Qualität von Q_{NX} ist darauf zurückzuführen, dass *t-SNE* Cluster in den Daten identifizieren kann, innerhalb derer lokale Nachbarschaften erhalten bleiben. Die Lage dieser Cluster sowie der ersten Zeitschritte ist jedoch nicht repräsentativ für das tatsächlich vorliegende Crashverhalten, sodass eine niedrige globale Qualität vorliegt. Dies ist erneut damit zu begründen, dass durch die niedrige Perplexität größere Nachbarschaften nicht hinreichend abgebildet werden.

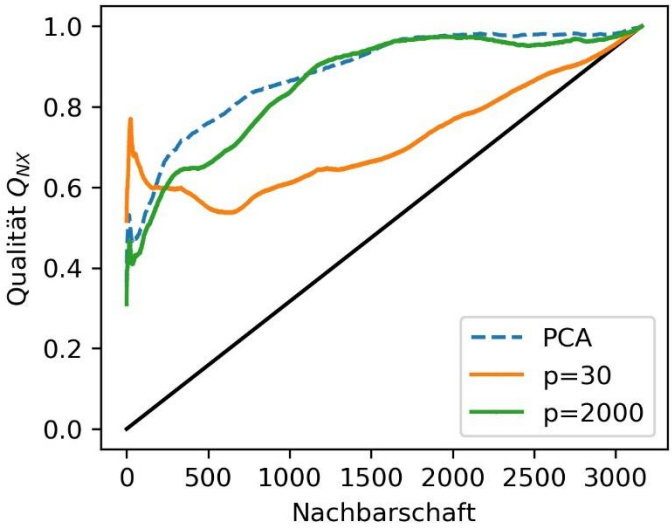


Abbildung 77: Verlauf von Q_{NX} in Abhängigkeit der betrachteten Nachbarschaftsgröße für die PCA und t-SNE mit p=30 und p=2000

Die Visualisierungen für die Perplexitäten 500, 1000 und 2000 erinnern an die Ergebnisse der PCA. Die Linien folgen mit fortschreitender Zeit ähnlichen Richtungen im latenten Raum, während gleichzeitig die Bifurkationen im Crashverhalten deutlich werden. Die roten Linien für das Crashverhalten *Faltet Hinten* heben sich von den anderen Simulationen ab und werden darüber hinaus in zwei Teil-Stränge (rot und lila) aufgeteilt. Des Weiteren ist gegenüber der PCA ein deutlicherer farblicher Unterschied der Simulationen des Verhaltens *Faltet Vorne* zu erkennen. Insbesondere die braunen Linien dieser Simulation sind vorliegend besser von den blauen zu unterscheiden. Des Weiteren sind die grünen Linien mit dem Verhalten *Faltet Vorne&Hinten* deutlicher zu erkennen. Die grünen Linien des Mischverhaltens liegen dagegen, nicht wie bei der PCA und intuitiv von dem/der Ingenieur*in erwartet, zwischen den Linien von *Faltet Hinten* und *Faltet Vorne*. Darüber hinaus ist es auch hier erneut nicht möglich, einen gemeinsamen Ursprung der Linien zum initialen Zeitschritt zu identifizieren.

Diese Beobachtungen spiegeln sich in Abbildung 77 im Verlauf von Q_{NX} für p=2000 wider. Zunächst ist die Qualität für kleinere Nachbarschaften verglichen mit der PCA niedriger. Dies ist zum einen auf den chaotischen Verlauf der Linien zum initialen Zeitschritt zurückzuführen. Darüber hinaus ist auch die Einbettung des Mischverhaltens im rechten Randbereich des Liniendiagramms anstatt – wie erwartet – zwischen den Verhalten *Faltet Vorne* und *Faltet Hinten*

eine weitere mögliche Erklärung hierfür. Ab einer Nachbarschaft von etwa 1000 ist die Qualität von *t-SNE* dagegen gleichauf mit der der PCA, sodass sich die weiteren Verläufe kaum unterscheiden.

8.6.4 Uniform Manifold Approximation and Projection

Neben der im hochdimensionalen Raum zu betrachtenden Nachbarschaft *k*, muss der Parameter *min-dist* für den UMAP-Algorithmus definiert werden. Die Nachbarschaften werden für die folgende Analyse zwischen 10 und 2000 variiert. Um den Einfluss von *min-dist* darzustellen werden die Werte 0.01, 0.1, 0.5 sowie 0.99 verwendet. Die Ergebnisse sind in Abbildung 78 dargestellt.

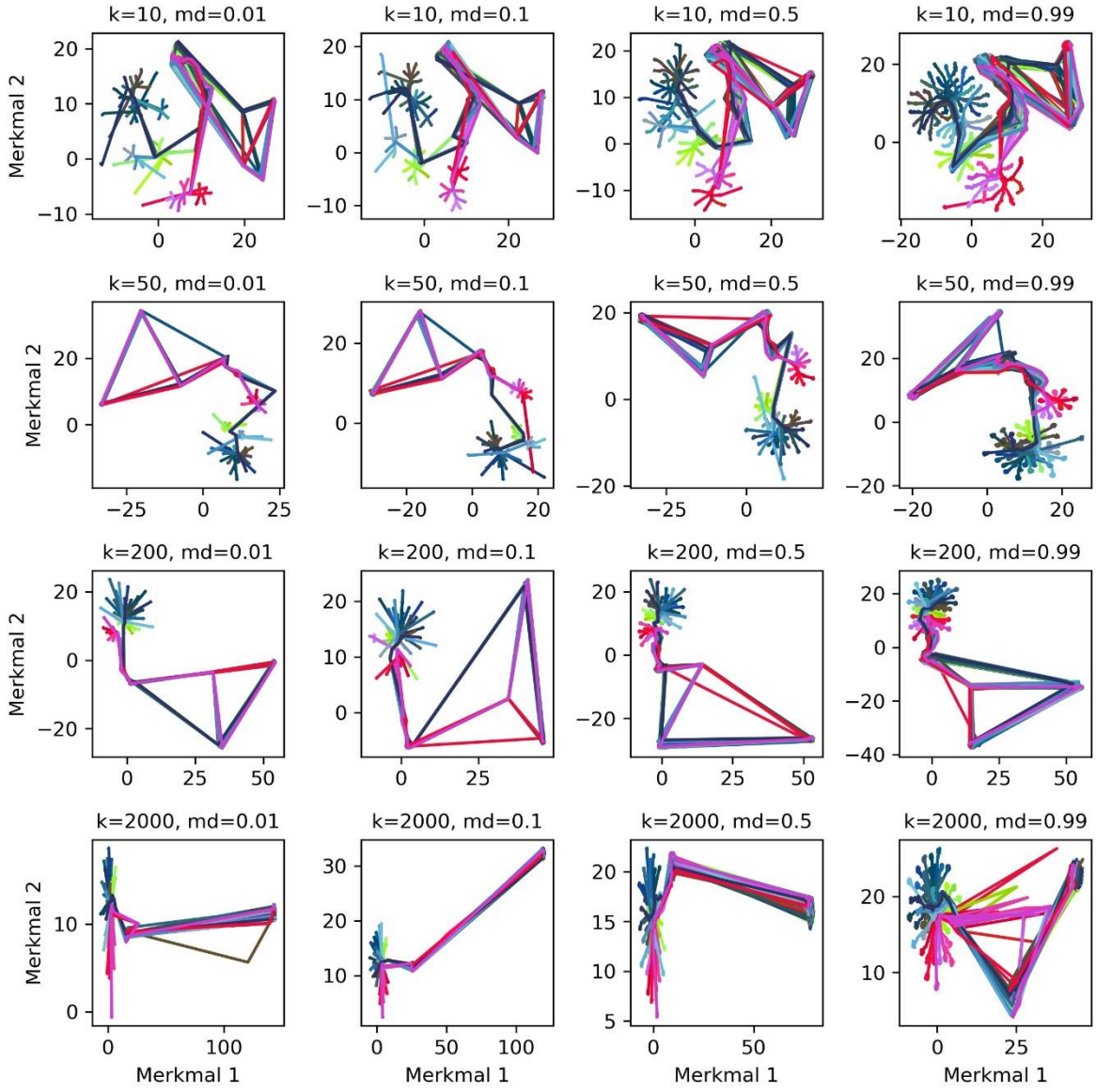


Abbildung 78: Dimensionsreduzierte Darstellung der 50 Simulationen mittels UMAP und der OLioS-Datenrepräsentation für verschiedene Hyperparameterwerte von k und mind-dist (md)

Daraus geht hervor, dass keine dieser Darstellungen für die Visualisierung der hochdimensionalen Crashdaten geeignet ist. Für kleine Werte von *min-dist* sind die einzelnen Linien im Vergleich zu größeren Werten unabhängig vom Hyperparameter *k* deutlicher zu erkennen. Die Liniendiagramme für kleine Werte von *k* erinnern an die von *t-SNE* für kleine Perplexitäten. Es ist kein gemeinsamer Ursprung der Linien zu erkennen und deren Verlauf wirkt für die ersten Zeitschritte chaotisch. Die Bifurkationen im Crashverhalten für die späteren Zeitschritte sind zwar zu erkennen, die Darstellungen eignen sich jedoch für den Anwender nicht wirklich, um eine Übersicht über das Crashverhalten zu erhalten.

Mit zunehmenden Werten für *k* wird mehr Darstellungsfläche für die ersten Zeitschritte verwendet. Die mittleren sowie späteren Zeitschritte sind auf einem engeren Raum dargestellt und die Identifikation der unterschiedlichen Crashverhalten wird zunehmend schwieriger. Eine nähere Betrachtung der späteren Zeitschritte zeigt darüber hinaus, dass teilweise keine farbliche Abstufung mehr zwischen den Linien zu erkennen ist. UMAP ist demnach nicht dazu in der Lage die unterschiedlichen Crashverhalten, insbesondere für große Werte von *k*, für den Anwender übersichtlich abzubilden.

UMAP trifft die Annahme, dass die Daten auf der Mannigfaltigkeit gleichverteilt sind. Die ersten 20 Zeitschritte weisen einen sehr ähnlichen Wertebereich auf, da hier das Bauteil kaum deformiert und die Streuung zwischen den Simulationen entsprechend gering ist. Diese Feststellung entspricht einem Anteil von ca. 32% der gesamten Daten, was verdeutlicht, dass die Annahme gleichverteilter Daten nicht gegeben ist. Dies führt dazu, dass die Mannigfaltigkeit durch den hochdimensionalen Graphen nicht korrekt abgebildet wird und dadurch die Ergebnisse der Dimensionsreduktion für die weitere Auswertung unzureichend sind.

In Abbildung 79 werden exemplarisch die Q_{NX}-Verläufe für k=10 und *min-dist*=0,01 sowie k=2000 und *min-dist*=0,01 dargestellt. Lediglich für kleine Nachbarschaften ist die Qualität zunächst größer als bei der PCA. Ab N=100 wird jedoch der Unterschied zwischen den beiden Verfahren deutlich. Während die PCA große Nachbarschaften sehr gut erhält, zeigt sich das Gegenteil bei Anwendung des UMAP-Algorithmus.

Sowohl die qualitative als auch die quantitative Bewertung zeigen, dass sich UMAP nicht für die Visualisierung der *OLioS*-Datenrepräsentation eignet.

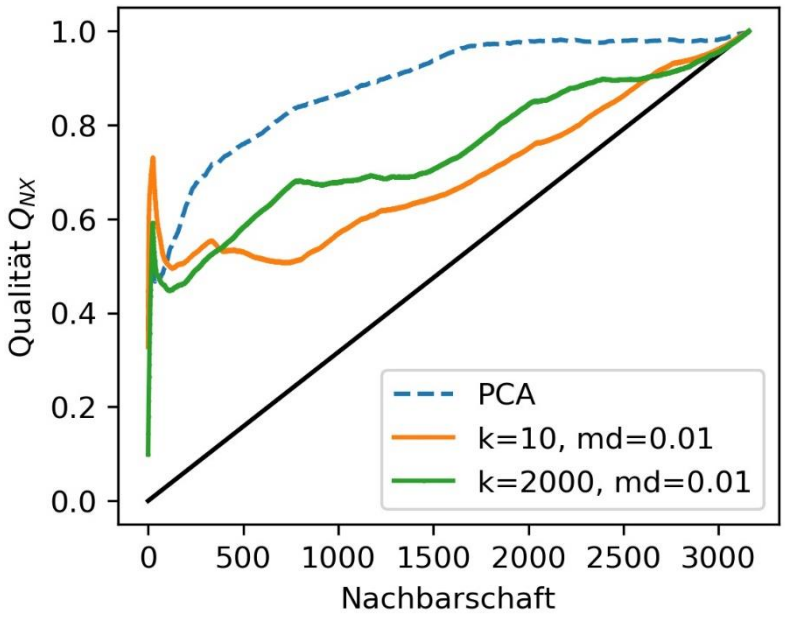


Abbildung 79: Verlauf von Q_{NX} in Abhängigkeit der betrachteten Nachbarschaftsgröße für die PCA und UMAP für verschiedene Hyperparameterwerte von k und min-dist (md)

8.6.5 Vergleich der Algorithmen hinsichtlich deren Qualität und Berechnungszeit

Außer der PCA verfügen die betrachteten Verfahren jeweils über Hyperparameter, deren untersuchte Wertebereiche für *k* beziehungsweise die Perplexität zwischen 5 und 2000 liegen. Bei UMAP kann die Visualisierung durch einen zweiten Hyperparameter *min-dist* gesteuert werden. Dieser beeinflusst die Berechnungszeit jedoch kaum, sodass dessen Einfluss in den nachfolgenden Betrachtungen vernachlässigt werden kann. Die Berechnungszeiten für die einzelnen Algorithmen sind in Abbildung 80 in Abhängigkeit der Größe der zu betrachtenden Nachbarschaft *k* (Isomap und UMAP) beziehungsweise der *Perplexität p* (t-SNE) dargestellt. Daraus geht hervor, dass der zeitliche Berechnungsaufwand für die PCA mit 0,17s am geringsten ausfällt, wodurch die Anforderung an eine Echtzeit-Fähigkeit der Dimensionsreduktion gewährleistet wird. Die übrigen drei Verfahren zeigen eine deutliche Abhängigkeit vom jeweiligen Hyperparameter. Bereits für kleine Werte der Hyperparameter liegen die Berechnungszeiten deutlich über denen der PCA. Mit steigenden Werten erhöht sich auch die benötigte Zeit. Während die Berechnung von *Isomap* deutlich unter einer Minute bleibt, liegt *t-SNE* für *p*=2000 bei ungefähr 60s und UMAP bei ca. 155s. Damit sind alle drei Verfahren für eine interaktive Analyse hinsichtlich der benötigten Berechnungszeit schlechter als die PCA.

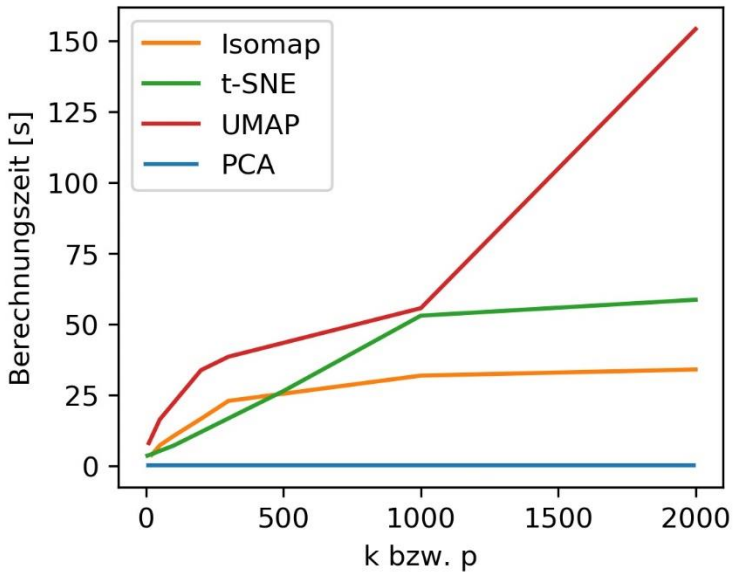


Abbildung 80: Darstellung des zeitlichen Berechnungsaufwands der Dimensionsreduktionsalgorithmen in Abhängigkeit deren Hyperparameter (falls vorhanden)

8.6.6 Zusammenfassung aus Algorithmen-Vergleich

Nachdem die Ergebnisse der Anwendungen der verschiedenen Algorithmen und deren Berechnungszeiten im Detail für die *OLioS*-Datenrepräsentation analysiert wurden, lässt sich zusammenfassend festhalten, dass die Visualisierung der PCA für den/die Ingenieur*in am intuitivsten ist. Es existiert ein gemeinsamer Ursprung der Daten zum initialen Zeitpunkt der Simulation, von dem aus die Linien in Abhängigkeit der Zeit und des Crashverhaltens in unterschiedliche Richtungen verlaufen. Dabei ist die Bifurkation zu den mittleren Zeitschschritten deutlich zu erkennen. Gleichzeitig ist die PCA ein deterministisches Verfahren und kann durch deren kurze Berechnungszeit in Echtzeit angewendet werden.

Der *Isomap*-Algorithmus ist ebenfalls deterministisch, benötigt jedoch in Abhängigkeit des Hyperparameters Berechnungszeiten, die um Größenordnungen über denen der PCA liegen. Zudem hängen die Resultate stark von der gewählten Nachbarschaft *k* ab. Für kleine Werte werden die Bifurkationen zwar sichtbar, globale Nachbarschaften bleiben allerdings nicht erhalten. Bei großen Werten für *k* liefert *Isomap* ähnliche Ergebnisse wie die PCA. Es besteht allerdings das Risiko, dass sogenannte „Kurzschlüsse“ bei der Approximation der hochdimensionalen Daten zu unzureichenden Ergebnissen führen.

Auch *t-SNE* und *UMAP* sind für die Dimensionsreduktion der *OLioS*-Daten eher ungeeignet. Zum einen handelt es sich um stochastische Verfahren, sodass stets mehrere Einbettungen berechnet werden müssen. Zum anderen weisen beide Algorithmen jeweils eine hohe Berechnungszeit auf, sodass eine Anwendung in Echtzeit nicht wirklich möglich ist. *t-SNE* ist für geeignete Hyperparameter zwar dazu in der Lage, die unterschiedlichen Crashverhalten übersichtlich dazustellen. Die erhöhte Berechnungszeit und Notwendigkeit mehrere Einbettungen zu berechnen erschweren allerdings das Auffinden eines geeigneten Parameters. Darüber hinaus werden die ersten Zeitschritte, in denen eine geringe Deformation sowie eine geringe Streuung unter den Simulationen auftritt, diffus eingebettet und vermitteln dem Anwender ein falsches Bild über das tatsächliche Crashverhalten.

8.6.7 Einfluss der Voxel-Diskretisierung für die *OLioS* Datenrepräsentation

Nachdem sich aus den vorangegangenen Untersuchungen die PCA für die *OLioS*-Datenrepräsentation als beste Methode für Visualisierungszwecke herausgestellt hat, wird im Nachfolgenden für diesen Algorithmus der Einfluss der Voxel-Diskretisierung für K_V = [10mm, 20mm, 50mm] untersucht. Die Ergebnisse sind in Abbildung 81 dargestellt. Durch die Mittelung der Informationen mehrerer FE wirkt die Diskretisierung als eine Art Filter, der gleichzeitig die Varianz in den Daten reduziert. Dies führt in dem aufgezeigten Beispiel dazu, dass die Eigenwerte für die zweite und dritte Hauptkomponente sehr ähnlich werden, wodurch sich das Crashverhalten nicht mehr eindeutig in zwei Dimensionen darstellen lässt. Aus diesem Grund wird für die Untersuchung des Diskretisierungseinflusses eine dreidimensionale Darstellung durch die ersten drei Hauptkomponenten der PCA verwendet.

Bei einer Diskretisierung mit Voxeln der Kantenlänge K_V=10mm sind kaum Unterschiede zu den originalen undiskretisierten Liniendiagrammen zu erkennen. Die Bifurkation zu den mittleren Zeitschritten ist nach wie vor vorhanden und es ist ersichtlich, dass die Linien entlang der ersten Hauptkomponente verlaufen. Der Wertebereich der Visualisierung hat sich jedoch für sämtliche Achsen verkleinert, was auf die Reduktion der Varianz durch die Diskretisierung zurückzuführen ist. Für K_V=20 ist eine weitere Abnahme des Wertebereichs der einzelnen Hauptkomponenten zu beobachten. Gleichzeitig wird die Bifurkation zu den mittleren Zeitschritten deutlicher. Dieser Effekt wird dadurch verstärkt, dass die Linien mit zunehmender Zeit nicht weiter auseinanderlaufen wie bei den originalen Daten. Stattdessen ziehen sie sich nach dem Auftreten der Bifurkation erneut zusammen. Auch dieser Effekt ist auf die Diskretisierung und die Mittelung von Informationen zurückzuführen. Durch die verringerte Varianz erhöht sich die Ähnlichkeit zwischen den einzelnen Simulationen für die späteren Zeitschritte, was dazu führt, dass die Linien zum Ende des Crashs wieder näher beieinander eingebettet werden. Die Beobachtungen für K_V =20 können auf die Diskretisierung mit K_V =50 übertragen werden. Der Wertebereich der Hauptkomponenten nimmt weiter ab, die Bifurkationen sind weiterhin deutlich zu erkennen und die einzelnen Linien ziehen sich für die späteren Zeitschritte noch stärker zusammen.

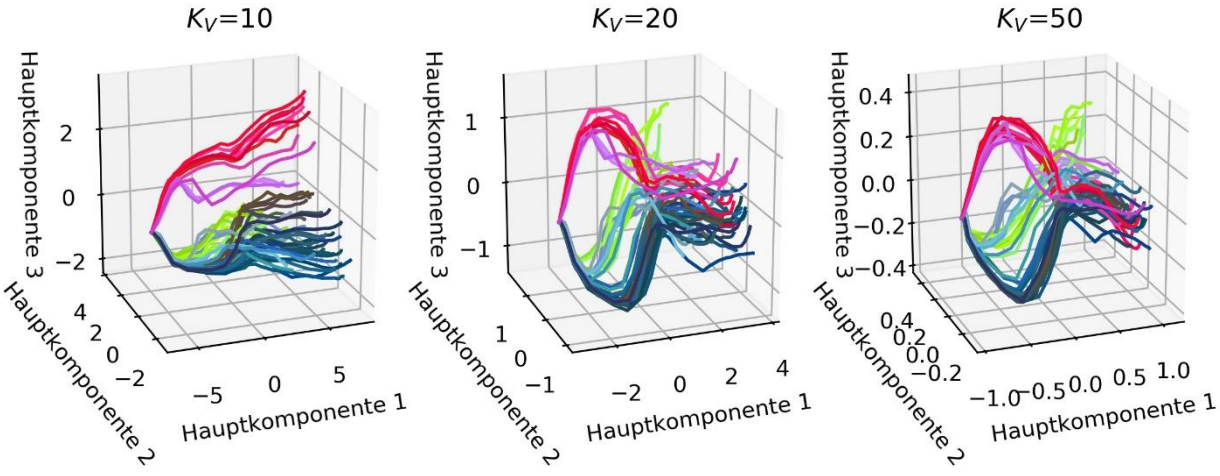


Abbildung 81: Dimensionsreduzierte Darstellung der 50 Simulationen mittels der PCA und der OLioS-Datenrepräsentation für verschiedene Diskretisierungsparameter K_V

8.7 Fazit

Hinsichtlich der **Berücksichtigung des zeitlichen Verhaltens**, eignen sich also sowohl die *OPioS*- als auch *OLioS*-Methode als Datenrepräsentation für die Dimensionsreduktion. *OPioS* bietet dabei den Vorteil einer übersichtlicheren Darstellung der Crashverhalten, da eine Simulation lediglich durch einen Punkt beschrieben wird. Es können aber nicht das zeitliche Verhalten und Bifurkationspunkte dargestellt werden. Hier zeigt sich wiederum der Vorteil der *OLioS*-Methode, die einen detaillierteren Einblick in das zeitliche Verhalten der einzelnen Simulationen ermöglicht. Die Einfärbung der Punkte beziehungsweise Linien gemäß den Resultaten der *OPioS*-Datenrepräsentation erweist sich als hilfreich, um die Algorithmen zu vergleichen.

Von den betrachteten **Skalierungsmethoden** hat sich die Zentrierung der Daten als am nützlichsten erwiesen. Die Standardisierung sorgt für Verzerrungen in den Daten, sodass dieses Verfahren ungeeignet ist.

Bei dem Vergleich der **Algorithmen zur Dimensionsreduktion** schneidet die lineare PCA am besten ab. Sie ist dazu in der Lage das Crashverhalten für den/die Ingenieur*in übersichtlich darzustellen, indem ähnliches Crashverhalten in ähnlichen Bereichen der Visualisierung eingebettet wird. Darüber hinaus handelt es sich um ein parameterfreies und deterministisches Verfahren mit einer geringen Berechnungszeit, was die Benutzerfreundlichkeit der Methode maximiert. Dagegen erweisen sich die nichtlinearen Verfahren *Isomap* und *UMAP* als ungeeignet. Diese sind stark von der Wahl der jeweiligen Hyperparameter abhängig. Bei *Isomap* sorgt ein zu kleiner Wert dafür, dass Nachbarschaften nicht hinreichend genau abgebildet werden. Zu große Werte können dagegen sogenannte Kurzschlüsse verursachen, was wiederum zu einer fehlerbehafteten Visualisierung führt. UMAP bedarf darüber hinaus der Festlegung eines zweiten Hyperparameters und erfordert zudem eine um eine Größenordnung höhere Berechnungszeit verglichen mit den anderen Algorithmen. Dies führt, gemeinsam mit seiner stochastischen Eigenschaft ebenfalls zu einem Ausschluss der Anwendung dieses Algorithmus für die Dimensionsreduktion. Auch wenn es sich bei *t-SNE* ebenfalls um einen stochastischen Algorithmus mit einem Hyperparameter handelt, zeigen sich bei Anwendung der *OPioS*-Datenrepräsentation Vorteile bei der detaillierten Analyse des Crashverhaltens. Gerade für kleine Perplexitäten ist es damit im Vergleich zur PCA möglich, unterschiedliche Crashmodi noch deutlicher in einzelnen Clustern zusammenzufassen. Darüber hinaus zeigt die Analyse des **Diskretisierungseinflusses**, dass auch bei einer groben Diskretisierung die Unterschiede zwischen den Simulationen besser als bei der linearen PCA abgebildet werden. Im Zuge der Analyse der Diskretisierung wird aufgezeigt, dass diese als Filter wirkt und die Varianz in den Daten reduziert. Dementsprechend verkleinert sich der Wertebereich sämtlicher Hauptkomponenten mit steigender Diskretisierung. Gleichzeitig kann das unterschiedliche Crashverhalten mittels der PCA nicht mehr durch nur zwei Hauptkomponenten dargestellt werden. Aufgrund der verkleinerten Varianz im Datenset ist hierfür die dritte Hauptkomponente notwendig. Da sich die Deformationen in dem verwendeten Beispiel jedoch über das gesamte Bauteil erstrecken, ist es auch mit einer groben Diskretisierung wie K_V =50mm möglich, die Crashmodi sinnvoll darzustellen. Um aber auch anderen Bauteilen und deren Deformationen

gerecht werden zu können, sollte der Anwender eine möglichst feine Diskretisierung wählen, sodass der Informationsverlust so klein wie möglich ist.

9 Verifikation des neuen Analyseansatzes an einem Gesamtfahrzeugbeispiel

In diesem Kapitel wird die Funktionsweise der entwickelten Methode an einem Gesamtfahrzeugbeispiel demonstriert. Es wird gezeigt, wie mittels des Voxel-Verfahrens diskretisierte Daten als Grundlage für die weiteren Analysen dienen, die Ausreißerdetektion auffälliges Crashverhalten automatisiert detektiert, die Dimensionsreduktion das Crashverhalten übersichtlich darstellt und der CVD ein vom Anwender vorgegebenes Crashverhalten in neuen Simulationen wiedererkennt.

9.1 Verwendete Crash-Simulationsdaten zur Verifikation des neuen Analyseansatzes

Für die Verifikation des neuen Analyseansatzes wird ein Gesamtfahrzeugmodell eines Sportwagens betrachtet. Die verwendeten Daten stammen aus Büttner (2022), in der für die Anwendung der multidisziplinären Optimierung mehrere Tausend Simulationen für verschiedene Lastfälle in Form von Versuchsplänen analysiert werden. Dabei werden die Wandstärken von 38 Bauteilen entlang des Hauptlastpfads des Rohbaus gestreut. Abbildung 82 ist aus Büttner (2022) übernommen und gibt hierzu eine Übersicht.

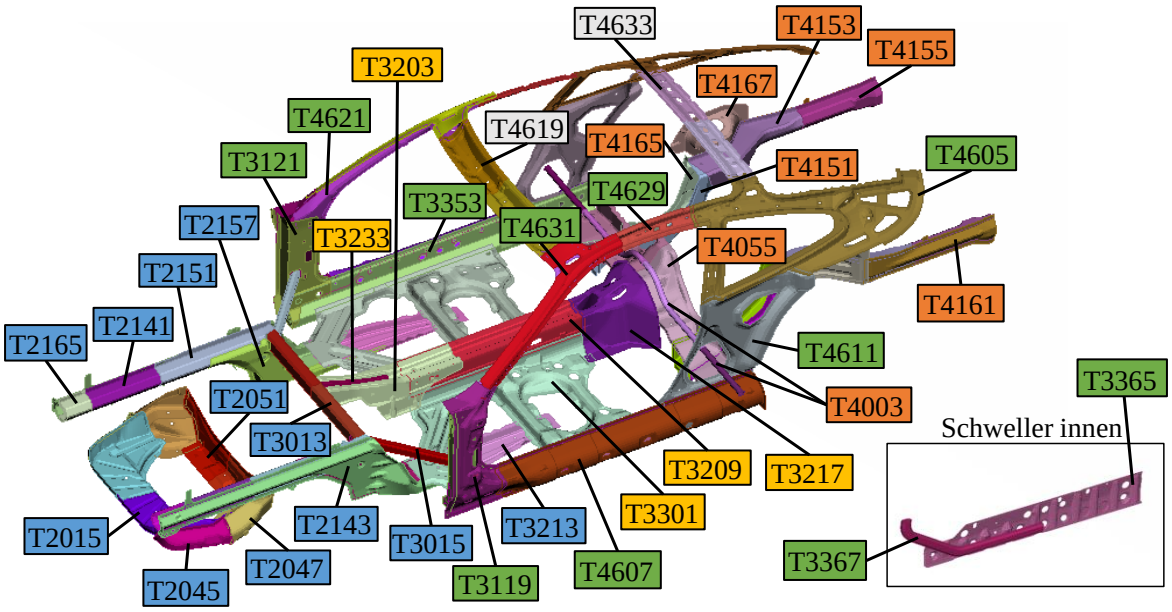


Abbildung 82: Darstellung aus Büttner (2022) übernommen. Abgebildet sind die Bauteile inklusive deren Bezeichnungen, deren Wandstärke im Rahmen des Design of Experiment in Büttner (2022) gestreut werden

Die Wandstärken warmumgeformter Rohre werden zwischen 1,5 und 5 mm gestreut, während die tiefgezogener Bleche zwischen 0,8 und 1,5 mm variiert werden. Der Crash-Lastfall *Front Offset Deformable Barrier* (ODB) (carhs GmbH 2012) wird für 120ms simuliert, wobei alle 2ms die Resultate der Simulation abgespeichert werden, sodass das Crashverhalten für 61 Zeitschritte ausgewertet werden kann. Insgesamt liegen 1500 Simulationen aus dem Versuchsplan vor, der

mittels des Optimal Latin Hypercube Sampling Verfahrens aufgestellt wird. In diesem Kapitel werden exemplarisch 50 Simulationen aus den 1500 vorliegenden betrachtet, da dies zum einen in der Praxis einer gängigen Anzahl an auszuwertenden Simulationen entspricht und zum anderen eine übersichtlichere Ergebnisdarstellung in dieser Arbeit erlaubt.

Im Folgenden wird erläutert, wie diese 50 Simulationen aus den insgesamt vorliegenden 1500 ausgewählt werden. In den Kapiteln 9.2 und 9.3 sollen Ausreißer identifiziert und unterschiedliches Crashverhalten mittels der Dimensionsreduktion visualisiert werden. Damit unterschiedliches Crashverhalten in den 50 Simulationen auftritt, ist es notwendig zunächst die sensitivste Variable aus dem Versuchsplan zu identifizieren. Es soll also herausgefunden werden, welches Bauteil durch seine Wandstärkenvariation über die größte Auswirkung auf das Crashverhalten verfügt. Auf Basis der sensitivsten Wandstärke werden im Anschluss daran die 50 Simulationen ausgewählt. Hierzu werden zunächst alle 1500 Simulationen nach aufsteigender Größe der sensitivsten Wandstärke sortiert, und im Anschluss 50 Simulationen in äquidistanten Schritten aus dem betrachteten Intervall gezogen. In (Büttner 2022) werden verschiedene Auswertungsgrößen herangezogen, um das Crashverhalten des Fahrzeugs zu bewerten. Es werden die Sensitivitäten der Wandstärkenstreuungen auf die Auswertungsgrößen untersucht. Aus den Analysen resultiert, dass das Bauteil 2151 die größten Sensitivitäten aufweist und damit den meisten Einfluss auf das ausgewertete Crashverhalten besitzt. Die 1500 Wandstärken dieses Bauteils werden in dem Versuchsplan zwischen 0,8 und 2,5mm gestreut. Aus diesem Intervall werden die 50 Simulationen in äquidistanten Schritten ausgewählt.

9.2 Ausreißerdetektion

Die Ausreißerdetektion wird automatisiert für alle Bauteile und Zeitschritte berechnet, sodass als Resultat ein kontinuierlicher Ausreißerkennwert vorliegt. Diesen kann der Anwender verwenden, um im Post-Prozessor die auffälligen Bereiche (räumlich und zeitlich) einer Simulation farblich darzustellen. Dabei wird der blau-Wert zu null gesetzt, der rot-Wert entspricht dem Ausreißerkennwert und der Grün-Wert der Differenz 1-Ausreißerkennwert.

Diese Vorgehensweise eignet sich besonders, um einen schnellen Überblick über die auffälligen Bereiche einzelner Simulationen zu erhalten. In diesem Beispiel liegen jedoch 50 Simulationen vor. Entweder schaut sich der Ingenieur nun sämtliche Simulationen und deren Ausreißerkennwerte im Post-Prozessor an, was gerade bei einer Vielzahl an Simulationen zeitintensiv ist. Darüber hinaus besteht jedoch die Möglichkeit, mittels der Ausreißerkennwerte die auffälligsten Simulationen noch vor einer Analyse im Post-Prozessor zu identifizieren, um im Anschluss daran den Fokus der Analyse gezielt auf diese zu legen.

Hierzu wird eine Kennzahl benötigt, die die Auffälligkeit einer gesamten Simulation darstellt. Eine Möglichkeit ist beispielweise, den Mittelwert der Ausreißerkennwerte über alle Bauteile und Zeitschritte einer Simulation hinweg zu bilden. Dieser ist in Abbildung 83 für die 50 Simulationen dargestellt.

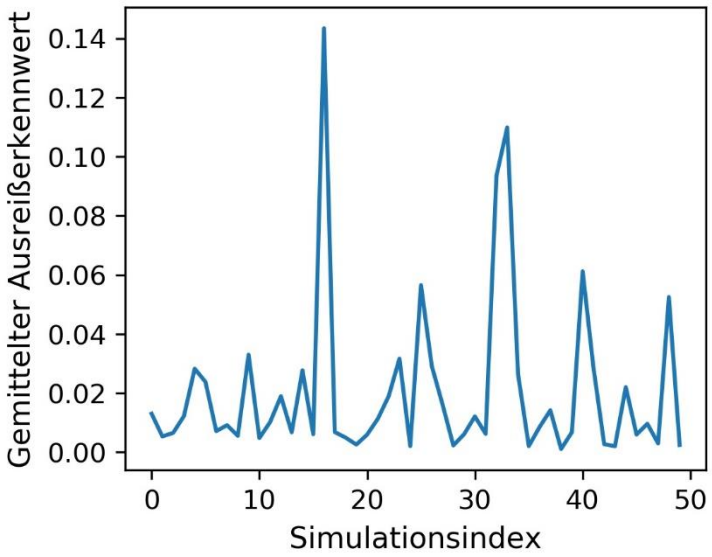


Abbildung 83: Darstellung des gemittelten Ausreißerkennwerts für die einzelnen Simulationen (Simulationsindex). Gemittelt sind die Ausreißerkennwerte der 50 Simulationen über die einzelnen Bauteile und Zeitschritte hinweg.

Aus der Darstellung geht hervor, dass die Simulation 16 mit 0,143 den höchsten gemittelten Ausreißerkennwert aufweist. Ein Beispiel für ein durchweg unauffälliges Crashverhalten ist dagegen Simulation 38, bei der mit 0,001 der niedrigste kumulierte Ausreißerkennwert vorliegt.

Abbildung 84 und Abbildung 85 zeigen für diese beiden Simulationen die farbliche Darstellung der Ausreißerkennwerte beispielhaft zum 15. Zeitschritt im Post-Prozessor. Der 15. Zeitschritt wird gewählt, da die Ausreißerdetektion zu diesem Zeitpunkt auffälliges Bauteilverhalten in Simulation 16 detektiert. Um die Bauteile besser in den Abbildungen identifizieren zu können, wird die undeformierte Geometrie für die Einfärbung der Ausreißerkennwerte verwendet.

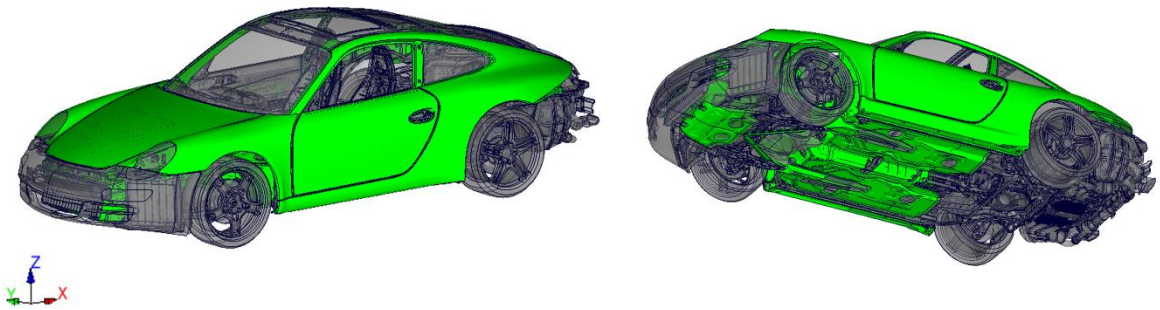


Abbildung 84: Visualisierung der Ausreißerkennwerte für die unauffällige Simulation 38 zum 15. Zeitschritt. Grün entspricht unauffälligen Bauteilen, Rot auffälligen

Aus Abbildung 84 geht hervor, dass für die Simulation 38 keine Bereiche mit auffälligem Crashverhalten zum 15. Zeitschritt identifiziert werden können. Da der Ausreißerkennwert für sämtliche Bauteile nahe 0 ist, werden sie gemäß der oben dargestellten RGB-Konvertierung in grün dargestellt.

Abbildung 85 zeigt exemplarisch das Resultat der Ausreißerdetektion für die Ausreißer Simulation 16 zum 15. Zeitschritt. Darin weist zwar der Großteil der Bauteile ebenfalls einen niedrigen Aureißerkennwert auf, einige Bauteile stechen jedoch sehr deutlich durch ihr Crashverhalten hervor, weshalb diese in rot abgebildet werden.

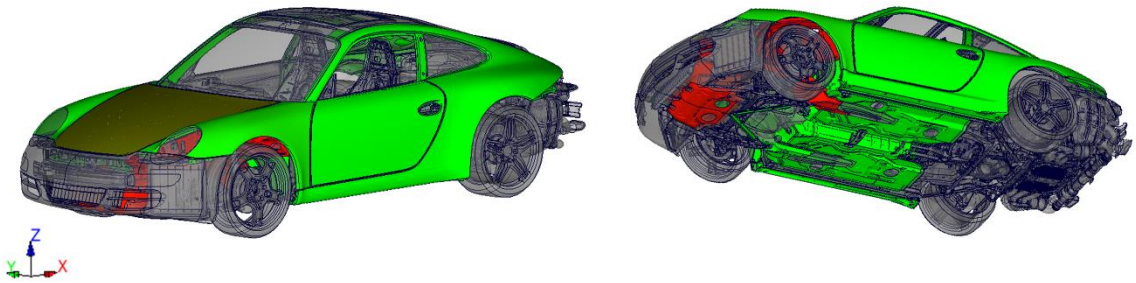


Abbildung 85: Visualisierung der Ausreißerkennwerte für die Ausreißer Simulation 16 zum 15. Zeitschritt. Grün entspricht unauffälligen Bauteilen, Rot auffälligen

Die Bauteile im vorderen Bereich auf der linken Fahrzeugseite fallen dabei besonders auf. Es ist gleichzeitig zu sehen, dass eine farbliche Abstufung unter den Bauteilen auftritt. Während der *Kofferraumdeckel* in einem eher bräunlichen grün dargestellt ist, zeigt beispielsweise die *Kofferraumwanne* ein tiefes rot, was hinsichtlich des Crashverhaltens im Vergleich zu den anderen Simulationen auf einen sehr deutlichen Ausreißer hindeutet.

Diese farbliche Darstellung im Post-Prozessor kann der/die Ingenieur*in in der praktischen Anwendung der Ausreißerdetektion für jeden Zeitschritt betrachten und erhält so Hinweise auf die besonders auffälligen Bereiche einer Crashsimulation (räumlich und zeitlich aufgelöst). Im Folgenden werden die drei Bauteile mit den höchsten Ausreißerkennwerten zum 15. Zeitschritt näher betrachtet.

Abbildung 86 stellt die Ausreißerkennwerte dieser Bauteile jeweils in Form von Bildern dar. Auf der horizontalen Achse ist das zeitliche Verhalten des Ausreißerkennwerts berücksichtigt. Jede Zeile des Bildes stellt die Ausreißerkennwerte einer Simulation dar. Die Nummerierung der Simulationen beginnt dabei in der ersten Zeile mit dem Wert 0 und endet in der untersten Zeile mit dem Wert 49. Desto höher der Ausreißerkennwert ist, desto heller sind die Farben in dem Bild. Unauffällige Zeitschritte und Simulationen sind in einem dunklen Farbton abgebildet.

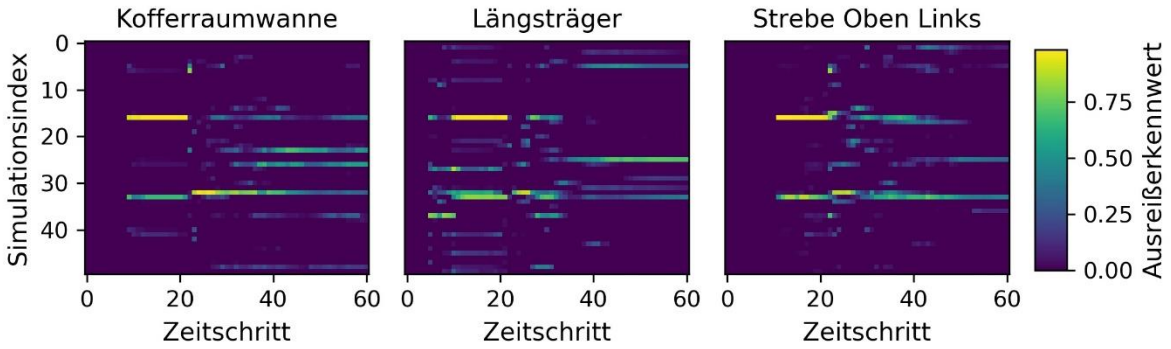


Abbildung 86: Darstellung der Ausreißerkennwerte für die drei auffälligen Bauteile Kofferraumwanne, Längsträger, Strebe Oben Links. Auf der horizontalen Achse sind die Zeitschritte dargestellt, auf der vertikalen der Simulationsindex

Für alle drei Bauteile sticht der Bereich zwischen dem 10. und 20. Zeitschritt für die Simulationen 16 und 33 als besonders auffällig (hell) hervor. Simulation 38 ist dagegen durchweg unauffällig (dunkel).

Bis hierhin bleibt jedoch noch unklar, warum die Bauteile als Ausreißer hervorstechen. Daher wird im Folgenden die Dimensionsreduktion angewendet, um eine Übersicht über das Crashverhalten dieser drei Bauteile in sämtlichen Simulationen zu erhalten.

9.3 Dimensionsreduktion

Um dem Anwender gerade bei einer Vielzahl an vorhandenen Simulationen einen Überblick über das Crashverhalten einzelner Bauteile verschaffen zu können, werden im Folgenden die Ergebnisse der PCA für die *OPioS*- und *OLioS*-Datenrepräsentation betrachtet. Dabei steht insbesondere die Frage im Vordergrund, weshalb die drei Bauteile in der Simulation 16 als Ausreißer detektiert werden.

Abbildung 87 stellt die Ergebnisse der *OPioS*-Datenrepräsentation mittels der ersten beiden Hauptkomponenten der PCA dar. Farblich hervorgehoben ist dabei die Ausreißer Simulation 16 sowie die unauffällige Simulation 38, die beispielhaft im vorangegangenen Kapitel vorgestellt wurde. Für keines der Bauteile hebt sich die Ausreißer Simulation von den anderen ab. Die Erwartung an die Dimensionsreduktion ist jedoch, dass die einzelnen Simulationen gemäß der Ähnlichkeit ihres Crashverhaltens in den neuen Dimensionen dargestellt werden können und damit auch das auffällige Crashverhalten deutlich hervortritt.

Der Grund dafür, dass die Ausreißer Simulation sich nicht deutlich vom Rest abhebt, liegt in der Art begründet, wie das zeitliche Verhalten bei *OPioS* berücksichtigt wird. Die Informationen sämtlicher Voxel und Zeitschritte werden als beschreibende Merkmale im hochdimensionalen Raum verwendet.

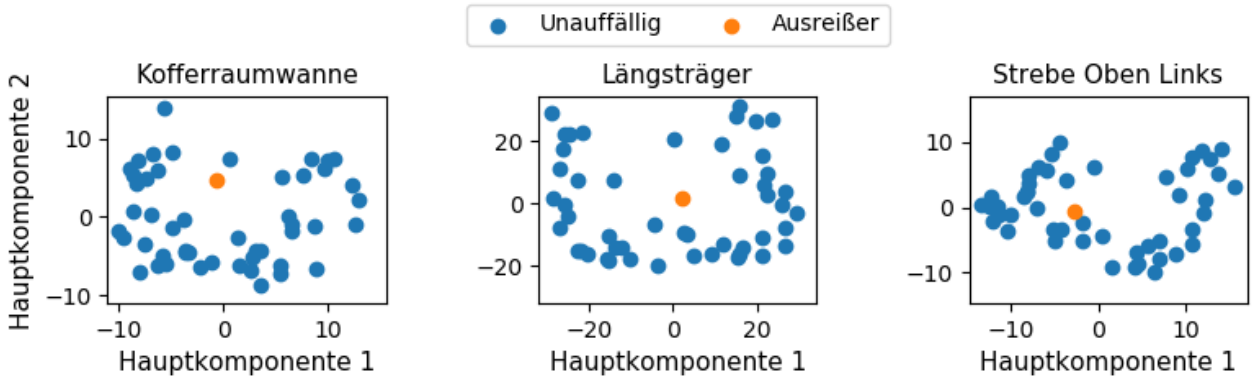


Abbildung 87: Visualisierung der Dimensionsreduktion mittels der PCA für die OPioS Datenrepräsentation. Dargestellt sind die ersten beiden Hauptkomponenten in Form von Streudiagrammen

Der Ausreißer sticht vor allem in den Zeitschritten 10-20, wie aus Abbildung 86 ersichtlich wird, für die drei Bauteile hervor. Mit fortschreitender Zeit im Crash sinken die Ausreißerkennwerte wieder ab (teilweise bis auf den Wert Null). Zu den frühen Zeitschritten im Crash liegen jedoch noch vergleichsweise geringe plastische Dehnungen vor. Desto stärker das Fahrzeug während des Crashs deformiert, desto höher steigen auch die Werte der plastischen Dehnungen. Dementsprechend stärker sind auch die Streuungen des Crashverhaltens zu den späten Zeitschritten. Da es sich bei der PCA um ein varianzbasiertes Verfahren handelt, werden die neuen Hauptachsen so gewählt, dass deren Varianz maximiert wird. Daher erhalten die Voxel zu den späten Zeitschritten mit den höheren Streuungen einen stärkeren Einfluss bei der Berechnung der neuen Merkmale. Effekte, die zu frühen Zeitschritten im Crash deutlich hervorstechen, verlieren dadurch zumindest für die ersten beiden Hauptkomponenten der niedrigdimensionalen Darstellung an Bedeutung.

Daher wird erwartet, dass die Information über das auffällige Crashverhalten zu den frühen Zeitschritten in anderen Hauptkomponenten enthalten ist und sichtbar wird. Um dies zu plausibilisieren, sind in Abbildung 88 die ersten 10 Hauptkomponenten für jedes der drei Bauteile in einem *Parallel Coordinates Plot* dargestellt.

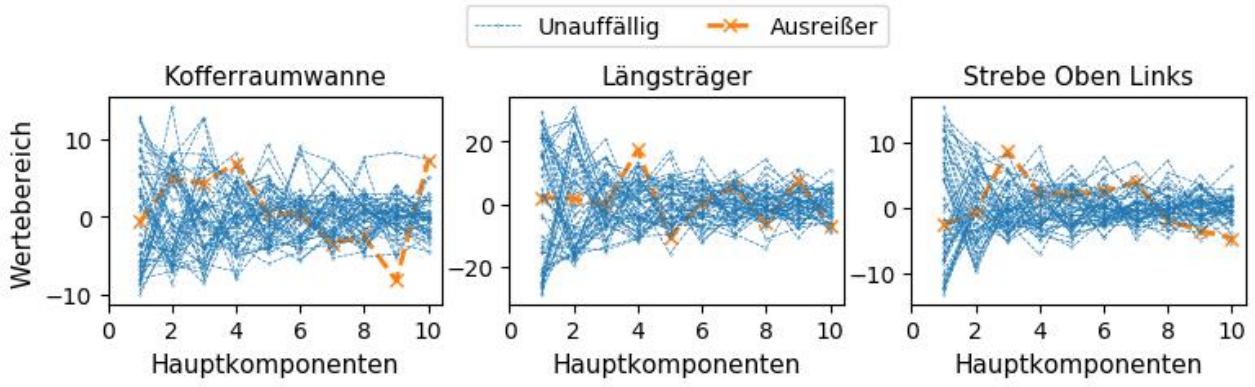


Abbildung 88: Visualisierung der Dimensionsreduktion mittels der PCA für die OPioS Datenrepräsentation. Dargestellt sind die ersten 10 Hauptkomponenten in Form von Parallel Coordinates Plots

Auf der horizontalen Achse sind die einzelnen Hauptkomponenten abgebildet, während auf der vertikalen Achse die zugehörigen Werte der einzelnen Simulationen dargestellt sind. Die einzelnen Punkte der verschiedenen Hauptkomponenten werden durch Linien miteinander verbunden. Diese zeigen jeweils die Eigenschaften einer einzelnen Simulation an. Erneut ist die Ausreißer Simulation farblich hervorgehoben. Mithilfe dieser Darstellung kann identifiziert werden, welche Hauptkomponente die Eigenschaften über die frühen Zeitschritte und das auffällige Crashverhalten der einzelnen Bauteile beinhaltet. Für die *Kofferraumwanne* sticht die Ausreißer Simulation für die Hauptkomponenten 4, 9 und 10 hervor, für den *Längsträger* die Hauptkomponente 4 und für die *Strebe Oben Links* die Hauptkomponente 3.

Wird die Dimensionsreduktion beispielsweise lediglich für den Zeitschritt 15 durchgeführt, ergeben sich die dimensionsreduzierten Darstellungen in Abbildung 89 anhand der ersten beiden Hauptkomponenten. Die farbliche Darstellung resultiert aus den ersten drei Hauptkomponenten, die für die RGB-Farbcodierung verwendet wird. Hier stechen die beiden Ausreißer Simulationen 16 und 33 sofort deutlich hervor. Für die *Kofferraumwanne* und den *Längsträger* zeigen die anderen Simulationen eine Transition entlang der ersten beiden Hauptkomponenten, während die anderen Simulation für die *Strebe Oben Links* in einem sehr dichten Cluster eingebettet werden.

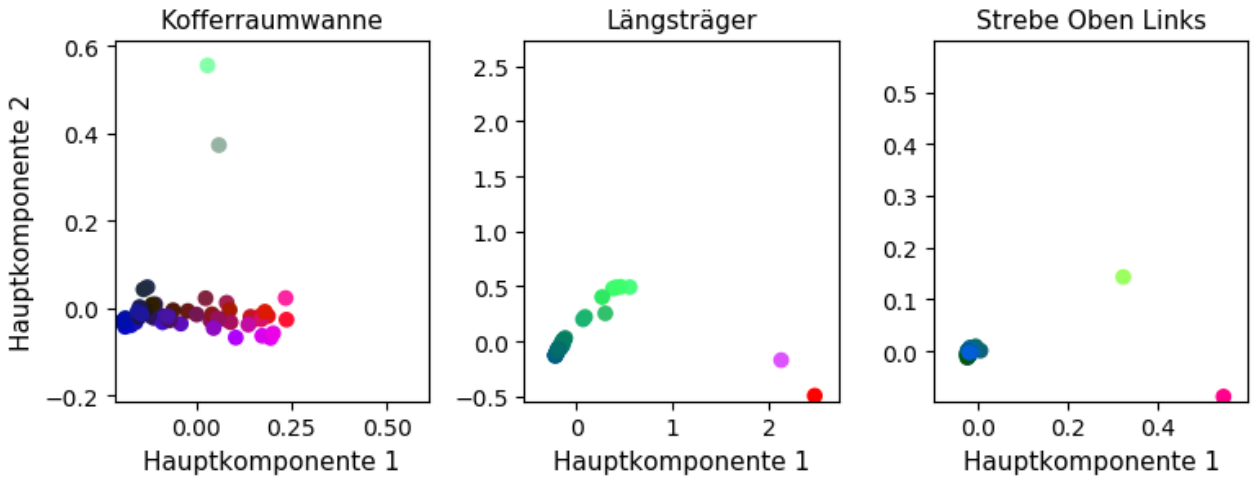


Abbildung 89: Visualisierung der Dimensionsreduktion mittels der PCA für den Zeitschritt 15. Dargestellt sind die ersten beiden Hauptkomponenten in Form von Streudiagrammen

Bis hierhin wird jedoch lediglich nachgewiesen, dass auch die Ausreißer Simulationen mittels der Dimensionsreduktion entsprechend entfernt von den unauffälligen Simulationen dargestellt werden können. Es wird daraus aber noch nicht klar, inwiefern sich deren Crashverhalten konkret unterscheidet. Diesen Überblick erlangt der/die Ingenieur*in dank einer interaktiven Darstellung, die zu jedem Punkt, der einer Simulation entspricht, entsprechende Metainformationen wie beispielsweise Momentaufnahmen zu bestimmten Zeitschritten der Simulation einblendet. So stellen Abbildung 90, Abbildung 91 und Abbildung 92 im Folgenden zusätzlich zur niedrigdimensionalen Abbildung das Crashverhalten zum Zeitschritt 15 in Form von Momentaufnahmen des Bauteils dar. Dabei sind die plastischen Dehnungen farblich auf einer Skala zwischen 0 und 0,05 abgebildet (0-5%).

Die *Kofferraumwanne* zeigt in Abbildung 90 insbesondere entlang der ersten Hauptkomponente eine Streuung der unauffälligen Simulationen. Bei der Betrachtung der entsprechenden Momentaufnahmen ist ersichtlich, dass die Simulationen im unteren linken Bereich geringere plastische Dehnungen im Vergleich zu denen im unteren rechten Bereich aufweisen. Entlang der ersten Hauptkomponente steigt also die plastische Verformung des Bauteils (von links nach rechts) im Bild an. Zudem wird so ersichtlich, weshalb die Simulationen 16 und 33 von der Ausreißerdetektion als auffällig detektiert werden. Zusätzlich zu der Deformation im unteren rechten Bereich, tritt eine deutlich stärkere Deformation im oberen rechten Bereich der *Kofferraumwanne* auf, die in dieser Ausmaß sonst in keiner anderen der Simulationen zu beobachten ist.

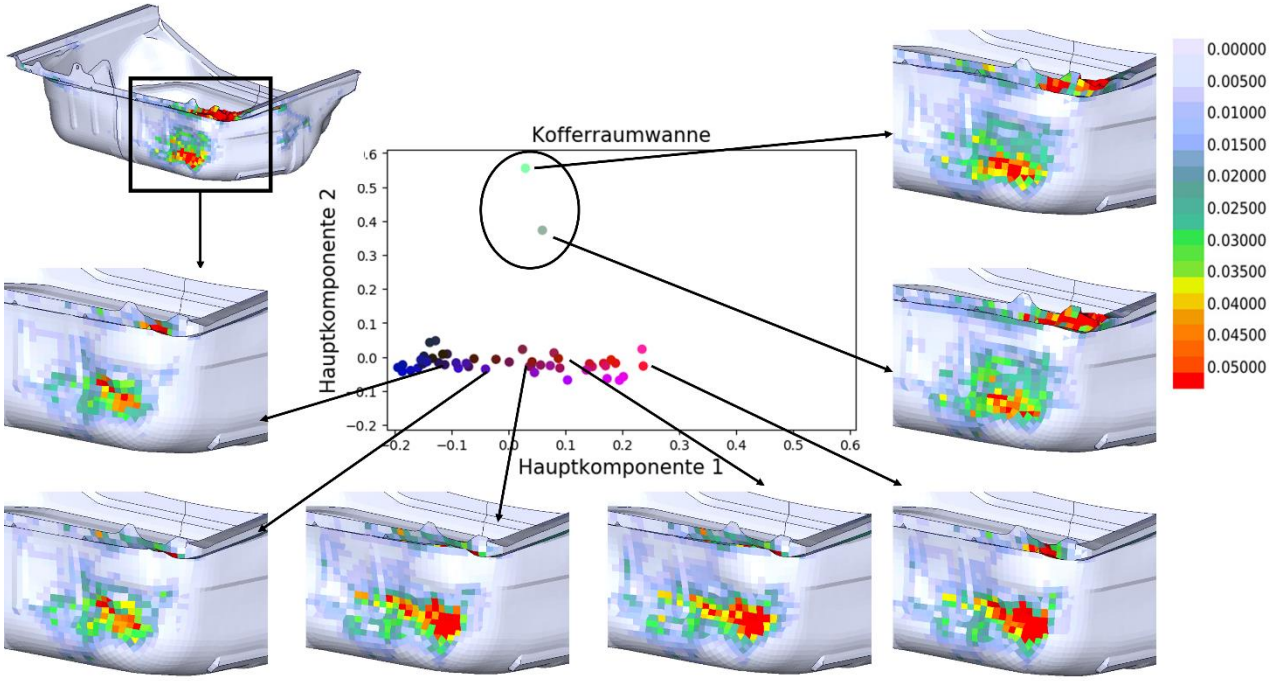


Abbildung 90: Verknüpfung der dimensionsreduzierten Darstellung mit dem Crashverhalten ausgewählter Simulationen für die Kofferraumwanne. Farblich dargestellt sind die plastischen Dehnungen mit einer Skala zwischen 0 und 0,05 (0-5%)

Der *Längsträger* zeigt ebenfalls eine Transition entlang der ersten beiden Hauptkomponenten (Abbildung 91). Dieser wird bei den Simulationen im unteren linken Bereich der dimensionsreduzierten Darstellung kaum plastisch deformiert. Mit steigenden Werten für die ersten beiden Hauptkomponenten steigt jedoch gleichzeitig die plastische Verformung im vorderen Bereich des Bauteils an. Die beiden Ausreißer Simulationen weisen ein deutlich anderes Crashverhalten mit wesentlich höheren Verformungen auf, die sich über einen größeren Bereich des Bauteils verteilen.

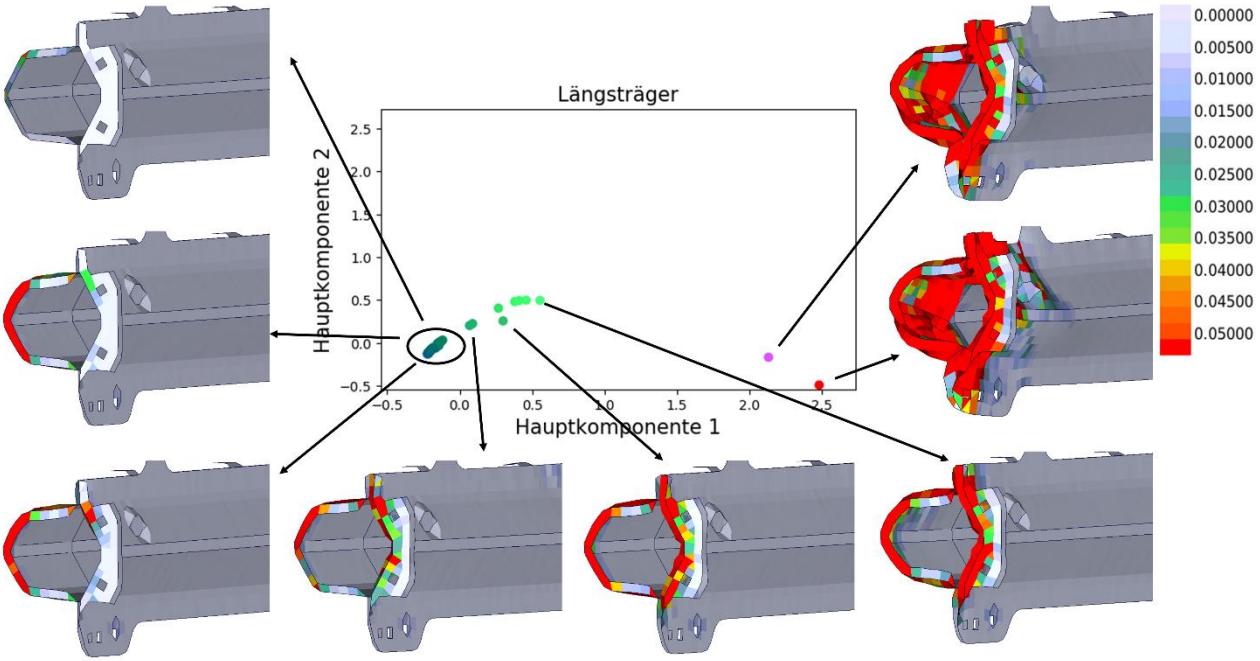


Abbildung 91: Verknüpfung der dimensionsreduzierten Darstellung mit dem Crashverhalten ausgewählter Simulationen für den Längsträger. Farblich dargestellt sind die plastischen Dehnungen mit einer Skala zwischen 0 und 0,05 (0-5%)

Die meisten Simulationen der *Strebe Oben Links* formen ein dicht gepacktes Cluster im unteren linken Bereich der Abbildung 92, wobei sich die beiden Ausreißer deutlich davon abheben und unten rechts in der Darstellung wiederzufinden sind. Zwischen den unauffälligen Simulationen im Cluster unten links zeigen sich keine wesentlichen Unterschiede. Nur in wenigen vereinzelten Bereichen sind niedrige plastische Dehnungen zu erkennen, die sich zwischen den Simulationen in diesem Cluster jedoch kaum voneinander unterscheiden. Die beiden Ausreißer zeigen dagegen erneut ein deutlich auffälliges Crashverhalten, indem wie beim *Längsträger* und den anderen Simulationen stärkere Deformationen im vorderen Bereich des Bauteils auftreten.

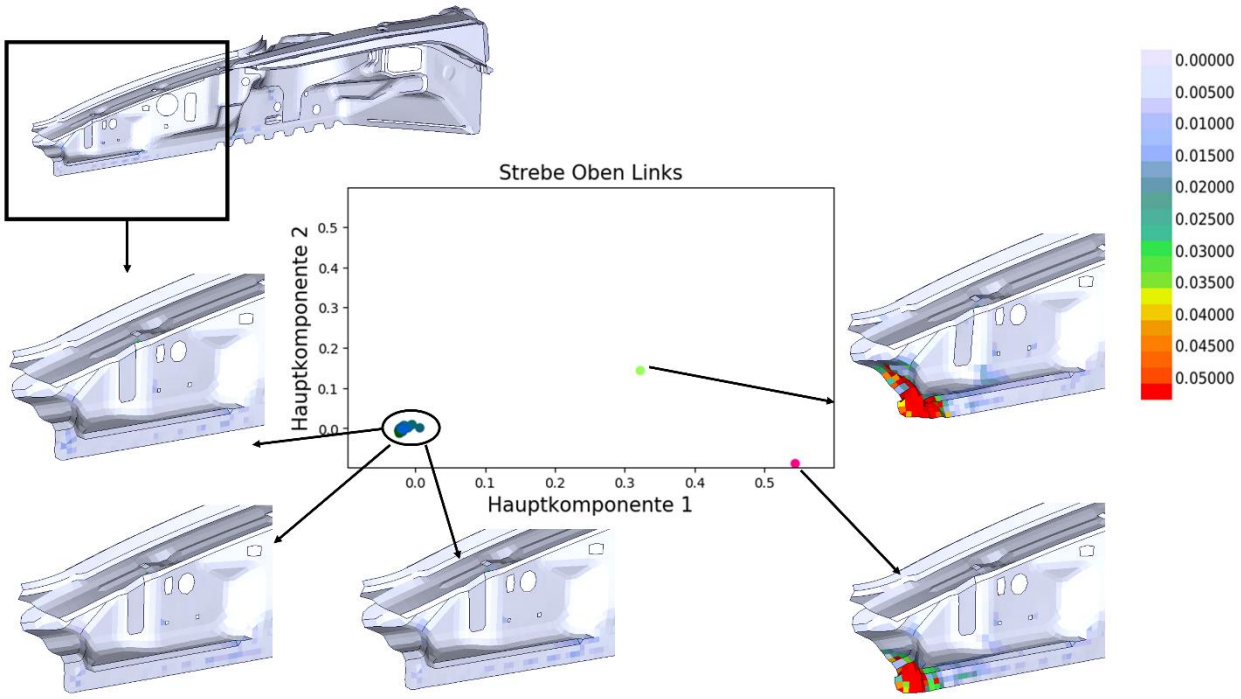


Abbildung 92: Verknüpfung der dimensionsreduzierten Darstellung mit dem Crashverhalten ausgewählter Simulationen für die Strebe Oben Links. Farblich dargestellt sind die plastischen Dehnungen mit einer Skala zwischen 0 und 0,05 (0-5%)

Nachdem nun das Crashverhalten der Simulationen für die drei Bauteile analysiert ist sowie die Ergebnisse der Ausreißerdetektion bestätigt sind, folgt die Betrachtung der aus der *OLioS*-Datenrepräsentation resultierenden dimensionsreduzierten Darstellung in Abbildung 93. Hier entspricht der Verlauf einer Linie dem zeitlichen Crashverhalten einer Simulation. Für alle drei Bauteile sind die beiden Ausreißer Simulationen durch längere Linien eindeutig zu identifizieren. Die farbliche Darstellung gemäß der ersten drei Hauptkomponenten zeigt gleichzeitig, dass auch zwischen den anderen Linien wie in den vorherigen Abbildungen Transitionen (*Kofferraumwanne* und *Längsträger*) beziehungsweise Cluster (*Schräge Strebe links*) zu erkennen sind. Die ersten beiden Hauptkomponenten spiegeln bei der *Kofferraumwanne* die Stärke der plastischen Deformation wider, weshalb insbesondere die Länge der Simulationen variiert. Die blauen Linien sind am kürzesten, während die roten Linien den Simulationen mit den stärkeren Deformationen entsprechen und länger sind. Dazwischen ist ein kontinuierlicher Übergang zu erkennen. Die anderen beiden Bauteile zeigen diese Transitionen weniger deutlich. Insbesondere bei der *Strebe Oben Links* verlaufen sämtliche Linien der unauffälligen Simulationen in dieselbe Richtung und variieren auch in der Länge kaum. Dies bestätigt die Ergebnisse aus den vorherigen Betrachtungen wonach bei diesem Bauteil ein sehr ähnliches Crashverhalten in sämtlichen Simulationen vorliegt.

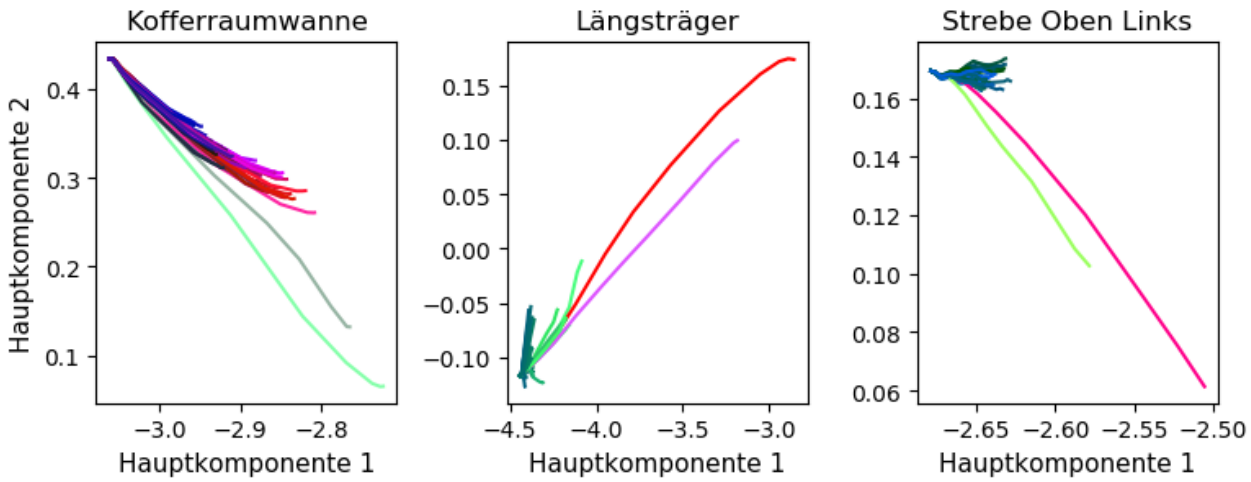


Abbildung 93: Visualisierung der Dimensionsreduktion mittels der PCA für die OLioS Datenrepräsentation. Dargestellt sind die ersten beiden Hauptkomponenten in Form von Liniendiagrammen

9.4 Crash-Verhalten-Detektor

Im Folgenden wird ergänzend aufgezeigt, wie ein vom Anwender vorgegebenes Crashverhalten in neuen Simulationen mittels des Crash-Verhalten-Detektors (CVD) wiedererkannt werden kann. Hierzu wird beispielhaft das Bauteil *Kofferraumwanne* verwendet. Die Bezeichnungen für die beiden auftretenden Crashverhalten werden aus Gründen der Konsistenz aus den vorangegangenen Untersuchungen übernommen und mit „Unauffällig“ beziehungsweise „Ausreißer“ bezeichnet. Bisher wurden aus den 1500 Simulationen aus (Büttner 2022) 50 Simulationen ausgewählt und im Detail analysiert. Dadurch konnten zwei Ausreißer Simulationen identifiziert werden. Es stellt sich jedoch die Frage, ob noch weitere der übrigen 1450 Simulationen des ODB Lastfalls das beobachtete Ausreißer Verhalten zeigen.

Zunächst wird der erste Teil des CVDs mit den nicht kategorisierten Daten trainiert, um relevante Merkmale zu extrahieren. Da dies unüberwacht erfolgt, können sämtliche vorliegenden Simulationen verwendet werden; in diesem Fall also die bisher betrachteten 50. Im Anschluss daran wird der überwachte Klassifikator trainiert. Da die Qualität der prädizierten Ergebnisse mit steigender Anzahl an Trainingsdaten zunimmt und die Prädiktionen insgesamt robuster werden, werden beide bisher identifizierten Ausreißer Simulationen sowie zwei weitere beliebige unauffällige Simulationen kategorisiert, sodass insgesamt vier Trainingsinstanzen vorliegen.

Der CVD zeigt bei der Analyse der 1450 Simulationen, dass zwei weitere Simulationen das beobachtete Ausreißerverhalten aufweisen. Deren *Kofferraumwanne* ist zum Zeitschritt 15 in Abbildung 94 dargestellt. Daraus geht hervor, dass das Bauteil zusätzlich zum unteren rechten Bereich auch im oberen rechten Bereich deutliche plastische Deformationen aufweist. Im vorangegangenen Abschnitt wurde dieses Verhalten mittels der Dimensionsreduktion analysiert und als deutliche Charakteristik eines Ausreißer identifiziert. Dieses Ergebnis zeigt, dass der CVD dazu in der Lage ist, das Ausreißer Crashverhalten wiederzuerkennen.

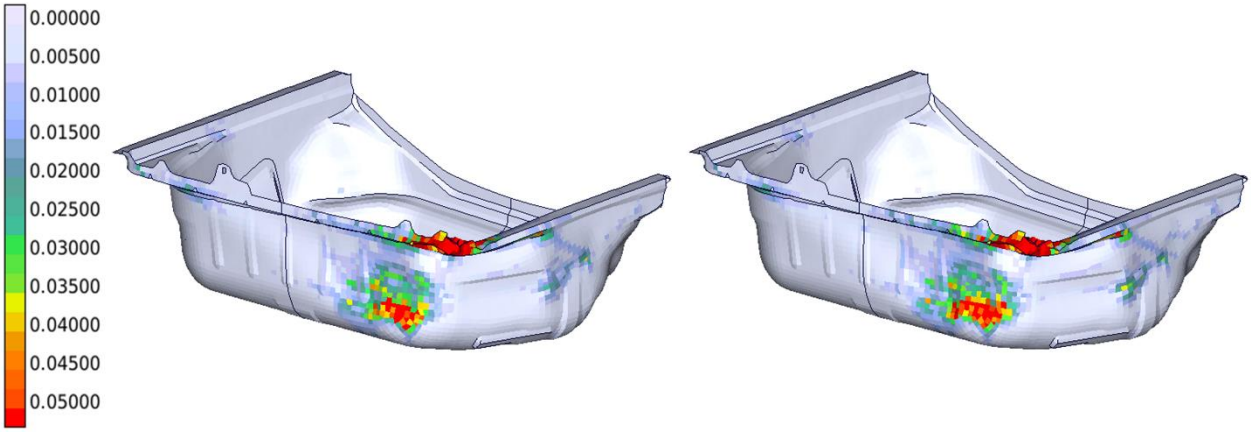


Abbildung 94: Kofferraumwanne der Simulationen 603 und 955 mit dem Crashverhalten „Ausreißer“, welche vom Crash-Verhalten-Detektor in 1450 weiteren Simulationen identifiziert wurden

Da es sich bei dem CVD um ein stochastisches Modell handelt, kann mit diesem Ergebnis allerdings noch nicht die Robustheit dessen Prädiktionen bewertet werden. Darüber hinaus können andere der 48 unauffälligen Simulationen für das Trainieren des Klassifikators verwendet werden. Variierende Trainingsinstanzen können dabei ebenfalls einen wesentlichen Einfluss auf die Prädiktionen haben. Darüber hinaus soll betrachtet werden, inwiefern sich die Ergebnisse unterscheiden, wenn zwei beziehungsweise vier kategorisierte Trainingsdaten zu Verfügung gestellt werden. Aus diesem Grund werden im Folgenden 100 Wiederholungen der Klassifikation mit jeweils unterschiedlichen Trainingsinstanzen durchgeführt.

Zunächst wird der Fall betrachtet, in dem lediglich zwei kategorisierte Daten vorliegen. Dies ist der für den Anwender im Hinblick auf die Vorarbeit „Kategorisieren“ am wenigsten aufwändige Fall, da pro Kategorie nur eine Simulation zur Verfügung gestellt werden muss. Für jede der 100 Wiederholungen wird eine zufällige Simulation aus den 48 unauffälligen sowie eine aus den 2 Ausreißer Simulationen für das Trainieren des Klassifikators genutzt. Der trainierte CVD wird anhand der zwei neu aufgefundenen Ausreißer Simulationen getestet. In dieser Arbeit soll dieselbe Anzahl unauffälliger Simulationen für den Test berücksichtigt werden, weshalb zwei weitere, unauffällige Simulationen aus den 1450 Testdaten verwendet werden.

Die Ergebnisse sind in Abbildung 95 als Verteilungsdiagramme für die 100 Wiederholungen dargestellt. Im linken Diagramm ist die Genauigkeit für die unauffälligen Simulationen abgebildet, während im rechten, diejenige für die Ausreißer zu sehen ist. Die Genauigkeit eins bedeutet, dass mit dem trainierten CVD beide unauffälligen Simulationen (links) beziehungsweise Ausreißer (rechts) aus den Testdaten korrekt prädiziert werden. Eine Genauigkeit von null bedeutet, dass alle Testdaten falsch klassifiziert werden.

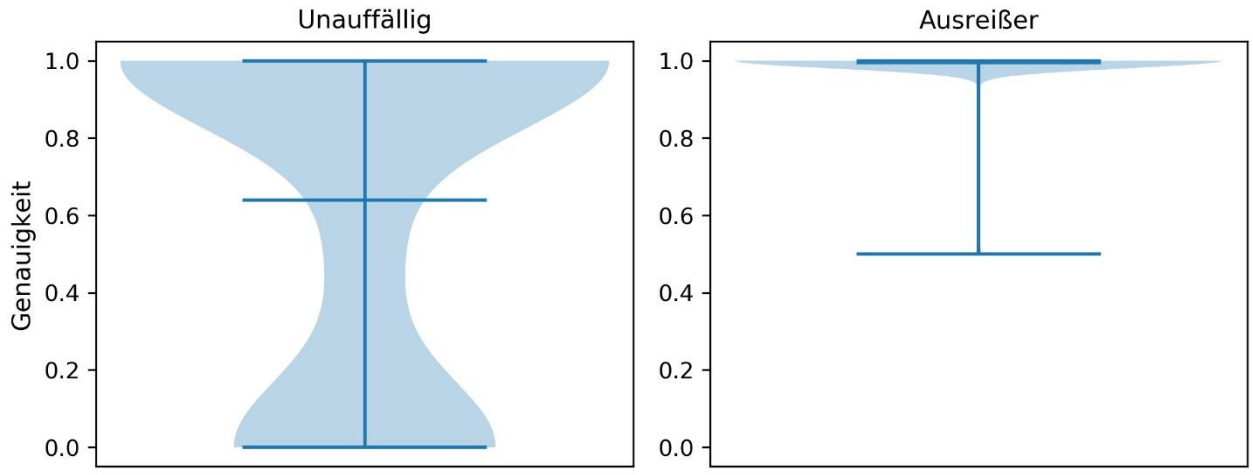


Abbildung 95: Verteilung der Genauigkeiten für die 100 Wiederholungen mit unterschiedlichen Dateninstanzen. Links für die unauffälligen Simulationen aus dem Testdatenset, rechts für die Ausreißer. Es wird mit je einem Ausreißer und einer unauffälligen Simulation aus dem Trainingsdatenset trainiert.

Aus der linken Abbildung geht hervor, dass einige der Klassifikatoren eine Genauigkeit nahe eins aufweisen, jedoch gleichzeitig eine Vielzahl der Modelle niedrige Genauigkeiten nahe null liefern. Der Mittelwert aller 100 Modelle beträgt 0,64. Mit diesem CVD sind damit keine zuverlässigen Prädiktionen der unauffälligen Simulationen möglich.

Die rechte Abbildung zeigt dagegen eine deutlich schärfere Verteilung der Genauigkeiten für die Ausreißer Simulationen. Hier zeigt das schlechteste Modell eine Genauigkeit von 0,5 auf, was bedeutet, dass einer der beiden Ausreißer im Testdatenset detektiert wird. Die meisten Modelle liefern jedoch Genauigkeiten von etwa eins. Der Mittelwert beträgt 0,995, was zeigt, dass dieser CVD dazu in der Lage ist, zuverlässig die Simulationen mit dem Ausreißer Crashverhalten zu identifizieren.

Nachdem bei der Verwendung von zwei kategorisierten Daten, die Ausreißer zwar zuverlässig detektiert werden können, unauffällige Simulationen dagegen aber nicht, wird im Folgenden der Fall betrachtet, bei dem vier kategorisierte Daten durch den Anwender für das Training des Klassifikators bereitgestellt werden. Dazu werden für die 100 Wiederholungen zufällig zwei Simulationen aus den 48 unauffälligen sowie beide Ausreißer Simulationen als Trainingsdaten verwendet. Das Testdatenset ist dasselbe wie bei den Experimenten mit nur 2 kategorisierten Trainingsdaten, damit die Ergebnisse vergleichbar sind.

Die Resultate sind in Abbildung 96 links für die unauffälligen und rechts für die Ausreißer Simulationen dargestellt. Aus beiden Darstellungen geht hervor, dass sämtliche 100 Modelle Genauigkeiten von exakt eins aufweisen. Dies bedeutet, dass sowohl unauffälliges als auch Ausreißer Crashverhalten mithilfe dieser Vorgehensweise zuverlässig in neuen Daten detektiert wird.

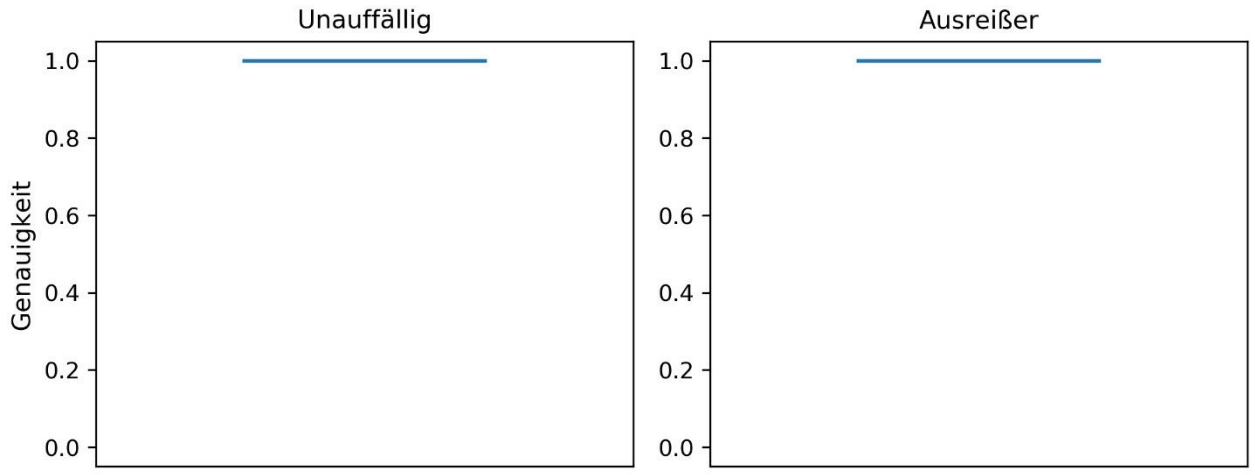


Abbildung 96: Verteilung der Genauigkeiten für die 100 Wiederholungen mit unterschiedlichen Dateninstanzen. Links für die unauffälligen Simulationen aus dem Testdatenset, rechts für die Ausreißer. Es wird mit je zwei Ausreißern und zwei unauffälligen Simulationen aus dem Trainingsdatenset trainiert.

Damit zeigt der CVD auch für dieses Anwendungsbeispiel, dass er dazu in der Lage ist, ein vorgegebenes Crashverhalten in neuen Simulationen automatisiert wiederzuerkennen. Werden lediglich zwei kategorisierte Daten für das Trainieren des Klassifikators verwendet, zeigen stochastische Einflüsse und unterschiedliche Trainingsdaten einen großen Einfluss auf die Prädiktionen. Werden hingegen vier kategorisierte Daten zur Verfügung gestellt, wird bereits das bestmögliche Ergebnis der Klassifikation erzielt, indem für sämtliche Wiederholungen und unterschiedliche Trainingsdaten Genauigkeiten in Höhe von 1 erreicht werden.

10 Zusammenfassung und Ausblick

10.1 Zusammenfassung

In dieser Arbeit wird ein neuer Analyseansatz vorgestellt, der den/die Ingenieur*in bei der Analyse von Crashsimulationen unterstützt. Bei der bisherigen Auswertung handelt es sich um einen sehr zeitaufwändigen Prozess und infolgedessen auch um eine unvollständige Verarbeitung sämtlicher vorliegender Informationen aus den Simulationen. Es existierte bislang keine automatisierte Auswertung, die sämtliche Informationen analysiert und dem Anwender anschließend übersichtlich darstellt. Ein Grund hierfür ist, dass sich die FE-Netze der einzelnen Simulationen unterscheiden und damit deren Vergleichbarkeit und automatisierte maschinelle Auswertung erschwert ist. Es werden jedoch vier Methoden vorgestellt, die diese Herausforderungen adressieren und entweder in der automatisierten Analyse ablaufen können, sobald eine neue Simulation berechnet ist oder im manuellen Analyseteil verortet sind. In der automatisierten Analyse ist zunächst die Diskretisierung der Bauteile durchzuführen, die das Ziel einer einheitlichen Datenrepräsentation verfolgt. Zudem werden Momentaufnahmen erzeugt, die mittels der Dimensionsreduktion später als Metainformationen für die interaktive Visualisierung des Crashverhaltens verwendet werden. Darüber hinaus wird die Ausreißerdetektion automatisiert berechnet, die in einem festgelegten Kontext (z.B. letzte 30 Simulationen) auffälliges Crashverhalten detektiert. Sie kann zudem im Rahmen der manuellen Analyse neu berechnet werden, wenn den Anwender beispielsweise die Auswertung in einem anderen Kontext an zu vergleichenden Simulationen interessiert. Daher wird als Anforderung an die Ausreißerdetektion eine geringe Berechnungszeit gestellt, sodass möglichst keine Wartezeiten auftreten und ein interaktives Arbeiten mit den Daten sichergestellt wird. Darüber hinaus können in der manuellen Analyse die Ausreißerkennwerte der Bauteile im Post-Prozessor farblich dargestellt werden, sodass es möglich wird, auffällige Bereiche sofort zu identifizieren.

Auch der Crash-Verhalten-Detektor ist in der manuellen Analyse zu verorten. Sobald der Anwender neues Crashverhalten für den CVD markiert, wird im Hintergrund ein entsprechendes Modell trainiert, das in neuen Simulationen dazu in der Lage ist, dieses vorgegebene Crashverhalten wiederzuerkennen. Die Resultate dieser Klassifikation werden ebenfalls farblich im Post-Prozessor visualisiert. Zuletzt findet die Dimensionsreduktion im manuellen Analyseteil Anwendung. Die Simulationen werden anschließend in einer niedrigdimensionalen Darstellung gemäß der Ähnlichkeit ihres Crashverhaltens übersichtlich dargestellt, sodass der Anwender einen anschaulichen Überblick über eine Vielzahl von Simulationen erhält.

Im Folgenden werden die Erkenntnisse dieser Arbeit zu den vier Methoden zusammengefasst:

Das Untersuchungsziel der Datenrepräsentation ist die einheitliche Darstellung der FE-Daten durch Diskretisierungs-Verfahren, sodass eine weitere maschinelle Datenverarbeitung gewährleistet ist. Hierfür wird das zweidimensionale Sphären-Verfahren der dreidimensionalen Voxel-Diskretisierung gegenübergestellt. Dabei werden als Anforderung an das Diskretisierungsverfahren ein geringer Informationsverlust der originalen Daten sowie eine einfache Interpretierbarkeit der neuen Datenrepräsentation und des Einflusses durch den Diskretisierungsparameter gestellt. Darüber hinaus soll nur ein geringer Speicherbedarf durch die

neue Datenrepräsentation beansprucht werden. Um den Informationsverlust zu bewerten, wird ein Qualitätskriterium verwendet, das aus der Dimensionsreduktion dafür bekannt ist, verschiedene Algorithmen und deren Hyperparameter zu vergleichen. Die Analyse einer Vielzahl an Bauteilen aus einem Vorderwagenabschnittsmodell zeigt, dass die sphärische Projektion zu Verzerrungen der FE-Daten führt. Dadurch, dass die vormals dreidimensionalen Daten auf eine zweidimensionale Kugeloberfläche projiziert werden, treten in unterschiedlichen Bereichen (insbesondere bei profilförmigen Bauteilen) Häufungen der FE auf, während in anderen Bereichen wenige FE abgebildet werden. Dadurch wird das Bauteil nicht gleichmäßig abgebildet und es wird auch weniger Information über die originalen Daten beibehalten. Die Voxel-Diskretisierung zeigt diesen Nachteil nicht, da die FE gleichmäßig abgebildet werden, keine Verzerrungen auftreten und damit ein höherer Informationsgehalt vorliegt.

Hinsichtlich der Anforderung einer möglichst intuitiven Interpretierbarkeit der Diskretisierungs-Verfahren stellt sich die Interpretierbarkeit der Ergebnisse der Sphären-Diskretisierung als schwierig dar. Dadurch, dass die dreidimensionalen Geometrien der Bauteile auf ein zweidimensionales Bild reduziert werden, fällt es dem Betrachter schwer, darin das ursprüngliche Bauteil wiederzuerkennen. Darüber hinaus ist der Einfluss des Diskretisierungsparameters K_S kaum abschätzbar. Zusätzlich müssen für unterschiedlich große Bauteile unterschiedliche Werte für K_S gewählt werden. Diese Nachteile bzw. Herausforderungen zeigt die Voxel-Diskretisierung nicht. Die dreidimensionale Struktur der zugrunde liegenden FE-Daten bleibt hierbei erhalten und ermöglicht dem Anwender so auch weiterhin das Bauteil wiederzuerkennen. Dadurch, dass die Kantenlänge K_V eine intuitive Einschätzung darüber liefert, wie viele FE in etwa in einem Voxel landen, kann der Einfluss durch die Diskretierierung auf den abgebildeten Informationsgehalt abgeschätzt werden. Des Weiteren kann K_V unabhängig von der Bauteilgröße gewählt werden. Hinsichtlich des Speicherbedarfs zeigt sich die Voxel-Diskretisierung ebenfalls als vorteilhafter gegenüber dem Sphären-Verfahren. Dadurch, dass die Informationen des zugrunde liegenden Crashverhaltens effizienter abgebildet werden, sind weniger diskretisierte Elemente (Voxel) und damit ein geringerer Speicherbedarf notwendig.

Aus diesen Gründen wird das Voxel-Verfahren als Diskretisierungsmethode empfohlen. Als Ergebnis liegen die Daten einheitlich in einem strukturierten Format vor und ermöglichen die weitere maschinelle Weiterverarbeitung.

Die daraufhin folgende Ausreißerdetektion hat das Ziel, auffälliges Crashverhalten zu detektieren und es dem Anwender anschaulich im Post-Prozessor darzustellen. In dieser Arbeit wurde eine diesen Anforderungen entsprechende neue Methode entworfen. Das Crashverhalten einzelner Bauteile wird dabei für jeden Zeitschritt getrennt auf Auffälligkeiten hin analysiert. Die Betrachtung des kontinuierlichen Ausreißerkennwertes ermöglicht dem Anwender eine Einschätzung darüber, wie auffällig die einzelnen Bereiche des Fahrzeugs sind. Damit die Ausreißerdetektion im manuellen Analyseteil ausgeführt werden kann, wurde bei deren Ausgestaltung auf eine geringe Berechnungszeit geachtet. In der Arbeit wurden deshalb sämtliche Algorithmen aus *Pyod* (Zhao et al. 2019) untersucht. Vier Algorithmen (PCAD, KNDD, LOF, SOS) eignen sich dabei besonders für die beschriebene Aufgabenstellung und werden im Detail betrachtet. Es wird verglichen, welcher Algorithmus die geringste Abhängigkeit gegenüber einem steigenden Ausreißeranteil in den Daten aufweist, wie sich vorhandene Hyperparameter auf die

Ergebnisse auswirken und sich daraus sinnvolle Wertebereiche ableiten lassen als auch, wie sich die Diskretisierung auf die Ausreißerkennwerte auswirkt.

Die Analysen zeigen, dass die PCA zwar die geringste Berechnungszeit und keine Hyperparameter aufweist, was sich grundsätzlich vorteilhaft in der praktischen Anwendung darstellt. Deren Ausreißerkennwerte werden jedoch durch andere Algorithmen übertroffen. Für SOS kann kein sinnvoller Wertebereich für dessen Hyperparameter abgeschätzt werden, aus dem anschließend ein Ensemble Modell für robustere Ergebnisse abgeleitet werden könnte. Die Ausreißerkennwerte sind durchweg gering, weshalb dieses Verfahren nicht empfohlen wird. Für den KNND Algorithmus lässt sich zwar der Hyperparametereinfluss abschätzen und ein Wertebereich ableiten, die resultierenden Ausreißerkennwerte werden jedoch durch den LOF-Algorithmus noch übertroffen. Bei letzterem handelt es sich um einen dichtebasierten Algorithmus, der durch diese Eigenschaft besser dazu in der Lage ist, die Simulationen mit auffälligem Crashverhalten zu detektieren. Er zeigt gleichzeitig die geringste Abhängigkeit gegenüber einem steigenden Ausreißeranteil und die höchsten Ausreißerkennwerte auch bei groben Diskretisierungen für das Voxel-Verfahren. Aus diesem Grund wird die mit dieser Arbeit vorgestellte Ausreißerdetektion mit dem LOF-Algorithmus empfohlen. Um die Ausreißerkennwerte aber nicht nur von einer Berechnung des LOF mit einem konkreten Hyperparameter abhängig zu machen, wird zudem ein Wertebereich identifiziert innerhalb dessen sich hohe Ausreißerkennwerte für auffällige Simulationen erzielen lassen. Dieser wird dazu verwendet, die Berechnungen der Ausreißerkennwerte mit dementsprechenden Hyperparametern zu wiederholen und in einem Ensemble-Modell zu kombinieren. Die einzige Information, die der Anwender hierzu bereitstellen muss, ist ein angenommener Anteil an Ausreißern in den vorliegenden Daten (beispielsweise 10 %).

Dadurch, dass die Ausreißerdetektion automatisiert berechnet werden kann, ist es möglich sämtliche Informationen aus der Simulation auszuwerten, wodurch sich eine insgesamt vollständigere Auswertung ergibt. Bei auffälligem Crashverhalten wird der/die Ingenieur*in gezielt durch die farbliche Hervorhebung der betroffenen Bauteile und Zeitschritte im Post-Prozessor gezielt gewarnt.

Mittels des Crash-Verhalten-Detektors soll ein vom Anwender vorgegebenes Crashverhalten in neuen Simulationen wiedererkannt werden. Bei erneutem Auftreten wird das entsprechende Bauteil im Post-Prozessor farblich hervorgehoben. Die Anforderungen an die neue Methode sind zum einen, dass möglichst wenig kategorisierte Daten hierfür bereitgestellt werden müssen sowie zum anderen, dass das markierte Crashverhalten auch bei einer gröberen Diskretisierung durch größere KV zuverlässig wiedererkannt werden soll.

Die in dieser Arbeit hierzu vorgestellte neue Methode basiert auf SimSiam, einem neuronalen Netz, das dem Bereich des selbstüberwachten Lernens zuzuordnen ist. Aus unkategorisierten Daten werden überwachte Lernaufgaben erzeugt, wodurch zunächst alle relevanten Merkmale extrahiert werden können. Diese werden im Anschluss daran dazu verwendet, einen Klassifikator mit den wenigen kategorisierten Daten des Anwenders zu trainieren. In dieser Arbeit wird exemplarisch die binäre Klassifikation am Beispiel eines *Längsträgers* mit zwei Deformationsmodi (Faltet Vorne und Faltet Hinten) untersucht. Die neue auf *SimSiam* basierende Methode wird mit dem Ausgangsmodell des klassischen maschinellen Lernens bestehend aus einer PCA und einem Klassifikator verglichen. Dabei wird insbesondere das

Konvergenzverhalten anhand der Anzahl der für das Training verwendeten kategorisierten Daten bewertet. Das Ziel ist, mit möglichst wenig kategorisierten Daten eine möglichst hohe Klassifikationsgenauigkeit zu erreichen. Für SimSiam ist die Wahl der Modifikationen des originalen Bildes entscheidend. Aus diesem Grund wurden auch diese im Detail untersucht und das Modell durch eine sukzessive Verbesserung der Parameter stetig verbessert.

Die Ergebnisse von SimSiam übertreffen die des Vergleichs-Ansatzes bestehend aus einer PCA und einem Machine Learning Klassifikator deutlich, da sogar der optimale Fall, bei dem nur zwei kategorisierte Dateninstanzen für eine erfolgreiche Klassifikation der Testdaten (Genauigkeit 100%) zur Verfügung gestellt werden müssen, erreicht wird. Bei der Analyse der Modifikationen stellt sich heraus, dass bei kleinen Datensets schwächere Modifikationen gewählt werden sollten, während es bei größeren Datensets besser ist, die Stärke der Modifikation zu erhöhen. Im Allgemeinen ist es nicht möglich ein allgemeingültiges ML-Modell für sämtliche Anwendungsfälle zu erstellen. Dieses muss stets an die zugrunde liegende Aufgabenstellung angepasst werden. Der in dieser Arbeit aufgezeigte Ansatz zur schrittweisen Verbesserung des Modells kann dabei auch für neue Aufgabenstellungen verwendet werden, um die Klassifikationsgenauigkeit des CVD zu erhöhen.

Mittels der Implementierung des CVD ist es dem Anwender künftig möglich, ein von ihm vorgegebenes Crashverhalten in neuen Simulationen automatisiert wiederzuerkennen. Dies hilft dabei, dass keine relevanten Effekt in der Auswertung übersehen werden. Die Kombination aus SimSiam und dem Klassifikator ermöglicht dem Anwender nur wenige kategorisierte Daten zur Verfügung stellen zu müssen und verringert dadurch den Aufwand für die Anwendung dieser neuen Methode.

In dieser Arbeit werden des Weiteren unterschiedliche Algorithmen zur Dimensionsreduktion miteinander verglichen. Es wird untersucht, inwiefern nichtlineare Verfahren gegenüber der linearen PCA Vorteile insbesondere im Hinblick auf die aus der Voxel-Diskretisierung resultierende Datenrepräsentation bieten. Darüber hinaus wird betrachtet, wie das zeitliche Verhalten während des Crashs in der Dimensionsreduktion berücksichtigt werden kann. Der Einfluss der Hyperparameter wird sowohl subjektiv als auch qualitativ mittels eines Qualitätskriteriums bewertet. Darüber hinaus wird der jeweils erforderliche zeitliche Berechnungsaufwand der Algorithmen untersucht.

Aus den Analysen resultiert, dass beide Verfahren zur Berücksichtigung des zeitlichen Verhaltens sinnvoll sind. *OPioS* stellt dabei jeweils eine Simulation als einen Punkt dar, was den Vorteil bietet, dass sich eine übersichtliche Darstellung ergibt. Nachteilig stellt sich heraus, dass das zeitliche Verhalten und mögliche Bifurkationspunkte damit nicht visualisiert werden können. *OLioS* ist dagegen dazu in der Lage den zeitlichen Verlauf des Crashs und der Bifurkationspunkte zu visualisieren. Es erweist sich dabei als sehr vorteilhaft, wenn die Einfärbung der Linien gemäß der drei Hauptkomponenten aus den *OLioS* Resultaten als RGB Farbwerte übernommen werden.

Der Vergleich mit nichtlinearen Algorithmen zeigt einige Nachteile gegenüber der linearen PCA auf. Zum einen verfügen nichtlinearen Algorithmen über Hyperparameter, was wiederrum bedeutet, dass der Anwender verschiedene Werte und deren Auswirkungen auf die Resultate ausprobieren muss, um eine für ihn sinnvolle niedrigdimensionale Darstellung zu erhalten. Darüber hinaus benötigen sie ein Vielfaches an Berechnungszeit im Vergleich zur PCA. Dies

wirkt sich insbesondere bei t-SNE und UMAP nachteilig aus, da es sich bei diesen zusätzlich um stochastische Algorithmen handelt. Dies bedeutet, dass sich die Ergebnisse selbst bei mehrmaliger Berechnung mit denselben Hyperparametern unterscheiden. Aus diesem Grund müssen stets mehrere Wiederholungen berechnet und mittels des Qualitätskriteriums bewertet werden. Dies erlaubt anschließend die automatisierte Auswahl der besten Einbettung hinsichtlich deren höchsten Qualität. UMAP weist daneben den weiteren Nachteil auf, dass die Definition zweier Hyperparameter erforderlich ist und sich damit die Anwendbarkeit für den/die Ingenieur*in aufwändiger gestaltet. Zusätzlich sind die resultierenden Einbettungen für den Anwender weniger nützlich verglichen mit denen des t-SNE-Algorithmus. Letztere zeigen insbesondere für kleine Werte des Hyperparameters Perplexität Vorteile bei grob diskretisierten Daten mit größeren Werten für K_V. Hier wird das unterschiedliche Crashverhalten im Vergliech zur PCA deutlicher voneinander getrennt dargestellt. Insgesamt wird für die praktische Anwendung der Dimensionsreduktion die PCA als Algorithmus der Wahl empfohlen. Es müssen keine Hyperparameter definiert werden und die Ergebnisse sind deterministisch. Darüber hinaus erlauben sie eine intuitive Einschätzung über die Verteilung des vorliegenden Crashverhaltens, indem ähnliche Simulationen in ähnlichen Bereichen dargestellt werden. T-SNE kann mit kleinen Werten der Perplexität zusätzlich herangezogen werden, um lokale Eigenschaften in den Daten noch besser darzustellen.

Durch die Dimensionsreduktion erhält der Anwender einen schnellen Überblick über das in einer Vielzahl an Simulationen auftretende Crashverhalten. Er kann so zudem schnell ähnliche Simulationen aus der Historie auffinden. Die Verknüpfung mit Metadaten aus der Simulation kann dem/der Ingenieur*in dabei helfen neue konstruktive Verbesserungen vorzunehmen, um das Crashverhalten des Fahrzeugs kontinuierlich zu verbessern.

10.2 Ausblick

Die Voxel-Diskretisierung wird in dieser Arbeit verwendet, um die Simulationen mit unterschiedlichen FE-Netzen in ein einheitliches strukturiertes Format zu überführen, sodass die weitere maschinelle Weiterverarbeitung gewährleistet ist. Das Verfahren sollte auf weiteren Daten mit unterschiedlichen FE-Netzen getestet werden. Darüber hinaus sollte ein Vergleich der Resultate, mit denen von Mapping-Verfahren durchgeführt werden. In dieser Arbeit wird als Anforderung definiert, dass die neue Datenrepräsentation unabhängig von anderen Simulationen sein sollte, sodass dieses Verfahren aus den Betrachtungen ausgeschlossen ist. Sollte diese Anforderung in einer anderen Anwendung jedoch nicht gestellt werden, kann es unter Umständen attraktiv sein, Mapping-Verfahren zu betrachten. Insbesondere der Informationsgehalt der neuen Datenrepräsentation und die Berechnungszeit sollten mit dem hier vorgestellten Ansatz verglichen werden, da diese beiden Kriterien ausschlaggebend über das zu verwendende Verfahren sind.

Die Ausreißerdetektion wurde mittels eines Datensets vorgestellt, in dem mehrere Ausreißer auftreten, die wiederum ein ähnliches Crashverhalten zeigen. In der Praxis treten jedoch auch Situationen auf, in denen zwar ebenfalls mehrere Ausreißer auftreten, die sich jedoch auch untereinander deutlich in ihrem Crashverhalten unterscheiden. Es sollte demnach überprüft

werden, inwiefern die hier vorgestellte Ausreißerdetektion dazu in der Lage ist, diese unterschiedlichen Ausreißer zu identifizieren. Eine mögliche Erweiterung für das Verfahren ist, verschiedene Auswertungsgrößen in der Ausreißerdetektion zu berücksichtigen. Dabei sollte zunächst untersucht werden, welche sich hierfür besonders eignen. In der mit dieser Arbeit vorgestellten Methode wird exemplarisch die plastische Dehnung verwendet. Das Crashverhalten eines Bauteils kann sich jedoch auch hinsichtlich seiner Kinematik deutlich von anderen Simulationen unterscheiden. Hierfür sollten dann beispielsweise die Verschiebungen anstelle der plastischen Dehnung ausgewertet werden. Es stellt sich dabei die Frage, ob für jede Auswertungsgröße der Simulation ein separater Ausreißerkennwert gemäß der hier vorgestellten Methode berechnet werden sollte oder auch mehrere Auswertungsgrößen gemeinsam ausgewertet werden und einen resultierenden Ausreißerkennwert liefern sollten. In dieser Arbeit wird das Crashverhalten eines gesamten Bauteils basierend auf dessen FE-Netz als Auswertungsgröße ausgewertet. Eine Erweiterung der Ausreißerdetektion würde beispielsweise die Detektion von auffälligem Crashverhalten in Kurven-Verläufen darstellen. An vorab definierten Positionen im Fahrzeug werden beispielsweise Beschleunigungen durch Beschleunigungssensoren oder Kräfte mittels Cross-Sections ausgewertet. Auch hier kann in einzelnen Simulationen ein auffälliges Crashverhalten auftreten, das künftig mittels der Ausreißerdetektion automatisiert ausgewertet werden könnte.

Der in dieser Arbeit vorgestellte Crash-Verhalten-Detektor wird am Beispiel der binären Klassifikation (zwei Crashverhalten werden unterschieden) entwickelt und getestet. Eine Erweiterung stellt die Klassifikation basierend auf mehr als zwei Kategorien des Crashverhaltens dar. Für die erfolgreiche Integration des CVD in die bisherige Auswertung ist es entscheidend, dass der Anwender wenig bis keine Zeit in die Optimierung der Architektur des dahinterstehenden neuronalen Netzes investieren muss. Dementsprechend sollte ein möglichst gut zu verallgemeinerndes Modell verwendet werden. Daher sollte der CVD für verschiedene Bauteile mit unterschiedlichem Crashverhalten untersucht und weiterentwickelt werden. Eine Möglichkeit, die Verallgemeinerungsfähigkeit des CVD zu erhöhen, stellt dabei beispielsweise die Verwendung weiterer Modifikations-Techniken dar. In dieser Arbeit wird bei den Modifikationen hingenommen, dass die resultierenden Bilder ein unphysikalisches Crashverhalten repräsentieren können. Stattdessen liegt der Fokus darauf, dass nach den Modifikationen nach wie vor die für das Crashverhalten charakteristischen Informationen enthalten sind. Künftig sollte überprüft werden, ob sich durch Modifikationen, die nur physikalisch realistisches Verhalten erzeugen, eine Verbesserung der Ergebnisgüte erzielen lässt.

Für den hier entworfenen CVD ist es notwendig, dass die dreidimensionalen Voxel-Daten zunächst in Bilder transformiert werden. Dabei bleiben zwar die Zusammenhänge des zeitlichen Verlaufs für die einzelnen Voxel erhalten, jedoch nicht die geometrischen Informationen, wo sie sich im dreidimensionalen Raum befinden. Eine Erweiterung des CVD stellt daher die direkte Verarbeitung der 3D Voxel-Daten in *SimSiam* dar. Es sollte untersucht werden, ob die Information über die relative geometrische Position der einzelnen Voxel dabei hilft, das Crashverhalten zuverlässiger zu detektieren.

In der hier vorgestellten Methode wird einzig die plastische Dehnung verwendet, um ein Crashverhalten zu klassifizieren. Insbesondere, wenn der Anwender ein Bauteil für den CVD markiert, welches sich in seinem Kinematikverhalten deutlich von den anderen Simulationen

unterscheidet, müssten andere Auswertungsgrößen verwendet werden. Um robust in neuen Simulationen auch das unterschiedlichste Crashverhalten wiedererkennen zu können, sollte eine Erweiterung des CVD in Form einer gleichzeitigen Verwendung mehrerer Auswertungsgrößen für die Klassifikation eines Crashverhaltens vorgenommen werden.

11 Literaturverzeichnis

Abbasloo, A.; Garcke, J.; Iza-Teran, R. (2019): Unsupervised Learning of Automotive 3D Crash Simulations using LSTMs. Online verfügbar unter https://openreview.net/pdf?id=BklekANtwr.

Aggarwal, C. C. (2017): Outlier analysis: Springer.

Alaız, C. M.; Fernández, A.; Dorronsoro, J. R. (2015): Diffusion Maps Parameters Selection Based on Neighbourhood Preservation. In: *Computational Intelligence 6*.

Amer, M.; Goldstein, M.; Abdennadher, S. (2013): Enhancing one-class support vector machines for unsupervised anomaly detection. In: *Proceedings of the ACM SIGKDD workshop on outlier detection and description*, 8-15.

Andricevic, N. (2016): Robustheitsbewertung crashbelasteter Fahrzeugstrukturen. Dissertation, Albert-Ludwigs-Universität Freiburg.

Balasubramanian, M.; Schwartz, E. L. (2002): The isomap algorithm and topological stability. In: *Science* 295 (5552), S. 7.

Bellman, R. (1961): Adaptive Control Processes: a Guided Tour. In: *Princeton University Press*.

Bickel, S.; Spruegel, T. C.; Schleich, B.; Wartzack, S. (2019): How do Digital Engineering and included AI based assistance tools change the product development process and the involved engineers. In: *Proceedings of the Design Society: International Conference on Engineering Design*, S. 2567–2576.

Bisong, E. (2019): Building machine learning and deep learning models on Google cloud platform: A comprehensive guide for beginners: Apress.

Bohn, B.; Garcke, J.; Griebel, M. (2016): A sparse grid based method for generative dimensionality reduction of high-dimensional data. In: *Journal of Computational Physics* 309, S. 1–17.

Bohn, B.; Garcke, J.; Iza-Teran, R.; Paprotny, A.; Peherstorfer, B.; Schepsmeier, U.; Thole, C. A. (2013): Analysis of Car Crash Simulation Data with Nonlinear Machine Learning Methods. In: *Procedia Computer Science* 18, S. 621–630.

Borg, I.; Groenen, P. J. F. (2005): Modern multidimensional scaling: Theory and applications: Springer Science & Business Media.

Borsotto, B.; Strickstrock, R.; Thole C. A. (2015): Robustness analysis – Significant reduction of scatter occurrence. Online verfügbar unter https://www.sidact.de/fileadmin/user_upload/Presentation_Robustness_NAFEMS.pdf.

Breimann, L.; Friedman, J. H.; Olshen, R. A.; Stone, C. J. (1984): Classification and regression trees. In: *Routledge*.

Breunig, M. M.; Kriegel, H. P.; Ng, R. T.; Sander, J. (2000): LOF: identifying density-based local outliers. In: *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, S. 93–104.

Bromley, J.; Guyon, I.; LeCun, Y.; Säckinger, E.; Shah, R. (1993): Signature verification using a" siamese" time delay neural network. In: *Advances in neural information processing systems* 6.

Brown, R.; Bloomfield, M.; Thole, C. A.; Nikitina, L. (2012): Analysis of the Scatter of a Deploying Airbag. In: *Proceedings 12th International LS-DYNA Users Conference, Detroit*.

Büttner, J. (2022): Effiziente Lösungsansätze zur Reduktion des numerischen Ressourcenbedarfs für den operativen Einsatz der Multidisziplinären Optimierung von Fahrzeugstrukturen. Dissertation, Fakultät für Maschinenbau und Sicherheitstechnik, Universität Wuppertal.

carhs GmbH (2012): SAFETY COMPANION 2012.

Chen, X.; He, K. (2021): Exploring simple siamese representation learning. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, S. 15750–15758.

Deng, L. (2012): The mnist database of handwritten digit images for machine learning research. In: *IEEE signal processing magazine* 29 (6), S. 141–142.

Diez, C. (2019): Process for Extraction of Knowledge from Crash Simulations by means of Dimensionality Reduction and Rule Mining. Dissertation, Fakultät für Maschinenbau und Sicherheitstechnik, Universität Wuppertal.

Diez, C.; Harzheim, L.; Schumacher, A. (2016): Effiziente Wissensgenerierung zur Robustheitsuntersuchung von Fahrzeugstrukturen mittels Modellreduktion und Ähnlichkeitsanalyse. In: *VDI-Berichte* 2279.

Estrach, J. B.; Zaremba, W.; Szlam, A.; LeCun, Y. (2014): Spectral networks and deep locally connected networks on graphs. In: *2nd International Conference on Learning Representations, ICLR*.

Feuersänger, C.; Griebel, M. (2009): Principal manifold learning by sparse grids. In: *Computing* 85, S. 267–299.

Gao, J.; Tan, P. N. (2006): Converting output scores from outlier detection algorithms into probability estimates. In: *Sixth International Conference on Data Mining (ICDM'06)*, S. 212–221.

Garcke, J.; Iza-Teran, R. (2015): Machine learning approaches for repositories of numerical simulation results. In: *Proceedings 10th European LS-DYNA Conference*.

Garcke, J.; Iza-Teran, R. (2017): Machine Learning Approaches for Data from Car Crashes and Numerical Car Crash Simulations. In: *Proceedings of the NAFEMS World Congress*.

Garcke, J.; Iza-Teran, R.; Prabakaran, N. (2016): Datenanalysemethoden zur Auswertung von Simulationsergebnissen im Crash und deren Abgleich mit dem Experiment. In: *VDI-Tagung SIMVEC*.

Garcke, J.; Pathare, M.; Prabakaran, N. (2017): ModelCompare. In: *Scientific Computing and Algorithms in Industrial Simulations*, S. 199–205.

Gracia, A.; González, S.; Robles, V.; Menasalvas, E. (2014): A methodology to compare dimensionality reduction algorithms in terms of loss of quality. In: *Information Sciences* 270, S. 1–27.

Graves, A.; Schmidhuber, J. (2005): Framewise phoneme classification with bidirectional LSTM networks. In: *Proceedings in IEEE International Joint Conference on Neural Networks*, S. 2047–2052.

Grill, J. B.; Strub, F.; Altché, F.; Tallec, C.; Richemond, P.; Buchatskaya, E. et al. (2020): Bootstrap your own latent-a new approach to self-supervised learning. In: *Advances in neural information processing systems* 33, S. 21271–21284.

Griparis, A.; Faur, D.; Datcu, M. (2016): A dimensionality reduction approach to support visual data mining: Co-ranking-based evaluation. In: *2016 International Conference on Communications*, 391-394.

Hawkins, D. M. (1980): Identification of outliers: Springer.

Huntington, D. E.; Lyrintzis, C. S. (1998): Improvements to and limitations of Latin hypercube sampling. In: *Probabilistic engineering mechanics* 13 (4), S. 245–253.

Inselberg, A.; Dimsdale, B. (1990): Parallel coordinates: a tool for visualizing multi-dimensional geometry. In: *Proceedings of the First IEEE Conference on Visualization*, S. 361–378.

Iza-Teran, R. (2014): Enabling the analysis of finite element simulation bundles. In: *International Journal for Uncertainty Quantification* 4 (2), S. 95–110.

Iza-Teran, R.; Garcke, J. (2014): Data analytics for simulation repositories in industry. In: *Tagungsband der Informatik, Stuttgart*.

Iza-Teran, R.; Garcke, J. (2019): A Geometrical Method for Low-Dimensional Representations of Simulations. In: *SIAM/ASA Journal on Uncertainty Quantification* 7 (2), S. 472–496.

Janssens, Jeroen H. M. (2013): Outlier selection and one-class classification. Dissertation, Tilburg University.

Karhunen, K. (1946): Zur spektraltheorie stochastischer prozesse. In: *Ann. Acad. Sci. Fennicae, AI, 34*.

Kipf, T. N.; Welling, M. (2016): Semi-supervised classification with graph convolutional networks. In: *arXiv preprint arXiv:1609.02907*.

Komarek, P. (2004): Logistic regression for data mining and high-dimensional classification: Carnegie Mellon University.

Kotsiantis, S. B.; Zaharakis, I.; Pintelas, P. (2007): Supervised Machine Learning: A Review of Classification Techniques. In: *Emerging artificial intelligence applications in computer engineering* 160 (1), S. 3–24.

Kriegel, H. P.; Kröger, P.; Schubert, E.; Zimek, A. (2009): Outlier detection in axis-parallel subspaces of high dimensional data. In: *Pacific-asia conference on knowledge discovery and data mining*, S. 831–838.

Krizhevsky, A.; Sutskever, I.; Hinton, G. E. (2012): Imagenet classification with deep convolutional neural networks. In: *Advances in neural information processing systems* 25.

LASSO GmbH (2022): Python Bibliothek, um d3plot Dateien aus LS-DYNA Simulationen zu analysieren. Online verfügbar unter https://lasso-gmbh.github.io/lasso-python/build/html/index.html.

LeCun, Y.; Boser, B.; Denker, J. S.; Henderson, D.; Howard, R. E.; Hubbard, W.; Jackel, L. D. (1989): Backpropagation applied to handwritten zip code recognition. In: *Neural computation* 1 (4), S. 541–551.

LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. (1998): Gradient-based learning applied to document recognition. In: *Proceedings of the IEEE* 86 (11), S. 2278–2324.

LeCun, Y.; Cortes C.; Burges C. (2022): MNIST handwritten digit database. Online verfügbar unter http://yann.lecun.com/exdb/mnist/.

Lee, J. A.; Verleysen, M. (2007): Nonlinear dimensionality reduction. New York: Springer.

Lee, J. A.; Verleysen, M. (2008a): Quality assessment of nonlinear dimensionality reduction based on K-ary neighborhoods. In: *Proceedings of the Workshop on New Challenges for Feature Selection in Data Mining and Knowledge Discovery at ECML/PKDD*, S. 21–35.

Lee, J. A.; Verleysen, M. (2008b): Rank-based quality assessment of nonlinear dimensionality reduction. In: *ESANN*, S. 49–54.

Lee, J. A.; Verleysen, M. (2010): Scale-independent quality criteria for dimensionality reduction. In: *Pattern Recognition Letters* 31 (14), S. 2248–2257.

Ley, C.; Klein, O.; Bernard, P.; Licata, L. (2013): Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median. In: *Journal of experimental social psychology* 49 (4), S. 764–766.

Liu, G. R.; Quek, S. S. (2013): The finite element method: a practical course: Butterworth-Heinemann.

McInnes, L.; Healy, J.; Melville, J. (2018): Umap: Uniform manifold approximation and projection for dimension reduction. In: *arXiv preprint arXiv:1802.03426*.

Meng, D.; Leung, Y.; Xu, Z. (2011): A new quality assessment criterion for nonlinear dimensionality reduction. In: *Neurocomputing* 74 (6), S. 941–948.

Mertler, Stefan; Muller, Stefan P.; Thole, Clemens-August (2015): Predictive Principal Component Analysis as a Data Compression Core in a Simulation Data Management System, S. 173–182. DOI: 10.1109/DCC.2015.50.

Miller, Jeff (1991): Reaction time analysis with outlier exclusion: Bias varies with sample size. In: *The quarterly journal of experimental psychology* 43 (4), S. 907–912.

Mu, X.; Ting, K. Ming; Zhou, Z. H. (2017): Classification under streaming emerging new classes: A solution using completely-random trees. In: *IEEE Transactions on Knowledge and Data Engineering* 29 (8), S. 1605–1618.

Pearson, K. (1901): LIII. On lines and planes of closest fit to systems of points in space. In: *The London, Edinburgh, and Dublin philosophical magazine and journal of science* 2 (11), S. 559–572.

Qi, C. R.; Su, H.; Mo, K.; Guibas, L. J. (2017a): Pointnet: Deep learning on point sets for 3d classification and segmentation. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, S. 652–660.

Qi, C. R.; Su H.; Nießner M.; Dai A.; Yan M.; Guibas L. J. (2016): Volumetric and Multi-view CNNs for Object Classification on 3d Data. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, S. 5648–5656.

Qi, C. R.; Yi, L.; Su, H.; Guibas, L. J. (2017b): Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In: *Advances in neural information processing systems* 30.

Ramaswamy, S.; Rastogi, R.; Shim, K. (2000): Efficient algorithms for mining outliers from large data sets. In: *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, S. 427–438.

Raschka, S.; Mirjalili, V. (2019): Python machine learning: Machine learning and deep learning with Python, scikit-learn, and TensorFlow 2: Packt Publishing Ltd.

Roth, S.; Chamoret, D.; Badin, J.; Imbert, J. R.; Gomes, S. (2011): Crash FE Simulation in the Design Process-Theory and Application: Numerical Analysis-Theory and Application. IntechOpen.

Schumacher, A. (2020): Optimierung mechanischer Strukturen. Grundlagen und industrielle Anwendungen. 3. Aufl. 2020. Berlin, Heidelberg: Springer.

Seeger, M. (2004): Gaussian processes for machine learning. In: *International journal of neural systems* 14 (02), S. 69–106.

Sible, S. (2020a): Spectral Coefficient Analysis on Geometerical Deformation using Laplace-Beltrami Operator. Dissertation, The Ohio State University.

Sible, S.; Iza-Teran, R.; Garcke, J.; Aulig, N.; Wollstadt, P. (2020b): A Compact Spectral Descriptor for Shape Deformations. Online verfügbar unter http://arxiv.org/pdf/2003.08758v1.

SIDACT GmbH (2021): Diff-Crash, Stability Analysis for Simulation Results. Online verfügbar unter https://www.sidact.de/diffcrash/.

Solanki, K.; Oglesby, D. L.; Burton, C. L.; Fang, H.; Horstemeyer, M. F. (2004): Crashworthiness simulations comparing PAM-CRASH and LS-DYNA. In: *SAE Technical Paper (2004-01-1174), Detroit*.

Spruegel, T.; Rothfelder, R.; Bickel, S.; Grauf, A.; Sauer, C.; Schleich, B.; Wartzack, S. (2018): Methodology for plausibility checking of structural mechanics simulations using deep learning on existing simulation data. In: *Proceedings of NordDesign 2018*.

Spruegel, T.; Schröppel, T.; Wartzack, S. (2017): Generic approach to plausibility checks for structural mechanics with deep learning. In: *Proceedings of the 21st International Conference on Engineering Design*.

Spruegel, T. C.; Bickel, S.; Schleich, B.; Wartzack, S. (2021): Approach and application to transfer heterogeneous simulation data from finite element analysis to neural networks. In: *Journal of Computational Design and Engineering* 8 (1), S. 298–315.

Spruegel, T. C.; Wartzack, S. (2014): Konzept zur automatischen Bauteilerkennung innerhalb der FE-SoftwareUmgebung mittels Künstlichen Neuronalen Netzen. In: *Proceedings of the 24th Symposium Design for X, Bamburg*, S. 47–56.

Spruegel, T. C.; Wartzack, S. (2015): Concept and application of automatic part-recognition with artificial neural networks for FE simulations. In: *Proceedings of the 20th International Conference on Engineering Design (ICED 15) Vol 6: Design Methods and Tools-Part 2 Milan, Italy*, S. 183–194.

Steffes-lai, D.; Clees, T. (2011): Statistical Analysis of Process Chains: Novel PRO-CHAIN Components. In: *Proceedings 8th LS-DYNA European Users Conference*.

Steffes-lai, D.; Pathare, M.; Garcke, J. (2021): Towards a Framework for Automatic Event Detection. In: *NAFEMS World Congress 2021*.

Su, H.; Maji, S.; Kalogerakis, E.; Learned-Miller, E. (2015): Multi-view Convolutional Neural Networks for 3D Shape Recognition. In: *Proceedings of the IEEE international conference on computer vision*, S. 945–953.

Suthaharan, S. (2016): Machine Learning Models and Algorithms for Big Data Classification. Boston, MA: Springer US (36).

Tenenbaum, J. B.; v. d. Silva; Langford, J. C. (2000): A global geometric framework for nonlinear dimensionality reduction. In: *Science* 290 (5500), S. 2319–2323.

Thole, C. A.; Nikitina, L.; Nikitin I.; Clees, T. (2010): Advanced mode analysis for crash simulation results. In: *Proceedings 9th LS-DYNA Users Conference, Bamberg*, 2010.

van der Maaten, L.; Hinton, G. (2008): Visualizing data using t-SNE. In: *Journal of machine learning research* 9 (11).

van der Maaten, L.; Postma, E.; van den Herik, J. (2009): Dimensionality reduction: a comparative Review. In: *Journal of Machine Learning Research - JMLR* 10 (66-71), S. 13.

Vapnik, V.; Chervonenkis, A. (1974): Theory of pattern recognition. In: *Nauka*.

Verleysen, M.; Lee, J. A. (2013): Nonlinear dimensionality reduction for visualization. In: *International Conference on Neural Information Processing*, S. 617–622.

Virtanen, P.; Gommers, R.; Oliphant, T. E.; Haberland, M.; Reddy, T.; Cournapeau, D. et al. (2020): SciPy 1.0: fundamental algorithms for scientific computing in Python. In: *Nature methods* 17 (3), S. 261–272.

Wang, C.; Cheng, M.; Sohel, F.; Bennamoun, M.; Li, J. (2019): NormalNet: A voxel-based CNN for 3D object classification and retrieval. In: *Neurocomputing* 323, S. 139–147.

Webseite ModelCompare. Online verfügbar unter https://www.scai.fraunhofer.de/de/geschaeftsfelder/numerische-datenbasierte-vorhersage/produkte/modelcompare.html.

Zhang, P.; Ren, Y.; Zhang, B. (2012): A new embedding quality assessment method for manifold learning. In: *Neurocomputing* 97, S. 251–266.

Zhao, Y.; Nasrullah, Z.; Li, Z. (2019): A python toolbox for scalable outlier detection. In: *Journal of Machine Learning Research - JMLR* (20), Artikel 96.

(Abbasloo et al. 2019)

Vorveröffentlichungen im Rahmen dieser Dissertation:

Kracker, D.; Garcke J.; Schumacher, A. (2020): Automatic analysis of crash simulations with dimensionality reduction algorithms such as PCA and t-SNE. In: Proceedings 16th International LS-DYNA Users Conference.

Kracker, D.; Dhanasekaran, R. K.; Schumacher, A.; Garcke, J. (2023): Method for automated detection of outliers in crash simulations. In: International Journal of Crashworthiness, 28(1), 96-107, 2023

Vom Autor betreute Masterarbeiten:

Azouni, A. [2019]. Evaluation of Crash Simulations using Machine Learning. Masterarbeit, Department of Electrical and Computer Engineering, Technische Universität München

Filker, D. [2021]. *Vergleich von Methoden des maschinellen Lernens für die Auswertung von Crash Simulationen.* Masterarbeit, Institut für Statik und Dynamik der Luft- und Raumfahrtkonstruktionen, Universität Stuttgart

Dhanasekaran, R. [2021]. Learning to detect critical crash behaviours from few samples using Few Shot and Self-Supervised Learning. Masterarbeit, Institüt für Data Science im Maschinenbau, RWTH Aachen University

12 Anhang A: Bauteile

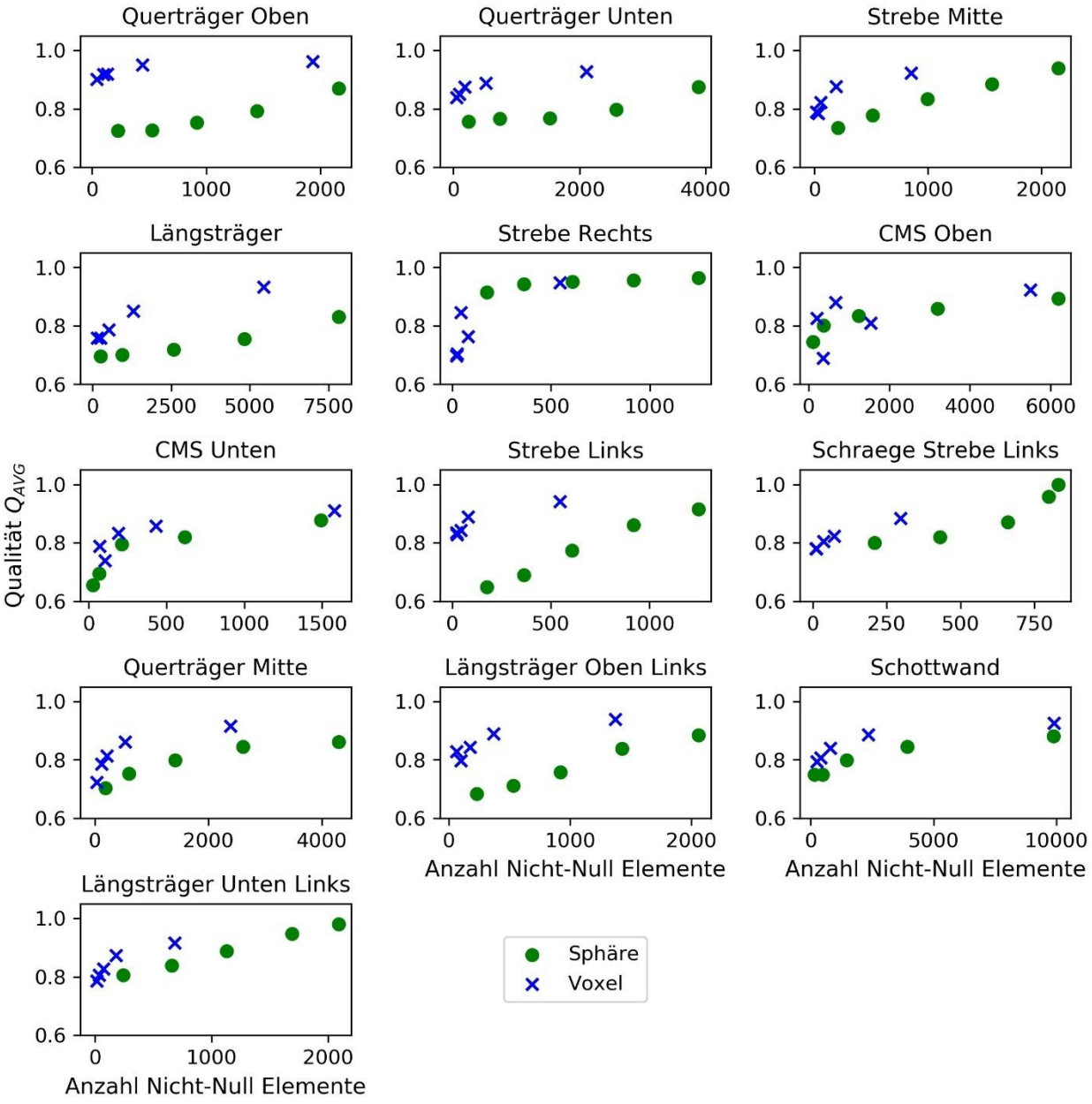


Abbildung 97: Darstellung des Qualitätskriteriums Q_{AVG} über der Anzahl Nicht-Null Elemente für unterschiedliche Bauteile aus dem Vorderwagenabschnittsmodell

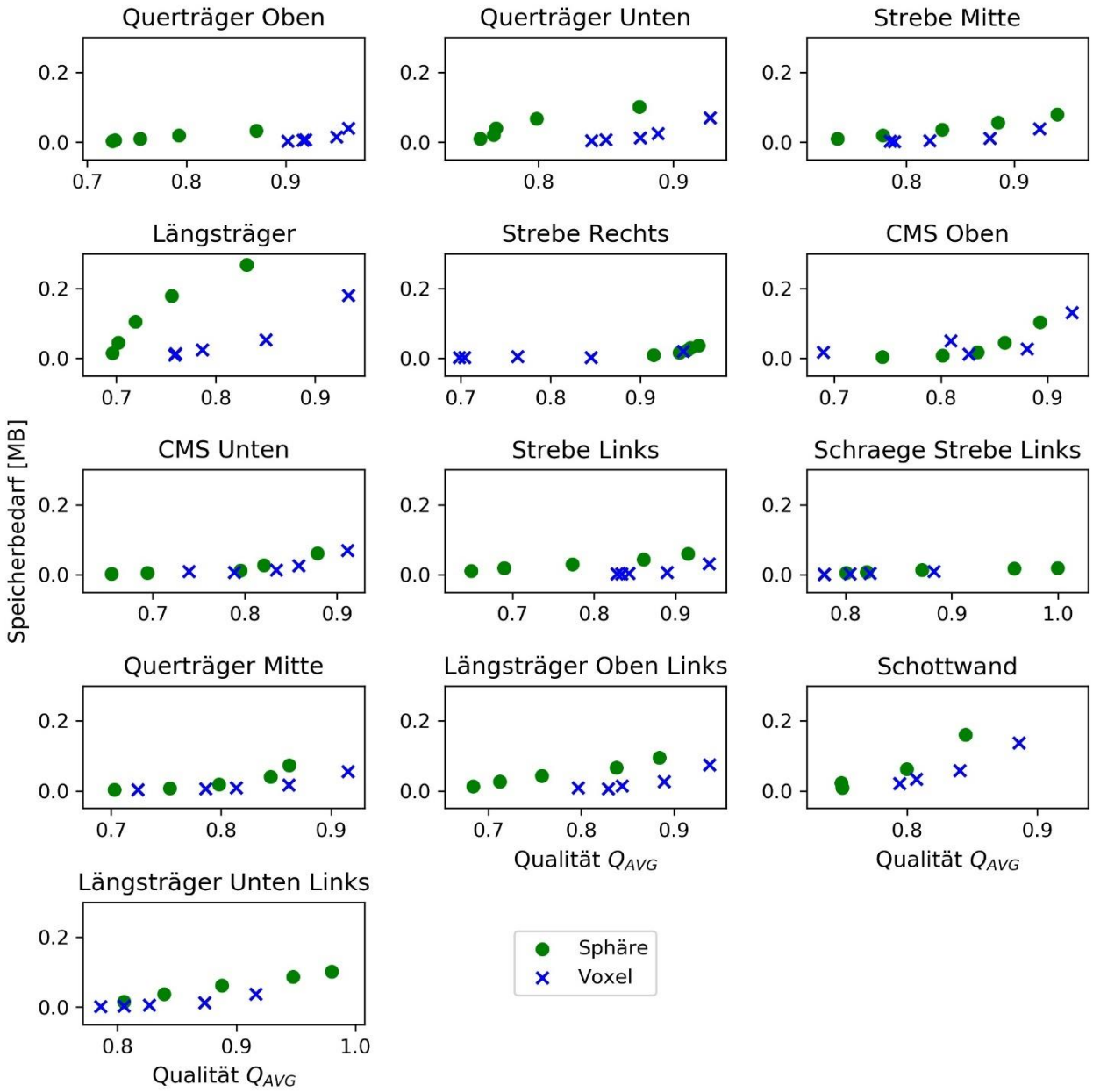


Abbildung 98: Darstellung des zeitlichen Berechnungsaufwands über der Anzahl Nicht-Null Elemente für unterschiedliche Bauteile aus dem Vorderwagenabschnittsmodell

Lebenslauf

Persönliche Daten

Name	David Kracker
Geburtsdatum	17.08.1993
Geburtsort	Sindelfingen

Schulbildung

1999 bis 2003	Justinus-Kerner-Grundschule Böblingen
2003 bis 2012	Albert-Einstein-Gymnasium Böblingen

Studium

10/2012 bis 10/2015	Universität Stuttgart – B.Sc. Physik
10/2015 bis 05/2018	Universität Stuttgart – M.Sc. Elektromobilität

Berufstätigkeit

06/2018 bis 10/2018	Berechnungsingenieur bei der *Dr. Ing. h.c. F. Porsche AG*, Entwicklungszentrum Weissach
11/2018 bis 10/2021	Doktorand (Entwicklung Karosserie Vorentwicklung & CAE) bei der *Dr. Ing. h.c. F. Porsche AG*, Entwicklungszentrum Weissach In Kooperation mit dem *Lehrstuhl für Optimierung mechanischer Strukturen* an der *Bergischen Universität Wuppertal*
11/2021 bis 10/2022	Data-Scientist/Berechnungsingenieur (Entwicklung Karosserie Vorentwicklung & CAE) bei der *Dr. Ing. h.c. F. Porsche AG*, Entwicklungszentrum Weissach
Seit 11/2022	Data-Scientist (Datengetriebene Entwicklung) bei der *Dr. Ing. h.c. F. Porsche AG*, Entwicklungszentrum Weissach

Dissertationen vom Lehrstuhl für Optimierung mechanischer Strukturen, Fakultät 7, Bergische Universität Wuppertal

1. Dr.-Ing. Christopher Ortmann (2015): Entwicklung eines graphen- und heuristikbasierten Verfahrens zur Topologieoptimierung von Profilquerschnitten für Crashlastfälle, Shaker Verlag, ISBN: 978-3-8440-3746-3

2. Dr.-Ing. Robert Dienemann (2018): Entwicklung einer Optimierungsmethodik für die Form- und Topologieoptimierung von tiefziehbaren Blechstrukturen, Shaker Verlag, ISBN: 978-3-8440-6196-3

3. Dr.-Ing. Constantin Diez (2018): Process for Extraction of Knowledge from Crash Simulations by means of Dimensionality Reduction and Rule Mining [https://d-nb.info/1182555063/34]

4. Dr.-Ing. Manuel Ramsair (2021): Integration der Topologie- und Formoptimierung in den automatisierten digitalen Entwurf von Fachwerkstrukturen, Shaker Verlag, ISBN: 978-3-8440-7788-9

5. Dr.-Ing. Niklas Klinke (2021): Strategien zur Optimierung von flexibel gewalzten Bauteilen in Karosseriestrukturen, Shaker Verlag, ISBN: 978-3-8440-7936-4

6. Dr.-Ing. Saad Eddine Hafsa (2021): Topology optimization method for the adaptation of mechanical structures, Shaker Verlag, ISBN: 978-3-8440-8306-4

7. Dr.-Ing. Katrin Weider (2021): Topologische Ableitung zur Optimierung crashbelasteter Strukturen, Shaker Verlag, ISBN: 978-3-8440-8248-7

8. Dr.-Ing. Jana Büttner (2022): Effiziente Lösungsansätze zur Reduktion des numerischen Ressourcenbedarfs für den operativen Einsatz der Multidisziplinären Optimierung von Fahrzeugstrukturen, Shaker Verlag, ISBN: 978-3-8440-8560-0

9. Dr.-Ing. Johannes Sperber (2022): Graphen- und Heuristikbasierte Topologieoptimierung axial belasteter Crashstrukturen, Shaker Verlag, ISBN: 978-3-8440-8634-8

10. Dr.-Ing. Stefan Mertler: Comparative Analysis of Crash Simulation Results using Generative Nonlinear Dimensionality Reduction, Shaker Verlag, ISBN: 978-3-8440-8761-1

11. Dr.-Ing. Sven Wielens: Automatische Erstellung von Submodellen für die Crashoptimierung von Fahrzeugkarosserien, Shaker Verlag, ISBN: 978-3-8440-8717-8