

Forschungsberichte aus dem
wbk Institut für Produktionstechnik
Karlsruher Institut für Technologie (KIT)

Carmen Maria Krahe

**KI-gestützte produktionsgerechte
Produktentwicklung**
Automatisierte Wissensextraktion aus
vorhandenen Produktgenerationen

Band 265



Forschungsberichte aus dem
wbk Institut für Produktionstechnik
Karlsruher Institut für Technologie (KIT)

Hrsg.: Prof. Dr.-Ing. Jürgen Fleischer
Prof. Dr.-Ing. Gisela Lanza
Prof. Dr.-Ing. habil. Volker Schulze

Carmen Maria Krahe

**KI-gestützte produktionsgerechte
Produktentwicklung**
Automatisierte Wissensextraktion aus vorhandenen
Produktgenerationen

Band 265

KI-gestützte produktionsgerechte Produktentwicklung
Automatisierte Wissensextraktion aus vorhandenen
Produktgenerationen

Zur Erlangung des akademischen Grades einer
Doktorin der Ingenieurwissenschaften (Dr.-Ing.)

bei der KIT-Fakultät für Maschinenbau des
Karlsruher Instituts für Technologie (KIT)

angenommene

Dissertation

von

M.Sc. Carmen Maria Krahe

aus Karlsruhe

Tag der mündlichen Prüfung: 05.12.2022

Hauptreferentin: Prof. Dr.-Ing. Gisela Lanza

Korreferent: Prof. Dr. Klaus-Robert Müller

Bibliografische Information der Deutschen Nationalbibliothek

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.d-nb.de> abrufbar.

Zugl.: Karlsruhe, Karlsruher Institut für Technologie, Diss., 2022

Copyright Shaker Verlag 2023

Alle Rechte, auch das des auszugsweisen Nachdruckes, der auszugsweisen oder vollständigen Wiedergabe, der Speicherung in Datenverarbeitungsanlagen und der Übersetzung, vorbehalten.

Printed in Germany.

ISBN 978-3-8440-8953-0

ISSN 0724-4967

Shaker Verlag GmbH • Am Langen Graben 15a • 52353 Düren
Telefon: 02421 / 99 0 11 - 0 • Telefax: 02421 / 99 0 11 - 9
Internet: www.shaker.de • E-Mail: info@shaker.de

Vorwort des Herausgebers

Die schnelle und effiziente Umsetzung innovativer Technologien wird vor dem Hintergrund der Globalisierung der Wirtschaft der entscheidende Wirtschaftsfaktor für produzierende Unternehmen. Universitäten können als "Wertschöpfungspartner" einen wesentlichen Beitrag zur Wettbewerbsfähigkeit der Industrie leisten, indem sie wissenschaftliche Grundlagen sowie neue Methoden und Technologien erarbeiten und aktiv den Umsetzungsprozess in die praktische Anwendung unterstützen.

Vor diesem Hintergrund wird im Rahmen dieser Schriftenreihe über aktuelle Forschungsergebnisse des Instituts für Produktionstechnik (wbk) am Karlsruher Institut für Technologie (KIT) berichtet. Unsere Forschungsarbeiten beschäftigen sich sowohl mit der Leistungssteigerung von additiven und subtraktiven Fertigungsverfahren, den Produktionsanlagen und der Prozessautomatisierung sowie mit der ganzheitlichen Betrachtung und Optimierung der Produktionssysteme und -netzwerke. Hierbei werden jeweils technologische wie auch organisatorische Aspekte betrachtet.

Prof. Dr.-Ing. Jürgen Fleischer

Prof. Dr.-Ing. Gisela Lanza

Prof. Dr.-Ing. habil. Volker Schulze

Vorwort des Verfassers

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftliche Mitarbeiterin am wbk Institut für Produktionstechnik des Karlsruher Instituts für Technologie (KIT) im Rahmen des vom Bundesministerium für Bildung und Forschung (BMBF) geförderten Projekts „Machine Learning-driven Engineering – CAx goes AIx“ (Förderkennzeichen 01IS18048).

Besonders bedanken möchte ich mich zunächst bei Frau Prof. Dr.-Ing. Gisela Lanza, die meine Arbeit nicht nur als Hauptreferentin betreut, sondern mich durch ihr Vertrauen und ihre kontinuierliche Unterstützung sowohl persönlich als auch fachlich fortwährend gefördert hat. Weiter danke ich Herrn Prof. Dr. Klaus-Robert Müller für das Interesse an meiner Arbeit und die Übernahme des Korreferats sowie Herrn Prof. Dr.-Ing. Frank Gauterin für den Prüfungsvorsitz. Allen Kolleginnen und Kollegen des wbk, insbesondere im Bereich PRO, danke ich für die fachliche als auch seelische Unterstützung in den letzten Jahren. Ein besonderer Dank gilt Constantin Hofmann, Daniel Gauder, Lukas Weiser, Leonard Schild, Lucas Bretz, Sina Peukert und Julia Dvorak, die mich durch alle Höhen und Tiefen begleitet und stets wieder ermutigt haben. Für das Korrekturlesen dieser Arbeit danke ich Constantin Hofmann und Marvin May. Mein besonderer Dank gilt zudem allen Studierenden, die mit großem Engagement zum Gelingen der Arbeit beigetragen haben. Speziell möchte ich mich bei Tassilo Holtzwardt, Maximilian Bräuner, Sebastian Ebi, Antonio Bräunche, Theresa Schmutz, Yannik Hermann und Stephan Didong bedanken.

Ein besonderer Dank gilt meinen Eltern Eva und Bernd und meinem Bruder Dominik für ihre vorbehaltlose Unterstützung und ihren stetigen Rückhalt.

Mein abschließender und allergrößter Dank gilt meinem Freund Pascal für seine unendliche Geduld, sein Vertrauen und seine positive Energie, die auch in den schwierigsten Phasen stets ein Lächeln in mein Gesicht gezaubert hat.

Karlsruhe, im Dezember 2022

Carmen Krahe

Abstract

Today, manufacturing companies face the challenge of bringing innovative products to the market at the lowest possible price in an ever shorter time due to strong global competition. This results in enormous time and cost pressure, especially for product development, as the so-called time-to-market is crucial for the market success of a product. The reuse of existing product models and the knowledge they contain offer great potential for reducing development time. However, this knowledge base is often not yet systematically used, since it is largely implicit and hardly formalizable. Yet, the increasing use of digital tools in product development and the resulting growth of data offer the possibility to extract, formalize and utilize this (implicit) knowledge by means of data-driven approaches.

The goal of the dissertation is the development of a method for an automated extraction of implicit knowledge in the form of features and patterns from existing product models with the help of machine learning and the subsequent use for support in design. For this purpose, the relevant geometric features are first learned from the product models, specifically CAD models, with the help of autoencoders. Additionally, further product describing information, which particularly affects subsequent producibility, is extracted. This geometric footprint of a product component forms the basis for the developed methodology. By means of design patterns learned from the product models by a Recurrent Neural Network, the subsequent state can be predicted for a given design state. Based on the geometric properties of this prediction, the most similar existing final models for the given (semi-finished) CAD model are shown. Those can then be sorted regarding further product information. Based on the set of geometrically similar models, the producibility of the current design is then evaluated by showing the usual value combinations for defined production-relevant product properties.

The procedure is demonstrated using mechanical components from an industrial data set. For exemplary products in different design states the similarity search, the evaluation of production-relevant product properties, the prediction of next design states as well as the interaction of the individual method modules are shown.

The methodology enables to identify similar product models for early design states in order to promote knowledge reuse and reduce the design of duplicates. In addition, first indications of possible problems with regard to later producibility can be given at an early stage.

Kurzfassung

Produzierende Unternehmen stehen heutzutage vor der Herausforderung, aufgrund des starken globalen Wettbewerbsdrucks in immer kürzerer Zeit dennoch innovative Produkte zu einem möglichst günstigen Preis auf den Markt zu bringen. Insbesondere für die Produktentwicklung entsteht dadurch ein enormer Zeit- und Kostendruck, allerdings ist die sogenannte *Time-to-Market* entscheidend für den Markterfolg eines Produktes. Durch die Wiederverwendung von bereits existierenden Produktmodellen sowie dem darin enthaltenen Wissen besteht großes Potenzial, diese Entwicklungszeit zu reduzieren. Jedoch wird diese Wissensbasis in Form bestehender Produktmodelle häufig noch nicht systematisch genutzt. Das Problem ist, dass dieses größtenteils implizite Wissen häufig nicht ohne weiteres formalisierbar ist. Durch die zunehmende Nutzung digitaler Tools auch im Bereich der Produktentwicklung und die damit einhergehende wachsende Datenbasis ergibt sich über datengetriebene Ansätze jedoch die Möglichkeit, dieses (implizite) Wissen zu extrahieren, zu formalisieren und nutzbar zu machen.

Das Ziel dieser Dissertation ist die Entwicklung einer Methode zur automatisierten Extraktion von implizitem Wissen in Form von *Features* und Mustern aus vorhandenen Produktmodellen mit Hilfe von Verfahren des Maschinellen Lernens sowie dessen anschließende Nutzung zur Unterstützung in der Produktentwicklung. Hierfür werden über *Autoencoder* zunächst die relevanten geometrischen Eigenschaften aus den Produktmodellen in Form von CAD-Modellen erlernt. Darüber hinaus werden weitere produktbeschreibende Informationen, die insbesondere die spätere Produzierbarkeit beeinflussen, extrahiert. Dieser geometrische Fußabdruck einer Produktkomponente bildet die Grundlage für die entwickelte Methode. Auf Basis von Konstruktionsmustern, die mit Hilfe von *Recurrent Neural Networks* aus den Produktmodellen erlernt werden, kann für einen gegebenen Konstruktionszustand der Folgezustand prädiziert werden. Auf Grundlage der geometrischen Eigenschaften dieser Vorhersage werden für das gegebene (halbfertige) CAD-Modell die ähnlichsten bereits existierenden finalen Modelle aufgezeigt. Diese können anschließend bezüglich der weiteren Produktinformationen sortiert werden. Anhand der Menge der ähnlichsten Modelle können die produktionsrelevanten Produkteigenschaften des aktuellen Konstruktionszustandes durch das Aufzeigen von üblichen Ausprägungskombinationen bewertet werden.

Das Vorgehen wird anhand von mechanischen Komponenten eines industriellen Datensatzes demonstriert. Für verschiedene Konstruktionszustände beispielhafter Produkte werden die Ähnlichkeitssuche, die Bewertung produktionsrelevanter Produkteigenschaften, die Vorhersage nächster Konstruktionszustände sowie die Interaktion der einzelnen Methodenbausteine aufgezeigt.

Durch die Methodik können bereits für anfängliche Konstruktionszustände ähnliche Produktmodelle identifiziert werden, wodurch die Wiederverwendung von Wissen gefördert sowie die Generierung von Dubletten reduziert werden. Darüber hinaus können bereits frühzeitig Hinweise auf mögliche Probleme bezüglich der späteren Produzierbarkeit gegeben werden.

Inhaltsverzeichnis

Inhaltsverzeichnis	I
Formelverzeichnis	IV
Abkürzungsverzeichnis	VIII
1 Einleitung	1
1.1 Motivation	1
1.2 Zielsetzung	2
1.3 Aufbau der Arbeit	4
2 Grundlagen	5
2.1 Produktentstehungsprozess	5
2.1.1 Produktentwicklung in Theorie und Praxis	6
2.1.2 Produktionsgerechte Produktentwicklung	8
2.1.3 Computergestützte Konstruktion mit CAD	11
2.1.4 Wissensbasiertes Konstruieren	15
2.2 Maschinelles Lernen	17
2.2.1 Feed Forward Neuronale Netze	18
2.2.2 Autoencoder	21
2.2.3 Clusteranalyse	25
2.2.4 Rekurrente Neuronale Netze	29
2.2.5 Deep Learning mit 3D-Daten	33
3 Stand der Technik	36
3.1 Maschinelles Lernen zur Repräsentation und Verarbeitung von 3D-Objekten	36
3.2 Ansätze zur Unterstützung im Produktentwicklungsprozess	39
3.2.1 Ansätze zur Ähnlichkeitssuche von CAD-Modellen	40
3.2.2 Ansätze zur Analyse produktionsrelevanter Produkteigenschaften	42
3.2.3 Ansätze zum Vorschlagen nächster Konstruktionsschritte	43
3.2.4 Ansätze für Assistenzsysteme	44
3.3 Kommerzielle Assistenzsysteme für die Konstruktion	46

3.4	Forschungsdefizit	47
4	Eigener Ansatz	52
4.1	Kategorisierung der Datenbasis	56
4.1.1	3D-Geometrie	58
4.1.2	Metainformationen	59
4.1.3	Konstruktionsvorgehen	59
4.1.4	Anwendungsfallspezifische Zuordnung der Umsetzungsstufen	62
4.2	Aufbereitung der Daten zur Repräsentation des Wissens	64
4.2.1	3D-Objekte	64
4.2.2	Kategorische und numerische Daten	66
4.2.3	Sequentielle Daten	67
4.3	Automatisierte Extraktion relevanter Eigenschaften mittels Autoencoder	70
4.3.1	Autoencoder-Architekturen für 3D-Geometrien	71
4.3.2	Autoencoder für sequentielle Daten des Konstruktionsvorgehens	77
4.3.3	Bewertung der Repräsentativität des latenten Raumes	78
4.4	Verwendung des Wissens zur Konstruktionsunterstützung	80
4.4.1	Ähnlichkeitssuche und Bewertung produktionsrelevanter Produkteigenschaften	82
4.4.2	Sequenzanalyse zur Vorhersage nächster Konstruktionsschritte	94
4.4.3	Integration in ein Gesamtsystem	101
5	Evaluation	104
5.1	Kategorisierung der Datenbasis	104
5.1.1	3D-Geometrie	105
5.1.2	Metainformationen	105
5.1.3	Konstruktionsvorgehen	107
5.1.4	Zuordnung der Umsetzungsstufen für den Anwendungsfall	110
5.2	Aufbereitung der Datenbasis zur Repräsentation des Wissens	111
5.3	Automatisierte Extraktion relevanter Eigenschaften mittels Autoencoder	114
5.3.1	Anwendung des Autoencoders auf die 3D-Geometrien finaler Modelle	115

5.3.2 Anwendung des Autoencoders auf die sequentiellen Daten des Konstruktionsvorgehens	123
5.4 Verwendung des Wissens zur Konstruktionsunterstützung	127
5.4.1 Ähnlichkeitssuche und Bewertung produktionsrelevanter Produkteigenschaften	128
5.4.2 Sequenzanalyse zur Vorhersage nächster Konstruktionsschritte	139
5.4.3 Integration in ein Gesamtsystem	146
6 Diskussion und Ausblick	153
6.1 Diskussion	153
6.2 Ausblick	157
7 Zusammenfassung	160
Liste eigener Publikationen	163
Literaturverzeichnis	165
Abbildungsverzeichnis	I
Tabellenverzeichnis	XII
Anhang	XIII

Formelverzeichnis

Formelzeichen	Bedeutung
$a(o)$	Mittlere Distanz eines Objektes o zu allen Objekten des gleichen Clusters
a_i	Output von Knoten i eines Neuronalen Netzes
b	Bias
$b(o)$	Kleinste mittlere Distanz eines Objektes o zu allen anderen Clustern, denen o nicht zugehört
c	Index der Klassen $c= 1, \dots, C$
CS	<i>Completeness Score</i>
d_2	Euklidische Distanz
$d_{2,max}$	Maximale euklidische Distanz zwischen zwei latenten Vektoren aus der Datenbasis
$d_{2,max,w}$	Maximale euklidische Distanz zwischen der Metainformation w zweier Objekte aus der Datenbasis
d_{CD}	Chamfer Distanz zwischen zwei Punktwolken
F	Anzahl Punkte auf Flächen Polygon
h	Index der Cluster der produktionsrelevanten Produkteigenschaften, $h= 1, \dots, H$
$H(c)$	Entropie der Gesamtheit der Klassen c
$H(c k)$	Bedingte Entropie der Klassen c bei gegebener Clusterzuordnung k
$H(k)$	Entropie der Gesamtheit der Cluster k
$H(k c)$	Bedingte Entropie der Cluster k bei gegebener Klassenzuordnung c
HS	<i>Homogeneity Score</i>
j	Index der Objekte aus der vorhandenen Datenbasis mit $j=1, \dots, J$

k	Index der Cluster bei k -Means mit $k=1, \dots, K$
L	Verlustfunktion eines Neuronalen Netz
m	Dimension des latenten Raumes mit $m=1, \dots, M$; damit auch Anzahl der Merkmale, die für k -Means genutzt werden
$MinPts$	Mindestanzahl an Nachbarn innerhalb der ε_{DB} -Umgebung eines Objektes im Rahmen des Clusterverfahrens DBSCAN
n	Dimension der Punkte der Inputpunktewolke mit $n=1, \dots, N$
o	Index der Objekte $o=1, \dots, O$ für eine Clusterbildung
o_c	Anzahl an Objekten in Klasse c
$o_{c,k}$	Anzahl an Objekten in Cluster k und Klasse c
o_k	Anzahl an Objekten in Cluster k
P	Punktewolke bestehend aus N Punkten
PP_h	Bereich der produktionsrelevanten Produkteigenschaften für Cluster h mit $h=1, \dots, H$
Q	Sequenzabschnittslänge
SC	<i>Silhouette Score</i>
$Sim_{ij,ges}$	Gesamtähnlichkeit (Geometrie und Metainformationen) zwischen Objekt i und j
Sim_{ij}	Ähnlichkeit zwischen Objekten i und j bezüglich der Geometrie (latente Vektoren)
$Sim_{ij,w}$	Ähnlichkeit zwischen Objekt i und j bezüglich Metainformation w
S_j	Menge aller Vektoren \vec{x}_j mit $j=1, \dots, J$
S_h	Die Menge der Komponenten j , die dem Cluster h zugeordnet werden
$S_{sim,i}$	Menge der (geometrisch) ähnlichsten Modelle für Input i
$S_{z,i}$	Menge der relevanten Clusterzentren für Input i
S_{z_k}	Menge aller Vektoren des Clusters mit Zentrum z_k

SL	Gesamte Sequenzlänge einer Konstruktionshistorie
V	Eckpunkte eines Polygonnetzes
VM	V-Maß
w	Index der Metainformationen, $w=1, \dots, W$
x	Allgemeiner Input in ein Neuronales Netz
\vec{x}_i	Latenter Vektor des Input-Modells i
\vec{x}_{int}	Linearkombination zweier latenter Vektoren
\vec{x}_j	Latenter Vektor des Objektes j aus der Datenbasis
$\vec{x}^{(t)}$	Latenter Vektor zum Zeitpunkt t
$\hat{\vec{x}}^{(t)}$	Vorhergesagter latenter Vektor zum Zeitpunkt t
$\vec{x}_{z,k}$	Latenter Vektor des Clusterzentrums z_k
y	Gewünschter Output eines Neuronalen Netzes
\hat{y}	Geschätzter Output durch ein Neuronales Netz
y_w	(Normierte) Metainformation w
\vec{y}_w	Vektor aller Metainformationen für $w=1, \dots, W$
$y_{w,max}$	Maximale Ausprägung einer Metainformation
$y_{w,min}$	Minimale Ausprägung einer Metainformation
$y_{w,j}$	Metainformation w des Objektes j
$y_{w,normiert}$	Normierte Metainformation w
z_k	Clusterzentrum des Clusters k (mit $k=1, \dots, K$)
α_{geo}	Gewichtung der Geometrie
α_w	Gewichtung der Metainformation w
γ	Schrittweite für die lineare Interpolation zweier Vektoren
ε	Suchradius der geometrischen Ähnlichkeitssuche
ε_{DB}	Schwellenwert für das Clusterverfahren DBSCAN

λ	Lernrate
θ	Gewichtungsmatrix
ϑ_{ij}	Gewichtung der Verbindung zwischen Knoten i und j in einem neuronalen Netz
ρ	Dichte
ϕ_j	Aktivierungsfunktion von Knoten j im neuronalen Netz

Abkürzungsverzeichnis

Abkürzung	Bedeutung
AE_ZW	Modell des <i>Autoencoders</i> , das auf allen Zwischenzuständen trainiert wurde
AE_Final	Modell des <i>Autoencoders</i> , das nur auf finalen Zuständen trainiert wurde
BNS	<i>Bottleneck Size</i>
CAD	<i>Computer Aided Design</i>
CAx	<i>Computer Aided x</i>
CD	<i>Chamfer Distance</i>
CNN	<i>Convolutional Neural Network</i>
CRISP-DM	<i>Cross Industry Standard Process for Data Mining</i>
CSG	<i>Constructive Solid Geometry</i>
DBSCAN	<i>Density-Based Spatial Clustering of Applications with Noise</i>
DFP	<i>Design for Production</i>
DFM	<i>Design for Manufacturing</i>
Gesamt_SO	Datensatz bestehend aus allen Zwischenzuständen in Standardausrichtung
Finale_Komponenten_NS0	Datensatz bestehend aus nur finalen Bauteilzuständen ohne Standardausrichtung
Finale_Komponenten_SO	Datensatz bestehend aus nur finalen Bauteilzuständen in Standardausrichtung
KBE	<i>Knowledge Based Engineering</i>
KDD	<i>Knowledge Discovery in Databases</i>
KE	Konstruktionselement
KI	Künstliche Intelligenz

LV	Latenter Vektor
ML	<i>Machine Learning</i>
MLP	<i>Multilayer Perceptron</i>
MSE	<i>Mean Squared Error</i>
PDM	<i>Product Data Management</i>
PLM	<i>Product Lifecycle Management</i>
PRT	Herstellerspezifisches CAD-Dateiformat
RNN	<i>Recurrent Neural Network</i>
STEP	<i>Standard for the External Representation of Product Definition Data</i>
STL	Stereolitographie-Format bzw. <i>Standard Triangulation Language</i>

1 Einleitung

In den nachfolgenden Unterkapiteln werden zunächst die dieser Arbeit zugrunde gelegte Motivation und Problemstellung dargelegt. Darauf basierend wird die Zielsetzung sowie der Aufbau der Arbeit vorgestellt.

1.1 Motivation

Produzierende Unternehmen stehen heutzutage vor der Herausforderung, aufgrund des starken globalen Wettbewerbsdrucks in immer kürzerer Zeit innovative Produkte zu einem möglichst günstigen Preis auf den Markt zu bringen (Fleischer 2019; Eversheim, Schuh & Assmus 2005). Durch die steigende Nachfrage nach kundenindividuellen Produkten wird zeitgleich eine enorme Variantenvielfalt gefordert (Fleischer 2019), die nicht nur für die Produktion, sondern auch für die Produktentwicklung eine hohe Komplexität bedeutet (Fleischer 2019; Bauernhansl 2014; Albers et al. 2014a). Insbesondere für die Produktentwicklung entsteht dadurch ein enormer Zeit- und Kostendruck (Humpa 2016; Fleischer 2019). Allerdings ist die sogenannte *Time-to-Market* entscheidend für den Markterfolg eines Produktes (Vajna et al. 2018). Um dem starken Wettbewerb standzuhalten, müssen stets neue Varianten oder Generationen auf den Markt gebracht werden, um attraktiv zu bleiben oder neuen Anforderungen gerecht zu werden. Aus Effizienzgründen basiert die Produktentwicklung daher häufig auf Referenzprodukten, das sind beispielsweise Produkte älterer Generationen oder von Wettbewerbern (Albers, Bursac & Wintergerst 2015). Etwa 80% der Produkte werden in Generationen entwickelt (Albers et al. 2016). Die Basisstruktur von Produkten bleibt beim Übergang von Generation zu Generation häufig gleich (Albers et al. 2014b), weshalb nur einzelne Teilsysteme neu entwickelt werden. Durch die Übernahme oder geringfügige Modifikation bestehender Teilsysteme werden der technische Neuheitsgrad, potenzielle Risiken, z.B. hinsichtlich Funktionalität und Produzierbarkeit, sowie erforderliche Investitionen reduziert (Albers, Bursac & Wintergerst 2015). Nachträglich teure und zeitintensive Änderungen können somit vermieden werden. In vorhandenen Produktgenerationen steckt folglich jahrelange Ingenieurserfahrung (Küstner 2020; Orth, Voigt & Kohl 2011; Albers et al. 2014a), worauf bei der Entwicklung einer neuen Variante oder Generation zurückgegriffen werden kann. Bei diesem Wissen handelt es sich beispielsweise um die gestalterische Umsetzung von Produktfunktionalitäten sowie die fertigungsgerechte und wirtschaftliche Gestaltung von Produkten (Humpa 2016; Wu,

Xiao & Zheng 2021). Iyer et al. (2005) bezeichnen die Wiederverwendung von vorhandenen Produktdesigns sowie des damit verknüpften Wissens als den Schlüssel zum Erfolg, um die Entwicklungszeit neuer Produkte zu reduzieren. Jedoch wird diese Wissensbasis häufig nicht systematisch genutzt (Küstner 2020; Orth, Voigt & Kohl 2011). Als Folge werden Dubletten generiert, wodurch Kosten verursacht und die Innovationsgeschwindigkeit verringert werden, da weniger Kapazität für kreative Tätigkeiten bleibt. Außerdem bedarf es langer Lernschleifen zum Einarbeiten neuer Mitarbeiter (Fleischer 2019). Turki (2014) nennt verschiedene Barrieren für die Weitergabe von Erfahrungswissen, unter anderem die Fluktuation von Mitarbeitern¹, an die das Wissen gebunden ist sowie fehlende Tools zur Wissensweitergabe. Zudem ergibt sich das Problem, dass implizites Wissen häufig nicht ohne weiteres formalisierbar ist (Albers, Börsting & Turki 2011). Durch die zunehmende Nutzung digitaler Tools im Bereich der Produktentwicklung und die damit einhergehende wachsende Datenbasis (Lupinetti et al. 2017; Vajna et al. 2018; Breitsprecher et al. 2015) ergibt sich über datengetriebene Ansätze jedoch die Möglichkeit, aus den vorhandenen Daten (implizites) Wissen zu extrahieren, zu formalisieren und nutzbar zu machen (Küstner 2020; Röhner, Breitsprecher & Wartzack 2011). Das selbstständige Erlernen von Mustern und Eigenschaften aus Daten wird auch als Maschinelles Lernen (ML) bezeichnet (Goodfellow, Bengio & Courville 2016). Die Anwendung von ML-Verfahren zur Formalisierung des (impliziten) Wissens und die dadurch ermöglichte systematische Nutzung im Bereich der Produktentwicklung, um auch produktionstechnische Aspekte im Sinne eines Produkt-Produktions-Codesigns (Albers et al. 2022) frühzeitig zu berücksichtigen, stehen im Fokus dieser Arbeit.

1.2 Zielsetzung

In vielen Unternehmen liegen im Zuge der Digitalisierung des Produktentstehungsprozesses eine große Menge an Produktdaten, z.B. aus dem *Computer Aided Design* (CAD) (Küstner 2020), aus vorhandenen Produktgenerationen vor (Vajna et al. 2018; Lupinetti et al. 2017). Neben dem impliziten Wissen, das in den Produktdaten selbst enthalten ist, stellt auch das Erfahrungswissen von Mitarbeitern über deren Wiederverwendbarkeit einen Wissensschatz dar. Die systematische Nutzung dieser Wissensbasis mit Hilfe von ML-Verfahren bietet ein großes Potenzial, um den Herausforderungen

¹ Zur besseren Lesbarkeit verzichtet diese Arbeit auf die gleichzeitige Verwendung männlicher und weiblicher Sprachformen. Es wird das generische Maskulinum verwendet, wobei alle Geschlechter gleichermaßen gemeint sind.

in der Produktentwicklung gerecht zu werden, und stellt den Fokus dieser Arbeit dar. Im Kern stehen dabei die drei nachfolgenden Forschungsfragen:

- F1 Wie kann **Maschinelles Lernen** genutzt werden, um **Wissen** in Form von vorhandenen CAD-Modellen sowie den damit verknüpften Informationen zielgerichtet zur Verfügung zu stellen, um deren **systematische Wiederverwendung** zu fördern?
- F2 Welche Rückschlüsse bezüglich **produktionsrelevanter Produkteigenschaften** lassen sich aus vorhandenen CAD-Modellen ziehen?
- F3 Wie können mittels **Maschinellern Lernen Muster im Konstruktionsvorgehen** aus vorhandenen CAD-Modellen **erlernt werden**? Wie können diese für den Konstruktionsprozess neuer Generationen genutzt werden?

Das auf Basis dieser Forschungsfragen abgeleitete Ziel der vorliegenden Arbeit ist die Entwicklung einer Methode zur automatisierten Extraktion von Wissen aus bestehenden Produktmodellen² mit Hilfe von Verfahren der Künstlichen Intelligenz (KI), speziell des MLs, und der anschließenden Nutzung dieses Wissens im Produktentstehungsprozess. Mittels der Methodik soll das implizite Wissen in Form von Eigenschaften und Mustern aus vorhandenen Produktgenerationen und –varianten extrahiert und formalisiert werden, um insbesondere eine produktionsgerechte Produktentwicklung zu unterstützen. Dabei wird der Fokus auf Produktmodelle in Form von CAD-Modellen mechanischer Komponenten gelegt. Durch eine geeignete Bereitstellung dieses Wissens soll der Konstrukteur bei der Entwicklung einer neuen Generation oder Variante dahingehend unterstützt werden, sich an bewährten Konstruktionsvorgehen bereits existierender Produkte zu orientieren oder diese sogar wiederzuverwenden. Dadurch wird das bestehende Erfahrungswissen genutzt, um somit möglichen Risiken (z.B. Nicht-Produzierbarkeit auf bestehenden Maschinen), Fehlern, unnötiger Variantenbildung und nachträglich teuren Änderungen entgegenzuwirken. Zur Erreichung dieser Zielstellung werden im Rahmen dieser Arbeit folgende Bestandteile umgesetzt:

- Die Kategorisierung des bestehenden Wissens in Form der vorhandenen Produktmodelle sowie anschließend dessen entsprechende Aufbereitung

² Im Rahmen dieser Arbeit wird unter einem Produktmodell das CAD-Modell sowie die damit verknüpften Informationen verstanden.

- Die automatisierte Extraktion relevanter Eigenschaften aus CAD-Modellen mittels ML-Verfahren, um das vorhandene Wissen in geeigneter Form zu repräsentieren
- Die Nutzung der extrahierten Informationen in Form einer Ähnlichkeitssuche, um auf Basis des aktuellen Konstruktionsstandes bereits existierende, geometrisch ähnliche CAD-Modelle aufzufinden
- Die Ableitung produktionsrelevanter Produkteigenschaften aus den gefundenen, ähnlichen CAD-Modellen zur Bewertung eines neuen Produktmodells
- Das Erlernen von Mustern im Konstruktionsprozess mittels ML-Verfahren zur aktiven Unterstützung des Konstrukteurs durch das Vorschlagen des nächsten Konstruktionszustandes
- Das Aufzeigen der prinzipiellen Machbarkeit und der erzielbaren Ergebnisse im Rahmen einer Validierung

Im nachfolgenden Kapitel wird zunächst die Struktur dieser Arbeit zur Umsetzung der genannten Bestandteile dargelegt.

1.3 Aufbau der Arbeit

Das nachfolgende Kapitel 2 übermittelt neben einer Darstellung der notwendigen theoretischen Grundlagen zum Produktentstehungsprozess die für diese Arbeit relevanten methodischen Kenntnisse über die genutzten ML-Verfahren. Darauf aufbauend fasst Kapitel 3 den aktuellen Stand der Forschung zur Verarbeitung von 3D-Daten mittels ML sowie existierender Ansätze zur Unterstützung in der Produktentwicklung zusammen. Darüber hinaus werden hier auch kommerzielle Systeme vorgestellt, die verschiedene Unterstützungsfunktionen im Bereich CAD anbieten. Abschließend erfolgt die Ableitung des Forschungsdefizits, welches mit dem in Kapitel 4 vorgestellten Lösungsansatz adressiert wird. Dieser gliedert sich zunächst in die Vorverarbeitung der Daten durch eine Kategorisierung (Kapitel 4.1) und entsprechende Aufbereitung (Kapitel 4.2). Darauf aufbauend erfolgt die automatisierte Extraktion der relevanten Informationen aus den Daten (Kapitel 4.3), die anschließend die Grundlage für die Konstruktionsunterstützung (Kapitel 4.4) bilden. In Kapitel 5 wird schließlich die exemplarische Anwendung anhand mechanischer Komponenten eines industriellen Datensatzes aufgezeigt. Eine Diskussion der entwickelten Methodik und der erhaltenen Erkenntnisse sowie die Darstellung möglicher Erweiterungen erfolgt in Kapitel 6. Kapitel 7 schließt mit einer Zusammenfassung der vorliegenden Arbeit.

2 Grundlagen

Zur Einordnung des im Rahmen dieser Arbeit entwickelten Vorgehens werden zunächst die wesentlichen theoretischen Grundlagen über den Produktentstehungsprozess dargestellt. Anschließend erfolgt ein Überblick über die genutzten ML-Verfahren.

2.1 Produktentstehungsprozess

Die Produktentstehung als Teil des Produktlebenszyklus umfasst die Phasen der Produktplanung, der Produktentwicklung, der Entwicklung des Produktionssystems sowie der Produktion bzw. Realisierung. Die im Rahmen dieser Arbeit fokussierte Phase der Konstruktion ist dabei Teil der Produktentwicklung. (VDI 2221-1:2019-11)

Diese Phase der Produktentwicklung wird in Kapitel 2.1.1 näher erläutert.

Häufig finden die verschiedenen Phasen des Produktentstehungsprozesses jedoch nicht sequentiell, sondern parallel statt, was auch unter dem Begriff des *Simultaneous Engineering* zusammengefasst wird. (VDI 2221-1:2019-11)

Neben den Hauptzielen Qualität, Kosten und Zeit der Produktentwicklung werden dabei häufig noch weitere Entwicklungsziele verfolgt, wie beispielsweise *Design-to-Manufacture and Assembly*, für die die Konzepte des *Simultaneous Engineering* die notwendigen organisatorischen Randbedingungen bieten (VDI 2221-1:2019-11). Diese Ziele sind in der Regel schon in der frühen Phase der Entwicklung zu berücksichtigen, da hier die Einflussmöglichkeiten noch am größten sind (Boothroyd 1994; VDI 2221-1:2019-11). Die produktionsgerechte Produktgestaltung ist ein wesentliches Werkzeug des *Simultaneous Engineering* (Boothroyd 1994) und wird in Kapitel 2.1.2 näher beschrieben.

In der Produktentwicklung werden verstärkt entwicklungsunterstützende IT-Werkzeuge eingesetzt, um beispielsweise den Produktentwicklungsprozess zu beschleunigen und eine durchgängige Dokumentation und Austausch von Informationen zu ermöglichen. Diese werden unter dem Begriff CAx (*Computer Aided x*) zusammengefasst. (VDI 2221-1:2019-11) Eine Schlüsselrolle für die virtuelle Produktentwicklung nimmt dabei die rechnergestützte Konstruktion, das sogenannte *Computer Aided Design* (CAD) ein (Hackenschmidt, Hautsch & Kleinschrod 2020). Die für diese Arbeit relevanten Grundlagen zum CAD werden in Kapitel 2.1.3 näher erläutert. Die Kenntnisse zur bisherigen Nutzung von Wissen im Bereich des wissensbasierten Konstruierens werden in Kapitel 2.1.4 dargelegt.

2.1.1 Produktentwicklung in Theorie und Praxis

In der VDI-Richtlinie 2221 wird ein allgemeines Modell der Produktentwicklung vorgestellt, in dem die zentralen Ziele, Aktivitäten und Ergebnisse des Produktentwicklungsprozesses definiert und in den notwendigen Bezugsrahmen gegliedert werden. Die in diesem Vorhaben adressierte Konstruktion ist dabei Teil des Produktentwicklungsprozesses. Sehr häufig werden die Begriffe auch als Synonyme verwendet (VDI 2221-1:2019-11). Das allgemeine Modell der Produktentwicklung ist in Abbildung 2.1 dargestellt.

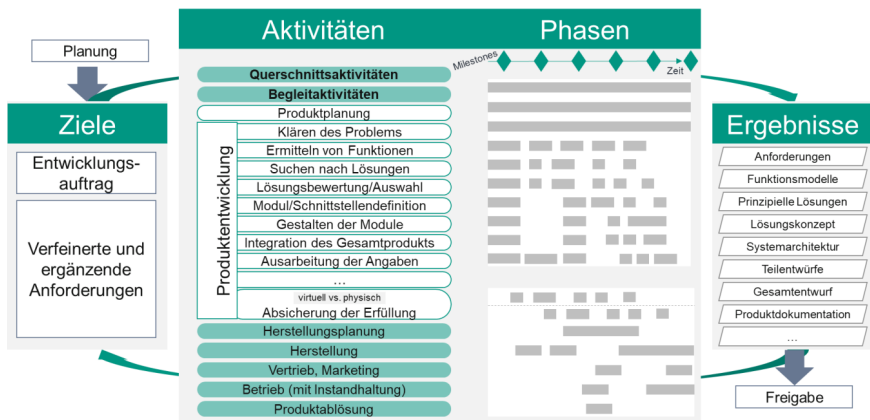


Abbildung 2.1: Modell der Produktentwicklung (in Anlehnung an VDI 2221-1:2019-11 und VDI 2221-2:2019-11)

Die gezeigten Arbeitsabschnitte werden dabei jedoch nicht sequenziell durchgeführt, sondern sind eng miteinander verzahnt und werden daher häufig iterativ sowie (teil-)parallelisiert durchlaufen (VDI 2221-1:2019-11).

Ausgangspunkt sind Ziele und Anforderungen, die im Rahmen der Produktplanung aufgestellt wurden und in Form eines Entwicklungsauftrages an die Produktentwicklung übergeben werden. Zur Umsetzung des Entwicklungsauftrages werden verschiedene Aktivitäten über die Phasen der Konzeptentwicklung, Detaillierung und Ausarbeitung durchgeführt, die in entsprechenden Ergebnissen resultieren. Die (Zwischen-)Ergebnisse werden dabei stets mit den Anforderungen abgeglichen und mittels Absicherung überprüft. Es werden neun verschiedene Aktivitäten definiert, die abhängig von der Entwicklungsaufgabe mit unterschiedlicher Intensität durchzuführen sind. (VDI 2221-1:2019-11)

Aufgrund der hohen Individualität eines jeden Entwicklungsprozesses muss der allgemeine Produktentwicklungsprozess in seinen jeweiligen Elementen und deren Ausprägung kontextspezifisch angepasst werden (VDI 2221-2:2019-11). Die wenigsten Produkte werden nach Eckert, Alink & Albers (2010) jedoch komplett neu entwickelt. Schätzungen zufolge müssen lediglich 20 % der Komponenten, für die ursprünglich eine Neuentwicklung angedacht war, tatsächlich neu entwickelt werden. Die restlichen 80 % können durch die Wiederverwendung und Modifikation bestehender Komponenten umgesetzt werden. (Iyer et al. 2005)

Laut Eckert, Alink & Albers (2010) werden häufig aus ökonomischen und risikoanalytischen Gründen die geforderten Funktionen und Eigenschaften eines neuen Produkts mit möglichst geringfügigen Modifikationen von etablierten Lösungen umgesetzt. Gerade bei komplexen Produkten werden zuverlässig funktionierende Komponenten und Teilsysteme weitestgehend übernommen. Nach Roj et al. (2021) zählt es zur Routine, existierende Komponenten zu überprüfen und zu modifizieren. Die dafür notwendigen Informationen stammen aus dem Erfahrungswissen oder bekannten Projekten (Ahmed, Wallace & Blessing 2003). Ein beträchtlicher Anteil der Arbeitszeit eines Konstrukteurs wird damit für die Beschaffung und Aufbereitung von Informationen aufgewendet (Iyer et al. 2005; Kirchner 2020). Da jedoch insbesondere Erfahrungswissen häufig nicht niedergeschrieben wird, ist es eine große Herausforderung, darauf zuzugreifen (Albers & Turki 2013).

Da in den meisten Entwicklungsprojekten in der Praxis folglich nur ein sehr geringer Anteil neu entwickelt, sondern vielmehr ein Großteil der Teilsysteme übernommen und modifiziert wird, wird in Albers, Bursac & Wintergerst (2015) ein Beschreibungsmodell zur Produktgenerationsentwicklung vorgeschlagen. Demnach basiert eine neue Produktgeneration auf einem Referenzprodukt, z.B. einem Vorgängerprodukt, von dem große Teile der grundsätzlichen Struktur vorgegeben werden. Dabei wird ein Großteil der Teilsysteme des Referenzprodukts übernommen oder lediglich geringfügig angepasst. Allein für die Differenzierungsmerkmale einer neuen Produktgeneration sind Neuentwicklungsanteile notwendig. Durch die Übernahme von zuverlässig funktionierenden Komponenten und Teilsystemen werden der technische Neuheitsgrad, potenzielle Risiken sowie erforderliche Investitionen, z.B. in Produktionsmittel, reduziert. (Albers, Bursac & Wintergerst 2015)

Ebenso werden vorhandene Herstellungswerkzeuge und Fertigungsinfrastrukturen wiederverwendet, weshalb diese einen großen Einfluss auf die Gestaltung der Teilsysteme ausüben. (VDI 2221-2:2019-11) Durch die Nutzung vorhandener Produktionsmittel und des damit verbundenen Erfahrungswissens entfallen zudem Änderungsaufwände und Probleme beim Produktionsanlauf (Kirchner 2020).

Obwohl durch die VDI-Richtlinie 2221 eine strukturierte Vorgehensweise für die Produktentwicklung vorgegeben wird, wird in der Praxis häufig nicht systematisch vorgegangen (Fleischer 2019). Gründe hierfür sind unter anderem der hohe Zeitdruck in den Konstruktionsabteilungen, der ein systematisches Abarbeiten durch den vermehrten Zeit- und Personalbedarf nicht wirtschaftlich erscheinen lässt (Fleischer 2019).

Häufig greifen Entwickler auf bestehende Produktkomponenten zurück (Albers et al. 2014c). Die Problematik dabei ist jedoch, dass der Überblick über bereits existierende Produkte oftmals fehlt, was zu unnötiger Variantenbildung führt, da die Nutzung von Information, die eigentlich schon vorhanden ist, unmöglich wird (Ehrlenspiel et al. 2014). Die kontinuierliche Verbesserung zukünftiger Produkte wird aufgrund dieser Tatsache deutlich erschwert (Ehrlenspiel et al. 2014). Nach einer Umfrage von (Albers et al. 2014c) sinkt mit steigender Unternehmensgröße die Transparenz darüber, welche Entwicklungsprojekte bereits existieren und abgeschlossen sind. Daher ist es wesentlich, die vorhandenen Informationen über abgeschlossene Projekte bereitzustellen.

Die in dieser Arbeit entwickelte Methodik soll insbesondere die Wiederverwendung existierender Komponenten, die aus bestehenden Generationen übernommen oder geringfügig angepasst werden müssen, unterstützen.

2.1.2 Produktionsgerechte Produktentwicklung

Bei der Umsetzung des allgemeinen Modells der Produktentwicklung unter dem Gesichtspunkt des *Simultaneous Engineering* ist eine frühzeitige Einbeziehung der Produktion für eine fertigungsgerechte Gestaltung³ des Produktes unabdingbar. (VDI 2221-2:2019-11; Feldhusen & Grote 2013) Neben einer Verkürzung der *Time-To-Market* wird eine Reduktion der Kosten erreicht, da ein Großteil der späteren Produktkosten bereits in der Produktentwicklung festgelegt wird (Boothroyd 1994; Kirchner 2020). Durch eine entsprechende Gestaltung des Produktes können einerseits die Material-, andererseits

³ Da im Rahmen dieser Arbeit Produkte auf Komponentenebene fokussiert werden, werden die Begriffe fertigungsgerecht und produktionsgerecht synonym verwendet.

die Fertigungskosten durch den Einsatz anderer Herstellprozesse gesenkt werden (Kirchner 2020). Fertigungsgerechtes Konstruieren bedeutet folglich, über konstruktive Maßnahmen sowohl Fertigungszeiten als auch –kosten zu reduzieren, zeitgleich jedoch die fertigungsabhängigen Qualitätsmerkmale einhalten zu können. Durch die Verkürzung des Regelkreises zwischen Konstruktion und Fertigung wird damit die Voraussetzung für eine wirtschaftliche Fertigung gelegt. (Meerkamm et al. 2012)

Beispielsweise muss der Konstrukteur für eine sinnvolle Formgebung der Bauteile Expertenwissen aus verschiedenen Domänen vereinen. Hierzu zählt auch, dass die fertigungstechnischen Gegebenheiten des Betriebes sowie die grundsätzliche fertigungstechnische Umsetzbarkeit von Geometrien bekannt sind. Im Normalfall findet hierfür ein fachlicher Austausch mit internen oder externen Experten statt. (Fleischer 2019)

Darüber hinaus gibt es eine Vielzahl von Gestaltungsrichtlinien, die unter dem Begriff *Design for/to x* zusammengefasst werden (Feldhusen & Grote 2013; VDI 2221-1:2019-11; Meerkamm et al. 2012, S. 451; Ehrlenspiel & Meerkamm 2017), in denen entsprechendes Expertenwissen zusammengetragen wird (Wartzack, Sauer & Küstner 2017). Ein Überblick ist beispielsweise in Feldhusen & Grote (2013) zu finden. Hierzu zählen auch die Gestaltungsrichtlinien zum fertigungsgerechten Konstruieren (*Design for Production (DFP)*, *Design for Manufacturing (DFM)*) (Meerkamm et al. 2012). Die Gestaltungsrichtlinien sind dabei spezifisch für ein Fertigungsverfahren und stark problemorientiert (Meerkamm et al. 2012). Die Umsetzung dieser Richtlinien wird meistens in Form von bildhaft dargestellten Gut- bzw. Schlechtbeispielen gezeigt (Meerkamm et al. 2012; Wartzack, Sauer & Küstner 2017). Die Akquisition des Wissens bezüglich verschiedener *Design for x* Richtlinien ist jedoch zeitintensiv (Kratzer et al. 2012).

Fertigungsbeeinflussende Parameter, die aus der Gestalt resultieren, sind neben der eigentlichen Form zusätzlich auch die Abmessungen, Oberflächenqualität sowie Toleranzen (Feldhusen & Grote 2013; Ehrlenspiel et al. 2014). Diese Parameter sind maßgeblich für die Auswahl der Fertigungsverfahren (Ehrlenspiel et al. 2014) und damit auch für die Fertigungskosten, -qualität und –zeit (Feldhusen & Grote 2013). Durch die Festlegung der Gestalt und des Werkstoffes von Bauteilen werden folglich weitgehend die möglichen Herstellprozesse und die damit verbundenen Herstellkosten bestimmt (Kirchner 2020).

Die für eine Fertigungstechnologieauswahl relevanten Produkteigenschaften sind abhängig vom jeweiligen Anwendungsfall (Jacob 2021). Dennoch können die relevanten

Produktparameter nach Fallböhmer (2000) in die Kategorien Abmessung, Form, Werkstoff, Stückzahl, Toleranzen, Oberfläche und Gewicht eingeteilt werden (siehe Abbildung 2.2).

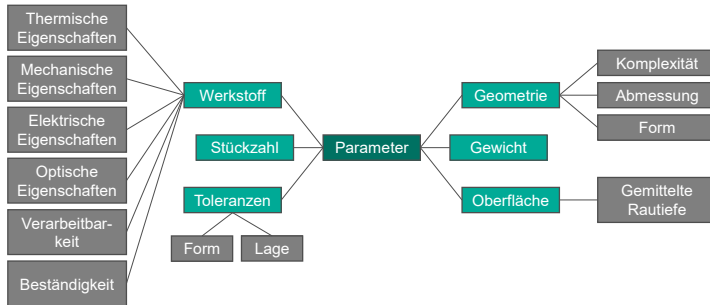


Abbildung 2.2: Parametermindmap möglicher Produktanforderungen (in Anlehnung an Jacob (2021), Fallböhmer (2000) und Schindler (2014))

Für eine fertigungsgerechte Produktgestaltung müssen in Abhängigkeit der Fertigungsmöglichkeiten des Betriebes sowie der generellen fertigungstechnischen Umsetzbarkeit die relevanten Parameter erkannt und entsprechend angepasst werden, um beispielsweise die Kosten zu beeinflussen (Fleischer 2019). Trotz der zahlreichen Gestaltungsrichtlinien basiert fertigungsgerechte Produktgestaltung dennoch auf dem Wissen und der Erfahrung aus verschiedenen Domänen (Fleischer 2019). Dabei kann gerade dieses implizite Wissen in Form von Erfahrung nicht über Gestaltungsrichtlinien formuliert werden (Albers, Deigendesch & Turki 2009; Wartzack, Sauer & Küstner 2017).

Bestehende Ansätze in Literatur und Praxis - wie *Simultaneous Engineering* und *Design for Manufacturing* – befassen sich insbesondere mit der Berücksichtigung von Produktionsaspekten in der Produktentwicklung. Diese Ansätze berücksichtigen jedoch nicht systematisch das Wissen aus vergangenen Produktentwicklungsprozessen, durch dessen Wiederverwendung Entwicklungsaufwand vermieden werden kann. Darüber hinaus werden die sich über Lebenszyklen und Generationen hinweg ändernden Anforderungen an Produkte nicht systematisch mit der Wandelbarkeit des Produktionssystems in Einklang gebracht, sodass dessen Potenzial teilweise ungenutzt bleibt. Im Rahmen des durch Albers et al. (2022) definierten Produkt-Produktions-Codesigns werden die bestehenden Ansätze dahingehend erweitert. Dieser Ansatz beschreibt die hoch vernetzte und parallele Planung, Entwicklung, Realisierung, Betrieb und Außer-

betriebsnahme von Produkten, Produktionssystemen und zugehörigen Geschäftsmodellen. Die Planung berücksichtigt dabei zwingend mehrere Produktgenerationen sowie Produktionssystemevolutionen. Zur Umsetzung sind geeignete Methoden und Werkzeuge notwendig, um die systematische Wiederverwendung von produkt- und produktionssystembezogenem Wissen sowie den Zusammenhängen zu unterstützen. Dabei sollte nicht nur die letzte, sondern alle vorangegangenen sowie prognostizierten zukünftigen Produktgenerationen und die zugehörigen Produktionssysteme in Betracht gezogen werden. Um dieses Wissen konsequent in der Entwicklung und Planung zu nutzen, muss es adäquat formalisiert werden. (Albers et al. 2022) Das im Rahmen dieser Arbeit entwickelte Vorgehen trägt zu diesem Aspekt bei.

2.1.3 Computergestützte Konstruktion mit CAD

Während die anfänglichen Phasen der Produktentwicklung, wie das Klären der Aufgabenstellung sowie das Suchen von Lösungsprinzipien, stark von kreativen Tätigkeiten geprägt sind, steigt der Anteil an Routine- und Wiederholtätigkeiten in den späteren Phasen deutlich an. Zudem wächst der Arbeitsaufwand am Ende eines Produktentwicklungsprojektes exponentiell. Aus diesen Gründen finden insbesondere in den späteren Phasen vermehrt rechnergestützte Werkzeuge, wie CAD, Einsatz. (Ehrlenspiel & Meerkamm 2017)

Neben dem reinen Visualisieren unterstützt der Einsatz von CAD-Systemen zudem das Informieren, Strukturieren und Korrigieren, da in den vorhandenen Datenbanken nach bestehenden CAD-Modellen gesucht werden kann, die korrigiert, wiederverwendet oder geändert werden können. Häufig werden CAD-Daten über ein Produktdaten-Management (PDM) in entsprechenden Datenbanken verwaltet. (Ehrlenspiel & Meerkamm 2017) Die 3D-Produktmodellierung bedeutet für die Konstruktion jedoch häufig einen Mehraufwand. Durch die gleichzeitige und durchgängige Verwendung der 3D-Modelle über den gesamten Produktentstehungsprozess (z.B. für NC-Programmierung oder Finite-Elemente-Analysen) entstehen jedoch ökonomische Vorteile, die den Zusatzaufwand rechtfertigen. (Vajna et al. 2018)

2.1.3.1 Modellierung von CAD-Modellen

Die virtuelle Modellierung von 3D-Objekten basiert auf einfachen geometrischen Grundelementen wie Punkten, Flächen, Linien oder Volumina, die über mathematische Operationen erstellt, verknüpft und manipuliert werden können (Hackenschmidt,

Hautsch & Kleinschrodt 2020). Zur Modellierung von 3D-Modellen im CAD werden als Repräsentationsformen hauptsächlich Kanten-, Flächen- und Volumenmodelle genutzt (Stark 2022; Vajna et al. 2018). Die umfassendste Form ist dabei das Volumenmodell (Stark 2022), weshalb dieses nachfolgend fokussiert wird.

Ein Vertreter der volumenbasierten Modellierungsverfahren ist die CSG-Modellierung (*Constructive Solid Geometry*), die bereits in den ersten CAD-Systemen Anwendung fand. Ein Modell wird damit als Sequenz boolescher Operationen einfacher Grundkörper wie Zylinder oder Quader gebildet. (Hackenschmidt, Hautsch & Kleinschrodt 2020) Eine weitere Möglichkeit zur Volumenmodellierung bieten *B-Rep*-Modelle (*Boundary-Representation*). Diese repräsentieren ein Volumen durch die begrenzenden Flächen und damit die entsprechenden Berandungskanten. In vielen CAD-Systemen wird eine Kombination beider Darstellungsformen genutzt. (Vajna et al. 2018)

Neben den expliziten geometrischen Modellierungsverfahren wird in der Praxis häufig die parametrische Modellierung eingesetzt. Bemaßungen sind hier assoziativ mit der Modellgeometrie verknüpft, sodass bei deren Änderungen die Geometrie neu berechnet und aktualisiert wird. (Hackenschmidt, Hautsch & Kleinschrodt 2020)

Eine weitere Möglichkeit bildet die *Feature*-Modellierung. *Features* sind produktgestaltende Elemente, wie beispielsweise eine Bohrung. Moderne CAD-Systeme nutzen in der Regel eine Kombination aus *Feature*-Modellierung und Parametrisierung und werden daher auch als Hybridmodellierer bezeichnet. (Hackenschmidt, Hautsch & Kleinschrodt 2020)

Ein CAD-Modell entsteht folglich aus einer Basisgeometrie, die durch schrittweises Anfügen weiterer einfacher Teilgeometrien anhand unterschiedlicher *Features* (auch Konstruktionselemente (KEs) genannt) ihre endgültige, durchaus auch komplexe Form erhält. So erzeugte Bauteile können wiederum zu Untergruppen oder Baugruppen kombiniert werden. (Hackenschmidt, Hautsch & Kleinschrodt 2020) Der Zusammenhang ist in Abbildung 2.3 (a) nochmals dargestellt. Der Fokus dieser Arbeit liegt jedoch auf CAD-Modellen auf Bauteil- bzw. Komponentenebene. Die einzelnen Konstruktionsschritte in Form der genutzten KEs sind im Strukturbaum abgebildet (Hackenschmidt, Hautsch & Kleinschrodt 2020; Stark 2022). Dieser enthält folglich die Konstruktionshistorie eines

Bauteils. Ein beispielhafter Strukturbaum aus der CAD-Software PTC Creo® Parametric⁴ ist in Abbildung 2.3 (b) dargestellt.

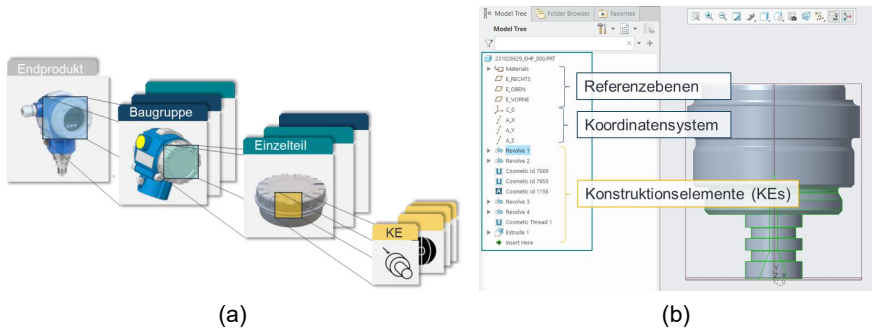


Abbildung 2.3: Aufbau eines CAD-Modells eines Endproduktes bestehend aus mehreren Baugruppen, Einzelteilen und Konstruktionselementen (a) sowie beispielhafter Strukturbaum in PTC Creo® Parametric (b)

Alle Funktionen, Einstellungen und Abläufe, die bei der Erzeugung eines virtuellen Bauteils genutzt werden, sind dabei im Strukturbaum (auch Modellbaum) abgespeichert. Bei der Betrachtung eines fertigen CAD-Modells sind die durchgeführten Konstruktions-schritte in der Regel nicht mehr nachzuvollziehen. Über den Strukturbaum können jedoch die Gedankengänge und das Vorgehen des Konstrukteurs nachvollziehbar und transparent gemacht werden. Dadurch entsteht unter anderem der Vorteil, dass spätere Änderungen einfach durchzuführen sind. (Roj 2016)

Es gibt mehrere Wege, eine gewünschte geometrische Form mit Hilfe der KEs zu erzeugen, wodurch viel Freiraum und kreative Lösungen ermöglicht werden (Hackenschmidt, Hautsch & Kleinschrodt 2020). Die Art und Weise, wie verschiedene KEs aneinandergereiht werden, um ein 3D-Modell zu erstellen, basiert dabei stark auf Expertenwissen, welches letztendlich in den resultierenden Strukturbäumen enthalten ist (Willis et al. 2021). Im Rahmen dieser Arbeit sollen typische Abfolgen von KEs innerhalb des Strukturbaumes und damit verbreitete Vorgehensweisen identifiziert und mit Hilfe von ML-Verfahren nutzbar gemacht werden. Diese typischen Abfolgen werden im Nachfolgenden als Konstruktionsmuster bezeichnet.

⁴ Parametric Technology GmbH (2022), PTC Creo Parametric <https://www.ptc.com/de/products/creo/parametric> (aufgerufen am 27.09.2022)

2.1.3.2 CAD-Dateiformate

Zur Verarbeitung von CAD-Modellen mit ML-Verfahren ist ein Verständnis der unterschiedlichen Dateiformate sowie des Informationsgehalts wichtig.

Grundsätzlich lassen sich nach Roj (2016) herstellerspezifische und herstellerunabhängige Dateiformate unterscheiden. Herstellerspezifische Formate umfassen dabei die meisten Informationen. Beispielsweise sind die Konstruktionsschritte, die zur Erstellung des CAD-Modells durchgeführt wurden, in diesem Dateiformat enthalten. Ein beispielhaftes herstellerspezifisches Dateiformat ist das PRT-Format, das im CAD-Programm PTC Creo® Parametric Verwendung findet⁵. PRT als parametrisches Format speichert die Modelle von Einzelkomponenten in Form der zugrundeliegenden Grundkörper.

Herstellerunabhängige Formate werden als Austauschformate bezeichnet. Sie bieten den Vorteil, dass im Modell hinterlegtes Wissen bewusst nicht abgespeichert wird, beispielsweise wenn CAD-Modelle an Kunden oder Lieferanten weitergegeben werden. Zudem bietet sich der Vorteil, dass so auch CAD-Modelle unterschiedlicher Softwareprogramme verarbeitet werden können. Ein gängiges nicht-parametrisches Austauschformat ist beispielsweise das STEP-Format (*Standard for the External Representation of Product Definition Data*) (Vajna et al. 2018), das durch die ISO 10303-242:2020(E) (ISO 10303-242:2020(E)) genormt und aktualisiert wird. Parametrische Informationen können über dieses Dateiformat nicht übertragen werden. (Vajna et al. 2018).

Ein weiteres herstellerunabhängiges Format ist das Stereolithographie- bzw. *Standard Triangulation Language* (STL)-Format. Die Oberfläche der Geometrie wird dabei über kleine Dreiecke (Polygone), die jeweils über ihre drei Eckpunkte und den Normalenvektor definiert werden, aufgebaut. (Roj 2016) Im Vergleich zum STEP-Format ist das STL-Format lediglich in der Lage, die Basisinformationen des Modells, also die triangulierte Repräsentation, zwischen verschiedenen Systemen zu übertragen. Zusätzliche Informationen, wie Texturen, gehen verloren. Ist jedoch die Oberflächeninformation zur Weiterverarbeitung ausreichend, sind weitere Informationen, die z.B. im STEP-Format enthalten sind, nicht notwendig.

⁵ Parametric Technology GmbH (2022), *Allgemeine Einführung: Dateitypen*
http://support.ptc.com/help/creo/creo_pma/german/index.html#page/fundamentals/fundamentals/fund_two_sub/About_File_Types.html (aufgerufen am 05.05.2022)

2.1.4 Wissensbasiertes Konstruieren

Nach Vajna et al. (2018) kann das Wissen neben Mensch, Maschine, Material, Finanzmittel und Information als sechster Produktionsfaktor angesehen werden. Ca. 60 bis 80 % der Gesamtwertschöpfung kann dabei auf den Faktor Wissen zurückgeführt werden. Allerdings werden nur ca. 20 bis 40 % des Wissens tatsächlich genutzt. Auch in der Produktentwicklung ist die Verarbeitung von bestehendem Wissen eine zentrale Aufgabe. (Vajna et al. 2018)

Bisher werden vom Konstrukteur bestehende Produkte analysiert und die entsprechenden Experten befragt (Fleischer 2019). Da jedoch entsprechendes Fachwissen personenbezogen ist (Fleischer 2019; Deigendesch 2009) und zudem aufgrund mangelnder Dokumentation die entsprechende Transparenz darüber fehlt, welche ähnlichen Produkte bzw. Komponenten es bereits gibt (Lupinetti et al. 2019), wird vorhandenes Wissen bisher nicht nachhaltig genutzt. Darüber hinaus ist der Prozess, entsprechendes Wissen bezüglich verschiedener *Design for X* Richtlinien zu akquirieren, sehr zeitintensiv und somit wenig wertschöpfend (Kratzer et al. 2012).

Im Bereich der wissensbasierten Konstruktion (engl. *Knowledge based Engineering* (KBE)) wird seit Jahren versucht, durch wissensbasierte Systeme bei Routinetätigkeiten zu entlasten, sodass eine Fokussierung auf kreative Tätigkeiten ermöglicht wird (Stokes 2001). Nach La Rocca (2012) werden in solchen Systemen Verfahren der KI, der CAD-Methodik und der Softwareentwicklung miteinander vereint.

In der Literatur (Breitsprecher & Wartzack 2012; Wartzack 2001) wird im Bereich des KBE die direkte, indirekte und automatische Wissensaneignung unterschieden. Bei der indirekten Wissensakquisition wird das Expertenwissen mit Hilfe eines Wissensingenieurs anhand verschiedener Methoden (z.B. Fragebögen, Interviews) extrahiert und abgebildet. Die Herausforderung dabei ist, dass insbesondere Erfahrungswissen nur sehr schwer formalisierbar ist. Im Gegensatz dazu kommuniziert der Experte bei der direkten Wissensakquisition direkt mit einer intelligenten Akquisitionskomponente und muss sein Fachwissen selbstständig definieren. (Vajna et al. 2018)

Basis für die direkte und indirekte Wissensakquisition sind folglich Experten, allerdings sind diese Methoden sehr zeit- und kostenintensiv, da sowohl die Experten als Wissensträger, als auch Wissensingenieure notwendig sind (Breitsprecher & Wartzack 2012). Grundlage für die automatische Wissensaneignung sind dagegen Daten, aus denen beispielsweise mit Verfahren der KI, speziell des MLs, auch bisher unbekanntes

implizites Wissen gewonnen werden kann (Düsing 2006). Solche lernenden Systeme sind in der Lage, das Wissen aus Fallbeispielen zu abstrahieren, beispielsweise durch die Bildung von Analogien oder durch Generalisierung. (Vajna et al. 2018)

Nach Vajna et al. (2018) wird die automatische Wissensakquisition sehr intensiv erforscht, da insbesondere die Lernfähigkeit der Systeme noch großes Potenzial eröffnet. Vor allem die Anwendung von künstlichen neuronalen Netzen, als Vertreter von ML-Verfahren, wird als vielversprechend angesehen, was auch Fokus der vorliegenden Arbeit ist.

Vorgehensmodelle, über die im Allgemeinen computergestützt Wissen aus Daten extrahiert werden kann, sind beispielsweise *Knowledge Discovery in Databases* (KDD) (Fayyad, Piatetsky-Shapiro & Smyth 1996) oder der *Cross Industry Standard Process for Data Mining* (CRISP-DM) (Chapman et al. 2000). Ziel ist es, vorhandenes Wissen aus Daten, welche aufgrund der Größe und Komplexität nicht mehr ohne Computerunterstützung interpretierbar sind, zu identifizieren und nutzbar zu machen. (Breitsprecher et al. 2015)

Der CRISP-DM Zyklus findet häufig im industriellen Kontext Anwendung (Breitsprecher et al. 2015, S. 744). Der grundsätzliche Ablauf des CRISP-DM ist in Abbildung 2.4 dargestellt. Die zeitliche Abfolge eines *Data-Mining*-Projektes wird dabei in sechs Phasen eingeteilt. In der ersten Phase des Geschäftsverständnisses (engl. *Business Understanding*) werden zunächst die Projektziele sowie die Anforderungen spezifiziert. In der darauf folgenden Phase des Datenverständnisses (engl. *Data Understanding*) werden die vorhandenen Daten gesichtet, selektiert und zusammengeführt. Ein wesentlicher Schritt ist die nachfolgende Vorbereitung der Daten (engl. *Data Preparation*). Hier werden die Daten für die spätere Anwendung von *Data-Mining*-Verfahren entsprechend vorverarbeitet, wie beispielsweise bereinigt und transformiert. Basierend auf diesen Daten können schließlich in der Phase der Modellierung (engl. *Modelling*) verschiedene *Data-Mining*-Verfahren eingesetzt werden, um die definierte Problemstellung zu lösen. Da die verschiedenen Techniken unterschiedliche Anforderungen an das Datenformat haben, erfolgt häufig ein erneutes Durchlaufen der *Data Preparation*. In der *Evaluation* werden die realisierten Modelle letztendlich auf den Daten angewandt und die Ergebnisse ausgewertet. Insbesondere wird geprüft, ob die gesetzten Ziele und Anforderungen mit dem Modell ausreichend gut erfüllt werden. In der letzten Phase der Bereitstellung (engl. *Deployment*) werden die Ergebnisse schließlich so aufbereitet, dass sie durch den Endanwender genutzt werden können. (Chapman et al. 2000)

Da die Anwendung von ML jedoch Besonderheiten aufweist, wird in Studer et al. (2021) der CRISP-DM speziell für Projekte in diesem Bereich erweitert. Im Gegensatz zu CRISP-DM werden im sogenannten CRISP-ML(Q) die Phasen des *Business* und *Data Understanding* zusammengelegt, da sehr häufig die gesetzten Ziele in Abhängigkeit der Datenlage angepasst werden müssen (Studer et al. 2021). Diese Besonderheiten treffen auch auf das im Rahmen dieser Arbeit vorgestellte Vorgehen zu.

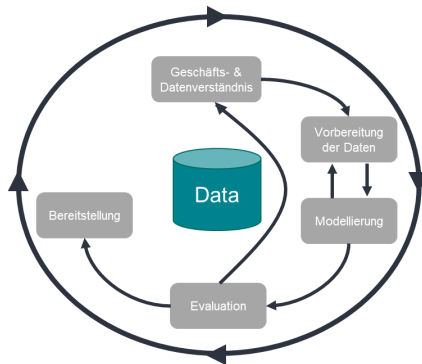


Abbildung 2.4: Ablauf des CRISP-DM (basierend auf Chapman et al. (2000))

2.2 Maschinelles Lernen

ML ist ein Teilgebiet der KI und umfasst Verfahren, die selbstständig Wissen aus Daten erlernen können, indem sie entsprechende Muster extrahieren. Die Funktionalität eines ML-Algorithmus ist stark von der Repräsentation der Daten abhängig, auf deren Basis Entscheidungen getroffen werden. Die Repräsentation besteht dabei aus den relevanten Merkmalen (engl. *Features*), über die die Daten beschrieben werden können. Gerade aber die Definition dieser *Features* ist ausschlaggebend und häufig sehr schwierig. Eine Möglichkeit besteht darin, die ML-Algorithmen selbst nicht nur die Zuordnung von der Repräsentation auf einen gewünschten Output erlernen zu lassen, sondern auch die Repräsentation selbst. Diese selbst erlernten Repräsentationen liefern in der Regel deutlich bessere Ergebnisse als individuell für die gegebene Lernaufgabe angepasste Repräsentationen. Zudem ist der Aufwand, manuell entsprechende *Features* für eine komplexe Aufgabe zu konzipieren, extrem hoch. Eine relevante Methode aus dem Bereich des sogenannten *Representation Learning* sind *Autoencoder*. Häufig ist es jedoch schwer, abstrakte *Features* aus Daten zu erlernen, da diese einer Vielzahl an Varia-

tionsfaktoren unterliegen (z.B. ein Akzent bei Sprache), weshalb ein nahezu menschliches Verständnis der Daten erforderlich ist. Dieses Problem wird durch Ansätze des *Deep Learning* (DL) gelöst, da damit auch für komplexe Daten sehr einfache Repräsentationen erlernt werden können. (Goodfellow, Bengio & Courville 2016)

DL ist folglich ein Teilgebiet des ML, der Zusammenhang wird nochmals in Abbildung 2.5 dargestellt. Ein grundlegendes DL-Modell ist dabei das *Multilayer Perceptron* (MLP), das häufig auch als *Feed-Forward-Netzwerk* bezeichnet wird (siehe Kapitel 2.2.1). (Goodfellow, Bengio & Courville 2016) Neuronale Netze, die speziell für die Verarbeitung von sequenziellen Daten sind, sogenannte *Recurrent Neural Networks* (RNNs), werden in Kapitel 2.2.4 erläutert.

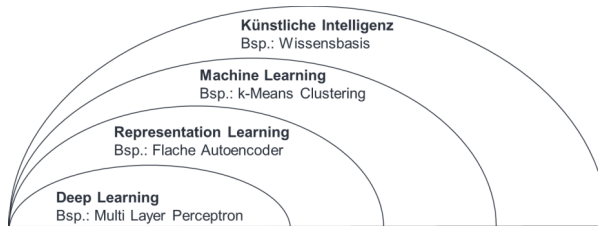


Abbildung 2.5: Übersicht über die Bereiche der Künstlichen Intelligenz mit beispielhaften Technologien (in Anlehnung an Goodfellow, Bengio & Courville (2016))

ML-Algorithmen können in die Teilbereiche *supervised*, *unsupervised* und *reinforcement learning* unterteilt werden. Eine klassische *unsupervised* Lernaufgabe ist es, die beste Repräsentation für die gegebenen Daten zu finden. Neben *Autoencodern* (Kapitel 2.2.2) zählen hierzu auch einfache Clusteralgorithmen (Kapitel 2.2.3). (Goodfellow, Bengio & Courville 2016)

In Kapitel 2.2.5 werden die Grundlagen zu DL auf 3D-Daten dargelegt, da im Rahmen dieser Arbeit eine Anwendung der Verfahren auf CAD-Daten erfolgt. Im Vergleich zum 2D-Bereich gibt es einige Herausforderungen, jedoch ist das Interesse daran, insbesondere im Bereich CAD, in den letzten Jahren stark gestiegen (Yoo et al. 2021).

2.2.1 Feed Forward Neuronale Netze

Ein neuronales Netz ist nach Russell & Norvig (2022) im Allgemeinen aus einer Menge an verknüpften Neuronen (auch Knoten oder Einheiten) aufgebaut, die in Schichten (engl. *Layers*) organisiert sind. Jedes Neuron verfügt dabei über mindestens jeweils einen Input und Output sowie eine nichtlineare Aktivierungsfunktion. Die Neuronen sind

untereinander über gerichtete Verbindungen verknüpft, denen jeweils eine bestimmte numerische Gewichtung zugeordnet ist. Für die Verarbeitung einer Eingabe a_i berechnet jedes Neuron zunächst deren gewichtete Summe und wendet anschließend darauf eine nichtlineare Aktivierungsfunktion ϕ_j an. Mathematisch kann dieser Zusammenhang nach Russell & Norvig (2022) wie folgt formuliert werden:

$$a_j = \phi_j \sum_i (\vartheta_{ij} a_i) \quad 2-1$$

Dabei ist a_j der Output von Einheit j und ϑ_{ij} die Gewichtung für die Verbindung zwischen Einheit i und j . Die damit generierte Ausgabe dient schließlich als Eingabe für die nachfolgenden Neuronen. Gängige Aktivierungsfunktionen sind beispielsweise die Sigmoid-Funktion, ReLU-Funktion (*Rectified Linear Unit*), Tangens Hyperbolicus-Funktion oder die Softmax-Funktion (Aggarwal 2021). Die Wahl der Aktivierungsfunktion ist dabei in Abhängigkeit des Anwendungsfalls zu treffen (Aggarwal 2021).

Der Begriff *Feed-forward* resultiert aus der Tatsache, dass der Informationsfluss lediglich vom Input in Richtung des Outputs verläuft und es keine Rückkopplungen gibt. Üblicherweise setzt sich ein *Feed-Forward*-Netz aus jeweils einer Eingabeschicht (engl. *Input Layer*), einer Ausgabeschicht (engl. *Output Layer*) sowie einer oder mehreren verborgenen Schichten (engl. *Hidden Layer*) dazwischen zusammen. Besteht ein solches neuronales Netz aus vielen verborgenen Schichten, wird auch von *Deep Feed Forward Neural Networks* gesprochen. Der grundlegende Aufbau eines mehrschichtigen *Feed Forward* Netzwerks mit zwei verborgenen Schichten wird in Abbildung 2.6 dargestellt. Die Anzahl der verwendeten Schichten (auch als Tiefe bezeichnet), die Menge an Neuronen pro Schicht (auch als Breite bezeichnet) sowie die Verknüpfung der Schichten wird auch als die Architektur des neuronalen Netzes bezeichnet. (Goodfellow, Bengio & Courville 2016) Sofern jede Einheit einer Schicht mit jeder Einheit der nächsten Schicht verknüpft ist, wird auch von einem vollständig verknüpften (engl. *fully connected*) Neuronales Netzwerk gesprochen (Russell & Norvig 2022).

Mathematisch formuliert ist es nach Goodfellow, Bengio & Courville (2016) das Ziel eines neuronalen Netzes, eine Funktion f^* zu approximieren, die eine Eingabe x einer bestimmten Ausgabe y zuordnet:

$$y = f^*(x) \quad 2-2$$

Über ein neuronales Netz wird folglich eine Zuordnung definiert, über die diese Funktion durch das Erlernen der Gewichtungsmatrix θ , bestehend aus den einzelnen Gewichtungen ϑ_{ij} , approximiert wird:

$$y = f(x, \theta) \quad 2-3$$

Da ein neuronales Netz üblicherweise aus mehreren Schichten besteht und jeder Schicht, mit Ausnahme der Input-Schicht (Aggarwal 2021), eine Funktion zugeordnet wird, wird der Input x über eine Verkettung mehrerer Funktionen einem Output y zugeordnet.

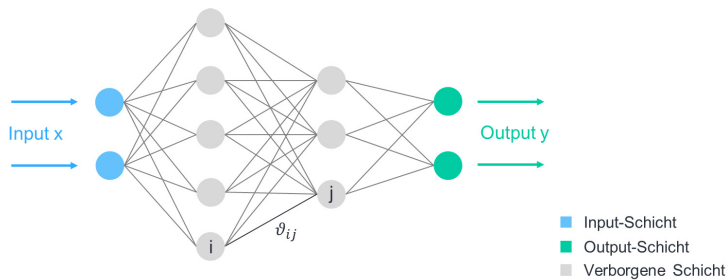


Abbildung 2.6: Grundsätzlicher Aufbau eines Feed-Forward-Netztes (basierend auf Aggarwal (2021))

Ziel des Lernens ist es nun, die Gewichtungen θ des neuronalen Netztes so anzupassen, dass der Fehler zwischen gewolltem Output y und dem durch das neuronale Netz geschätzten Output \hat{y} für einen gegebenen Input minimal wird. Zur Berechnung dieses Fehlers werden Verlust-Funktionen (engl. *Loss function*) genutzt, die je nach Anwendungsfall gewählt werden (Aggarwal 2021). Für eine lineare Regression wird beispielsweise der quadratische Fehler herangezogen (Aggarwal 2021):

$$L = (y - \hat{y})^2 \quad 2-4$$

Der berechnete Fehler beschreibt die Differenz zwischen der durch das neuronale Netz geschätzten Ausgabe und der gewollten Ausgabe für die Output-Schicht. Für die verborgenen Schichten sind dagegen die gewollten Werte unbekannt. Über die sogenannte *Backpropagation* wird der Fehler der Output-Schicht daher an die verborgenen Schichten zurückgegeben. Grundidee ist dabei, dass ein Teil des Fehlers jedes Ausgabeneurons auf ein damit verbundenes Neuron einer verborgenen Schicht zurückzuführen ist. Auf diese Weise kann letztendlich der Gradient in Abhängigkeit der Gewichtsparameter berechnet werden, auf dessen Basis dann das Lernen und damit das

Anpassen der Netzgewichte durchgeführt werden kann. (Goodfellow, Bengio & Courville 2016) Hierfür kann beispielsweise das *Stochastic Gradient Descent* Verfahren herangezogen werden (Goodfellow, Bengio & Courville 2016). Grundgedanke des Verfahrens ist, die Verlustfunktion zu minimieren, indem man in die Richtung des negativen Gradienten abwärts geht. Die Schrittweite wird dabei durch die Lernrate λ definiert. Beim stochastischen Gradientenabstiegsverfahren wird davon ausgegangen, dass der Gradient als Erwartungswert betrachtet werden kann und somit über eine kleine Teilmenge der Trainingsdaten, sogenannte *Mini-Batches*, geschätzt werden kann. Zur Berechnung des Gradienten und damit zur Anpassung der Netzgewichte ist folglich nur eine kleine Teilmenge der Trainingsdaten notwendig. (Goodfellow, Bengio & Courville 2016) Eine detaillierte Beschreibung des Verfahrens ist beispielsweise in Goodfellow, Bengio & Courville (2016) zu finden.

Im Rahmen einer sogenannten Epoche werden alle Trainingsdaten einmal zur Netzanpassung genutzt. Im Normalfall werden mehrere Trainingsepochen durchgeführt. Die Güte der erlernten Approximation kann schließlich anhand von (ungesehenen) Testdaten validiert werden. Bei geringen Datenmengen wird häufig die *k-fold Cross Validation* herangezogen. Der Datensatz wird dabei in k etwa gleich große Mengen eingeteilt, von denen jeweils $k-1$ zum Training und die übrige zum Testen herangezogen wird. Insgesamt werden somit k Approximationen auf jeweils unterschiedlichen Daten erlernt, deren Bewertung üblicherweise anhand der durchschnittlichen Performance erfolgt. Gängige Werte für k sind 5 oder 10 (Kuhn & Johnson 2016). Stellhebel, um die erlernte Approximation zu verbessern, werden als Hyperparameter bezeichnet. Diese können durch Ausprobieren und auf Basis von Erfahrungswerten optimiert werden. Sofern nur wenige Parameter vorliegen, können über eine sogenannte *Grid Search* alle möglichen Kombinationen erprobt werden. Sind die erlernten Netzgewichte und damit die Funktionsapproximation zu sehr auf den zugrunde gelegten Datensatz angepasst, sodass das neuronale Netz auf ungesehenen Daten nur schlechte Vorhersagen treffen kann, so wird auch von *Overfitting* gesprochen. Können dagegen die in den Daten enthaltenen Muster nicht ausreichend abgebildet werden, wird dies als *Underfitting* bezeichnet. (Russell & Norvig 2022)

2.2.2 Autoencoder

Autoencoder zählen zu den Verfahren des *Representation Learning* (Goodfellow, Bengio & Courville 2016) und können für eine Vielzahl von *unsupervised* Lernaufgaben

eingesetzt werden (Aggarwal 2021). Nachfolgend wird zunächst der grundsätzliche Aufbau eines *Autoencoders* beschrieben. Durch den *Autoencoder* kann schließlich eine komprimierte Darstellung eines Inputs im sogenannten latenten Raum erzeugt werden. In Kapitel 2.2.2.2 werden entsprechende Analysen aufgezeigt, um die Eigenschaften dieses latenten Raumes zu untersuchen.

2.2.2.1 Grundlegender Aufbau

Ein *Autoencoder* ist ein neuronales Netz, das genutzt wird, um von einem gegebenen Input eine interne Repräsentation zu erlernen, auf deren Basis eine Rekonstruktion des ursprünglichen Inputs ausgegeben werden kann. Verfügt der *Autoencoder* dabei über mehrere Schichten, wird auch von einem *Deep Autoencoder* gesprochen (Aggarwal 2021). Der *Autoencoder* besteht dabei aus zwei Komponenten, dem *Encoder* und dem *Decoder*. (Goodfellow, Bengio & Courville 2016) Der grundsätzliche Aufbau eines *Autoencoders* sowie das grundlegende Schema sind in Abbildung 2.7 dargestellt.

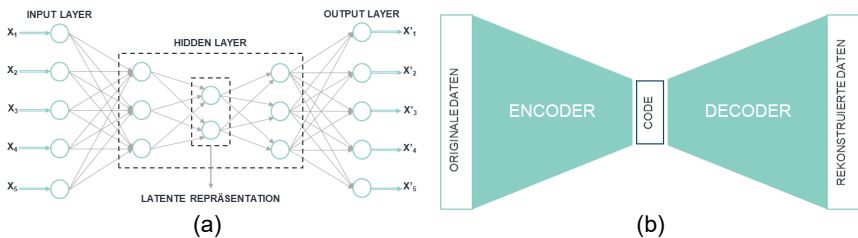


Abbildung 2.7: Aufbau eines Autoencoders mit drei verborgenen Schichten (a) sowie die allgemeine schematische Darstellung (b) (basierend auf Aggarwal (2021))

Üblicherweise sind sowohl *Encoder* als auch *Decoder* mehrschichtige neuronale Netze. Der *Encoder* komprimiert dabei den Input x in eine reduzierte Repräsentation. Die Dimensionalität dieser Engstelle (engl. *Bottleneck*) ist im Vergleich zur Eingabe-Schicht deutlich geringer. Auf Basis dieser komprimierten Darstellung, auch als latente Repräsentation (Achlioptas et al. 2018), latenter Vektor (Peste, Malagó & Sârbu 2017) oder *Global Feature Vector* (Sarmad, Lee & Kim 2019) bezeichnet, versucht der *Decoder*, den ursprünglichen Input zu rekonstruieren. Folglich stimmen die Dimensionalität der Ausgabe und der Eingabe überein. Aufgrund der komprimierten Darstellung, die auch als *Code* bezeichnet wird, kann der *Decoder* die Eingabe nicht einfach kopieren. Vielmehr müssen die relevanten Eigenschaften aus dem Input in diesem *Code* reduziert dargestellt werden. Ziel ist es folglich, diejenigen Merkmale zu extrahieren, auf deren Basis eine möglichst gute Rekonstruktion erzeugt werden kann. (Goodfellow, Bengio &

Courville 2016) Durch *Deep Autoencoder* können dabei auch komplexe Daten in eine anspruchsvolle Repräsentation überführt werden (Aggarwal 2021). Im Rahmen dieser Arbeit werden ausschließlich *Deep Autoencoder* genutzt, die nachfolgend vereinfacht als *Autoencoder* bezeichnet werden.

Der Verlust wird über die Abweichung von Eingabe und rekonstruierter Ausgabe errechnet, beispielsweise über den mittleren quadratischen Fehler (Goodfellow, Bengio & Courville 2016; Aggarwal 2021). Während des Trainings wird die Verlustfunktion minimiert und eine Anpassung der Netzgewichte des *Autoencoders* erfolgt dabei, wie für *Feed-Forward*-Netze, über eine Form des Gradientenabstiegsverfahrens und *Backpropagation* (Goodfellow, Bengio & Courville 2016).

2.2.2.2 Analysen und Eigenschaften des latenten Raumes

Die Erklärbarkeit und Interpretierbarkeit von DL-Verfahren ist ein aktives Forschungsgebiet (Samek & Müller 2019). Um dennoch ein Verständnis der extrahierten Eigenschaften und somit des latenten Raumes zu erhalten (Spinner et al. 2018), können die hochdimensionalen Vektoren mit Hilfe des von van der Maaten & Hinton (2008) entwickelten Verfahrens *t-distributed stochastic Neighbor Embedding* (t-SNE) visualisiert werden. Mit Hilfe von statistischen Methoden erfolgt eine Dimensionsreduzierung, wodurch es möglich wird, hochdimensionale Daten in einer zwei- oder dreidimensionalen Karte darzustellen. Dabei können lokale Strukturen der Daten aufrechterhalten und zeitlich globale Strukturen, wie beispielsweise Cluster, aufgedeckt werden. Laut den Autoren ist das Verfahren besonders für die Visualisierung von Datenrepräsentationen, die durch *Autoencoder* erzeugt wurden, geeignet. Es ist anzumerken, dass es dennoch unmöglich ist, in zwei oder drei Dimensionen den ursprünglichen Informationsgehalt der Daten vollständig zu visualisieren. Zudem ist das erhaltene Ergebnis stark abhängig von der zugrundeliegenden Parameterwahl.

Nach Goodfellow, Bengio & Courville (2016) ist eine erlernte Repräsentation gut, sofern sie die darauf aufbauende Lernaufgabe vereinfacht. Folglich kann neben der rein visuellen Überprüfung, basierend auf den latenten Vektoren, zusätzlich eine Clusteranalyse oder, sofern Klassen-Label vorhanden sind, eine Klassifikation vorgenommen werden (Aljalbout et al. 2018; Hassani & Haley 2019; Achlioptas et al. 2018; Wu et al. 2016; Sun et al. 2021). Dadurch kann die Effizienz der erlernten Repräsentation und damit der extrahierten Eigenschaften gezeigt werden (Achlioptas et al. 2018; Yang et al.

2018). In Achlioptas et al. (2018) werden neben der Klassifizierung noch weitere Experimente aufgezeigt, um die Repräsentativität des latenten Raumes zu zeigen. Diese werden beispielsweise auch in Yang et al. (2018) herangezogen. Eine Möglichkeit stellt die Interpolation dar (Yoo et al. 2021; Park et al. 2019). Hierfür werden Punkte auf dem Differenzvektor zweier latenter Vektoren betrachtet und untersucht, ob die entsprechenden Vektoren sinnvolle Objekte mit Eigenschaften beider Ausgangsobjekte darstellen, sodass das eine Objekt sukzessive in das andere übergeht. Eine weitere Möglichkeit, die ursprünglich aus der Sprachverarbeitung stammt (Tasse & Dodgson 2016), bieten algebraische Operationen auf den latenten Vektoren, wie Addition und Subtraktion, um deren semantische Bedeutung zu untersuchen. Unter Semantik wird dabei verstanden, ob die latenten Vektoren bestimmte semantische Merkmale der zugrundeliegenden Objekte (z.B. die Anzahl der Tischbeine) abbilden.

Eine problematische Eigenschaft des latenten Raumes ist es, dass Distanzen keine physikalischen Einheiten zugrunde liegen, wodurch sie nur sehr schwer interpretierbar sind (Arvanitidis, Hansen & Hauberg 2018). Nach Frenzel, Teleaga & Ushio (2019) gibt es im Allgemeinen keine Garantie dafür, dass Distanzen im latenten Raum als sinnvolles Maß herangezogen werden können. Viele auf erlernten latenten Repräsentationen basierende Anwendungen stützen sich dennoch auf die Distanzberechnung im latenten Raum. Sehr häufig wird dabei die euklidische Distanz herangezogen, obwohl nicht allgemeingültig davon ausgegangen werden kann, dass der latente Raum ein euklidischer Raum ist. Achlioptas et al. (2018) zeigen anhand einer Reihe von Experimenten auf den latenten Vektoren, wie z.B. Interpolationen oder algebraischen Operationen, die euklidische Natur des durch ihre *Autoencoder*-Architektur erlernten latenten Raumes. Auch in Pang, Li & Tian (2021), Yang et al. (2018) und Wu et al. (2016) wird die euklidische Distanz herangezogen. Welches Distanzmaß für den latenten Raum jedoch sinnvoll ist, hängt stark vom jeweiligen Anwendungsfall ab (Frenzel, Teleaga & Ushio 2019). Da im Rahmen dieser Arbeit auf der *Autoencoder*-Architektur von Yang et al. (2018) aufgebaut wird, wird ebenfalls die euklidische Distanz herangezogen. Die euklidische Distanz d_2 zwischen zwei latenten Vektoren \vec{x}_1 und \vec{x}_2 der Länge M berechnet sich wie folgt:

$$d_2(\vec{x}_1, \vec{x}_2) = \|\vec{x}_1 - \vec{x}_2\|_2 = \sqrt{\sum_{i=1}^M (x_{1,i} - x_{2,i})^2} \quad 2-5$$

2.2.3 Clusteranalyse

Die Clusteranalyse, die zum Bereich des *unsupervised Learning* zählt, verfolgt im Allgemeinen das Ziel, einen Datensatz in mehrere homogene Cluster einzuteilen. Dabei haben Datenpunkte eines Clusters ähnliche Eigenschaften. (Pham & Afify 2007) Als Ähnlichkeit wird dabei in der Regel definiert, wie nahe die einzelnen Datenpunkte räumlich beieinander liegen. Hierfür werden Distanzfunktionen herangezogen. (Han, Kamber & Pei 2012) Die vorhandenen Clusteralgorithmen können in vier Gruppen eingeteilt werden: partitionierende, hierarchische, dichte-basierte und gitterbasierte Verfahren. (Pham & Afify 2007; Han, Kamber & Pei 2012)

Im Rahmen dieser Arbeit wird einerseits eine Clusteranalyse auf latenten Repräsentationen durchgeführt. Da hierfür insbesondere *k-Means* als Vertreter der partitionierenden Clusteralgorithmen genutzt wird (Aljalbout et al. 2018; Yang et al. 2017), wird auch in der später beschriebenen Methodik dieses Verfahren gewählt. Andererseits wird ein Verfahren benötigt, über das eine Menge von Datenpunkten in Bereiche größerer Dichte eingeteilt wird, weshalb sich dichte-basierte Verfahren eignen. Als wesentlicher Vertreter dieser Kategorie wird hierfür *Density-Based Spatial Clustering of Applications with Noise* (DBSCAN) (Ester et al. 1996) genutzt, das den großen Vorteil besitzt, auch Ausreißer zu detektieren (Bickel et al. 2021). Nachfolgend werden die für diese Arbeit relevanten Verfahren *k-Means* und DBSCAN näher erläutert sowie abschließend Methoden zur Evaluierung von Clusterlösungen aufgezeigt.

2.2.3.1 Clusteranalyse mit k-Means

k-Means ist aufgrund der einfachen Umsetzung der am häufigsten verwendete Vertreter der partitionierenden Verfahren (Han, Kamber & Pei 2012; Steinley & Brusco 2007). Nachfolgend wird der Ablauf in Anlehnung an Steinley & Brusco (2007) beschrieben.

Ziel des Algorithmus ist es, die Objekte $o=1, \dots, O$ mit Merkmalen $m=1, \dots, M$ in eine vorab definierte Anzahl von K Klassen einzuteilen. Die Anzahl der zu bestimmenden Cluster $k=1, \dots, K$ muss dabei vorab definiert werden (Han, Kamber & Pei 2012). S_{z_k} ist dabei die Menge an Objekten, die dem k -ten Cluster angehören. Jedes Cluster wird dabei über ein möglichst repräsentatives Clusterzentrum z_k charakterisiert, das einen Punkt im M -dimensionalen Raum, der durch den Mittelwert der Merkmale $x_{o,m}$ aller Objekte innerhalb des Clusters gebildet wird, darstellt. Der Wert dieses Zentrums z_k , dargestellt über

dessen Merkmalsvektor $\overrightarrow{x_{z,k}}$, wird folglich für das m -te Merkmal $x_{z,k,m}$ in $\overrightarrow{x_{z,k}}$ wie folgt berechnet:

$$x_{z,k,m} = \frac{1}{|S_{z_k}|} \sum_{i \in S_{z_k}} x_{o,m} \quad 2-6$$

Um nun die O Objekte in K Zentren zu partitionieren, wird die nachfolgende Summe der quadrierten euklidischen Distanz zwischen jedem Datenpunkt o und seinem nächstgelegenen Clusterzentrum z_k minimiert:

$$d_2(\overrightarrow{x_o}, \overrightarrow{x_{z,k}}) = \sum_{i=1}^M (x_{o,i} - x_{z,k,i})^2 \quad 2-7$$

Zu Beginn werden die Clusterzentren zufällig initialisiert. Anschließend werden alle Datenpunkte dem Clusterzentrum zugeordnet, zu dem die Distanz gemäß Formel 2-7 minimal ist. In der nächsten Iteration werden die initialen Clusterzentren gemäß Formel 2-6 aktualisiert und alle Objekte erneut auf Basis der berechneten Distanzen einem Cluster zugeordnet. Diese Restrukturierung wird so lange iterativ durchlaufen, bis die resultierenden Cluster unverändert bleiben.

Zur Bestimmung der optimalen Cluster-Anzahl wird beispielsweise die *Elbow*-Methode (Thorndike 1953) oder der *Davies Bouldin Score* (Davies & Bouldin 1979) genutzt.

2.2.3.2 DBSCAN

Grundlage zur Bildung von Clustern ist die Dichte eines Objektes, die über die Anzahl der nahe liegenden Objekte gemessen werden kann. Zur Bestimmung dieser Dichte verfügt das Verfahren *Density-Based Spatial Clustering of Applications with Noise* (DBSCAN) (Ester et al. 1996) über zwei wesentliche Parameter, ε_{DB} und *MinPts*. Um für ein gegebenes Objekt diese Nachbarschaft zu quantifizieren, wird der Bereich mit einem bestimmten Radius ε_{DB} um ein Objekt betrachtet. Der Parameter *MinPts* legt schließlich einen Schwellenwert fest, wie viele Objekte (inkl. dem Objekt selbst) innerhalb dieser ε_{DB} -Nachbarschaft mindestens liegen müssen, sodass der Bereich als dicht angesehen werden kann. Hat ein Objekt ausreichend viele Nachbarn in seiner ε_{DB} -Umgebung, so wird es als Kernpunkt bezeichnet. (Han, Kamber & Pei 2012) Objekte, die innerhalb der ε_{DB} -Umgebung eines Kernpunktes liegen, jedoch selbst in ihrer ε_{DB} -Nachbarschaft weniger als *MinPts* Objekte aufweisen, werden als Randpunkte bezeichnet (Aggarwal 2015). Trifft keine der oben genannten Bedingungen zu, handelt es sich um

einen Rauschpunkt (Aggarwal 2015). In Abbildung 2.8 wird die Unterscheidung nochmals verdeutlicht. Der Parameter $MinPts$ wird hier beispielhaft auf 7 gesetzt. In der ε_{DB} -Umgebung des Punktes A befinden sich insgesamt 9 Punkte, weshalb dieser als Kernpunkt definiert wird. Punkt B liegt zwar in der ε_{DB} -Umgebung von A, weist jedoch nicht ausreichend Nachbarn im Bereich von ε_{DB} auf, weshalb B als Randpunkt deklariert wird. Punkt C liegt weder in der ε_{DB} -Umgebung eines Kernpunktes, noch liegen ausreichend Punkte innerhalb des ε_{DB} -Bereichs, weshalb dieser als Rauschpunkt zählt.

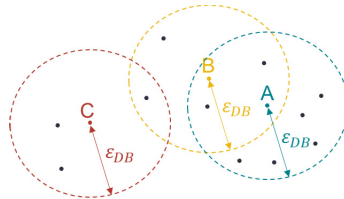


Abbildung 2.8: Beispielhafte Darstellung eines Kernpunktes (A), Randpunktes (B) und Rauschpunktes (C)

Startpunkt des Algorithmus ist ein beliebiges Objekt, für das überprüft wird, ob innerhalb der ε_{DB} -Nachbarschaft mindestens $MinPts$ Objekte liegen. Ist das nicht der Fall, wird das Objekt als Rauschpunkt markiert. Andernfalls werden alle Objekte innerhalb der ε_{DB} -Nachbarschaft zu einem Cluster zusammengefasst und in die Liste der relevanten Objekte für dieses Cluster übernommen. Für alle Objekte aus dieser Liste wird iterativ überprüft, ob in deren ε_{DB} -Umgebung mindestens $MinPts$ Objekte liegen. Falls ja, werden diese der Liste relevanter Objekte angefügt. Das Cluster wird schließlich so lange erweitert, bis die Liste relevanter Objekte leer ist. Um das nächste Cluster zu finden, wird zufällig ein noch nicht betrachtetes Objekt ausgewählt und der beschriebene Clusterbildungsprozess durchgeführt. (Han, Kamber & Pei 2012)

Ein Vorteil des Verfahrens ist es, dass durch die Betrachtung der lokalen Dichte Cluster von beliebiger Form entdeckt werden können. Im Gegensatz zu k -Means muss die Anzahl der Cluster nicht vorab definiert werden. (Aggarwal 2015)

2.2.3.3 Evaluation der Clusterbildung

Zur Bewertung der Qualität der Clusterlösung können grundsätzlich zwischen extrinsischen (auch externen) und intrinsischen (auch internen) Methoden unterschieden werden. Extrinsische Methoden setzen voraus, dass die optimale Clusterlösung, die

Ground Truth (z.B. in Form von Klassen-Labels), bekannt ist. Folglich werden die erhaltenen Lösungen mit dieser optimalen Lösung verglichen. Sofern keine *Ground Truth* vorhanden ist, werden intrinsische Methoden eingesetzt. Diese bewerten, wie gut die Cluster voneinander separiert werden. (Han, Kamber & Pei 2012)

Ein Vertreter der extrinsischen Methoden ist das Entropie-basierte V-Maß (engl. *V-Measure* (VM)) nach Rosenberg & Hirschberg (2007), dessen Berechnung nachfolgend nach Rosenberg & Hirschberg (2007) beschrieben wird.

Das VM setzt sich aus den zwei Kriterien *Homogeneity* und *Completeness* zusammen. Der *Homogeneity Score* (HS) beschreibt dabei die Verteilung der Klassen innerhalb eines Clusters und folglich die Entropie. Er wird wie folgt definiert:

$$HS = 1 - \frac{H(c|k)}{H(c)} \quad 2-8$$

Gemäß Formel 2-8 berechnet sich der HS als Quotient aus der bedingten Entropie der Klassen c bei gegebener Zuordnung der Cluster k , $H(c|k)$, und der Entropie der Gesamtheit der Klassen $H(c)$. Deren genaue Definition wird nachfolgend formuliert. Die Variablen o_c , o_k bzw. $o_{c,k}$ bestimmen die Anzahl an Objekten mit $o = 1, \dots, O$ in Klasse c , Cluster k bzw. übereinstimmend in c und k .

$$H(c|k) = - \sum_{c=1}^C \sum_{k=1}^K \frac{o_{c,k}}{O} \cdot \log\left(\frac{o_{c,k}}{o_k}\right) \quad 2-9$$

$$H(c) = - \sum_{c=1}^C \frac{o_c}{O} \cdot \log\left(\frac{o_c}{O}\right) \quad 2-10$$

Der HS liegt im Bereich $[0,1]$ und weist maximal einen Wert von 1 auf, sofern nur Objekte einer Klasse in einem Cluster anzutreffen sind.

Der *Completeness Score* (CS) bewertet die Verteilung der Cluster-Zuordnungen innerhalb einer Klasse. Er ist über die bedingte Entropie der Clusterzuordnung k bei gegebener Klasse c , $H(k|c)$, und der Entropie der Gesamtheit der Cluster k , $H(k)$, definiert:

$$CS = 1 - \frac{H(k|c)}{H(k)} \quad 2-11$$

$$H(k|c) = - \sum_{c=1}^C \sum_{k=1}^K \frac{o_{c,k}}{O} \cdot \log\left(\frac{o_{c,k}}{o_c}\right) \quad 2-12$$

$$H(k) = - \sum_{k=1}^K \frac{o_k}{O} \cdot \log\left(\frac{o_k}{O}\right) \quad 2-13$$

Werden alle Objekte einer Klasse demselben Cluster zugeordnet, ist der CS mit einem Wert von 1 maximal. Der minimale Wert des CS ist 0.

Das VM wird schließlich als gewichtetes harmonisches Mittel aus HS und CS wie nachfolgend berechnet:

$$VM = (1 + \beta) * \frac{HS * CS}{(\beta * HS) + CS} \quad 2-14$$

Wird der Gewichtungsparmeter β mit einem Wert größer 1 belegt, so wird der CS stärker gewichtet, für einen Wert kleiner 1 fällt dagegen der HS mehr ins Gewicht. Für eine gleichmäßige Gewichtung beider Maßzahlen wird in der vorliegenden Arbeit β gleich 1 gesetzt.

Ein Vertreter der intrinsischen Methoden ist der *Silhouette Score* (SC) nach Rousseeuw (1987). Dieser bemisst gemäß Han, Kamber & Pei (2012) einerseits die Kompaktheit innerhalb eines Clusters, andererseits die Separierung von anderen Clustern. Für ein Objekt o berechnet sich der SC aus der mittleren Distanz zwischen o und allen weiteren Objekten aus dem gleichen Cluster, wodurch die Kompaktheit $a(o)$ des Clusters beschrieben wird. Die Separierung $b(o)$ berechnet sich dagegen als minimale mittlere Distanz zwischen o und allen anderen Clustern, denen o nicht angehört. Der SC eines Objektes o definiert sich daraus zu:

$$SC = \frac{b(o) - a(o)}{\max\{a(o), b(o)\}} \quad 2-15$$

Der Wert dieser Maßzahl liegt im Bereich $[-1,1]$, wobei ein Wert von 1 das Optimum darstellt. Um nun die Güte einer Clusterlösung zu bestimmen, wird der durchschnittliche SC über alle Objekte des Datensatzes bestimmt. (Han, Kamber & Pei 2012)

2.2.4 Rekurrente Neuronale Netze

Herkömmliche Neuronale Netze verarbeiten die Elemente einer Sequenz von Eingaben einzeln und damit unabhängig voneinander (Chollet 2018). Zur Verarbeitung von sequenziellen Daten sind dagegen Rekurrente Neuronale Netze (RNNs) geeignet (Goodfellow, Bengio & Courville 2016). Im Gegensatz zu *Feed Forward* Netzen verfügen RNNs zusätzlich über *Feedback*-Verbindungen, auch Rückkopplungen oder Schleifen genannt, wodurch die Ausgaben des Modells wiederum als Eingang dienen (Goodfellow, Bengio & Courville 2016). Die Funktionalität sowie die Eigenschaften von RNNs werden nachfolgend, soweit nicht gesondert gekennzeichnet, auf Basis von Goodfellow, Bengio & Courville (2016) und Chollet (2018) beschrieben.

Die Grundidee eines RNNs ist es, die Eingaben einer Sequenz elementweise zu verarbeiten und im sogenannten *Hidden State* einen Zustand $h^{(t)}$ zum Zeitpunkt t zu speichern, der alle Informationen der bisher gesehenen Elemente enthält. Dieser interne Zustand wird häufig auch als Gedächtnis bezeichnet (Graves 2012). Ein RNN ist folglich ein neuronales Netz, das über eine interne Schleife verfügt. In Abbildung 2.9 ist der einfachste Aufbau eines RNN, bestehend aus einer Input-, Output- und *Hidden*-Schicht, dargestellt. Dabei ist links die kompakte Darstellung aufgezeigt, in der die interne Schleife verdeutlicht wird. Rechts dagegen sind die einzelnen Zeitschritte explizit im sogenannten Berechnungsgraphen repräsentiert.

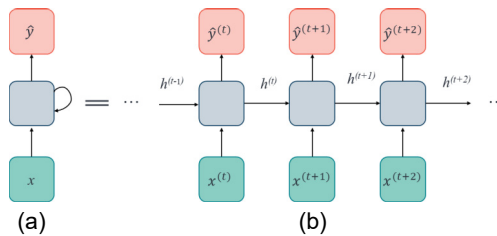


Abbildung 2.9: Schematischer Aufbau eines RNNs in kompakter Darstellung (a) und als Berechnungsgraph (b) (in Anlehnung an Graves (2012), Goodfellow, Bengio & Courville (2016) und Russell & Norvig (2022))

Mathematisch lässt sich die Verarbeitung einer Input-Sequenz wie folgt beschreiben:

$$h^{(t)} = \phi_1(b_1 + \theta_1 * h^{(t-1)} + \theta_2 * x^{(t)}) \quad 2-16$$

$$\hat{y}^{(t)} = \phi_2(b_2 + \theta_3 * h^{(t)}) \quad 2-17$$

Dabei sind b_1 und b_2 *Bias*-Vektoren, θ_1 , θ_2 und θ_3 Gewichtungsmatrizen. ϕ_1 und ϕ_2 beschreiben nicht-lineare Aktivierungsfunktionen (z.B. *Softmax*, *tanh*). Der interne Zustand $h^{(t)}$ berechnet sich folglich iterativ als Funktion aus dem vorherigen Zustand $h^{(t-1)}$, dem neuen Input $x^{(t)}$ und entsprechenden Netzparametern θ_1 , θ_2 und b_1 . Dabei kann nach Verarbeitung jedes Sequenzelementes $x^{(t)}$ bereits ein Output $\hat{y}^{(t)}$ erzeugt werden. Diese rekurrente Formulierung hat das sogenannte *Parameter Sharing* zur Folge. Diese Eigenschaft ist unter anderem dann relevant, wenn spezifische Informationen auch an unterschiedlichen Positionen in der Eingabesequenz stehen können. Durch die Verwendung der gleichen Parameter über alle Zeitschritte der Eingabe verarbeitet das RNN diese Information an jeder Position gleich. (Goodfellow, Bengio & Courville 2016) In Abbildung 2.10 wird der mathematische Zusammenhang visualisiert.

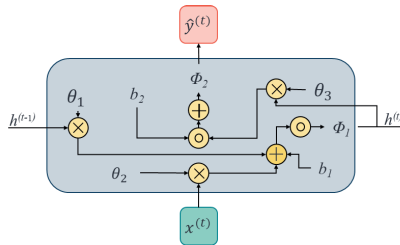


Abbildung 2.10: Berechnungen innerhalb einer RNN-Zelle (in Anlehnung an Goodfellow, Bengio & Courville (2016))

Die durch die *Hidden*-Schicht generierten internen Zustände können schließlich von der Output-Schicht für Vorhersagen verwendet werden. Auf Basis einer Verlustfunktion wird dabei der vorhergesagte Output $\hat{y}^{(t)}$ mit dem wahren Output $y^{(t)}$ verglichen und ein Fehler berechnet (siehe Abbildung 2.11). Der herkömmliche *Backpropagation*-Algorithmus wird dabei um den Faktor Zeit erweitert und daher als *Backpropagation through time* bezeichnet. Über den Gradienten der Verlustfunktion können entsprechend die Netzparameter angepasst und optimiert werden. Für eine detaillierte Beschreibung des Algorithmus wird auf Goodfellow, Bengio & Courville (2016) verwiesen.

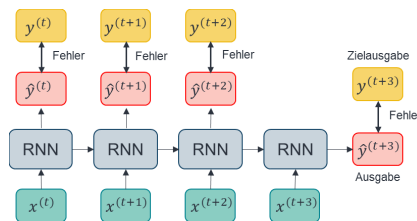


Abbildung 2.11: Schematische Darstellung der Verlustberechnung (in Anlehnung an Goodfellow, Bengio & Courville (2016))

Ein Problem, das typischerweise bei RNNs auftritt, ist das Verschwinden oder Explodieren des Gradienten⁶ (engl. *exploding/ vanishing gradient*), der über viele Schritte propagiert wird, wobei ersteres deutlich häufiger auftritt (Goodfellow, Bengio & Courville 2016, S. 396). Abhängigkeiten zwischen weit entfernten Zeitpunkten können somit deutlich schlechter gelernt werden. Je weiter ein Input in der Vergangenheit liegt, desto geringer ist dessen Einfluss auf den aktuellen Input (Graves 2012). Um diesem Nachteil zu begegnen und Informationen auch über mehrere Zeitschritte aufrechtzuerhalten,

⁶ Sind die Netzgewichte der Matrix θ kleiner null, so verschwindet der Gradient, sind sie größer null, so explodiert er (Russell & Norvig 2022).

wurden spezielle RNN-Architekturen entwickelt (Russell & Norvig 2022). Der bekannteste Vertreter ist der von Hochreiter & Schmidhuber (1997) entwickelte Ansatz des *Long Short-Term Memory* (LSTM) (Russell & Norvig 2022). Eine weitere Möglichkeit bilden Netze basierend auf den von Cho et al. (2014b) eingeführten *Gated Recurrent Unit* (GRU)-Zellen (Goodfellow, Bengio & Courville 2016).

Im Vergleich zu herkömmlichen RNN-Zellen verfügen LSTM-Zellen über eine zusätzliche interne Schleife (Goodfellow, Bengio & Courville 2016). Diese Gedächtnis-Zellen sind der Langzeitspeicher und werden von Zeitschritt zu Zeitschritt kopiert⁷ (Russell & Norvig 2022). Jede LSTM-Zelle hat dabei die gleichen Inputs und Outputs wie ein herkömmliches RNN, jedoch zusätzlich noch sogenannte *Gated Units*, über die der Informationsfluss gesteuert werden kann. (Goodfellow, Bengio & Courville 2016) Das sogenannte *Forget Gate* bestimmt, welche Elemente aus der Gedächtniszelle verworfen bzw. beibehalten werden. Über das *Input Gate* wird dagegen gesteuert, ob die Elemente der Gedächtniszelle durch den aktuellen Input-Vektor aktualisiert werden. Das *Output Gate* definiert, ob die Elemente der Gedächtniszelle in das Kurzzeitgedächtnis überführt werden. Dieses ist vergleichbar mit den *Hidden State* der Basis-RNN-Architektur. (Russell & Norvig 2022) Durch diese Architektur ist es möglich, Informationen über eine lange Zeit zu speichern und wieder abzurufen (Graves 2012).

Ebenso wie LSTM-Zellen nutzt die GRU-Zelle zur Regelung des Informationsflusses *Gates* (Chung et al. 2014). Allerdings werden lediglich zwei *Gates* benötigt, das *Reset Gate* und das *Update Gate*. Im Gegensatz zur LSTM-Zelle verfügt die GRU-Zelle über keinen internen Speicher (Chung et al. 2014). Das *Reset-Gate* entscheidet, ob der vorherige *Hidden State* ignoriert wird, sofern die Informationen für irrelevant gehalten werden. Das *Update Gate* dagegen regelt, wie viel Informationen aus dem vorangegangenen *Hidden State* in den aktuellen *Hidden State* übernommen werden (Cho et al. 2014a).

In Chung et al. (2014) wird anhand mehrerer Experimente gezeigt, dass sowohl LSTM- als auch GRU-Zellen im Vergleich zu herkömmlichen RNN-Zellen bessere Ergebnisse liefern. Eine Aussage, welche der beiden Typen überlegen ist, kann jedoch nicht getroffen werden. Im weiteren Verlauf der Arbeit werden daher alle drei Typen betrachtet.

⁷ Bei einem herkömmlichen RNN wird das Gedächtnis je Zeitschritt mit einer Gewichtungsmatrix multipliziert.

2.2.5 Deep Learning mit 3D-Daten

Deep Learning-Verfahren finden im Bereich von 2D-Daten bereits breite Anwendung in der Praxis. Erst in jüngster Zeit, durch das Aufkommen von öffentlichen Datensätzen mit 3D-Objekten, wird auch deren Anwendung im Bereich von 3D-Daten betrachtet (Guo et al. 2021). Zuvor wurden zur Verarbeitung von 3D-Daten händisch definierte Deskriptoren genutzt (Gezawa et al. 2020). Bei der Verarbeitung von 3D-Daten mittels DL gibt es jedoch einige Herausforderungen, um bestehende Algorithmen aus dem 2D-Bereich übertragen zu können. Neben der Anpassung der entsprechenden Architekturen der neuronalen Netze ist eine geeignete Repräsentationsform der 3D-Objekte zur Weiterverarbeitung ein wesentlicher Aspekt. (Gezawa et al. 2020; Ahmed et al. 2018)

Nachfolgend wird zunächst ein Überblick über die relevanten Repräsentationsformen gegeben. Häufig dienen die Repräsentationsformen als Grundlage, um im Rahmen des zuvor beschriebenen *Representation Learning*, beispielsweise mittels *Autoencoder*, eine komprimierte Darstellung als Merkmalsvektor zu extrahieren (Genova et al. 2020).

In Abbildung 2.12 sind die nach Gezawa et al. (2020) gängigsten Repräsentationsformen, die in aktuellen DL-Anwendungen genutzt werden, zusammengefasst. Die verschiedenen Formen werden dabei in fünf Hauptgruppen eingeteilt. Nachfolgend werden die für diese Arbeit relevanten Vertreter aus jeder Hauptgruppe näher erläutert. Für eine umfassende Darstellung aller Repräsentationsformen wird auf Gezawa et al. (2020) und Ahmed et al. (2018) verwiesen. Bisher gibt es noch keine Aussage darüber, welche Repräsentationsform für DL am geeignetsten ist (Sun et al. 2021). Vielmehr ist deren Eignung jeweils abhängig vom Anwendungsfall⁸ (Xiao et al. 2020).

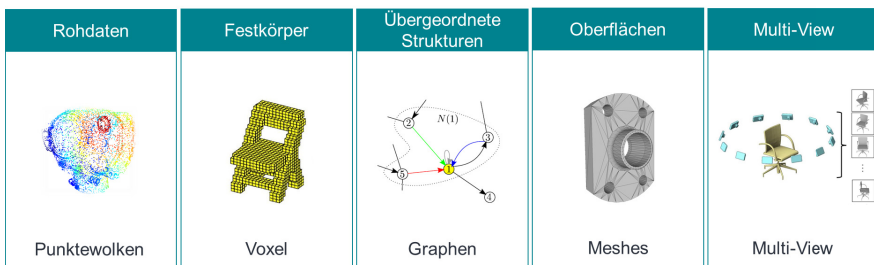


Abbildung 2.12: Taxonomie aktueller 3D-Repräsentationsformen mit beispielhaften Vertretern (in Anlehnung an Gezawa et al. (2020))

⁸ Im Rahmen dieser Arbeit erfolgt daher zur Auswahl eine Voruntersuchung, siehe Kapitel 4.2.1 und Anhang B.

2.2.5.1 Punktwolken

Grundsätzlich kann eine Punktwolke als eine Menge von unstrukturierten Punkten angesehen werden, über die die Geometrie eines 3D-Objektes angenähert wird (Ahmed et al. 2018). Als häufig verwendetes Format bewahrt sie die ursprüngliche geometrische Information im 3D-Raum ohne eine Diskretisierung vorzunehmen (Guo et al. 2021). Eine Punktwolke besteht aus einer Menge an 3D-Punkten $\{P_i | i = 1, \dots, n\}$, wobei jeder Punkt P_i einen Vektor bestehend aus den entsprechenden x-, y- und z-Koordinaten darstellt (Qi et al. 2017a). Die Reihenfolge der Punkte in der Punktwolke ist dabei unspezifisch, weshalb Punktwolken als unstrukturiert betrachtet werden. (Qi et al. 2017a) Einzelne Punkte können dabei nicht isoliert betrachtet werden, sondern bilden gemeinsam mit benachbarten Punkten eine lokale Struktur. Eine weitere Eigenschaft von Punktwolken ist deren Transformationsinvarianz. Das bedeutet, dass die Punktwolke selbst durch eine Translation oder Rotation aller Punkte unverändert bleibt. (Qi et al. 2017a) Die Verarbeitung von Punktwolken ist aufgrund dieser Eigenschaft eine herausfordernde Aufgabe (Ahmed et al. 2018). Ein erster Ansatz, der direkt Punktwolken mit Hilfe von *Convolutional Neural Networks* (CNNs) für eine spätere Klassifikation und Segmentierung verarbeiten kann, wurde von Qi et al. (2017a) entwickelt. Ein umfassender Überblick über die in den letzten Jahren entwickelten Anwendungen von DL auf Punktwolken wird in Guo et al. (2021) gegeben.

2.2.5.2 Voxel

Die voxelbasierte Darstellung beschreibt ein 3D-Objekt über eine volumenbasierte Gitterstruktur aus quadratischen Würfeln (Xiao et al. 2020). Ein 3D-Objekt wird dabei durch die Anordnung von vielen Voxeln über eine gestufte Oberfläche angenähert (Roj 2016). Um eine höhere Auflösung erreichen zu können, müssen entsprechend Voxel mit kleineren Kantenlängen gewählt werden (Roj 2016). Damit können Voxel als eine Erweiterung von Pixeln im dreidimensionalen Raum betrachtet werden, wodurch sich Architekturen von neuronalen Netzen aus dem 2D-Bereich recht einfach adaptieren lassen (Xiao et al. 2020). Durch die zusätzliche Dimension steigt jedoch der Rechenaufwand exponentiell (Xiao et al. 2020). Als Resultat sind Voxel für eine hochauflösende Repräsentation von 3D-Objekten nicht geeignet (Ahmed et al. 2018; Xiao et al. 2020). Aus diesem Grund werden sie im weiteren Verlauf dieser Arbeit nicht näher betrachtet.

2.2.5.3 Meshes und Graphen

Meshes bestehen aus einer Kombination von Ecken, Kanten und Flächen (auch Polygone genannt (Ahmed et al. 2018)) und werden häufig für die graphische Datenverarbeitung verwendet (Gezawa et al. 2020). Eine große Herausforderung bei dieser Repräsentationsform ist deren Irregularität und Komplexität. Hanocka et al. (2019) und Feng et al. (2019) zeigen erste Ansätze, um diese Probleme zu lösen, dennoch finden sie bisher kaum im Bereich DL-Anwendung (Gezawa et al. 2020; Ahmed et al. 2018). Eine weitere, eng mit den *Meshes* verwandte Darstellungsform nicht-euklidischer Daten sind Graphen. *Meshes* können auch als Graph angesehen werden, indem die Ecken als Knoten und die Kanten als Verbindungen im Graphen dargestellt werden. (Gezawa et al. 2020; Ahmed et al. 2018) Graphen können gerichtet oder ungerichtet sein. Durch die Analyse der spektralen Eigenschaften von Graphen ist die Verarbeitung mit neuronalen Netzen möglich (Ahmed et al. 2018). In Simonovsky & Komodakis (2017) wird beispielsweise ein graphbasiertes CNN entwickelt, über das 3D-Objekte in Form von Graphen klassifiziert werden können. Die entwickelten Ansätze auf Graphen können auf *Meshes* übertragen werden, umgekehrt jedoch nicht. (Ahmed et al. 2018)

2.2.5.4 Multi-View

Die grundlegende Idee des *Multi-View*-Ansatzes ist es, ein 3D-Objekt über eine Menge von zweidimensionalen Bildern, die aus verschiedenen Perspektiven aufgenommen werden, darzustellen. Über die perspektivischen Darstellungen, die jeweils unterschiedliche, sich gegenseitig ergänzende Informationen beinhalten, kann schließlich auf die ursprüngliche Form des 3D-Objektes geschlossen werden (Xiao et al. 2020). Der große Vorteil ist, dass auf Basis der so erzeugten zweidimensionalen Bilder klassische DL-Architekturen aus dem 2D-Bereich eingesetzt werden können, ohne aufwendige Anpassungen vornehmen zu müssen (Gezawa et al. 2020). Beispielsweise werden in Su et al. (2015) CNNs dazu genutzt, aus den Bildern der verschiedenen Perspektiven sogenannte *Feature Maps* zu erstellen, die anschließend über ein *Pooling* in eine gemeinsame Darstellung überführt werden. Darauf basierend wird ein zweites CNN trainiert, um die zugrundeliegenden 3D-Objekte zu klassifizieren. Eine Herausforderung des *Multi-View*-Ansatzes ist es jedoch, eine ausreichende Anzahl an Perspektiven festzulegen (Gezawa et al. 2020). Zudem ist es nicht möglich, innenliegende geometrische Eigenschaften abzubilden (Gezawa et al. 2020). Solche inneren Strukturen sind jedoch gerade für (mechanische) Komponenten aus dem industriellen Bereich relevant.

3 Stand der Technik

Im Rahmen dieses Kapitels werden bereits existierende Ansätze aufgezeigt, die sich mit der in diesem Vorgehen betrachteten Problemstellung der automatisierten Wissensextraktion aus 3D-Daten und der nachfolgenden Nutzung dieses Wissens in Form eines Assistenzsystems für die Produktentwicklung beschäftigen.

Um den aktuellen Stand der Forschung datengetriebener Assistenzsysteme in der Produktentwicklung zu erfassen, müssen Arbeiten aus zwei Bereichen betrachtet werden: Zum einen stellen Arbeiten über die Repräsentation und Verarbeitung von 3D-Modellen mit Hilfe von ML die Grundlage zur Extraktion des Wissens aus vorhandenen Produktmodellen dar. Diese lassen sich auf den Kontext der Produktentwicklung übertragen, um zunächst die CAD-Modelle in geeigneter Form zu repräsentieren und anschließend beispielsweise ähnliche 3D-Objekte zu identifizieren oder sequentielle Daten aus dem Konstruktionsvorgehen zu verarbeiten. Damit leisten sie einen Beitrag zur Beantwortung der Forschungsfragen 1 und 3 – wie mittels ML Wissen aus Produktmodellen und dem darin enthaltenen Konstruktionsvorgehen nutzbar gemacht werden kann. Zum anderen werden Ansätze beleuchtet, die bisher zur Unterstützung in der Produktentwicklung konzeptioniert wurden und damit auf die Beantwortung der Forschungsfragen 1 bis 3 abzielen. Teilweise wird auch hier bereits auf zuvor erwähnten ML-Ansätze zurückgegriffen. Dabei werden nur solche Arbeiten betrachtet, die in den hinteren Phasen der Produktentwicklung, in denen verstärkt CAD-Systeme zum Einsatz kommen, unterstützen.

Neben den forschungsseitigen Ansätzen werden auch bereits existierende, kommerzielle Softwaresysteme betrachtet, die als Unterstützung im Konstruktionsprozess eingesetzt werden können. Abschließend erfolgt das Aufzeigen des Forschungsdefizits, das durch den in dieser Arbeit entwickelten Ansatz abgedeckt werden soll.

3.1 Maschinelles Lernen zur Repräsentation und Verarbeitung von 3D-Objekten

Zu Verarbeitung von 3D-CAD-Modellen mittels ML-Verfahren müssen diese zunächst in eine geeignete Repräsentationsform gemäß Kapitel 2.2.5 umgewandelt werden. Einige Ansätze führen direkt auf diesen Repräsentationsformen Aufgaben wie die Klassifikation, Segmentierung oder Ähnlichkeitssuche von 3D-Modellen aus, indem die entsprechenden ML-Modelle selbstständig relevante *Features* erlernen. Als erste Ansätze

sind hier beispielsweise Su et al. (2015) für *Multi-View*, Qi et al. (2017a) und Qi et al. (2017b) für Punktwolken, Simonovsky & Komodakis (2017) für Graphen und Hanocka et al. (2019) für *Meshes* anzuführen. Zur Extraktion der relevanten Merkmale werden hier in den meisten Fällen CNNs herangezogen (Aljalbout et al. 2018). Für ein erfolgreiches Lernen sind große Mengen gelabelter Daten notwendig (Hassani & Haley 2019), die in vielen Fällen jedoch nicht vorliegen (Goodfellow, Bengio & Courville 2016). Zur effizienteren Verarbeitung mit ML gibt es Ansätze, die zunächst im Rahmen des *Representation Learning* selbständig relevante Informationen aus den (ungelabelten) Daten extrahieren, die dann beispielsweise für Klassifizierungsaufgaben genutzt werden können (Bengio, Courville & Vincent 2013). Hierzu zählen Ansätze basierend auf *Generative Adversarial Networks* (GANs) (Goodfellow et al. 2014; Li et al. 2019; Wu et al. 2016), Encoder-Decoder-Architekturen (Achlioptas et al. 2018; Yuan et al. 2018; Tchapmi et al. 2019; Wang et al. 2020; Yang et al. 2018) oder impliziten Funktionen (Park et al. 2019; Mescheder et al. 2019). Basierend auf den so erzeugten latenten Repräsentationen werden 3D-Objekte mit diesen Ansätzen beispielsweise generiert oder komplettiert.

Der Ansatz von Achlioptas et al. (2018) wandelt 3D-Objekte basierend auf Punktwolken zunächst über einen *Autoencoder* in eine latente Repräsentation um. Auf Basis des *ShapeNet* Datensatzes (Chang et al. 2015) mit synthetisch erzeugten 3D-Objekten aus dem Alltag werden die erstellten latenten Vektoren genutzt, um die 3D-Objekte mit entsprechenden ML-Verfahren zu klassifizieren, ähnliche Objekte zu finden (*Shape Retrieval*), neue Objekte einer Klasse zu generieren (*Shape Generation*) oder unvollständige Objekte in Form von Punktwolken zu komplettieren (*Shape Completion*). Eine Weiterentwicklung des *Autoencoders* wird in Yang et al. (2018) vorgestellt. Hier werden ebenfalls die auf Basis des *ShapeNet* Datensatzes erlernten latenten Vektoren genutzt, um die zugehörigen 3D-Modelle zu clustern oder zu klassifizieren.

In einem Großteil der Datensätze sind nach Sun et al. (2021) die 3D-Objekte bereits vorausgerichtet. Das bedeutet, dass die 3D-Modelle auf den Einheitswürfel normiert und so ausgerichtet sind, dass die Semantik des Objektes entsprechend den euklidischen Achsen angeordnet ist (z.B. steht ein Auto immer auf der x-Ebene und dessen Front ist stets in x-Richtung ausgerichtet). Diese Eigenschaft trifft auch auf den *ShapeNet*-Datensatz zu. Um auch Daten ohne eine solche Vorausrichtung verarbeiten zu können, wird in Sun et al. (2021) eine Architektur mit *Capsule Nets* und vorgeschalteter

Kanonisierung der Modelle eingeführt. Dafür nutzen die Autoren eine Architektur bestehend aus *Autoencoder* und *Capsule Networks*. Die Architektur ist in der Lage, die Pose der 3D-Objekte in Form von Punktwolken zu erkennen und entsprechend alle Objekte einer Objektklasse einheitlich auszurichten. Für ihre Untersuchungen nutzen sie einen Teil der *ShapeNet*-Daten, die sie durch Rotation und Translation manipulieren. Die Normierung auf den Einheitswürfel bleibt jedoch erhalten. Die auf Basis der vorausgerichteten 3D-Objekte erlernten latenten Repräsentationen werden anschließend in mehreren Experimenten einerseits für überwachte Lernaufgaben in Form einer Klassifikation mittels *Support Vector Machines*, andererseits für unüberwachte Clusterbildung mittels *k-Means* herangezogen. Die erhaltenen Ergebnisse auf dem *ShapeNet*-Datensatz übertreffen dabei laut den Autoren die bisherigen Ergebnisse aus dem Stand der Technik zum Zeitpunkt der Veröffentlichung des Ansatzes.

Eine ebenso auf *Encoder* und *Decoder* basierende Architektur nutzen Yuan et al. (2018) zur Komplettierung (*Shape Completion*) von Punktwolken. Weitere Ansätze zur *Shape Completion* mit ähnlichen Architekturen sind Xie et al. (2020), Tchapmi et al. (2019), Wang et al. (2020) und Sarmad, Lee & Kim (2019). Generell zielen diese darauf ab, unvollständig gescannte Objekte, z.B. über LiDAR, in Form von Punktwolken zu komplettieren. Die für das Training notwendigen, nur teilweise vollständigen Punktwolken werden künstlich aus den bestehenden Datensätzen wie *ShapeNet* erzeugt (Yuan et al. 2018). Häufig wird auch auf den Datensatz *The KITTI Vision Benchmark Suite* (Geiger et al. 2013) zurückgegriffen, der per Laserscanner aufgenommene Objekte aus dem Straßenverkehr in Form von Punktwolken beinhaltet. In den zuvor genannten Datensätzen entsprechen fehlende Teile der Objekte folglich keinen (für die Produktentwicklung) logischen Komponenten. Mit der Veröffentlichung des *PartNet*-Datensatzes (Mo et al. 2019), in dem 3D-Objekte in Form von Einrichtungsgegenständen in logische Komponenten (z.B. ein Stuhl in Beine, Sitzfläche etc.) aufgeteilt sind, wurden in jüngster Zeit auch Ansätze vorgestellt, die auf halbfertigen 3D-Objekten erprobt wurden, bei denen logische Komponenten fehlen. Im Ansatz von Wu et al. (2020a) wird zudem nicht nur eine mögliche Komplettierung ausgegeben, sondern mehrere Optionen. Eine schrittweise Komplettierung, wie es im Rahmen dieser Arbeit durch die Prädiktion nächster Konstruktionsschritte vorgesehen ist, ist jedoch mit keinem der Ansätze möglich.

In Zou et al. (2017) werden 3D-Objekte des *PartNet*-Datensatzes mit Hilfe von RNNs erstmals als Sequenz von Box-Primitiven rekonstruiert. Diese Rekonstruktion zielt jedoch nur darauf ab, die äußere Form des Modells nachzubilden und ist unabhängig vom tatsächlichen Konstruktionsvorgehen. Wu et al. (2020b) entwickeln einen Ansatz, bei dem über einen *Sequence-to-Sequence-Autoencoder* eine Repräsentation eines 3D-Objektes über die Sequenz der enthaltenen Komponenten erlernt wird. Die Komponenten werden dabei jeweils über eine implizite Funktion repräsentiert. Neben der reinen Geometrie, die über den vom *Autoencoder* erlernten Merkmalsvektor dargestellt wird, werden je Komponente auch die *Bounding Box* sowie deren Position im Raum einbezogen, um zusätzlich Informationen über die Skalierung sowie die globale Anordnung der Komponenten zu berücksichtigen. Das Modell kann dazu genutzt werden, fehlende Komponenten eines Bauteiles zu ergänzen.

Zusammenfassend kann festgehalten werden, dass in den letzten Jahren deutliche Verbesserungen im Bereich der Verarbeitung von 3D-Modellen mittels ML erzielt wurden. Ausschlaggebend dafür ist insbesondere die wachsende Zahl an öffentlich verfügbaren Datensätzen mit 3D-Modellen (Gezawa et al. 2020). Ein Großteil der Arbeiten basiert jedoch auf Datensätzen, die lediglich alltägliche Gegenstände und keine tatsächlichen CAD-Dateien umfassen (Koch et al. 2019). Obwohl das Interesse an der Anwendung der ML-Verfahren in der CAD-Domäne in jüngster Zeit deutlich gestiegen ist, mangelt es an Datensätzen, die realistische CAD-Modelle beinhalten (Willis et al. 2021). Zwar existieren auch einige Datensätze, die tatsächliche CAD-Konstruktionen enthalten (z.B. der ABC-Datensatz (Koch et al. 2019)), dennoch sind auch hier keine Informationen über das (reale) Konstruktionsvorgehen hinterlegt (Wu, Xiao & Zheng 2021). Diese Information wird erstmalig durch den Datensatz von Willis et al. (2021) bereitgestellt. Darauf aufbauende Ansätze werden im nächsten Kapitel vorgestellt.

3.2 Ansätze zur Unterstützung im Produktentwicklungsprozess

Im Nachfolgenden werden zunächst Ansätze aufgezeigt, die die systematische Wiederverwendung vorhandener CAD-Modelle (Forschungsfrage 1) fokussieren (Kapitel 3.2.1). Anschließend werden in Kapitel 3.2.2 solche Lösungen vorgestellt, die sich mit der Bewertung produktionsrelevanter Produkteigenschaften beschäftigen (Forschungsfrage 2). Das Nutzen von Konstruktionsmustern zum Vorschlagen nächster Konstruktionsschritte (Forschungsfrage 3) wird in Kapitel 3.2.3 beleuchtet. Bereits existierende Ansätze für Assistenzsysteme in der Produktentwicklung, wodurch die übergeordnete

Zielstellung dieser Arbeit und damit alle drei Forschungsfragen adressiert werden, werden in Kapitel 3.2.4 aufgezeigt.

3.2.1 Ansätze zur Ähnlichkeitssuche von CAD-Modellen

Zur Wiederverwendung vorhandener Designs und der darin enthaltenen Lösungen gibt es bereits mehrere Ansätze mit dem Ziel, ähnliche CAD-Modelle aufzufinden oder diese anhand ihrer Ähnlichkeit zu gruppieren. Ein umfassender Überblick ist beispielsweise in Lupinetti et al. (2019) zu finden. Um die Ähnlichkeit zu bewerten, werden häufig sogenannte *Shape Signatures* aus der entsprechenden 3D-Geometrie abgeleitet, über die deren Eigenschaften und Charakteristika repräsentiert werden (Hong, Lee & Kim 2006).

Rea et al. (2002) entwickeln eine Suchmaschine zur Ähnlichkeitsbestimmung von 3D-Modellen. Der hinterlegte Algorithmus arbeitet auf zuvor definierten *Global shape metrics*, die die Kerneigenschaften der Geometrie beschreiben sollen. Volumen, Oberfläche, Volumen-Oberfläche-Ratio, *Bounding Box*, *Crinkliness* (Verhältnis von Oberfläche des Modells zur Oberfläche einer Kugel gleichen Volumens), Kompaktheit sowie die Anzahl an Flächen und Löchern kennzeichnen das jeweilige Objekt. Anhand der Ähnlichkeit dieser Eigenschaften werden schließlich ähnliche 3D-Modelle aufgefunden.

Ähnlich zum zuvor beschriebenen Ansatz nutzen Machalica & Matyjewski (2019) 45 geometrische Eigenschaften (z.B. Volumen, Oberfläche) zur Beschreibung der CAD-Modell-Geometrie. Darauf basierend werden verschiedene ML-Verfahren zur Clusterbildung der Modelle angewendet.

Ebenso erstellen Roj et al. (2021) auf Basis von CAD-Modellen im proprietären Dateiformat zunächst einen sogenannten Fingerabdruck zur Beschreibung der Geometrie. Dieser setzt sich aus den verschiedenen Oberflächentypen (z.B. rund, dreieckig), die aus dem CAD-Modell ausgelesen werden können, der Position der jeweiligen Zentren im kartesischen Koordinatensystem, deren Verbindungen zu anderen Oberflächen sowie den entsprechenden Distanzen zusammen. Diese Informationen werden schließlich in eine Graphstruktur überführt. Auf Basis der Fingerabdrücke werden 18 Attribute (z.B. *Bounding Box*, Oberflächen-Verhältnisse) abgeleitet, um die Fingerabdrücke verschiedener CAD-Modelle miteinander vergleichen zu können. Basierend auf diesen Parametervektoren wird schließlich eine Clusterbildung mit *k-Means* durchgeführt, um ähnliche Teile in Klassen zusammenzufassen.

Die zuvor beschriebenen Ansätze haben gemeinsam, dass die Ähnlichkeitssuche auf vordefinierten Eigenschaften basiert. Die eigentliche geometrische Gestalt eines Bauteils kann dadurch nicht direkt repräsentiert werden, sondern wird hier lediglich über entsprechende Merkmale, wie z.B. die *Bounding Box*, beschrieben. Weitere Ansätze nutzen die im CAD-Modell hinterlegten Konstruktionselemente zur Beschreibung der Geometrie.

Der Ansatz von Bai et al. (2010) betrachtet den CAD-Strukturbaum, um über Ähnlichkeiten von Sub-Bäumen geometrisch ähnliche Modelle oder Modellkomponenten zu finden. Die Ähnlichkeit wird auf Basis von den verwendeten CAD-KEs bestimmt. Dabei werden nur sehr einfache KEs (z.B. Loch, Kerbe) betrachtet, die durch die Autoren vordefiniert wurden. Diese Tatsache wird von den Autoren als Schwäche ihres Ansatzes angemerkt. Viele CAD-Programme haben komplexere KEs, die zunächst über vordefinierte KEs nachgestellt werden müssen.

In Tsai & Chang (2005) wird eine zweistufige Suche für *Feature*-basierende Konstruktionen (CSG) mittels eines *Fuzzy Adaptive Resonance Theory (ART) Networks* realisiert. Nach der Geometrieanalyse über die verwendeten geometrischen KEs, wie z.B. Bohrungen und Fasen, erfolgt hier zusätzlich in der zweiten Phase der Suche eine Betrachtung der zugehörigen technologischen Attribute. Dazu zählen Volumen, Materialhärte, Korrosionsbeständigkeit, Oberflächengüte und Toleranzen. Die Suchergebnisse der ersten Phase werden schließlich anhand der Ähnlichkeit der technologischen Attribute bewertet.

Ein Ansatz, der sowohl die globale als auch lokale Geometrie von Bauteilen zur Bestimmung der Ähnlichkeit berücksichtigt, wird von Bickel et al. (2021) vorgestellt. Hier wird eine geometrische Ähnlichkeitssuche im Anwendungsgebiet von blechumgeformten Teilen entwickelt. Die Ähnlichkeitssuche läuft dabei in zwei Stufen ab. Im ersten Schritt wird das betrachtete Bauteil in eine Punktwolke umgewandelt und mittels der Projektion der Punkte auf eine Oberfläche in eine Matrix transformiert. Um ähnliche Bauteile zu finden, werden die Korrelationskoeffizienten der Matrizen verglichen und die Ergebnisse geclustert. Im nächsten Schritt werden innerhalb der Cluster die Bauteile anhand lokaler geometrischer Ähnlichkeit sortiert. Die entsprechenden geometrischen Eigenschaften werden dabei manuell festgelegt. Dafür wird zunächst das Bauteil mit Hilfe von ML anhand der festgelegten *Features* segmentiert und die relevanten geometrischen lokalen Segmente mit dem Projektionsverfahren auf ihre Ähnlichkeit hin untersucht.

3.2.2 Ansätze zur Analyse produktionsrelevanter Produkteigenschaften

In Zhang, Jaiswal & Rai (2018) wird ein ML-basierter Ansatz vorgestellt, über den für die Fertigung relevante Merkmale, sogenannte *Machining-Features*, mittels eines CNN in 3D-CAD-Modellen erkannt werden. Hierfür wird ein Voxel-basiertes CNN anhand eines künstlich generierten Datensatzes, bestehend aus 24 verschiedenen, häufig vorkommenden *Machining-Features*, trainiert. Das trainierte Modell kann schließlich dazu genutzt werden, die 24 verschiedenen *Feature*-Typen in 3D-Objekten zu erkennen. Sofern mehrere *Features* in einem Modell vorhanden sind, muss dieses zunächst durch einen Separierungsalgorithmus aufgeteilt werden, sodass jedes Sub-Modell nur ein *Feature* enthält. Über diesen Ansatz lassen sich folglich produktionsrelevante Produkteigenschaften in Form von *Machining-Features* direkt in den CAD-Modellen erkennen. Allerdings sind nur einfache *Features* erkennbar und es können auch nur solche erkannt werden, auf denen das CNN zuvor trainiert wurde.

Einen ähnlichen Ansatz präsentieren Peddireddy et al. (2020) zur Erkennung von Dreh- und Fräs-*Features*, jedoch wenden sie zusätzlich *Transfer Learning* an, indem vortrainierte Netzwerkschichten des Grundmodells in ein Zielmodell übertragen werden. Zunächst werden im Grundmodell voxelisierte CAD-Modelle nach *Machining-Features* klassifiziert. Das Training der CNN-Architektur findet dabei auf einem künstlich generierten Datensatz statt, in dem jedes CAD-Modell nur eines der *Features* beinhaltet. Im Gegensatz zu Zhang, Jaiswal & Rai (2018) werden die einzelnen Dreh- und Fräs-*Features* in einem weiteren Datensatz zufällig miteinander kombiniert. Im Rahmen des *Transfer Learnings* werden die trainierten CNN-Schichten des Grundmodells in ein Zielmodell überführt und die Gewichte eingefroren. Alle weiteren Schichten des Zielmodells werden schließlich auf dem kombinierten Datensatz trainiert. Ziel ist es, für ein CAD-Modell mit kombinierten Fertigungsmerkmalen auszugeben, welche Prozesse (Drehen, Fräsen, Drehen und Fräsen) zur Herstellung notwendig sind. Die dafür notwendigen Label werden für den betrachteten Datensatz händisch durch Experten erzeugt.

Schönhof & Fechter (2020) entwickeln einen Ansatz namens *NeuroCAD*, um CAD-Modelle von Komponenten bezüglich der Automatisierungsmöglichkeiten der entsprechenden Montageprozesse zu bewerten. Mittels eines CNN werden die voxelisierten CAD-Modelle anhand relevanter Eigenschaften (z.B. Greifflächen, Separierung) in 11 Machbarkeitsstufen klassifiziert. Für das Training werden die zugrunde gelegten CAD-Modelle zunächst von Experten anhand der für die Automatisierung relevanten Kriterien

in 11 Stufen eingeteilt. Mit Hilfe von *Data Augmentation* in Form von Rotation und Skalierung wird dieser händisch gelabelte Datensatz vergrößert und für das Training von *NeuroCAD* genutzt. Welche Bereiche des CAD-Modells jedoch ausschlaggebend für die Bewertung sind, kann nicht aufgezeigt werden.

Im Gegensatz zu den zuvor aufgezeigten Ansätzen zur Erkennung fertigungsrelevanter geometrischer Merkmale zeigen Germani, Mandolini & Cicconi (2011) einen Ansatz zur Herstellkostenschätzung von CAD-Modellen. Ziel ist es, verschiedene Alternativen bereits in der frühen Entwurfsphase anhand der erwarteten Kosten zu bewerten. Sie definieren sogenannte *Advanced Manufacturing Features* mit einer Menge an geometrischen Elementen, die jeweils mit Fertigungsoperationen verknüpft sind. Die Fertigungsoperationen werden dabei über geometrische Parameter eindeutig charakterisiert. Für ein CAD-Modell wird schließlich überprüft, welche *Features* es beinhaltet und darauf basierend die notwendigen Fertigungsoperationen und deren Reihenfolge abgeleitet. Anhand von Bearbeitungs-, Rüst-, Lager- und Werkzeugkosten werden anschließend die Fertigungskosten kalkuliert.

3.2.3 Ansätze zum Vorschlagen nächster Konstruktionsschritte

In Jaiswal, Huang & Rai (2016) wird ein Ansatz basierend auf probabilistischen Faktorgraphen vorgestellt, der fehlende Komponenten auf Baugruppenbasis vorschlägt, um schnell konzeptionelle Entwürfe, inspiriert durch existierende Konstruktionen, erstellen zu können. Über die Faktorgraphen werden geometrische und semantische Beziehungen zwischen den Modellen und deren Komponenten abgebildet. Anhand der Wahrscheinlichkeitsverteilung, die aus den Faktorgraphen abgeleitet werden kann, wird die Reihenfolge bestimmt, in der fehlende Komponenten vorgeschlagen werden. Die Autoren validieren ihren Ansatz anhand von 3D-Modellen, wie Tischen und Stühlen, die zuvor in Komponenten segmentiert wurden. In einer prototypischen Benutzeroberfläche werden dem Nutzer für ein gegebenes Objekt fehlende Teile vorgeschlagen.

Im Gegensatz zu Jaiswal, Huang & Rai (2016) wenden die nachfolgenden Ansätze erstmals ML-Verfahren an, um Muster aus dem Konstruktionsvorgehen zu erlernen.

In Willis et al. (2021) wird erstmals mit der Einführung von *Fusion 360 Gallery* ein Datensatz bereitgestellt, der realistische CAD-Modelle mit Konstruktionssequenzen, die von Konstrukteuren gestaltet wurden, enthält. Dieser enthält unter anderem auch CAD-Modelle mechanischer Komponenten, wie z.B. Unterlegscheiben. Die Konstruktionssequenzen beschränken sich dabei auf die KEs „Skizze“ und „Extrusion“. Auf Basis dieses

Datensatzes entwickeln die Autoren die interaktive Umgebung namens *Fusion 360 Gym*. Hier werden die Sequenzen als Markov-Entscheidungsprozess modelliert. Darauf basierend wird ein *Reinforcement Learning*-Agent trainiert, der von einem aktuellen Zustand eines Objektes durch Hinzufügen der KEs Skizze oder Extrusion einen gewünschten Endzustand generiert. Als Limitationen des Ansatzes wird angemerkt, dass lediglich Skizzen und Extrusionen als CAD Modellierungsoperationen genutzt und zudem nur kurze Konstruktionssequenzen betrachtet werden.

Wu, Xiao & Zheng (2021) entwickeln eine Repräsentationsform, die speziell CAD-Modelle über die Sequenz der genutzten CAD-Kommandos und der entsprechenden Parametrisierung darstellt. Sie beschränken sich dabei ebenso auf die KEs „Skizze“ und „Extrusion“. Die erstellten CAD-Kommando-Sequenzen werden dann mit Hilfe eines Transformer-Netzwerks verarbeitet. Das Transformer-Netzwerk besteht dabei aus einem *Encoder* und einem *Decoder*. Der *Encoder* wandelt die entsprechende Sequenz in einen latenten Vektor um. Der *Decoder* kann aus diesen latenten Vektoren wieder die ursprüngliche Kommando-Sequenz rekonstruieren. Diese Sequenz lässt sich schließlich wiederum in ein CAD-Modell übersetzen. Die Repräsentation der Kommando-Sequenz als latenten Vektor nutzen die Autoren schließlich, um ein GAN zu trainieren, das neue Sequenzen in Form von latenten Vektoren generiert. Über den trainierten *Decoder* können daraus schließlich wieder Kommando-Sequenzen und damit CAD-Modelle erzeugt werden. Als mögliche Anwendung zeigen sie auf, dass beispielsweise Punktwolken von Objekten, die über 3D Scans erzeugt wurden, nach Umwandlung in einen entsprechenden latenten Vektor anschließend durch den *Decoder* in eine Kommando-Sequenz umgewandelt werden können. Der große Vorteil ist, dass die so erzeugten Objekte im CAD vom Nutzer editierbar sind – im Gegensatz beispielsweise zu Punktwolken. Als Schwäche nennen die Autoren, dass weitere CAD-Kommandos, wie Abrundungen, mit dem aufgezeigten Ansatz nicht abgebildet werden können, da sie auf bereits existierenden KEs referenzieren. Zudem steigt mit wachsender Sequenzlänge auch die Wahrscheinlichkeit, dass die erzeugten Sequenzen fehlerbehaftet sind und somit zu ungültigen Topologien führen.

3.2.4 Ansätze für Assistenzsysteme

Kratzer, Binz & Roth (2010) entwickeln ein Unterstützungssystem für die Entwurfs- und Ausarbeitungsphase, das die Konstrukteure bei Routinetätigkeiten entlasten soll. Das agentenbasierte System ProKon (Proaktive Unterstützung von Konstruktionsprozessen

durch Softwareagentensysteme) soll dabei einerseits eine proaktive Konsistenzprüfung von CAD-Modellen sowie eine multikriterielle Lösungsfindung bezüglich verschiedener DfX-Kriterien bereitstellen (Kratzer, Binz & Roth 2010). Andererseits soll es als aktives Nachschlagewerk für Konstrukteure dienen (Kratzer, Binz & Roth 2010). Das entsprechende Konstruktions- bzw. Erfahrungswissen muss jedoch durch einen Wissensingenieur zunächst aus Normen, Fachbüchern oder Interviews mit Konstrukteuren extrahiert werden, um es anschließend in das Unterstützungssystem zu überführen (Kratzer, Binz & Roth 2010). Die Wissensextraktion erfolgt dementsprechend nicht automatisiert.

Zur vollen Potenzialausnutzung der Blechmassivumformung wurde im Rahmen des Sonderforschungsbereiches TR 73 ein selbstlernendes Assistenzsystem (SLASSY) entwickelt (Röhner, Breitsprecher & Wartzack 2011; Breitsprecher & Wartzack 2012). Mit Hilfe von *Data Mining* wird implizites Wissen bezüglich der Fertigbarkeit aus vorhandenen Daten extrahiert und in explizites Wissen transformiert. In einer Simulationsstudie werden dabei verschiedene Parameter eines Bauteils variiert und die Auswirkungen auf die spätere Produzierbarkeit betrachtet (Röhner, Breitsprecher & Wartzack 2011). Für jeden fertigungsrelevanten Parameter wird dabei auf Basis der Simulationsdaten mit Hilfe von ML ein eigenes Metamodell antrainiert (Breitsprecher & Wartzack 2012). Mit Hilfe dieser Metamodelle kann schließlich eine Komponente bezüglich verschiedener fertigungsrelevanter Parameter analysiert werden. Auf Basis einer multikriteriellen Optimierung werden dem Konstrukteur verschiedene pareto-optimale Designs vorgeschlagen, aus denen er wählen kann (Wartzack, Sauer & Küstner 2017).

Küstner (2020) entwickelt ein Konzept zur intensiveren Nutzung produktbeschreibender Daten vorangegangener Produktgenerationen, um insbesondere die tatsächlichen Eigenschaften von Serienprodukten besser vorhersagen zu können. Als produktbeschreibende Daten werden dabei insbesondere Geometriedaten (CAD-Daten), Toleranzangaben, Stücklisten, dokumentierte Anforderungen und Auslegung sowie Simulations- und Messdaten betrachtet. Zunächst wird eine geeignete Repräsentation der Daten entwickelt, sodass darauf Verfahren des ML angewandt werden können. Darauf aufbauend wird ein Konzept für ein Assistenzsystem entwickelt und umgesetzt, das einerseits Wissen akquiriert und bereitstellt, andererseits auch den Wissenstransfer zwischen Experten aus verschiedenen Domänen integriert. Über das Assistenzsystem wird den Produktentwicklern ein Softwarewerkzeug bereitgestellt, um selbstständig datengetriebene Analysen mittels ML durchführen zu können.

Yoo et al. (2021) entwickeln ein ML-basiertes Rahmenwerk für die konzeptionelle Phase der Produktentwicklung. Mit Hilfe des Systems kann eine Vielzahl an möglichen Konstruktionen automatisch generiert und deren technische Leistungsfähigkeit bewertet werden. Produktentwickler können somit aus den automatisch generierten Entwürfen einen geeigneten Kandidaten auswählen, den sie anschließend ausdetaillieren. Der Ansatz wird dabei anhand einer Fallstudie mit Laufrädern gezeigt. Grundlage bilden 2D-Skizzen, die zunächst mittels eines *Autoencoders* in einen latenten Raum reduziert werden. Der Datensatz wird dabei durch *Sampling* aus dem latenten Raum vergrößert. Die entsprechenden latenten Vektoren können anschließend wieder in 2D-Skizzen rekonstruiert werden. Auf Basis des so generierten Datensatzes werden automatisiert CAD-Modelle generiert. Für diese Modelle werden anschließend CAE-Simulationen durchgeführt, um deren Leistungsfähigkeit mit Hinblick auf verschiedene technische Parameter zu bewerten. Die resultierenden Simulationsdaten werden anschließend für das Training eines CNNs genutzt, das die 2D-Skizzen bezüglich deren technischer Leistungsfähigkeit bewerten kann.

3.3 Kommerzielle Assistenzsysteme für die Konstruktion

Es existieren bereits einige kommerzielle CAD-Assistenzsysteme. Eine wichtige Grundfunktion für den Umgang mit CAD-Daten ist die Modellklassifikation und das Auffinden ähnlicher Bauteile. Diese Eigenschaften werden von vielen Systemen, wie SIMILIA®, Simus Classmate®, Geolus Shape Search®, Exalead One Part®, Modelsearch®, Cadenas Part Solutions® und CADBAS® bereits umgesetzt. Entsprechende Eigenschaften, über die die Ähnlichkeit bestimmt wird, werden größtenteils händisch definiert. Beispielsweise wird die Geometrie eines Bauteiles in SIMILIA® über das *Shape Indexing* Verfahren beschrieben. Hier werden geometrisch signifikante Merkmale mit Indizes versehen, die im Zuge der Ähnlichkeitssuche miteinander verglichen werden.⁹

In Modelsearch® wird ein sogenannter geometrischer Fußabdruck erstellt, der eine Geometrie über eine Vielzahl von Eigenschaften (z.B. Abmessungen, Flächen) beschreibt und verschiedene Geometrien dadurch vergleichbar macht¹⁰.

⁹ SIMUFORM® Search Solutions GmbH (2022), *Geometrische Ähnlichkeitssuche – Was ist das?* <https://blog.simuform.com/geometrische-aehnlichkeitssuche-was-ist-das> (aufgerufen am 30.03.2022)

¹⁰ NET AG system integration (2022), *ModelSearch – Geometrieähnlichkeitssuche von TECHSOFT* <https://www.net-online.de/modelsearch/> (aufgerufen am 30.03.2022)

Exalead One Part® bietet dagegen eine Nutzeroberfläche, die der einer Internetsuchmaschine ähnelt. Über ein Textfeld können Begriffe oder verallgemeinerte Informationen eingegeben werden. Zusätzlich können Informationen wie Erstellungsdatum, Abmessungen oder Material angegeben werden. Eine ähnliche Funktionalität bietet auch Geolus Shape Search. (Roj 2016)

Im Gegensatz dazu basiert die Software von MonolithAI® auf ML-Verfahren, die über eine Plattform den Anwendern zugänglich gemacht werden. Basierend auf vorhandenen Simulations- und Testdaten kann damit beispielsweise das Verhalten neuer Produkte antizipiert werden.¹¹

Produktionsrelevante Produkteigenschaften werden insbesondere in Form von Kosten adressiert. In SIMILIA® kann als zusätzlicher Filter in der Ähnlichkeitssuche beispielsweise der Preis berücksichtigt werden¹². Simus Classmate® bietet die Möglichkeit, bereits während der Konstruktion eine Vorkalkulation der Herstellkosten auf Basis der Geometrie sowie der verfügbaren Fertigungstechnologien zu erzeugen¹³.

Ein von der Daimler AG speziell für die CAD-Umgebung Siemens NX® entwickelte Assistenzsystem namens *NeuroCAD* basiert ebenfalls auf ML-Verfahren. Über den sogenannten *Feature*-Assistenten werden, basierend auf typischen Kommando-Sequenzen, die drei wahrscheinlichsten nächsten Kommandos in Form von KEs vorgeschlagen. Der Konstrukteur kann aus diesen Vorschlägen den geeignetsten auswählen oder, sofern kein passendes Kommando enthalten ist, selbstständig das nächste Konstruktionselement wählen. (Stark 2022)

3.4 Forschungsdefizit

Basierend auf den Herausforderungen, den darauf aufbauenden forschungsleitenden Fragestellungen sowie den aufgezeigten Grundlagen wird der Beitrag des aktuellen Stands der Forschung zu datengetriebenen Assistenzsystemen anhand der nachfolgenden Kriterien bewertet.

¹¹ Monolith AI® (2022), *An Advanced AI platform fine-tuned for engineering* <https://www.monolithai.com/> (aufgerufen am 30.03.2022)

¹² SIMUFORM® Search Solutions GmbH (2022), *Teilevielfalt reduzieren* <https://www.simuform.com/software/teilevielfalt-reduzieren.html> (aufgerufen am 30.03.2022)

¹³ @simus systems GmbH (2022), *classmate PLAN* <https://www.simus-systems.com/produkte/classmate-plan/> (aufgerufen am 30.03.2022)

Ein Großteil der bestehenden ML-Verfahren auf 3D-Daten nutzt eine der in Kapitel 2.2.5 vorgestellten Repräsentationsformen. Diese beschreiben jedoch lediglich die geometrische Form. Insbesondere im Bereich der Produktentwicklung sind jedoch weitere Informationen, wie Material oder Abmessungen, sehr relevant. Das Kriterium *Multipler Informationsinput* bewertet daher die verschiedenen Informationsarten, die zeitgleich genutzt werden (z.B. Geometrie, geometriebeschreibende Parameter, produktionsrelevante Produkteigenschaften, Konstruktionshistorie).

Gemäß Forschungsfrage 1 ist es das Ziel, explizit das bereits existierende (fallspezifische) Wissen aus bestehenden Produktdaten zu nutzen. Über das Kriterium *Nutzung bestehender Informationen/ Daten* wird daher bewertet, ob dieser Informationsgehalt bereits existiert oder ob zunächst entsprechende künstliche Daten oder manuelle Labels generiert werden müssen. Ob das entsprechende Wissen (z.B. in Form von Merkmalen oder Mustern) automatisiert mit ML-Verfahren aus den vorhandenen Daten extrahiert werden kann (siehe auch Kapitel 2.1.4), wird durch das Kriterium *Automatisierte Merkmalsextraktion* evaluiert.

Da ein Großteil der ML-basierten Ansätze auf öffentlichen Datenbanken mit einfachen 3D-Objekten angewandt wird, die explizit dafür ausgelegt sind (z.B. durch vorausgerichtete 3D-Modelle), wird auch die *Anwendung auf industriellen CAD-Daten* berücksichtigt.

Die in dieser Arbeit angestrebte systematische Nutzung des Wissens aus bestehenden Produktgenerationen wird über verschiedene Funktionalitäten, die das Assistenzsystem umfassen soll, umgesetzt. Diese werden über die fünf nachfolgenden Kriterien beschrieben.

Unter *Strukturierung der Daten* wird die Funktion eines Systems verstanden, vorhandene Bauteile in Bauteilklassen/-cluster einzuteilen, um somit die vorhandene Datenbasis z.B. nach der geometrischen Form zu strukturieren. Dadurch wird die Grundlage für die *Ähnlichkeitssuche* gelegt. Diese beinhaltet das Auffinden von ähnlichen Bauteilen während oder zu Beginn der Konstruktion. Dabei ist wichtig, dass auch eine halbfertige Konstruktion als Input dienen kann und dennoch ausreichend gute Ergebnisse geliefert werden können. Die Ähnlichkeit bezieht sich dabei sowohl auf die Geometrie als auch auf weitere relevante Informationen wie produktionsrelevante Produkteigenschaften.

Die *Prädiktion nächster Konstruktionsschritte* beinhaltet das aktive Führen durch den Konstruktionsprozess, beispielsweise durch das Vorschlagen des nächsten Bauteilzustandes in Form der 3D-Geometrie. Basis hierfür bilden Muster im Konstruktionsvorgehen der vorhandenen CAD-Modelle.

Die Bewertung des aktuellen Bauteils hinsichtlich produktionsrelevanter Produkteigenschaften, die für die spätere Produzierbarkeit auf vorhandenen Maschinen relevant sind, wird über das Kriterium *Bewertung produktionsrelevanter Produkteigenschaften* abgebildet.

Eine Übersicht über den Stand der Technik sowie dessen Einschätzung hinsichtlich der aufgestellten Kriterien ist in Abbildung 3.1 gegeben. Die Arbeiten zur **Repräsentation und Verarbeitung von 3D-Modellen mit ML** weisen generell das Defizit auf, dass meist lediglich die reine Geometrie des 3D-Modells als Informationsbasis genutzt wird. Grund dafür ist, dass die Ansätze im Wesentlichen darauf abzielen, über verschiedene Aufgaben (Klassifikation, Komplettierung, etc.) geeignete Repräsentationsformen von 3D-Objekten zu erlernen. Daraus resultiert, dass ein Großteil der Ansätze auf Datensätzen mit einfachen 3D-Objekten, wie beispielsweise *ShapeNet*, erprobt wird, die keinen tatsächlichen CAD-Modellen entsprechen. Um diese Anwendungsmöglichkeiten auf die Produktentwicklung übertragen zu können, müssen die Ansätze zunächst auf Daten aus dem technischen Bereich erprobt und ggf. entsprechend angepasst werden.

Bei den Arbeiten zur **Unterstützung im Produktentwicklungsprozess** wird häufig entweder die Geometrie oder geometriebeschreibende Parameter als Informationsbasis genutzt. Lediglich in Kratzer, Binz & Roth (2010) und Küstner (2020) kann prinzipiell sämtliches Konstruktionswissen in das System eingebracht werden. Wie jedoch in den meisten anderen Ansätzen auch ist eine automatisierte Extraktion der relevanten Merkmale nicht möglich, da bisher noch kaum Ansätze aus dem Bereich des ML eingesetzt werden. Gerade im Bereich der Ähnlichkeitssuche von CAD-Modellen werden häufig noch vordefinierte Eigenschaften genutzt, über die datenspezifische Besonderheiten oftmals nicht ausreichend dargestellt werden können (Ren, Niu & Fang 2017).

Ansätze zur Bewertung **produktionsrelevanter Produkteigenschaften** wie beispielsweise Schönhof & Fechter (2020), Zhang, Jaiswal & Rai (2018) oder Peddireddy et al. (2020) nutzen zwar bereits ML-Verfahren, jedoch ist hier zunächst eine künstliche Erzeugung von Daten oder Labels notwendig.

Anforderungen

- Erfüllt
- ◐ Teilweise erfüllt
- Nicht erfüllt
- ✘ Keine Bewertung möglich

		Datennutzung				Assistenzfunktionen				
		Multipler Informationsinput	Nutzung bestehender Daten/ Informationen	Automatisierte Merkmalsextraktion	Anwendung auf industriellen CAD-Daten	Strukturierung der Daten	Ähnlichkeitsanalyse	Prädiktion nächster Konstruktionschritte	Bewertung produktionsrelevanter Produkteigenschaften	
Repräsentation und Verarbeitung von 3D-Modellen mit Maschinellem Lernen	Su et al. (2015)	○	●	●	○	●	○	○	○	
	Qi et al. (2017a/b)	○	●	●	○	●	○	○	○	
	Simonovsky & Komodakis (2017)	○	●	●	○	●	○	○	○	
	Zou et al. (2017)	○	●	●	○	○	○	◐	○	
	Achlioptas et al. (2018)	○	●	●	○	●	◐	○	○	
	Yang et al. (2018)	○	●	●	○	●	●	○	○	
	Yuan et al. (2018)	○	●	●	○	○	○	○	○	
	Hanocka et al. (2019)	○	●	●	○	●	○	○	○	
	Sarmad, Lee & Kim (2019)	○	●	●	○	◐	○	○	○	
	Tchapmi et al. (2019)	○	●	●	○	○	○	○	○	
	Wang et al. (2020)	○	●	●	○	●	○	○	○	
	Wu et al. (2020a)	○	●	●	○	○	○	○	○	
	Wu et al. (2020b)	○	●	●	○	○	○	◐	○	
	Xie et al. (2020)	○	●	●	○	○	○	○	○	
	Sun et al. (2021)	○	●	●	○	●	●	○	○	
	Unterstützung im Produktentwicklungsprozess	Ähnlichkeitssuche	Rea et al. (2002)	○	●	○	●	○	●	○
Tsai & Chang (2005)			◐	◐	○	●	○	●	○	●
Bai et al. (2010)			○	◐	○	●	○	●	○	○
Machalica & Matyjewski (2019)			◐	●	○	●	○	○	○	○
Bickel et al. (2021)			◐	●	○	●	◐	●	○	○
Roj et al. (2021)			○	●	○	●	●	◐	○	○
Produktionsrelevante Produkteigenschaften		Germani, Mandolini & Cicconi (2011)	◐	●	○	●	○	○	○	●
		Zhang, Jaiswal & Rai (2018)	○	◐	●	●	◐	○	○	●
		Peddireddy et al. (2020)	○	◐	●	●	◐	○	○	●
		Schönhof & Fechter (2020)	○	◐	●	●	○	○	○	●
Prädiktion Konstruktionschritte		Jaiswal, Huang & Rai (2016)	○	◐	●	○	○	●	◐	○
		Willis et al. (2021)	○	◐	●	●	○	○	●	○
Assistenzsysteme		Wu, Xiao & Zheng (2021)	○	◐	●	●	○	○	◐	○
		Kratzer, Binz & Roth (2010)	●	◐	○	●	○	○	◐	●
		Röhner, Breitsprecher & Wartzack (2011) (TR 73)	○	○	○	●	○	○	○	●
		Küstner (2020)	●	●	○	◐	○	○	○	◐
Yoo et al. (2020)	◐	○	◐	●	○	○	○	●		
Kommerzielle Systeme	CADBAS®	✘	●	●	●	●	●	○	○	
	Cadenas Part Solutions®	✘	●	●	●	●	●	○	○	
	Exalead One Part®	○	●	●	●	●	●	○	○	
	Geolus Shape Search®	○	●	●	●	●	●	○	○	
	ModelSearch®	○	●	●	●	●	●	○	○	
	Monolith AI®	◐	◐	●	●	○	○	○	●	
	SIMILIA®	○	●	●	●	●	●	○	◐	
	Simus Classmate®	✘	●	●	●	●	●	○	●	

Abbildung 3.1: Übersicht Stand der Technik

Die Arbeiten von Willis et al. (2021) und Wu, Xiao & Zheng (2021) nutzen erstmals die Konstruktionshistorie in Form des CAD-Strukturbaumes, um mittels ML-Verfahren Muster zu erlernen und nächste KEs oder komplette Modelle vorzuschlagen. Allerdings werden vereinfacht nur Sequenzen mit lediglich zwei verschiedenen KEs herangezogen.

In Bezug auf die verschiedenen **Funktionalitäten eines Assistenzsystems** adressieren die bestehenden Arbeiten in der Regel nur einzelne Anwendungen wie beispielsweise eine Ähnlichkeitssuche oder die Bewertung produktionsrelevanter Produkteigenschaften. Ein System mit einer umfassenden Funktionalität existiert noch nicht. Zudem betrachten die Verfahren zum Auffinden geometrisch ähnlicher Modelle lediglich finale Komponenten, eine Suche auf Basis halbfertiger Komponenten wird nicht adressiert.

Ein Großteil der existierenden **kommerziellen Softwaresysteme** bietet bereits Lösungen zur Strukturierung oder Ähnlichkeitssuche für CAD-Daten. In der Regel basieren die zugrunde gelegten Algorithmen jedoch nicht auf ML-Verfahren. Produktionsrelevante Produkteigenschaften werden häufig nur in Form der Kosten bzw. Preise berücksichtigt. Eine aktive Unterstützung im eigentlichen Konstruktionsvorgehen wird lediglich durch das Assistenzsystem *NeuroCAD* geboten. Nächste Konstruktionsschritte werden in Form von den drei wahrscheinlichsten Kommandos vorgeschlagen. Eine Vorhersage des nächsten Konstruktionszustandes in Form der 3D-Geometrie wird nicht betrachtet.

Abschließend lassen sich drei wesentliche Aspekte ableiten. Erstens wird im Stand der Forschung meist nur eine Informationsquelle genutzt. Zweitens werden in den Ansätzen im Bereich der Produktentwicklung noch kaum ML-Verfahren eingesetzt, um automatisiert entsprechendes Wissen aus vorhandenen Produktmodellen zu extrahieren. Für einen Großteil der bestehenden ML-Ansätze muss zunächst ein entsprechender künstlicher Datensatz generiert oder die vorhandenen Produktmodelle gemäß der gewünschten Zielstellung händisch mit Labeln versehen werden. Der in diesem Vorhaben entwickelte Ansatz dagegen soll bereits existierende Daten und das darin enthaltene Wissen nutzen, um schon frühzeitig u.a. auf abweichende produktionsrelevante Produkteigenschaften, die auf mögliche Probleme bei der späteren Produzierbarkeit mit den vorhandenen Maschinen hinweisen können, hinzuweisen. Drittens lassen sich im Hinblick auf mögliche Funktionalitäten eines Assistenzsystems Defizite identifizieren. Bestehende Ansätze der Ähnlichkeitssuche betrachten bisher noch keine halbfertigen Bauteilzustände als Sucheingabe. Darüber hinaus betrachten existierende Ansätze zur Vorhersage nächster Konstruktionsschritte lediglich einfache und kurze Konstruktionssequenzen. Reale und komplexere Konstruktionssequenzen werden nicht adressiert.

4 Eigener Ansatz

In den nachfolgenden Kapiteln wird die entwickelte Methodik zur Nutzung der existierenden Wissensbasis in Form von Produkten aus bestehenden Generationen mit Hilfe von Verfahren des Maschinellen Lernens vorgestellt.

Aus dem Stand der Forschung lässt sich dabei ableiten, dass eine neue Methodik zu entwickeln ist, die bestehende ML-Verfahren auf CAD-Daten aus dem Bereich des industriellen Produktentwicklungsprozesses überträgt und entsprechend anpasst. Neben der Geometrie werden dabei auch weitere Informationen wie geometriebeschreibende Parameter und produktionsrelevante Produkteigenschaften sowie die Konstruktionshistorie betrachtet. Als Datenbasis dienen ausschließlich bereits existierende Produktkomponenten und damit verknüpfte Informationen, sodass keine künstliche Datengenerierung notwendig ist. Durch eine geeignete Aufbereitung und Repräsentation wird diese Wissensbasis in Form eines Konstruktionsassistenzsystems nutzbar gemacht. Dieses unterstützt einerseits durch das sukzessive Vorschlagen von nächsten Konstruktionschritten in Abhängigkeit des aktuellen Konstruktionsstandes eines halbfertigen CAD-Modells, sodass schnell ein erster Grobentwurf gemäß bewährter Lösungsmuster erstellt werden kann. Auf Basis des aktuellen Konstruktionszustandes werden andererseits ähnliche, bereits existierende Produktmodelle in der Datenbank gesucht und daher schon in der frühen Entwurfsphase aufgezeigt. Von diesen geometrisch ähnlichen Produktmodellen werden zudem produktionsrelevante Produkteigenschaften abgeleitet und mit den entsprechenden Ausprägungen dieser Eigenschaften des aktuellen Konstruktionsstandes verglichen. Mögliche Abweichungen innerhalb dieser Eigenschaften, die für die spätere Produzierbarkeit auf den vorhandenen Produktionsmitteln kritisch sein könnten, werden somit frühzeitig aufgedeckt. Der entwickelte Ansatz greift folglich vor allem in der Phase der Ausgestaltung der Module ein (siehe Kapitel 2.1). Die auf Basis der Grobentwürfe aufgezeigten ähnlichen Modelle, die bereits ausdetailliert sind, können durch Wiederverwendung die Notwendigkeit einer weiteren Ausdetaillierung der Grobgestalt reduzieren. Die grundsätzliche Vorgehensweise sowie die Funktionalität des Assistenzsystems sind in Abbildung 4.1 dargestellt.

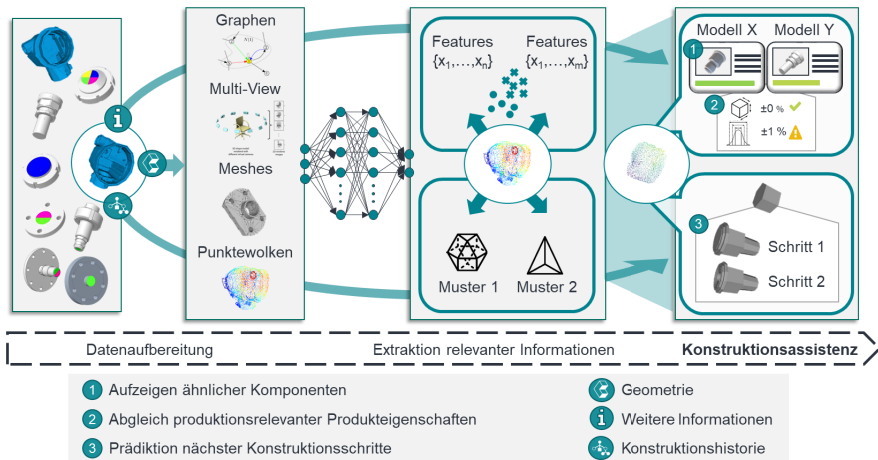


Abbildung 4.1: Grundsätzliche Vorgehensweise zur Wissensextraktion und Funktionalität der darauf basierenden Konstruktionsassistenten

Durch die systematische Wiederverwendung bereits existierender Komponenten können unnötige Mehrfahrkonstruktionen vermieden sowie teure nachträgliche Änderungen (z.B. aufgrund mangelnder Funktionalität oder Nicht-Produzierbarkeit auf vorhandenen Produktionsmitteln) reduziert werden. Zur Umsetzung dieses Konstruktionsassistentensystems werden die folgenden Bestandteile (vgl. Kapitel 1.2) aufgezeigt:

- Die Kategorisierung des bestehenden Wissens in Form der vorhandenen Produktmodelle sowie anschließend dessen entsprechende Aufbereitung wird in Kapitel 4.1 und 4.2 aufgezeigt. Die zugänglichen Daten sind je nach Anwendungsfall unterschiedlich. In Abhängigkeit der Datenlage je Kategorie kann anschließend definiert werden, welche Funktionen des Assistenzsystems für den jeweiligen Anwendungsfall umsetzbar sind.
- Die automatisierte Extraktion relevanter Eigenschaften aus CAD-Modellen mittels ML-Verfahren, um das vorhandene Wissen in geeigneter Form zu repräsentieren, wird in Kapitel 4.3 adressiert.
- Die Nutzung der extrahierten Informationen in Form einer Ähnlichkeitssuche, um auf Basis des aktuellen Konstruktionsstandes bereits existierende, geometrisch ähnliche CAD-Modelle aufzufinden, sowie die Ableitung produktionsrelevanter Produkteigenschaften aus diesen ähnlichen CAD-Modellen zur Bewertung eines neuen Produktes sind Inhalt von Kapitel 4.4.1.

- Fokus von Kapitel 4.4.2 ist das Erlernen von Mustern im Konstruktionsprozess mittels ML, um den Konstrukteur aktiv durch das Vorschlagen nächster Konstruktionsschritte in Form des nächsten Konstruktionszustandes zu unterstützen.
- Das Aufzeigen der prinzipiellen Machbarkeit und der erzielbaren Ergebnisse erfolgt in Kapitel 5.

Grundlage für die Umsetzung des Konstruktionsassistenzsystems sind CAD-Daten von Einzelkomponenten aus dem Produktportfolio eines Industrieunternehmens. Baugruppen, die aus einer Zusammensetzung von Einzelkomponenten bestehen, werden im Rahmen dieser Arbeit nicht betrachtet. Die entwickelte Methodik ist in Abbildung 4.2 dargestellt. Das Vorgehen orientiert sich an dem in Kapitel 2.1.4 eingeführtem CRISP-DM, wobei die aufgezeigten Besonderheiten bei Anwendungen von ML-Verfahren zu berücksichtigen sind. Da es jedoch nicht das Ziel ist, ein in der Praxis anwendbares Softwaretool zu implementieren, wird die Phase des *Deployments* nicht betrachtet.

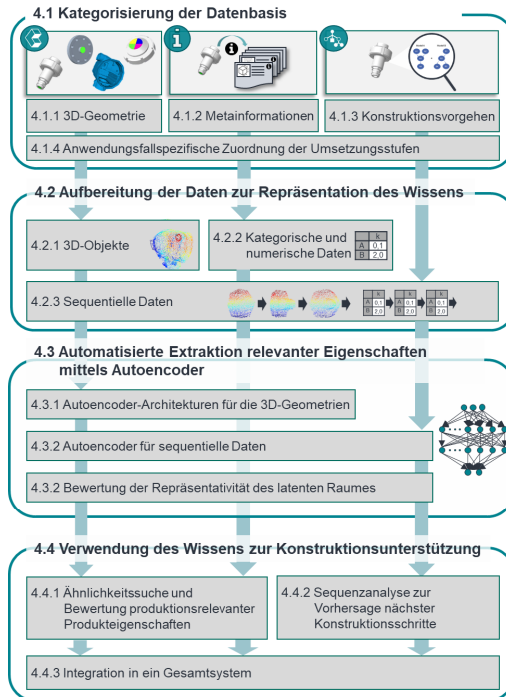


Abbildung 4.2: Überblick und Struktur der entwickelten Methodik

In einem ersten Schritt wird die vorhandene Datenbasis bezüglich ihres Informationsgehalts untersucht und die entsprechenden Informationen in ein Datenmodell, bestehend aus drei Kategorien, zugeordnet (Kapitel 4.1). Neben der eigentlichen 3D-Geometrie aus den CAD-Daten werden zusätzliche Informationen, beispielsweise geometriebeschreibende Parameter oder produktionsrelevante Produkteigenschaften, in Betracht gezogen. Diese Informationen werden unter dem Begriff Metainformationen zusammengefasst. Darüber hinaus lassen sich aus den CAD-Daten über den Strukturbaum Rückschlüsse auf das Vorgehen des Konstrukteurs zum Erstellen einer Einzelkomponente ziehen. Dieses Wissen über das Konstruktionsvorgehen stellt die dritte und letzte Datenkategorie dar. Welche dieser Datenkategorien für einen gegebenen Anwendungsfall erfüllbar sind, hängt von der entsprechenden Datenverfügbarkeit ab. In Abhängigkeit davon kann schließlich anwendungsfallspezifisch der Umfang der Funktionalität des Konstruktionsassistenten und damit dessen umsetzbarer Reifegrad bestimmt werden (Kapitel 4.1.4). In einem nächsten Schritt werden die Daten, je nach Datentyp, für eine Weiterverarbeitung entsprechend aufbereitet. Unterschieden wird dabei zwischen 3D-Geometrie-Daten (Kapitel 4.2.1) sowie numerischen und kategorischen Daten (Kapitel 4.2.2). Sequentielle Daten (Kapitel 4.2.3), die aus dem Konstruktionsvorgehen resultieren, können dabei aus den zuvor genannten Datentypen aufgebaut sein. Anschließend folgt die automatisierte Extraktion der relevanten (anwendungsfallspezifischen) Eigenschaften aus den vorhandenen 3D-Geometrien. Hier wird mit Hilfe eines *Autoencoders* eine geeignete und leicht verarbeitbare Repräsentation der 3D-Geometrie spezifisch für den jeweils betrachteten Datensatz erlernt (Kapitel 4.3). Der *Autoencoder* wird dabei sowohl für die 3D-Geometrie der finalen Komponente (Kapitel 4.3.1) als auch für die einzelnen 3D-Geometrien, die die zugehörige Konstruktionshistorie abbilden (sog. Zwischenzustände), genutzt (Kapitel 4.3.2). Die Hyperparameter des *Autoencoders* sind dabei anwendungsfallspezifisch zu wählen, um einen möglichst repräsentativen latenten Raum zu erlernen. Dessen Repräsentativität kann über verschiedene Untersuchungen überprüft werden (Kapitel 4.3.3). Ergebnis ist letztendlich die Darstellung jeder (finalen) Komponente über einen geometrischen Fußabdruck, bestehend aus den relevanten geometrischen Eigenschaften und zugehörigen Metainformationen. Die entsprechende Konstruktionshistorie wird aus der Sequenz der Zwischenzustände und KEs einer Komponente abgebildet. Die Zwischenzustände werden ebenfalls über deren relevante geometrische Merkmale repräsentiert. Das auf diese Weise repräsentierte Wissen aus den vorhandenen Produktkomponenten wird

schließlich in Form eines Assistenzsystems nutzbar gemacht. Dessen Funktionalität umfasst einerseits die Ähnlichkeitssuche, die, je nach Datenverfügbarkeit, entweder rein auf der Geometrie oder auf Geometrie und Metainformationen beruhen kann (Kapitel 4.4.1). Auf Basis ähnlicher, existierender Modelle werden außerdem produktionsrelevante Produkteigenschaften abgeleitet und für das aktuell in der Konstruktion befindliche Modell bewertet (Kapitel 4.4.1.3). Andererseits werden aus den sequentiellen Daten des Konstruktionsvorgehens bestehender Komponenten mit Hilfe von *Recurrent Neural Networks* (RNNs) Muster erlernt, auf deren Grundlage beim Konstruieren einer neuen Komponente nächste Konstruktionsschritte vorgeschlagen werden (siehe Kapitel 4.4.2). Durch die Nutzung dieser Vorschläge als erweiterter Such-Input für die Ähnlichkeitssuche werden beide Funktionalitäten zu einem Gesamtsystem integriert (Kapitel 4.4.3).

4.1 Kategorisierung der Datenbasis

Voraussetzung für die Anwendung von ML-Verfahren ist die Verfügbarkeit entsprechender Daten (Studer et al. 2021). Da diese in Abhängigkeit des Anwendungsfalls einen unterschiedlichen Informationsgehalt aufweisen, wird zunächst die anwendungsfallspezifische Datenbasis gesichtet und identifiziert, welche Informationen daraus entsprechend definierter Kategorien vorhanden sind.

Grundlage für das entwickelte Vorgehen sind vorhandene CAD-Modelle von Komponenten sowie weitere, damit verknüpfte Informationen, wie beispielsweise aus dem PDM (*Product Data Management*) oder PLM (*Product Lifecycle Management*). Wesentliche Voraussetzung ist, dass es sich dabei um Komponenten handelt, die bei dem betrachteten Anwendungsunternehmen bereits erfolgreich produziert wurden und nach wie vor dem Produktportfolio angehören. Hintergrund ist, dass die herangezogenen ML-Verfahren nur so gut sind wie das Wissen in Form der Daten, auf denen sie trainiert werden. Werden Schlechteile, beispielsweise mit unnötigen Produkteigenschaften, für das Training herangezogen, so wird auch das Assistenzsystem lernen, solche Eigenschaften vorzuschlagen.

Die Informationen, über die jede Komponente im Rahmen dieser Arbeit beschrieben wird, werden gemäß Abbildung 4.3 in die drei Datenkategorien 3D-Geometrie, Metainformationen und Konstruktionsvorgehen aufgeteilt. Innerhalb der Metainformationen wird nochmals unterschieden, ob diese produktionsrelevant sind oder nicht.

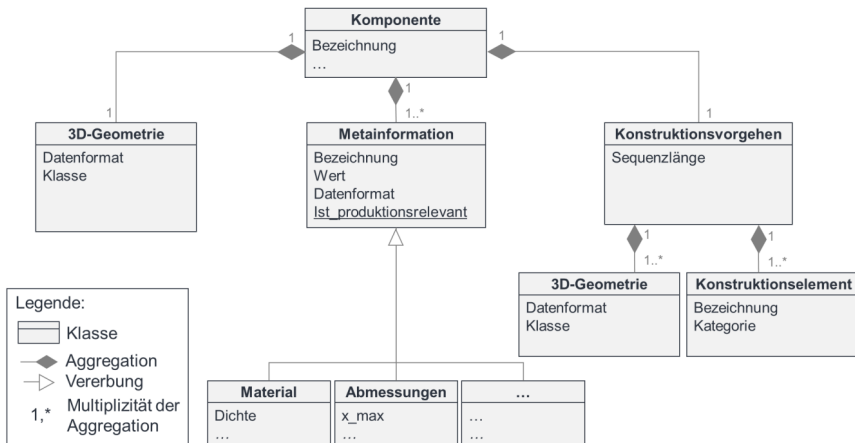


Abbildung 4.3: UML Klassendiagramm der drei Datenkategorien

Um zunächst den Informationsgehalt für eine gegebene Datenbasis zu bestimmen, kann gemäß Abbildung 4.4 vorgegangen werden. Grundvoraussetzung ist, dass CAD-Modelle von Komponenten in ausreichender Anzahl¹⁴ vorhanden sind, da die Datenverfügbarkeit maßgeblich die Umsetzbarkeit und Performanz von ML-Verfahren beeinflusst (Studer et al. 2021). Gemäß Kapitel 2.1.3.2 ist der Informationsgehalt der CAD-Modelle abhängig vom Dateiformat. Liegt ein Austauschformat, wie z.B. STEP, vor, so sind Informationen über die Konstruktionshistorie in Form des Strukturbaums nicht vorhanden. Die geometrische Form des CAD-Modells liefert dennoch Informationen der Kategorie 3D-Geometrie. Sind aus den CAD-Modellen neben der reinen Geometrie auch weitere Metainformationen, z.B. die Abmessungen, extrahierbar, so liegen zusätzlich Daten aus der Kategorie Metainformationen vor. Sind zusätzlich noch Daten aus dem PDM oder PLM vorhanden, kann der Umfang der potenziell relevanten Metainformationen noch erweitert werden. Sind dagegen die CAD-Modelle im proprietären, herstellerspezifischen Dateiformat vorhanden, so können über den Strukturbaum zusätzlich Informationen über das Konstruktionsvorgehen ausgelesen werden. Darüber hinaus sind aus den herstellerspezifischen Dateiformaten üblicherweise weitere geometriebeschreibende

¹⁴ Grundsätzlich gilt, je mehr, desto besser. Nach Erfahrungswerten im Rahmen der Evaluation in Kapitel 5 sollte die Anzahl an CAD-Modellen insgesamt im dreistelligen Bereich liegen, wobei je Klasse mindestens eine zweistellige Anzahl an Objekten notwendig ist.

Parameter, wie Abmessungen, Material oder Toleranzen, extrahierbar. Durch weitere Daten aus dem PDM bzw. PLM können diese Metainformationen angereichert werden.

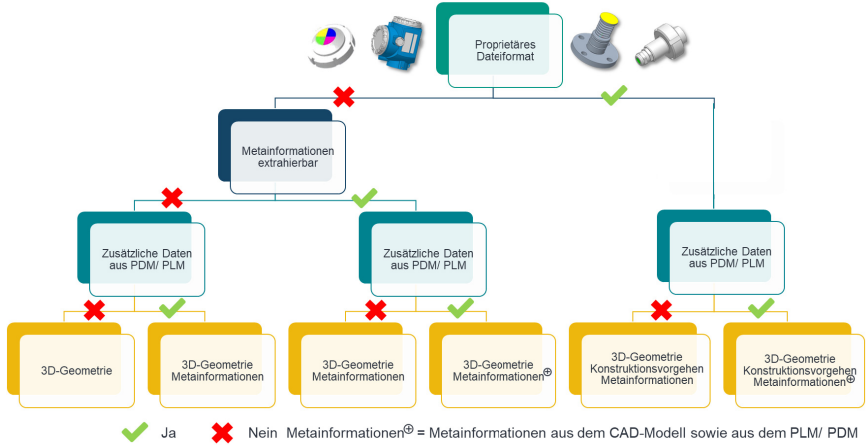


Abbildung 4.4: Vorgehen zur Bestimmung der vorhandenen Datenkategorien

Eine Beschreibung dieser Kategorien erfolgt in den nachfolgenden Kapiteln. Die verfügbaren Daten sind ausschlaggebend für die Machbarkeit von ML-Anwendungen und damit für die Umsetzbarkeit der gesetzten Ziele (Studer et al. 2021). Basierend darauf, in welchem Umfang Daten aus den genannten Kategorien vorhanden sind, kann entsprechend die umsetzbare Funktionalität des Assistenzsystems und damit die Zielsetzung für den jeweiligen Anwendungsfall abgeleitet werden.

4.1.1 3D-Geometrie

Grundlage für diese Datenkategorie bilden die (finalen) CAD-Modelle, die die 3D-Geometrie der entsprechenden Produktkomponenten verkörpern. Gemäß Abbildung 4.3 verfügt die 3D-Geometrie über ein bestimmtes Dateiformat. Der entwickelte Ansatz benötigt zur Weiterverarbeitung CAD-Modelle im STL (*Standard Triangulation Language*)-Format (siehe Kapitel 2.1.3.2), weshalb in einem ersten Schritt die Konvertierung der CAD-Modelle in STL erfolgt. Dieses Format zählt zu den Austauschformaten und kann sowohl aus den proprietären CAD-Dateiformaten als auch aus weiteren Austauschformaten (z.B. STEP-Format) erzeugt werden. Hintergrund ist, dass das STL-Format für die spätere Generierung von Punktwolken (siehe Kapitel 4.2.1) gut geeignet ist. Prinzipiell sind jedoch auch andere Datei-Formate möglich. Sofern zumindest von einem

Teil der Komponenten eine entsprechende Information über die Bauteilklasse vorhanden ist, wird diese der 3D-Geometrie als kategorische Variable angehängt (in Abbildung 4.3 Attribut Klasse). Generell ist das Klassen-Label optional, kann jedoch zu einer Verbesserung der Merkmalsextraktion (siehe Kapitel 4.3.1.2) sowie zu Untersuchungen der erlernten Merkmalsrepräsentationen (siehe Kapitel 4.3.3) genutzt werden.

4.1.2 Metainformationen

Unter Metainformationen werden sämtliche Daten numerischer oder kategorischer Natur zusammengefasst, die neben der reinen Geometrie in Form des CAD-Modells noch zusätzlich über die entsprechende Komponente vorhanden sind. Hierzu zählen einerseits Informationen, die direkt aus dem CAD-Modell abgeleitet werden können, insbesondere geometriebeschreibende Daten (z.B. Volumen, maximale Abmessungen) oder produktionsrelevante Daten (z.B. Toleranzen, Material). Andererseits fallen hierunter auch sämtliche Daten, die für die betrachtete Komponente in der Datenbank, z.B. im PLM-System, vorhanden sind. Diese können ebenfalls produktionsrelevant sein (z.B. Stückkosten, Stückzahlen). Um eine konsistente Vergleichbarkeit der Komponenten hinsichtlich der Metainformationen zu ermöglichen, ist es wesentlich, dass die Verfügbarkeit dieser Informationen für alle betrachteten Komponenten des Datensatzes gegeben ist. Ist dies der Fall, werden in einem nächsten Schritt diejenigen Parameter ausgewählt, die für die Ähnlichkeit zweier Komponenten für den jeweiligen Anwendungsfall als relevant eingestuft werden (beispielsweise über Experteninterviews). Darüber hinaus kann für jedes Attribut angegeben werden, ob es für die Produzierbarkeit auf den vorhandenen Produktionsmitteln von Bedeutung ist (in Abbildung 4.3 Attribut Ist_productionrelevant). Die Produktionsrelevanz ist ein Klassenattribut und damit für alle Instanzen der Klasse Metainformation identisch.

Ergebnis ist letztendlich eine Liste an Metainformationen, die für die spätere Ähnlichkeitssuche und die Bewertung der produktionsrelevanten Produkteigenschaften (siehe Kapitel 4.4.1) herangezogen wird.

4.1.3 Konstruktionsvorgehen

Informationen über das Konstruktionsvorgehen sind in den CAD-Modellen über den Strukturbaum (siehe Kapitel 2.1.3.1) ableitbar. Voraussetzung ist das Vorliegen von CAD-Modellen der entsprechenden Komponenten im proprietären CAD-Dateiformat. Sind die CAD-Modelle nur in Austauschformaten (z.B. STL) verfügbar, kann lediglich

die 3D-Geometrie des finalen Zustandes abgebildet werden. Im proprietären Dateiformat lässt sich über den Strukturbaum schließlich das Konstruktionsvorgehen für jedes CAD-Modell über die Sequenz der genutzten KEs abbilden. Zunächst werden die im betrachteten Datensatz genutzten KEs identifiziert. Diese sind meist spezifisch je nach genutzter CAD-Software und können in verschiedene Kategorien eingeteilt werden. Für die CAD-Software PTC Creo® Parametric lassen sich beispielsweise volumenerzeugende, editierende, konstruierende und unterstützende KEs unterscheiden.

Das Konstruktionsvorgehen selbst kann schließlich über zwei Wege dargestellt werden (siehe Abbildung 4.5). Zum einen lässt sich das Vorgehen über die Sequenz der genutzten KEs ausdrücken. Dabei wird jedes KE über dessen Bezeichnung (z.B. Rundung) und dessen Kategorie (z.B. konstruierend) beschrieben. In Abbildung 4.5 wird die Komponente zunächst durch das Erstellen zweier Körper und das anschließende Wegschneiden der Aushöhlung erzeugt. Zum anderen kann das Vorgehen über die zugehörige geometrische Entwicklung des 3D-Modells in Form der Zwischenzustände dargestellt werden. Diese entsprechen der Datenkategorie 3D-Geometrie und werden durch sukzessives Entfernen der KEs aus dem CAD-Modell erzeugt. Eine Zustandsänderung wird dabei definiert über eine Volumenänderung zwischen den jeweiligen Modellen zweier aufeinanderfolgender Zustände. Resultat ist eine Sequenz an Zwischenzuständen, die sich (geometrisch) durch das Hinzufügen eines (volumenändernden) KEs unterscheiden. Um welchen Typ von KE es sich dabei handelt, kann aus der entsprechenden Sequenz der genutzten KEs entnommen werden. Der letzte Zustand der erzeugten Sequenz an 3D-Geometrien entspricht dabei dem finalen Zustand und verkörpert folglich die 3D-Geometrie der Komponenten (siehe Kapitel 4.1.1).¹⁵ Das Konstruktionsvorgehen kann folglich über die Sequenz der 3D-Geometrie und die zugehörige Sequenz der KEs repräsentiert werden. Dabei stimmen die Sequenzlängen (siehe Attribut Sequenzlänge in Abbildung 4.3) und damit die Anzahl an Elementen in einer Sequenz für beide Darstellungsformen überein.

¹⁵ Da das Erstellen der entsprechenden Zwischenzustände zeitaufwändig ist, kann dieser Schritt in einem realen Anwendungsfall auch nur für einen Teil der Daten geschehen.

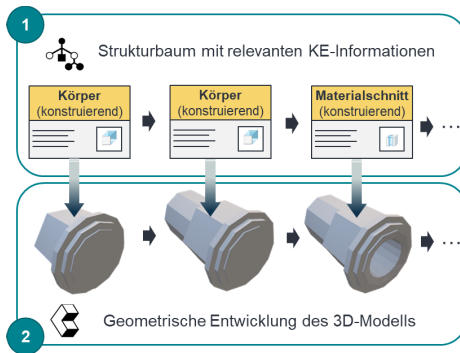


Abbildung 4.5: Darstellung des Konstruktionsvorgehens über die Sequenz der KEs (1) und die geometrische Entwicklung (2)

Zur Konstruktion eines Bauteils über KEs gibt es verschiedene Wege (Hackenschmidt, Hautsch & Kleinschrod 2020). Um aus dem Konstruktionsvorgehen mit Hilfe von Verfahren des MLs ein allgemeines Vorgehen erlernen zu können, ist es eine Grundvoraussetzung, dass Muster vorliegen. Als Konstruktionsmuster werden dabei typische Abfolgen von KEs innerhalb des Strukturbaumes definiert. Ein einfaches Beispiel für ein solches Muster ist die Sequenz Skizze-Extrusion. Im Rahmen einer Voruntersuchung anhand eines beispielhaften Datensatzes wurde daher basierend auf dem Strukturbaum zunächst überprüft, ob eine globale Ähnlichkeit im Konstruktionsvorgehen vorzufinden ist. Das Vorgehen hierzu wurde in der von der Autorin angeleiteten studentischen Arbeit A_Merklinger (2021) entwickelt. Hierfür wird der Strukturbaum jedes CAD-Modells als Graph modelliert. Jedes KE bildet dabei einen Knoten, die Abhängigkeiten zwischen den einzelnen KEs werden über die Kanten abgebildet. Die Ähnlichkeit dieser Graphen wird schließlich auf struktureller und auf kontextueller Ebene betrachtet. Dafür wird das von Nikolić (2012) entwickelte Verfahren des *Neighbourhood Matching* zur Bestimmung der strukturellen Ähnlichkeit von Graphen um den kontextuellen Aspekt erweitert. Die strukturelle Komponente betrachtet dabei, ob die Knoten die gleiche Anzahl an Vorgängern und Nachfolgern haben. Die kontextuelle Komponente bewertet, ob die Knoten das gleiche KE repräsentieren. Die kontextuelle Ähnlichkeit wird dabei als Gewichtungsfaktor der strukturellen Ähnlichkeit betrachtet. Ergebnis der Analyse ist eine Ähnlichkeitsmatrix mit Werten im Bereich $[0, 1]$, wobei der Wert 1 angibt, dass zwei Bauteile bezüglich ihres Strukturbaumes (nahezu) identisch sind. Basierend auf der Ähnlichkeitsmatrix wird schließlich ein dichtebasiertes Clusterverfahren (*DBSCAN*)

durchgeführt. Unter der Annahme, dass in den betrachteten Strukturbäumen wiederkehrende Muster vorliegen, wird erwartet, dass innerhalb der Daten Cluster aufzufinden sind, die ähnliche Strukturbäume beinhalten. Hierzu wurde die Untersuchung sowohl innerhalb einer Bauteilkategorie als auch über verschiedene Bauteilkategorien hinweg durchgeführt. Die Ergebnisse auf den in Kapitel 5 genutzten Daten sind in Anhang A zu finden. Ähnlichkeiten der Strukturbäume sind insbesondere innerhalb von Bauteilkategorien anzutreffen, wobei es auch hier teilweise mehrere Cluster und damit mehrere Vorgehensweisen gibt, ein Bauteil zu konstruieren. Wiederkehrende Muster können jedoch auch über verschiedene Bauteilkategorien hinweg aufgefunden werden.

Aus der Voruntersuchung kann schließlich gefolgert werden, dass offensichtlich Ähnlichkeiten zwischen den Strukturbäumen verschiedener CAD-Modelle – insbesondere innerhalb einer Bauteilkategorie – existieren und damit auch Muster im Konstruktionsvorgehen zu erwarten sind.

4.1.4 Anwendungsfallsspezifische Zuordnung der Umsetzungsstufen

Gemäß Abbildung 4.1 umfasst das Assistenzsystem grundsätzlich zwei Funktionalitäten, die Ähnlichkeitssuche sowie das Vorschlagen nächster Konstruktionsschritte. Die Bewertung produktionsrelevanter Produkteigenschaften beruht auf den gefundenen ähnlichen Modellen und wird daher als Teil der Ähnlichkeitssuche betrachtet. Je nach vorhandener Datenlage können für den betrachteten Anwendungsfall unterschiedliche Funktionalitäten des Konstruktionsassistenzsystems umgesetzt werden. Ausschlaggebend ist die Verfügbarkeit der eingeführten Datenkategorien je Komponente. Zur Bewertung der umsetzbaren Funktionalitäten kann das in Abbildung 4.6 dargestellte Reifegradmodell herangezogen werden.

Für die Ähnlichkeitssuche können verschiedene Umsetzungsstufen unterschieden werden. Grundvoraussetzung ist jedoch, dass Geometriedaten finaler Komponenten existieren. In der Basisstufe werden lediglich Daten der Kategorie 3D-Geometrie benötigt. Die Ähnlichkeit zwischen zwei CAD-Modellen wird in dieser Stufe rein auf der geometrischen Form bestimmt. Weitere Metainformationen, wie Material oder Abmessungen, sind nicht vorhanden und können daher nicht mit einbezogen werden. Ebenso ist aufgrund der fehlenden Informationen keine Bewertung der produktionsrelevanten Produkteigenschaften möglich. Sind zusätzliche Metainformationen vorhanden, so kann die Funktionalität der Ähnlichkeitssuche auf die nächste Stufe gehoben werden. Neben

der reinen geometrischen Form werden für die Ähnlichkeitsbestimmung die zusätzlichen Metainformationen genutzt. Produktionsrelevante Produkteigenschaften finden jedoch keine Berücksichtigung. In der dritten Ausbaustufe sind unter den Metainformationen auch solche, die als produktionsrelevant eingestuft wurden, sodass ein Abgleich der produktionsrelevanten Produkteigenschaften stattfinden kann.

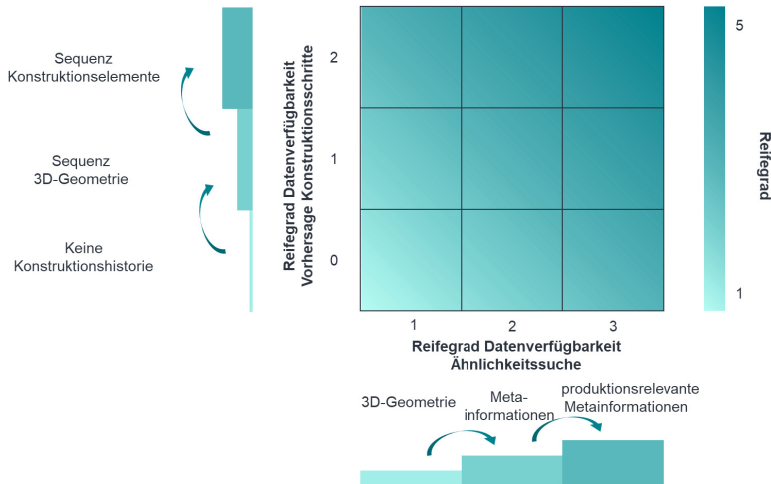


Abbildung 4.6: Reifegradmodell mit Umsetzungsstufen in Abhängigkeit der anwendungsfall-spezifischen Datenverfügbarkeit

Für das Vorschlagen nächster Konstruktionsschritte können zwei Ausbaustufen unterschieden werden. Grundvoraussetzung ist jedoch, dass Daten aus der Kategorie Konstruktionsvorgehen für die betrachteten Komponenten vorhanden sind. In der ersten Ausbaustufe ist dafür das Konstruktionsvorgehen in Form der Sequenzen von 3D-Geometrien ausreichend¹⁶. Sind zusätzlich Informationen über die genutzten KEs vorhanden, kann die Vorhersagegenauigkeit verbessert und damit Stufe 2 umgesetzt werden. Ähnlichkeitssuche und Vorschlagswesen sind zunächst als eigenständige Funktionen umsetzbar. Die Kombination beider Funktionalitäten ist jedoch über die verschiedenen Stufen möglich. Beispielsweise kann die Ähnlichkeitssuche in Stufe 1 mit dem Vor-

¹⁶ Sofern Daten zur Abbildung des Konstruktionsvorgehens über die Sequenz der 3D-Geometrie vorhanden sind, sind darin auch die finalen Bauteilzustände enthalten, die folglich die Datenkategorie 3D-Geometrie verkörpern.

schlagswesen in Stufe 2 kombiniert werden. Durch das Zusammenspiel beider Funktionen ergibt sich eine umfassendere Funktionalität, weshalb durch deren Kombination ein höherer Reifegrad von Stufe 3 erreicht werden kann. Sind Sequenzen der 3D-Geometrien vorhanden, so ist auch implizit durch den letzten Zustand der Sequenz der finale Bauteilzustand gegeben und somit Stufe 1 der Ähnlichkeitssuche umsetzbar. Zusätzlich kann die Ähnlichkeitssuche in allen Ausbaustufen durch die Nutzung der Konstruktionshistorie weiter verbessert werden (siehe Kapitel 4.3.2). Das Vorhandensein von Daten aus dem Konstruktionsvorgehen ist jedoch für die reine Ähnlichkeitssuche nicht notwendig, weshalb im Reifegradmodell für das Vorhersagen nächster Konstruktionsschritte zusätzlich eine Stufe 0 eingeführt wird.

4.2 Aufbereitung der Daten zur Repräsentation des Wissens

Um das Wissen aus den vorhandenen Datenkategorien mittels ML verarbeiten zu können, ist eine entsprechende Aufbereitung notwendig. Auf Basis dieser aufbereiteten Daten können dann die relevanten Informationen in einem nächsten Schritt extrahiert werden. Das so repräsentierte Wissen ist schließlich Grundlage für die Ähnlichkeitssuche und das Vorschlagswesen nächster Konstruktionsschritte. In den nachfolgenden Kapiteln wird die Aufbereitung der drei eingeführten Datenkategorien erläutert.

4.2.1 3D-Objekte

In der Literatur sind die in Kapitel 2.2.5 gezeigten Ansätze zur Darstellung von 3D-Objekten gängig. Welche Repräsentationsform am geeignetsten ist, hängt dabei von der spezifischen Anwendung ab (Xiao et al. 2020). Da für industrielle CAD-Daten aus dem Bereich des Maschinenbaus besondere Herausforderungen zu erfüllen sind, wurden im Rahmen dieser Arbeit die vielversprechendsten Repräsentationsformen im Zuge einer Voruntersuchung erprobt. Um zu testen, wie gut geometrische Eigenschaften mit den Repräsentationsformen dargestellt werden, wurde auf dem *AIAx CAD Dataset* (Tayyub et al. 2021) jeweils eine Klassifikation durchgeführt. Hierfür wurde der Ansatz nach Su et al. (2015) in der von der Autorin dieser Arbeit in Krahe et al. (2019) aufgezeigten Abwandlung für *Multi-View*, der Ansatz von Qi et al. (2017b) für Punktwolken, der Ansatz von Simonovsky & Komodakis (2017) für Graphen sowie der Ansatz von Hanocka et al. (2019) für *Meshes* herangezogen. Die Ergebnisse sind entsprechend im Anhang B zu finden. Die besten Ergebnisse wurden mit Punktwolken erzielt, weshalb diese im weiteren Verlauf der Arbeiten als Repräsentationsform ausgewählt wurden.

Die in Kapitel 4.1.1 beschriebenen Geometrie-Daten im STL-Format werden zur Weiterverarbeitung entsprechend in Punktwolken umgewandelt. Dafür werden Punkte auf der Oberfläche des 3D-Modells basierend auf dem in A_Schmutz (A_ 2021) und A_Holtzwarth (2022) entwickelten Verfahren verteilt. Grundlage für die Verteilung der Punkte ist die dem STL-Format zugrundeliegende Netzstruktur. Diese besteht aus kleinen Dreiecken, die jeweils über drei Eckpunkte und den Normalenvektor definiert sind (Roj 2016). Zur späteren Weiterverarbeitung werden Punktwolke mit einer festen Anzahl N an Punkten benötigt. Da die geometrischen Details insbesondere über die Menge der Eckpunkte V eines Objektes abgebildet werden, werden zunächst alle Eckpunkte über einen entsprechenden Punkt in der Punktwolke repräsentiert. Sofern die definierte Anzahl der Punkte N geringer ist als die Anzahl V der Eckpunkte, werden aus diesen zufällig N Punkte belegt. Ist die Anzahl der Eckpunkte V dagegen kleiner als die gewünschte Anzahl an Punkten N , so wird die Differenz $F = N - V$ auf den Dreiecksflächen verteilt. Die Wahrscheinlichkeit für die Auswahl einer Fläche erfolgt dabei anhand des Verhältnisses dieser Fläche zur Gesamtoberfläche. Größere Flächen erhalten damit eine höhere Wahrscheinlichkeit, mit einem Punkt versehen zu werden¹⁷. Die Platzierung des Punktes auf der gewählten Fläche erfolgt zufällig.

Im Vergleich zu den im Stand der Technik dargestellten öffentlichen Datensätzen sind reale CAD-Modelle aus der Industrie in der Regel nicht normiert und zentriert. Da dies jedoch für die herangezogenen *Autoencoder* (siehe Kapitel 4.3) vorausgesetzt wird (Sun et al. 2021), müssen die CAD-Modelle in Form der zuvor erstellten Punktwolken zunächst entsprechend vorverarbeitet werden.¹⁸ Im Rahmen der Normierung werden die erstellten Punktwolken auf Basis der maximalen Ausdehnung in x-, y- und z-Richtung zunächst auf den Einheitswürfel der Größe $1 \times 1 \times 1$ skaliert. Die anschließende Zentrierung verschiebt den Mittelpunkt jeder Punktwolke auf den Ursprung des Koordinatensystems. Abbildung 4.7 fasst die notwendigen Schritte nochmals zusammen. Als Ergebnis liegen folglich sämtliche Objekte der Kategorie 3D-Geometrie als normierte und zentrierte Punktwolke vor. Das optionale Attribut Klasse, das der Datenka-

¹⁷ Die Wahrscheinlichkeitsberechnung entspricht einem Experiment der Form Ziehen mit Zurücklegen, damit große Flächen auch mit mehr als nur einem Punkt belegt werden können. Darüber hinaus existieren auch Modelle, die weniger Flächen als zu verteilende Punkte besitzen.

¹⁸ In Voruntersuchungen im Rahmen der von der Autorin angeführten Arbeit A_Leoni (2020) konnte gezeigt werden, dass ohne vorherige Normierung und Zentrierung keine guten Trainingsergebnisse erreicht werden.

torie 3D-Geometrie angehängt sein kann, ist kategorischer Natur und wird entsprechend Kapitel 4.2.2 aufbereitet. Die Punktwolken dienen schließlich als Grundlage für die in Kapitel 4.3 beschriebene Extraktion der relevanten Merkmale.

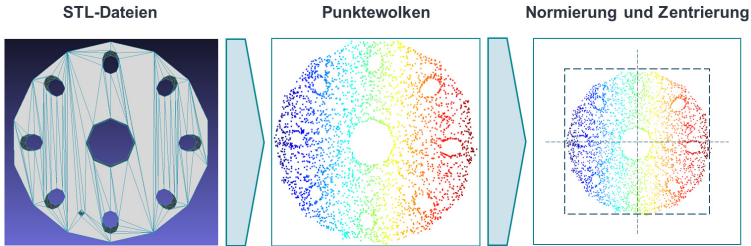


Abbildung 4.7: Vorverarbeitungsschritte für die 3D-Geometrie (in Anlehnung an A_Schmutz (2021))

4.2.2 Kategorische und numerische Daten

Gemäß Kapitel 4.1.2 können die extrahierten Metainformationen kategorischer (z.B. Material) und numerischer Natur (z.B. maximale Abmessungen) sein. ML-Verfahren benötigen im Allgemeinen Daten in numerischer Form (Hirschle 2020). Kategorische Daten müssen dementsprechend vorverarbeitet werden.

Zum Abbilden kategorischer Daten ist ein *One-Hot-Encoding* gängig (Hirschle 2020). Voraussetzung hierfür ist eine feste Anzahl an Kategorien. Für eine Anzahl von X Kategorien werden X binäre Variablen erzeugt. Für jedes Datenobjekt wird beim Zutreffen der Kategorie die entsprechende binäre Variable auf 1 gesetzt, ansonsten auf 0. Zur Darstellung einer kategorischen Variable ergibt sich schließlich ein Vektor der Länge X . (Hirschle 2020; Chollet 2018) Abbildung 4.8 veranschaulicht dies für das kategorische Attribut „Klasse“ einer 3D-Geometrie.

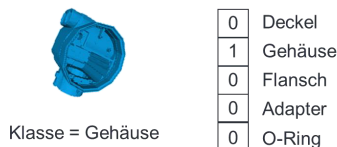


Abbildung 4.8: Beispielhaftes One-Hot-Encoding des kategorischen Attributs „Klasse“

Da die meisten numerischen Daten dimensionsbehaftet sind, ist zunächst zwingend eine Transformation notwendig (Otte et al. 2020). Insbesondere zur Messung von Distanzen, wie es beispielsweise für die in Kapitel 4.4.1 gezeigte Clusterbildung notwendig

ist, ist eine solche Transformation auf einen einheitlichen Wertebereich unabdingbar (Mohamad & Usman 2013; Han, Kamber & Pei 2012). Andernfalls haben numerische Attribute mit hohen Wertebereichen grundlegend mehr Einfluss als Attribute aus niedrigen Wertebereichen (Otte et al. 2020; Studer et al. 2021). Zudem sollten die entsprechenden Werte dimensionslos, also unabhängig von der gewählten Maßeinheit sein (Otte et al. 2020). Üblicherweise werden die Daten hierfür normiert bzw. standardisiert¹⁹ (Otte et al. 2020; Han, Kamber & Pei 2012). Eine allgemeingültige Regel, welche Vorgehensweise zur Normierung genutzt werden sollte, existiert jedoch nicht (Mohamad & Usman 2013). Zu den gängigen Methoden zählt unter anderem die Min-Max-Normierung (Han, Kamber & Pei 2012), die in dieser Arbeit genutzt wird. Hierbei wird ein Wert einer Metainformation y_w anhand der minimalen $y_{w,min}$ und maximalen $y_{w,max}$ Merkmalsausprägung auf den Bereich $[0,1]$ transformiert (Otte et al. 2020):

$$y_{w,normiert} = \frac{y_w - y_{w,min}}{y_{w,max} - y_{w,min}} \quad 4-1$$

Nachfolgend bezeichnet y_w die normierte Metainformation.

Als Ergebnis der Aufbereitung liegen sämtliche kategorische Metainformationen in Form eines *One-Hot-Encodings* sowie sämtliche numerische Metainformationen normiert auf den Bereich $[0,1]$ vor.

4.2.3 Sequentielle Daten

Als sequentielle Daten werden im Rahmen dieser Arbeit die zwei in Kapitel 4.1.3 vorgestellten Möglichkeiten zum Abbilden des Konstruktionsvorgehens betrachtet. Die Sequenz der genutzten KEs stellt dabei eine Sequenz kategorialer Daten dar. Für den betrachteten Datensatz wird zunächst die Anzahl der genutzten Konstruktionselementtypen identifiziert. In Abhängigkeit davon wird gemäß Kapitel 4.2.2 für jeden Typ Konstruktionselement eine binäre Variable definiert. Entsprechend ergibt sich eine Sequenz von kategorialen Daten, die jeweils über ein *One-Hot-Encoding* repräsentiert werden (siehe Abbildung 4.9 oben). Die entsprechende Sequenz der geometrischen Entwicklung des 3D-Modells setzt sich jeweils aus Elementen zusammen, die eine 3D-Geometrie in Form der Zwischenzustände repräsentieren. Entsprechend Kapitel 4.2.1 werden diese 3D-Geometrien in Punktwolken umgewandelt, normiert und zentriert.

¹⁹ Die Begriffe Standardisieren und Normalisieren werden in der Datenvorverarbeitung synonym verwendet, obwohl der letztere Begriff in der Statistik auch andere Bedeutungen hat (Han, Kamber & Pei 2012).

Die geometrische Entwicklung der Komponente wird somit durch eine Sequenz an Punktwolken dargestellt (siehe Abbildung 4.9 unten). Mithilfe der im nachfolgenden Kapitel 4.3 aufgezeigten Verfahren können schließlich aus der Sequenz der Punktwolken die relevanten Eigenschaften je Zwischenstand extrahiert werden.

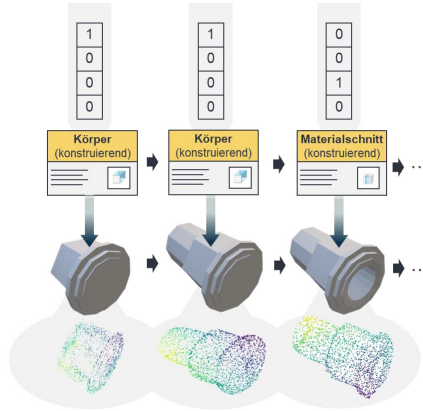


Abbildung 4.9: Aufbereitung der Konstruktionssequenzen

Für die Analyse der Sequenzen mittels der in Kapitel 4.4.2 beschriebenen Verfahren ist ein weiterer Aufbereitungsschritt notwendig. Da sich die Anzahl notwendiger Konstruktionsschritte und damit die Sequenzlängen von verschiedenen Komponenten in der Regel unterscheiden, jedoch für das spätere Vorgehen feste Sequenzlängen notwendig sind, müssen diese zunächst auf eine einheitliche Länge gebracht werden. Intuitiv könnten hierfür Sequenzen, die eine definierte Länge überschreiten, abgeschnitten, sowie Sequenzen, die diese Länge unterschreiten, verworfen werden. Bei sehr stark variierenden Sequenzlängen würde dieses Vorgehen jedoch zu einem starken Datenverlust führen. Eine weitere Möglichkeit bietet das sogenannte *Zero-Padding* (Goodfellow, Bengio & Courville 2016). Hier werden alle Sequenzen, deren Länge kürzer ist als die maximale Länge des betrachteten Datensatzes, durch das Hinzufügen von Einträgen mit dem Wert 0 aufgefüllt. Vorteil ist, dass somit der gesamte Datensatz genutzt werden kann. Nachteil ist jedoch, dass dieses Vorgehen in sehr langen Sequenzen resultiert, was sich negativ auf die Kapazität des internen Speichers von RNNs auswirken kann (vgl. Kapitel 2.2.4). Eine dritte Möglichkeit bietet das sogenannte *Windowing bzw. Sliding Window* (Russell & Norvig 2022). Über ein Fenster (engl. *Window*) definierter

Größe wird damit eine Sequenz in Untersequenzen fester Größe aufgeteilt. Die Funktionsweise wird in Abbildung 4.10 veranschaulicht. Im dargestellten Beispiel wird die Fenstergröße Q auf 3 gesetzt. Aus der ursprünglichen Sequenz der Länge SL von 5 werden durch Verschieben (engl. *Sliding*) des Fensters 3 Sequenzabschnitte gebildet. Die Länge der resultierenden Sequenzabschnitte entspricht der Fenstergröße Q .



Abbildung 4.10: Vorgehensweise zur Bildung von Sequenzabschnitten über ein Sliding Window

Neben der Verarbeitung unterschiedlicher Sequenzlängen wird dadurch außerdem der Trainingsdatensatz vergrößert. Lediglich Sequenzen mit einer Länge kleiner der Fenstergröße müssen für das Training verworfen werden²⁰. Dadurch entsteht unter anderem der Vorteil, dass Elemente mehrfach innerhalb verschiedener Sequenzabschnitte und damit Kontexte verarbeitet werden, wodurch die Generalisierbarkeit des RNNs gesteigert werden kann²¹. Die Fenstergröße ist dabei ausschlaggebend dafür, wie viel Konstruktionsschritte das RNN in der späteren Anwendung zurückschauen muss, um eine valide Vorhersage zu machen. Je kleiner die Fenstergröße gewählt wird, desto allgemeinere Muster können aus den Daten erlernt werden. Im Gegenzug werden die Vorhersagen jedoch unspezifischer. Je größer die Fenstergröße, desto spezifischer werden die erlernten Muster in Bezug auf die zugrunde gelegten Trainingsdaten. Allerdings können so allgemeine Muster schlechter erlernt werden. Die Fenstergröße ist folglich ein Hyperparameter und sollte in Abhängigkeit des zugrunde gelegten Datensatzes gewählt werden. Dabei ist darauf zu achten, dass durch die Wahl der Fenstergröße die Datenbasis nicht zu sehr verringert wird, da, wie oben beschrieben, kürzere Sequenzen verworfen werden.

²⁰ In der Anwendung kann das RNN dennoch auch kürzere Sequenzabschnitte verarbeiten.

²¹ Lee, T.-P. & Narayan, T. (2016), *Distributed Language Models Using RNNs* https://stanford.edu/~rezab/classes/cme323/S16/projects_reports/narayan_lee.pdf (aufgerufen am 27.09.2022)

Als Resultat werden folglich aus den Sequenzen der 3D-Geometrie sowie der zugehörigen Sequenzen der KEs Sequenzabschnitte gebildet, die schließlich in Kapitel 4.4.2 als Datengrundlage für das RNN dienen.

4.3 Automatisierte Extraktion relevanter Eigenschaften mittels Autoencoder

Im Gegensatz zu numerischen oder kategorischen Daten sind 3D-Daten sehr komplex und nur schwer bis gar nicht über vordefinierte Merkmale beschreibbar. Daher werden in dieser Arbeit ML-Verfahren dazu genutzt, entsprechende Merkmale selbstständig aus den Daten im Rahmen eines *Representation Learning* zu extrahieren. Ein gängiges Verfahren hierfür sind *Autoencoder* (vgl. Kapitel 2.2). Deren Eignung zum selbstständigen Erlernen relevanter Merkmale aus 3D-Objekten zeigt deren sehr breite Anwendung in der Literatur (Park et al. 2019; Pang, Li & Tian 2021) sowie die großen Fortschritte bezüglich Effizienz und Leistungsstärke, die erzielt werden konnten (Tchapmi et al. 2019). Gemäß Kapitel 2.2.2 sind *Autoencoder* in der Lage, eine komprimierte und effiziente Codierung eines Inputs, einen sogenannten latenten Vektor, zu erlernen. Im Fall von 3D-Objekten charakterisiert dieser Vektor folglich die relevanten geometrischen Eigenschaften (Pang, Li & Tian 2021). Sind die relevanten Merkmale zweier Inputs sehr ähnlich, wird angenommen, dass daraus entsprechend auch ähnliche latente Repräsentationen resultieren (diese Annahme wird in Kapitel 4.3.3 für den jeweiligen Anwendungsfall überprüft). Die latenten Vektoren weisen folglich im latenten Raum geringe Distanzen auf. Im Rahmen dieser Arbeit dienen als Input die gemäß Kapitel 4.2.1 aufbereiteten 3D-Objekte in Form von Punktwolken. Geometrisch ähnliche Modelle resultieren folglich in latenten Vektoren, die geringe Distanzen aufweisen. Zum Erlernen der latenten Repräsentationen aus den 3D-Geometrien werden dabei verschiedene *Autoencoder*-Architekturen aufgezeigt, die in Kapitel 4.3.1 vorgestellt werden. Voraussetzung für die Anwendung ist mindestens das Vorhandensein der finalen 3D-Geometrie der Komponenten. Die erlernten Repräsentationen dienen schließlich als Grundlage für die Ähnlichkeitssuche (siehe Kapitel 4.4.1). Sind darüber hinaus Daten zum Konstruktionsvorgehen vorhanden, so werden als Input zusätzlich die einzelnen Zwischenstände aus den Sequenzen der 3D-Geometrien genutzt, worauf in Kapitel 4.3.2 näher eingegangen wird. Da die Sequenzen des Konstruktionsvorgehens auch die finalen Zustände der Objekte beinhalten, können die damit erlernten Repräsentationen sowohl für die Ähnlichkeitssuche (siehe Kapitel 4.4.1) als auch für das Vorschlagen nächster

Konstruktionsschritte (siehe Kapitel 4.4.2) sowie deren Zusammenspiel (siehe Kapitel 4.4.3) genutzt werden. In beiden Fällen wird der Datensatz in einen Trainings- und Testdatensatz aufgeteilt. Das Vorgehen wird in Abbildung 4.11 nochmals dargestellt.

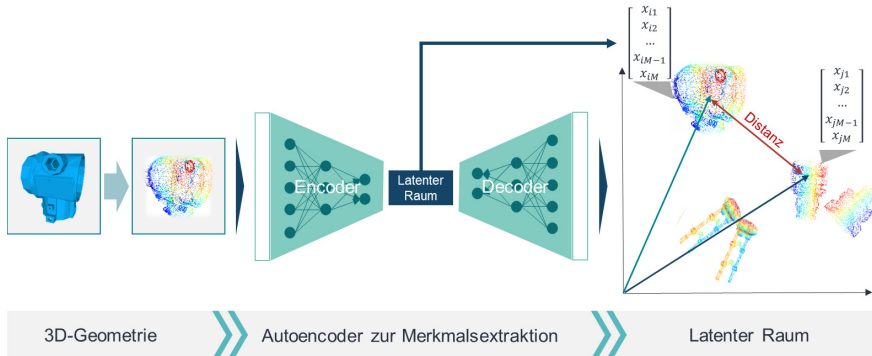


Abbildung 4.11: Prinzipielle Vorgehensweise zur Extraktion relevanter Eigenschaften aus den Punktwolken mittels Autoencoder

Um die Repräsentativität und Vollständigkeit des erlernten latenten Raumes zu überprüfen und die verschiedenen *Autoencoder*-Architekturen diesbezüglich bewerten zu können, werden in Kapitel 4.3.3 verschiedene Experimente und Untersuchungen vorgestellt. Ergebnis ist schließlich die Auswahl der für den Anwendungsfall geeignetsten *Autoencoder*-Architektur, über die die 3D-Geometrien am besten dargestellt werden. Der trainierte *Encoder* dieser Architektur wird schließlich genutzt, um die vorhandenen 3D-Geometrien über einen entsprechenden latenten Vektor, der gemeinsam mit den Metainformationen den geometrischen Fußabdruck einer Komponenten bildet, für die Weiterverarbeitung im Konstruktionsassistenzsystem (Kapitel 4.4) darzustellen.

4.3.1 Autoencoder-Architekturen für 3D-Geometrien

Der grundsätzliche Aufbau des *Autoencoders* sowie dessen Funktionalität ist in der nachfolgenden Abbildung 4.12 visualisiert. Eine detailliertere Darstellung ist in Anhang C.1 zu finden. Die herangezogene Architektur des *Autoencoders* nach Krahe et al. (2022) basiert auf Yang et al. (2018). Input für den *Encoder* sind die 3D-Geometrien in Form von Punktwolken. Die Punktwolken werden dabei als eine Liste von Tripeln bestehend aus x -, y - und z -Koordinate (in Abbildung 4.12 sind es N Tripel ($N \times 3$)) übergeben. Der Output des *Decoders* ist ebenfalls eine Liste von Koordinaten-Tripeln, die die rekonstruierte Punktwolke darstellen. Ziel des *Encoders* ist es, die Dimensionalität

des Inputs auf die Größe der sogenannten *Bottleneck Size* (BNS), in Abbildung 4.12 also $M \times 1$, zu reduzieren. Die Punktwolke kann somit als ein Vektor mit den Einträgen $m=1, \dots, M$ dargestellt werden.

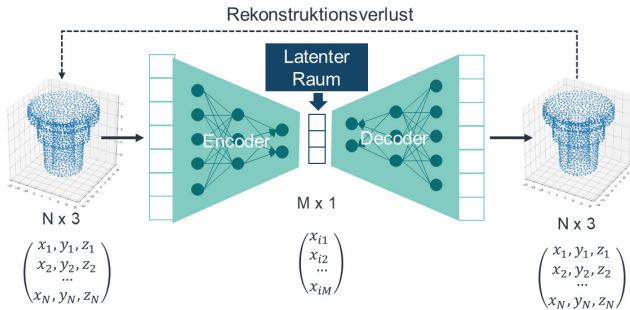


Abbildung 4.12: Grundsätzlicher Aufbau und Funktionalität des Autoencoders für Punktwolken

Die Größe der BNS ist dabei ausschlaggebend für die Menge an Merkmalen, die zur Beschreibung der 3D-Geometrie genutzt werden und damit für die Repräsentativität der latenten Vektoren (Peste, Malagó & Sárbu 2017). Da es keine spezifische Reihenfolge der n Punkte in der Liste gibt, ist es nicht möglich, übliche Faltungsschichten (engl. *Convolutional Layer*) aus der Bildverarbeitung anzuwenden. Gemäß Yang et al. (2018) wird daher eine Kombination aus MLPs und graphbasierten *Max-Pooling*-Schichten verwendet, wodurch lokale Eigenschaften der Punktwolken erfasst werden können. Der *Decoder* transformiert bildlich ausgedrückt über mehrere gemäß Yang et al. (2018) definierte Faltungsoperationen eine Sphäre dergestalt, dass sie die ursprüngliche Punktwolke repräsentiert. Die latenten Vektoren, die der *Decoder* als Input erhält, repräsentieren dabei die Krafteinwirkung, die notwendig ist, um die Sphäre zu verformen. Die genaue Architektur der neuronalen Schichten kann Anhang C.1 entnommen werden.

Während des Trainings erhält der *Encoder* paketweise Punktwolken (sogenannte *Batches*), die zunächst auf die Größe der BNS reduziert und anschließend vom *Decoder* wieder rekonstruiert werden. Die Rekonstruktionen eines *Batches* werden schließlich mit den originalen Punktwolken verglichen. Der Fehler zwischen Rekonstruktion und ursprünglichem Input, nachfolgend auch als Rekonstruktionsverlust bezeichnet, wird dabei im 3D-Raum über die *Chamfer Distance* (CD) (Barrow et al. 1977) bestimmt und die Gewichte des neuronalen Netzes entsprechend via *Back Propagation* angepasst.

Ein weiteres mögliches Distanzmaß für Punktwolken bildet die *Earth Mover Distance* (Rubner, Tomasi & Guibas 2004). Da jedoch die CD nach Achlioptas et al. (2018) effizienter zu berechnen ist und zudem in der zugrunde liegenden Architektur von Yang et al. (2018) verwendet wird, wird diese nachfolgend zur Bestimmung des Rekonstruktionsverlusts herangezogen. Nach Achlioptas et al. (2018) wird die CD zwischen zwei Punktwolken $P_1 \subseteq \mathbb{R}^3$ und $P_2 \subseteq \mathbb{R}^3$ wie folgt definiert:

$$d_{CD}(P_1, P_2) = \sum_{x \in P_1} \min_{y \in P_2} \|x - y\|_2^2 + \sum_{y \in P_2} \min_{x \in P_1} \|x - y\|_2^2 \quad 4-2$$

In welchem Umfang die Anpassung der Netzgewichte des berechneten Fehlers erfolgt, wird über die Lernrate λ festgelegt. Eine Trainingsepoche umfasst das Durchlaufen aller Datenpunkte aus dem Trainingsdatensatz. Die Anzahl notwendiger Trainingsepochen ist abhängig vom Anwendungsfall und zählt zu den Hyperparametern, die optimiert werden können. Anhang C.2 fasst die Hyperparameter zusammen, die während des Trainings variiert werden können, um die erlernte Repräsentation der Input-Punktwolken für den jeweiligen Anwendungsfall zu optimieren.

In der Testphase kann der *Encoder* mit den erlernten Netzgewichten dazu genutzt werden, für eine gegebene Input-Punktwolke einen latenten Vektor in Größe der BNS zu erzeugen. Dieser Vektor repräsentiert die Geometrie der Input-Punktwolke und dient als Basis für die Ähnlichkeitssuche und die Vorhersage nächster Konstruktionsschritte.

Wie im Stand der Technik (Kapitel 3) gezeigt, wird ein Großteil der bisherigen Anwendungen lediglich auf Datensätzen angewandt, die aus einfachen und vorausgerichteten 3D-Objekten bestehen. Da im betrachteten Anwendungsgebiet mit mechanischen Komponenten aus dem Bereich des Maschinenbaus diese Voraussetzungen in der Regel nicht gegeben sind, werden nachfolgend drei Ansätze betrachtet, die jeweils auf der Basis-Architektur des *Autoencoders* beruhen und je nach Anwendungsfall zu wählen sind:

- Ansatz 1: *Autoencoder* ohne Standardorientierung der Input-Punktwolke
- Ansatz 2: *Autoencoder* mit Standardorientierung der Input-Punktwolke
- Ansatz 3: *Autoencoder* ohne/ mit Standardorientierung der Input-Punktwolke und zusätzlichem Klassifikationsverlust

Ansatz 1 umfasst die einfachste Variante, da hier keine weitere Vorverarbeitung in Form einer Ausrichtung der Input-Daten notwendig ist. Als Input dienen folglich die Punktwolken der 3D-Objekte, die gemäß Kapitel 4.2.1 zuvor normiert und zentriert wurden. Sind die 3D-Objekte der betrachteten Datenbasis jedoch nicht (größtenteils) einheitlich vorausgerichtet, besteht die Gefahr, dass keine geeignete Repräsentation erlernt werden kann (Sun et al. 2021). In Ansatz 2 werden daher die 3D-Objekte zunächst vor dem Training des *Autoencoders* standardmäßig ausgerichtet. Als *Loss*-Funktion wird in den Ansätzen 1 und 2 der Rekonstruktionsverlust verwendet. Um verschiedene Bauteilklassen trotz möglicher geometrischer Ähnlichkeiten im latenten Raum noch besser voneinander abgrenzen zu können, wird in Ansatz 3 ein zusätzlicher Klassifikationsverlust eingeführt. Dieser kann als weiterer Optimierungsstellhebel auf den besten Ansatz (Ansatz 1 oder 2) angewandt werden, sofern zumindest ein Teil der Daten über ein Klassen-Label verfügt²². Nachfolgend werden die vorgestellten Ansätze näher beschrieben.

4.3.1.1 Autoencoder mit Standardausrichtung

Die für diesen Ansatz verwendete *Autoencoder*-Architektur entspricht der zuvor dargestellten Basis-Architektur. Optimierungsstellhebel ist hier die Vorverarbeitung der Input-Punktwolken, sodass diese einheitlich vorausgerichtet sind²³. Der dafür herangezogene Algorithmus basiert auf Krahe et al. (2019). Grundlage für die Standardausrichtung ist der minimal umhüllende Quader (engl. *Bounding Box*) eines jeden Objektes. Basierend auf dieser *Bounding Box* wird die Orientierung der Objekte über fünf Kriterien bestimmt: Der Mittelpunkt der *Bounding Box* liegt im Ursprung des Koordinatensystems (1), die kürzeste Seite der *Bounding Box* ist in y-Richtung ausgerichtet (2), die längste Seite der *Bounding Box* liegt parallel zur x-Achse (3), das Punktzentrum des Polygonnetzes, welches das 3D-Objekt darstellt, hat einen Wert der y-Koordinate (4) und der z-Koordinate (5) kleiner oder gleich 0. Das Prinzip wird in Abbildung 4.13 anhand eines in Standardorientierung dargestellten Quaders mit seitlicher Einbuchtung visualisiert.

Die so standardausgerichteten Objekte dienen nun als Trainings-Input für den *Autoencoder*. Ungesehene Objekte, die während der Testphase vom *Autoencoder* verarbeitet werden, werden zuvor ebenfalls standardausgerichtet.

²² Das Attribut „Klasse“ der Kategorie 3D-Geometrie muss folglich belegt sein.

²³ Im Rahmen einer Voruntersuchung in A. Schmutz (2021) wurde gezeigt, dass die künstliche Erweiterung des Datensatzes durch Rotation der Objekte (*Data Augmentation*) keine bessere Generalisierbarkeit bezüglich unterschiedlich ausgerichteter Komponenten bezwecken konnte.

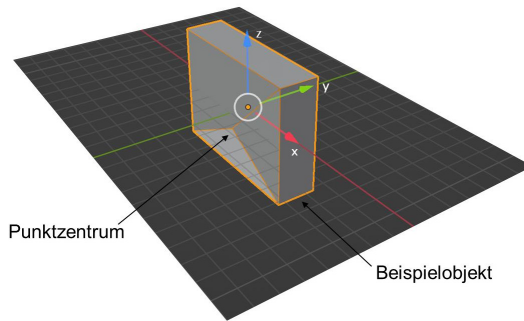


Abbildung 4.13: Prinzip der Standardorientierung (in Anlehnung an Krahe et al. (2019))

4.3.1.2 Autoencoder mit zusätzlichem Klassifikationsverlust

Um im latenten Raum noch besser zwischen unterschiedlichen Bauteilklassen trotz ähnlicher Geometrie unterscheiden zu können, wird die Basis-Architektur des *Autoencoders* um eine weitere Verlustfunktion, den Klassifizierungsverlust, erweitert. Je nachdem, ob gelabelte Daten (auch nur im geringen Maße) verfügbar sind, also gemäß Kapitel 4.1.1 das Attribut Klassenlabel einer 3D-Geometrie belegt ist, kann diese zusätzliche Verlustfunktion aktiviert oder deaktiviert werden. Die Architektur selbst wird dabei um eine einfache MLP Architektur erweitert, die während des Trainings dahingehend trainiert wird, basierend auf den latenten Repräsentationen die entsprechende Klasse vorherzusagen. Abbildung 4.14 veranschaulicht die prinzipielle Architektur dieses *Autoencoders*.

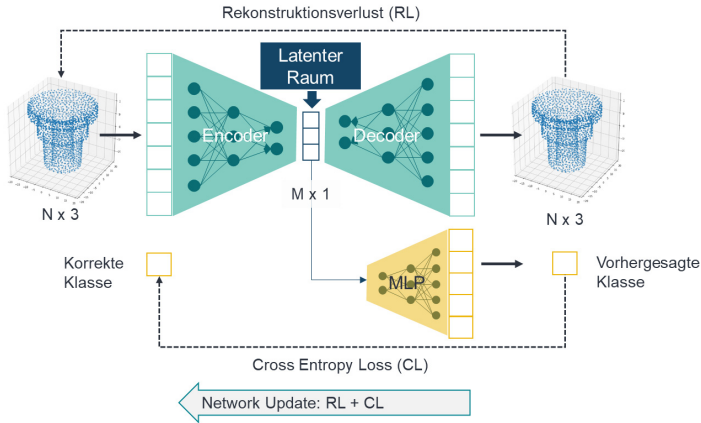


Abbildung 4.14: Architektur des Autoencoders mit zusätzlichem Klassifikationsverlust

Die Kombination aus *supervised* und *unsupervised Learning* wird auch als *semisupervised Learning* bezeichnet und wird häufig angewandt, wenn nur eine geringe Anzahl der Daten über Labels verfügt (Russell & Norvig 2022). Die Größe der Ausgabeschicht des MLP entspricht dabei der Anzahl möglicher Klassen. Für den gegebenen Input-Vektor wird letztendlich die Wahrscheinlichkeit für jede Klasse ausgegeben. Als Verlustfunktion wird dabei die Kreuzentropie (engl. *Cross Entropy Loss*) herangezogen, die die Differenz der empirischen Verteilung, die durch die Trainingsdatenmenge gegeben ist, und der Wahrscheinlichkeitsverteilung, die durch das MLP Modell definiert wird, darstellt (Goodfellow, Bengio & Courville 2016; Russell & Norvig 2022). Dadurch wird die durch den *Autoencoder* erlernte Repräsentation dahingehend optimiert, unterschiedliche Bauteilklassen noch besser voneinander zu differenzieren. In der Trainingsphase wird die Architektur zunächst auf den gelabelten Daten trainiert. Die Verlustfunktion setzt sich in dieser Phase aus der gewichteten Summe von Rekonstruktions- und Klassifikationsverlust (also Kreuzentropie) zusammen, auf deren Basis die Netzgewichte des *Encoders* angepasst werden. Die Anpassungen der Netzgewichte des *Decoders* erfolgt nach wie vor rein auf dem Rekonstruktionsverlust. Das MLP dagegen wird ausschließlich über den Klassifikationsverlust angepasst. In der zweiten Phase des Trainings wird der Klassifikationsverlust deaktiviert und *Encoder* und *Decoder* nur noch auf Basis des Rekonstruktionsverlusts aktualisiert. Über die Lernrate kann dabei

beeinflusst werden, wie stark die Anpassungen in dieser Trainingsphase noch Änderungen an der bereits erlernten Repräsentation bewirken sollen. Ziel ist es, die in Phase 1 auf den gelabelten Daten erlernte und für die Klassifikation optimierte Repräsentation auf die ungelabelten Daten aus Phase 2 zu übertragen. Daher sollte die Lernrate in der zweiten Phase deutlich kleiner gewählt werden, um nur noch geringfügigere Anpassungen zu bewirken.

4.3.2 Autoencoder für sequentielle Daten des Konstruktionsvorgehens

Das Vorgehen, um neben der 3D-Geometrie der finalen Komponenten auch die zugehörigen 3D-Geometrien aller Zwischenzustände aus der Konstruktionshistorie mittels des *Autoencoders* verarbeiten zu können, entspricht prinzipiell der in Kapitel 4.3.1 dargestellten Methodik. Nachfolgend werden daher nur die Besonderheiten aufgeführt.

Als Input für das Training aller *Autoencoder*-Architekturen (Ansätze 1 bis 3) dienen nicht nur die Punktwolke des finalen Bauteilzustandes, sondern auch die Punktwolken aller zugehörigen Zwischenzustände. Die Abhängigkeiten zwischen den einzelnen Zwischenständen, die über die Reihenfolge innerhalb der Sequenzen dargestellt werden, werden hierfür aufgehoben. Trainings-Input sind folglich die einzelnen Punktwolken aller Zwischenzustände inklusive finale Zustand einer Komponenten.

Im Unterschied zu Ansatz 2 (Kapitel 4.3.1.1) wird zunächst nur der finale Konstruktions-schritt standardausgerichtet. Die entsprechenden Transformationen zur Überführung in die Standardorientierung werden schließlich auf allen weiteren Konstruktions-schritten durchgeführt. Dadurch soll verhindert werden, dass Zwischenzustände eines Konstruktionsvorgehens unterschiedlich ausgerichtet sind.

Die in Ansatz 3 für einen Teil des Datensatzes notwendigen Klassen-Label können für die Zwischenzustände aus der zugehörigen finalen Komponente abgeleitet werden. Es ist jedoch auch möglich, lediglich finale Komponenten für die erste Phase des Trainings mit zusätzlichem Klassifikationsverlust zu nutzen und alle zugehörigen Zwischenzustände erst in Phase 2 einfließen zu lassen.

Im Vergleich zum reinen Training auf den finalen 3D-Geometrien ergibt sich durch das Training auf den zusätzlichen Zwischenzuständen der Vorteil, dass auch halbfertige Komponenten, die einem Zwischenzustand entsprechen, dem *Autoencoder* bekannt

sind und somit in eine geeignete Repräsentation gewandelt werden können. Insbesondere für die Ähnlichkeitssuche auf anfänglichen Konstruktionsständen ist das vorteilhaft (siehe Kapitel 4.4.1.2).

Für die spätere Verarbeitung der Sequenzabschnitte (siehe Kapitel 4.2.3) mittels des RNN können nun die entsprechenden Zwischenzustände eines Sequenzabschnittes mit Hilfe der auf den Trainingsdaten trainierten *Autoencoder*-Architektur über den *Encoder* in einen latenten Vektor überführt werden. Die Sequenzabschnitte des Konstruktionsvorgehens werden folglich, wie in Abbildung 4.15 visualisiert, als Sequenz der entsprechenden latenten Vektoren abgebildet.

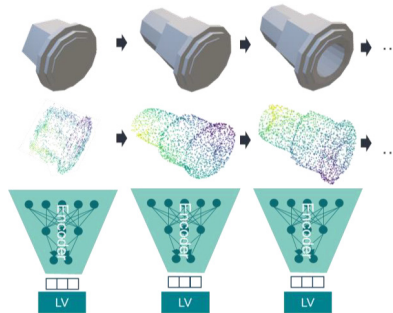


Abbildung 4.15: Darstellung des Konstruktionsvorgehens in Form der Sequenz der 3D-Geometrie über die Sequenz entsprechender latenter Vektoren (LV)

4.3.3 Bewertung der Repräsentativität des latenten Raumes

Um zu entscheiden, welche *Autoencoder*-Architektur für den jeweiligen Anwendungsfall am geeignetsten ist, kann die Repräsentativität der erlernten latenten Repräsentationen anhand mehrerer Untersuchungen bewertet werden (vgl. Kapitel 2.2.2.2). Unter Repräsentativität wird dabei definiert, wie gut die erlernten Darstellungen die tatsächliche Geometrie in Form relevanter Merkmale abbilden und somit für die spätere Anwendung im Konstruktionsassistenzsystem geeignet sind.

Um zunächst eine Vorstellung davon zu bekommen, wie die latenten Vektoren im latenten Raum angeordnet sind, kann das Verfahren t-SNE nach van der Maaten & Hinton (2008) herangezogen werden. Unter der Annahme, dass die Geometrien von Objekten derselben Bauteilkategorie ähnlich sind, können beispielsweise latente Vektoren von Objekten verschiedener Bauteilklassen über dieses Verfahren visualisiert werden. Darauf basierend kann überprüft werden, ob die ersichtlichen Cluster im latenten Raum

auch den tatsächlichen Klassen entsprechen und somit geometrische Unterschiede zwischen den Bauteilklassen offensichtlich abgebildet werden. Sind keine Klassen-Labels vorhanden, kann dennoch über die Visualisierung festgestellt werden, ob in den Daten globale Strukturen, wie z.B. Cluster, zu erkennen sind. Allerdings ist durch die starke Reduzierung der Dimensionen mit einem gewissen Informationsverlust zu rechnen (van der Maaten & Hinton 2008), weshalb die Visualisierung selbst nicht als Grundlage für eine Clusterbildung herangezogen werden sollte.

Sind zusätzlich Klassen-Labels bekannt, kann neben der rein visuellen Überprüfung basierend auf den latenten Vektoren eine Clusteranalyse oder eine Klassifikation vorgenommen werden. Für die Clusterbildung, z.B. über *k-means* (vgl. Kapitel 2.2.3.1), ist zu erwarten, dass die gefundenen Cluster weitestgehend den Klassen entsprechen, sofern Objekte einer Bauteilkategorie geometrische Ähnlichkeiten aufweisen²⁴. Als Bewertungsgröße für die Cluster-Güte kann das V-Maß (vgl. Kapitel 2.2.3.3) herangezogen werden. Wird eine Klassifikation durchgeführt, z.B. über einen *Random Forest Classifier* oder eine *Support Vector Machine*, kann deren Güte anhand der *Accuracy* bemessen werden (Aggarwal 2021). Diese gibt den Anteil der Testdaten an, für die die vorhergesagte Klasse der tatsächlichen Klasse entspricht (Aggarwal 2021).

Eine weitere Möglichkeit, die Repräsentativität der latenten Vektoren zu zeigen, ist die Interpolation. Hierfür werden gemäß Abbildung 4.16 zwei latente Vektoren \vec{x}_1 und \vec{x}_2 von Objekten herangezogen²⁵. Auf dem Differenzvektor zwischen \vec{x}_1 und \vec{x}_2 wird schließlich linear interpoliert. Mathematisch formuliert werden hierfür verschiedene Linearkombinationen \vec{x}_{int} beider Vektoren mit Schrittweite γ berechnet:

$$\vec{x}_{int} = (1 - \gamma) * \vec{x}_1 + \gamma * \vec{x}_2 \quad 4-3$$

Bei einem repräsentativen latenten Raum entsprechen die Vektoren, die auf Punkte auf dieser Interpolationsgeraden zeigen (in Abbildung 4.16 (a) sind dies die orangefarbenen Vektoren), Objekten, die nach und nach vom Ausgangs- ins Zielobjekt deformieren (Yoo et al. 2021). Je näher die Punkte dabei an Objekt 1 liegen, desto mehr Eigen-

²⁴ Werden für die Clusterbildung auch die latenten Vektoren aller Zwischenzustände herangezogen, kann es jedoch sein, dass ein Zwischenzustand einem finalen Zustand einer anderen Klasse mehr ähnelt, als einem finalen Zustand der eigentlichen Klasse. Daher sind in diesem Fall weniger gute Clusterlösungen zu erwarten.

²⁵ Die Objekte können dabei aus verschiedenen Bauteilklassen stammen, es kann sich jedoch auch um zwei Zwischenzustände einer Komponente handeln.

schaften von 1 sollten sie auch beinhalten. Gleiches gilt für Objekt 2. Zudem kann darüber gezeigt werden, dass der *Autoencoder* auch solche Vektoren verarbeiten kann, die er zuvor noch nicht gesehen hat.

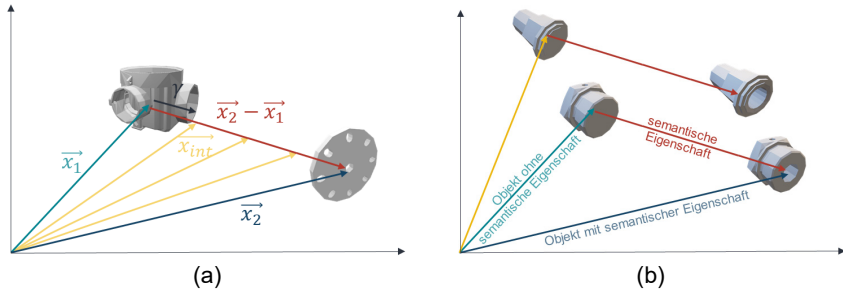


Abbildung 4.16: Prinzip der linearen Interpolation zwischen zwei latenten Vektoren (a) sowie algebraischer Operationen auf latenten Vektoren (b)

Die semantische Interpretierbarkeit des latenten Raumes kann darüber hinaus über algebraische Operationen auf den latenten Vektoren untersucht werden. Hierfür wird zunächst der Differenzvektor zwischen den latenten Vektoren zweier Objekte, dem eine semantische Bedeutung entsprechend der geometrischen Differenz der Objekte zugeordnet werden kann, bestimmt. In Abbildung 4.16 (b) ist diese semantische Eigenschaft beispielhaft eine Bohrung. Anschließend kann dieser Differenzvektor auf ein Objekt ohne diese Eigenschaft aufaddiert werden, wodurch dem Objekt diese Eigenschaft hinzugefügt wird. In Abbildung 4.16 (b) wird entsprechend dem Adapter mit orangefarbenem Vektor durch die Addition eine Bohrung angefügt. Diese Untersuchung ist insbesondere zwischen verschiedenen Zwischenzuständen einer Komponente interessant, da hier der Differenzvektor semantisch einem Konstruktionselement (z.B. einer Bohrung) entspricht.

4.4 Verwendung des Wissens zur Konstruktionsunterstützung

Im Rahmen dieses Kapitels wird beschrieben, wie das zuvor extrahierte und aufbereitete Wissen in die Konstruktion eines neuen Bauteiles einfließen kann, um damit den Konstrukteur zu unterstützen und den Konstruktionsprozess zu beschleunigen. Gemäß Abbildung 4.3 liegen, je nach anwendungsfallspezifischer Datenlage (siehe Kapitel 4.1 Abbildung 4.4), von jeder Komponente Informationen über die 3D-Geometrie, Metainformationen sowie das Konstruktionsvorgehen vor. Basierend auf dem in Kapitel 4.2

beschriebenen Vorgehen werden die Daten zunächst entsprechend aufbereitet. Anschließend werden die relevanten Eigenschaften der Geometrien in Form der latenten Repräsentation mit Hilfe einer *Autoencoder*-Architektur (siehe Kapitel 4.3) extrahiert. In Abbildung 4.17 ist für das Beispiel eines Gehäuses aufgezeigt, welche Daten in welcher Form für diese Komponente nach der Aufbereitung zur Verfügung stehen.

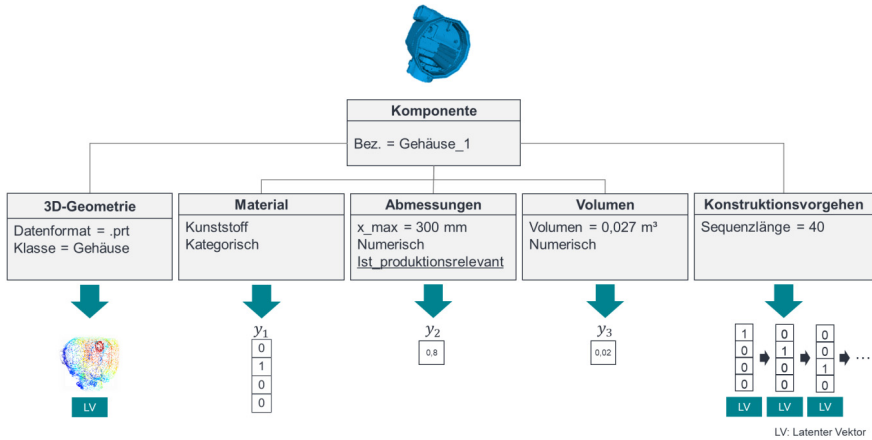


Abbildung 4.17: Exemplarische Darstellung der vorliegenden Daten je Komponente

Aus der Datenkategorie der Metainformationen sind exemplarisch drei Instanzen dargestellt. Das Material ist dabei kategorischer, die Abmessungen und das Volumen numerischer Natur. Der geometrische Fußabdruck des (finalen) Gehäuses setzt sich folglich aus dessen latenter Repräsentation (LV) sowie den drei Metainformationen zusammen. Das Konstruktionsvorgehen für diese Komponente besteht aus 40 Schritten. Diese werden über die Sequenz der latenten Repräsentationen der jeweiligen Zwischenzustände sowie die genutzten KEs, dargestellt über *One Hot Encoding*, abgebildet. Basierend auf diesen Informationen für alle Komponenten der betrachteten Datenbasis wird im Folgenden ein Konstruktionsassistenzsystem modelliert, das die Funktionalitäten der Ähnlichkeitssuche sowie des Vorschlagens nächster Konstruktionszustände umfasst. Nachfolgend werden zunächst die Implementierung der Ähnlichkeitssuche sowie die darin inkludierte Bewertung produktionsrelevanter Produkteigenschaften beschrieben. In Kapitel 4.4.2 wird dargelegt, wie aus dem aufbereiteten Wissen des Konstruktionsvorgehens zunächst über eine Sequenzanalyse Muster erlernt werden, die schließlich zur Vorhersage nächster Konstruktionschritte genutzt werden können.

Das Zusammenspiel beider Funktionalitäten wird abschließend in Kapitel 4.4.3 vorgestellt.

4.4.1 Ähnlichkeitssuche und Bewertung produktionsrelevanter Produkteigenschaften

Als Grundlage für die Ähnlichkeitssuche dient das extrahierte Wissen in Form des geometrischen Fußabdruckes je Komponente. Dieser setzt sich – je nach Umsetzungsstufe (siehe Kapitel 4.1.4) – aus Geometrie (Stufe 1), Metainformationen (Stufe 2) und produktionsrelevanten Produkteigenschaften (Stufe 3) zusammen. Von jeder existierenden Komponente aus der zugrunde liegenden Datenbasis ist diese Information abgespeichert. Zur Strukturierung dieser Datenbasis wird diese zunächst anhand der latenten Repräsentationen (und damit der Geometrie) in Cluster eingeteilt (Kapitel 4.4.1.1). Das anschließende Vorgehen zum Auffinden ähnlicher Modelle für ein gegebenes Input-Modell wird in Abbildung 4.18 dargestellt. Das Input-Modell entspricht dabei einem aktuellen Konstruktionsstand in Form eines CAD-Modells. Für einen Vergleich mit den bestehenden Komponenten muss dieses Modell in eine latente Repräsentation mittels des auf der Datenbasis trainierten *Autoencoders* überführt werden. Dafür erfolgt zunächst die Umwandlung in eine Punktwolke. Diese kann schließlich über den *Encoder* in die Vektordarstellung transformiert werden. Zudem werden, abhängig von der Umsetzungsstufe, die Metainformationen $y_{w,i}$ extrahiert, sodass insgesamt ein geometrischer Fußabdruck der Komponenten generiert wird. Um nun die ähnlichsten Komponenten für ein gegebenes Bauteil auffinden zu können, wird zunächst anhand der Clusterzentren das Cluster mit dem geringsten Abstand gesucht. Anschließend wird die Distanz dieses Bauteiles zu allen anderen Bauteilen innerhalb dieses Clusters berechnet. Das genaue Vorgehen hierzu wird in Kapitel 4.4.1.2 beschrieben. Sind außerdem Metainformationen vorhanden, werden die gefundenen (geometrisch) ähnlichen Modelle anhand dieser in eine Rangfolge gebracht. Basierend auf den gefundenen ähnlichsten Komponenten können schließlich über die angehängten, als produktionsrelevant definierten (siehe Kapitel 4.1.2) Metainformationen die produktionsrelevanten Produkteigenschaften des aktuellen Bauteilzustandes bewertet werden, worauf in Kapitel 4.4.1.3 näher eingegangen wird.

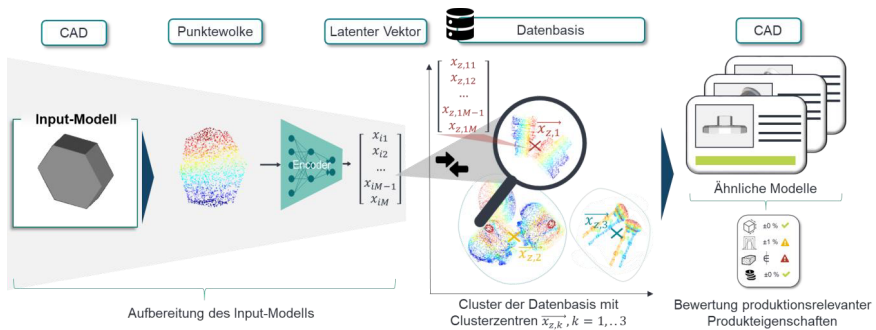


Abbildung 4.18: Vorgehen zum Auffinden ähnlicher Komponenten

4.4.1.1 Clustering der vorhandenen Datenbasis

Um die Suche nach ähnlichsten Teilen zu beschleunigen, wird die Menge aller latenten Vektoren der vorhandenen Datenbasis, S_j , zunächst in Cluster eingeteilt²⁶. Die Clusterbildung erfolgt dabei über die latenten Vektoren \bar{x}_j der finalen Zustände aller Komponenten j , stellvertretend für die Geometrie der entsprechenden Bauteile. Hierfür wird das partitionierende Verfahren *k-Means*²⁷ (vgl. Kapitel 2.2.3.1) angewandt, das üblicherweise im Bereich des *Deep Learning* zum Clustern auf latenten Vektoren herangezogen wird (Aljalbout et al. 2018; Yang et al. 2017; Hassani & Haley 2019). Zur Distanzberechnung wird die euklidische Distanz d_2 herangezogen (siehe Formel 2-5). Die Anzahl der Cluster $k=1, \dots, K$ kann dabei über die in Kapitel 2.2.3.1 beschriebenen Verfahren festgelegt werden. Ist die Anzahl der verschiedenen Bauteilklassen für die betrachtete Datenbasis bekannt, kann k auch entsprechend dieser definiert werden²⁸. Die initialisierten Cluster-Zentren z_k werden bei *k-Means* iterativ angepasst und die Objekte in Form der latenten Vektoren auf Basis der minimalen Distanz zu den Cluster-Zentren in jeder Iteration neu zugeordnet, bis sich keine Änderungen der Cluster-Zentren z_k mehr ergeben. Da *k-Means* nicht deterministisch ist und damit die Bestimmung der

²⁶ Die Clustering der Datenbasis kann bereits im Rahmen der Untersuchung der latenten Repräsentationen erfolgt sein (siehe Kapitel 4.3.3).

²⁷ Das im Rahmen dieser Arbeit genutzte dichte-basierte Verfahren *DBSCAN* ist für hochdimensionale Daten weniger geeignet (Pham & Afify 2007).

²⁸ Sofern K entsprechend der Anzahl an vorhandenen Bauteilklassen gewählt wird, wird angenommen, dass sich die Objekte einer Bauteilkategorie geometrisch ähnlich und Objekte verschiedener Bauteilkategorien geometrisch unähnlich sind.

Cluster-Zentren z_k und so auch die Zuordnung von Objekten zu Clustern zwischen verschiedenen Durchläufen leicht variieren kann, werden mehrere Durchläufe ausgeführt. Die gefundenen Clusterlösungen der verschiedenen Durchläufe können anhand der in Kapitel 2.2.3.3 gezeigten Maßzahlen bewertet und darauf basierend eine Lösung ausgewählt werden. Jedes Cluster wird schließlich durch dessen Zentrum z_k repräsentiert, das ebenfalls in Form eines latenten Vektors $\vec{x}_{z,k}$ vorliegt. Alle latenten Vektoren \vec{x}_j aus der Datenbasis sind schließlich einem Cluster über das Cluster-Zentrum $\vec{x}_{z,k}$ zugeordnet. Die Menge aller Vektoren eines Clusters wird durch S_{z_k} beschrieben. Abbildung 4.19 veranschaulicht dieses Vorgehen für eine Datenbasis bestehend aus $j=8$ Komponenten, die in $k=3$ Cluster eingeteilt werden. Werden neue Komponenten der Datenbasis hinzugefügt, so wird die entsprechende Distanz zu den Cluster-Zentren berechnet. Die Zuordnung erfolgt schließlich zu demjenigen Cluster, zu dessen Zentrum die Komponente die geringste Distanz aufweist. Die Cluster-Zentren spielen eine wesentliche Rolle beim Auffinden ähnlicher Modelle. Das Vorgehen hierzu wird in Kapitel 4.4.1.2 näher beschrieben.

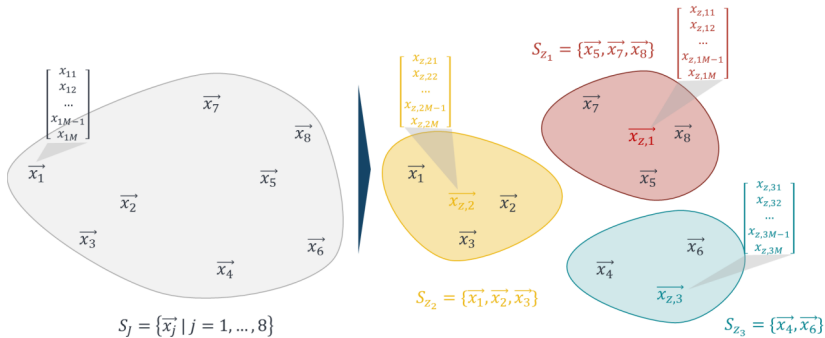


Abbildung 4.19: Clustern der vorhandenen Datenbasis in Form der latenten Vektoren

4.4.1.2 Auffinden geometrisch ähnlicher Modelle

Für ein gegebenes neues Bauteil i (bzw. einen Bauteilzustand), repräsentiert über dessen latenten Vektor, wird nun zunächst aus den vorhandenen Komponenten die Menge der ähnlichsten Modelle identifiziert. Zur Bestimmung dieser Menge ist rein die 3D-Geometrie ausschlaggebend. Für die Komponente i wird hierfür in einem ersten Schritt auf Basis der in Kapitel 4.4.1.1 bestimmten Cluster-Zentren das nahe gelegenste Cluster identifiziert. Bei einer geringen Datenmenge ist es auch möglich, die Distanz des Input-

Bauteiles zu allen existierenden Bauteilen zu bestimmen. Aus Gründen der Rechenzeit bei großen Datenmengen ist dieses Vorgehen jedoch weniger effizient. Zur Distanzbestimmung wird gemäß Formel 4-4 die euklidische Distanz d_2 zwischen dem gegebenen latenten Input-Vektor \vec{x}_i und den latenten Vektoren der jeweiligen Cluster-Zentren $\vec{x}_{z,k}$ berechnet und das Minimum ausgewählt.

$$d_2(\vec{x}_i, \vec{x}_{z,k}) = \|\vec{x}_i - \vec{x}_{z,k}\|_2 \tag{4-4}$$

Ausschlaggebend für die Distanz ist folglich die Geometrie des Bauteiles (also der latente Vektor).

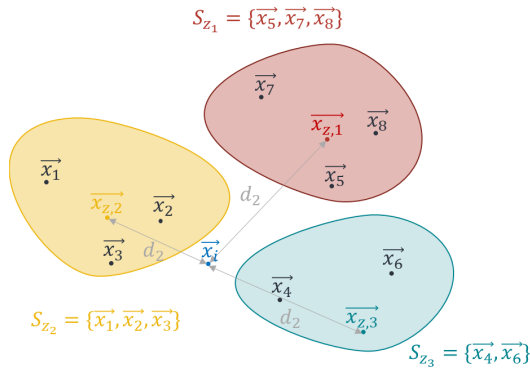


Abbildung 4.20: Bestimmung der Menge relevanter Clusterzentren für eine Input-Komponente i durch Berechnung der Distanzen von i zu allen Clusterzentren

Gemäß Abbildung 4.20 ist es möglich, dass die Distanz zu Mitgliedern eines weiter entfernten Clusterzentrums dennoch gering sein kann. Im visualisierten Beispiel weist die Komponente \vec{x}_4 zur Input-Komponenten \vec{x}_i eine geringere euklidische Distanz auf als Mitglieder aus dem nahe gelegenen Cluster mit Zentrum $\vec{x}_{z,2}$. Daher wird gemäß Formel 4-5 nicht nur das am nächsten liegende Cluster betrachtet, sondern die Menge der für den Input i relevanten Clusterzentren $S_{z,i}$:

$$S_{z,i} = \left\{ \arg \min_{\vec{x}_{z,k}} (d_2(\vec{x}_i, \vec{x}_{z,k})) \right\} \text{ mit } |S_{z,i}| = \text{fest} \tag{4-5}$$

Die feste Anzahl der relevanten Clusterzentren $|S_{z,i}|$, die für jeden Input i betrachtet werden sollten, ist abhängig von der jeweils zugrunde liegenden Datenbasis und muss

initial empirisch ermittelt werden (siehe Kapitel 5.4.1.2). Anschließend werden die euklidischen Distanzen d_2 zwischen dem Input-Vektor \vec{x}_i und allen Cluster-Mitgliedern $\vec{x}_j \in S_{z_k}$ aller Cluster mit Zentrum $\vec{x}_{z,k}$ berechnet, für die gilt $\vec{x}_{z,k} \in S_{z,i}$. Im Beispiel von Abbildung 4.21 ist $|S_{z,i}|$ gleich zwei und es gilt $S_{z,i} = \{\vec{x}_{z,2}, \vec{x}_{z,3}\}$. Es werden folglich die euklidischen Distanzen zu allen Mitgliedern von Cluster 2 und 3 ermittelt.

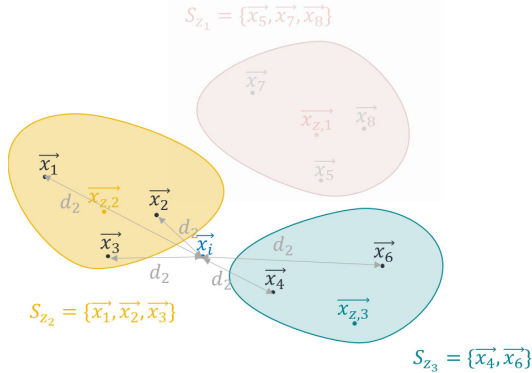


Abbildung 4.21: Bestimmung der Distanzen der Input-Komponenten i zu allen Komponenten j aus den Clustern mit relevanten Clusterzentren

Auf Basis der so berechneten Distanzen wird anschließend für ein gegebenes Input-Modell i die Menge der (geometrisch) ähnlichsten Modelle, $S_{sim,i}$, bestimmt. Hierfür wird ein Schwellenwert ε definiert. Für das gegebene Input-Modell werden alle Modelle als ähnlich deklariert, die innerhalb der Umgebung mit einem maximalen Suchradius von ε liegen. Es gilt folglich:

$$S_{sim,i} = \{\vec{x}_j \in \cup S_{z_k} \mid \vec{x}_{z,k} \in S_{z,i} \text{ und } d_2(\vec{x}_i, \vec{x}_j) \leq \varepsilon\} \quad 4-6$$

Die entsprechenden CAD-Modelle, die hinter den latenten Vektoren $\vec{x}_j \in S_{sim,i}$ mit minimalem Abstand liegen, werden folglich als (geometrisch) ähnlichste Modelle ausgegeben. Abbildung 4.22 veranschaulicht dieses prinzipielle Vorgehen. Im dargestellten Beispiel gilt $S_{sim,i} = \{\vec{x}_2, \vec{x}_4\}$.

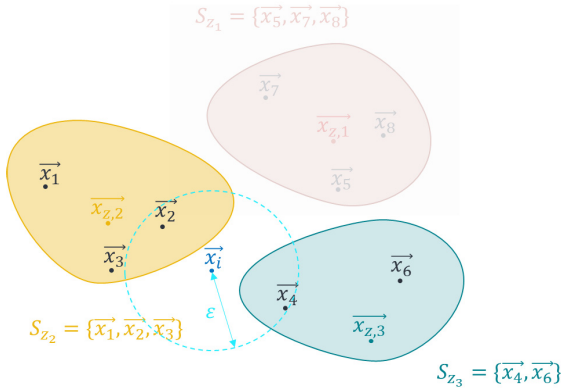


Abbildung 4.22: Bestimmung der Menge der ähnlichsten Modelle für eine Input-Komponente i über den Suchradius ε

Gemäß Kapitel 2.2.2.2 liegt dem latenten Raum keine physikalische Einheit zugrunde. Um dennoch die geometrische Ähnlichkeit der Modelle aus der Menge $S_{sim,i}$ zu bestimmen, werden die euklidischen Distanzen $d_2(\vec{x}_i, \vec{x}_j)$ mit der maximalen euklidischen Distanz, die in der Datenbasis zu finden ist, $d_{2,max}$ normiert. Für die Ähnlichkeit Sim_{ij} eines Objektes j zum Input i gilt folglich:

$$Sim_{ij} = 1 - \frac{d_2(\vec{x}_i, \vec{x}_j)}{d_{2,max}} \in [0,1] \quad 4-7$$

Ein Wert von $Sim_{ij} = 1$ steht dabei für eine Ähnlichkeit zwischen Komponente i und j von 100 %. Sofern für den betrachteten Datensatz keine Metainformationen vorhanden sind (siehe Umsetzungsstufe 1 der Ähnlichkeitssuche in Kapitel 4.1.4), wird die Ähnlichkeit anhand der Geometrie und damit rein über die latenten Vektoren bestimmt. Die Menge der ähnlichsten Modelle $S_{sim,i}$ wird folglich auf Basis der berechneten Ähnlichkeiten Sim_{ij} absteigend sortiert. Sofern Metainformationen vorliegen (siehe Umsetzungsstufe 2 der Ähnlichkeitssuche in Kapitel 4.1.4), wird die Ähnlichkeit als gewichtete Summe aus geometrischer Ähnlichkeit und Ähnlichkeit der Metainformationen berechnet. Die Gewichtungen für die Geometrie, α_{geo} , als auch für die jeweiligen Metainformationen, α_w , können dabei durch den Anwender bestimmt werden.

Gemäß Kapitel 4.1.2 liegen für ein Objekt j aus der Datenbasis die Anzahl W als für relevant erklärte Metainformationen $y_{w,j}$ vor. Für jede dieser Metainformationen $y_{w,j}$ kann zudem eine entsprechende Gewichtung α_w definiert werden, die für alle Objekte

j gleich ist. Als Distanzmaß zwischen den Metainformationen des Input-Objektes i und der Objekte j mit $\vec{x}_j \in S_{sim,i}$ wird ebenfalls die euklidische Distanz herangezogen. Zur Bestimmung der Ähnlichkeit $Sim_{ij,w}$ bezüglich der Metainformation y_w zwischen dem Objekt j und dem Input-Objekt i wird die berechnete Distanz ebenfalls mit der maximalen Distanz $d_{2,max,w}$, die zwischen der Metainformation y_w zweier Objekten j aus dem betrachteten Datensatz aufzufinden ist, normiert. Für die Ähnlichkeit $Sim_{ij,w}$ gilt folglich:

$$Sim_{ij,w} = 1 - \frac{d_2(y_{w,i}, y_{w,j})}{d_{2,max,w}} \in [0,1] \quad 4-8$$

Somit kann für jedes Objekt j aus der Menge der ähnlichsten Modelle die Ähnlichkeit zum Input-Objekt i auf Basis der Geometrie sowie der definierten Metainformationen angegeben werden. Die Gesamtähnlichkeit $Sim_{ij,ges}$ zwischen dem Input-Objekt i und einem Objekt j aus der Menge der ähnlichsten Modelle berechnet sich schließlich als gewichtete Summe dieser Ähnlichkeiten wie folgt:

$$Sim_{ij,ges} = \alpha_{geo} * Sim_{ij} + \sum_{w=1}^W \alpha_w * Sim_{ij,w} \quad 4-9$$

Für die Gewichtungsfaktoren gilt dabei:

$$\alpha_{geo} + \sum_{w=1}^W \alpha_w = 1 \quad 4-10$$

Die Menge der ähnlichsten Komponenten $S_{sim,i}$ wird anschließend anhand der gewichteten Ähnlichkeiten absteigend sortiert.

Durch das beschriebene Vorgehen können schließlich für ein gegebenes Input-Objekt i die ähnlichsten Objekte aus der bestehenden Datenbasis aufgefunden werden. Ausschlaggebend für die Bestimmung der Menge der ähnlichsten Modelle $S_{sim,i}$ ist dabei zunächst die Geometrie in Form der latenten Vektoren. Die Reihenfolge, in der diese Modelle vorgeschlagen werden, kann durch die weitere Berücksichtigung der Metainformationen angepasst werden. Auf Basis der Menge der ähnlichsten Modelle können schließlich die produktionsrelevanten Produkteigenschaften des Input-Modells i bewertet werden. Das Vorgehen hierzu wird nachfolgend beschrieben.

4.4.1.3 Abgleich produktionsrelevanter Produkteigenschaften

Gemäß Kapitel 2.1.2 ist unter anderem die geometrische Form eines Bauteiles ausschlaggebend für die Wahl der Fertigungsverfahren. Folglich wird die Annahme zugrunde gelegt, dass geometrisch ähnliche Objekte auch ähnliche Fertigungsverfahren bedingen²⁹. Unter der Voraussetzung, dass die Komponenten aus der betrachteten Datenbasis auf den vorhandenen Fertigungsmitteln bereits erfolgreich produziert wurden, kann angenommen werden, dass die daraus ableitbaren Ausprägungen der produktionsrelevanten Produkteigenschaften für die jeweiligen Geometrien mit den vorhandenen Fertigungsmitteln umsetzbar sind. Für ein gegebenes Input-Objekt i bildet daher die in Kapitel 4.4.1.2 bestimmte Menge der geometrisch ähnlichsten Modelle $S_{sim,i}$ die Grundlage zur Bewertung der produktionsrelevanten Produkteigenschaften. Dafür werden zunächst von den Komponenten $j \in S_{sim,i}$ die Ausprägungen der in Kapitel 4.1.2 als produktionsrelevant definierten Metainformationen y_w extrahiert. Für jede Komponente j werden folglich die produktionsrelevanten Produkteigenschaften in Form eines Vektors $\overrightarrow{y_{w,j}}$ dargestellt, der einem Punkt in einem w -dimensionalen Raum entspricht. Insgesamt werden somit für alle ähnlichen Geometrien die entsprechenden Kombinationen der betrachteten Eigenschaften aufgezeigt, die bereits produzierbar waren. In Abbildung 4.23 ist dieser Raum beispielhaft für die Ausprägungen der drei produktionsrelevanten Produkteigenschaften x -, y - und z -Abmessung, die in der Menge der ähnlichsten Modelle j auftreten, dargestellt. Zusätzlich sind die entsprechenden Werte der Input-Komponenten i , dargestellt über ein rotes „x“, abgebildet. Es wird ersichtlich, dass es dichte und weniger dicht besiedelte Bereiche im Raum der produktionsrelevanten Produkteigenschaften geben kann. Für Bereiche, für die in der Menge der ähnlichsten Modelle keine entsprechenden Daten vorliegen, kann keine sichere Aussage darüber getroffen werden, ob die entsprechende Parameterkombination auf den vorhandenen Fertigungsmitteln produzierbar ist oder nicht. Daher wird der aufgespannte Raum zunächst in Bereiche mit höherer Dichte, also mit ausreichend Datenpunkten, eingeteilt. Hierfür wird auf den Vektoren $\overrightarrow{y_{w,j}}$ der produktionsrelevanten Produkteigenschaften aller Komponenten $j \in S_{sim,i}$ sowie auf $\overrightarrow{y_{w,i}}$ der Komponenten i zunächst eine dichtebasierte Clustering mit Hilfe von *DBSCAN* (vgl. Kapitel 2.2.3.2) durchgeführt. Dieses Verfahren ist

²⁹ Weitere Einflussgrößen, wie z.B. das Material, werden erst später berücksichtigt, um ggf. auch Hinweise auf mögliche Verbesserungen der Komponente, z.B. durch eine Materialänderung, geben zu können.

dazu geeignet, da es Cluster bildet, indem Regionen mit ausreichend vielen Datenpunkten (also ausreichender Dichte) zusammengefasst werden (Pham & Afify 2007).

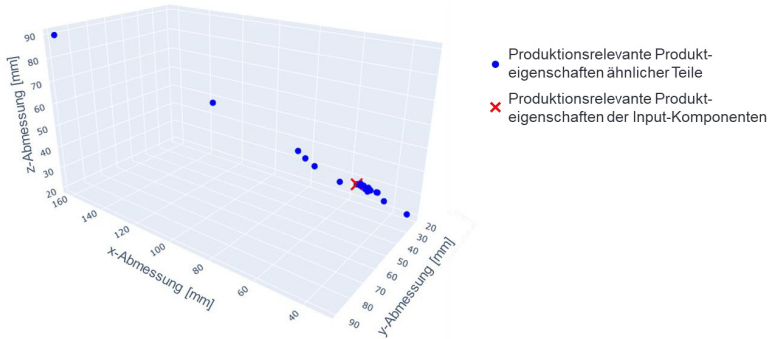


Abbildung 4.23: Mögliche Kombinationen aus x-, y- und z-Abmessung der Komponenten j aus der Menge der ähnlichsten Modelle (blau) sowie die gewählte Kombination der Input-Komponente i (rot)

Für einen Datenpunkt innerhalb eines Clusters gilt dabei, dass in der Nachbarschaft im Bereich eines definierten Radius ε_{DB} eine Mindestanzahl an Datenpunkten (*MinPts*) liegen muss und somit der Raum ausreichend dicht besiedelt ist (Pham & Afify 2007). Dabei wird zwischen Kern-, Rand- und Rauschpunkten unterschieden. Kernpunkte sind solche Punkte, in deren ε_{DB} -Umgebung mindestens *MinPts* Nachbarn liegen. Randpunkte dagegen erfüllen diese Bedingung nicht, fallen jedoch in die ε_{DB} -Umgebung eines Kernpunktes. Folglich sind es Punkte in weniger dicht besiedelten Bereichen. Trifft keine der Bedingungen zu, so handelt es sich um einen Rauschpunkt. Ein Cluster besteht folglich aus Kern- und ggf. Randpunkten. Die erhaltene Anzahl an H Clustern ist abhängig von den gewählten Parametern ε_{DB} und *MinPts*, die je nach Anwendungsfall zu bestimmen sind. Die Menge der Komponenten j , die dem Cluster h zugeordnet werden, wird mit S_h bezeichnet. Alle Komponenten innerhalb eines Clusters weisen folglich Ähnlichkeiten bezüglich der produktionsrelevanten Produkteigenschaften auf.

Abbildung 4.24 (a) zeigt das Ergebnis der Clusterbildung. Für das gezeigte Beispiel werden zwei Cluster gefunden. Cluster 2 besteht dabei aus zwei Komponenten, deren produktionsrelevante Produkteigenschaften übereinstimmen, weshalb das Cluster lediglich einen Punkt im Raum darstellt. Violett markiert sind Rauschpunkte, die keinem Cluster zugeordnet werden konnten. Um nun die Ausprägungen der produktionsrelevanten Produkteigenschaften der Komponenten i zu bewerten, wird zunächst bestimmt,

ob diese einem Cluster zugeordnet werden können und folglich bereits Komponenten mit ähnlichen Ausprägungen existieren. Dabei wird unterschieden, ob es sich um einen Kern- oder Randpunkt handelt. Ist keine Zuordnung möglich, erfolgt die Markierung als Rauschpunkt. Für den Fall eines Kern- oder Randpunktes wird der Bereich der produktionsrelevanten Produkteigenschaften PP_h des entsprechenden Clusters h anhand der darin enthaltenen Ausprägungen der Komponenten j bestimmt. Die durch das rote „x“ in Abbildung 4.24 gekennzeichneten Ausprägungen der Komponenten i werden Cluster 1 als Kernpunkt zugeordnet. Der sich ergebende Bereich PP_1 wird in Abbildung 4.24 (b) über den rot markierten Quader dargestellt. Können die Ausprägungen der Komponenten i dagegen keinem Cluster zugeordnet werden oder sind die entsprechenden Werte noch nicht bekannt, werden alle möglichen Bereiche PP_h der gefundenen Cluster $h=1, \dots, H$ bestimmt.

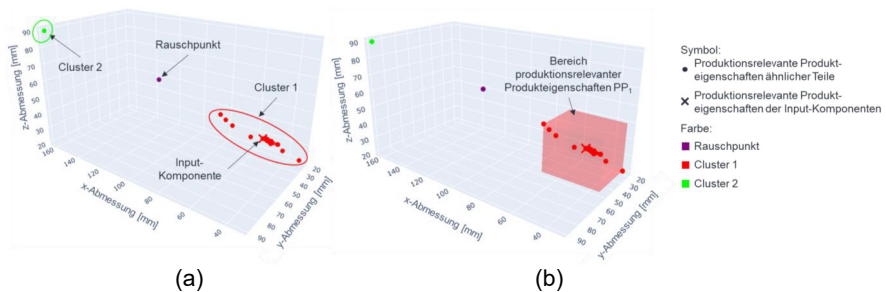


Abbildung 4.24: Exemplarische Einteilung der Kombination von x-, y- und z-Abmessung in zwei Cluster (a) sowie exemplarischer Bereich der produktionsrelevanten Produkteigenschaften (b)

Zur Bestimmung von PP_h wird zwischen numerischen und kategorischen Werten unterschieden. Für numerische Werte, wie z.B. die maximalen Abmessungen, wird anhand der Ausprägungen die mögliche Spannweite je produktionsrelevanter Produkteigenschaft bestimmt. Hierfür wird anhand der Komponenten j aus dem Cluster h der minimale ($y_{w,min}$) und maximale ($y_{w,max}$) Wert der Ausprägungen der jeweiligen Metainformation $y_{w,j}$ bestimmt:

$$y_{w,max} = \max_{y_{w,j}, j \in S_h} (y_{w,j}) \quad 4-11$$

$$y_{w,min} = \min_{y_{w,j}, j \in S_h} (y_{w,j}) \quad 4-12$$

Im Fall von kategorischen Variablen dagegen wird abgeleitet, welche diskreten Ausprägungen $y_{w,j}$ der kategorischen Variable w in der Menge S_h aufzufinden sind. Mathematisch wird also die nachfolgende Menge bestimmt:

$$\bigcup_{y_{w,j} \in S_h} \{y_{w,j}\} \quad 4-13$$

Sind für die Input-Komponente i die Metainformationen w bereits festgelegt, kann überprüft werden, ob die gewählten Ausprägungen $y_{w,i}$ in dem für diese Geometrie üblichen Werte-Bereich liegen. Für numerische Variablen wird folglich überprüft, ob gilt:

$$y_{w,i} \in [y_{w,min}, y_{w,max}] \forall w = 1, \dots, W \quad 4-14$$

Für kategorische Variablen wird bewertet, ob die gewählte Ausprägung $y_{w,i}$ in der Menge der Ausprägungen der Komponenten j dieses Merkmals enthalten ist:

$$y_{w,i} \in \bigcup_{y_{w,j} \in S_{sim,i}} \{y_{w,j}\} \quad 4-15$$

Zur Bewertung der produktionsrelevanten Produkteigenschaften werden die in Abbildung 4.25 aufgezeigten Fälle unterschieden, die anhand eines Ampelsystems visualisiert werden. Können die produktionsrelevanten Produkteigenschaften der Komponente i einem Cluster h zugeordnet werden, so existieren bereits geometrisch ähnliche Komponenten mit ähnlichen Ausprägungen. Liegen alle produktionsrelevanten Produkteigenschaften innerhalb des entsprechenden Bereichs PP_h und handelt es sich zudem um einen Kernpunkt, so werden diese als voraussichtlich auf den vorhandenen Produktionsmitteln produzierbar (grün) bewertet³⁰. Liegt die Kombination der Ausprägungen im weniger dicht besiedelten Bereich von PP_h und wird folglich als Randpunkt deklariert, wird sie als möglicherweise kritisch (gelb) eingestuft. Handelt es sich um einen Rauschpunkt, so wird die gewählte Kombination als kritisch (rot) eingeschätzt. Sofern die produktionsrelevanten Produkteigenschaften (teilweise) außerhalb von PP_h liegen, wird die Kombination stets als kritisch bewertet. Dabei wird aufgezeigt, welche produktionsrelevanten Produkteigenschaften außerhalb von PP_h liegen.

³⁰ Eine definitive Aussage ist jedoch anhand dieser Bewertung nicht möglich.

	$\overline{y_{w,i}} \in PP_h$	$\overline{y_{w,i}} \notin PP_h$
Kernpunkt		
Randpunkt		
Rauschpunkt		

Abbildung 4.25: Bewertung der produktionsrelevanten Produkteigenschaften einer Komponente *i* über ein Ampelsystem

Die Bewertung der produktionsrelevanten Produkteigenschaften für die in Abbildung 4.24 gezeigte Komponente ist in Abbildung 4.26 exemplarisch dargestellt. Neben der tatsächlichen Ausprägung der Komponenten *i* (Spalte „Wert“) sind auch die minimalen und maximalen Ausprägungen aus der Menge der ähnlichsten Modelle aufgezeigt. Für die Komponente *i* liegen alle Eigenschaften im üblichen Wertebereich, was über die grüne Ampel in der Spalte „Bewertung“ visualisiert wird. Zudem handelt es sich um einen Kernpunkt, wodurch die Gesamtbewertung gemäß Abbildung 4.25 ebenfalls auf grün gesetzt wird. Folglich können die produktionsrelevanten Produkteigenschaften als voraussichtlich mit den vorhandenen Produktionsmitteln produzierbar eingestuft werden.

	Bewertung PP ₁					
	Eigenschaft	Einheit	Wert	Minimum	Maximum	Bewertung
	Abmessung x-Richtung	mm	50,2	42	69,5	
	Abmessung y-Richtung	mm	34,9	20	49,5	
	Abmessung z-Richtung	mm	34,9	20	49,5	
Gesamtbewertung						

Abbildung 4.26: Exemplarische Bewertung der produktionsrelevanten Produkteigenschaften

Anhand dieser Bewertung können schließlich durch den Konstrukteur für die Komponente *i* in Abhängigkeit der betrachteten produktionsrelevanten Produkteigenschaften verschiedenste Rückschlüsse getroffen werden. Gemäß Kapitel 2.1.2 übt die gewählte Genauigkeit, mit der eine Komponente aus einem bestimmten Material gefertigt werden kann, starken Einfluss auf die einsetzbare Fertigungstechnologie aus. Liegt für einen Anwendungsfall die gewählte Toleranz außerhalb der üblichen Wertebereiche, so existieren bisher noch keine geometrisch ähnlichen Komponenten, die mit entsprechender Ausprägung bereits mit den vorhandenen Fertigungsmitteln produziert wurden. Folglich

sollte durch den Konstrukteur überprüft werden, ob der aktuell gesetzte Wert mit den vorhandenen Fertigungsmitteln umsetzbar ist. Ähnliche Rückschlüsse können auch anhand des Vergleichs der gewählten Baugröße mit den abgeleiteten Grenzwerten aus der Menge der ähnlichsten Objekte gezogen werden.

Als weitere produktionsrelevante Produkteigenschaft sind die Fertigungskosten und Stückzahlen aufzuführen. Können für die Komponente i die Fertigungskosten bereits angegeben werden, so ist ein Vergleich mit den üblichen Kosten für ähnliche Komponenten durchführbar. Liegen die Fertigungskosten – bei ähnlicher Stückzahl – außerhalb des üblichen Rahmens und wird zudem angezeigt, dass die gewählten Toleranzen außerhalb der Grenzwerte liegen oder das angedachte Material für diese Geometrie bisher nicht genutzt wurde, kann dadurch impliziert werden, dass die Toleranzen angepasst oder das Material gewechselt werden sollten, um niedrigere Kosten zu erreichen.

Sind für die Input-Komponente i die produktionsrelevanten Produkteigenschaften (zum Teil) noch nicht definiert (z.B. Toleranzen), können die entsprechenden Werte aus den erhaltenen Bereichen der produktionsrelevanten Produkteigenschaften PP_n der Menge der ähnlichsten Modelle zudem als Anhaltspunkt genutzt werden. Ergeben sich beispielsweise mehrere Bereiche mit jeweils unterschiedlichen Materialien, so lassen sich je Bereich geeignete Toleranzen ableiten. Darüber hinaus können auch Schätzungen für die späteren Fertigungskosten getätigt werden, indem die Kosten ähnlicher Komponenten mit vergleichbarer Stückzahl herangezogen werden.

4.4.2 Sequenzanalyse zur Vorhersage nächster Konstruktionsschritte

Ziel der Sequenzanalyse ist es, aus dem vorhandenen Konstruktionsvorgehen bestehender Komponenten Muster zu erlernen, auf deren Basis bei der Konstruktion einer neuen Komponente nächste Konstruktionsschritte vorgeschlagen werden können. Zum Erlernen von Mustern aus sequentiellen Daten sind *Recurrent Neural Networks* (RNNs) geeignet (vgl. Kapitel 2.2.4).

Grundlage für das Vorhersagen nächster Konstruktionsschritte bilden die gemäß Kapitel 4.2.3 erstellten Sequenzabschnitte in Form der KEs und der zugehörigen geometrischen Entwicklung des 3D-Modells. Letztere wird über die entsprechenden latenten Vektoren dargestellt werden (siehe Abbildung 4.15). Die Sequenzabschnitte repräsentieren das Konstruktionsvorgehen der verschiedenen Komponenten aus der Datenbasis, aus dem letztendlich Muster erlernt werden sollen. Das prinzipielle Vorgehen wird in Abbildung 4.27 dargestellt.

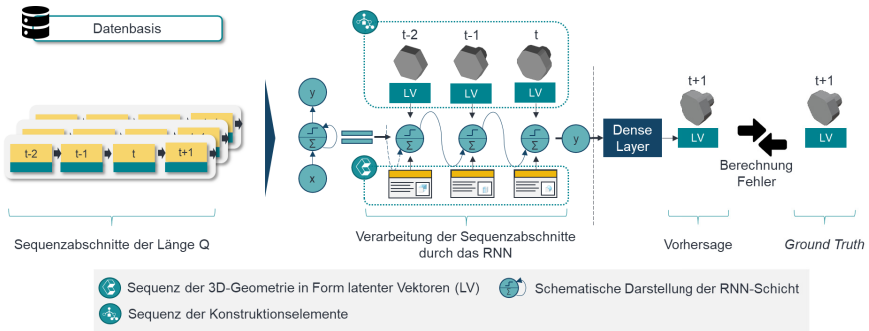


Abbildung 4.27: Prinzipielle Vorgehensweise der Sequenzanalyse mittels RNN

Input für das Training des RNNs sind die erstellten Sequenzabschnitte der Länge Q , die in Trainings- und Testdaten aufgeteilt werden. Jede Sequenz repräsentiert dabei zu einem Zustand t die jeweils letzten $Q - 1$ ausgeführten Konstruktionsschritte sowie den Q -ten Konstruktionsschritt, der für den nächsten Zustand $t+1$ vorhergesagt werden soll. Die Sequenz bestehend aus den $Q - 1$ ausgeführten Konstruktionsschritten wird schließlich durch die RNN-Schicht(en) verarbeitet. Im Vergleich zu einfachen *Feed Forward* Netzwerken verarbeiten RNNs eine Sequenz von Eingaben nicht unabhängig voneinander, sondern nutzen zusätzlich auch Informationen aus den vorangegangenen Eingaben, die in Form des *Hidden State* übergeben werden. Dieser *Hidden State* beinhaltet folglich Informationen relativ zu den vorangegangenen Elementen und bildet quasi das Gedächtnis des RNN (Chollet 2018). Nach Verarbeitung einer Sequenz wird dieses Gedächtnis wieder zurückgesetzt, sodass zwei unabhängige Sequenzeingaben als einzelne Datenpunkte betrachtet werden können (Chollet 2018). Bezogen auf den Anwendungsfall bedeutet das, dass zur Vorhersage des Konstruktionsschrittes $t+1$ nicht nur die Informationen des aktuellen Zustandes t , sondern über das Gedächtnis des RNN auch die Informationen der vorangehenden Zustände (in Abbildung 4.27 also alle Zustände bis $t - 2$) genutzt werden. Somit kann die gesamte Information aus der gegebenen Input-Sequenz über die letzten Konstruktionsschritte verarbeitet werden. Die gewählte Länge der Input-Sequenz bzw. der Sequenzabschnitte ist dabei ausschlaggebend dafür, nach wie vielen Schritten der interne Speicher des RNNs wieder zurückgesetzt wird. Während des Trainings wird der vorhergesagte nächste Konstruktionsschritt mit dem wahren Konstruktionsschritt abgeglichen und über eine Fehlerfunktion die Verfahrensparameter entsprechend angepasst.

Das so trainierte RNN kann schließlich dafür genutzt werden, anhand erlernter Muster aus dem Trainingsdatensatz für eine gegebene Input-Sequenz an durchgeführten Konstruktionsschritten den wahrscheinlichsten nächsten Konstruktionsschritt in Form des entsprechenden Bauteilzustandes vorherzusagen.

Gemäß Kapitel 4.1.4 werden für das Vorschlagen nächster Konstruktionsschritte in Abhängigkeit der Datenverfügbarkeit zwei Umsetzungsstufen unterschieden. Diese grenzen sich durch die genutzten Input-Daten für die Sequenzanalyse voneinander ab, was im nachfolgenden Kapitel näher erläutert wird. In Kapitel 4.4.2.2 folgt schließlich die Beschreibung der genauen Architektur sowie des Trainings des RNNs. Auf dessen Anwendung zur Vorhersage nächster Konstruktionsschritte wird abschließend in Kapitel 4.4.2.3 eingegangen.

4.4.2.1 Mögliche Input-Kombinationen

Bezüglich der Verfügbarkeit der Input-Daten werden zwei verschiedene Varianten betrachtet. Die einfachere Variante 1 (gemäß Kapitel 4.1.4 Umsetzungsstufe 1) nutzt als Input für das RNN lediglich die geometrische Entwicklung des 3D-Modells, repräsentiert über die Sequenz der latenten Vektoren $\vec{x}^{(t)}$ bis zum Zeitpunkt t . In dieser Input-Sequenz sind implizit alle Informationen enthalten, beispielsweise sind auch die genutzten KEs, die zur Veränderung zwischen zwei Zuständen geführt haben, durch die geometrische Änderung der Geometrie abgebildet.

Als weitere Informationsbasis kann jedoch die Sequenz der genutzten KEs explizit als Information dem Input angefügt werden (gemäß Kapitel 4.1.4 Umsetzungsstufe 2). Durch diese Informationsanreicherung kann der aktuelle Zustand einer Komponente noch konkreter beschrieben werden, sodass eine präzisere Vorhersage durch das RNN getroffen werden kann. Diese Kombination aus der geometrischen Entwicklung und den zugehörigen KEs wird in Variante 2 betrachtet. In Abbildung 4.28 sind die Unterschiede beider Varianten nochmals aufgezeigt. Auf Basis beider Varianten an Input-Daten wird schließlich das RNN trainiert, um den nächsten Konstruktionsschritt in Form des nächsten Bauteilzustandes vorherzusagen. Das Vorgehen hierzu wird in den nachfolgenden Kapiteln näher beschrieben.

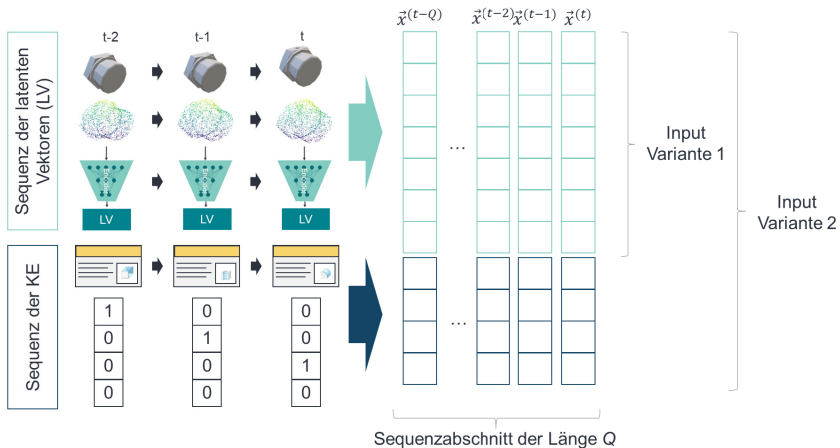


Abbildung 4.28: Betrachtete Varianten des Inputs für das RNN in Abhängigkeit der Umsetzungsstufe

4.4.2.2 Allgemeine Architektur des Recurrent Neural Networks

Ziel ist es, auf Basis einer Input-Sequenz mit den letzten $Q-1$ Konstruktionsschritten den nächsten Konstruktionsschritt vorherzusagen. Der grundsätzliche Aufbau der RNN-Architektur ist in Abbildung 4.29 dargestellt.

Input sind die Sequenzabschnitte der Länge Q bestehend aus den erstellten latenten Vektoren und – in Abhängigkeit der Umsetzungsstufe - den codierten KEs (siehe Abbildung 4.28). Die Länge dieses Input-Vektors definiert die Anzahl der Inputs für das RNN, das aus einer oder mehreren RNN-Schichten aufgebaut ist. Die Anzahl der dazwischen liegenden RNN-Schichten (Tiefe) sowie deren Größe (Breite) sind ein Hyperparameter zur Verbesserung der Repräsentationsfähigkeit und entsprechend in Abhängigkeit des betrachteten Datensatzes zu optimieren (Chollet 2018). Dabei können diese Schichten entweder aus einfachen RNN-Zellen, GRU-Zellen oder LSTM-Zellen aufgebaut werden. Gemäß Kapitel 2.2.4 unterscheiden sich diese insbesondere in ihrem Langzeitspeicher. Während einfache RNNs bis zu einer Sequenzlänge von 10-20 Schritten geeignet sind, werden LSTM und GRU herangezogen, wenn Informationen, die noch weiter in der Vergangenheit liegen, berücksichtigt werden sollen (Goodfellow, Bengio & Courville 2016). Nachteilig ist hier, dass die Architektur deutlich komplexer ist und die Anzahl der zu erlernenden Parameter steigt. Die über die RNN-Schichten er-

lernte Repräsentation der Input-Sequenz wird schließlich an die Output-Schicht übergeben. Diese erzeugt darauf basierend einen latenten Vektor, der die gleiche Anzahl an Dimensionen aufweist wie die latenten Vektoren der Input-Sequenz. Im Vergleich zu typischen Anwendungen aus der Sprachverarbeitung gibt es folglich keine feste Menge an Outputs, wie beispielsweise ein Wörterbuch, sondern es können neue latente Vektoren und damit 3D-Geometrien erzeugt werden. Dementsprechend stellt die Ausgabe auch keine Wahrscheinlichkeitsverteilung dar, die in der Sprachverarbeitung die Wahrscheinlichkeit bestimmter Wörter ausgibt, sondern die Werte eines latenten Vektors.

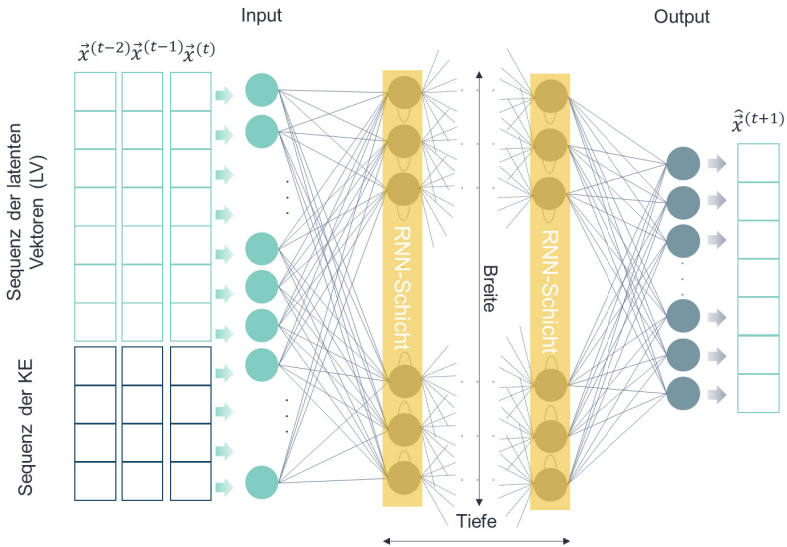


Abbildung 4.29: Grundsätzlicher Aufbau der RNN-Architektur

Während des Trainings dienen die Sequenzabschnitte der Länge Q als Input für das RNN. Gemäß Abbildung 4.29 werden dabei die letzten $Q-1$ Sequenzelemente, bestehend aus den latenten Vektoren der Zwischenzustände $\vec{x}^{(t)}$ bis zum Zeitpunkt t als Input verarbeitet. Darauf basierend soll das RNN den nächsten Konstruktionsstand zum Zeitpunkt $t+1$ in Form des latenten Vektors $\hat{\vec{x}}^{(t+1)}$ vorhersagen. Diese Vorhersage wird schließlich mit dem tatsächlichen nächsten Konstruktionsstand, $\vec{x}^{(t+1)}$, der dem Q -ten Element der Sequenzabschnitte entspricht, verglichen und der resultierende Fehler berechnet. Als Fehlerfunktion wird dabei der *Mean Squared Error* (MSE) herangezogen.

Neben der Vorhersage des letzten Konstruktionsschrittes der Sequenzabschnitte mit Länge Q kann gemäß Kapitel 2.2.4 bereits nach Erhalt des ersten Sequenzelements eine Vorhersage getroffen werden. Für einen Sequenzabschnitt mit Q Elementen können folglich $Q-1$ Prädiktionen in Form des nächsten Konstruktionsschrittes, repräsentiert über den jeweiligen latenten Vektor $\hat{x}^{(t)}$, ausgegeben werden. Folglich können auch $Q-1$ Fehler berechnet werden. Auf Basis des so errechneten Verlustes werden die Netzgewichte entsprechend dem *Backpropagation Through Time* Algorithmus (vgl. Kapitel 2.2.4) angepasst.

Die wesentlichen Hyperparameter für die Architektur sowie das Training des RNNs sind in Anhang D aufgezeigt und anwendungsfallspezifisch zu wählen. Hierzu zählt die Länge der Sequenzabschnitte Q und damit der Input-Sequenz. Darüber wird bestimmt, wie weit das RNN in die Vergangenheit schaut und entsprechende Informationen berücksichtigt. Für den betrachteten Anwendungsfall bedeuten kleinere Sequenzlängen, dass allgemeinere Muster erlernt werden können³¹. Je länger die Sequenzen sind, desto mehr Informationen sind darin enthalten und desto bauteilspezifischer sind folglich die durchgeführten Konstruktionsschritte. Entsprechend der festgelegten Länge der Eingabesequenz ist schließlich der Typ der RNN-Schicht zu wählen. Wie oben erwähnt sind für längere Sequenzen grundsätzlich LSTM oder GRU geeigneter.

In der Testphase bzw. bei Anwendung des trainierten RNNs kann schließlich für eine gegebene Input-Sequenz der Länge $Q-1$ der nächste Konstruktionsschritt in Form des latenten Vektors $\hat{x}^{(t+1)}$ vorhergesagt werden. Prinzipiell wären auch Vorhersagen auf kürzeren Input-Sequenzen möglich. Sofern jedoch bereits ausreichend viele Konstruktionsschritte durchgeführt wurden, sollten diese auch in der Vorhersage genutzt werden, um möglichst viele Informationen aus dem vorangegangenen Konstruktionsvorgehen zu berücksichtigen. Folglich werden bei der Anwendung alle Outputs bis auf die letzte Vorhersage $\hat{x}^{(t+1)}$ deaktiviert. Für eine Input-Sequenz wird also ein einziger Output generiert. Die genaue Anwendung des RNNs zur Vorhersage nächster Konstruktionsschritte wird im nachfolgenden Kapitel erläutert.

³¹ Lee, T.-P. & Narayan, T. (2016), *Distributed Language Models Using RNNs* https://stanford.edu/~rezab/classes/cme323/S16/projects_reports/narayan_lee.pdf (aufgerufen am 27.09.2022)

4.4.2.3 Anwendung des Recurrent Neural Networks zur Vorhersage nächster Konstruktionszustände

Gemäß Abbildung 4.30 stellt der aktuelle Bauteilzustand die Grundlage für die Vorhersage des nächsten Konstruktionsschrittes dar. Für dessen Weiterverarbeitung mittels des RNNs wird dieser zunächst entsprechend aufbereitet. Dabei ist zu beachten, dass die je nach Umsetzungsstufe benötigten Datenkategorien für den aktuellen Konstruktionszustand den für das Training des RNN genutzten Datenkategorien entsprechen. In Umsetzungsstufe 1 ist hierfür das Konstruktionsvorgehen in Form der Sequenz der 3D-Geometrie notwendig. Für den gegebenen Konstruktionszustand kann diese Sequenz über den Strukturbaum des zugrunde liegenden CAD-Modells hergeleitet werden. Hierfür werden gemäß Kapitel 4.1.3 die bisher genutzten Konstruktionselemente sukzessive entfernt. Entsprechend der für das Training des RNNs gewählten Sequenzlänge Q muss folglich auch die Anzahl der $Q - 1$ zuletzt durchgeführten Konstruktionsschritte betrachtet werden³². Im Beispiel von Abbildung 4.30 wurde das RNN mit einer Sequenzlänge von 4 trainiert, entsprechend werden 3 Schritte in die Vergangenheit geschaut. Die erhaltene Sequenz von 3D-Geometrien wird gemäß Kapitel 4.2.3 zunächst in Punktwolken umgewandelt und schließlich mit dem gemäß Kapitel 4.3.2 trainierten *Autoencoder* in den latenten Raum überführt. Resultat sind die latenten Vektoren aller Zwischenzustände $(\vec{x}_i^{(t-(Q-2))}, \dots, \vec{x}_i^{(t)})$. In Umsetzungsstufe 2 werden neben der geometrischen Sequenz zudem die genutzten KEs extrahiert und entsprechend Kapitel 4.2.3 über ein *One-Hot-Encoding* dargestellt. Basierend auf dem gegebenen Input wird folglich über das RNN der nächste Konstruktionszustand in Form eines latenten Vektors prädiziert. Dieser kann mit Hilfe des *Autoencoders* wieder in eine Punktwolke umgewandelt und visualisiert werden oder dient direkt als Input für die Ähnlichkeitssuche (siehe 4.4.3).

³² Prinzipiell sind auch kürzere Sequenzen möglich. Je mehr Informationen über die vergangenen Konstruktionsschritte jedoch vorhanden sind, desto präziser können Vorhersagen getroffen werden.

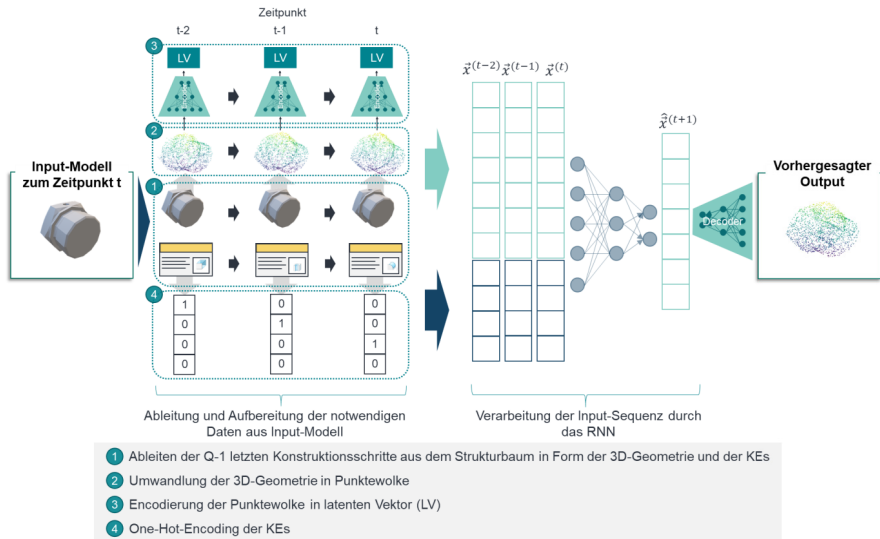


Abbildung 4.30: Anwendung des trainierten RNNs zur Vorhersage des nächsten Konstruktionszustandes für einen gegebenen Bauteilzustand

4.4.3 Integration in ein Gesamtsystem

Das entwickelte Assistenzsystem umfasst die zwei grundsätzlichen Funktionalitäten, die Ähnlichkeitssuche und das Vorhersagen nächster Konstruktionsschritte, die zunächst auch separat voneinander genutzt werden können. Beide Funktionalitäten können jedoch auch integriert angewendet werden. Abbildung 4.31 zeigt schematisch deren Zusammenspiel auf. Input für das Assistenzsystem ist der aktuelle Bauteilzustand zum Zeitpunkt t einer sich in der Konstruktion befindlichen Komponente i im CAD. Über den aktuellen Strukturbaum dieser Komponente können die bisher durchgeführten Konstruktionsschritte gemäß Kapitel 4.4.2.3 rekonstruiert und zur Weiterverarbeitung aufbereitet werden. Ebenso wird gemäß Kapitel 4.4.1 der geometrische Fußabdruck der Komponente i extrahiert.

Die Input-Sequenz des Konstruktionsvorgehens in Form der Sequenz der latenten Vektoren und codierten KEs dient nun zunächst als Eingabe für das gemäß Kapitel 4.4.2 auf den Komponenten j der Datenbasis trainierte RNN. Dieses gibt für den aktuellen Konstruktionszustand $\vec{x}_i^{(t)}$ eine Prädiktion für den nächsten Konstruktionszustand

$\widehat{x}_i^{(t+1)}$ aus. Dieser kann einerseits durch Dekodierung über eine Punktwolke visualisiert und dem Konstrukteur als Orientierung angezeigt werden. Andererseits dient dieser weiter fortgeschrittene Bauteilzustand schließlich als Input für die Ähnlichkeitssuche. Da der Zustand zum Zeitpunkt $t+1$ bereits mehr Informationen über die Geometrie enthält als der aktuelle Zustand zum Zeitpunkt t , lässt sich die Menge der ähnlichsten Modelle $S_{sim,i}$ aus der Datenbasis bereits präziser einschränken. Darüber hinaus kann es insbesondere für anfängliche Bauteilzustände sein, dass die Menge der ähnlichsten Modelle für den Zustand zum Zeitpunkt t leer ist. Durch die Nutzung der Prädiktion $\widehat{x}_i^{(t+1)}$ als Eingabe für die Ähnlichkeitssuche können durch die Vorausschau bereits zu früheren Zuständen ähnliche Komponenten gefunden werden. Gemäß Kapitel 4.4.1.2 wird folglich die Menge der (geometrisch) ähnlichsten Modelle $S_{sim,i}$ für den prädizierten Folgezustand $\widehat{x}_i^{(t+1)}$ bestimmt. Sofern Metainformationen zur Bestimmung der Ähnlichkeit genutzt werden, werden diese vom Zustand t übernommen. Entsprechend können die ähnlichsten Modelle j mit $\bar{x}_j \in S_{sim,i}$ anhand der auf Basis der Metainformationen berechneten Gesamtähnlichkeit $Sim_{ij,ges}$ in eine Rangfolge gebracht und dem Konstrukteur vorgeschlagen werden. Anhand der Menge der ähnlichsten Modelle erfolgt zudem gemäß Kapitel 4.4.1.3 die Bewertung der produktionsrelevanten Produkteigenschaften für den aktuellen Konstruktionsstand der Komponenten i zum Zeitpunkt t . Für den gegebenen Bauteilzustand i zum Zeitpunkt t bietet der entwickelte Ansatz dem Konstrukteur folglich mehrere Unterstützungsmöglichkeiten. Der prädizierte Folgezustand dient zunächst als Orientierung, welcher nächste Konstruktionsschritt üblicherweise durchzuführen ist. Werden basierend auf diesem Vorschlag ähnliche finale Komponenten gefunden, die ggf. wiederverwertet werden können, so kann der Konstrukteur durch die Nutzung dieser seine Konstruktionsaufgabe beenden. Andernfalls werden anhand der gefunden ähnlichen Komponenten für den aktuellen Bauteilzustand (bzw. des prädizierten Folgezustandes) Abweichungen produktionsrelevanter Produkteigenschaften aufgezeigt und dem Konstrukteur dadurch Hinweise auf eine mögliche Nicht-Produzierbarkeit mit den vorhandenen Fertigungsmitteln gegeben. Sind unter den ähnlichsten Modellen keine geeigneten Vorschläge enthalten, so kann der Konstrukteur durch das Durchführen eines nächsten Konstruktionsschrittes den Bauteilzustand weiter präzisieren. Dadurch ergibt sich eine aktualisierte Input-Sequenz

$(\vec{x}_i^{(t-(Q-1))}, \dots, \vec{x}_i^{(t+1)})$ für das RNN³³, auf deren Basis erneut der Folgezustand für $t+2$ vorhergesagt wird, der wiederum als Eingabe für die Ähnlichkeitssuche dient. Dieses iterative Vorgehen wird so lange wiederholt, bis ein ausreichend ähnliches Bauteil gefunden oder die Konstruktionsaufgabe durch das Hinzufügen weiterer Konstruktions-schritte beendet wird.

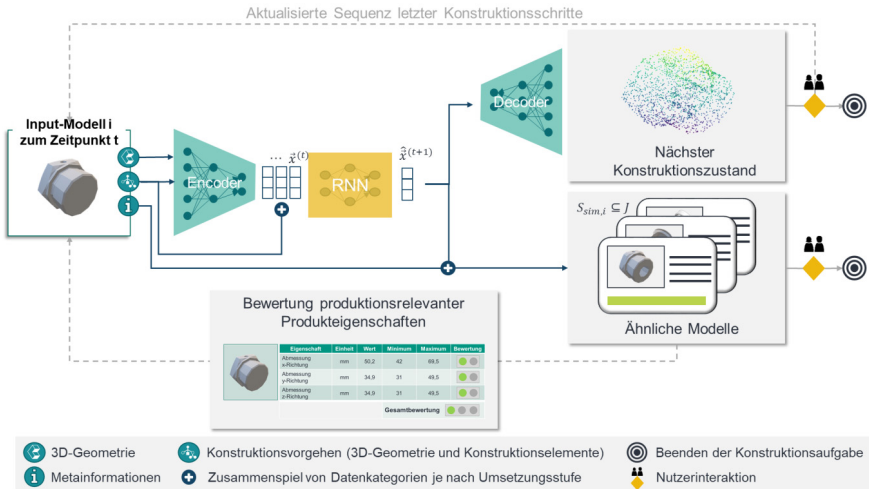


Abbildung 4.31: Zusammenspiel der Funktionalitäten als Gesamtsystem

³³ Der aktuelle Zeitpunkt entspricht nun $t+1$.

5 Evaluation

Das in Kapitel 4 beschriebene Vorgehen wird im Folgenden anhand eines beispielhaften Anwendungsfalles erprobt. Grundlage sind Daten mechanischer Komponenten von Sensoren aus dem aktuellen Produktportfolio eines Unternehmens. Diese Daten wurden gemeinsam mit dem Industriepartner im Rahmen des BMBF geförderten Verbundprojektes AIAx (Förderkennzeichen 01IS18048) erarbeitet.

5.1 Kategorisierung der Datenbasis

Die bereitgestellten Daten umfassen CAD-Modelle auf Einzelkomponentenebene im proprietären Dateiformat PRT der CAD Umgebung PTC Creo® Parametric von fünf verschiedenen Bauteilklassen. PTC Creo® Parametric nutzt die *Feature*-basierte Modellierung (Hackenschmidt, Hautsch & Kleinschrodt 2020), weshalb die einzelnen Bauteile aus KEs aufgebaut sind (vgl. Kapitel 2.1.3.1). Neben den CAD-Modellen sind noch weitere Daten aus dem PDM-System an einzelne Komponenten geknüpft (z.B. Kosten, Stückzahl). Gemäß Abbildung 5.1 können folglich Informationen aus allen drei definierten Datenkategorien extrahiert werden. Nachfolgend wird beschrieben, welche Informationen aus dem Datensatz für die Datenkategorien 3D-Geometrie, Konstruktionsvorgehen und Metainformationen zugrunde gelegt werden.

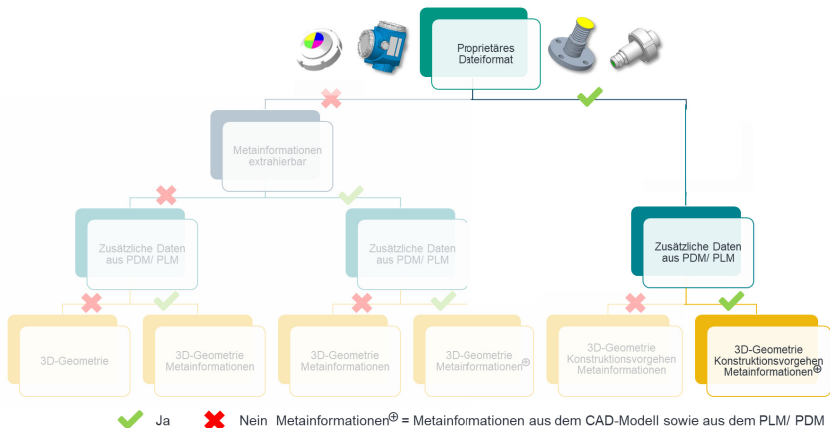


Abbildung 5.1: Extrahierbare Datenkategorien für den betrachteten Anwendungsfall

5.1.1 3D-Geometrie

In Abbildung 5.2 ist ein Überblick über den genutzten Datensatz mit insgesamt 957 CAD-Modellen aus fünf Bauteilklassen bereitgestellt. Grundlage für die Datenkategorie der 3D-Geometrie bilden die CAD-Modelle der fünf verschiedenen Bauteilklassen Adapter, Deckel, Flansch, Gehäuse und O-Ring im PRT-Format. Zur Weiterverarbeitung werden die CAD-Modelle in das STL-Format überführt. Jedem Modell ist dabei ein entsprechendes Klassen-Label zugeordnet. Die Klassenzuordnung erfolgte dabei anhand der hinterlegten Materialnummern. Die Ausrichtung der Modelle ist zufällig und folgt keiner Systematik.

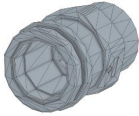

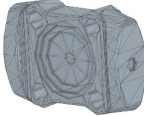
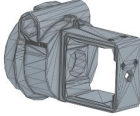

Adapter	Deckel	Flansch	Gehäuse	O-Ring
				
524 Modelle	101 Modelle	93 Modelle	96 Modelle	143 Modelle

Abbildung 5.2: Übersicht über die fünf Bauteilklassen der Datenbasis mit beispielhaftem Vertreter und Anzahl an Objekten je Klasse

5.1.2 Metainformationen

Die Datenbasis für relevante Metainformationen bilden die CAD-Modelle selbst sowie weitere Daten aus dem PDM-System des Industriepartners. Da die für die einzelnen CAD-Modelle im PDM-System hinterlegten Daten zu Kosten und Stückzahl nicht konsistent vorhanden und zudem fehlerbehaftet sind, werden sie für den weiteren Verlauf nicht näher betrachtet.

Die CAD-Modelle selbst beinhalten jedoch einige Metainformationen, die über die Softwareschnittstelle PTC Creo® Object TOOLKIT Java³⁴ aus den PRT-Dateien ausgelesen werden können. Eine Liste mit den 18 möglichen Parametern ist in Anhang E zu finden. Im Rahmen von Expertengesprächen mit dem Industriepartner wurde auf Basis dieser Parameterliste erarbeitet, welche der extrahierbaren Informationen zur Bewertung der Ähnlichkeit zweier CAD-Modelle und insbesondere auch für die spätere Pro-

³⁴ Parametric Technology GmbH (2021), PTC Creo Object TOOLKIT Java. Development Standalone/ Designated Computer

duzierbarkeit relevant sind. Für den betrachteten Anwendungsfall wurden diesbezüglich die Parameter maximale Abmessungen (*Bounding Box*), Volumen, Material (hinterlegt über die Dichte) und Toleranzen genannt. Davon sind die maximalen Abmessungen, Material sowie Toleranzen ausschlaggebend für die möglichen Fertigungsverfahren und liefern somit einen Anhaltspunkt für die Produzierbarkeit auf den vorhandenen Fertigungsmitteln³⁵. Die hinterlegten Dichten sind größtenteils verschiedenen Edelstahl- oder Kunststoffarten zuzuordnen. Vereinfacht werden daraus die zwei Materialklassen Kunststoff und Edelstahl zusammengefasst. Anhand der hinterlegten Dichten ergeben sich insgesamt vier verschiedene Materialklassen (Edelstahl, Kunststoff, Sonstiges, undefiniert)³⁶. Da die Klasse „Sonstiges“ vor allem fehlerhaft angegebene Dichten umfasst und für „undefiniert“ keine Dichte hinterlegt ist, werden die entsprechenden Modelle den Klassen „Kunststoff“ oder „Edelstahl“ zugeordnet. Die Zuteilung erfolgt dabei in Abhängigkeit der Häufigkeit von Edelstahl oder Kunststoff je Klasse. Als Dichtewert werden für Kunststoff die häufigste Dichte ($2,68 \cdot 10^{-6} \text{ kg/mm}^3$) bzw. für Edelstahl die zwei häufigsten Dichten ($7,85 \cdot 10^{-6} \text{ kg/mm}^3$ bzw. $8,0 \cdot 10^{-6} \text{ kg/mm}^3$) entsprechend ihrer Auftrittswahrscheinlichkeit hinterlegt. Die Verteilungen der Dichten sind in Anhang F.2 und Anhang F.3 zu finden. Die sich ergebende Verteilung der Materialarten je Klasse ist in Abbildung 5.3 visualisiert.

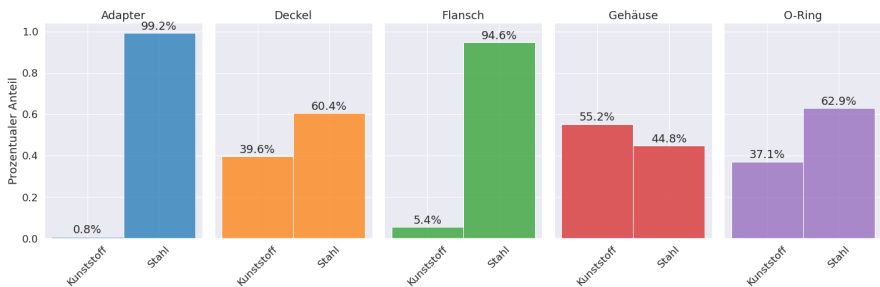


Abbildung 5.3: Bereinigter Anteil der Materialarten je Bauteilkategorie

³⁵ Lediglich ein Großteil der Adapter wird über Dreh- und Fräsmaschinen eigengefertigt. Alle weiteren Komponenten sind Zukaufteile und werden im Fall von halbfertigen Teilen im Unternehmen ggf. spanend bearbeitet.

³⁶ Für Stahl gilt $7,7 \cdot 10^{-6} \frac{\text{kg}}{\text{mm}^3} < \rho < 8,5 \cdot 10^{-6} \frac{\text{kg}}{\text{mm}^3}$, für Kunststoff gilt $\rho < 5 \cdot 10^{-6} \frac{\text{kg}}{\text{mm}^3}$. Alle weiteren Dichten werden als „Sonstiges“ definiert. Sind keine Dichten hinterlegt, wird die Materialklasse „undefiniert“ zugeordnet. Für die ursprüngliche Verteilung der vier Materialklassen siehe Anhang F.1.

Da die Toleranzen nicht konsistent in den CAD-Modellen hinterlegt sind, werden diese nicht weiter betrachtet. Initial erwogene Parameter, wie Oberflächengüte oder Trägheitsmomente, wurden von den Experten als untergeordnet eingestuft und werden daher nicht weiter berücksichtigt. Abbildung 5.4 zeigt beispielhaft die betrachteten Metainformationen für einen Adapter. Insgesamt werden je Komponente fünf Metainformationen betrachtet, von denen vier als produktionsrelevant eingestuft wurden.


	Metainformation	Einheit	Wert	Produktionsrelevant	Variablenart
Abmessung x-Richtung	mm	59,9	Ja	Numerisch	
Abmessung y-Richtung	mm	41,0	Ja	Numerisch	
Abmessung z-Richtung	mm	59,9	Ja	Numerisch	
Volumen	mm ³	69636,79822	Nein	Numerisch	
Dichte (Material)	[kg/mm ³]	8,0*10 ⁻⁶ (Kunststoff)	Ja	Numerisch	

Abbildung 5.4: Exemplarische Darstellung der Metainformationen für einen Adapter

5.1.3 Konstruktionsvorgehen

Gemäß Kapitel 4.1.3 bildet der Strukturbaum die Grundlage für das Abbilden des Konstruktionsvorgehens. Die im betrachteten Datensatz genutzten KEs der CAD-Software PTC Creo® Parametric können grundsätzlich in vier Kategorien eingeteilt werden (Hackenschmidt, Hautsch & Kleinschrodt 2020): volumenerzeugende KEs (z.B. Körper), konstruierende KEs (z.B. Bohrung), editierende KEs (z.B. Muster) und unterstützende KEs (z.B. Bezugsebene). Unterstützende KEs haben dabei keinen direkten Einfluss auf die Gestalt des 3D-Modells und werden daher nicht berücksichtigt. Mit Hilfe der Softwareschnittstelle PTC Creo® Object TOOLKIT Java können für jedes Modell die Sequenzen der genutzten KEs ausgelesen werden.

In Abbildung 5.5 ist eine Übersicht über die Verteilung der Sequenzlängen je Klasse gegeben. Eine tabellarische Übersicht ist zudem in Anhang G.2 zu finden. Es wird ersichtlich, dass die Sequenzlängen – bis auf die Klasse der O-Ringe – innerhalb der Klassen recht stark variieren. Insbesondere die Klasse der Gehäuse zeigt stark schwankende Längen auf. Die im Vergleich zu den anderen Klassen insgesamt längeren Sequenzen der Klasse Gehäuse lassen sich auf die komplexere Geometrie dieser Klasse zurückführen. Die starken Schwankungen innerhalb der Klasse sind durch die sehr unterschiedlichen Geometrien erklärbar. Im Gegensatz dazu lassen sich die sehr kurzen Sequenzen der Klasse O-Ring durch die einfache Gestalt dieser Klasse begründen. Für die Erstellung eines Großteils der O-Ringe wird lediglich ein Konstruktions-

schritt benötigt. Die Klasse der Deckel zeigt ebenfalls eine große Varianz in den Sequenzlängen auf. In der Klasse der Flansche lassen sich zwei Ansammlungen erkennen.

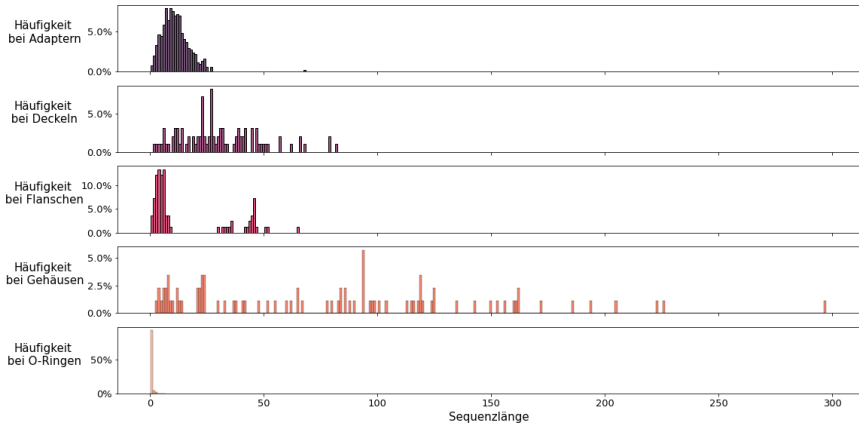


Abbildung 5.5: Verteilung der Häufigkeiten bestimmter Sequenzlängen je Bauteilklasse (in Anlehnung an A_Holtzwardt (2022))

Der erste Bereich hat eine durchschnittliche Sequenzlänge von sieben KEs, der zweite dagegen ca. 45. Diese beiden Bereiche lassen sich auf die Subklassen innerhalb der Flansche zurückführen, die deutlich unterschiedliche Geometrien aufweisen. In Abbildung 5.6 sind zwei repräsentative Vertreter beider Subklassen abgebildet.

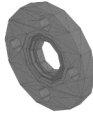

Runder Flansch	Seitenflansch
	
Anzahl KEs: 6	Anzahl KEs: 46

Abbildung 5.6: Exemplarische Vertreter von Flanschen aus den zwei üblichen Sequenzlängenbereichen

Insgesamt ergeben sich aus dem Datensatz 14 verschiedene KEs. In Abbildung 5.7 ist die relative Verteilung der KEs für die fünf Bauteilklassen dargestellt. Hier wird ersichtlich, dass für alle Bauteilklassen zunächst ein Volumen (per Extrusion oder Drehen) über das Konstruktionselement Körper erzeugt wird (vgl. Anhang G.3). Das Konstruktionselement Materialschnitt dagegen ist zum Entfernen eines Volumens (per Extrusion

oder Drehen) und wird in der Regel zur weiteren Ausdetaillierung des Modells genutzt³⁷. Die geringe Komplexität der Geometrie der Klasse O-Ringe lässt sich auch anhand der geringen Anzahl unterschiedlicher KEs erkennen.

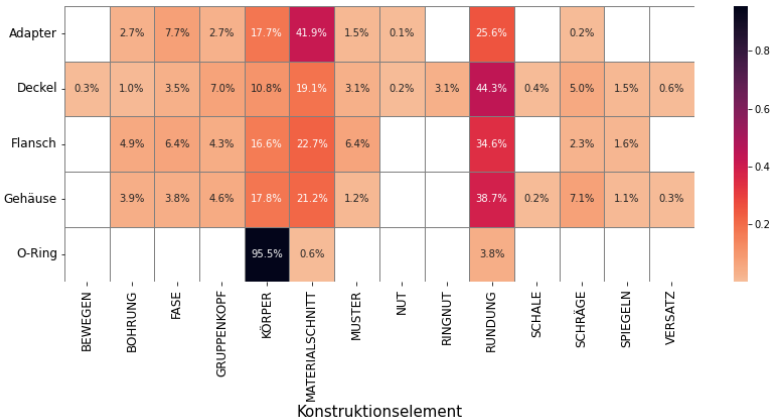


Abbildung 5.7: Anteil einzelner KEs je Klasse (in Anlehnung an A_Holtzwardt (2022))

Zum Abbilden der geometrischen Entwicklung des 3D-Modells über die Zwischenzustände (inkl. dem finalen Zustand) können mit Hilfe des PTC Creo® Object TOOLKIT Java automatisiert sukzessive die KEs aus dem Strukturbaum entfernt und der jeweilige Zustand als STL-Datei abgespeichert werden. Gleichzeitig erfolgt das Auslesen der relevanten Metainformationen (siehe Kapitel 5.1.2) für die Zwischenzustände, da diese für die spätere Validierung benötigt werden³⁸. In Abbildung 5.8 wird beispielhaft für ein Objekt aus der Klasse Adapter die entsprechende Sequenz der Konstruktionsschritte in Form der geometrischen Entwicklung und der zugehörigen KEs gezeigt. Die geometrische Sequenz des Adapters setzt sich dabei aus elf verschiedenen Zwischenzuständen zusammen. Der elfte Schritt entspricht dabei dem finalen Zustand. In Anhang H ist eine Übersicht der generierten Zwischenzustände je Bauteilkategorie dargestellt. Insgesamt werden aus den fünf betrachteten Komponentenklassen 18.734 Zwischenmodelle erzeugt.

³⁷ Sowohl dem Konstruktionselement Körper als auch dem Konstruktionselement Materialschnitt ist jeweils eine Skizze untergeordnet. Diese werden jedoch nicht als separates Konstruktionselement betrachtet.

³⁸ Da ein Teil der Zwischenzustände später als Testdaten dienen, für die ähnliche Modelle gesucht werden, werden diese Informationen benötigt. In der tatsächlichen Anwendung werden diese Informationen aus dem aktuellen Konstruktionsstand in Form der CAD-Datei (PRT-Format) extrahiert.



Abbildung 5.8: Sequenz der Zwischenzustände und KEs eines exemplarischen Adapters bestehend aus elf Schritten. Blau markiert ist der finale Zustand.

5.1.4 Zuordnung der Umsetzungsstufen für den Anwendungsfall

Gemäß Kapitel 4.1.4 kann die Funktionalität des Assistenzsystems in zwei Bereiche eingeteilt werden, die Ähnlichkeitssuche sowie das Vorschlagswesen nächster Konstruktionsschritte. Für den betrachteten Anwendungsfall sind neben der Datenkategorie 3D-Geometrie noch weitere Metainformationen verfügbar, die teilweise als produktionsrelevant eingestuft wurden. Folglich ist die Ähnlichkeitssuche gemäß Abbildung 5.9 mit der höchsten Stufe umsetzbar.

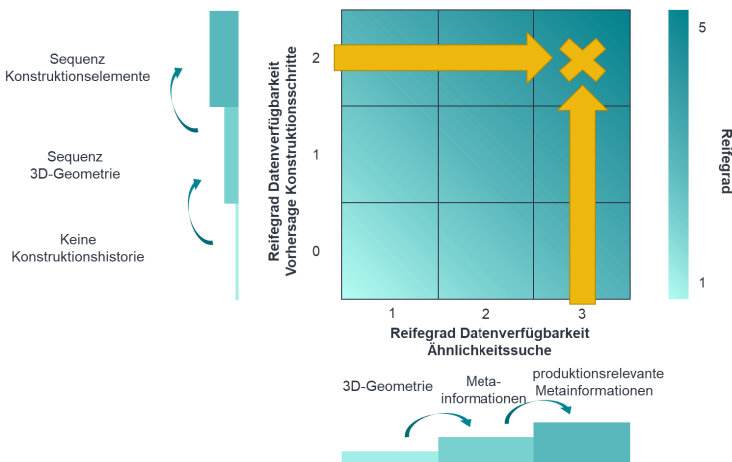


Abbildung 5.9: Umsetzbarer Reifegrad des Assistenzsystem für den Anwendungsfall

Die Daten der Kategorie Konstruktionsvorgehen liegen in Form der Sequenz der 3D-Geometrie vor, zudem sind die entsprechenden KEs vorhanden. Somit lässt sich auch das Vorschlagswesen nächster Konstruktionsschritte mit der höchsten Stufe realisieren. Insgesamt kann mit dem vorhandenen Datensatz der höchste Reifegrad des Assistenzsystems implementiert werden. Aus Gründen der Veranschaulichung werden jedoch in den nachfolgenden Kapiteln auch die Ergebnisse aufgezeigt, die für geringere Reifegrade erreichbar sind.

5.2 Aufbereitung der Datenbasis zur Repräsentation des Wissens

Die zuvor identifizierten Daten werden nun gemäß Kapitel 4.2 für die Weiterverarbeitung mittels der ML-Verfahren aufbereitet. Um die Effekte der unterschiedlichen *Autoencoder*-Architekturen sowie der Datenverfügbarkeit und damit der möglichen Umsetzungsstufen zu veranschaulichen, werden drei Varianten der Datenbasis betrachtet. Diese werden in Anhang I detailliert aufgeführt. Der Datensatz „Finale_Komponenten_NS0“ dient rein zur Veranschaulichung des Effektes einer Standardorientierung für das Training des *Autoencoders*, weshalb nur die 3D-Geometrien finaler Komponenten ohne Standardorientierung betrachtet werden. Der Datensatz „Finale_Komponenten_SO“ enthält ebenfalls nur die finalen 3D-Geometrien der Komponenten, die zudem standardorientiert wurden, sowie die zugehörigen Metainformationen. Im Datensatz „Gesamt_SO“ wird zudem noch das Konstruktionsvorgehen über die Sequenz der 3D-Geometrie sowie der KEs berücksichtigt. Dieser Datensatz umfasst folglich nicht nur die 3D-Geometrien der finalen Komponenten, sondern auch sämtliche 3D-Geometrien der Zwischenzustände, siehe Abbildung 5.10. Alle 3D-Geometrien liegen zudem in Standardorientierung vor.

Für die spätere Anwendung der ML-Verfahren werden alle drei Datensätze in Trainings- und Testdaten aufgeteilt. Für ein möglichst repräsentatives Set an Testdaten werden aus jeder Bauteilkategorie 5 % der finalen Modelle ausgewählt, als Trainingsdaten folgen entsprechend 95 %³⁹. Sowohl für die Anwendung des *Autoencoders* als auch für die Sequenzanalyse zur Vorhersage nächster Konstruktionsschritte wird dieselbe Aufteilung herangezogen.

³⁹ Im Fall des Datensatzes „Gesamt_SO“ erfolgt die Aufteilung der Zwischenzustände in Trainings- und Testdaten entsprechend dem zugehörigen finalen Modell.

Die 3D-Geometrien der finalen Komponenten aller fünf Bauteilklassen, die in allen drei aufgeführten Datensatzkonstellationen enthalten sind, werden gemäß Kapitel 4.2.1 zunächst in Punktwolken überführt. Die zusätzlich im Datensatz „Gesamt_SO“ enthaltenen Zwischenzustände aller Komponenten werden ebenso in Punktwolken umgewandelt. Dabei werden je 3D-Geometrie in Anlehnung an Yang et al. (2018) 2.048 Punkte auf den Eckpunkten und Flächen des Polygonnetzes verteilt. Die Anzahl an Eckpunkten variiert dabei zwischen den unterschiedlichen Bauteilklassen (siehe Anhang J).

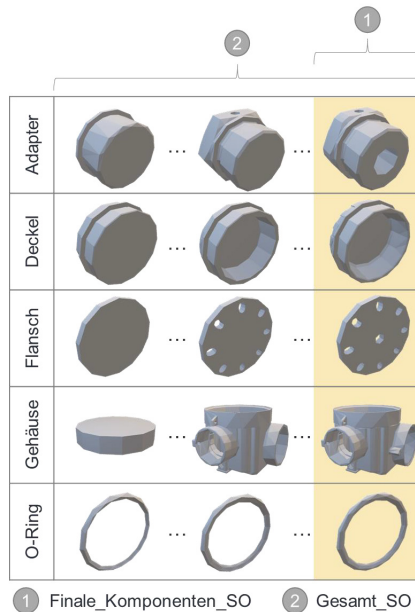



Abbildung 5.10: Zusammenhang zwischen den Datensätzen Gesamt_SO und Finale_Komponenten_SO (orange hinterlegt)

Für die Datensätze „Finale_Komponenten_SO“ und „Gesamt_SO“ werden die 3D-Geometrien zudem gemäß dem in Kapitel 4.3.1.1 gezeigten Verfahren in eine Standardorientierung überführt. Für den Datensatz „Gesamt_SO“, der neben dem finalen Zustand einer Komponente auch alle Zwischenzustände enthält, wird dabei berücksichtigt, dass alle Zwischenzustände einer Komponente – trotz möglicherweise variierender *Bounding Box* – in der gleichen Standardorientierung vorliegen (siehe Kapitel 4.3.2). In

einem letzten Schritt werden die Punktwolken für alle betrachteten Datensätze normiert und zentriert.⁴⁰ Das jeder 3D-Geometrie zugeordnete Klassen-Label wird in ein *One-Hot-Encoding* überführt.

Neben der 3D-Geometrie umfassen die Datensätze „Finale_Komponenten_SO“ und „Gesamt_SO“ zusätzlich die in Kapitel 5.1.2 aufgezeigten Metainformationen. Als numerische Daten werden die Abmessungen in x-, y- und z-Richtung sowie das Volumen betrachtet. Das Material wird zwar gemäß Kapitel 5.1.2 in die Kategorien Kunststoff und Edelstahl eingeteilt, allerdings wird es dennoch über die entsprechende Dichte repräsentiert und daher ebenso als numerische Variable betrachtet. In Anhang K ist eine Übersicht über die Verteilungen der maximalen Abmessungen, des Volumens und der Dichte dargestellt. Da sich die Größenordnungen stark unterscheiden, werden diese für eine bessere Vergleichbarkeit über eine Min-Max-Normierung standardisiert. Dafür wird je Variable jeweils die minimale und maximale Ausprägung aus dem Datensatz herangezogen. Abbildung 5.11 zeigt für die beispielhafte Komponente aus Abbildung 5.4 die normierten numerischen Metainformationen.



Metainformation	Normierter Wert	Max	Min
Abmessung x-Richtung	0,121	482,00 mm	1,89 mm
Abmessung y-Richtung	0,174	232,36 mm	0,72 mm
Abmessung z-Richtung	0,124	469,92 mm	1,89 mm
Volumen	0,035	1986848,49 mm ³	7,84 mm ³
Dichte (Material)	0.9205	8,69*10 ⁻⁶ kg/mm ³	7,85*10 ⁻⁹ kg/mm ³

Abbildung 5.11: Exemplarische Darstellung der normierten Metainformationen für einen Adapter

Für den Datensatz „Gesamt_SO“ liegen neben der Sequenz der 3D-Geometrien gemäß Kapitel 5.1.3 auch die entsprechenden Sequenzen der KEs je Komponente vor. Diese sind kategorialer Natur und können gemäß Kapitel 5.1.3 in 14 Klassen eingeteilt werden. Dabei entspricht jedes KE einer Klasse. Entsprechend dem Vorgehen zur Aufbereitung kategorialer Daten wird jedes KE schließlich über ein *One-Hot-Encoding* dargestellt. Die Sequenzen der KEs werden folglich über eine Sequenz von *One-Hot-Encoding* Vektoren repräsentiert.

⁴⁰ Die Erzeugung der Punktwolken sowie Normierung und Zentrierung finden bei Anwendung des *Autoencoders* zur Laufzeit statt. Dadurch wird gewährleistet, dass für ein Objekt die generierten Punktwolken je Trainings-epoche unterschiedlich sind, wodurch eine bessere Generalisierbarkeit erreicht werden kann.

Für die Sequenzanalyse mittels RNN werden darüber hinaus aus den Sequenzen des Konstruktionsvorgehens (3D-Geometrie und KEs) über das *Sliding Window Prinzip* Sequenzabschnitte mit Fenstergröße Q generiert. Da die Länge der Sequenzabschnitte einen Hyperparameter für das Training darstellt, wird diese anhand der Fenstergröße Q variiert. Sequenzen, die kürzer sind als die gewählte Fenstergröße, müssen verworfen werden. In Anhang L ist aufgezeigt, wie viele Komponenten und resultierende Sequenzabschnitte sich in Abhängigkeit der Fenstergröße ergeben. Bei einer Fenstergröße von 2 findet ein Großteil der Daten Verwendung, lediglich die Mehrheit der O-Ringe, die nur über einen Konstruktionsschritt⁴¹ verfügen, muss verworfen werden (siehe auch Abbildung 5.5 und Anhang G.2). Im Gegenzug dazu wird ab einer Fenstergröße von 12 neben den O-Ringen auch ein beträchtlicher Anteil der Adapter aussortiert. Da bei einer Sequenzabschnittslänge von zwei kaum Muster zu erwarten und ab einer Länge von 12 ein Großteil der Daten verworfen werden muss, werden für den betrachteten Datensatz die Fenstergrößen 4 und 8 betrachtet.

5.3 Automatisierte Extraktion relevanter Eigenschaften mittels Autoencoder

Zur Extraktion der relevanten Eigenschaften aus den 3D-Geometrien des betrachteten Anwendungsfalls werden nachfolgend die in Kapitel 4.3.1 vorgestellten *Autoencoder*-Architekturen herangezogen. Da es sich bei den betrachteten Daten um nicht vorausgerichtete CAD-Modelle mit verfügbaren Klassen-Labeln handelt, können prinzipiell alle vorgestellten Ansätze darauf angewandt werden. Aufgrund der nicht-ausgerichteten Modelle ist jedoch zu erwarten, dass über Ansatz 1 ohne vorgeschaltete Standardausrichtung keine zufriedenstellenden Repräsentationen erlernt werden. Um dennoch die Unterschiede zwischen den verschiedenen Architekturen zu veranschaulichen, werden nachfolgend die erlernten Repräsentationen mit allen Ansätzen vorgestellt. Diese werden exemplarisch anhand der finalen 3D-Geometrien der Datensätze „Finale_Komponenten_NS0“ und „Finale_Komponenten_SO“ (siehe Anhang I) aufgezeigt.

In Kapitel 5.3.2 wird der *Autoencoder* schließlich auf den Sequenzen der 3D-Geometrien (inkl. finaler 3D-Geometrien) des Datensatzes „Gesamt_SO“ (siehe Anhang I) trainiert, wodurch die Trainingsdatenmenge deutlich erhöht wird.

⁴¹ Die Vorhersage nächster Konstruktionsschritte ist nur ab einer Sequenzlänge größer 1 möglich.

5.3.1 Anwendung des Autoencoders auf die 3D-Geometrien finaler Modelle

In den nachfolgenden Kapiteln werden die aufbereiteten 3D-Geometrien in Form von Punktwolken aus den zwei eingeführten Datensätzen mit nur finalen Komponenten für Untersuchungen der verschiedenen *Autoencoder*-Ansätze herangezogen. Die über den trainierten Encoder erzeugten latenten Vektoren werden schließlich auf Basis der in Kapitel 4.3.3 vorgestellten Verfahren auf deren Repräsentativität hin überprüft. Für diese Untersuchungen werden die latenten Vektoren zunächst über t-SNE visualisiert und anschließend eine Clusteranalyse über *k-Means* durchgeführt. Zudem erfolgt eine Klassifikation über *Random Forest*.

In einer ersten Untersuchung wird die Basis-Architektur des *Autoencoders* ohne eine Vorausrichtung der Punktwolken erprobt. Hierfür werden die 3D-Geometrien des Datensatzes „Finale_Komponenten_NS0“ herangezogen. Darauffolgend wird der Einfluss durch die Standardausrichtung der Input-Daten untersucht, was anhand der 3D-Geometrien des Datensatzes „Finale_Komponenten_SO“ gezeigt wird. Diese werden ebenso in einem weiteren Experiment genutzt, um den Effekt durch die zusätzliche Berücksichtigung von Klassen-Labels aufzuzeigen.

5.3.1.1 Autoencoder ohne Standardausrichtung

Input für den *Autoencoder* sind die Punktwolken der entsprechenden Komponenten aus den fünf verschiedenen Bauteilklassen des Datensatzes „Finale_Komponenten_NS0“. Diese werden in ihrer ursprünglichen Ausrichtung in den *Autoencoder* eingespeist. Die für das Training gewählten Hyperparameter können Anhang M.1 entnommen werden. In Abbildung 5.12 sind entsprechende Rekonstruktionen aus dem Testdatensatz visualisiert. Generell wird ersichtlich, dass die ursprüngliche geometrische Form in der Rekonstruktion zwar abgebildet werden kann, Details jedoch nicht ausreichend dargestellt werden.

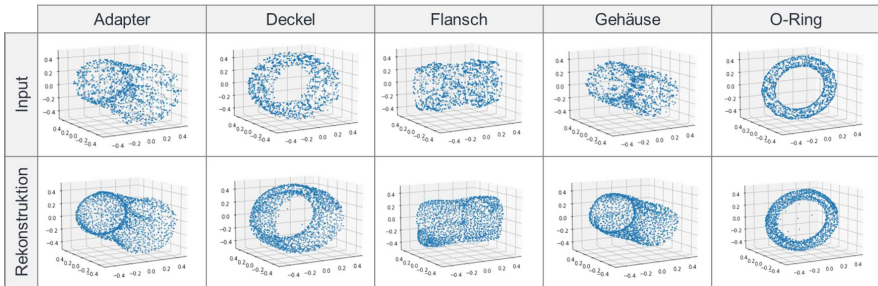


Abbildung 5.12: Beispielhafte Rekonstruktionen von Input-Punktewolken für den Autoencoder ohne Standardausrichtung

Die Visualisierung der latenten Vektoren über *t-SNE* ist in Abbildung 5.13 (a) dargestellt. Hier fällt zunächst auf, dass sich lediglich die Klasse der O-Ringe in Form eines länglichen Clusters deutlich von den anderen Komponenten separiert.

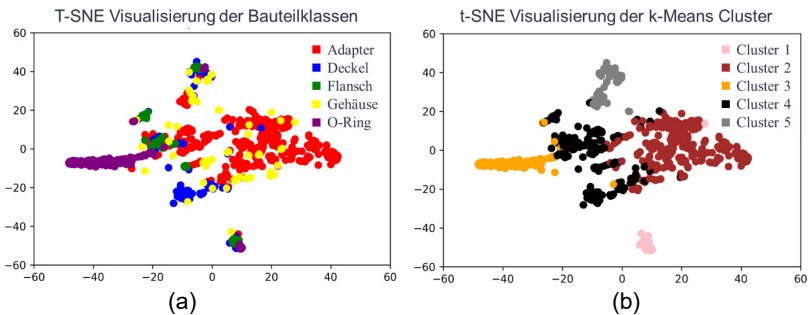


Abbildung 5.13: *t-SNE* Visualisierung der latenten Vektoren der fünf Bauteilklassen (a) sowie der gefundenen Cluster über *k-Means* (b) für den Autoencoder ohne Standardausrichtung

Für die Klasse der Flansche (grün) können mehrere kleine Ansammlungen identifiziert werden. Für ein besseres Verständnis werden in Abbildung 5.14 entsprechende Objekte aus diesen Bereichen dargestellt. Dabei fällt auf, dass innerhalb der Bereiche entweder runde (Bereich III bis V) oder eckige (Bereich I und II) Flansche (Seitenflansche) vorliegen. Eine weitere Unterteilung dieser zwei Gruppen erfolgt offensichtlich anhand der unterschiedlichen Orientierung der Teile. In Bereich V sind vertikal ausgerichtete Flansche anzutreffen, in Bereich III dagegen zwar auch vertikal ausgerichtete, jedoch um 90 Grad gedrehte. Alle Flansche der Bereiche I, II und IV sind dagegen horizontal orientiert. Dabei unterscheiden sich die zwei Bereiche I und II der eckigen Seitenflansche ebenfalls durch eine um 90 Grad gedrehte Ausrichtung der Objekte. Daraus lässt

sich schließen, dass der *Autoencoder* geometrisch ähnliche Komponenten aufgrund unterschiedlicher Ausrichtung voneinander separiert und daher, wie zuvor angenommen, keine ausreichende Repräsentativität erlernen konnte. Für die übrigen Bauteilklassen lässt sich keine eindeutige Clusterbildung erkennen. Entsprechend ist es über eine Clusterbildung mit *k-Means* auch nicht möglich, Cluster in Form der zugrundeliegenden Klassen zu identifizieren. Die erhaltenen Cluster sind in Abbildung 5.13 (b) dargestellt, die Zuordnung der Klassen auf die Cluster ist in Anhang N.1 zu finden. Zur Bewertung der Cluster-Güte wird der in Kapitel 2.2.3.3 eingeführte *V-Score*, bestehend aus *Homogeneity Score* (HS) und *Completeness Score* (CS), eingesetzt. Dafür werden 100 *k-Means* Durchläufe ausgeführt und der Mittelwert der jeweiligen *Scores* berechnet. Es kann lediglich ein HS von knapp 36 % erreicht werden, da innerhalb der gefundenen Cluster Objekte verschiedener Bauteilklassen enthalten sind. Damit einhergehend liegt auch der CS lediglich bei ca. 35 %, da ein Großteil der Objekte einer Bauteilkategorie auf verschiedene Cluster aufgeteilt wird. Als Resultat folgt insgesamt ein *V-Score* von etwa 35 %. Lediglich die Objekte der Klasse O-Ringe werden in großen Teilen einem gemeinsamen Cluster zugeordnet.

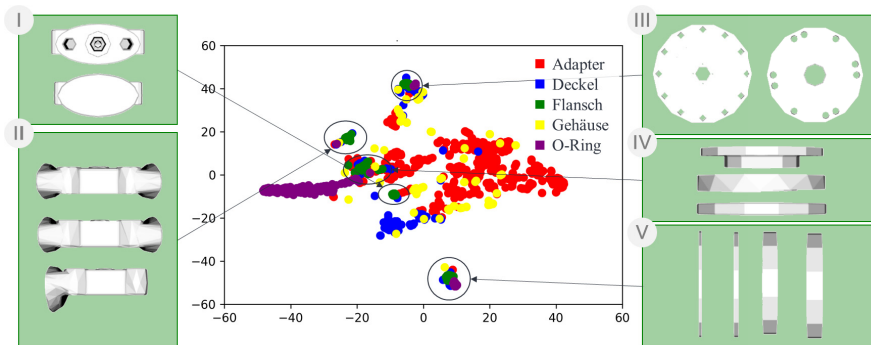


Abbildung 5.14: Darstellung exemplarischer Vertreter von Flanschen aus den visuellen Ansammlungen in der *t-SNE* Visualisierung für den *Autoencoder* ohne Standardausrichtung

Deutlich besser fallen die Ergebnisse der Klassifikation über *Random Forest* mit einer Genauigkeit von 90 % aus. Die entsprechende Konfusionsmatrix ist in Anhang O.1 dargestellt. Hier wird nochmals deutlich, dass insbesondere zwischen den Klassen Gehäuse, Deckel und Adapter Verwechslungen auftreten. Die Klassen O-Ring und Flansch dagegen werden korrekt klassifiziert.

Insgesamt kann aus den Untersuchungen gefolgert werden, dass die stark variierenden Orientierungen der Objekte ein einheitliches Erlernen der relevanten geometrischen Merkmale nicht ermöglichen. Offensichtlich können zwar klassenspezifische Eigenschaften schon ausreichend extrahiert werden, wofür die bereits hohe Genauigkeit der Klassifizierung spricht. Dennoch scheint die Codierung dieser Merkmale innerhalb einer Klasse aufgrund der unterschiedlichen Orientierungen stark zu variieren, weshalb die latenten Vektoren innerhalb einer Klasse im Raum verstreut sind und folglich die Ergebnisse der Clusteranalyse nur mäßig ausfallen. Dieser Nachteil wird durch die Ausrichtung der Modelle in eine Standardorientierung behoben, was im nachfolgenden Kapitel beschrieben wird.

5.3.1.2 Autoencoder mit Standardausrichtung

Vor dem eigentlichen Training des *Autoencoders* werden die Trainings- und Testobjekte aus dem Datensatz „Finale_Komponenten_SO“ zunächst in eine Standardorientierung überführt. In Abbildung 5.15 wird der Effekt der Standardausrichtung am Beispiel der Klasse Flansch gezeigt. Diese war in der zuvor beschriebenen Untersuchung aufgrund der unterschiedlichen Orientierung in mehrere Cluster eingeteilt worden. Resultat der Ausrichtung ist es, dass alle Flansche nun senkrecht gestellt mit der Anflanschfläche nach vorne oder hinten angeordnet sind.

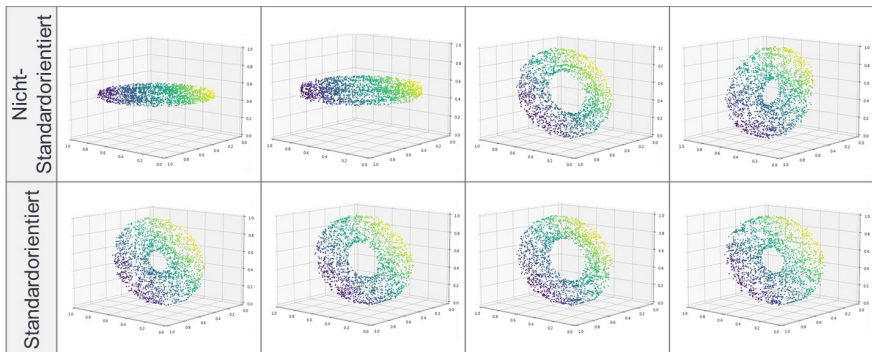


Abbildung 5.15: Effekt der Standardorientierung am Beispiel von runden Flanschen
 Der *Autoencoder* wird gemäß Anhang M.1 mit den gleichen Hyperparametern wie in der vorherigen Untersuchung trainiert. Die resultierende Visualisierung der latenten Vektoren über t-SNE ist in Abbildung 5.16 (a) dargestellt. Es fällt auf, dass die verschiedenen Bauteilklassen besser voneinander separiert werden. Insbesondere die Klasse

der Flansche wird nun in zwei große Cluster eingeteilt, die entweder runde oder eckige Flansche beinhalten. Durch die vorherige Standardausrichtung kann folglich vermieden werden, dass Objekte unterschiedlicher Ausrichtung als geometrisch verschieden interpretiert werden. In der Klasse der Adapter können ebenfalls drei größere Ansammlungen identifiziert werden, wie in Abbildung 5.17 dargestellt. In Bereich I sind insbesondere längliche und schmale Adapter anzutreffen, in II und III dagegen sind die Adapter deutlich kürzer.

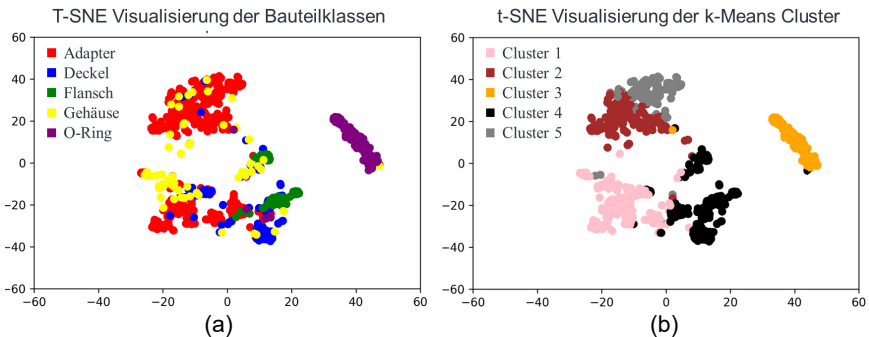


Abbildung 5.16: t-SNE Visualisierung der latenten Vektoren der fünf Bauteilklassen (a) sowie der gefundenen Cluster über k-Means (b) für den Autoencoder mit Standardausrichtung

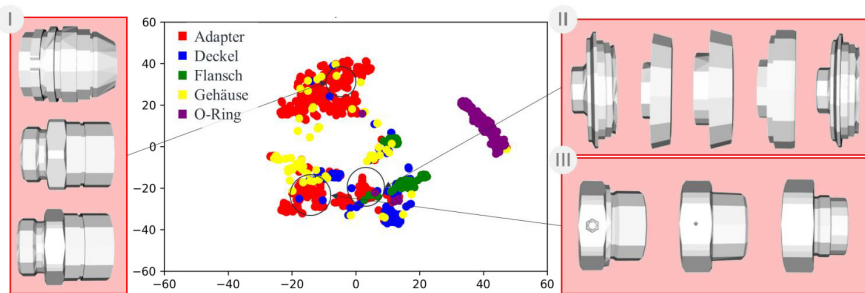


Abbildung 5.17: Darstellung exemplarischer Vertreter von Adaptern aus den visuellen Ansammlungen in der t-SNE Visualisierung für den Autoencoder mit Standardausrichtung

Nichtsdestotrotz sind die latenten Vektoren der Klassen Deckel und Gehäuse nach wie vor im latenten Raum sehr verstreut. Eine nähere Betrachtung der einzelnen Ansamm-

lungen verdeutlicht jedoch, dass diese Streuung auf die starken geometrischen Unterschiede innerhalb der Bauteilklassen zurückzuführen ist. In Abbildung 5.18 auf der linken Seite werden beispielhafte Deckel dargestellt, die sich innerhalb von Bereich I der Adapter in Abbildung 5.17 befinden. Die längliche Form dieser Deckel ist dabei geometrisch sehr ähnlich zu der entsprechenden Form der länglichen Adapter. Zum Vergleich sind in Abbildung 5.18 rechts einige Objekte aus der größeren Ansammlung der Deckel dargestellt, die die übliche Form eines Deckels aufweisen. Die über *k-Means* erhaltene Clusterlösung ist in Abbildung 5.16 (b) dargestellt, die Zuordnung der Klassen auf die Cluster ist in Anhang N.2 zu finden. Die Ergebnisse der *k-Means* Clustering können zwar mit einem HS von ca. 46 %, einen CS von ca. 39 % und einem *V-Score* von ca. 43 % deutlich verbessert werden, dennoch ist eine Unterscheidung zwischen den einzelnen Bauteilklassen, insbesondere bei Deckeln und Gehäusen, durch die sehr stark variierenden Geometrien schwer (siehe auch Ergebnisse der Sequenzlängen aus Kapitel 5.1.3). Bei der Klassifizierung mittels *Random Forest* (siehe Anhang O.2) kann folglich auch keine Verbesserung der Genauigkeit erzielt werden. Auch hier können hauptsächlich Deckel und Gehäuse nicht korrekt klassifiziert werden. Durch die Hinzunahme der Information über die jeweilige Bauteilkategorie soll eine Separierung zwischen den Klassen trotz ähnlicher Geometrien verbessert werden, worauf im nachfolgenden Kapitel eingegangen wird.

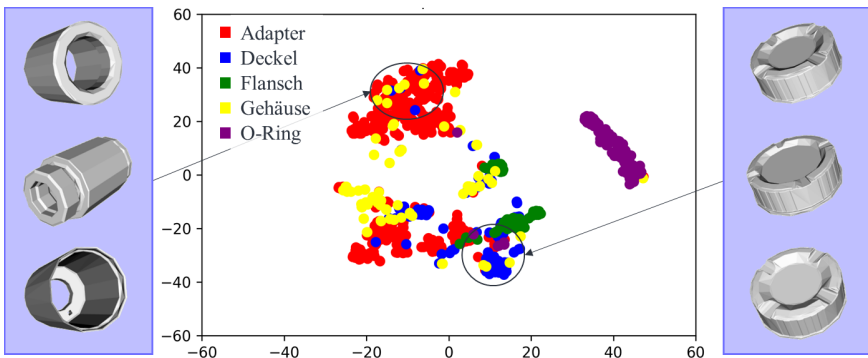


Abbildung 5.18: Darstellung exemplarischer Vertreter von Deckeln aus verschiedenen Bereichen der *t-SNE* Visualisierung für den Autoencoder mit Standardausrichtung

5.3.1.3 Autoencoder mit zusätzlichem Klassifikationsverlust

Gemäß Kapitel 4.3.1.2 wird für den *Autoencoder* mit Klassifikationsverlust die Basis-Architektur des *Autoencoders* um ein zusätzliches MLP erweitert. Dieses soll basierend auf den durch den *Encoder* erzeugten latenten Vektoren die entsprechende Bauteilklasse vorhersagen. Als Datengrundlage wird der Datensatz „Finale_Komponenten_SO“ herangezogen, der bereits standardausgerichtete Objekte enthält. Das Training der gesamten Architektur wird dabei in zwei Phasen aufgeteilt. Für die erste Phase des Trainings sind Daten mit entsprechendem Klassen-Label notwendig. Da in einem realen industriellen Anwendungsfall im Normalfall nur ein kleiner Teil der Daten über ein solches Label verfügt, werden 10 % des betrachteten Trainingsdatensatzes hierfür herangezogen. Die restlichen 90 % werden erst in der zweiten Phase des Trainings, die rein auf dem Rekonstruktionsverlust basiert, genutzt. Die gewählten Hyperparameter sind in Anhang M.1 dargestellt. Für die Anzahl an Trainingsepochen sowie die Lernrate werden für die zwei Phasen unterschiedliche Ausprägungen gewählt. Um die in Phase 1 erlernten Repräsentationen, die für eine Unterscheidung zwischen Klassen optimiert sind, auch auf Phase 2 zu übertragen, wird die Lernrate hier deutlich größer gewählt. In Phase 2 dagegen sollen nur noch geringfügigere Anpassungen durchgeführt werden.

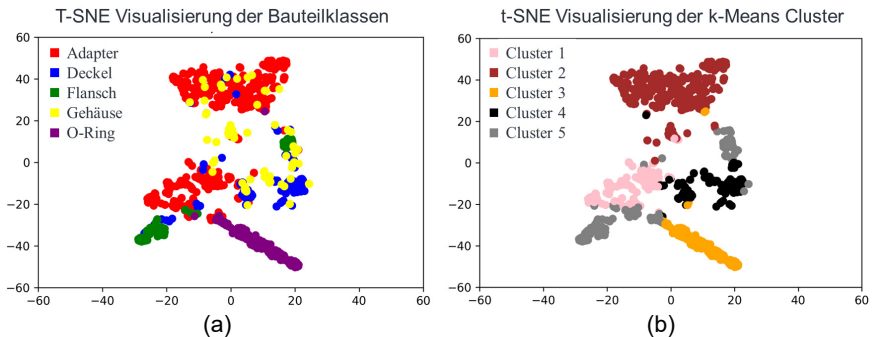


Abbildung 5.19: t-SNE Visualisierung der latenten Vektoren der fünf Bauteilklassen (a) sowie der gefundenen Cluster über k-Means (b) für den Autoencoder mit Klassifikationsverlust

Abbildung 5.19 (a) zeigt die Visualisierung der erhaltenen latenten Vektoren mittels t-SNE. Im Vergleich zu Kapitel 5.3.1.2 kann jedoch auch mit dieser Architektur in der rein visuellen Bewertung keine weitere Verbesserung für die Klassen Deckel und Ge-

häuse erzielt werden. Insbesondere die Objekte der Klasse Gehäuse scheinen im latenten Raum aufgrund der stark variierenden Geometrien sehr verstreut zu sein. In Abbildung 5.20 sind zur Veranschaulichung beispielhafte Objekte der Klasse Gehäuse und Deckel aus verschiedenen Bereichen des latenten Raumes dargestellt. Die sehr länglichen Gehäuse aus Bereich IV werden nahe des oberen Adapter-Clusters mit länglichen Vertretern angesiedelt. Die typische Form eines Gehäuses in Bereich I bildet dagegen eine kleinere, sich separierende Ansammlung. Sehr flache und runde Gehäuse (Bereich II) dagegen werden nahe der Deckel platziert. Im Gegenzug werden Deckel-Geometrien aus Bereich III, die sehr einem runden Flansch ähneln, sehr nahe an dem entsprechenden Flansch-Cluster lokalisiert. Hier ist jedoch anzumerken, dass bei diesen Objekten durchaus eine Falsch-Klassifizierung vorliegen kann. Da es sich bei solchen Objekten folglich um Ausreißer handelt, ist ein Erlernen eines Zusammenhangs zwischen der Geometrie und dem entsprechenden Klassen-Label nur sehr schwer, da kaum entsprechende Trainingsobjekte im Datensatz vorhanden sind.

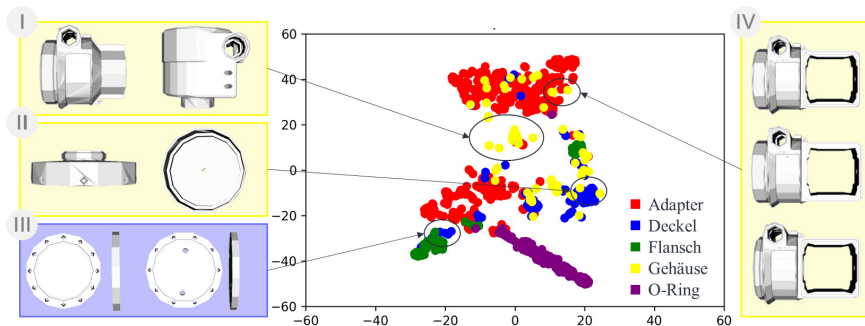


Abbildung 5.20: Darstellung exemplarischer Vertreter von Gehäusen und Deckeln aus verschiedenen Bereichen der t-SNE Visualisierung für den Autoencoder mit Klassifikationsverlust

Die Durchführung von *k-Means* zeigt, dass durch den zusätzlichen Klassifikationsverlust eine Steigerung des HS auf knapp 54 %, des CS auf 48% und des V-Score auf knapp 51 % erreicht werden kann. In Abbildung 5.19 (b) wird außerdem deutlich, dass die gefundene Clusterlösung nun deutlich mehr den tatsächlichen Klassen entspricht. Das trifft insbesondere auf die Klassen O-Ring, Flansch und Adapter zu. Die Zuordnung der Klassen auf die Cluster ist in Anhang N.3 zu finden. Für Gehäuse und Deckel kann

jedoch nach wie vor keine eindeutige Clusterzuordnung gefunden werden. Die Genauigkeit der Klassifikation mittels *Random Forest* kann ebenfalls auf 92 % leicht verbessert werden. Die entsprechende Konfusionsmatrix ist in Anhang O.3 zu finden.

Generell ist jedoch anzumerken, dass die zweidimensionale Darstellung über t-SNE mit einem gewissen Informationsverlust verbunden ist und daher die tatsächliche Anordnung im latenten Raum nur näherungsweise widerspiegeln kann.

5.3.2 Anwendung des Autoencoders auf die sequentiellen Daten des Konstruktionsvorgehens

Anhand von Kapitel 5.3.1 wurde gezeigt, dass für den betrachteten Datensatz der *Autoencoder* mit standardorientierten Input-Objekten sowie der *Autoencoder* mit zusätzlichem Klassifikationsverlust gute Ergebnisse auf den finalen Komponenten liefern. Da in den meisten industriellen Anwendungen keine Daten mit Label vorliegen, wird für die weiteren Untersuchungen der *Autoencoder* mit Standardausrichtung herangezogen. Als Daten werden nun neben den finalen Komponenten ebenfalls die zugehörigen Zwischenzustände herangezogen, die im Datensatz „Gesamt_SO“ bereits standardorientiert vorliegen.

Die für das Training gewählten Hyperparameter sind in Anhang M.2 dargestellt. Aufgrund der deutlich größeren Datenmenge wurde im Vergleich zu Kapitel 5.3.1 die *Batch Size* erhöht, um das Training zu beschleunigen. Abbildung 5.21 (a) zeigt die t-SNE Visualisierung der erlernten latenten Repräsentationen aller Zwischenzustände. Auf den ersten Blick scheinen die verschiedenen Objekte und deren zugehörige Zwischenzustände einer Klasse in keine eindeutigen Cluster eingeteilt zu werden. Lediglich sämtliche Objekte der Klasse O-Ring bilden eine gemeinsame Ansammlung. Grund für die zerstreute Darstellung sind die vielen verschiedenen Zwischenzustände einer finalen Komponente, die zu stark variierenden Geometrien führen. Da die Klasse der O-Ringe meist nur über zwei Zwischenzustände verfügt, fällt die Variation hier geringer aus. Am zerstreutesten scheinen die Objekte der Klasse Gehäuse im latenten Raum angeordnet zu sein. Gemäß Abbildung 5.5 verfügt diese Klasse über die längsten Sequenzen und damit auch die meisten Zwischenzustände.

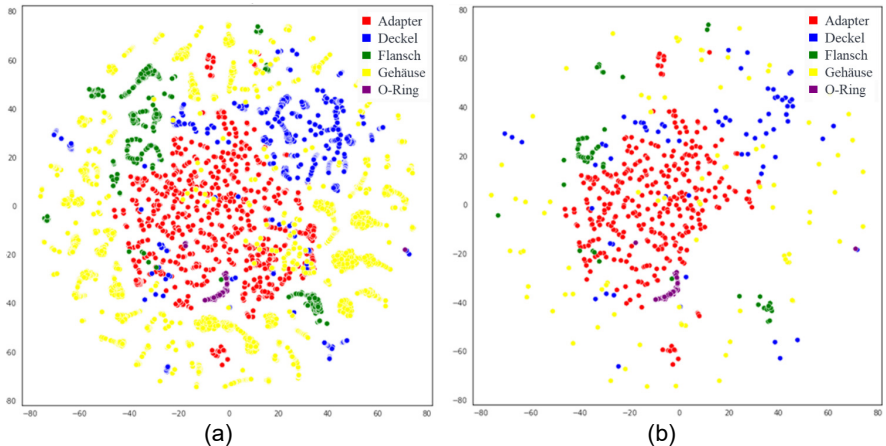
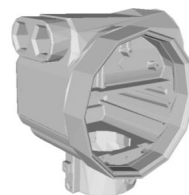


Abbildung 5.21: t-SNE Visualisierung der latenten Vektoren der fünf Bauteilklassen inklusive aller Zwischenzustände (a) und ohne Zwischenzustände (b)

In Abbildung 5.22 ist beispielhaft der erste und finale Zustand eines Gehäuses abgebildet. Das dargestellte Gehäuse besteht aus 226 Konstruktionsschritten und damit Zwischenzuständen. Zum Vergleich ist in Abbildung 5.21 (b) die t-SNE Visualisierung der latenten Vektoren von lediglich finalen Bauteilzuständen dargestellt. Hier wird ersichtlich, dass durch die stark variierenden Geometrien der Zwischenzustände scheinbar die „Lücken“ im latenten Raum gefüllt werden und somit geometrische Übergänge zwischen den Klassen geschaffen werden.



(a)



(b)

Abbildung 5.22: Erster (a) und letzter (b) Zwischenzustand eines beispielhaften Gehäuses mit 226 Konstruktionsschritten

Für das Clustern und die Klassifikation wurden zur besseren Vergleichbarkeit zu den Ergebnissen aus Kapitel 5.3.1.2 zwei Varianten durchgeführt, die sich in den zugrunde gelegten Daten unterscheiden. In der ersten Variante wurden beide Verfahren auf dem

kompletten Datensatz und somit auf allen Zwischenzuständen ausgeführt. Die erhaltene Clusterlösung ist in Abbildung 5.23 (a) dargestellt.

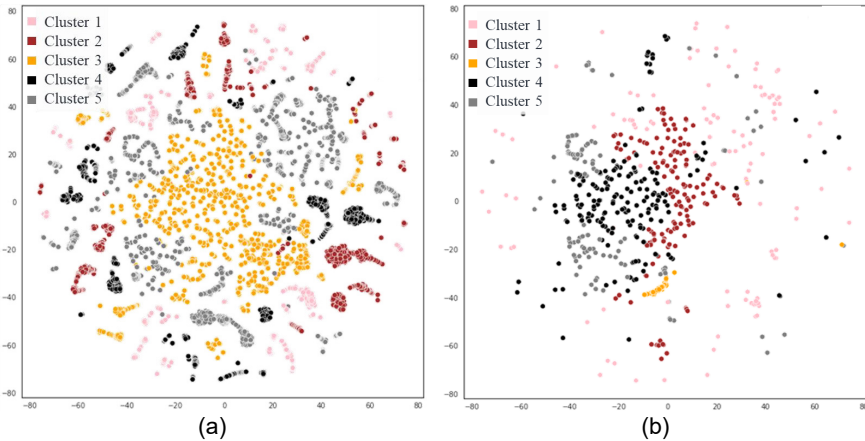


Abbildung 5.23: *t-SNE Visualisierung der über k -Means gefundenen Cluster für die latenten Vektoren aller Zwischenzustände (a) und ohne Zwischenzustände (b)*

Hier wird deutlich, dass die scheinbar sehr gestreute Darstellung in Abbildung 5.21 (a) zumindest in Teilen auf die t-SNE Visualisierung zurückzuführen ist, da auch die gefundenen Cluster in Abbildung 5.23 (a) in der Visualisierung im Raum verteilt dargestellt werden. Die Clusterbildung mit k -Means erzielt hier einen HS von knapp 44 % (im Vergleich zu 46 % in Kapitel 5.3.1.2), einen CS von knapp 37 % (im Vergleich zu 39 % in Kapitel 5.3.1.2) und einen V-Score von 40 % (im Vergleich zu 43 % in Kapitel 5.3.1.2). Die Klassifikationsgenauigkeit kann dagegen auf 98,35 % (im Vergleich zu 90 % in Kapitel 5.3.1.2) gesteigert werden (siehe Anhang O.4). In der zweiten Variante werden Clusteranalyse und Klassifikation nur mit den finalen Objekten durchgeführt, wie es auch in Kapitel 5.3.1.2 der Fall war. In Abbildung 5.23 (b) wird die gefundene Clusterlösung dargestellt. Es wird ein HS von 50,26%, ein CS von 41,50 % und ein V-Score von 45,46 % erreicht. Alle Scores der Clusteranalyse können folglich im Vergleich zu Kapitel 5.3.1.2 gesteigert werden. Die Genauigkeit der Klassifikation liegt in dieser Variante bei 95,35 % (siehe Anhang O.5). Im Vergleich zur ersten Variante ist jedoch anzumerken, dass durch die Einschränkung auf nur finale Objekte der Testdatensatz deutlich reduziert wird. Lediglich ein Objekt wird falsch klassifiziert. Insgesamt lassen sich die quantitativen Verbesserungen im Vergleich zu Kapitel 5.3.1 auf die deutlich vergrößerte Datenbasis zurückführen. Ein qualitativer Vergleich der Rekonstruktionen

der latenten Vektoren stützt diese Vermutung. Abbildung 5.24 zeigt den Vergleich der Rekonstruktionen von exemplarischen Komponenten aus dem Testdatensatz für die *Autoencoder*-Architektur aus Kapitel 5.3.1.2, trainiert auf nur finalen Zuständen, und für die in diesem Kapitel beschriebene Architektur, die auch auf Zwischenzuständen trainiert wurde. Dabei entspricht der in der oberen Zeile dargestellte Flansch einem finalen Bauteilzustand, der in der unteren Zeile gezeigte Adapter dagegen einem Zwischenzustand. Details, wie beispielsweise die spezielle äußere Form des Flansches sind in der Rekonstruktion der Architektur, die auf allen Zwischenzuständen trainiert wurde, deutlich besser repräsentiert. Außerdem wird deutlich, dass der *Autoencoder* aus Kapitel 5.3.1.2, obwohl er nur auf finalen Zuständen trainiert wurde, zwar auch unbekannte Zwischenzustände verarbeiten kann, jedoch sind die Rekonstruktionen im Vergleich zum *Autoencoder*, der auf allen Zwischenzuständen trainiert wurde, deutlich schlechter. Durch die größere Datenbasis können offensichtlich mehr Details erlernt werden.

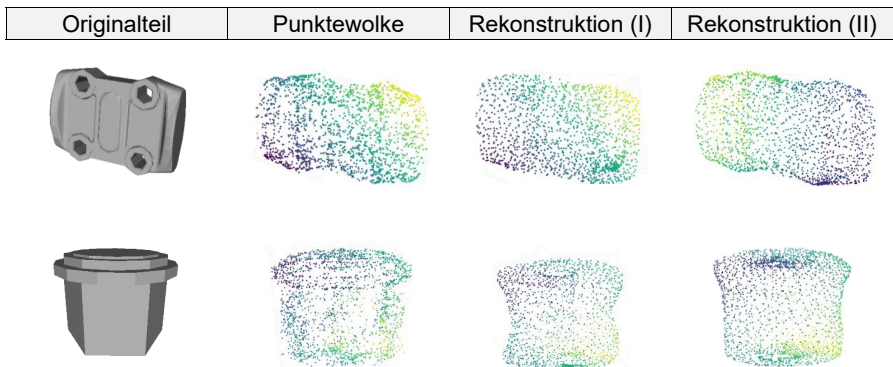


Abbildung 5.24: Vergleich der Rekonstruktionen des *Autoencoders* trainiert auf nur finalen Komponenten (Rekonstruktion (I)) und des *Autoencoders* trainiert auf allen Zwischenzuständen (Rekonstruktion (II))

Zur weiteren Untersuchung der erlernten Repräsentationen kann zudem die Interpolation betrachtet werden (siehe Kapitel 4.3.3). Hierfür wird zwischen zwei latenten Vektoren linear interpoliert und die entsprechenden Vektoren mittels des *Decoders* in eine Punktewolke überführt. Nachfolgend werden die Untersuchungen basierend auf A_Bräuner (A_2022) dargestellt. In Abbildung 5.25 sind die Interpolationen zwischen einem anfänglichen Zustand und dem Endzustand für ein beispielhaftes Objekt eines

Adapters dargestellt. Es wird deutlich, dass der Adapter nach und nach mehr Eigenschaften des O-Rings annimmt, was die Repräsentativität des latenten Raumes bestärkt. Weitere Interpolationen sowie eine exemplarische algebraische Operation auf latenten Vektoren sind in Anhang P zu finden.

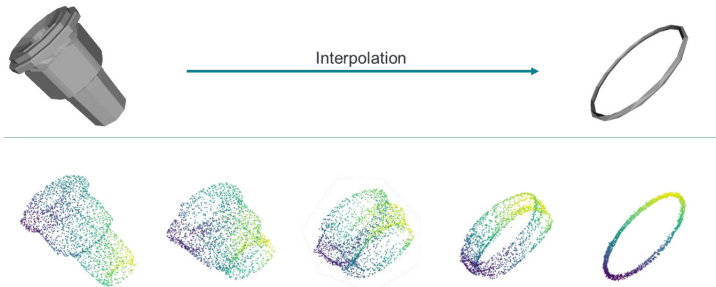


Abbildung 5.25: Beispielhafte Interpolation zwischen einem Adapter und einem O-Ring (A_Bräuner 2022)

5.4 Verwendung des Wissens zur Konstruktionsunterstützung

Im Rahmen dieses Kapitels wird das zuvor aufbereitete und extrahierte Wissen herangezogen, um die Ähnlichkeitssuche (Kapitel 5.4.1), das Vorschlagen nächster Konstruktionsschritte (Kapitel 5.4.2) sowie das Zusammenspiel beider Funktionalitäten (Kapitel 5.4.3) zu erproben. Um den in Kapitel 4.1.4 genannten Vorteil der Nutzung der sequentiellen Daten aus dem Konstruktionsvorgehen im Vergleich zu den rein finalen Bauteilzuständen für die Ähnlichkeitssuche aufzuzeigen, werden zwei verschiedene *Autoencoder*-Modelle zur Erzeugung der latenten Repräsentationen der 3D-Geometrien betrachtet. Die je *Autoencoder*-Modell unterschiedlich erlernten latenten Repräsentationen sind für die Bestimmung der Menge der geometrisch ähnlichsten Modelle ausschlaggebend. Um folglich die jeweils zur Darstellung der 3D-Geometrie erlernten Repräsentationen zu vergleichen, ist die rein geometrische Ähnlichkeitssuche (Umsetzungsstufe 1) ausreichend. Das erste *Autoencoder*-Modell (nachfolgend mit „AE_Final“ bezeichnet) wurde gemäß Kapitel 5.3.1.2 auf den finalen Komponenten des Datensatzes „Finale_Komponenten_SO“ trainiert. Das zweite *Autoencoder*-Modell (nachfolgend „AE_ZW“ genannt) dagegen wurde auf allen finalen Komponenten sowie den zugehörigen Zwischenzuständen gemäß Kapitel 5.3.2 trainiert. Der dafür genutzte Datensatz

„Gesamt_SO“ enthält alle notwendigen Informationen, um sowohl die Ähnlichkeitssuche als auch das Vorschlagen von Konstruktionsschritten in allen Umsetzungsstufen zu implementieren.

5.4.1 Ähnlichkeitssuche und Bewertung produktionsrelevanter Produkteigenschaften

Die Umsetzungsstufe 1 der Ähnlichkeitssuche, die das Auffinden geometrisch ähnlicher Modelle umfasst, wird anhand der erzeugten Repräsentationen der zwei verschiedenen *Autoencoder*-Modelle „AE_Final“ und „AE_ZW“ erprobt. Für beide Fälle werden aus jeder der fünf Bauteilklassen zwei beispielhafte Vertreter ausgewählt. Da die Ähnlichkeitssuche insbesondere auch für halbfertige, sich in der Konstruktion befindliche Input-Modelle geeignet ist, werden von jeder Komponente verschiedene Zwischenzustände betrachtet. Hierzu werden ein anfänglicher Zustand, ein Zustand nach etwa der Hälfte an Schritten sowie der Endzustand herangezogen. In Anhang Q.1 ist eine Übersicht über die untersuchten Komponenten mit den jeweiligen Zwischenzuständen gegeben. Die gewählten Komponenten stammen aus den für den *Autoencoder* unbekanntesten Testdaten. Dadurch soll der reale Anwendungsfall nachgestellt werden, in dem eine neue Komponente gerade konstruiert wird. Den Lösungsraum für die Suche stellen dabei die finalen Komponenten der Trainingsdaten dar. Übertragen auf einen realen Anwendungsfall entsprechen diese allen vorhandenen Komponenten, die bereits bekannt sind und für das Training des *Autoencoders* genutzt werden können. Zum Vergleich der Resultate der Ähnlichkeitssuche wird eine Suche rein auf Metainformationen dargelegt. Hierzu werden die in Anhang E aufgezeigten 18 Attribute in Anlehnung an Machalica & Matyjewski (2019) genutzt.

5.4.1.1 Initialisierung der Ähnlichkeitssuche

Zum Auffinden ähnlicher Komponenten wird die zugrunde gelegte Datenbasis aller finalen Bauteilzustände zunächst mit Hilfe von *k-Means* geclustert. Da für den betrachteten Anwendungsfall die Anzahl der Klassen bekannt ist, wird $k=5$ gesetzt⁴². Die Clustering wird dabei jeweils für die latenten Vektoren des „AE_Final“ und „AE_ZW“ durchgeführt⁴³. Um für eine gegebene Input-Komponente i , repräsentiert über deren latenten

⁴² Der Wert $k=5$ wird auch durch den *Davies Bouldin Score* bestätigt, siehe Anhang Q.2.

⁴³ Es wurden jeweils 100 Durchläufe ausgeführt und der durchschnittliche HS, CS und V-Score berechnet. Anschließend wurde eine konkrete Clusterlösung mit Werten nahe des Durchschnitts herangezogen.

Vektor \vec{x}_i , die Menge der (geometrisch) ähnlichsten Modelle $S_{sim,i}$ zu finden, wird zunächst die Distanz zu den zuvor definierten fünf Clusterzentren z_k , ebenfalls dargestellt über deren latente Repräsentation $\vec{x}_{z,k}$, bestimmt und die Menge der relevanten Clusterzentren $S_{z,i}$ festgelegt. Die nachfolgend beschriebenen Ergebnisse basieren auf der von der Autorin angeleiteten Arbeit A_Bräuner (2022). Gemäß Kapitel 4.4.1.2 muss dafür zunächst anwendungsfallspezifisch festgelegt werden, wie viele Cluster betrachtet werden müssen, sodass möglichst alle ähnlichen Komponenten gefunden werden. Ein weiterer wesentlicher Parameter ist der Suchradius ε , der ebenfalls in Abhängigkeit des Anwendungsfalls gewählt werden muss. Der optimale Wert für ε wird dabei empirisch in Anlehnung an das Verfahren von Ester et al. (1996) bestimmt⁴⁴. Die Ergebnisse sind in Anhang Q.3 und Anhang Q.4 dargestellt. Für die Untersuchungen auf den latenten Vektoren von rein finalen Komponenten (Modell „AE_Final“) wird $\varepsilon = 0,15$, für die Untersuchungen auf finalen Komponenten und Zwischenzuständen (Modell „AE_ZW“) wird $\varepsilon = 0,3$ gesetzt⁴⁵. Zur Bestimmung der Anzahl der relevanten Cluster $|S_{z,i}|$ wurde die Ähnlichkeitssuche anhand exemplarischer Objekte zunächst ohne Clusterzentren durchgeführt. Für die gleichen Objekte erfolgte die Bestimmung der ähnlichsten Komponenten anschließend über die Clusterzentren. Dabei wurden so viele Clusterzentren sukzessive hinzugezogen, bis die Menge der ähnlichsten Modelle identisch war. Dies ist für $|S_{z,i}| = 2$ der Fall. Die Ergebnisse sind in Anhang Q.5 und Anhang Q.6 zu finden.

5.4.1.2 Auffinden ähnlicher Modelle und Bewertung der produktionsrelevanten Produkteigenschaften

Zur Bestimmung von $S_{sim,i}$ für die Komponente i wird zunächst die euklidische Distanz zu allen fünf Clusterzentren z_k berechnet und die Menge der relevanten Clusterzentren $S_{z,i}$ bestimmt. Darauf folgend wird die Distanz zwischen \vec{x}_i und allen Komponenten $\vec{x}_j \in S_{z_k}$ mit $\vec{x}_{z_k} \in S_{z,i}$ berechnet. Die Menge der ähnlichsten Modelle $S_{sim,i}$ umfasst schließlich diejenigen Komponenten, für die gilt $d_2(\vec{x}_i, \vec{x}_j) \leq \varepsilon$. Eine Übersicht über die Anzahl gefundener ähnlicher Modelle für die Untersuchungskomponenten ist in Anhang

⁴⁴ Für jeden Datenpunkt wird dabei die Distanz zu seinen r nächsten Nachbarn bestimmt. Die erhaltenen Distanzen werden anschließend in einem Distanz-Graphen absteigend sortiert aufgetragen. Der optimale Wert für ε kann anhand des Knicks der Kurve abgelesen werden.

⁴⁵ Die unterschiedlichen Werte der Untersuchungen resultieren aus den verschiedenen latenten Räumen und damit der Anordnung der latenten Vektoren, die erlernt wurden.

Q.5 und Anhang Q.6 zu finden. Die Ähnlichkeit Sim_{ij} zwischen der Komponente i und den ähnlichsten Komponenten j wird gemäß Kapitel 4.4.1.2 über die maximale Distanz zwischen zwei latenten Vektoren des betrachteten Datensatzes bestimmt.

In Abbildung 5.26 werden beispielhaft für die Input-Komponente i in Form eines Flansches die Ergebnisse der Ähnlichkeitssuche aufgezeigt. Dabei wird unterschieden, ob die Ähnlichkeitssuche auf den latenten Vektoren des Modells „AE_Final“ (in Abbildung 5.26 Zeile 1) oder den latenten Vektoren des Modells „AE_ZW“ (in Abbildung 5.26 Zeile 2) durchgeführt wurde. Für die drei unterschiedlichen Bauteilzustände der Komponente i sind in den drei ersten Spalten jeweils die drei ähnlichsten Modelle aus $S_{sim,i}$ dargestellt. In der vierten Spalte wird die unähnlichste Komponente j aus $S_{sim,i}$ aufgezeigt. Die darauf folgende Spalte enthält jeweils die Anzahl der Komponenten in $S_{sim,i}$. Um den Effekt des Suchradius ε aufzuzeigen, beinhaltet die letzte Spalte die Komponente mit der geringsten Distanz, die jedoch gerade außerhalb des Suchradius liegt. Anhand der Ergebnisse wird ersichtlich, dass für anfängliche Zustände die Menge der ähnlichsten Modelle $S_{sim,i}$ leer ist. Grund hierfür ist, dass sich die anfängliche Geometrie noch sehr stark von der finalen Geometrie unterscheidet und somit auch die Distanzen der latenten Vektoren noch groß sind. Die dennoch ähnlichsten Komponenten haben eine geringe Ähnlichkeit zum tatsächlichen finalen Zustand der Komponente i . Für den weiter fortgeschrittenen Zustand können für beide Untersuchungsfälle 17 ähnlichste Teile gefunden werden, die alle der Klasse Flansch angehören. Die Komponente mit der größten Ähnlichkeit stimmt dabei in beiden Fällen überein. Die Komponente, die gerade außerhalb des Suchradius ε liegt, ist zwar noch ein Flansch, dennoch sind die geometrischen Unterschiede sehr groß, was auch anhand der absoluten Differenz der Ähnlichkeit um über 10 % ersichtlich wird⁴⁶. Die Ausgrenzung dieses Teiles aus $S_{sim,i}$ erscheint daher plausibel.

⁴⁶ Die Ähnlichkeit kann zwar als Anhaltspunkt betrachtet werden, dennoch ist sie stark abhängig von der zugrundeliegenden Datenbasis, da sie auf der maximalen Distanz innerhalb dieser basiert und sollte deshalb nicht als quantitatives Maß genutzt werden.



















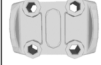
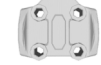


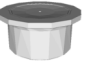

		Menge der ähnlichsten Modelle $S_{sim,i}$ des Bauteilzustandes der Komponenten i				Ähnlichstes Modell außerhalb von $S_{sim,i}$						
		Top 3 der ähnlichsten Modelle aus $S_{sim,i}$			Unähnlichstes Modell aus $S_{sim,i}$	$ S_{sim,i} $	$\min \{ \bar{x}_i^j d_{\mathcal{L}}(\bar{x}_i, \bar{x}_i^j) > \epsilon \}$					
Bauteilzustand Komponente i		Latente Vektoren AE_Final	X				0	 Flange_230035164 (79,04%)				
		Latente Vektoren AE_ZW					0	 Cover_230005932 (64,26%)				
		Latente Vektoren AE_Final					Flange_230005963 (95,13 %)	Flange_230037523 (94,72 %)	Flange_230041038 (94,48 %)	Flange_230033736 (88,58 %)	17	 Flange_230045640 (78,28%)
		Latente Vektoren AE_ZW					Flange_230005963 (96,24 %)	Flange_230036423 (89,14 %)	Flange_230036438 (87,16 %)	Flange_230038315 (81,22 %)	17	 Flange_230045640 (69,73%)
		Latente Vektoren AE_Final					Flange_230036423 (97,26 %)	Flange_230005963 (96,42 %)	Flange_230005964 (93,51 %)	Flange_230033736 (88,65 %)	17	 Flange_230045640 (78,28%)
		Latente Vektoren AE_ZW					Flange_230036423 (95,57 %)	Flange_230005963 (93,03 %)	Flange_230033728 (89,03 %)	Flange_230038315 (82,31 %)	17	 Flange_230045640 (71,18%)

Abbildung 5.26: Ergebnisse der Ähnlichkeitssuche für drei Bauteilzustände von Flansch 230036437 für die latenten Vektoren des Modells „AE_Final“ und „AE_ZW“. Identische Modelle je Bauteilzustand sind in der gleichen Farbe hinterlegt.

Bei der Suche nach ähnlichen Komponenten für den finalen Zustand von i ist $S_{sim,i}$ identisch, lediglich die Reihenfolge ändert sich. Die qualitative Bewertung dieses Beispiels zeigt, dass die Ähnlichkeitssuche für beide Untersuchungsfälle gleichermaßen gut zu funktionieren scheint⁴⁷.

⁴⁷ Die Ähnlichkeitssuche ohne Clusterzentren liefert ebenfalls in beiden Fällen die gleichen 17 Komponenten. Allerdings wird zum Auffinden der Komponenten knapp die dreifache Rechenzeit benötigt.

Als Vergleich sind die Ergebnisse der Ähnlichkeitssuche entsprechend dem Ansatz von Machalica & Matyjewski (2019) rein über Metainformationen (siehe Anhang E) in Abbildung 5.27 für die unterschiedlichen Zustände der Komponente i dargestellt.

		Modelle von verschiedenen Positionen der ähnlichsten Modelle					
		Top 3 der ähnlichsten Modelle			Position 17	Position 18	
Bauteilzustand Komponente i		Adapter_230006165 (96,13 %)	Adapter_230035564 (95,93 %)	Adapter_230006161 (95,81 %)	X		
							
		Flange_230035164 (95,49 %)	Adapter_230043870 (94,81 %)	Adapter_230006115 (95,79 %)	Adapter_230006163 (92,99 %)	Adapter_230006160 (92,85 %)	
							
		Flange_230034846 (98,31 %)	Flange_230036438 (96,98 %)	Flange_230033728 (94,86 %)	Adapter_230035564 (92,71 %)	Adapter_230006165 (92,58 %)	
							


 Übereinstimmung innerhalb der Top 3 Ergebnisse mit der Suche auf Basis der latenten Vektoren

Abbildung 5.27: Ergebnisse der Ähnlichkeitssuche für drei Bauteilzustände von Flansch 230036437 auf Metainformationen

Dabei werden für den mittleren und finalen Zustand die 18 ähnlichsten Komponenten betrachtet, um die Ergebnisse mit denen aus Abbildung 5.26 vergleichen zu können. Entsprechend sind die drei ähnlichsten Komponenten sowie die Komponenten an Position 17 und 18 abgebildet. Insbesondere für den anfänglichen und mittleren Zustand der Komponente i zeigen die gefundenen finalen Komponenten wenig geometrische Ähnlichkeit auf. Für den finalen Zustand wird als ähnlichstes Teil ein Flansch mit einer Lasche an der Oberkante ausgegeben. Die tatsächliche geometrische Gestalt wird über die Metainformationen offensichtlich weniger gut abgebildet. Außerdem besteht die Menge der 17 ähnlichsten Teile aus 10 Flanschen und 7 Adaptern. Die Adapter (in Abbildung 5.27 die Komponente an Position 17) haben dabei eine stark abweichende geometrische Form.

Der Effekt des Suchradius ε kann außerdem anhand eines weiteren, in Anhang Q.7 gezeigten Flansches verdeutlicht werden. Für den anfänglichen Bauteilzustand in Form

eines Quaders ist die Menge der ähnlichsten Komponenten für die Suche auf den latenten Vektoren des Modells „AE_ZW“ leer, für die latenten Vektoren des Modells „AE_Final“ können drei ähnliche Endzustände gefunden werden. Diese sind zwar geometrisch ähnlich zum Input, jedoch werden aufgrund des sehr anfänglichen Zustandes noch keine Komponenten gefunden, die dem tatsächlichen finalen Zustand ähnlich sind. Für den mittleren Zustand dagegen werden für beide Varianten die gleichen drei ähnlichsten Flansche gefunden. Die Komponenten außerhalb des Suchradius ε weisen im Vergleich zum Input größere geometrische Unterschiede auf. Für die Suche auf Basis des finalen Zustandes ändert sich lediglich die Reihenfolge der gefundenen ähnlichen Flansche. Im Vergleich dazu kann gemäß Anhang Q.7 für den mittleren Zustand via Metainformationensuche noch keiner der tatsächlich ähnlichen Flansche gefunden werden. Die drei ähnlichsten Komponenten sind runde Adapter, erst an vierter Stelle wird ein Flansch gefunden, der jedoch ebenfalls rund ist. Für den finalen Zustand wird lediglich einer der drei ähnlichen Flansche gefunden. Wie im Beispiel zuvor wird deutlich, dass die Suche auf den latenten Vektoren bereits auf den mittleren Bauteilzuständen im Vergleich zu den Metainformationen gute Ergebnisse liefert. Ein ähnliches Bild zeigt sich für die weiteren Untersuchungsteile. Die entsprechenden Ergebnisse sind in Anhang Q.8 bis Anhang Q.12 angefügt.

Eine Schwäche des Ansatzes gegenüber fehlerhafter Datenvorbereitung wird anhand der Ergebnisse für das in Anhang Q.8 betrachtete Gehäuse deutlich. Hier werden mit beiden Varianten der latenten Vektoren für den mittleren und finalen Zustand der Input-Komponente zwar ähnliche Gehäuse innerhalb von ε gefunden, jedoch wird anhand der Suche auf dem finalen Zustand für das Modell „AE_Final“ deutlich, dass es offensichtlich noch ein weiteres ähnliches Modell (Housing_230034945) im Datensatz gibt. Dies wird auch über die Suche auf Metainformationen bestätigt. Hier werden allerdings tatsächlich ähnliche Gehäuse erst mit dem finalen Zustand als Sucheingabe gefunden. Der Grund für die nicht ausreichende Ähnlichkeit zum Such-Input ist, dass die Standardausrichtung von Housing_230034945 fälschlicherweise um 180 Grad gedreht ist, weshalb sich die Enkodierung dieser Komponenten im latenten Vektor unterscheidet. Zur Veranschaulichung sind die Rekonstruktionen dieser Komponenten mit beiden *Autoencoder*-Varianten in Abbildung 5.28 dargestellt. Hier wird ersichtlich, dass die Rekonstruktion beider Varianten um 180 Grad verdreht ist. Außerdem wird hier nochmals der Unterschied beider *Autoencoder*-Varianten deutlich. Durch das Training auf dem Datensatz inklusive aller Zwischenstände ist die Rekonstruktion erkennbar detaillierter.

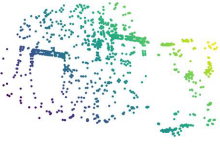

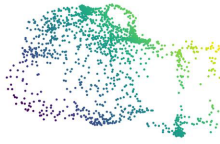
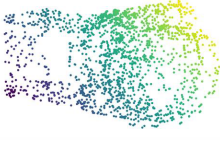


Housing_230044255		
Original	Rekonstruktion „AE_Final“	Rekonstruktion „AE_ZW“
		
Housing_230034945		
Original	Rekonstruktion „AE_Final“	Rekonstruktion „AE_ZW“
		

Abbildung 5.28: Vergleich der Rekonstruktionen von „AE_Final“ und „AE_ZW“ für zwei Gehäuse

Für einen Großteil der Untersuchungsteile gilt: je fortgeschrittener deren Bauteilzustand ist, umso mehr ähnliche Komponenten werden gefunden. Für anfängliche Zustände, die geometrisch noch kaum Ähnlichkeiten zum resultierenden Endzustand aufweisen, ist die Menge der ähnlichsten Modelle häufig leer. Zudem werden auch Unterschiede zwischen den einzelnen Klassen deutlich. Insbesondere für O-Ringe scheinen sehr viele ähnliche Geometrien gefunden zu werden. Grund hierfür ist, dass die geometrische Distanz zwischen zwei Objekten der O-Ring-Klasse im Vergleich zu den anderen Klassen deutlich geringer ist (siehe Anhang Q.13). Die größten Distanzen weist die Klasse der Gehäuse auf, hier werden auch die wenigsten ähnlichen Teile gefunden.

Eine weitere Sortierung der Suchergebnisse anhand der Metainformationen gemäß Umsetzungsstufe 2 der Ähnlichkeitssuche ist vor allem hilfreich, wenn viele ähnliche Modelle gefunden werden. Gemäß der Übersicht aus Anhang Q.5 bzw. Anhang Q.6 ist dies für die Klasse der O-Ringe der Fall. Die Ergebnisse der erweiterten Ähnlichkeitssuche auf Basis der latenten Vektoren des Modells „AE_ZW“ sind in Abbildung 5.29 für einen O-Ring dargestellt. Neben der geometrischen Ähnlichkeit sim_{ij} ist zudem die Ähnlichkeit bezüglich Abmessungen, Material, Volumen sowie die gewichtete Gesamtähnlichkeit angegeben.

Input-Komponente:  **O-Ring 231033780**
 Material: Kunststoff
 Abmessungen: 86,03 x 83,87 x 3,53 [mm]
 Volumen: 2536,55 [mm³]













	Gewichtung	Top 3 der ähnlichsten Modelle			Unähnlichstes Modell aus $S_{sim,i}$	$S_{sim,i}$
		O-Ring 231031089	O-Ring 231028774	O-Ring 230006766	O-Ring 230006863	
Umsetzungsstufe 1 $\alpha_{geo} = 1$ $\alpha_{mat} = 0$ $\alpha_{abm} = 0$ $\alpha_{vol} = 0$		Geometrie: 98,56% Material: Kunststoff 100% Abmessung: 96 x 93 x 3,5 98,8899% Volumen: 2829,56 99%	Geometrie: 98,53% Material: Stahl 0% Abmessung: 28 x 27 x 1 91,55% Volumen: 66,62 99,97%	Geometrie: 98,47% Material: Stahl 0% Abmessung: 19 x 19 x 0,7 90,27% Volumen: 30,14 99,97%	Geometrie: 82,52% Material: Stahl 0% Abmessung: 16 x 16 x 1,7 90,01% Volumen: 141,63 99,97%	112
						
		Geometrie: 98,47% Material: Kunststoff 100% Abmessung: 87 x 87 x 3,6 99,70% Volumen: 2390,50 99,99%	Geometrie: 97,07% Material: Kunststoff 100% Abmessung: 87 x 86 x 2,8 99,64% Volumen: 1318,18 99,98%	Geometrie: 98,56% Material: Kunststoff 100% Abmessung: 96 x 93 x 3,5 98,68% Volumen: 2829,56 99,99%	Geometrie: 82,83% Material: Stahl 0% Abmessung: 15 x 15 x 1,2 89,83% Volumen: 93,15 99,97%	
						
Umsetzungsstufe 2 $\alpha_{geo} = 0,2$ $\alpha_{mat} = 0,2$ $\alpha_{abm} = 0,5$ $\alpha_{vol} = 0,1$		Gewichtete Ähnlichkeit: 99,54%	Gewichtete Ähnlichkeit: 99,23%	Gewichtete Ähnlichkeit: 99,05%	Gewichtete Ähnlichkeit: 71,48%	
						

Abbildung 5.29: Ergebnisse der Ähnlichkeitssuche auf den latenten Vektoren von „AE_ZW“ über Umsetzungsstufe 1 und Umsetzungsstufe 2

Die 112 ähnlichsten Komponenten umfassen ausschließlich O-Ringe. Im Vergleich dazu befinden sich bei Suche über Metainformationen unter den 112 ähnlichsten Komponenten 55 klassenfremde⁴⁸. Die obere Zeile beschreibt die drei ähnlichsten Komponenten sowie die unähnlichste Komponente aus $S_{sim,i}$ ohne Berücksichtigung weiterer Metainformationen. Dabei fällt auf, dass insbesondere die Größe der O-Ringe für die geometrische Ähnlichkeit aufgrund der notwendigen Skalierung (siehe Kapitel 5.2) nicht ausschlaggebend ist⁴⁹. Unter den ähnlichsten Modellen finden sich folglich O-Ringe mit einem deutlich geringeren Durchmesser. Ebenso stimmt das Material nicht überein. Durch die zusätzliche Berücksichtigung von Abmessungen, Material und Volumen in der Bestimmung der Ähnlichkeit werden die in Abbildung 5.29 in der unteren Reihe dargestellten ähnlichsten O-Ringe vorgeschlagen. Dabei wird die Gewichtung für Geometrie und Material $\alpha_{geo} = \alpha_{mat} = 0,2$ gesetzt, für die Gewichtung der Abmessungen gilt $\alpha_{abm} = 0,5$, das Volumen wird mit $\alpha_{vol} = 0,1$ berücksichtigt. Hier wird deutlich, dass

⁴⁸ Die 112 ähnlichsten Teile umfassen zwei Flansche, neun Gehäuse, elf Deckel und 33 Adapter.

⁴⁹ Die dargestellten relativen Größenunterschiede sind in Abbildung 5.29 annähernd maßstabsgetreu dargestellt.

nun die ähnlichsten Komponenten auch in den Abmessungen sowie im Material übereinstimmen. Das unähnlichste Modell dagegen ist deutlich kleiner und aus einem anderen Material. Das Volumen ist aufgrund der größeren Dicke des O-Rings zwar ähnlich, allerdings hat es aufgrund der geringen Gewichtung nur wenig Einfluss.

Weitere Ergebnisse der um Metainformationen erweiterten Ähnlichkeitssuche sind in Anhang Q.14 und Anhang Q.15 dargestellt.

Zur Bewertung der produktionsrelevanten Produkteigenschaften in Umsetzungsstufe 3 der Ähnlichkeitssuche können nun anhand der gefundenen ähnlichen Modelle die entsprechenden Ausprägungen gemäß Kapitel 4.4.1.3 abgeglichen werden. Für den Anwendungsfall wurden hierfür gemäß Kapitel 5.1.2 die Abmessungen und das Material gewählt. Zur Bestimmung der Cluster wird der Schwellenwert ε_{DB} gemäß dem Verfahren nach Ester et al. (1996) auf 0,06 gesetzt (siehe Anhang Q.17). In Abbildung 5.30 sind für einen O-Ring die erhaltenen Bereiche der produktionsrelevanten Produkteigenschaften PP_h für die gefundenen Cluster $h=1, \dots, 5$ dargestellt⁵⁰. Um die Bereiche dreidimensional visualisieren zu können, sind dabei nur die Abmessungen in x-, y- und z-Richtung aufgetragen. Das Material wird über die Form der Symbole differenziert. Das Überlappen der verschiedenen Bereiche PP_h ist der vereinfachten dreidimensionalen Visualisierung der eigentlich vierdimensionalen Datenpunkte geschuldet. Zudem werden über die Symbole nur die Materialklassen Kunststoff und Stahl unterschieden, innerhalb derer jedoch weitere Differenzierungen über die Dichte vorliegen. Der O-Ring selbst (rotes „x“) wird dem Cluster $h=1$ (rot) als Kernpunkt zugeordnet und dementsprechend anhand des Bereichs PP_1 bewertet. Die Ergebnisse der Bewertung sind in Abbildung 5.31 aufgeführt. Sowohl Abmessungen als auch Material⁵¹ stimmen mit den entsprechenden Parameterausprägungen von PP_1 überein, was über die grüne Ampel in der Spalte „Bewertung“ dargestellt wird. Da der O-Ring zudem ein Kernpunkt des Clusters $h=1$ darstellt, wird auch die Gesamtbewertung mit einer grünen Ampel beurteilt. Es existieren folglich ausreichend Komponenten mit ähnlicher Geometrie und ähnlichen produktionsrelevanten Produkteigenschaften, weshalb die Produzierbarkeit auf den vorhandenen Fertigungsmitteln voraussichtlich gegeben ist.

⁵⁰ Für die Clusterung wird ein *Silhouette Score* von 0,77 erreicht. Die große Anzahl an Cluster ist auf die verschiedenen Kunststoffarten, die über die Dichten hinterlegt sind, sowie die unterschiedlichen Größen zurückzuführen.

⁵¹ Beim Material wird lediglich zwischen den übergeordneten Klassen Kunststoff und Stahl differenziert, denen verschiedene Dichten zugeordnet sind (siehe Kapitel 5.1.2)

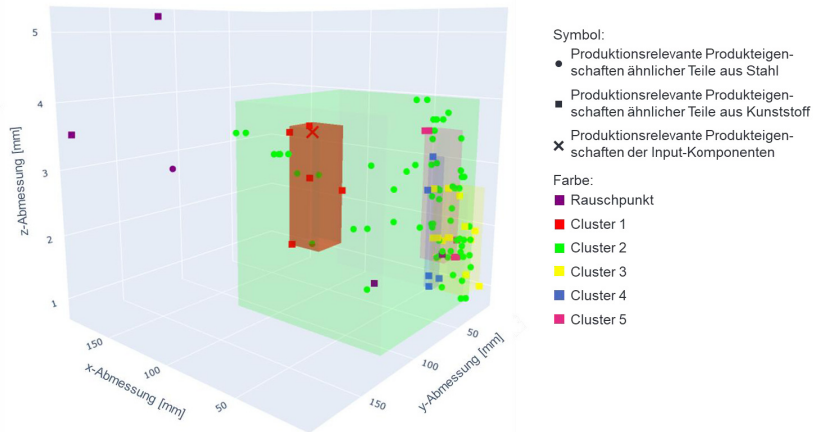


Abbildung 5.30: Bereiche produktionsrelevanter Produkteigenschaften auf Basis der fünf gefundenen Cluster für O-Ring 231033780

Eigenschaft	Bewertung PP_1				Bewertung
	Einheit	Wert	Minimum	Maximum	
Abmessung x-Richtung	mm	86,03	73,19	95,56	
Abmessung y-Richtung	mm	83,87	73,19	95,66	
Abmessung z-Richtung	mm	3,53	1,78	3,62	
Material	-	Kunststoff	Kunststoff		
Gesamtbewertung					

Abbildung 5.31: Bewertung der produktionsrelevanten Produkteigenschaften anhand des relevanten Bereichs PP_1 für O-Ring 231033780

Die Visualisierung der Bereiche produktionsrelevanter Produkteigenschaften für den siebten Zwischenzustand eines Adapters (siehe Anhang Q.1 mittlerer Zwischenzustand) ist in Abbildung 5.32 dargestellt⁵². Ein Großteil der Komponenten j aus der Menge der ähnlichsten Modelle wird auf Basis der produktionsrelevanten Produkteigenschaften in das Cluster $h=1$ (rot) eingeteilt, durch das der rot markierte Bereich PP_1 aufgespannt wird. Zwei weitere Komponenten werden in das grüne Cluster $h=2$ eingeteilt. Dabei stimmen diese in ihren produktionsrelevanten Produkteigenschaften überein, weshalb PP_2 lediglich einen Punkt im Raum darstellt. Zwei weitere Komponenten

⁵² Für die Clustering wird ein *Silhouette Score* von 0,56 erreicht.

werden als Rauschpunkt markiert (violett)⁵³. Der Adapter selbst (violette „x“) scheint zwar innerhalb von PP_1 zu liegen, wird jedoch dennoch als Rauschpunkt (violett) markiert, weshalb alle Bereiche überprüft werden. Betrachtet man die in Abbildung 5.33 visualisierte Bewertung für PP_1 , wird klar, dass das für den Adapter hinterlegte Material ausschlaggebend für die Markierung als Rauschpunkt ist. Die gewählten Abmessungen werden anhand der gefundenen ähnlichen Adapter zwar als unkritisch bewertet, unter den ähnlichsten Modellen kann jedoch kein Adapter gefunden werden, der ebenfalls aus Kunststoff hergestellt wurde. Die resultierende Bewertung der Eigenschaft Material sowie der Gesamtbewertung aufgrund der Deklaration als Rauschpunkt mit einer roten Ampel liefert folglich für den betrachteten Zwischenzustand des Adapters einen Hinweis auf mögliche Nicht-Produzierbarkeit mit den vorhandenen Fertigungsmitteln. Da es sich bei dem betrachteten Adapter tatsächlich um ein Zukaufteil handelt, weil in Eigenfertigung nur Dreh- und Frästeile aus Edelstahl gefertigt werden können, ist diese Einstufung plausibel⁵⁴. Die in Abbildung 5.33 visualisierte Bewertung von PP_2 zeigt bei keinem Parameter des Adapters eine Übereinstimmung mit dem üblichen Bereich, weshalb hier alle Ampeln auf Rot gesetzt werden.

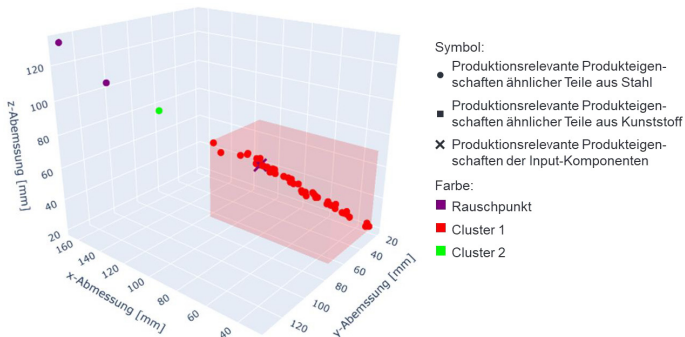


Abbildung 5.32: Bereiche produktionsrelevanter Produkteigenschaften auf Basis der zwei gefundenen Cluster für Adapter 230021093

⁵³ Beide Adapter haben einen Durchmesser größer 100 mm und können daher im Anwenderunternehmen nicht mehr in Eigenfertigung hergestellt, sondern müssen zugekauft werden.

⁵⁴ Der Anteil an Adaptern aus Kunststoff beträgt lediglich 0,8 %. Im gesamten Datensatz liegen lediglich 3 weitere Adapter aus Kunststoff vor, die jedoch geometrisch sehr unähnlich sind (Bilder siehe Anhang Q.16).



Abbildung 5.33: Bewertung der produktionsrelevanten Produkteigenschaften anhand aller gefundenen Bereiche PP_n für Adapter 230021093 (Zwischenzustand 7)

5.4.2 Sequenzanalyse zur Vorhersage nächster Konstruktionsschritte

Grundlage für die Sequenzanalyse zur Vorhersage nächster Konstruktionsschritte bilden die in Kapitel 5.2 erstellten Sequenzabschnitte. Je nach Umsetzungsstufe bestehen diese aus der Sequenz der latenten Vektoren (Umsetzungsstufe 1) oder noch zusätzlich aus den zugehörigen KEs (Umsetzungsstufe 2). Zur Veranschaulichung werden nachfolgend beide Umsetzungsstufen betrachtet. Aufgrund der sehr einfachen Geometrie der O-Ringe und der damit einhergehenden sehr kurzen Konstruktionssequenzen werden diese in der Untersuchung nicht mehr mit eingezogen⁵⁵. Folglich werden nur die Klassen Adapter, Deckel, Flansch und Gehäuse aus der zugrunde gelegten Datenbasis „Gesamt_SO“ berücksichtigt.

In Kapitel 5.4.2.1 werden zunächst die allgemeine Architektur des RNNs sowie die im Rahmen einer *Grid Search* untersuchten Konfigurationen der Hyperparameter erläutert. Die betrachteten Hyperparameterkonfigurationen werden anschließend in Kapitel 5.4.2.2 zunächst anhand einer quantitativen Bewertung verglichen und die optimale

⁵⁵ Gemäß Kapitel 5.2 und Anhang G.2 verfügt ein Großteil der O-Ringe lediglich über einen Konstruktionsschritt.

Konfiguration ausgewählt. Basierend auf dieser optimalen Parametrisierung erfolgt in Kapitel 5.4.2.3 eine qualitative Bewertung der Ergebnisse.

5.4.2.1 Architektur des Recurrent Neural Networks

Der grundsätzliche Aufbau des RNNs entspricht der in Kapitel 4.4.2.2 beschriebenen Architektur. Dabei werden zwei Varianten an Input-Vektoren betrachtet, die je nach Datenverfügbarkeit und damit Umsetzungsstufe des Assistenzsystems genutzt werden können. Dienen gemäß Umsetzungsstufe 1 lediglich die latenten Vektoren als Input, so erhält das RNN entsprechend deren Länge 512 Eingabewerte⁵⁶. Wird zusätzlich noch das KE je Zwischenzustand übergeben, das gemäß Kapitel 5.2 in ein *One-Hot-Encoding* überführt wurde, so ergeben sich entsprechend der Anzahl verschiedener KEs 14 weitere Eingaben. Da die Ausgabe des RNNs den nächsten Konstruktionszustand in Form eines latenten Vektors umfasst, ist die Output-Schicht aus 512 Neuronen aufgebaut. Die Anzahl und Breite sowie die Art der zwischen Input und Output liegenden RNN-Schichten sind Hyperparameter, die anwendungsfallsspezifisch gewählt werden müssen. Die Länge der Sequenzabschnitte und damit die Anzahl der latenten Vektoren, die durch das RNN verarbeitet werden, ist ein weiterer Hyperparameter. Um die optimalen Werte für diese Hyperparameter zu bestimmen, werden die in Tabelle 5-1 aufgeführten verschiedenen Ausprägungen getestet⁵⁷. Dafür werden alle Kombinationen für die zwei Varianten des Input-Vektors im Rahmen einer *k-fold Cross Validation* überprüft (siehe Kapitel 2.2.1)⁵⁸. Die Netztiefe, die Lernrate sowie die Anzahl an Epochen werden für alle Konfigurationen mit den gleichen Werten belegt. Die Länge der Sequenzabschnitte wird gemäß Kapitel 5.2 zwischen 4 und 8 variiert.

Tabelle 5-1: Betrachtete Hyperparameterkonfigurationen

Hyperparameter	Betrachtete Konfigurationen
# Schichten (Tiefe)	1
# Neuronen pro Schicht (Breite)	256, 512, 1024, 2048, 4096
Typ RNN-Zelle	Einfache RNN-Zelle, LSTM, GRU
# Epochen	2000
Lernrate λ	0,0005
Länge der Sequenzabschnitte Q	4, 8

⁵⁶ Die Länge der latenten Vektoren entspricht der Größe der gewählten *Bottleneck Size*, siehe Kapitel 5.3.2.

⁵⁷ Die gewählten Ausprägungen basieren dabei auf den Erkenntnissen aus der von der Autorin angeleiteten Arbeit A_Holtzward (2022)

⁵⁸ Hierfür wurden die Trainingsdaten herangezogen, die in 5 *Folds* aufgeteilt wurden.

5.4.2.2 Quantitative Betrachtung

Datenbasis für das Training sind die aus den Trainingsdaten erstellten Sequenzabschnitte. Bereits nach Erhalt des ersten Elements aus der Input-Sequenz kann eine Vorhersage getätigt und der entsprechende Verlust in Form des MSE berechnet werden. Der Verlust basiert dabei auf dem Vergleich zwischen Vorhersage und tatsächlich nächstem Bauteilzustand. Um eine optimale Konfiguration der in Tabelle 5-1 aufgezeigten Hyperparameter zu bestimmen, erfolgt ein quantitativer Vergleich anhand des im Rahmen der *k-fold Cross Validation* erzielbaren minimalen Verlustes. Der Testverlust⁵⁹ wird dabei als gleitender Mittelwert über jeweils 50 Epochen berechnet. Da die Länge der Sequenzabschnitte beeinflusst, wie viele Daten für das Training verwendet werden können (siehe Kapitel 5.2), ist ein Vergleich verschiedener Konfigurationen nur für gleiche Sequenzabschnittslängen aussagekräftig. Außerdem sind die Länge der Sequenzabschnitte sowie die Position eines Zwischenzustandes in der Gesamtsequenz entscheidend dafür, wie häufig ein Zwischenstand für die Fehlerberechnung einbezogen wird (siehe Anhang R).

Nachfolgend wird der Einfluss der verschiedenen Hyperparameter in Anlehnung an die von der Autorin angeleitete Arbeit A_Holtzwardt (2022) beschrieben. In Abbildung 5.34 ist der Einfluss von Breite, Zelltyp des RNNs sowie der Betrachtung zusätzlicher Metainformationen in Form des KEs (Umsetzungsstufe 2) auf den Testverlust für verschiedene Längen der Sequenzabschnitte dargestellt. Hier wird deutlich, dass, unabhängig von der Sequenzlänge, zwischen Schichtbreiten von 512 und 1024 Neuronen kaum Unterschiede vorliegen. Sowohl geringere als auch größere Breiten liefern dagegen einen höheren Testverlust. Bei Betrachtung der verschiedenen Zelltypen liefert jeweils die Architektur mit GRU-Zellen den geringsten Testverlust. Die einfache RNN-Zelle dagegen schneidet am schlechtesten ab. Darüber hinaus kann durch die Hinzunahme weiterer Metainformationen in Form der KEs (Umsetzungsstufe 2) ein geringerer Verlust erzielt werden.

Für die Untersuchung der verschiedenen Längen der Sequenzabschnitte wird eine einheitliche Konfiguration bezüglich der Hyperparameter Breite, Zelltyp und Metainformationen herangezogen und ein entsprechendes Modell auf dem gesamten Trainingsdatensatz trainiert. Dabei wird diejenige Konfiguration der optimal gefundenen ausge-

⁵⁹ Gemeint ist hiermit der Verlust, der jeweils auf dem *Fold*, das gerade als Testdatensatz dient, erzielt wird.

wählt, die für alle Sequenzlängen gemeinsam in der vorherigen Untersuchung den geringsten Testverlust erreicht. Eine Übersicht über die gewählte Parametrisierung des Modells ist in Anhang S.1 zu finden.

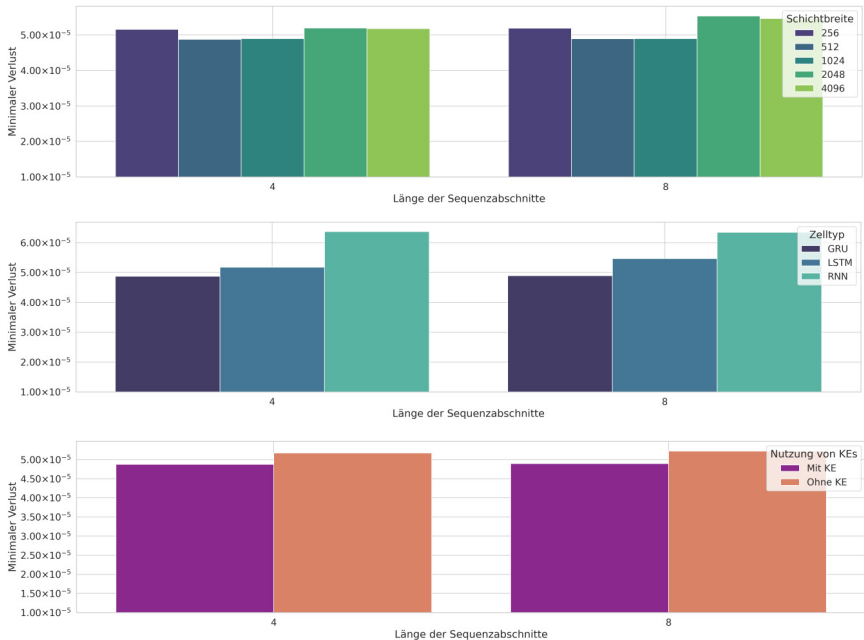


Abbildung 5.34: Minimaler Testverlust für verschiedene Sequenzlängen in Abhängigkeit unterschiedlicher Hyperparameterkonfigurationen

Um einen Vergleich des Verlustes auf verschiedenen Fenstergrößen durchführen zu können, wird nun von den Testdaten je Zwischenzustand⁶⁰ nur einmal der Verlust berechnet⁶¹. Bei Sequenzlängen, die kleiner als die Fenstergröße sind, werden für die Vorhersage entsprechend weniger Zwischenzustände genutzt. Die Ergebnisse dieser Untersuchung zeigt Abbildung 5.35. Dargestellt sind der durchschnittliche Testverlust je Sequenzabschnittlänge über alle Klassen sowie der je Bauteilkategorie durchschnittlich erzielte Testverlust. Für beide Sequenzabschnittslängen scheinen die erzielten Testverluste über alle Klassen sowie innerhalb der einzelnen Klassen nahezu identisch.

⁶⁰ Da für den ersten Zwischenzustand kein Input für das RNN existiert, ist hier keine Vorhersage möglich.

⁶¹ Bei den Untersuchungen zuvor fließt jeder Zwischenzustand so häufig in die Verlustberechnung ein, wie er in verschiedenen Sequenzabschnitten vorkommt.

Lediglich für die Klasse der Flansche und Adapter werden mit einer Sequenzabschnittslänge von 4 deutlich geringere Verluste erzielt. Grund hierfür sind die hohen Vorhergesageverluste für Bauteile, deren Gesamtsequenzlänge geringer als die Fenstergröße von 8 ist und die damit vom Training ausgeschlossen wurden. Gemäß Abbildung 5.5 sind das insbesondere runde Flansche und ein Teil der Adapter. Die vergleichsweise hohen durchschnittlichen Verluste für die Klasse Adapter lassen sich auf die deutlich größere Anzahl an unterschiedlichen Objekten und damit Varianz an Konstruktionsmustern im Datensatz erklären (siehe Abbildung 5.2). Insgesamt kann der geringste Verlust mit einer Sequenzlänge von 4 erzielt werden, weshalb diese für die im nachfolgenden Kapitel beschriebenen qualitativen Betrachtungen zugrunde gelegt wird.

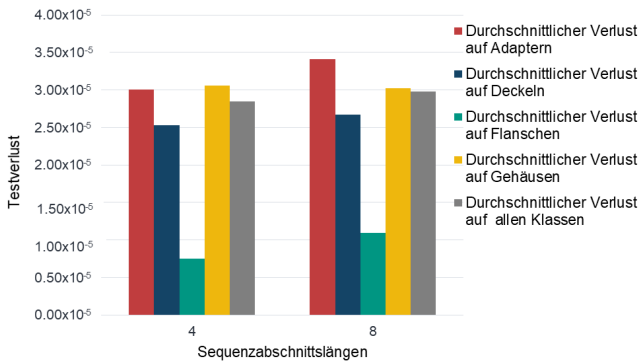


Abbildung 5.35: Durchschnittlicher Testverlust in Abhängigkeit der Sequenzlängen sowie der durchschnittliche Beitrag der einzelnen Klassen zu diesem Verlust

5.4.2.3 Qualitative Betrachtung

Die zuvor identifizierte optimale Parametrisierung wird im Rahmen dieses Kapitels für eine qualitative Bewertung herangezogen. Hierfür wird mittels des auf den optimalen Parametern trainierten RNN-Modells für jeden Zwischenzustand der Testdaten eine Vorhersage in Form eines latenten Vektors generiert, die schließlich über das genutzte *Autoencoder*-Modell „AE_ZW“ in eine Punktwolke rekonstruiert werden kann. Da insbesondere in den anfänglichen Konstruktionsschritten größere geometrische Veränderungen vorgenommen werden, wird in den nachfolgend gezeigten Beispielen der Fokus auf diese gelegt. In Abbildung 5.36 sind die ersten fünf Vorhersagen für einen Adapter visualisiert. Der tatsächliche Zwischenzustand wird einerseits über das CAD-Modell, andererseits über die daraus generierte Punktwolke sowie die durch den *Autoencoder*

rekonstruierte Punktwolke dargestellt. Die Vorhersage des Zustandes zum Zeitpunkt 2 basiert dabei lediglich auf dem Zustand zum Zeitpunkt 1, die Prädiktion von Zustand 3 auf Zeitpunkt 1 und 2⁶². Für alle weiteren Vorhersagen werden jeweils drei vorhergehende Zustände betrachtet.







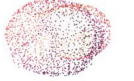












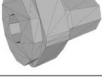



Zeitpunkt t	Tatsächlicher Zwischenzustand zum Zeitpunkt t			Vorhersage des Zwischenzustandes für Zeitpunkt t
	CAD-Modell	Generierte Punktwolke	Rekonstruierte Punktwolke	
1				
2				
3				
4				
5				
6				

Abbildung 5.36: Vorhersage des nächsten Zwischenzustandes für Adapter 230006082

Bereits nach dem ersten Input kann das RNN-Modell Zwischenzustand 2 vorhersagen. Hier wird eine größere geometrische Veränderung in Form einer Extrusion vorgenommen, die korrekt, wenn auch mit geringerer Extrusionstiefe⁶³, vorhergesagt wird. Kleinere Änderungen, wie Abrundungen des hexagonalen Körpers zwischen Zeitpunkt 2 und 3, können anhand der ursprünglichen Punktwolken nur unzureichend abgebildet und damit auch nicht erlernt werden. Die in Schritt 5 hinzugefügte Extrusion ist bereits

⁶² Auch wenn das RNN mit Q=4 trainiert und folglich dahingehend optimiert wurde, auf Basis der letzten drei Schritte den Folgezustand zu präzisieren, ist es dennoch möglich, auch für weniger Inputs Vorhersagen zu treffen.

⁶³ Grund ist, dass es auch eine Vielzahl kürzerer Adapter im Datensatz gibt, die aus dem ersten Konstruktions-schritt resultieren können, siehe Anhang S.2.

in der rekonstruierten Punktwolke nicht zu erkennen (siehe in Abbildung 5.36 an der linken Seite des Adapters).

Ein möglicher Grund hierfür ist, dass die für das Training des *Autoencoders* genutzte CD kleinere Unterschiede nicht ausreichend abbilden kann (Urbach, Ben-Shabat & Lindenbaum 2020), weshalb sie auch im latenten Vektor nicht enkodiert werden. Das RNN kann folglich die Veränderung zwischen Zeitpunkt 4 und 5 nicht erlernen. Die zum Zeitpunkt 6 hinzugefügte Bohrung, markiert in orange, wird korrekt präzidiert.

Eine beispielhafte Vorhersage für einen Seitenflansch ist in Abbildung 5.37 dargestellt. Typischerweise werden zunächst die beiden Flanken konstruiert, die anschließend zusammengesetzt werden. Diese Vorgehensweise wird korrekt erlernt. In den rekonstruierten Punktwolken zeigt sich jedoch eine Schwäche des genutzten *Autoencoders* bei der Verarbeitung von nicht zusammenhängenden Körpern. Diese können nicht vollständig voneinander separiert abgebildet werden.





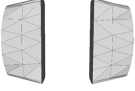


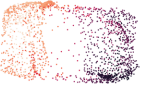
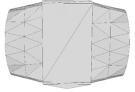
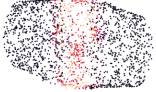
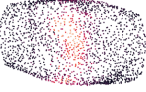
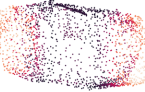
Zeitpunkt t	Tatsächlicher Zwischenzustand zum Zeitpunkt t			Vorhersage des Zwischenzustandes für Zeitpunkt t
	CAD-Modell	Generierte Punktwolke	Rekonstruierte Punktwolke	
1				
2				
3				

Abbildung 5.37: Vorhersage des nächsten Zwischenzustandes für Flansch 230036437

Abbildung 5.38 zeigt die Ergebnisse der Vorhersage für einen Deckel. Die größere geometrische Änderung in Form einer „Aushöhlung“ von Zeitpunkt 18 auf 19 kann durch das RNN nicht vorhergesagt werden. Grund hierfür ist, dass in den vorherigen Schritten nur marginale Änderungen vorgenommen wurden, die über die Punktwolke nicht abgebildet und damit auch nicht im latenten Vektor enkodiert werden. Für das RNN ist es somit nicht möglich, den genauen Zustand zu erkennen, um die Aushöhlung zum korrekten Zeitpunkt vorherzusagen.

Die Vorhersagen für ein exemplarisches Gehäuse sind in Anhang S.3 zu finden. Insgesamt lässt sich anhand der dargebrachten Beispiele zeigen, dass ein prinzipielles Erlernen von Konstruktionsmustern möglich ist. Aufgrund der Darstellung über Punktwolken lassen sich jedoch Details, wie Abrundungen von Kanten, nicht ausreichend abbilden, weshalb nur Vorhersagen von größeren geometrischen Veränderungen möglich sind.











Zeitpunkt t	Tatsächlicher Zwischenzustand zum Zeitpunkt t			Vorhersage des Zwischenzustandes für Zeitpunkt t
	CAD-Modell	Generierte Punktwolke	Rekonstruierte Punktwolke	
16				
17				
18				
19				

Abbildung 5.38: Vorhersage des nächsten Zwischenzustandes für Deckel 230039626

5.4.3 Integration in ein Gesamtsystem

Gemäß Kapitel 2.4.1 wurde gezeigt, dass insbesondere für anfängliche Bauteilzustände die Menge der ähnlichsten Modelle für eine Komponente i häufig leer ist. Durch die Nutzung des in Kapitel 2.4.2 trainierten RNNs zur Vorhersage des nächsten Bauteilzustandes der Komponenten i können jedoch passende ähnliche Modelle gefunden werden. In Abbildung 5.39 sind die Ergebnisse der Ähnlichkeitssuche für den ersten und zweiten Zwischenzustand eines Flanschs (siehe Spalte i) dargestellt. In Zeile 1 sind dabei die Ergebnisse auf den latenten Vektoren des Modells „AE_ZW“ gezeigt, in Zeile 2 die Ergebnisse basierend auf dem durch das RNN vorhergesagten nächsten Bauteilzustand in Form des latenten Vektors. Der vorhergesagte Zustand ist dabei über eine Punktwolke angedeutet. Für den ersten Zwischenzustand ist über beide Ansätze die Menge der geometrisch ähnlichen Modelle leer. Jedoch wird als nächstes Modell

außerhalb des Suchradius ϵ für die Suche mittels vorhergesagtem Bauteilzustand bereits ein Flansch gefunden, der große Ähnlichkeit zum tatsächlichen finalen Zustand des Flanschs aufweist. Wird die Vorhersage auf dem zweiten Zwischenzustand getätigt und als Sucheingabe genutzt, so werden bereits 9 ähnliche Flansche gefunden. Grün markiert sind dabei diejenigen, die gemäß Abbildung 5.26 bei der Suche auf Basis des finalen Zustandes ebenfalls unter den besten 3 Ergebnissen zu finden sind. Eine weitere Sortierung auf Basis der Metainformationen wird aufgrund der geringen Anzahl ähnlicher Modelle nicht vorgenommen.

Bauteilzustand der Komponente i			Menge der ähnlichsten Modelle $S_{sim,i}$ des Bauteilzustandes der Komponente i				Ähnlichstes Modell außerhalb von $S_{sim,i}$	
			Top 3 der ähnlichsten Modelle aus $S_{sim,i}$			Unähnlichstes Modell aus $S_{sim,i}$		$ S_{sim,i} $
Ohne Vorhersage	 Ohne Vorhersage	Latente Vektoren AE_ZW	X				0	Cover_230005932 (64,26%) 
			X				0	Flange_230033736 (75,85%) 
Mit Vorhersage	 Mit Vorhersage	Latente Vektoren AE_ZW mit Vorhersage durch RNN	X				0	Flange_230033736 (78,00%) 
			Flange_230005963 (86,92%) 	Flange_230036423 (82,51%) 	Flange_230036438 (82,40%) 	Flange_230036650 (80,95%) 	Flange_230005964 (80,89%) 	

 Übereinstimmung innerhalb der Top 3 Ergebnisse mit der Suche auf Basis des finalen Zustandes

Abbildung 5.39: Ergebnisse der Ähnlichkeitssuche für den Flansch 230036437 für den ersten und zweiten Zwischenzustand auf den latenten Vektoren des Modells „AE_ZW“ sowie der Vorhersage durch das RNN

Das Ergebnis der durch die Vorausschau ermöglichten Bewertung der produktionsrelevanten Produkteigenschaften auf Basis der gefundenen ähnlichen Modelle für den zweiten Zwischenzustand des Flansches ist in Abbildung 5.40 visualisiert. Mittels der produktionsrelevanten Produkteigenschaften der 9 ähnlichsten Modelle werden ein Cluster und damit ein Bereich PP_1 definiert. Alle Flansche in diesem Bereich müssen zunächst gegossen und anschließend spanend nachbearbeitet werden. Die geringe Varianz in den möglichen Abmessungen der ähnlichen Teile ist auf das zur Fertigung

Gussaufmaß zurückzuführen und somit vom Konstrukteur bewusst gewählt. Insbesondere für unerfahrene Konstrukteure kann jedoch das Aufzeigen der abweichenden Abmessungen einen Hinweis darauf geben, hier nochmals genauer bereits existierende Flansche zu betrachten. Im Anwendungsunternehmen werden tatsächlich drei verschiedene Varianten von Rohflanschen unterschieden, aus denen durch die spanende Bearbeitung schließlich die verschiedenen Varianten erzeugt werden. Diese sollen in der Konstruktion möglichst allen verschiedenen Varianten zugrunde gelegt werden. Die entsprechenden CAD-Modelle sind ebenfalls im Datensatz enthalten und befinden sich für den in Abbildung 5.39 gezeigten anfänglichen Zustand des Flansches in der Menge der ähnlichsten Modelle. Dabei entspricht das ähnlichste Modell dem tatsächlichen Rohling, der dem betrachteten Flansch zugrunde gelegt ist.



Abbildung 5.41: Funktionsflächen von Flansch 230036437 aus verschiedenen Perspektiven

Eine Grenze des Ansatzes wird anhand der aufgezeigten Spannweite der y -Abmessung deutlich. Diese lässt sich durch den in Abbildung 5.42 gezeigten Flansch mit zusätzlicher Lasche an der Oberkante erklären. Dieser wird über den Suchradius ε noch als ausreichend geometrisch ähnlich zum aktuellen Bauteilzustand deklariert⁶⁴, weshalb dessen produktionsrelevante Produkteigenschaften in die Bewertung mit einfließen. Der Flansch wird zwar ebenfalls zunächst gegossen und anschließend spanend bearbeitet, jedoch ist für dessen Urform aufgrund der Lasche ein gesondertes Werkzeug notwendig. Alle weiteren Flansche weisen dagegen einen einheitlichen Wert bezüglich der y -Abmessung auf. Grund hierfür ist, dass im aufgezeigten Beispiel vor allem die äußere Form entscheidend ist, geometrische Unterschiede, z.B. aufgrund von Bohrungen, sind dagegen weniger relevant. Mit dem gezeigten Vorgehen kann dies jedoch nicht differenziert werden, da nur die gesamte Geometrie ausschlaggebend ist. Darüber hinaus ist der für das Clusterverfahren DBSCAN gewählte Schwellenwert ε_{DB} entscheidend dafür, ob eine Kombination von produktionsrelevanten Produkteigenschaften

⁶⁴ Auch bei der Suche auf Basis des finalen Bauteilzustandes des Flansches ist dieser gerade noch in der Menge der geometrisch ähnlichen Modelle enthalten.

noch einem Cluster zugeordnet wird oder aufgrund zu starker Abweichungen als Rauschpunkt oder - sofern noch weitere Bauteile mit ähnlicher Kombination vorliegen - als eigener Bereich definiert wird. Folglich beeinflusst die Wahl von ε_{DB} die erlaubte Variabilität innerhalb der Ausprägungen der produktionsrelevanten Produkteigenschaften. Diese kann jedoch beispielsweise in Abhängigkeit der Fertigungsverfahren sehr unterschiedlich sein. Da im betrachteten Anwendungsfall verschiedene Fertigungsverfahren für die unterschiedlichen Bauteile zum Einsatz kommen, der Wert von ε_{DB} aber auf Basis aller Bauteile bestimmt wird, ist dieser für das gezeigte Beispiel des Flansches tendenziell zu groß. Durch eine geringere Wahl von ε_{DB} würde die entsprechende Kombination aus dem Cluster ausgeschlossen werden.

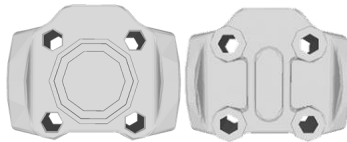








Abbildung 5.42: Seitenflansch mit zusätzlicher oberer Lasche

Die Ähnlichkeitssuche auf Basis des vierten Zwischenzustandes eines Adapters mit zusätzlicher Berücksichtigung von Metainformationen liefert gemäß Abbildung 5.43 Zeile 1 ohne eine Vorhersage des Folgezustandes keine ähnlichen Modelle. Durch die Hinzunahme der Vorausschau (dem Adapter wird eine Bohrung hinzugefügt) dagegen werden 36 ähnliche Adapter gefunden. Dabei stimmen die drei ähnlichsten Modelle mit den Ergebnissen der Suche auf dem finalen Zustand des Adapters überein. Die Bewertung der produktionsrelevanten Produkteigenschaften des Adapters auf Basis der über die Vorausschau gefundenen ähnlichen Teile ist in Abbildung 5.44 oben dargestellt. Es werden insgesamt zwei Bereiche identifiziert. Der Adapter wird dabei PP_1 als Kernpunkt zugeordnet. Da alle Eigenschaften im üblichen Bereich liegen, ist die Produzierbarkeit auf den vorhandenen Fertigungsmitteln voraussichtlich gegeben. Der Adapter wird tatsächlich auf einem Universal-Drehautomaten für Stangenbearbeitung mit einem Spindeldurchlass von 65 mm und einer maximalen Drehlänge von 710 mm gefertigt. Folglich sollten die Bauteile einen maximalen Durchmesser von 65 mm und eine maximale Länge von 710 mm nicht überschreiten. Der aus den ähnlichen Teilen abgeleitete Bereich der Abmessungen mit einem maximalen Wert von 49,5 mm für y- und z-Abmessung (entspricht dem Durchmesser) und einer maximalen x-Abmessung von 69,5 mm (entspricht der Länge) liegt innerhalb dieser Grenzen und scheint daher plausibel.

Input-Komponente:  **Adapter 230005421 (Zwischenzustand 4)**
 Material: Stahl
 Abmessungen: 50,2 x 34,9 x 34,9 [mm]
 Volumen: 25.069 [mm³]

Gewichtung		Top 3 der ähnlichsten Modelle				Unähnlichstes Modell aus $S_{sim,l}$	$ S_{sim,l} $
Latente Vektoren AE_ZW ohne Vorhersage	$\alpha_{geo} = 0,2$ $\alpha_{mat} = 0,2$ $\alpha_{abm} = 0,5$ $\alpha_{vol} = 0,1$	[X]					0
	Ansicht von oben 	[X]					
Latente Vektoren AE_ZW mit Vorhersage	$\alpha_{geo} = 0,2$ $\alpha_{mat} = 0,2$ $\alpha_{abm} = 0,5$ $\alpha_{vol} = 0,1$	Adapter 230005420	Adapter 230005419	Adapter 230038868	Adapter 230002357		36
	Ansicht von oben 	Geometrie: 94,1% Material: 100% Abmessung: 100% Volumen: 99,9% Gewichtete Ähnlichkeit: 98,8%	Geometrie: 93,2% Material: 100% Abmessung: 99,9% Volumen: 99,9% Gewichtete Ähnlichkeit: 98,6%	Geometrie: 90,1% Material: 100% Abmessung: 99,7% Volumen: 99,9% Gewichtete Ähnlichkeit: 97,8%	Geometrie: 90,4% Material: 0% Abmessung: 100% Volumen: 99,9% Gewichtete Ähnlichkeit: 78,1%	   	

Übereinstimmung innerhalb der Top 3 Ergebnisse mit der Suche auf Basis des finalen Zustandes

Abbildung 5.43: Ergebnisse der Ähnlichkeitssuche für den Adapter 230005421 für den vierten Zwischenzustand auf den latenten Vektoren des Modells „AE_ZW“ so wie der Vorhersage durch das RNN

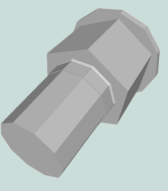
	Bewertung PP ₁					
	Eigenschaft	Einheit	Wert	Minimum	Maximum	Bewertung
Abmessung x-Richtung	mm	50,20	33,50	69,50		
Abmessung y-Richtung	mm	34,90	20,00	49,50		
Abmessung z-Richtung	mm	34,90	20,00	49,50		
Material	-	Stahl	Stahl			
Gesamtbewertung						

Abbildung 5.44: Bewertung der produktionsrelevanten Produkteigenschaften für den Zwischenzustand von Adapter 230005421

Der zweite Bereich beinhaltet dagegen Adapter, die einen Durchmesser von 90 mm aufweisen und auf einem anderen Typ Dreh-Fräszentrum hergestellt werden. Im An-

wenderunternehmen werden für die fünf baugleichen Universal-Drehautomaten Rüstfamilien gebildet, um möglichst wenige Werkzeugwechsel zu erreichen. Ausschlaggebend für die benötigten Drehwerkzeuge ist dabei die geometrische Form eines Adapters. Entsprechend werden die verschiedenen Adapter auf die fünf baugleichen Universal-Drehautomaten aufgeteilt. Von den 32 ähnlichen Adaptern aus dem Bereich PP₁ werden 20 auf derselben Maschine wie der untersuchte Adapter gefertigt. Lediglich ein Adapter, der an drittletzter Stelle in der Menge der ähnlichsten Modelle steht, zählt zu einer anderen Rüstfamilie. Für die weiteren liegt keine entsprechende Zuordnung vor⁶⁵.

Die dargestellten Beispiele zeigen, dass durch die Vorhersage des nächsten Konstruktionszustandes insbesondere für anfängliche Bauteilzustände bereits geometrisch ähnliche finale Modelle aufgefunden werden können, wodurch ein frühzeitiges Aufzeigen von wiederverwendbaren Komponenten ermöglicht wird. Darüber hinaus können auf Basis dieser geometrisch ähnlichen Komponenten die für einen Konstruktionszustand gewählten produktionsrelevanten Produkteigenschaften bewertet werden. Anhand der dargelegten Beispiele kann gezeigt werden, dass die gefundenen ähnlichen Teile große Ähnlichkeiten zu den tatsächlichen finalen Bauteilzuständen aufweisen. Über die daraus abgeleiteten üblichen Ausprägungen der produktionsrelevanten Produkteigenschaften und das frühzeitige Aufzeigen abweichender Werte kann eine produktionsgerechte Produktentwicklung gefördert werden. Darüber hinaus werden die Bauteile innerhalb eines Bereichs zur Bewertung der produktionsrelevanten Produkteigenschaften tatsächlich auch mit den gleichen, im Anwenderunternehmen verfügbaren Fertigungsverfahren hergestellt. Eine generelle Bewertung, ob ein Bauteil prinzipiell produzierbar ist oder nicht, ist jedoch nicht möglich.

⁶⁵ Die vorhandenen Daten zur Maschinenzuordnung stammen aus einem Auszug des ERP-Systems für die in den Folgetagen bereits eingeplanten Adapter, weshalb nicht für alle Materialnummern eine Zuordnung vorliegt. Nach Aussage eines Experten des Anwenderunternehmens sind jedoch auch alle weiteren Adapter auf demselben Typ Maschine mit ähnlichen Werkzeugen prinzipiell fertigbar. Aufgrund von großen Stückzahlen wird ein Teil jedoch auch fremdgefertigt.

6 Diskussion und Ausblick

In den nachfolgenden Kapiteln wird aufgezeigt, inwieweit das dargestellte Vorgehen zur ML-basierten Extraktion von Wissen aus Produktmodellen zur anschließenden Nutzung in der Produktentwicklung zur Beantwortung der eingeführten Forschungsfragen beiträgt sowie das abgeleitete Forschungsdefizit deckt. Auf Basis dieser Bewertung wird anschließend ein Ausblick für mögliche Forschungsarbeiten zur Weiterentwicklung der Methodik gegeben.

6.1 Diskussion

Über die im Rahmen dieser Arbeit entwickelte und beispielhaft angewandte Methodik können die in Kapitel 1.2 eingeführten Forschungsfragen beantwortet werden.

Die erste Forschungsfrage lautet, wie speziell ML-Verfahren dazu genutzt werden können, die systematische Wiederverwendung von Wissen in Form von vorhandenen CAD-Modellen und damit verknüpften Informationen schon frühzeitig zu fördern. Die Anwendung von ML-Verfahren im Bereich von 3D-Daten ist in jüngster Zeit zwar stark gestiegen, jedoch gibt es im Stand der Technik noch kaum Ansätze, die diese Verfahren auf den Bereich der Produktentwicklung und damit reale und industrielle CAD-Daten übertragen. Im Rahmen dieser Arbeit wird daher ein *Autoencoder* basierend auf Yang et al. (2017) dazu herangezogen, zunächst relevante geometrische Informationen aus den CAD-Objekten zu extrahieren. Im Gegensatz zu herkömmlichen Verfahren aus der Domäne der Produktentwicklung, die auf vordefinierten Merkmalen zur Beschreibung einer Produktkomponente basieren, ist der *Autoencoder* fähig, selbstständig die datensatzspezifischen relevanten Merkmale zu extrahieren. In Kombination mit weiteren Metainformationen, wie beispielsweise produktionsrelevante Produkteigenschaften, bildet dieser Merkmalsvektor den geometrischen Fußabdruck einer Komponente. Durch einfache Berechnung der euklidischen Distanz werden für einen aktuellen Konstruktionszustand, repräsentiert über dessen geometrischen Fußabdruck, schon frühzeitig (geometrisch) ähnliche finale CAD-Modelle aufgezeigt. Anhand der beispielhaften Anwendung des Vorgehens auf einem industriellen Datensatz wird deutlich, dass im Vergleich zu Ansätzen basierend auf vordefinierten Eigenschaften über den extrahierten Merkmalsvektor geometrische Eigenschaften deutlich besser abgebildet werden. Über die gefundenen ähnlichen CAD-Modelle können zudem die produktionsrelevanten Produkteigenschaften bewertet werden, was in der Beantwortung von Forschungsfrage

zwei näher beleuchtet wird. Darüber hinaus kann die erlernte Repräsentation je Komponente zur Abbildung von Konstruktionsmustern genutzt werden, worauf näher im Rahmen der Beantwortung der dritten Forschungsfrage eingegangen wird.

Die möglichen Rückschlüsse bezüglich produktionsrelevanter Produkteigenschaften, die aus vorhandenen CAD-Modellen gezogen werden können, sind Inhalt von Forschungsfrage zwei. Ein wesentlicher Einflussparameter auf die Produzierbarkeit einer Komponente ist deren geometrische Gestalt. Um die Ausprägung produktionsrelevanter Produkteigenschaften, wie beispielsweise die Abmessungen einer Komponente, bewerten zu können, werden im Rahmen dieser Arbeit daher die über die geometrische Ähnlichkeitssuche aufgefundenen, bereits produzierten Produktkomponenten herangezogen. Anhand dieser Komponenten können die für eine gewisse Geometrie bereits produzierbaren Ausprägungen der Eigenschaften abgeleitet werden, die im geometrischen Fußabdruck hinterlegt sind. Diese spannen einen oder mehrere Bereiche auf, die zulässige Kombinationen der (ggf. voneinander abhängigen) Eigenschaften aufzeigen. Beispielsweise lassen sich in Abhängigkeit des Materials, das die Wahl des Fertigungsverfahrens stark beeinflusst, verschiedene Bereiche für die Abmessungen identifizieren. Für einen aktuellen Konstruktionsstand kann folglich überprüft werden, ob dessen produktionsrelevante Produkteigenschaften innerhalb eines üblichen Bereichs liegen oder ob deren Werte als kritisch einzustufen sind.

Die dritte Forschungsfrage, die das Erlernen von Mustern im Konstruktionsvorgehen mittels ML und die anschließende Nutzung dieser Muster im Konstruktionsprozess adressiert, kann im aktuellen Stand der Technik nur unzureichend beantwortet werden. Erst in jüngster Zeit wurde durch Willis et al. (2021) und Wu, Xiao & Zheng (2021) gezeigt, dass das Erlernen von Mustern aus dem Konstruktionsvorgehen prinzipiell möglich ist. Ein wesentlicher Schritt hierfür ist zunächst eine für ML geeignete Darstellung und Aufbereitung der Konstruktionshistorie des CAD-Modells. Das Konstruktionsvorgehen wird in den zuvor genannten Arbeiten dafür über die Sequenz der KEs, die aus dem Strukturbaum abgeleitet werden können, abgebildet. Im Rahmen der entwickelten Methodik wird das Konstruktionsvorgehen zudem über die geometrische Entwicklung des 3D-Modells repräsentiert. Hierfür werden die Konstruktionszwischenstände nach jeder weiteren Bearbeitung durch ein Konstruktionselement als 3D-Objekt abgespeichert und in eine latente Repräsentation überführt. Die KEs selbst werden als kategoriale Variable betrachtet und entsprechend kodiert. Als Resultat liegen Sequenzen von Vektoren vor, über die die Konstruktionshistorie eines CAD-Modells dargestellt wird

und die als Grundlage zum Erlernen von Mustern dienen. Zur Verarbeitung dieser sequentiellen Daten werden anschließend RNNs herangezogen. Diese verfügen über einen internen Speicher und sind somit fähig, nicht nur den aktuellen Konstruktionszustand, sondern auch den zeitlichen Verlauf der zu diesem Stand führenden Konstruktions-schritte zu verarbeiten. Auf diese Weise können typische Muster im Konstruktions-vorgehen erlernt und auf die Konstruktion neuer Komponenten übertragen werden. Für einen gegebenen Bauteilzustand lässt sich so der nächste Konstruktionszustand vorhersagen. Dieser dient einerseits dem Konstrukteur als Orientierung, andererseits kann diese Vorausschau als verbesserte Ausgangslösung für die Ähnlichkeitssuche genutzt werden. Im dargestellten Anwendungsfall konnte gezeigt werden, dass insbesondere in den anfänglichen Konstruktionszuständen größere geometrische Änderungen vorgenommen werden, die durch das RNN korrekt prädiziert werden. Mittels des vorhergesagten Bauteilzustandes wird darüber hinaus für den gezeigten Anwendungsfall ein frühzeitiges Auffinden passender geometrisch ähnlicher CAD-Modelle ermöglicht.

Den in Kapitel 3.4 aufgezeigten *Forschungsdefiziten* wird durch die entwickelte Methodik anhand der nachfolgend aufgeführten Punkte begegnet. Im Vergleich zu bestehenden Ansätzen basiert das Vorgehen dieser Arbeit auf *mehreren Informationsarten*, die zur Beschreibung einer Produktkomponente vorliegen. Neben der reinen Geometrie werden zusätzliche Metainformationen, die entweder direkt aus dem CAD-Modell selbst extrahiert werden können oder im PLM-System mit der Komponenten verknüpft sind, sowie die Konstruktionshistorie berücksichtigt. Dadurch können die Vorteile bestehender ML-Ansätze auf 3D-Daten mit Ansätzen aus der Domäne der Produktentwicklung kombiniert werden. Im Gegensatz zu bestehenden ML-Anwendungen im Bereich der Produktentwicklung werden lediglich *Informationen zugrunde gelegt, die bereits vorhanden oder direkt extrahierbar* sind. Eine *künstliche Erzeugung von Wissen* (z.B. Simulationsdaten) *ist nicht notwendig*. Die entwickelte Methodik zum Auffinden ähnlicher Komponenten betrachtet im Gegensatz zu bestehenden Ansätzen speziell auch den Fall halbfertiger Bauteilzustände. In Kombination mit der Vorhersage nächster Konstruktionszustände, die auf realen und komplexen Konstruktionssequenzen erprobt wird, kann die Ergebnisqualität der *Ähnlichkeitssuche auf halbfertigen Komponenten* noch weiter gesteigert werden. Insbesondere für anfängliche Bauteilzustände ergeben sich im Vergleich zur dargestellten Suche rein auf Metainformationen basierend deutlich bessere Suchergebnisse.

Dem aufgezeigten Forschungsbeitrag der entwickelten Methodik sind die nachfolgend aufgeführten kritischen Aspekte gegenüberzustellen. Wie im aufgeführten Anwendungsbeispiel gezeigt wurde, ist die Güte der durch den *Autoencoder* erlernten Repräsentation stark abhängig von einer einheitlichen Ausrichtung der Modelle. Ist dies nicht gewährleistet, so resultieren zwei geometrisch identische Komponenten, die jedoch unterschiedlich orientiert sind, in verschiedenen latenten Repräsentationen. Darüber hinaus können Unterschiede in geometrischen Details, die in der Darstellung der CAD-Modelle über eine Punktwolke oder in der für das Training des *Autoencoders* genutzten *Chamfer Distance* nicht ausreichend abgebildet werden, auch nicht in der latenten Repräsentation enkodiert werden. Da eine ausreichende Trainingsdatenmenge eine Grundvoraussetzung für die Anwendung von ML-Verfahren darstellt, können sehr seltene Geometrien bzw. geometrische Eigenschaften nur unzureichend abgebildet werden. Dieser Nachteil spiegelt sich auch in der Vorhersage nächster Konstruktionszustände wider. Größere geometrische Veränderungen können zwar korrekt prädictiert werden, kleinere Details, wie Abrundungen, sind jedoch in der Visualisierung als rekonstruierte Punktwolke, auch wenn sie korrekt prognostiziert werden, kaum zu erkennen. Werden diese Details darüber hinaus bereits in den für das Training des *Autoencoders* genutzten Punktwolken oder der für den Rekonstruktionsverlust genutzten CD nicht ausreichend dargestellt und somit in der latenten Repräsentation nicht oder nur unzureichend enkodiert, können sie nicht als Veränderungen zwischen zwei Konstruktionszuständen erlernt werden.

Ein wesentlicher Parameter zur Bestimmung der Menge der geometrisch ähnlichen Modelle ist der Suchradius ϵ . Dieser hängt stark von der Dichte der Datenpunkte und damit der Distanz der latenten Repräsentationen im betrachteten Datensatz ab. Geometrische Distanzen können jedoch in Abhängigkeit der Bauteilkategorie unterschiedlich ausfallen. Im aufgezeigten Anwendungsfall weist beispielsweise die Klasse der O-Ringe aufgrund der sehr geringen Komplexität allgemein niedrigere Distanzen auf als die Klasse der Gehäuse. Da der Suchradius in Abhängigkeit der durchschnittlichen Distanzen über den kompletten Datensatz gewählt wird, resultieren für Gehäuse tendenziell weniger ausreichend ähnliche Suchergebnisse, als es für O-Ringe der Fall ist. Ein ähnlicher Aspekt lässt sich im Hinblick auf die Bestimmung der Bereiche produktionsrelevanter Produkteigenschaften anmerken. Grundlage hierfür bilden ebenfalls Bereiche unterschiedlicher Dichte innerhalb der relevanten Datenbasis. Bei einer sehr geringen Datenmenge ist es daher möglich, dass ein potenzieller Bereich aufgrund mangelnder

Datenpunkte und damit unzureichender Dichte nicht als solcher identifiziert wird. Darüber hinaus ist die Wahl des Schwellenwerts ε_{DB} ausschlaggebend für die erlaubte Varianz innerhalb der Ausprägungen der produktionsrelevanten Produkteigenschaften. Diese ist jedoch beispielsweise stark abhängig vom eingesetzten Fertigungsverfahren. Werden jedoch, wie im aufgezeigten Anwendungsfall, Bauteile mit verschiedenen Fertigungsverfahren eingesetzt, können spezifische Gegebenheiten nicht ausreichend abgedeckt werden.

6.2 Ausblick

Die aufgezeigte Methodik bildet die Grundlage für weitere Forschungs- und Anwendungsmöglichkeiten, die einerseits die zuvor aufgezeigten Schwächen adressieren und andererseits die Vorgehensweise auf andere Domänen übertragen können.

Die Extraktion der relevanten geometrischen Eigenschaften mittels eines *Autoencoders* lässt sich anhand mehrerer Aspekte verbessern. Im Rahmen dieser Arbeit wurde aufgezeigt, dass eine einheitliche Ausrichtung der CAD-Modelle für das spätere Trainingsergebnis des *Autoencoders* ausschlaggebend ist. Anstelle der aufgezeigten Standardorientierung, die vorab durchgeführt wird, kann eine einheitliche Ausrichtung durch den Ansatz von Sun et al. (2021) im Training integriert und somit erlernt werden.

Im gezeigten Vorgehen dient die *Chamfer Distance* (CD) zur Bewertung der Rekonstruktionsgüte und somit als Verlustfunktion für das Training. Diese bildet jedoch nicht zwingend die tatsächlichen Distanzen zwischen zwei Punktwolken korrekt ab (Urbach, Ben-Shabat & Lindenbaum 2020). Da die Verteilung der Punkte auf der Oberfläche eines CAD-Modells zufällig erfolgt, können zwei identische Objekte über zwei unterschiedliche Punktwolken dargestellt werden. Hier ist es möglich, dass eine CD ungleich null auftritt, obwohl die Objekte identisch sind (Urbach, Ben-Shabat & Lindenbaum 2020). Darüber hinaus bildet die CD kleine Unterschiede nur unzureichend ab (Urbach, Ben-Shabat & Lindenbaum 2020). Eine Verbesserung der CD wird beispielsweise in Wagner & Schwanecke (2022) vorgeschlagen.

Die Architektur des *Autoencoders* selbst kann gemäß dem Vorgehen von Pang, Li & Tian (2021) angepasst werden, um eine topologiefreundlichere Repräsentation zu ermöglichen, über die auch nicht zusammenhängende Körper korrekt als solche dargestellt werden. Darüber hinaus können neben der reinen Geometrie auch Metainformationen, wie beispielsweise die Abmessungen oder das Material, mit in den latenten Raum überführt werden, sodass diese Aspekte bereits bei der Bestimmung der Menge

der ähnlichsten Modelle berücksichtigt werden. Ein möglicher Ansatz hierfür wird in Krahe et al. (2022) vorgestellt. Der zur Bestimmung der Menge der geometrisch ähnlichen Modelle ausschlaggebende Suchradius ε wurde für den betrachteten Anwendungsfall empirisch auf Basis aller Daten ermittelt. Um jedoch je nach Komplexität der Bauteile und somit der erhaltenen Distanzen im latenten Raum dessen Wert festzulegen, wäre eine Bestimmung je Cluster denkbar. Werden zum Auffinden der ähnlichen Modelle mehrere Cluster betrachtet, könnte beispielsweise der durchschnittliche Wert von ε herangezogen werden. Ein ähnlicher Nachteil ergibt sich für den Schwellenwert ε_{DB} , der zur Abgrenzung der verschiedenen Bereiche der produktionsrelevanten Produkteigenschaften der geometrisch ähnlichen Modelle genutzt wird. Verbesserungen könnten hier beispielsweise erzielt werden, indem ε_{DB} nicht im vorab für den gesamten Datensatz, sondern jeweils basierend auf der Menge der ähnlichsten Modelle bestimmt wird.

Zum Erlernen von Konstruktionsmustern wurden im Rahmen dieser Arbeit verschiedene RNN-Architekturen herangezogen. Mögliche Verbesserungen können hier durch die Anwendung von *Transformer*-Architekturen (Vaswani et al. 2017) erzielt werden, die in der Sprachverarbeitung bereits erfolgreich Anwendung finden. Darüber hinaus kann eine iterative Anwendung des Vorhersagemodells erprobt werden, um eine umfassendere Vorausschau zu ermöglichen.

Eine wesentliche Grundvoraussetzung für die Anwendung von ML-Verfahren auf 3D-Objekten ist zunächst eine geeignete Repräsentationsform. In der vorliegenden Arbeit wurden hierfür Punktwolken herangezogen. Diese sind jedoch nicht fähig, geometrische Details ausreichend abzubilden. Anstelle von Punktwolken kann gemäß dem Ansatz von Wu, Xiao & Zheng (2021) der CAD-Strukturbaum selbst als Repräsentation und Input für den *Autoencoder* herangezogen werden. Neben der exakten Darstellung der Geometrie ergibt sich dadurch der große Vorteil, dass die durch das Vorhersagemodell generierten nächsten Konstruktionszustände in ein CAD-lesbares Format dekodiert und somit direkt in eine CAD-Umgebung integriert werden können. Dadurch können nächste Konstruktionszustände nicht nur visualisiert, sondern im CAD-Modell selbst umgesetzt werden.

Insgesamt sollte in zukünftigen Forschungsarbeiten die Frage adressiert werden, zu welchem Zeitpunkt ein erneutes Training der ML-Verfahren aufgrund einer veränderten Datenbasis durch das Hinzufügen neuer Komponenten notwendig ist.

Durch die aufgezeigten Potenziale kann die ML-basierte Extraktion und Formalisierung von Wissen aus vorhandenen Produkten noch weiter verbessert werden, um eine zielgerichtete und systematische Nutzung dieses Wissens in der Produktentwicklung zu fördern. Dadurch kann ein wesentlicher Beitrag für eine integrierte Betrachtung von Produkt- und Produktionssystementwicklung im Sinne eines Produkt-Produktions-Codesigns geleistet werden.

Neben der Anwendung in der Produktentwicklung ist eine Nutzung der entwickelten CAD-Ähnlichkeitssuche auch in anderen Domänen denkbar. Im Bereich der Fertigungsplanung können für ein neues Bauteil beispielsweise aus vorhandenen, sehr ähnlichen Komponenten Fertigungsinformationen (z.B. Fertigungsverfahren, Richtpreise) abgeleitet werden. Eine weitere Anwendung ist auch im Bereich des taktischen Einkaufs denkbar. Hier können ähnliche Produkte gebündelt und damit das Bestellvolumen bei Lieferanten erhöht werden, wodurch bessere Preise erreicht werden können.

7 Zusammenfassung

Produzierende Unternehmen stehen heutzutage vor der Herausforderung, in immer kürzerer Zeit dennoch innovative, vermehrt kundenindividuelle und zeitgleich kostengünstige Produkte auf den Markt zu bringen. Insbesondere für den Bereich der Produktentwicklung entsteht dadurch ein enormer Kosten- und Zeitdruck. Durch die systematische Wiederverwendung von vorhandenen Produktmodellen sowie des damit verknüpften Wissens ist es jedoch möglich, die Entwicklungszeit neuer Produkte zu reduzieren, Risiken, wie beispielsweise eine Nicht-Produzierbarkeit auf vorhandenen Fertigungsmitteln, zu minimieren oder Dubletten zu vermeiden und somit Kosten einzusparen. Problem ist jedoch, dass dieses Wissen häufig impliziter Natur und damit nicht ohne weiteres formalisierbar ist. Durch den verstärkten Einsatz digitaler Tools in der Produktentwicklung und der damit einhergehenden wachsenden Datenbasis eröffnet sich jedoch die Möglichkeit, datengetriebene Ansätze basierend auf künstlicher Intelligenz, speziell Maschinellem Lernen (ML), zu nutzen. Diese sind fähig, selbstständig aus den vorhandenen Daten (implizites) Wissen zu extrahieren, zu formalisieren und somit nutzbar zu machen. Die übergeordnete Fragestellung dieser Arbeit ist folglich, wie ML-Verfahren dazu genutzt werden können, bestehendes Wissen in Form von vorhandenen Produktmodellen, speziell hinsichtlich CAD-Modellen, schon frühzeitig in der Konstruktion zielgerichtet zur Verfügung zu stellen, um eine systematische Nutzung und Wiederverwendung zu fördern. Dadurch wird einem wesentlichen Aspekt zur Umsetzung eines umfassenden Produkt-Produktions-Codesigns beigetragen.

Aus dem Stand der Technik zu ML-Anwendungen auf 3D-Objekten sowie bestehenden Assistenzsystemen im Produktentwicklungsprozess folgt, dass in jüngster Zeit starke Fortschritte auf dem Gebiet der Verarbeitung von 3D-Daten mittels ML erzielt wurden, deren Übertragung und Nutzung auf den Bereich der Produktentwicklung jedoch noch kaum anzutreffen sind. Insbesondere für eine geeignete Repräsentation der geometrischen Form von CAD-Modellen werden in bisherigen Ansätzen lediglich vordefinierte Eigenschaften herangezogen, auf deren Basis beispielsweise die Ähnlichkeit zweier Modelle bestimmt werden kann. Das große Potenzial, mit Hilfe von ML-Verfahren selbstständig die datenspezifisch relevanten Merkmale zu extrahieren und beispielsweise für eine Ähnlichkeitssuche zu nutzen, wird jedoch noch kaum genutzt.

In dieser Arbeit wird eine Methodik entwickelt, um zunächst aus einer vorhandenen Datenbasis von (industriellen) CAD-Modellen auf Komponentenebene das darin enthaltene Wissen aufzubereiten und zu extrahieren. Hierfür werden in einem ersten Schritt drei Datenkategorien definiert, die die Wissensbasis für die spätere Konstruktionsunterstützung bilden. Neben der eigentlichen 3D-Geometrie eines CAD-Modells zählen hierzu auch dessen Konstruktionshistorie sowie weitere Metainformationen, die im CAD-Modell selbst hinterlegt (z.B. Abmessungen) oder beispielsweise im PLM-System damit verknüpft sind (z.B. Kosten). Nach einer entsprechenden Aufbereitung dieser Daten erfolgt die automatisierte Extraktion der relevanten geometrischen Eigenschaften mittels *Autoencoder*. Die zugrunde gelegte *Autoencoder*-Architektur basierend auf Yang et al. (2018) wird für die Nutzung von (industriellen) CAD-Daten angepasst. Mit Hilfe der *Autoencoder*-Architektur ist es schließlich möglich, für jedes CAD-Modell einen zugehörigen Merkmalsvektor zu extrahieren, der die relevanten geometrischen Eigenschaften enthält. Gemeinsam mit den zusätzlichen Metainformationen bildet dieser Vektor den geometrischen Fußabdruck der entsprechenden Komponente. Darüber hinaus wird aus jedem CAD-Modell die Konstruktionshistorie extrahiert, die aus der geometrischen Entwicklung des 3D-Modells, jeweils dargestellt über den Merkmalsvektor, sowie den zugehörigen Konstruktionselementen (KEs) besteht. Das so repräsentierte Wissen bildet schließlich die Grundlage für die Konstruktionsunterstützung. Diese umfasst zum einen das Auffinden ähnlicher CAD-Modelle für einen gegebenen Konstruktionszustand. Basis hierfür bilden die extrahierten geometrischen Fußabdrücke jeder Komponente. Die komprimierte Darstellung als Vektor ermöglicht die Bestimmung der geometrischen Ähnlichkeit über die Berechnung der euklidischen Distanz. Hierfür werden zunächst die relevanten geometrisch ähnlichen Modelle identifiziert, die schließlich anhand der weiteren Metainformationen sortiert werden. Auf Basis der geometrisch ähnlichen Modelle erfolgt anschließend eine Bewertung der produktionsrelevanten Produkteigenschaften des aktuellen Konstruktionszustandes. Hierfür werden entsprechende Wertebereiche aus den geometrisch ähnlichen Komponenten, wie beispielsweise zulässige Abmessungen oder Materialien, abgeleitet. Können die gewählten Eigenschaften des aktuellen Konstruktionszustandes keinem Wertebereich zugeordnet werden, so werden sie als kritisch eingestuft und die identifizierten Abweichungen aufgezeigt. Hierdurch können dem Konstrukteur zum einen Hinweise auf eine mögliche Nicht-Produzierbarkeit mit den vorhandenen Fertigungsmitteln gegeben werden.

Zum anderen beinhaltet die Konstruktionsunterstützung das Vorschlagen nächster Konstruktionsschritte in Form von Folgezuständen. Hierfür werden zunächst anhand der extrahierten Konstruktionshistorien mittels RNNs typische Konstruktionsmuster erlernt. Auf Basis dieser erlernten Muster kann dann für einen gegebenen Konstruktionszustand der nächste Folgezustand, der durch die Ausführung eines weiteren Bearbeitungsschritts resultiert, vorhergesagt werden. Dieser dient einerseits als Orientierung für den Konstrukteur, andererseits kann die Vorausschau auch als verbesserte Sucheingabe für die Ähnlichkeitssuche genutzt werden. So wird es möglich, auch für anfängliche Konstruktionszustände, die noch starke (geometrische) Unterschiede zum gewünschten finalen Zustand aufweisen, dennoch ähnliche Komponenten aufzufinden.

Dieses Vorgehen wurde anhand von industriellen CAD-Daten in Form von mechanischen Komponenten aus fünf verschiedenen Bauteilklassen erprobt. Für verschiedene Konstruktionszustände konnte gezeigt werden, dass die im Rahmen dieser Arbeit entwickelte Ähnlichkeitssuche, basierend auf den mittels *Autoencoder* extrahierten Merkmalsvektoren, im Vergleich zu herkömmlichen Ansätzen auf vordefinierten Eigenschaften qualitativ bessere Ergebnisse, insbesondere auf unfertigen Konstruktionszuständen, liefert.

Die prinzipielle Machbarkeit des Erlernens von Konstruktionsmustern mittels RNNs und des darauf basierenden Vorhersagens nächster Konstruktionsschritte kann anhand des betrachteten Datensatzes gezeigt werden. Insbesondere am Anfang des Konstruktionsprozesses, wo größere geometrische Änderungen vorgenommen werden, werden durch die erlernten Muster korrekte Folgezustände prädiziert. Durch die zusätzliche Nutzung des Vorhersagemodells für die Ähnlichkeitssuche können darüber hinaus für anfängliche Bauteilzustände bereits passende geometrisch ähnliche Komponenten aufgefunden werden, wodurch frühzeitig eine Wiederverwendung vorhandener Komponenten gefördert wird. Die entsprechenden Resultate herkömmlicher Ansätze liefern dagegen nur Ergebnisse, die zwar dem aktuellen Konstruktionszustand, jedoch nicht dem gewünschten Zielzustand ähneln. Darüber hinaus können auf Basis dieser ähnlichen Komponenten die produktionsrelevanten Produkteigenschaften, wie z.B. die gewählten Abmessungen, auch für anfängliche Konstruktionszustände bewertet werden.

Durch die automatisierte Extraktion von produkt- und produktionssystembezogenem Wissen aus vorhandenen Produktmodellen mittels ML leistet diese Arbeit einen innovativen Ansatz, um dessen systematische Wiederverwendung zu unterstützen.

Liste eigener Publikationen

Albers et al. 2022

Albers, A.; Lanza, G.; Klippert, M.; Schäfer, L.; Frey, A.; Hellweg, F.; Müller-Welt, P.; Schöck, M.; Krahe, C.; Nowoseltschenko, K. & Rapp, S. (2022), „Product-Production-CoDesign: An Approach on Integrated Product and Production Engineering Across Generations and Life Cycles“, *Procedia CIRP*, 109, S. 167-172.

Hofmann et al. 2018

Hofmann, C.; Brakemeier, N.; Krahe, C.; Stricker, N. & Lanza, G. (2018), „The Impact of Routing and Operation Flexibility on the Performance of Matrix Production Compared to a Production line“, *Advances in Production Research*, Hrsg. Schmitt, R. & Schuh, G., S. 155-165.

Hofmann et al. 2020

Hofmann, C.; Krahe, C.; Stricker, N. & Lanza, G. (2020), „Autonomous production control for matrix production based on deep Q-learning“, *Procedia CIRP*, 88, S. 25-30.

Greinacher et al. 2020

Greinacher, S.; Overbeck, L.; Kuhnle, A.; Krahe, C. & Lanza, G. (2020), „Multi-objective optimization of lean and resource efficient manufacturing systems“, *Production Engineering*, Band 14, S. 165-176.

Kandler, Lanza & Krahe 2020

Kandler, M.; Lanza, G. & Krahe, C. (2020), „Development of a Human-centered Industry 4.0 Philosophy“, *New Developments in Sheet Metal Forming*, S. 123-135. ISBN: 9783947085033.

Krahe et al. 2019

Krahe, C.; Iberl, M.; Jacob, A. & Lanza, G. (2019), „AI-based Computer Aided Engineering for automated product design - A first approach with a Multi-View based classification“, *Procedia CIRP*, 86, S. 104–109.

Krahe et al. 2020

Krahe, C.; Bräunche, A.; Jacob, A.; Stricker, N. & Lanza, G. (2020), „Deep Learning for Automated Product Design“, *Procedia CIRP*, 91, S. 3–8.

Krahe et al. 2021

Krahe, C.; Kalaidov, M.; Doellken, M.; Gwosch, T.; Kuhnle, A.; Lanza, G. & Matthiesen, S. (2021), „AI-Based knowledge extraction for automatic design proposals using design-related patterns“, *Procedia CIRP*, 100(3), S. 397-402.

Krahe et al. 2022

Krahe, C.; Marinov, M.; Schmutz, T.; Hermann, Y.; Bonny, M.; May, M. & Lanza, G. (2022), „AI-based geometric similarity search supporting component reuse in engineering design“, *Procedia CIRP*, 109, S. 275-280.

Lanza, Klenk & Krahe 2019

Lanza, G.; Klenk, F. & Krahe, C. (2019), „Track & Trace als Basis für die Kreislaufwirtschaft in der Automobilindustrie“, *Werkstoffe in der Fertigung*, S. 22-23.

Schäfer & Krahe 2022

Schäfer, L. & Krahe, C. (2022), „KI-Assistenzsysteme in der Produktentwicklung“ in *Potentiale digitaler Führung und Technologien für die Teaminteraktion von morgen*, Hrsg. Lanza, G.; Nieken, P.; Nyhuis, P. & Trübswetter, A., TEWISS - Technik und Wissen GmbH, Garbsen, S. 68-74. ISBN: 9783959006507.

Literaturverzeichnis

A_Bräuner 2022

Bräuner, M. (2022), *Machine Learning zur Ähnlichkeitsanalyse von 3D-Objekten*, Bachelorthesis, Karlsruher Institut für Technologie, Karlsruhe, wbk Institut für Produktionstechnik.

A_Ebi 2021

Ebi, S. (2021), *Entwicklung eines Machine Learning Modells zur Vorhersage von Konstruktionsschritten basierend auf CAD-Daten*, Bachelorthesis, Karlsruher Institut für Technologie, Karlsruhe, wbk Institut für Produktionstechnik.

A_Holtzwardt 2022

Holtzwardt, T. (2022), *Vorhersage nächster Konstruktionsschritte im computergestützten Design mit Hilfe von maschinellem Lernen*, Bachelorthesis, Karlsruher Institut für Technologie, Karlsruhe, wbk Institut für Produktionstechnik.

A_Leoni 2020

Leoni, M. (2020), *Generative Modelle zur automatisierten Erstellung und Anpassung von 3D-Modellen*, Masterthesis, Karlsruher Institut für Technologie, Karlsruhe, wbk Institut für Produktionstechnik.

A_Merklinger 2021

Merklinger, P. (2021), *Gestaltung der Produktentwicklung der Zukunft mithilfe von Verfahren des maschinellen Lernens*, Masterthesis, Karlsruher Institut für Technologie, Karlsruhe, wbk Institut für Produktionstechnik.

A_Schmutz 2021

Schmutz, T. (2021), *Machine Learning zur Klassifizierung und Ähnlichkeitsanalyse von 3D Objekten*, Bachelorthesis, Karlsruher Institut für Technologie, Karlsruhe, wbk Institut für Produktionstechnik.

A_Spuler 2020

Spuler, K. (2020), *Machine Learning im Produktentstehungsprozess – Konzeptionierung und Ausarbeitung eines intelligenten Konstruktionsassistenten und Implementierung neuer Klassifizierungsalgorithmen*, Bachelorthesis, Karlsruher Institut für Technologie, Karlsruhe, wbk Institut für Produktionstechnik.

Achlioptas et al. 2018

Achlioptas, P.; Diamanti, O.; Mitliagkas, I. & Guibas, L. J. (2018), „Learning Representations and Generative Models for 3D Point Clouds“, *Proceedings of the 35th International Conference on Machine Learning (ICML)*, Hrsg. J. Dy & A. Krause, 35th International Conference on Machine Learning (ICML), 10.07.-15.07.2018, Stockholm, Schweden. *Proceedings of Machine Learning Research (PMLR)*, S. 40–49.

Aggarwal 2015

Aggarwal, C. C. (2015), *Data Mining : The Textbook*, Springer, Cham. ISBN: 9783319141428.

Aggarwal 2021

Aggarwal, C. C. (2021), *Artificial intelligence. A textbook*, Springer Nature Switzerland AG, Cham. ISBN: 9783030723569.

Ahmed et al. 2018

Ahmed, E.; Saint, A.; Shabayek, A. E. R.; Cherenkova, K.; Das, R.; Gusev, G.; Aouada, D. & Ottersten, B. (2018), „A survey on Deep Learning Advances on Different 3D Data Representations“, *arXiv preprint*. arXiv-ID: 1808.01462.

Ahmed, Wallace & Blessing 2003

Ahmed, S.; Wallace, K. M. & Blessing, L. T. (2003), „Understanding the differences between how novice and experienced designers approach design tasks“, *Research in Engineering Design*, 14(1), S. 1–11.

Albers et al. 2014

Albers, A.; Lüdcke, R.; Bursac, N. & Reiss, N. (2014), „Connecting knowledge-management-systems to improve a continuous flow of engineering design processes“, *Proceedings of the 10th International Symposium on Tools and Methods of Competitive Engineering - TMCE 2014*, Hrsg. I. Horváth & Z. Rusak, TMCE 2014: Tools and Methods of Competitive Engineering - Sustainability and Cyber-Physical Systems, 19.05.-23.05.2014, Budapest, Ungarn. University of Technology, Budapest, S. 393–402.

Albers et al. 2014

Albers, A.; Reiß, N.; Bursac, N.; Urbanec, J. & Lüdcke, R. (2014), „Situation-appropriate method selection in product development process – empirical study of method application“, *DS 81: Proceedings of NordDesign 2014*, Hrsg. M. Laakso & K. Ekman, NordDesign 2014, 27.08.-29.08.2014, Espoo, Finnland. Aalto Design Factory, Aalto, S. 550–559.

Albers et al. 2014

Albers, A.; Bursac, N.; Urbanec, J.; Lüdcke, R. & Rachenkova, G. (2014), „Knowledge Management in Product Generation Development – an empirical study“, *DFX 2014: Proceedings of the 24th Symposium Design for X*, Hrsg. D. Krause, K. Paetzold & S. Wartzack, 24. DfX-Symposium 2014, 01.10.-02.10.2014, Bamberg, Deutschland. TuTech Verlag, Hamburg, S. 13–24.

Albers et al. 2016

Albers, A.; Reiss, N.; Bursac, N. & Richter, T. (2016), „iPeM – Integrated Product Engineering Model in Context of Product Generation Engineering“, *Procedia CIRP*, 50, S. 100–105.

Albers et al. 2022

Albers, A.; Lanza, G.; Klippert, M.; Schäfer, L.; Frey, A.; Hellweg, F.; Müller-Welt, P.; Schöck, M.; Krahe, C.; Nowoseltschenko, K. & Rapp, S. (2022), „Product-Production-CoDesign: An Approach on Integrated Product and Production Engineering Across Generations and Life Cycles“, *Procedia CIRP*, 109, S. 167–172.

Albers, Börsting & Turki 2011

Albers, A.; Börsting, P. & Turki, T. (2011), „Application of Design Patterns for the Development of Primary Shaped Microsystems: A Case Study“, *Proceedings of the 7th International Conference on Multi-Material Micro Manufacture*, Hrsg. B. Filion, C. Khan Malek & S. Dimov, 7th International Conference on Multi-Material Micro Manufacture, 17.11.-19.11.2010, Bourg en Bresse, Oyonnax, Frankreich. Research Publishing Services, Singapur, S. 235–238. ISBN: 9789810865559.

Albers, Bursac & Wintergerst 2015

Albers, A.; Bursac, N. & Wintergerst, E. (2015), „Produktgenerationsentwicklung – Bedeutung und Herausforderungen aus einer entwicklungsmethodischen Perspektive“, *Stuttgarter Symposium für Produktentwicklung (SSP)*, Hrsg. H. Binz, B. Bertsche, W. Bauer & D. Roth, Stuttgarter Symposium für Produktentwicklung (SSP), 19.06.2015, Stuttgart. Fraunhofer Verlag, Stuttgart, S. 1–10.

Albers, Deigendesch & Turki 2009

Albers, A.; Deigendesch, T. & Turki, T. (2009), „Design Patterns in Microtechnology“, *DS 58-5: Proceedings of ICED 09, the 17th International Conference on Engineering Design*, Hrsg. N. Bergendahl, M. Grimheden, L. Leifer, P. Skogstad & U. Lindemann, 17th International Conference on Engineering Design, 24.08.-27.08.2009, Palo Alto, CA, USA. The Design Society, Glasgow, S. 385–396.

Albers & Turki 2013

Albers, A. & Turki, T. (2013), „Supporting design of primary shaped micro parts and systems through provision of experience“, *Microsystem Technologies*, 19(3), S. 471–476.

Aljalbout et al. 2018

Aljalbout, E.; Golkov, V.; Siddiqui, Y.; Strobel, M. & Cremers, D. (2018), „Clustering with Deep Learning: Taxonomy and New Methods“, *arXiv preprint*. arXiv-ID: 1801.07648.

Arvanitidis, Hansen & Hauberg 2018

Arvanitidis, G.; Hansen, L. K. & Hauberg, S. (2018), „Latent Space Oddity: on the Curvature of Deep Generative Models“, *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, 6th International Conference on Learning Representations, ICLR 2018, 30.04.-03.05.2018, Vancouver, Kanada. OpenReview.

Bai et al. 2010

Bai, J.; Gao, S.; Tang, W.; Liu, Y. & Guo, S. (2010), „Design reuse oriented partial retrieval of CAD models“, *Computer-Aided Design*, 42(12), S. 1069–1084.

Barrow et al. 1977

Barrow, H. G.; Tenenbaum, J. M.; Bolles, R. C. & Wolf, H. C. (1977), „Parametric Correspondence and Chamfer Matching: Two New Techniques for Image Matching“, *Proceedings of the 5th International Joint Conference on Artificial Intelligence - Volume 2*, 5th International Joint Conference on Artificial Intelligence (IJCAI), 22.08.-25.08.1977, Cambridge, USA. Morgan Kaufmann Publishers Inc, San Francisco, S. 659–663.

Bauernhansl 2014

Bauernhansl, T. (2014), „Die Vierte Industrielle Revolution – Der Weg in ein wertschaffendes Produktionsparadigma“ in *Industrie 4.0 in Produktion, Automatisierung und Logistik. Anwendung, Technologien, Migration*, Hrsg. T. Bauernhansl, M. ten Hompel & B. Vogel-Heuser. Springer Vieweg, Wiesbaden, S. 5–35.

Bengio, Courville & Vincent 2013

Bengio, Y.; Courville, A. & Vincent, P. (2013), „Representation Learning: A Review and New Perspectives“, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8), S. 1798–1828.

Bickel et al. 2021

Bickel, S.; Sauer, C.; Schleich, B. & Wartzack, S. (2021), „Comparing CAD part models for geometrical similarity: A concept using machine learning algorithms“, *Procedia CIRP*, 96(1), S. 133–138.

Boothroyd 1994

Boothroyd, G. (1994), „Product design for manufacture and assembly“, *Computer-Aided Design*, 26(7), S. 505–520.

Breitsprecher et al. 2015

Breitsprecher, T.; Kestel, P.; Küster, C.; Sprügel, T. & Wartzack, S. (2015), „Einsatz von Data-Mining in modernen Produktentstehungsprozessen“, *Zeitschrift für wirtschaftlichen Fabrikbetrieb*, 110(11), S. 744–750.

Breitsprecher & Wartzack 2012

Breitsprecher, T. & Wartzack, S. (2012), „Architecture and realization of a selflearning engineering assistance system for the use within sheetbulk metal forming“, *DS 71: Proceedings of NordDesign 2012*, Hrsg. P. K. Hansen, J. Rasmussen, K. A. Jørgensen & C. Tollestrup, The 9th NordDesign conference, 22.08.-24.08.2012, Aarlborg, Dänemark. The Design Society, Glasgow. ISBN: 9788791831515.

Chang et al. 2015

Chang, A. X.; Funkhouser, T.; Guibas, L. J.; Hanrahan, P.; Huang, Q.; Li, Z.; Savarese, S.; Savva, M.; Song, S.; Su, H.; Xiao, J.; Yi, L. & Yu, F. (2015), „ShapeNet: An Information-Rich 3D Model Repository“, *Computing Research Repository (CoRR)*, abs/1512.03012. arXiv-ID: 1512.03012.

Chapman et al. 2000

Chapman, P.; Clinton, J.; Kerber, R.; Khabaza, T.; Reinartz, T. P.; Shearer, C. & Wirth, R. (2000), „CRISP-DM 1.0: Step-by-step data mining guide“, *SPSS inc*, 9(13), S. 1–73.

Cho et al. 2014

Cho, K.; van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H. & Bengio, Y. (2014), „Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation“, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Hrsg. A. Moschitti & W. Daelemans, 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 25.10.-29.10.2014, Doha, Qatar. Association for Computational Linguistics, Stroudsburg, S. 1724–1734.

Cho et al. 2014

Cho, K.; van Merriënboer, B.; Bahdanau, D. & Bengio, Y. (2014), „On the Properties of Neural Machine Translation: Encoder–Decoder Approaches“, *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, Hrsg. D. Wu, M. Carpuat, X. Carreras & E. Maria Vecchi, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, 25.10.2014, Doha, Qatar. Association for Computational Linguistics, Stroudsburg, S. 103–111.

Chollet 2018

Chollet, F. (2018), *Deep learning mit Python und Keras. Das Praxis-Handbuch vom Entwickler der Keras-Bibliothek*, MITP Verlags GmbH & Co. KG, Frechen. ISBN: 3958458408.

Chung et al. 2014

Chung, J.; Gulcehre, C.; Cho, K. & Bengio, Y. (2014), „Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling“, *arXiv preprint*. arXiv-ID: 1412.3555.

Davies & Bouldin 1979

Davies, D. L. & Bouldin, D. W. (1979), „A Cluster Separation Measure“, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(2), S. 224–227.

Deigendesch, T. 2009

Deigendesch, T. (2009), *Kreativität in der Produktentwicklung und Muster als methodisches Hilfsmittel*. Dissertation, Karlsruher Institut für Technologie (KIT), Karlsruhe.

Düsing 2006

Düsing, R. (2006), „Knowledge Discovery in Databases and Data Mining“ in *Analytische Informationssysteme: Business Intelligence-Technologien und -Anwendungen*, Hrsg. P. Chamoni & P. Gluchowski. Springer, Berlin, Heidelberg, S. 241–262.

Eckert, Alink & Albers 2010

Eckert, C. M.; Alink, T. & Albers, A. (2010), „Issue driven analysis of an existing product at different levels of abstraction“, *DS 60: Proceedings of DESIGN 2010, the 11th International Design Conference*, Hrsg. D. Marjanović, M. Štorga, N. Pavković & N. Bojčević, DESIGN 2010, the 11th International Design Conference, 17.05.-20.05.2010, Dubrovnik, Kroatien. The Design Society, Glasgow, S. 673–682. ISBN: 9789537738037.

Ehrlenspiel et al. 2014

Ehrlenspiel, K.; Kiewert, A.; Lindemann, U. & Mörtl, M. (2014), *Kostengünstig Entwickeln und Konstruieren*, Springer, Berlin, Heidelberg. ISBN: 9783642419584.

Ehrlenspiel & Meerkamm 2017

Ehrlenspiel, K. & Meerkamm, H. (2017), *Integrierte Produktentwicklung*, Carl Hanser Verlag GmbH & Co. KG, München. ISBN: 9783446440890.

Ester et al. 1996

Ester, M.; Kriegel, H.-P.; Sander, J. & Xu, X. (1996), „A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise“, *Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96)*, Hrsg. E. Simoudis, J. Han & U. Fayyad, 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96), 02.08.-04.08.1996, Portland, OR, USA. AAAI Press, Palo Alto, S. 226–231.

Eversheim, Schuh & Assmus 2005

Eversheim, W.; Schuh, G. & Assmus, D. (2005), „Integrierte Produkt- und Prozessgestaltung“ in *Integrierte Produkt- und Prozessgestaltung*, Hrsg. W. Eversheim & G. Schuh. Springer, Berlin, S. 5–19.

Fallböhmer, M. 2000

Fallböhmer, M. (2000), *Generieren alternativer Technologieketten in frühen Phasen der Produktentwicklung*. Dissertation, Technische Hochschule Aachen, Aachen.

Fayyad, Piatetsky-Shapiro & Smyth 1996

Fayyad, U.; Piatetsky-Shapiro, G. & Smyth, P. (1996), „From Data Mining to Knowledge Discovery in Databases“, *AI Magazine*, 17(3), S. 37–54.

Feldhusen & Grote 2013

Feldhusen, J. & Grote, K.-H. (2013), *Pahl/Beitz Konstruktionslehre. Methoden und Anwendung erfolgreicher Produktentwicklung*, Springer Vieweg, Berlin, Heidelberg. ISBN: 9783642295683.

Feng et al. 2019

Feng, Y.; Feng, Y.; You, H.; Zhao, X. & Gao, Y. (2019), „MeshNet: Mesh Neural Network for 3D Shape Representation“, *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI-19)*, 33rd AAAI Conference on Artificial Intelligence (AAAI-19), 27.01.-01.02.2019, Honolulu, Hawaii, USA. AAAI Press, Palo Alto, S. 8279–8286.

Fleischer 2019

Fleischer, B. (2019), *Methodisches Konstruieren in Ausbildung und Beruf*, Springer Fachmedien Wiesbaden, Wiesbaden. ISBN: 9783658276898.

Frenzel, Teleaga & Ushio 2019

Frenzel, M. F.; Teleaga, B. & Ushio, A. (2019), „Latent Space Cartography: Generalised Metric-Inspired Measures and Measure-Based Transformations for Generative Models“, *arXiv preprint*. arXiv-ID: 1902.02113.

Geiger et al. 2013

Geiger, A.; Lenz, P.; Stiller, C. & Urtasun, R. (2013), „Vision meets Robotics: The KITTI Dataset“, *International Journal of Robotics Research (IJRR)*, 32, S. 1229–1235.

Genova et al. 2020

Genova, K.; Cole, F.; Sud, A.; Sarna, A. & Funkhouser, T. (2020), „Local Deep Implicit Functions for 3D Shape“, *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, Hrsg. L. O'Conner, IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 13.07.-19.07.2020, Virtuell. IEEE, Piscataway, S. 4856–4865. ISBN: 9781728171685.

Germani, Mandolini & Cicconi 2011

Germani, M.; Mandolini, M. & Cicconi, P. (2011), „Manufacturing cost estimation during early phases of machine design“, *DS 68-5: Proceedings of the 18th International Conference on Engineering Design (ICED 11), Impacting Society through Engineering Design, Vol. 5: Design for X / Design to X*, Hrsg. S. J. Culley, B. J. Hicks, T. C. McAlone, T. J. Howard & J. Malmqvist, 18th International Conference on Engineering Design ICED11, 15.08.-19.09.2011, Kopenhagen, Dänemark. The Design Society, Glasgow, S. 124–134. ISBN: 9781904670322.

Gezawa et al. 2020

Gezawa, A. S.; Zhang, Y.; Wang, Q. & Yunqi, L. (2020), „A Review on Deep Learning Approaches for 3D Data Representations in Retrieval and Classifications“, *IEEE Access*, 8, S. 57566–57593.

Goodfellow et al. 2014

Goodfellow, I. J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A. & Bengio, Y. (2014), „Generative Adversarial Networks“, *Advances in Neural Information Processing Systems 27 (NIPS 2014)*, Hrsg. Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence & K. Q. Weinberger, 28th Annual Conference on Neural Information Processing Systems 2014, NIPS 2014, 08.12.-13.12.2014, Montreal, Kanada. Neural Information Processing Systems Foundation, Inc. (NeurIPS), San Diego, CA, S. 2672–2680. ISBN: 9781510800410.

Goodfellow, Bengio & Courville 2016

Goodfellow, I.; Bengio, Y. & Courville, A. (2016), *Deep learning*, MIT Press, Cambridge, Massachusetts, London, England. ISBN: 9780262035613.

Graves 2012

Graves, A. (2012), *Supervised Sequence Labelling with Recurrent Neural Networks*, Springer, Berlin, Heidelberg. ISBN: 9783642247972.

Guo et al. 2021

Guo, Y.; Wang, H.; Hu, Q.; Liu, H.; Liu, L. & Bennamoun, M. (2021), „Deep Learning for 3D Point Clouds: A Survey“, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(12), S. 4338–4364.

Hackenschmidt, Hautsch & Kleinschrodt 2020

Hackenschmidt, R.; Hautsch, S. & Kleinschrodt, C. (2020), *Creo Parametric für Einsteiger. Bauteile, Baugruppen und Zeichnungen*, Carl Hanser Fachbuchverlag, München. ISBN: 9783446460478.

Han, Kamber & Pei 2012

Han, J.; Kamber, M. & Pei, J. (2012), *Data mining. Concepts and techniques*, Elsevier/Morgan Kaufmann, Amsterdam. ISBN: 9781283171175.

Hanocka et al. 2019

Hanocka, R.; Hertz, A.; Fish, N.; Giryas, R.; Fleishman, S. & Cohen-Or, D. (2019), „MeshCNN: A Network with an Edge“, *ACM Transactions on Graphics*, 38(4), S. 1–12.

Hassani & Haley 2019

Hassani, K. & Haley, M. (2019), „Unsupervised Multi-Task Feature Learning on Point Clouds“, *Proceedings of the IEEE/CVF International Conference on Computer Vision*, IEEE/CVF International Conference on Computer Vision (ICCV), 27.10.-02.11.2019, Seoul, Südkorea. IEEE, Piscataway, S. 8160–8171.

Hirschle 2020

Hirschle, J. (2020), *Machine Learning für Zeitreihen*, Carl Hanser Verlag GmbH & Co. KG, München. ISBN: 9783446467262.

Hochreiter & Schmidhuber 1997

Hochreiter, S. & Schmidhuber, J. (1997), „Long short-term memory“, *Neural computation*, 9(8), S. 1735–1780.

Hong, Lee & Kim 2006

Hong, T.; Lee, K. & Kim, S. (2006), „Similarity comparison of mechanical parts to reuse existing designs“, *Computer-Aided Design*, 38(9), S. 973–984.

Humpa, M. W. 2016

Humpa, M. W. (2016), *CAD-Methodik zur Produktivitätssteigerung in der Prozess-kette Konstruktion-Fertigung*. Dissertation, Universität Duisburg-Essen, Duisburg-Essen.

ISO 10303-242:2020(E)

ISO 10303-242:2020(E) (2020), *Industrial automation systems and integration - Product data representation and exchange*, International Standard ISO, Beuth Verlag, Vernier.

Iyer et al. 2005

Iyer, N.; Jayanti, S.; Lou, K.; Kalyanaraman, Y. & Ramani, K. (2005), „Three-dimensional shape searching: state-of-the-art review and future trends“, *Computer-Aided Design*, 37(5), S. 509–530.

Jacob, A. 2021

Jacob, A. (2021), *Hochiterative Technologieplanung*. Dissertation, Karlsruher Institut für Technologie (KIT), Shaker Verlag, Düren. ISBN: 9783844081633.

Jaiswal, Huang & Rai 2016

Jaiswal, P.; Huang, J. & Rai, R. (2016), „Assembly-based conceptual 3D modeling with unlabeled components using probabilistic factor graph“, *Computer-Aided Design*, 74, S. 45–54.

Kirchner 2020

Kirchner, E. (2020), *Werkzeuge und Methoden der Produktentwicklung. Von der Idee zum erfolgreichen Produkt*, Springer Vieweg, Berlin. ISBN: 9783662617625.

Koch et al. 2019

Koch, S.; Matveev, A.; Jiang, Z.; Williams, F.; Artemov, A.; Burnaev, E.; Alexa, M.; Zorin, D. & Panozzo, D. (2019), „ABC: A Big CAD Model Dataset For Geometric Deep Learning“, *Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 15.06.-19.06.2019, Long Beach, CA, USA. IEEE, Los Alamitos, CA, S. 9593–9603. ISBN: 9781728132938.

Krahe et al. 2019

Krahe, C.; Iberl, M.; Jacob, A. & Lanza, G. (2019), „AI-based Computer Aided Engineering for automated product design - A first approach with a Multi-View based classification“, *Procedia CIRP*, 86, S. 104–109.

Krahe et al. 2022

Krahe, C.; Marinov, M.; Schmutz, T.; Hermann, Y.; Bonny, M.; May, M. & Lanza, G. (2022), „AI-based geometric similarity search supporting component reuse in engineering design“, *Procedia CIRP*, 109, S. 275–280.

Kratzer et al. 2012

Kratzer, M.; Crostack, A.; Binz, H. & Roth, D. (2012), „Distribution of engineering design knowledge within the development of multi-agent design systems“, *DS 70: Proceedings of International Design Conference, DESIGN*, Hrsg. D. Marjanović, M. Štorga, N. Pavković & N. Bojčetić, 12th International Design Conference, DESIGN 2012, 21.05.-24.05.2012, Dubrovnik, Kroatien. The Design Society, Glasgow, S. 1495–1506. ISBN: 9789537738174.

Kratzer, Binz & Roth 2010

Kratzer, M.; Binz, H. & Roth, D. (2010), „Wissensstruktur zur Integration von Konstruktionswissen in agentenbasierte Unterstützungssysteme“, *DFX 2010: Proceedings of the 21st Symposium on Design for X*, Hrsg. D. Krause, K. Paetzold & S. Wartzack, 21st Symposium on Design for X, DFX 2010, 23.09.-24.09.2010, Buchholz/Hamburg, Deutschland. TuTech Innovation GmbH, Hamburg, S. 235–246. ISBN: 9783941492233.

Kuhn & Johnson 2016

Kuhn, M. & Johnson, K. (2016), *Applied predictive modeling*, Springer, New York. ISBN: 9781461468493.

Küstner, C. 2020

Küstner, C. (2020), *Assistenzsystem zur Unterstützung der datengetriebenen Produktentwicklung*. Dissertation, FAU Erlangen, FAU University Press, Erlangen. ISBN: 9783961473489.

La Rocca 2012

La Rocca, G. (2012), „Knowledge based engineering: Between AI and CAD. Review of a language based technology to support engineering design“, *Advanced Engineering Informatics*, 26(2), S. 159–179.

Li et al. 2019

Li, C.-L.; Zaheer, M.; Zhang, Y.; Poczos, B. & Salakhutdinov, R. (2019), „Point Cloud GAN“, *ICLR 2019 Workshop Deep Generative Models for Highly Structured Data*, ICLR 2019 Workshop, 06.05.-09.05.2019, New Orleans, LA, USA. OpenReview.

Lupinetti et al. 2017

Lupinetti, K.; Chiang, L.; Giannini, F.; Monti, M. & Pernot, J.-P. (2017), „Regular patterns of repeated elements in CAD assembly model retrieval“, *Computer-Aided Design and Applications*, 14(4), S. 516–525.

Lupinetti et al. 2019

Lupinetti, K.; Pernot, J.-P.; Monti, M. & Giannini, F. (2019), „Content-based CAD assembly model retrieval: Survey and future challenges“, *Computer-Aided Design*, 113(5), S. 62–81.

Machalica & Matyjewski 2019

Machalica, D. & Matyjewski, M. (2019), „CAD models clustering with machine learning“, *Archiv of Mechanical Engineering*, 66(2), S. 133–152.

Meerkamm et al. 2012

Meerkamm, H.; Wartzack, S.; Bauer, S.; Krehmer, H.; Stockinger, A. & Walter, M. (2012), „Design for X (Dfx)“ in *Handbuch Konstruktion*, Hrsg. F. Rieg & R. Steinhilper. Carl Hanser Verlag GmbH & Co. KG, München, S. 443–462.

Mescheder et al. 2019

Mescheder, L.; Oechsle, M.; Niemeyer, M.; Nowozin, S. & Geiger, A. (2019), „Occupancy Networks: Learning 3D Reconstruction in Function Space“, *Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 15.06.-19.06.2019, Long Beach, CA, USA. IEEE, Los Alamitos, CA. ISBN: 9781728132938.

Mo et al. 2019

Mo, K.; Zhu, S.; Chang, A. X.; Yi, L.; Tripathi, S.; Guibas, L. J. & Su, H. (2019), „PartNet: A Large-scale Benchmark for Fine-grained and Hierarchical Part-level 3D Object Understanding“, *Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 15.06.-19.06.2019, Long Beach, CA, USA. IEEE, Los Alamitos, CA, S. 909–918. ISBN: 9781728132938.

Mohamad & Usman 2013

Mohamad, I. B. & Usman, D. (2013), „Standardization and Its Effects on K-Means Clustering Algorithm“, *Research Journal of Applied Sciences, Engineering and Technology*, 6(17), S. 3299–3303.

Nikolić 2012

Nikolić, M. (2012), „Measuring similarity of graph nodes by neighbor matching“, *Intelligent Data Analysis*, 16(6), S. 865–878.

Orth, Voigt & Kohl 2011

Orth, R.; Voigt, S. & Kohl, I. (2011), *Praxisleitfaden Wissensmanagement. Prozessorientiertes Wissensmanagement nach dem ProWis-Ansatz einführen*, Fraunhofer-Verlag, Stuttgart. ISBN: 9783839603062.

Otte et al. 2020

Otte, R.; Wippermann, B.; Schade, S. & Otte, V. (2020), *Von Data Mining bis Big Data. Handbuch für die industrielle Praxis inklusive Small Data und Mind Data*, Carl Hanser Verlag, München. ISBN: 9783446457171.

Pang, Li & Tian 2021

Pang, J.; Li, D. & Tian, D. (2021), „TearingNet: Point Cloud Autoencoder to Learn Topology-Friendly Representations“, *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 20.06.-25.06.2021, Nashville, TN, USA. IEEE, Piscataway, S. 7449–7458. ISBN: 9781665445092.

Park et al. 2019

Park, J. J.; Florence, P.; Straub, J.; Newcombe, R. & Lovegrove, S. (2019), „DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation“, *Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 15.06.-19.06.2019, Long Beach, CA, USA. IEEE, Los Alamitos, CA, S. 165–174. ISBN: 9781728132938.

Peddireddy et al. 2020

Peddireddy, D.; Fu, X.; Wang, H.; Joung, B. G.; Aggarwal, V.; Sutherland, J. W. & Byung-Guk Jun, M. (2020), „Deep Learning Based Approach for Identifying Conventional Machining Processes from CAD Data“, *Procedia Manufacturing*, 48, S. 915–925.

Peste, Malagó & Sârbu 2017

Peste, A.; Malagó, L. & Sârbu, S. (2017), „An Explanatory Analysis of the Geometry of Latent Variables Learned by Variational Auto-Encoders“, *NIPS 2017 Workshop on Bayesian Deep Learning (2017)*, NIPS 2017 Workshop on Bayesian Deep Learning (2017), 09.12.2017, Long Beach, CA, USA. Neural Information Processing Systems Foundation, Inc. (NeurIPS), San Diego, CA.

Pham & Afify 2007

Pham, D. T. & Afify, A. A. (2007), „Clustering techniques and their applications in engineering“, *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 221(11), S. 1445–1459.

Qi et al. 2017

Qi, C. R.; Su, H.; Mo, K. & Guibas, L. J. (2017), „PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation“, *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Hrsg. IEEE Computer Society, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 21.07.-26.07.2017, Honolulu, Hawaii, USA. IEEE, Piscataway, S. 77–85. ISBN: 9781538604588.

Qi et al. 2017

Qi, C. R.; Yi, L.; Su, H. & Guibas, L. J. (2017), „PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space“, *Proceedings of the 31st International Conference on Neural Information Processing Systems*, Hrsg. U. von Luxburg, I. Guyon, S. Bengio, H. Wallach & R. Fergus, 31st Conference on Neural Information Processing Systems (NIPS 2017), 04.12.-09.12.2017, Long Beach, CA, USA. Curran Associates Inc, Red Hook, New York, S. 5105–5114. ISBN: 9781510860964.

Rea et al. 2002

Rea, H. J.; Corney, J. R.; Clark, D. E. R.; Pritchard, J.; Breaks, M. L. & Macleod, R. A. (2002), „Part-sourcing in a Global Market“, *Concurrent Engineering*, 10(4), S. 325–333.

Ren, Niu & Fang 2017

Ren, M.; Niu, L. & Fang, Y. (2017), „3D-A-Nets: 3D Deep Dense Descriptor for Volumetric Shapes with Adversarial Networks“, *Computing Research Repository (CoRR)*, abs/1711.10108. arXiv-ID: 1711.10108.

Röhner, Breitsprecher & Wartzack 2011

Röhner, S.; Breitsprecher, T. & Wartzack, S. (2011), „Acquisition of design-relevant knowledge within the development of sheet-bulk metal forming“, *DS 68-6: Proceedings of the 18th International Conference on Engineering Design (ICED 11), Impacting Society through Engineering Design, Vol. 6: Design Information and Knowledge*, Hrsg. S. J. Culley, B. J. Hicks, T. C. McAloone, T. J. Howard & W. Chen, 18th International Conference on Engineering Design (ICED 11), 15.08.-19.08.2011, Kopenhagen, Dänemark. The Design Society, Glasgow, S. 108–120. ISBN: 9781904670261.

Roj et al. 2021

Roj, R.; Sommer, M.; Woyand, H.-B.; Theiss, R. & Dueltgen, P. (2021), „Classification of CAD-Models Based on Graph Structures and Machine Learning“, *Computer-Aided Design and Applications*, 19(3), S. 449–469.

Roj, R. 2016

Roj, R. (2016), *Eine Methode für eine automatisierte Informationsextraktion aus großen CAD-Datenbeständen zur Bauteilsuche und Klassifikation*. Dissertation, Bergische Universität Wuppertal, Wuppertal.

Rosenberg & Hirschberg 2007

Rosenberg, A. & Hirschberg, J. (2007), „V-Measure: A Conditional Entropy-Based External Cluster Evaluation Measure“, *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, Hrsg. J. Eisner, Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL) 2007, 28.06.-30.06.2007, Prag, Tschechische Republik. Association for Computational Linguistics, Stroudsburg, S. 410–420.

Rousseeuw 1987

Rousseeuw, P. J. (1987), „Silhouettes: A graphical aid to the interpretation and validation of cluster analysis“, *Journal of Computational and Applied Mathematics*, 20, S. 53–65.

Rubner, Tomasi & Guibas 2004

Rubner, Y.; Tomasi, C. & Guibas, L. J. (2004), „The Earth Mover’s Distance as a Metric for Image Retrieval“, *International Journal of Computer Vision*, 40, S. 99–121.

Russell & Norvig 2022

Russell, S. J. & Norvig, P. (2022), *Artificial intelligence. A modern approach*, Pearson, Harlow. ISBN: 9781292401133.

Samek & Müller 2019

Samek, W. & Müller, K.-R. (2019), „Towards Explainable Artificial Intelligence“ in *Explainable AI: interpreting, explaining and visualizing deep learning*, Hrsg. W. Samek, G. Montavon, A. Vedaldi, L. K. Hansen & K.-R. Müller. Springer International Publishing, Cham, S. 5–22.

Sarmad, Lee & Kim 2019

Sarmad, M.; Lee, H. J. & Kim, Y. M. (2019), „RL-GAN-Net: A Reinforcement Learning Agent Controlled GAN Network for Real-Time Point Cloud Shape Completion“, *Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 15.06.-19.06.2019, Long Beach, CA, USA. IEEE, Los Alamitos, CA, S. 5891–5900. ISBN: 9781728132938.

Schindler, S. 2014

Schindler, S. (2014), *Strategische Planung von Technologieketten für die Produktion*. Dissertation, Technische Universität München, Utz Verlag, München. ISBN: 9783831671052.

Schönhof & Fechter 2020

Schönhof, R. G.C. & Fechter, M. (2020), „Towards automated Capability Assessment leveraging Deep Learning“, *Procedia CIRP*, 91, S. 433–438.

Simonovsky & Komodakis 2017

Simonovsky, M. & Komodakis, N. (2017), „Dynamic Edge-Conditioned Filters in Convolutional Neural Networks on Graphs“, *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Hrsg. IEEE Computer Society, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 21.07.-26.07.2017, Honolulu, Hawaii, USA. IEEE, Piscataway, S. 29–38. ISBN: 9781538604588.

Spinner et al. 2018

Spinner, T.; Körner, J.; Görtler, J. & Deussen, O. (2018), „Towards an Interpretable Latent Space : an Intuitive Comparison of Autoencoders with Variational Autoencoders“, *Proceedings of the Workshop on Visualization for AI Explainability 2018 (VISxAI)*, IEEE VIS 2018, 21.10.-26.10.2018, Berlin, Deutschland. IEEE, Piscataway.

Stark 2022

Stark, R. (2022), *Virtual Product Creation in Industry. The Difficult Transformation from IT Enabler Technology to Core Engineering Competence*, Springer, Berlin, Heidelberg. ISBN: 9783662643013.

Steinley & Brusco 2007

Steinley, D. & Brusco, M. J. (2007), „Initializing K-means Batch Clustering: A Critical Evaluation of Several Techniques“, *Journal of Classification*, 24(1), S. 99–121.

Stokes 2001

Stokes, M. (Hrsg.) (2001), *Managing engineering knowledge. MOKA: methodology for knowledge based engineering applications*, John Wiley and Sons Ltd, Hoboken. ISBN: 1860582958.

Studer et al. 2021

Studer, S.; Bui, T. B.; Drescher, C.; Hanuschkin, A.; Winkler, L.; Peters, S. & Muel-ler, K.-R. (2021), „Towards CRISP-ML(Q): A Machine Learning Process Model with Quality Assurance Methodology“, *Machine Learning & Knowledge Extraction*, 3(2), S. 392–413.

Su et al. 2015

Su, H.; Maji, S.; Kalogerakis, E. & Learned-Miller, E. (2015), „Multi-View Convoluti-onal Neural Networks for 3D Shape Recognition“, *Proceedings of the IEEE Inter-national Conference on Computer Vision*, Hrsg. L. O’Conner, IEEE International Conference on Computer Vision (ICCV) 2015, 07.12.-13.12.2015, Santiago, Chile. IEEE, Piscataway, S. 945–953. ISBN: 9781467383912.

Sun et al. 2021

Sun, W.; Tagliasacchi, A.; Deng, B.; Sabour, S.; Yazdani, S.; Hinton, G. & Yi, K. M. (2021), „Canonical Capsules: Unsupervised Capsules in Canonical Pose“, *Advan-ces in Neural Information Processing Systems*, 30, S. 24993–25005.

Tasse & Dodgson 2016

Tasse, F. P. & Dodgson, N. (2016), „Shape2Vec“, *ACM Transactions on Graphics*, 35(6), S. 1–12.

Tayyub et al. 2021

Tayyub, J., et al. (2021), *AIAX CAD Dataset*, Karlsruher Institut für Technologie (KIT), Endress+Hauser, Karlsruhe, Maulburg. DOI: 10.35097/420.

Tchapmi et al. 2019

Tchapmi, L. P.; Kosaraju, V.; Rezatofighi, H.; Reid, I. & Savarese, S. (2019), „TopNet: Structural Point Cloud Decoder“, *Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019 IEEE/CVF Con-ference on Computer Vision and Pattern Recognition (CVPR), 15.06.-19.06.2019, Long Beach, CA, USA. IEEE, Los Alamitos, CA, S. 383–392. ISBN: 9781728132938.

Thorndike 1953

Thorndike, R. L. (1953), „Who belongs in the family?“, *Psychometrika*, 18(4), S. 267–276.

Tsai & Chang 2005

Tsai, C.-Y. & Chang, C. A. (2005), „A two-stage fuzzy approach to feature-based design retrieval“, *Computers in Industry*, 56(5), S. 493–505.

Turki, T. 2014

Turki, T. (2014), *Bedeutung von Erfahrungswissen in der Produktentwicklung und Ansätze zu dessen Evaluierung und Transfer am Beispiel studentischer Gruppen*. Dissertation, Karlsruher Institut für Technologie (KIT), Karlsruhe.

Urbach, Ben-Shabat & Lindenbaum 2020

Urbach, D.; Ben-Shabat, Y. & Lindenbaum, M. (2020), „DPDist : Comparing Point Clouds Using Deep Point Cloud Distance“, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Hrsg. A. Vedaldi, H. Bischof, T. Brox & J. Frahm, 16th European Conference on Computer Vision (ECCV) 2020, 23.08.-28.08.2020, Glasgow, UK. Springer Science and Business Media Deutschland GmbH, Berlin, S. 545–560. ISBN: 9783030585471.

Vajna et al. 2018

Vajna, S.; Weber, C.; Zeman, K.; Hehenberger, P.; Gerhard, D. & Wartzack, S. (2018), *CAX für Ingenieure*, Springer, Berlin, Heidelberg. ISBN: 9783662546239.

van der Maaten & Hinton 2008

van der Maaten, L. & Hinton, G. (2008), „Visualizing data using t-SNE“, *Journal of Machine Learning Research*, 9(86), S. 2579–2605.

Vaswani et al. 2017

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, u. & Polosukhin, I. (2017), „Attention is All You Need“, *Proceedings of the 31st International Conference on Neural Information Processing Systems*, Hrsg. U. von Luxburg, I. Guyon, S. Bengio, H. Wallach & R. Fergus, 31st Conference on Neural Information Processing Systems (NIPS 2017), 04.12.-09.12.2017, Long Beach, CA, USA. Curran Associates Inc, Red Hook, New York, S. 6000–6010. ISBN: 9781510860964.

VDI 2221-1:2019-11

VDI 2221-1:2019-11 (2019), *Entwicklung technischer Produkte und Systeme*, Verein Deutscher Ingenieure e.V., Beuth Verlag, Berlin.

VDI 2221-2:2019-11

VDI 2221-2:2019-11 (2019), *Entwicklung technischer Produkte und Systeme*, Verein Deutscher Ingenieure e.V., Beuth Verlag, Berlin.

Wagner & Schwanecke 2022

Wagner, N. & Schwanecke, U. (2022), „NeuralQAAD: An Efficient Differentiable Framework for Compressing High Resolution Consistent Point Clouds Datasets“, *Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, Hrsg. G. M. Farinella, P. Radeva & K. Bouatouch, 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP), 06.02.-08.02.2022, Virtuell. Science and Technology Publications (SciTePress), Setúbal, S. 811–822. ISBN: 9789897585555.

Wang et al. 2020

Wang, Y.; Tan, D. J.; Navab, N. & Tombari, F. (2020), „SoftPoolNet: Shape Descriptor for Point Cloud Completion and Classification“, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Hrsg. A. Vedaldi, H. Bischof, T. Brox & J. Frahm, 16th European Conference on Computer Vision (ECCV) 2020, 23.08.-28.08.2020, Glasgow, UK. Springer Science and Business Media Deutschland GmbH, Berlin, S. 70–85. ISBN: 9783030585471.

Wartzack, Sauer & Küstner 2017

Wartzack, S.; Sauer, C. & Küstner, C. (2017), „What does Design for Production mean? – From Design Guidelines to Self-learning Engineering Workbenches“, *Proceedings of the 11th International Workshop on Integrated Design Engineering*, Hrsg. A. Meyer, R. Schirmeyer & S. Vajna, 11th International Workshop on Integrated Design Engineering, 05.04.-07.04.2017, Magdeburg, Deutschland. Universität Magdeburg Lehrstuhl für Maschinenbauinformatik, Magdeburg, S. 93–102.

Wartzack, S. 2001

Wartzack, S. (2001), *Predictive Engineering - Assistenzsystem zur multikriteriellen Analyse alternativer Produktkonzepte*. Dissertation, Universität Erlangen-Nürnberg, VDI-Verlag, Düsseldorf. ISBN: 3183336014.

Willis et al. 2021

Willis, K. D. D.; Pu, Y.; Luo, J.; Chu, H.; Du, T.; Lambourne, J. G.; Solar-Lezama, A. & Matusik, W. (2021), „Fusion 360 Gallery: A Dataset and Environment for Programmatic CAD Construction from Human Design Sequences“, *ACM Transactions on Graphics*, 40(4), S. 1–24.

Wu et al. 2016

Wu, J.; Zhang, C.; Xue, T.; Freeman, W. T. & Tenenbaum, J. B. (2016), „Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling“, *Advances in Neural Information Processing Systems*, Hrsg. R. Garnett, D. D. Lee, U. von Luxburg, I. Guyon & M. Sugiyama, 30th Annual Conference on Neural Information Processing Systems, NIPS 2016, 05.12.-10.12.2016, Barcelona, Spanien. Neural Information Processing Systems Foundation, Inc. (NeurIPS), San Diego, CA, S. 82–90.

Wu et al. 2020

Wu, R.; Chen, X.; Zhuang, Y. & Chen, B. (2020), „Multimodal Shape Completion via Conditional Generative Adversarial Networks“, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Hrsg. A. Vedaldi, H. Bischof, T. Brox & J. Frahm, 16th European Conference on Computer Vision (ECCV) 2020, 23.08.-28.08.2020, Glasgow, UK. Springer Science and Business Media Deutschland GmbH, Berlin, S. 281–296. ISBN: 9783030585471.

Wu et al. 2020

Wu, R.; Zhuang, Y.; Xu, K.; Zhang, H. & Chen, B. (2020), „PQ-NET: A Generative Part Seq2Seq Network for 3D Shapes“, *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, Hrsg. L. O’Conner, IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 13.07.-19.07.2020, Virtuell. IEEE, Piscataway, S. 826–835. ISBN: 9781728171685.

Wu, Xiao & Zheng 2021

Wu, R.; Xiao, C. & Zheng, C. (2021), „DeepCAD: A Deep Generative Network for Computer-Aided Design Models“, *Proceedings of the IEEE International Conference on Computer Vision*, 18th IEEE/CVF International Conference on Computer Vision, ICCV 2021, 11.10.-17.10.2021, Virtuell. IEEE, Los Alamitos, CA, S. 6752–6762. ISBN: 9781665428125.

Xiao et al. 2020

Xiao, Y.-P.; Lai, Y.-K.; Zhang, F.-L.; Li, C. & Gao, L. (2020), „A survey on deep geometry learning: From a representation perspective“, *Computational Visual Media*, 6(2), S. 113–133.

Xie et al. 2020

Xie, H.; Yao, H.; Zhou, S.; Mao, J.; Zhang, S. & Sun, W. (2020), „GRNet: Gridding Residual Network for Dense Point Cloud Completion“, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Hrsg. A. Vedaldi, H. Bischof, T. Brox & J. Frahm, 16th European Conference on Computer Vision (ECCV) 2020, 23.08.-28.08.2020, Glasgow, UK. Springer Science and Business Media Deutschland GmbH, Berlin, S. 365–381. ISBN: 9783030585471.

Yang et al. 2017

Yang, B.; Fu, X.; Sidiropoulos, N. D. & Hong, M. (2017), „Towards K-means-friendly Spaces: Simultaneous Deep Learning and Clustering“, *34th International Conference on Machine Learning, ICML 2017*, 34th International Conference on Machine Learning, ICML 2017, 06.08.-11.08.2017, Sydney, Australien. International Machine Learning Society (IMLS), S. 5888–5901. ISBN: 9781510855144.

Yang et al. 2018

Yang, Y.; Feng, C.; Shen, Y. & Tian, D. (2018), „FoldingNet: Point Cloud Auto-encoder via Deep Grid Deformation“, *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 18.06.-22.06.2018, Salt Lake City, USA. IEEE, Los Alamitos, CA, S. 206–215. ISBN: 9781538664209.

Yoo et al. 2021

Yoo, S.; Lee, S.; Kim, S.; Hwang, K. H.; Park, J. H. & Kang, N. (2021), „Integrating Deep Learning into CAD/CAE System: Generative Design and Evaluation of 3D Conceptual Wheel“, *Structural and Multidisciplinary Optimization*, 64(4), S. 2725–2747.

Yuan et al. 2018

Yuan, W.; Khot, T.; Held, D.; Mertz, C. & Hebert, M. (2018), „PCN: Point Completion Network“, *Proceedings - 2018 International Conference on 3D Vision, 3DV 2018*, 6th International Conference on 3D Vision, 3DV 2018, 05.09.-08.09.2018, Verona, Italien. IEEE, Piscataway, S. 728–737. ISBN: 9781538684252.

Zhang, Jaiswal & Rai 2018

Zhang, Z.; Jaiswal, P. & Rai, R. (2018), „FeatureNet: Machining feature recognition based on 3D Convolution Neural Network“, *Computer-Aided Design*, 101(6), S. 12–22.

Zou et al. 2017

Zou, C.; Yumer, E.; Yang, J.; Ceylan, D. & Hoiem, D. (2017), „3D-PRNN: Generating Shape Primitives with Recurrent Neural Networks“, *Proceedings of the IEEE International Conference on Computer Vision*, 16th IEEE International Conference on Computer Vision, ICCV 2017, 22.10.-29.10.2017, Venedig, Italien. IEEE, Piscataway, New Jersey, S. 900–909. ISBN: 9781538610329.

Abbildungsverzeichnis

Abbildung 2.1: Modell der Produktentwicklung	6
Abbildung 2.2: Parametermindmap möglicher Produkthanforderungen (in Anlehnung an Jacob (2021), Fallböhmer (2000) und Schindler (2014))	10
Abbildung 2.3: Aufbau eines CAD-Modells eines Endproduktes bestehend aus mehreren Baugruppen, Einzelteilen und Konstruktionselementen (a) sowie beispielhafter Strukturbaum in PTC Creo® Parametric (b)	13
Abbildung 2.4: Ablauf des CRISP-DM (basierend auf Chapman et al. (2000))	17
Abbildung 2.5: Übersicht über die Bereiche der Künstlichen Intelligenz mit beispielhaften Technologien (in Anlehnung an Goodfellow, Bengio & Courville (2016))	18
Abbildung 2.6: Grundsätzlicher Aufbau eines Feed-Forward-Netzes (basierend auf Aggarwal (2021))	20
Abbildung 2.7: Aufbau eines Autoencoders mit drei verborgenen Schichten (a) sowie die allgemeine schematische Darstellung (b) (basierend auf Aggarwal (2021))	22
Abbildung 2.8: Beispielhafte Darstellung eines Kernpunktes (A), Randpunktes (B) und Rauschpunktes (C)	27
Abbildung 2.9: Schematischer Aufbau eines RNNs in kompakter Darstellung (a) und als Berechnungsgraph (b) (in Anlehnung an Graves (2012), Goodfellow, Bengio & Courville (2016) und Russell & Norvig (2022))	30
Abbildung 2.10: Berechnungen innerhalb einer RNN-Zelle (in Anlehnung an Goodfellow, Bengio & Courville (2016))	31
Abbildung 2.11: Schematische Darstellung der Verlustberechnung (in Anlehnung an Goodfellow, Bengio & Courville (2016))	31
Abbildung 2.12: Taxonomie aktueller 3D-Repräsentationsformen mit beispielhaften Vertretern (in Anlehnung an Gezawa et al. (2020))	33
Abbildung 3.1: Übersicht Stand der Technik	50
Abbildung 4.1: Grundsätzliche Vorgehensweise zur Wissensextraktion und Funktionalität der darauf basierenden Konstruktionsassistenten	53

Abbildung 4.2: Überblick und Struktur der entwickelten Methodik	54
Abbildung 4.3: UML Klassendiagramm der drei Datenkategorien	57
Abbildung 4.4: Vorgehen zur Bestimmung der vorhandenen Datenkategorien	58
Abbildung 4.5: Darstellung des Konstruktionsvorgehens über die Sequenz der KEs (1) und die geometrische Entwicklung (2)	61
Abbildung 4.6: Reifegradmodell mit Umsetzungsstufen in Abhängigkeit der anwendungsfallspezifischen Datenverfügbarkeit	63
Abbildung 4.7: Vorverarbeitungsschritte für die 3D-Geometrie (in Anlehnung an A_Schmutz (2021))	66
Abbildung 4.8: Beispielhaftes One-Hot-Encoding des kategorischen Attributs „Klasse“	66
Abbildung 4.9: Aufbereitung der Konstruktionssequenzen	68
Abbildung 4.10: Vorgehensweise zur Bildung von Sequenzabschnitten über ein Sliding Window	69
Abbildung 4.11: Prinzipielle Vorgehensweise zur Extraktion relevanter Eigenschaften aus den Punktwolken mittels Autoencoder	71
Abbildung 4.12: Grundsätzlicher Aufbau und Funktionalität des Autoencoders für Punktwolken	72
Abbildung 4.13: Prinzip der Standardorientierung (in Anlehnung an Krahe et al. (2019))	75
Abbildung 4.14: Architektur des Autoencoders mit zusätzlichem Klassifikationsverlust	76
Abbildung 4.15: Darstellung des Konstruktionsvorgehens in Form der Sequenz der 3D-Geometrie über die Sequenz entsprechender latenter Vektoren (LV)	78
Abbildung 4.16: Prinzip der linearen Interpolation zwischen zwei latenten Vektoren (a) sowie algebraischer Operationen auf latenten Vektoren (b)	80
Abbildung 4.17: Exemplarische Darstellung der vorliegenden Daten je Komponente	81
Abbildung 4.18: Vorgehen zum Auffinden ähnlicher Komponenten	83

Abbildung 4.19: Clustern der vorhandenen Datenbasis in Form der latenten Vektoren	84
Abbildung 4.20: Bestimmung der Menge relevanter Clusterzentren für eine Input-Komponente i durch Berechnung der Distanzen von i zu allen Clusterzentren	85
Abbildung 4.21: Bestimmung der Distanzen der Input-Komponenten i zu allen Komponenten j aus den Clustern mit relevanten Clusterzentren	86
Abbildung 4.22: Bestimmung der Menge der ähnlichsten Modelle für eine Input-Komponente i über den Suchradius ε	87
Abbildung 4.23: Mögliche Kombinationen aus x -, y - und z -Abmessung der Komponenten j aus der Menge der ähnlichsten Modelle (blau) sowie die gewählte Kombination der Input-Komponente i (rot)	90
Abbildung 4.24: Exemplarische Einteilung der Kombination von x -, y - und z -Abmessung in zwei Cluster (a) sowie exemplarischer Bereich der produktionsrelevanten Produkteigenschaften (b)	91
Abbildung 4.25: Bewertung der produktionsrelevanten Produkteigenschaften einer Komponenten i über ein Ampelsystem	93
Abbildung 4.26: Exemplarische Bewertung der produktionsrelevanten Produkteigenschaften	93
Abbildung 4.27: Prinzipielle Vorgehensweise der Sequenzanalyse mittels RNN	95
Abbildung 4.28: Betrachtete Varianten des Inputs für das RNN in Abhängigkeit der Umsetzungsstufe	97
Abbildung 4.29: Grundsätzlicher Aufbau der RNN-Architektur	98
Abbildung 4.30: Anwendung des trainierten RNNs zur Vorhersage des nächsten Konstruktionszustandes für einen gegebenen Bauteilzustand	101
Abbildung 4.31: Zusammenspiel der Funktionalitäten als Gesamtsystem	103
Abbildung 5.1: Extrahierbare Datenkategorien für den betrachteten Anwendungsfall	104

Abbildung 5.2: Übersicht über die fünf Bauteilklassen der Datenbasis mit beispielhaftem Vertreter und Anzahl an Objekten je Klasse	105
Abbildung 5.3: Bereinigter Anteil der Materialarten je Bauteilklasse	106
Abbildung 5.4: Exemplarische Darstellung der Metainformationen für einen Adapter	107
Abbildung 5.5: Verteilung der Häufigkeiten bestimmter Sequenzlängen je Bauteilklasse (in Anlehnung an A_Holtzwardt (2022))	108
Abbildung 5.6: Exemplarische Vertreter von Flanschen aus den zwei üblichen Sequenzlängenbereichen	108
Abbildung 5.7: Anteil einzelner KEs je Klasse (in Anlehnung an A_Holtzwardt (2022))	109
Abbildung 5.8: Sequenz der Zwischenzustände und KEs eines exemplarischen Adapters bestehend aus elf Schritten. Blau markiert ist der finale Zustand.	110
Abbildung 5.9: Umsetzbarer Reifegrad des Assistenzsystem für den Anwendungsfall	110
Abbildung 5.10: Zusammenhang zwischen den Datensätzen Gesamt_SO und Finale_Komponenten_SO (orange hinterlegt)	112
Abbildung 5.11: Exemplarische Darstellung der normierten Metainformationen für einen Adapter	113
Abbildung 5.12: Beispielhafte Rekonstruktionen von Input-Punktwolken für den Autoencoder ohne Standardausrichtung	116
Abbildung 5.13: t-SNE Visualisierung der latenten Vektoren der fünf Bauteilklassen (a) sowie der gefundenen Cluster über k-Means (b) für den Autoencoder ohne Standardausrichtung	116
Abbildung 5.14: Darstellung exemplarischer Vertreter von Flanschen aus den visuellen Ansammlungen in der t-SNE Visualisierung für den Autoencoder ohne Standardausrichtung	117
Abbildung 5.15: Effekt der Standardorientierung am Beispiel von runden Flanschen	118

Abbildung 5.16: t-SNE Visualisierung der latenten Vektoren der fünf Bauteilklassen (a) sowie der gefundenen Cluster über k-Means (b) für den Autoencoder mit Standardausrichtung	119
Abbildung 5.17: Darstellung exemplarischer Vertreter von Adaptern aus den visuellen Ansammlungen in der t-SNE Visualisierung für den Autoencoder mit Standardausrichtung	119
Abbildung 5.18: Darstellung exemplarischer Vertreter von Deckeln aus verschiedenen Bereichen der t-SNE Visualisierung für den Autoencoder mit Standardausrichtung	120
Abbildung 5.19: t-SNE Visualisierung der latenten Vektoren der fünf Bauteilklassen (a) sowie der gefundenen Cluster über k-Means (b) für den Autoencoder mit Klassifikationsverlust	121
Abbildung 5.20: Darstellung exemplarischer Vertreter von Gehäusen und Deckeln aus verschiedenen Bereichen der t-SNE Visualisierung für den Autoencoder mit Klassifikationsverlust	122
Abbildung 5.21: t-SNE Visualisierung der latenten Vektoren der fünf Bauteilklassen inklusive aller Zwischenzustände (a) und ohne Zwischenzustände (b)	124
Abbildung 5.22: Erster (a) und letzter (b) Zwischenzustand eines beispielhaften Gehäuses mit 226 Konstruktionsschritten	124
Abbildung 5.23: t-SNE Visualisierung der über k-Means gefundenen Cluster für die latenten Vektoren aller Zwischenzustände (a) und ohne Zwischenzustände (b)	125
Abbildung 5.24: Vergleich der Rekonstruktionen des Autoencoders trainiert auf nur finalen Komponenten (Rekonstruktion (I)) und des Autoencoders trainiert auf allen Zwischenzuständen (Rekonstruktion (II))	126
Abbildung 5.25: Beispielhafte Interpolation zwischen einem Adapter und einem O-Ring (A_Bräuner 2022)	127

Abbildung 5.26: Ergebnisse der Ähnlichkeitssuche für drei Bauteilzustände von Flansch 230036437 für die latenten Vektoren des Modells „AE_Final“ und „AE_ZW“. Identische Modelle je Bauteilzustand sind in der gleichen Farbe hinterlegt.	131
Abbildung 5.27: Ergebnisse der Ähnlichkeitssuche für drei Bauteilzustände von Flansch 230036437 auf Metainformationen	132
Abbildung 5.28: Vergleich der Rekonstruktionen von „AE_Final“ und „AE_ZW“ für zwei Gehäuse	134
Abbildung 5.29: Ergebnisse der Ähnlichkeitssuche auf den latenten Vektoren von „AE_ZW“ über Umsetzungsstufe 1 und Umsetzungsstufe 2	135
Abbildung 5.30: Bereiche produktionsrelevanter Produkteigenschaften auf Basis der fünf gefundenen Cluster für O-Ring 231033780	137
Abbildung 5.31: Bewertung der produktionsrelevanten Produkteigenschaften anhand des relevanten Bereichs PP_1 für O-Ring 231033780	137
Abbildung 5.32: Bereiche produktionsrelevanter Produkteigenschaften auf Basis der zwei gefundenen Cluster für Adapter 230021093	138
Abbildung 5.33: Bewertung der produktionsrelevanten Produkteigenschaften anhand aller gefundenen Bereiche PP_h für Adapter 230021093 (Zwischenzustand 7)	139
Abbildung 5.34: Minimaler Testverlust für verschiedene Sequenzlängen in Abhängigkeit unterschiedlicher Hyperparameterkonfigurationen	142
Abbildung 5.35: Durchschnittlicher Testverlust in Abhängigkeit der Sequenzlängen sowie der durchschnittliche Beitrag der einzelnen Klassen zu diesem Verlust	143
Abbildung 5.36: Vorhersage des nächsten Zwischenzustandes für Adapter 230006082	144
Abbildung 5.37: Vorhersage des nächsten Zwischenzustandes für Flansch 230036437	145
Abbildung 5.38: Vorhersage des nächsten Zwischenzustandes für Deckel 230039626	146

Abbildung 5.39: Ergebnisse der Ähnlichkeitssuche für den Flansch 230036437 für den ersten und zweiten Zwischenzustand auf den latenten Vektoren des Modells „AE_ZW“ sowie der Vorhersage durch das RNN	147
Abbildung 5.40: Bewertung der produktionsrelevanten Produkteigenschaften für den zweiten Zwischenzustand (oben) sowie den finalen Zustand (unten) von Flansch 230036437	148
Abbildung 5.41: Funktionsflächen von Flansch 230036437 aus verschiedenen Perspektiven	149
Abbildung 5.42: Seitenflansch mit zusätzlicher oberer Lasche	150
Abbildung 5.43: Ergebnisse der Ähnlichkeitssuche für den Adapter 230005421 für den vierten Zwischenzustand auf den latenten Vektoren des Modells „AE_ZW“ sowie der Vorhersage durch das RNN	151
Abbildung 5.44: Bewertung der produktionsrelevanten Produkteigenschaften für den Zwischenzustand von Adapter 230005421	151
Abbildung A-1: Auf Basis der Ähnlichkeit der Strukturbäume gefundene Cluster. In Cluster 0 sind dabei solche Strukturbäume, die keinem Cluster zugeordnet werden konnten.	XIII
Abbildung B-1: Konfusionsmatrix der Klassifikation mittels Multi-View-Ansatz nach Su et al. (2015) mit Standardorientierung gemäß Krahe et al. (2019)	XIV
Abbildung B-2: Konfusionsmatrix der Klassifikation mittels graphbasiertem CNN nach Simonovsky & Komodakis (2017)	XIV
Abbildung B-3: Konfusionsmatrix der Klassifikation mittels Punktwolken nach Qi et al. (2017b)	XV
Abbildung B-4: Konfusionsmatrix der Klassifikation mittels Mesh-CNN nach Hanocka et al. (2019)	XV
Abbildung C-1: Detaillierter Aufbau des Autoencoders in Anlehnung an A_Holtzwardt (2022)	XVI
Abbildung F-1: Anteil der vier ursprünglichen Materialarten des Datensatzes je Klasse	XVIII
Abbildung F-2: Verteilung der im Datensatz auftretenden Dichten für Edelstahl	XVIII

Abbildung F-3: Verteilung der im Datensatz auftretenden Dichten für Kunststoff	XIX
Abbildung G-1: Übersicht über die Verteilung der Sequenzlängen innerhalb und zwischen den Bauteilklassen	XIX
Abbildung G-2: Position der häufig genutzten KEs im Konstruktionsprozess (in Anlehnung an A_Holtzwardt (2022))	XX
Abbildung J-1: Übersicht über die Anzahl an Eckpunkten je Bauteilkategorie sowie für den gesamten Datensatz (in Anlehnung an A_Holtzwardt (2022))	XXII
Abbildung J-2: Punktwolke eines Gehäuses mit 125 KEs auf Basis von nur Eckpunkten (a), eines Adapters mit 10 KEs auf Basis von nur Eckpunkten (b) sowie Eckpunkten und Flächenpunkten (c)	XXII
Abbildung K-1: Verteilung der betrachteten Metainformationen	XXIII
Abbildung M-1: Verlauf des Rekonstruktionsverlustes in Form der CD für Trainings- und Testdaten sowie der Verlauf der Differenz beider Verluste	XXV
Abbildung N-1: Konfusionsmatrix für die Clusterlösung der latenten Vektoren des Autoencoders ohne Standardausrichtung	XXV
Abbildung N-2: Konfusionsmatrix für die Clusterlösung der latenten Vektoren des Autoencoders mit Standardausrichtung	XXVI
Abbildung N-3: Konfusionsmatrix für die Clusterlösung der latenten Vektoren des Autoencoders mit Klassifikationsverlust	XXVI
Abbildung O-1: Konfusionsmatrix für die Klassifikation der latenten Vektoren des Autoencoders ohne Standardausrichtung. Die erreichte Accuracy beträgt 90 %.	XXVII
Abbildung O-2: Konfusionsmatrix für die Klassifikation der latenten Vektoren des Autoencoders mit Standardausrichtung. Die erreichte Accuracy beträgt 90 %.	XXVII
Abbildung O-3: Konfusionsmatrix für die Klassifikation der latenten Vektoren des Autoencoders mit Klassifikationsverlust. Die erreichte Accuracy beträgt 92 %.	XXVIII

Abbildung O-4: Konfusionsmatrix für die Klassifikation der latenten Vektoren aller Zwischenzustände des auf allen Zwischenzuständen trainierten Autoencoders. Die erreichte Accuracy beträgt 98,40 %.	XXVIII
Abbildung O-5: Konfusionsmatrix für die Klassifikation der latenten Vektoren nur finaler Zustände des auf allen Zwischenzuständen trainierten Autoencoders. Die erreichte Accuracy beträgt 95,35 %.	XXIX
Abbildung P-1: Interpolation zwischen den finalen Zuständen von Adapter 230005421 und Adapter 230007998 (in Anlehnung an A_Bräuner (2022))	XXIX
Abbildung P-2: Interpolation zwischen dem ersten und finalen Zustand von Adapter 230005421 (in Anlehnung an A_Bräuner (2022))	XXX
Abbildung P-3: Addition der von Adapter 230005421 extrahierten Eigenschaft Bohrung auf einen Zustand ohne Bohrung von Adapter 230000597 (in Anlehnung an A_Bräuner (2022))	XXX
Abbildung Q-1: Überblick über die Untersuchungsteile mit jeweils erstem und finalen Zwischenzustand sowie einem mittleren Zustand, der je Bauteil etwa der Hälfte der Konstruktionsschritte entspricht	XXXI
Abbildung Q-2: Bestimmung der optimalen Anzahl k an Clustern mittels Davies Bouldin (DB) Score. Das Optimum liegt bei $k=5$.	XXXII
Abbildung Q-3: Bestimmung des Suchradius ϵ für AE_Final über das Verfahren nach Ester et al. (1996)	XXXII
Abbildung Q-4: Bestimmung des Suchradius ϵ für AE_ZW über das Verfahren nach Ester et al. (1996)	XXXIII
Abbildung Q-5: Ergebnisse auf latenten Vektoren von „AE_Final“ und „AE_ZW“ für Flansch 230017072. Gleiche Modelle je Bauteilzustand sind farblich hervorgehoben.	XXXVI
Abbildung Q-6: Ergebnisse der Suche über Metainformationen für Flansch 230017072. Gleiche Teile sind farblich hervorgehoben.	XXXVII
Abbildung Q-7: Ergebnisse auf latenten Vektoren von „AE_Final“ und „AE_ZW“ für Gehäuse 230044255. Gleiche Teile je Bauteilzustand sind farblich hervorgehoben.	XXXVIII

Abbildung Q-8: Ergebnisse der Suche über Metainformationen für Gehäuse 230044255	XXXIX
Abbildung Q-9: Ergebnisse auf latenten Vektoren von „AE_Final“ und „AE_ZW“ für Gehäuse 230018316. Gleiche Teile je Bauteilzustand sind farblich hervorgehoben.	XL
Abbildung Q-10: Ergebnisse der Suche über Metainformationen für Gehäuse 230018316	XLI
Abbildung Q-11: Ergebnisse auf latenten Vektoren von „AE_Final“ und „AE_ZW“ für Deckel 230030276. Gleiche Teile je Bauteilzustand sind farblich hervorgehoben.	XLII
Abbildung Q-12: Ergebnisse der Suche über Metainformationen für Deckel 230030276	XLIII
Abbildung Q-13: Ergebnisse auf latenten Vektoren von „AE_Final“ und „AE_ZW“ für Deckel 230039626. Gleiche Teile je Bauteilzustand sind farblich hervorgehoben.	XLIV
Abbildung Q-14: Ergebnisse der Suche über Metainformationen für Deckel 230039626. Dargestellt ist je Zwischenzustand der Komponenten i das Bauteil, dessen Position dem unähnlichsten Teil innerhalb und dem ersten Teil außerhalb $S_{\text{sim},i}$ basierend auf den latenten Vektoren von AE_ZW entspricht.	XLV
Abbildung Q-15: Ergebnisse auf latenten Vektoren von „AE_Final“ und „AE_ZW“ für Adapter 230021093. Gleiche Teile sind je Bauteilzustand farblich hervorgehoben.	XLVI
Abbildung Q-16: Ergebnisse der Suche über Metainformationen für Adapter 230021093. Dargestellt ist je Zwischenzustand der Komponenten i das Bauteil, dessen Position dem unähnlichsten Teil innerhalb und dem ersten Teil außerhalb $S_{\text{sim},i}$ basierend auf den latenten Vektoren von AE_ZW entspricht. Gleiche Teile sind farblich hervorgehoben.	XLVII
Abbildung Q-17: Prozentuale Häufigkeit der euklidischen Distanzen zwischen den latenten Vektoren der finalen Modelle je Bauteilklasse	XLVIII

Abbildung Q-18: Ergebnisse der Ähnlichkeitssuche auf den latenten Vektoren von „AE_ZW“ über Umsetzungsstufe 1 und Umsetzungsstufe 2 für O-Ring 230016675. Gleiche Teile sind farblich hervorgehoben.	XLIX
Abbildung Q-19: Ergebnisse der Ähnlichkeitssuche auf den latenten Vektoren von „AE_ZW“ über Umsetzungsstufe 1 und Umsetzungsstufe 2 für Adapter 23005421 (mittlerer Zwischenzustand)	L
Abbildung Q-20: Weitere Adapter aus Kunststoff im betrachteten Datensatz neben Adapter 230021093	LI
Abbildung Q-21: Bestimmung von ε_{DB} nach Ester et al. (1996) für die Bewertung der produktionsrelevanten Produkteigenschaften in Form von Abmessungen und Dichte auf Basis der euklidischen Distanz	LII
Abbildung R-1: Anzahl der Sequenzabschnitte in Abhängigkeit der Fenstergröße sowie Häufigkeit eines Zwischenzustandes in Abhängigkeit seiner Position sowie der Fenstergröße	LII
Abbildung S-1: Beispielhafter Adapter mit kürzerer Extrusion im zweiten Zwischenzustand als CAD-Modell (a), generierte Punktwolke (b) und durch den Autoencoder rekonstruierte Punktwolke (c)	LII
Abbildung S-2: Visualisierung der Vorhersage für Gehäuse 230044255	LIII
Abbildung T-1: Bereiche produktionsrelevanter Produkteigenschaften auf Basis der zwei gefundenen Cluster für Adapter 230005421 (Zwischenzustand 4)	LIII

Tabellenverzeichnis

Tabelle 5-1: Betrachtete Hyperparameterkonfigurationen	140
Tabelle C-1: Übersicht über die Hyperparameter des Autoencoders	XVI
Tabelle D-1: Übersicht über die Hyperparameter des RNNs	XVII
Tabelle E-1: Übersicht über die auslesbaren Metainformationen mittels des PTC Creo® Object TOOLKIT Java	XVII
Tabelle G-1: Übersicht über die Sequenzlängen je Bauteilklass	XX
Tabelle H-1: Anzahl der finalen Modelle und Zwischenzustände je Bauteilklass	XX
Tabelle H-2: Anzahl der finalen Modelle und Zwischenzustände je Bauteilklass nach Bereinigung für die Sequenzanalyse mittels RNN	XXI
Tabelle I-1: Betrachtete Konstellationen des Datensatzes mit jeweils genutzten Datenkategorien, Anzahl an Objekten und Train/Test Split. In Klammern ist der Anteil finaler Bauteilzustände angegeben.	XXI
Tabelle L-1: Anzahl der betrachteten Modelle in Trainings- und Testdaten sowie Menge der daraus erstellten Sequenzabschnitte für verschiedene Fenstergrößen Q	XXIII
Tabelle M-1: Hyperparameter für die verschiedenen Autoencoder-Ansätze auf finalen Komponenten	XXIV
Tabelle M-2: Hyperparameter des Autoencoders auf allen Zwischenzuständen	XXIV
Tabelle S-1: Optimale Parametrisierungen der Hyperparameter des RNNs	LII

Anhang

Anhang A Clusterergebnisse auf Basis der Strukturbäume

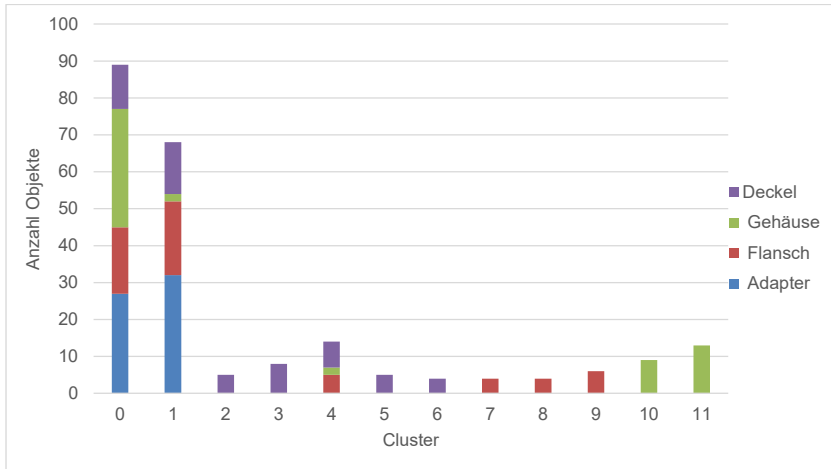
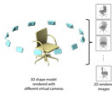


Abbildung A-1: Auf Basis der Ähnlichkeit der Strukturbäume gefundene Cluster. In Cluster 0 sind dabei solche Strukturbäume, die keinem Cluster zugeordnet werden konnten.

Anhang B Klassifikationsergebnisse für verschiedene Repräsentationsformen

Die nachfolgenden Ergebnisse basieren auf dem *AIAx CAD Dataset* (Tayyub et al. 2021).

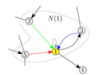
Anhang B.1 Multi-View



		Prognostizierte Klasse						
		Deckel	Flansch	Gehäuse	Werkzeug	Stabsonde	Sensor	Rohr
Tatsächliche Klasse	Deckel	0,84	0,04	0,08	0,00	0,00	0,00	0,04
	Flansch	0,00	1,00	0,00	0,00	0,00	0,00	0,00
	Gehäuse	0,00	0,08	0,84	0,08	0,00	0,00	0,00
	Werkzeug	0,04	0,04	0,48	0,44	0,00	0,00	0,00
	Stabsonde	0,00	0,00	0,00	0,00	0,92	0,00	0,08
	Sensor	0,00	0,00	0,00	0,00	0,00	1,00	0,00
	Rohr	0,00	0,00	0,00	0,00	0,12	0,00	0,88
Accuracy:		84,57 %						

Abbildung B-1: Konfusionsmatrix der Klassifikation mittels Multi-View-Ansatz nach Su et al. (2015) mit Standardorientierung gemäß Krahe et al. (2019)


Anhang B.2 Graphen



		Prognostizierte Klasse						
		Deckel	Flansch	Gehäuse	Werkzeug	Stabsonde	Sensor	Rohr
Tatsächliche Klasse	Deckel	79,33	0,00	7,00	0,00	0,00	13,67	0,00
	Flansch	0,00	100,00	0,00	0,00	0,00	0,00	0,00
	Gehäuse	4,00	0,00	94,33	0,00	0,00	1,67	0,00
	Werkzeug	0,00	10,33	11,67	70,33	7,67	0,00	0,00
	Stabsonde	0,00	0,00	0,00	0,00	68,00	0,00	32,00
	Sensor	2,00	0,00	0,00	0,00	0,00	94,33	3,67
	Rohr	0,00	0,00	0,00	0,00	8,00	0,67	91,33
Accuracy:		85,38 %						

Abbildung B-2: Konfusionsmatrix der Klassifikation mittels graphbasiertem CNN nach Simonovsky & Komodakis (2017)

Anhang B.3 Punktwolken

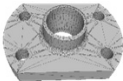


		Prognostizierte Klasse						
		Deckel	Flansch	Gehäuse	Werkzeug	Stabsonde	Sensor	Rohr
Tatsächliche Klasse	Deckel	0,97	0,00	0,00	0,00	0,00	0,00	0,03
	Flansch	0,00	1,00	0,00	0,00	0,00	0,00	0,00
	Gehäuse	0,00	0,00	0,95	0,02	0,00	0,02	0,00
	Werkzeug	0,00	0,07	0,33	0,60	0,00	0,00	0,00
	Stabsonde	0,00	0,00	0,00	0,00	0,96	0,04	0,00
	Sensor	0,00	0,00	0,00	0,00	0,07	0,89	0,04
	Rohr	0,00	0,00	0,00	0,00	0,00	0,03	0,97
Accuracy:		96,59 %						

Abbildung B-3: Konfusionsmatrix der Klassifikation mittels Punktwolken nach Qi et al. (2017b)

Anhang B.4 Meshes

Aufgrund der sehr aufwändigen Datenaufbereitung wurde hier nur ein verkleinerter Datensatz mit vier verschiedenen Bauteilklassen herangezogen (A_Spuler 2020).



		Prognostizierte Klasse			
		Adapter	Flansch	Ring	Rohr
Tatsächliche Klasse	Adapter	1,00	0,00	0,00	0,00
	Flansch	0,00	1,00	0,00	0,00
	Ring	0,13	0,00	0,88	0,00
	Rohr	0,00	0,00	0,00	1,00
Accuracy:		96,80 %			

Abbildung B-4: Konfusionsmatrix der Klassifikation mittels Mesh-CNN nach Hanocka et al. (2019)

Anhang C Autoencoder

Anhang C.1 Detaillierter Aufbau des Autoencoders

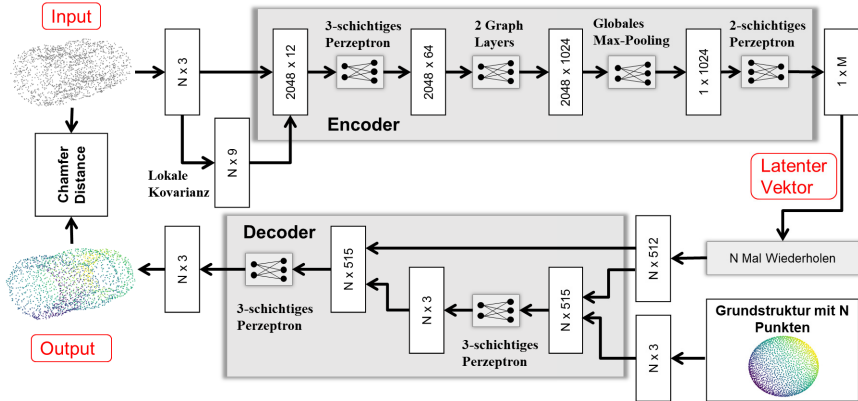


Abbildung C-1: Detaillierter Aufbau des Autoencoders in Anlehnung an A_Holtzwardt (2022)

Anhang C.2 Hyperparameter des Autoencoders

Tabelle C-1: Übersicht über die Hyperparameter des Autoencoders

Hyperparameter	Beschreibung
# Punkten n	Anzahl an Punkten, die auf der Oberfläche des 3D-Modells verteilt werden
# Trainingsepochen	Anzahl der Trainingsepochen, die durchgeführt werden, um die optimalen Netzgewichte zu erlernen
Batch Size	Anzahl an Punktwolken, die während des Trainings zu einem Bündel zusammengefasst werden
Bottleneck Size m	Die Anzahl der Dimensionen, auf die der ursprüngliche Input reduziert werden soll
Lernrate λ	Steuert die Schrittweite, mit der die Netzgewichte in Richtung Fehlerminimierung angepasst werden

Anhang D Hyperparameter des RNNs

Tabelle D-1: Übersicht über die Hyperparameter des RNNs

Hyperparameter	Beschreibung
# Schichten (Tiefe)	Anzahl der RNN-Schichten
# Neuronen pro Schicht (Breite)	Anzahl der Neuronen pro RNN-Schicht
Typ RNN-Zelle	Typ der RNN-Zelle (einfaches RNN, LSTM oder GRU)
# Epochen	Anzahl der Trainingsepochen
Lernrate λ	Steuert die Schrittweite, mit der die Netzgewichte in Richtung Fehlerminimierung angepasst werden
Länge Q der Sequenzabschnitte	Anzahl der Elemente einer Sequenz, die als Input für das RNN dient

Anhang E Übersicht über die auslesbaren Metainformationen

Tabelle E-1: Übersicht über die auslesbaren Metainformationen mittels des PTC Creo® Object TOOLKIT Java

Nr.	Variable	Einheit
1	Max. Abmessung erste Achse	mm
2	Max. Abmessung zweite Achse	mm
3	Max. Abmessung dritte Achse	mm
4	Volumen	mm ³
5	Oberfläche	mm ²
6	Volumen-Oberfläche Ratio R ₁	mm
7	Volumen-Oberfläche Ratio R ₂	mm
8	Kompaktheit	-
9	Crinkliness	-
10	Gewicht	kg
11	Dichte	kg/mm ³
12	Anzahl Flächen	-
13	Anzahl Kanten	-
14	Anzahl Flächentypen: Ebenen	-
15	Anzahl Flächentypen: Kegel	-
16	Anzahl Flächentypen: Zylinder	-
17	Anzahl Flächentypen: Sonstige	-
18	Größte Fläche	mm ²

Anhang F.3 Verteilung der Dichten für Kunststoff

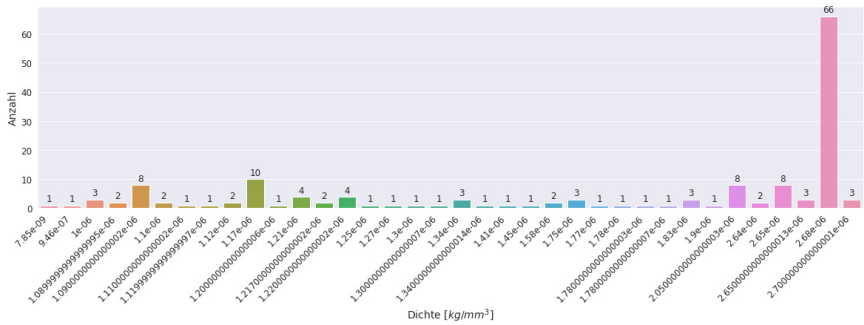


Abbildung F-3: Verteilung der im Datensatz auftretenden Dichten für Kunststoff

Anhang G Untersuchung der Konstruktionssequenzen verschiedener Bauteilklassen

Anhang G.1 Übersicht über die Verteilung der Sequenzlängen innerhalb und zwischen den Bauteilklassen

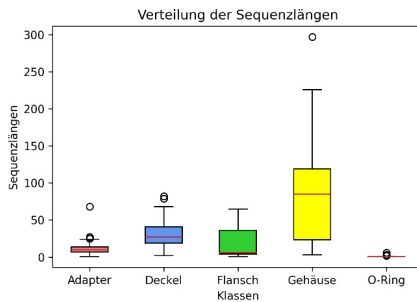


Abbildung G-1: Übersicht über die Verteilung der Sequenzlängen innerhalb und zwischen den Bauteilklassen

Anhang G.2 Übersicht über die Sequenzlängen je Bauteilklasse

Tabelle G-1: Übersicht über die Sequenzlängen je Bauteilklasse

Klasse	Min	Max	Ø	Std
Adapter	1	68	11,17	5,95
Deckel	2	82	30,37	17,93
Flansch	1	65	17,25	19,99
Gehäuse	3	297	83,83	62,70
O-Ring	1	6	1,15	0,57

Anhang G.3 Position der häufig genutzten KEs im Konstruktionsprozess

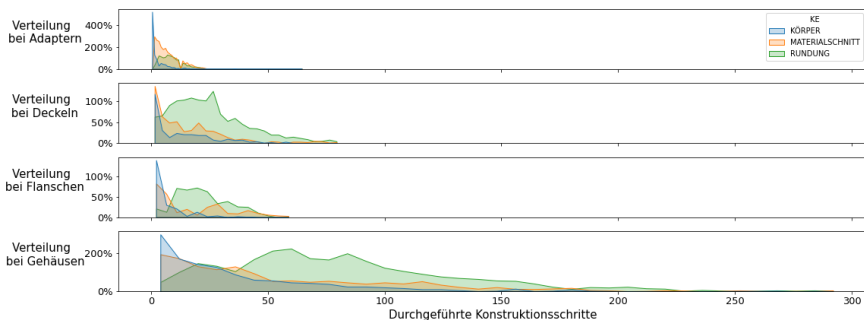


Abbildung G-2: Position der häufig genutzten KEs im Konstruktionsprozess (in Anlehnung an A_Holtzwardt (2022))

Anhang H Anzahl der generierten Modelle für das Training und Testen

In der nachfolgenden Tabelle ist die Anzahl der Zwischenzustände je Bauteilklasse aufgezählt, die für das Training und Testen des *Autoencoders* genutzt wurden.

Tabelle H-1: Anzahl der finalen Modelle und Zwischenzustände je Bauteilklasse

	Adapter	Deckel	Flansch	Gehäuse	O-Ring
Anzahl finale Modelle	524	101	93	96	143
Generierte Zwischenstände	5851	3067	1604	8048	164

Einige der im Datensatz enthaltenen Modelle mussten aufgrund von Fehlern in den Referenzen des Modellbaums oder einer nicht übereinstimmenden Anzahl an ausgelesenen KEs und erzeugten geometrischen Zwischenständen für die Sequenzanalyse

mittels des RNN aussortiert werden. Daher wird hierfür die ursprüngliche Gesamtanzahl gemäß Tabelle H-2 von 957 Modellen auf 925 reduziert.

Tabelle H-2: Anzahl der finalen Modelle und Zwischenzustände je Bauteilkategorie nach Bereinigung für die Sequenzanalyse mittels RNN

	Adapter	Deckel	Flansch	Gehäuse	O-Ring
Anzahl finaler Modelle	521	97	82	87	138
Generierte Zwischenstände	5821	2948	1294	6984	157

Anhang I Betrachtete Konstellationen des Datensatzes

Tabelle I-1: Betrachtete Konstellationen des Datensatzes mit jeweils genutzten Datenkategorien, Anzahl an Objekten und Train/Test Split. In Klammern ist der Anteil finaler Bauteilzustände angegeben.

Bezeichnung	Genutzte Datenkategorien	# Objekte (davon final)	Train/Test Split
Finale_Komponenten_NS0	3D-Geometrie	958 (958)	95 %/5 %
Finale_Komponenten_SO	3D-Geometrie Metainformationen	958 (958)	95 %/5 %
Gesamt_SO	3D-Geometrie Metainformationen Konstruktionsvorgehen	18734 (958)	95 %/5 %

Anhang J Verteilung der Anzahl an Eckpunkten der Polygonnetze je Bauteilkategorie

Nachfolgend ist eine Übersicht über die Anzahl an Eckpunkten je Bauteilkategorie sowie für den gesamten Datensatz dargestellt. Betrachtet werden dabei alle Zwischenzustände inklusive der zugehörigen finalen Komponente. Die Anzahl der Eckpunkte korreliert wie auch die Länge der Konstruktionssequenz (und damit die Anzahl an Zwischenzuständen) mit der Komplexität der Modelle. Für ca. 86% aller Zwischenzustände liegen weniger als 2.048 Eckpunkte vor (horizontale gestrichelte Linie). Lediglich für die Klassen Gehäuse und Deckel gibt es Modelle, deren Eckpunkte nicht vollständig über die Punktwolke abgebildet werden.

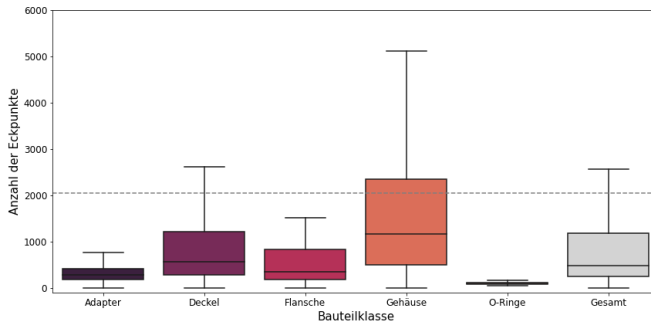


Abbildung J-1: Übersicht über die Anzahl an Eckpunkten je Bauteilklasse sowie für den gesamten Datensatz (in Anlehnung an A_Holtzwardt (2022))

In Abbildung J-2 sind beispielhaft die Punktwolke eines Gehäuses mit 125 Konstruktionsschritten (a) sowie die Punktwolke eines Adapters mit 10 Konstruktionsschritten abgebildet. Das Gehäuse verfügt über 3.146 Eckpunkte, die 2.048 Punkte der Punktwolke werden folglich ausschließlich auf Eckpunkte gesetzt. Der Adapter verfügt lediglich über 306 Eckpunkte (siehe (b)), weshalb die weiteren Punkte auf den Flächen verteilt werden (siehe (c)).

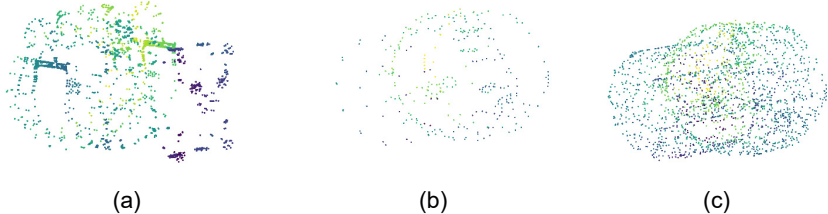


Abbildung J-2: Punktwolke eines Gehäuses mit 125 KEs auf Basis von nur Eckpunkten (a), eines Adapters mit 10 KEs auf Basis von nur Eckpunkten (b) sowie Eckpunkten und Flächenpunkten (c)

Anhang K Verteilung der betrachteten Metainformationen

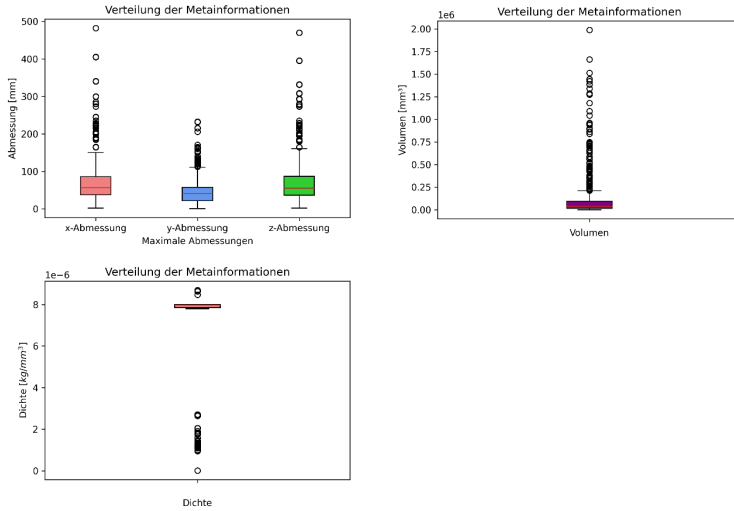


Abbildung K-1: Verteilung der betrachteten Metainformationen

Anhang L Anzahl an betrachteten Modellen in Abhängigkeit der Sequenzabschnittslänge

Tabelle L-1: Anzahl der betrachteten Modelle in Trainings- und Testdaten sowie Menge der daraus erstellten Sequenzabschnitte für verschiedene Fenstergrößen Q

	Sequenzabschnittslänge/ Fenstergröße Q			
	2	4	8	12
Modelle in Trainingsdaten	755	700	540	385
Erstellte Sequenzabschnitte aus Trainingsdaten	15386	13899	11323	9401
Modelle in Testdaten	36	35	28	19
Erstellte Sequenzabschnitte aus Testdaten	893	821	688	586

Anhang M Hyperparameter für die *Autoencoder*-Ansätze

Anhang M.1 Hyperparameter für die verschiedenen *Autoencoder*-Ansätze auf finalen Komponenten

Die Wahl der Hyperparameter in Tabelle M-1 basiert auf Voruntersuchungen aus den von der Autorin angeleiteten Arbeiten A_Schmutz (2021) und A_Ebi (2021).

Tabelle M-1: Hyperparameter für die verschiedenen Autoencoder-Ansätze auf finalen Komponenten

Hyperparameter	Ansatz 1	Ansatz 2	Ansatz 3
Anzahl an Punkten n	2048	2048	2048
# Trainingsepochen	1500	1500	Phase 1: 250 Phase 2: 1000
<i>Batch Size</i>	50	50	10
<i>Bottleneck Size m</i>	512	512	512
Lernrate λ	0,0005	0,0005	Phase 1: 0,001 Phase 2: 0,0001

Anhang M.2 Hyperparameter für den *Autoencoder* auf sequentiellen Daten

Die Wahl der Hyperparameter in Tabelle M-2 basiert auf Voruntersuchungen von A_Schmutz (2021) und A_Ebi (2021).

Tabelle M-2: Hyperparameter des Autoencoders auf allen Zwischenzuständen

Hyperparameter	Ausprägung
Anzahl an Punkten n	2048
# Trainingsepochen	1000
<i>Batch Size</i>	256
<i>Bottleneck Size m</i>	512
Lernrate λ	0,001

Anhang M.3 Trainingsverlauf des *Autoencoders* auf sequentiellen Daten

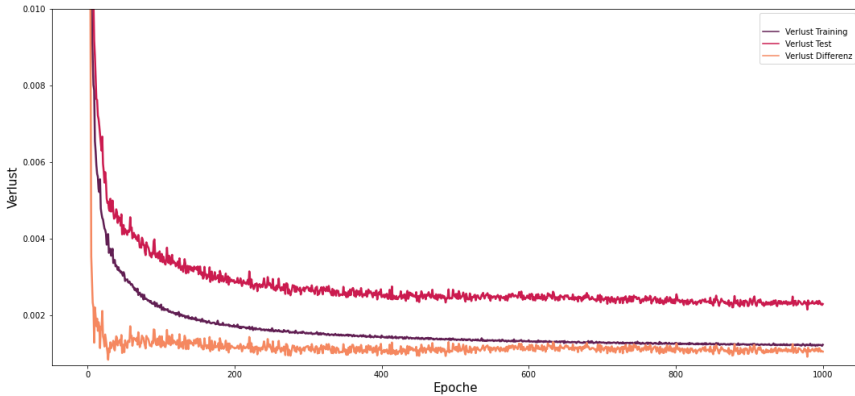


Abbildung M-1: Verlauf des Rekonstruktionsverlustes in Form der CD für Trainings- und Testdaten sowie der Verlauf der Differenz beider Verluste

Anhang N Konfusionsmatrizen der Clusterlösungen

Anhang N.1 Konfusionsmatrix für die Clusterlösung der latenten Vektoren des *Autoencoders* ohne Standardausrichtung

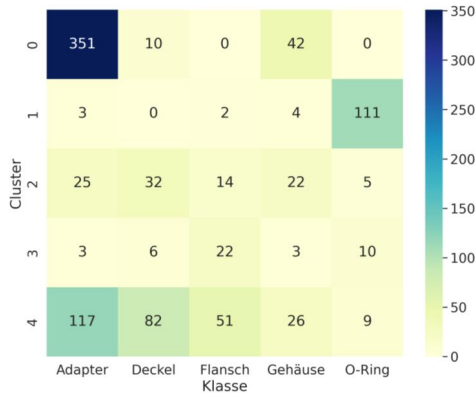


Abbildung N-1: Konfusionsmatrix für die Clusterlösung der latenten Vektoren des *Autoencoders* ohne Standardausrichtung

Anhang N.2 Konfusionsmatrix für die Clusterlösung der latenten Vektoren des Autoencoders mit Standardausrichtung

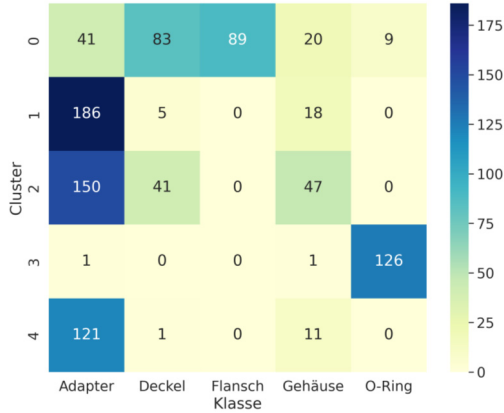


Abbildung N-2: Konfusionsmatrix für die Clusterlösung der latenten Vektoren des Autoencoders mit Standardausrichtung

Anhang N.3 Konfusionsmatrix für die Clusterlösung der latenten Vektoren des Autoencoders mit Klassifikationsverlust

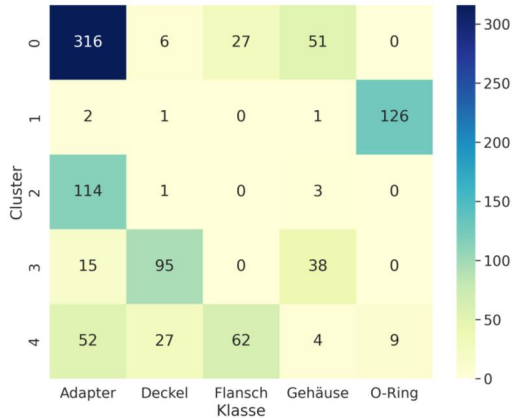


Abbildung N-3: Konfusionsmatrix für die Clusterlösung der latenten Vektoren des Autoencoders mit Klassifikationsverlust

Anhang O Konfusionsmatrizen

Anhang O.1 Konfusionsmatrix für die Klassifikation der latenten Vektoren des Autoencoders ohne Standardausrichtung

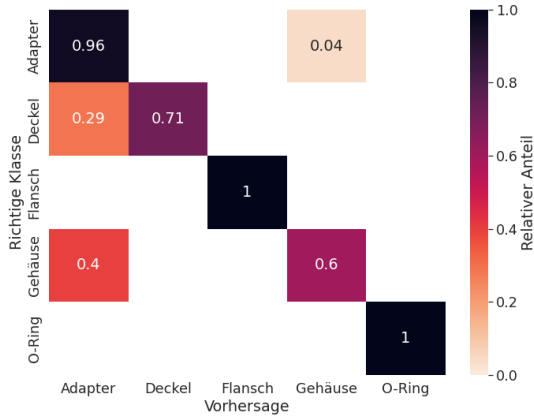


Abbildung O-1: Konfusionsmatrix für die Klassifikation der latenten Vektoren des Autoencoders ohne Standardausrichtung. Die erreichte Accuracy beträgt 90 %.

Anhang O.2 Konfusionsmatrix für die Klassifikation der latenten Vektoren des Autoencoders mit Standardausrichtung

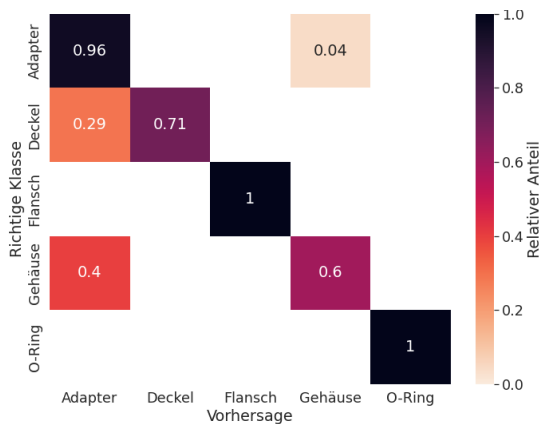


Abbildung O-2: Konfusionsmatrix für die Klassifikation der latenten Vektoren des Autoencoders mit Standardausrichtung. Die erreichte Accuracy beträgt 90 %.

Anhang O.3 Konfusionsmatrix für die Klassifikation der latenten Vektoren des Autoencoders mit Klassifikationsverlust

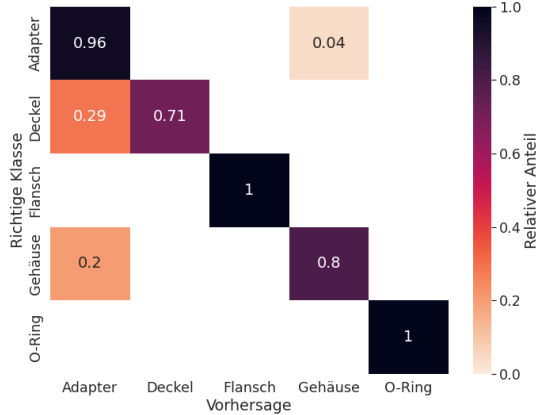


Abbildung O-3: Konfusionsmatrix für die Klassifikation der latenten Vektoren des Autoencoders mit Klassifikationsverlust. Die erreichte Accuracy beträgt 92 %.

Anhang O.4 Konfusionsmatrix für die Klassifikation der latenten Vektoren aller Zwischenzustände des Autoencoders mit sequentiellen Daten

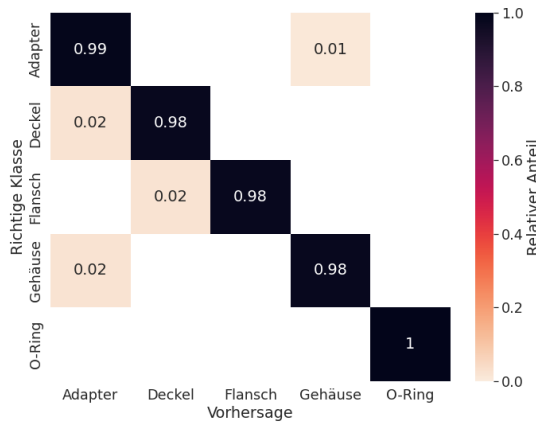


Abbildung O-4: Konfusionsmatrix für die Klassifikation der latenten Vektoren aller Zwischenzustände des auf allen Zwischenzuständen trainierten Autoencoders. Die erreichte Accuracy beträgt 98,40 %.

Anhang O.5 Konfusionsmatrix für die Klassifikation der latenten Vektoren nur finaler Komponenten des Autoencoders mit sequentiellen Daten

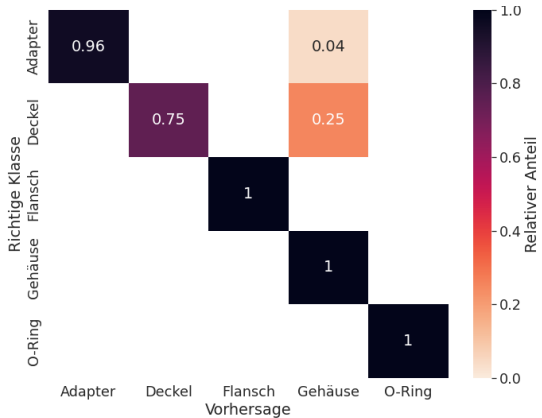


Abbildung O-5: Konfusionsmatrix für die Klassifikation der latenten Vektoren nur finaler Zustände des auf allen Zwischenzuständen trainierten Autoencoders. Die erreichte Accuracy beträgt 95,35 %.

Anhang P Interpolationen und algebraische Operationen

Anhang P.1 Interpolationen zwischen den finalen Zuständen zweier Adapter

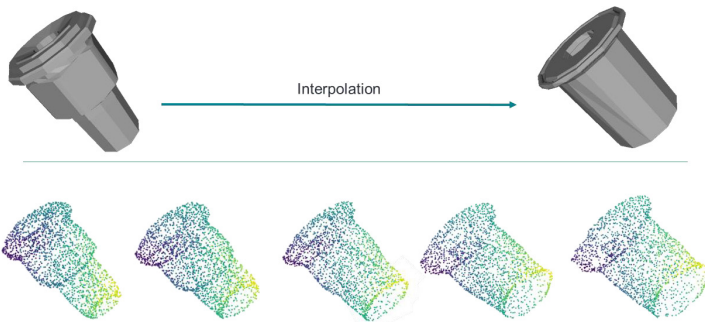


Abbildung P-1: Interpolation zwischen den finalen Zuständen von Adapter 230005421 und Adapter 230007998 (in Anlehnung an A_Bräuner (2022))

Anhang P.2 Interpolationen zwischen dem ersten Zwischenzustand und dem finalen Zustand eines Adapters

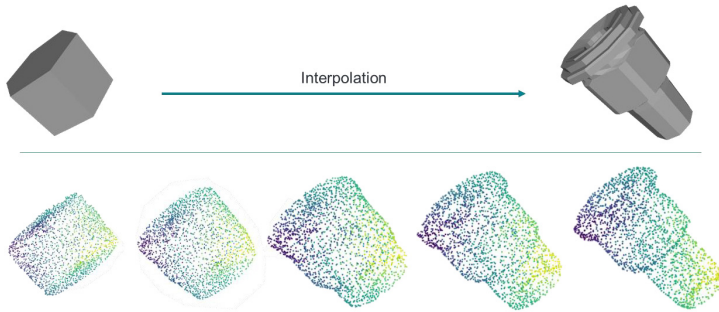


Abbildung P-2: Interpolation zwischen dem ersten und finalen Zustand von Adapter 230005421 (in Anlehnung an A_Bräuner (2022))

Anhang P.3 Addition der Eigenschaft Bohrung am Beispiel eines Adapters

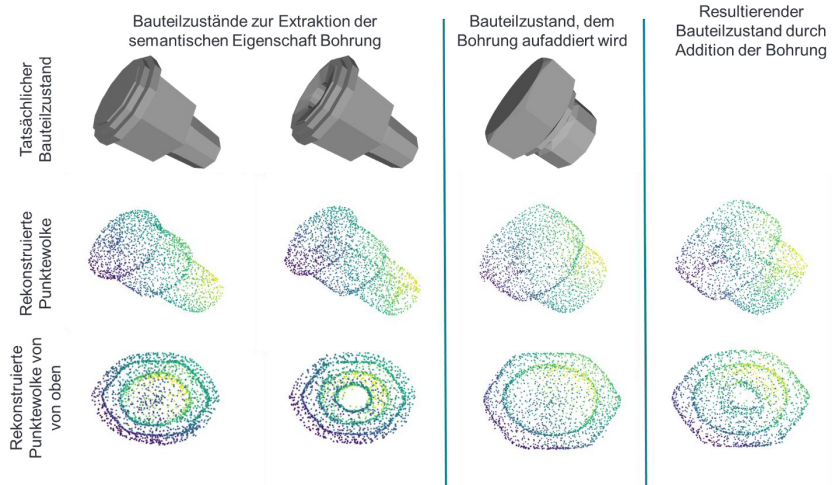


Abbildung P-3: Addition der von Adapter 230005421 extrahierten Eigenschaft Bohrung auf einen Zustand ohne Bohrung von Adapter 230000597 (in Anlehnung an A_Bräuner (2022))

Anhang Q Ähnlichkeitssuche

Anhang Q.1 Überblick Untersuchungsteile














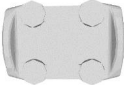
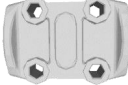

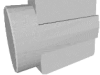
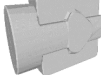



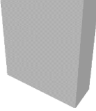




Name	Erster Schritt	Mittlerer Schritt	Letzter Schritt
Adapter_230021093 (Schritt 1, 7 und 14)			
Adapter_230005421 (Schritt 1, 5 und 10)			
Cover_230030276 (Schritt 1, 29 und 57)			
Cover_230039626 (Schritt 1, 11 und 23)			
Flange_230036437 (Schritt 1, 16 und 32)			
Flange_230017072 (Schritt 1, 18 und 36)			
Housing_230044255 (Schritt 1, 63 und 125)			
Housing_230018316 (Schritt 1, 62 und 125)			
Oring_230016675 (Schritt 1)		x	x
Oring_231033780 (Schritt 1)		x	x

Abbildung Q-1: Überblick über die Untersuchungsteile mit jeweils erstem und finalen Zwischenzustand sowie einem mittleren Zustand, der je Bauteil etwa der Hälfte der Konstruktionsschritte entspricht

Anhang Q.2 Bestimmung der optimalen Anzahl k an Clustern mittels Davies Bouldin Score

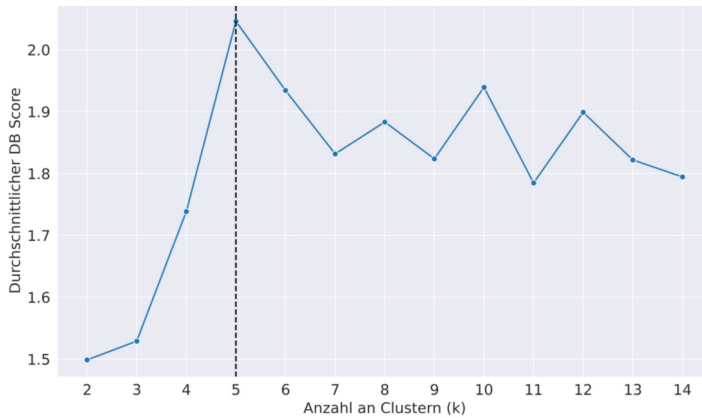


Abbildung Q-2: Bestimmung der optimalen Anzahl k an Clustern mittels Davies Bouldin (DB) Score. Das Optimum liegt bei $k=5$.

Anhang Q.3 Bestimmung des Suchradius ϵ für AE_Final

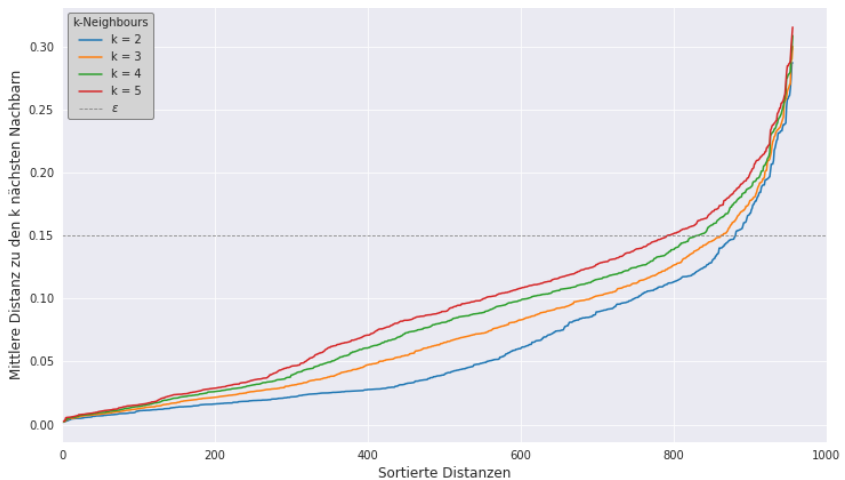


Abbildung Q-3: Bestimmung des Suchradius ϵ für AE_Final über das Verfahren nach Ester et al. (1996)

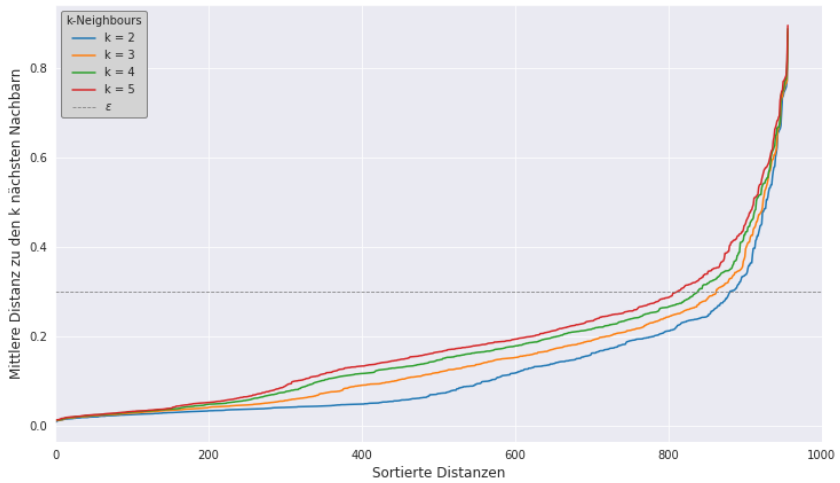
Anhang Q.4 Bestimmung des Suchradius ϵ für AE_ZW

Abbildung Q-4: Bestimmung des Suchradius ϵ für AE_ZW über das Verfahren nach Ester et al. (1996)

Anhang Q.5 Ergebnisse auf latenten Vektoren AE_Final

Input-Komponente	Suche nach finalen Komponenten mit latenten Vektoren AE_Final			
	$ S_{sim,i} $ für $\varepsilon = 0,15$			
	Suche ohne Cluster	1 Cluster	2 Cluster	Benötigte Cluster um gleiches Ergebnis zu erzielen
Adapter_230021093_step_001	22	22	22	1
Adapter_230021093_step_007	20	20	20	1
Adapter_230021093_step_014	22	22	22	1
Adapter_230005421_step_001	0	0	0	1
Adapter_230005421_step_005	22	19	22	2
Adapter_230005421_step_010	21	3	21	2
Cover_230030276_step_001	1	1	1	1
Cover_230030276_step_029	2	2	2	1
Cover_230030276_step_057	1	1	1	1
Cover_230039626_step_001	11	11	11	1
Cover_230039626_step_011	22	21	22	2
Cover_230039626_step_023	2	2	2	1
Flange_230036437_step_001	0	0	0	1
Flange_230036437_step_016	17	17	17	1
Flange_230036437_step_032	17	17	17	1
Flange_230017072_step_001	3	2	3	2
Flange_230017072_step_018	3	3	3	1
Flange_230017072_step_036	3	3	3	1
Housing_230044255_step_001	0	0	0	1
Housing_230044255_step_063	1	1	1	1
Housing_230044255_step_125	2	2	2	1
Housing_230018316_step_001	1	1	1	1
Housing_230018316_step_062	2	2	2	1
Housing_230018316_step_125	2	2	2	1
Oring_230016675_step_001	79	79	79	1
Oring_231033780_step_001	69	69	69	1

Anhang Q.6 Ergebnisse auf latenten Vektoren AE_ZW

Input-Komponente	Suche nach finalen Komponenten mit latenten Vektoren AE_ZW			
	$ S_{sim,i} $ für $\varepsilon = 0,3$			
	Suche ohne Cluster	1 Cluster	2 Cluster	Benötigte Cluster um gleiches Ergebnis zu erzielen
Adapter_230021093_step_001	68	43	68	2
Adapter_230021093_step_007	67	43	67	2
Adapter_230021093_step_014	71	46	71	2
Adapter_230005421_step_001	0	0	0	1
Adapter_230005421_step_005	34	34	34	1
Adapter_230005421_step_010	34	34	34	1
Cover_230030276_step_001	0	0	0	1
Cover_230030276_step_029	1	1	1	1
Cover_230030276_step_057	1	1	1	1
Cover_230039626_step_001	3	3	3	1
Cover_230039626_step_011	14	14	14	1
Cover_230039626_step_023	2	0	2	2
Flange_230036437_step_001	0	0	0	1
Flange_230036437_step_016	17	17	17	1
Flange_230036437_step_032	17	17	17	1
Flange_230017072_step_001	0	0	0	1
Flange_230017072_step_018	3	3	3	1
Flange_230017072_step_036	3	3	3	1
Housing_230044255_step_001	0	0	0	1
Housing_230044255_step_063	2	2	2	1
Housing_230044255_step_125	2	2	2	1
Housing_230018316_step_001	0	0	0	1
Housing_230018316_step_062	2	2	2	1
Housing_230018316_step_125	2	2	2	1
Oring_230016675_step_001	126	126	126	1
Oring_231033780_step_001	112	112	112	1

Anhang Q.7 Ergebnisse der Ähnlichkeitssuche für Flansch 230017072















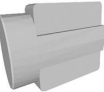









		Menge der ähnlichsten Modelle $S_{sim,i}$ des Bauteilzustandes der Komponenten i			Ähnlichstes Modell außerhalb von $S_{sim,i}$		
		Top 3 der ähnlichsten Modelle aus $S_{sim,i}$			$ S_{sim,i} $ $\min \{ \bar{x}_j d_x(\bar{x}_i, \bar{x}_j) > \epsilon \}$		
Bauteilzustand Komponente i		Latente Vektoren AE_Final	Housing_230005691 (89,93 %)	Adapter_230006115 (88,25 %)	Housing_230005692 (86,22 %)	3	
		Latente Vektoren AE_ZW	X			0	Adapter_230006115 (72,15 %) 
		Latente Vektoren AE_Final	Flange_230035164 (94,75 %)	Flange_230017667 (92,08 %)	Flange_230017133 (91,83 %)	3	Adapter_230006115 (84,68 %) 
		Latente Vektoren AE_ZW	Flange_230035164 (97,43 %)	Flange_230017667 (90,27 %)	Flange_230017133 (89,99 %)	3	Flange_230015448 (78,30 %) 
		Latente Vektoren AE_Final	Flange_230035164 (98,95 %)	Flange_230017133 (94,87 %)	Flange_230017667 (94,59 %)	3	Adapter_230006115 (82,86 %) 
		Latente Vektoren AE_ZW	Flange_230035164 (98,29 %)	Flange_230017667 (90,34 %)	Flange_230017133 (90,08 %)	3	Flange_230015448 (77,37 %) 

Abbildung Q-5: Ergebnisse auf latenten Vektoren von „AE_Final“ und „AE_ZW“ für Flansch 230017072. Gleiche Modelle je Bauteilzustand sind farblich hervorgehoben.

Modelle von verschiedenen Positionen der ähnlichsten Modelle					
Top 3 der ähnlichsten Modelle				Position 4	
Bauteilzustand Komponente I		Adapter_230006161 (97,12 %)	Adapter_230043870 (97,09 %)	Adapter_230006153 (96,68 %)	Adapter_230006154 (96,29 %)
					
		Adapter_230004879 (97,08 %)	Adapter_230043870 (96,89 %)	Adapter_230006153 (96,89 %)	Flange_230006167 (96,83 %)
					
		Flange_230035164 (100,00 %)	Flange_230005963 (95,83 %)	Adapter_230004879 (94,86 %)	Flange_230006167 (94,58 %)
					


 Übereinstimmung innerhalb der Top 3 Ergebnisse mit der Suche auf Basis der latenten Vektoren AE_ZW

Abbildung Q-6: Ergebnisse der Suche über Metainformationen für Flansch 230017072. Gleiche Teile sind farblich hervorgehoben.

Anhang Q.8 Ergebnisse der Ähnlichkeitssuche für Gehäuse 230044255
















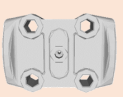






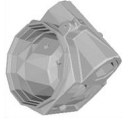





		Menge der ähnlichsten Modelle $S_{sim,i}$ des Bauteilzustandes der Komponenten i		Ähnlichstes Modell außerhalb von $S_{sim,i}$						
		$S_{sim,i}$		$ S_{sim,i} $	$\min \{ \bar{x}_i^* \mid d_2(\bar{x}_i, \bar{x}_j^*) > \epsilon \}$					
Bauteilzustand Komponente i		Latente Vektoren AE_Final	X		0	Cover_230037451 (85,10 %) 				
		Latente Vektoren AE_ZW			0	Adapter_230034621 (76,21 %) 				
		Latente Vektoren AE_Final			Housing_230014922 (87,80 %) 	X		1	Housing_230019399 (85,47 %) 	
		Latente Vektoren AE_ZW			Flange_230005963 (96,24 %) 			Flange_230036423 (89,14 %) 	2	Flange_230036423 (58,55 %) 
		Latente Vektoren AE_Final			Housing_230014922 (88,00 %) 			Housing_230019399 (85,47 %) 	2	Housing_230034945 (83,08 %) 
		Latente Vektoren AE_ZW			Housing_230014922 (98,45 %) 			Housing_230019399 (93,70 %) 	2	Flange_230036423 (55,13 %) 

Abbildung Q-7: Ergebnisse auf latenten Vektoren von „AE_Final“ und „AE_ZW“ für Gehäuse 230044255. Gleiche Teile je Bauteilzustand sind farblich hervorgehoben.

		Modelle von verschiedenen Positionen der ähnlichsten Modelle		
		Top 3 der ähnlichsten Modelle		
Bauteilzustand Komponente i		ORing_230030308 (99,60 %)	ORing_230035731 (98,82 %)	ORing_230018702 (98,70 %)
				
		Adapter_230016768 (92,85 %)	Housing_230040774 (91,92 %)	Housing_231029768 (91,92 %)
				
		Housing_230014922 (99,28 %)	Housing_230019399 (98,95 %)	Housing_230034945 (93,28 %)
				


 Übereinstimmung innerhalb der Top 3 Ergebnisse mit der Suche auf Basis der latenten Vektoren AE_ZW

Abbildung Q-8: Ergebnisse der Suche über Metainformationen für Gehäuse 230044255

Anhang Q.9 Ergebnisse der Ähnlichkeitssuche für Gehäuse 230018316





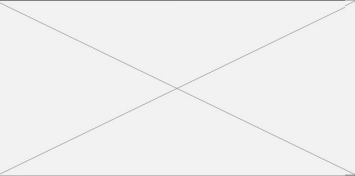









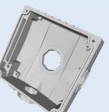







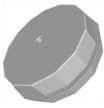
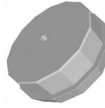







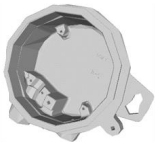
		Menge der ähnlichsten Modelle $S_{sim,i}$ des Bauteilzustandes der Komponenten i		$ S_{sim,i} $	Ähnlichstes Modell außerhalb von $S_{sim,i}$	
		$S_{sim,i}$			$\min \{ \bar{x}_j \mid d_2(\bar{x}_i, \bar{x}_j) > \epsilon \}$	
Bauteilzustand Komponente i		Latente Vektoren AE_Final	Housing_230005692 (90,72 %) 		1 Adapter_230006115 (84,73 %) 	
		Latente Vektoren AE_ZW			0 Adapter_230006115 (68,80 %) 	
		Latente Vektoren AE_Final	Housing_230030275 (93,84 %) 	Housing_230038654 (92,42 %) 	2	Cover_230038656 (85,88 %) 
		Latente Vektoren AE_ZW	Housing_230038654 (93,49 %) 	Housing_230030275 (92,48 %) 		2 Cover_230045528 (66,52 %) 
		Latente Vektoren AE_Final	Housing_230038654 (97,33 %) 	Housing_230030275 (95,97 %) 	2	Cover_230038656 (81,99 %) 
		Latente Vektoren AE_ZW	Housing_230038654 (97,23 %) 	Housing_230030275 (96,16 %) 		2 Cover_230045528 (64,79 %) 

Abbildung Q-9: Ergebnisse auf latenten Vektoren von „AE_Final“ und „AE_ZW“ für Gehäuse 230018316. Gleiche Teile je Bauteilzustand sind farblich hervorgehoben.

		Modelle von verschiedenen Positionen der ähnlichsten Modelle		
		Top 3 der ähnlichsten Modelle		
Bauteilzustand Komponente i		Adapter_230006158 (93,85 %)	Adapter_230034136 (92,76 %)	Adapter_230034134 (91,49 %)
				
		Cover_230013040 (95,76 %)	Cover_230018317 (95,38 %)	Cover_230038792 (94,95 %)
				
		Housing_230030275 (93,98 %)	Housing_230038654 (92,87 %)	Housing_230018684 (88,78 %)
				


 Übereinstimmung innerhalb der Top 3 Ergebnisse mit der Suche auf Basis der latenten Vektoren AE_ZW

Abbildung Q-10: Ergebnisse der Suche über Metainformationen für Gehäuse 230018316

Anhang Q.10 Ergebnisse der Ähnlichkeitssuche für Deckel 230030276









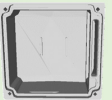
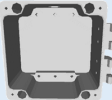





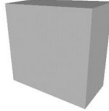


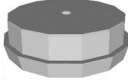
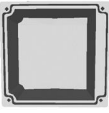
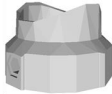
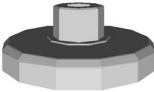
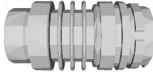
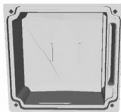


		Menge der ähnlichsten Modelle $S_{sim,i}$ des Bauteilzustandes der Komponenten i		$S_{sim,i}$	Ähnlichstes Modell außerhalb von $S_{sim,i}$		
		$S_{sim,i}$			$\min \{\bar{x}_j \mid d_2(\bar{x}_i, \bar{x}_j) > \epsilon\}$		
Bauteilzustand Komponente i		Latente Vektoren AE_Final	Housing_230005692 (87,14 %) 	X	1	Adapter_230006115 (74,22 %) 	
		Latente Vektoren AE_ZW			0	Housing_230005692 (66,10 %) 	
		Latente Vektoren AE_Final	Cover_230018317 (93,40 %) 	Housing_230035183 (89,47 %) 	X	2	Cover_230035182 (76,08 %) 
		Latente Vektoren AE_ZW	Cover_230018317 (95,52 %) 			1	Housing_230035183 (77,95 %) 
		Latente Vektoren AE_Final	Cover_230018317 (91,50 %) 		X	1	Housing_230035183 (85,06 %) 
		Latente Vektoren AE_ZW	Cover_230018317 (91,94 %) 			1	Housing_230035183 (79,22 %) 

Abbildung Q-11: Ergebnisse auf latenten Vektoren von „AE_Final“ und „AE_ZW“ für Deckel 230030276. Gleiche Teile je Bauteilzustand sind farblich hervorgehoben.

		Modelle von verschiedenen Positionen der ähnlichsten Modelle		
		Top 3 der ähnlichsten Modelle		
Bauteilzustand Komponente i		Adapter_230044893 (91,49 %)	Adapter_230006158 (87,90 %)	Adapter_230034136 (87,13 %)
				
		Adapter_230038897 (95,79 %)	Cover_230002736 (94,79 %)	Adapter_230037259 (94,61 %)
				
		Cover_230018317 (93,44 %)	Cover_230013040 (92,87 %)	Cover_230006848 (92,75 %)
				



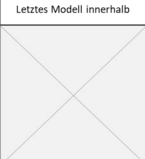















 Übereinstimmung innerhalb der Top 3 Ergebnisse mit der Suche auf Basis der latenten Vektoren

Abbildung Q-12: Ergebnisse der Suche über Metainformationen für Deckel 230030276

Anhang Q.11 Ergebnisse der Ähnlichkeitssuche für Deckel 230039626

		Menge der ähnlichsten Modelle $S_{sim,i}$ des Bauteilzustandes der Komponente i				[$S_{sim,i}$]	Ähnlichstes Modell außerhalb von $S_{sim,i}$ $\min(\vec{x}_i - d_2(\vec{x}_i, \vec{x}_j)) > \epsilon$
		Top 3 der ähnlichsten Modelle aus $S_{sim,i}$			Unähnlichstes Modell aus $S_{sim,i}$		
Bauteilzustand Komponente i	AE_Final	Adapter_230034136 (89,97 %)	Adapter_230019195 (88,91 %)	Adapter_230034134 (88,90 %)	Cover_230038884 (86,30 %)	11	Cover_230042769 (85,89 %)
		Cover_230046107 (84,64 %)	Cover_230038884 (84,51 %)	Flange_230006167 (82,61 %)			
	AE_ZW					3	
	AE_Final	Adapter_230030776 (95,78 %)	Adapter_230005677 (94,35 %)	Oring_231000656 (90,07 %)	Adapter_230044924 (86,20 %)	22	Oring_230037201 (85,98 %)
	AE_ZW						14
AE_Final	Cover_230035401 (89,11 %)	Cover_230007698 (88,26 %)			2	Housing_230000879 (80,42 %)	
AE_ZW						Housing_230000879 (80,00 %)	
AE_Final	Cover_230035401 (89,66 %)	Cover_230007698 (89,44 %)					
AE_ZW							

Abbildung Q-13: Ergebnisse auf latenten Vektoren von „AE_Final“ und „AE_ZW“ für Deckel 230039626. Gleiche Teile je Bauteilzustand sind farblich hervorgehoben.

Modelle von verschiedenen Positionen der ähnlichsten Modelle					
Bauteilzustand Komponente i	Top 3 der ähnlichsten Modelle			Letztes Modell innerhalb	Erstes Modell außerhalb
		Flange_230006167 (95,94 %)	Flange_230041312 (95,68 %)	Flange_230006168 (94,25 %)	
					
		Cover_230038884 (96,61 %)	Flange_230045157 (96,60 %)	Adapter_230006179 (96,02 %)	Adapter_230018056 (93,67 %)
					
	Cover_230035402 (98,16 %)	Cover_230007699 (98,16 %)	Cover_230035401 (96,81 %)		
					

 Übereinstimmung innerhalb der Top 3 Ergebnisse mit der Suche auf Basis der latenten Vektoren

Abbildung Q-14: Ergebnisse der Suche über Metainformationen für Deckel 230039626. Dargestellt ist je Zwischenzustand der Komponente i das Bauteil, dessen Position dem unähnlichsten Teil innerhalb und dem ersten Teil außerhalb $S_{sim,i}$ basierend auf den latenten Vektoren von AE_{ZW} entspricht.

Anhang Q.12 Ergebnisse der Ähnlichkeitssuche für Adapter 230021093





































		Menge der ähnlichsten Modelle $S_{sim,i}$ des Bauteilzustandes der Komponente i				Ähnlichstes Modell außerhalb von $S_{sim,i}$	
		Top 3 der ähnlichsten Modelle aus $S_{sim,i}$			Unähnlichstes Modell aus $S_{sim,i}$	$ S_{sim,i} $	$\min \{\bar{V}_j d_2(\bar{V}_i, \bar{V}_j) > \epsilon\}$
Bauteilzustand Komponente i	 Latente Vektoren AE_Final	Adapter_230030777 (97,93 %)	Adapter_230020947 (95,51 %)	Adapter_230030783 (96,05 %)	Cover_230009417 (86,42 %)	22	 Cover_230009417 (86,08 %)
							
	 Latente Vektoren AE_ZW	Adapter_230030777 (96,96 %)	Adapter_230020947 (94,25 %)	Adapter_230030783 (93,26 %)	Adapter_230015330 (81,05 %)	68	 Adapter_230016988 (80,85 %)
							
	 Latente Vektoren AE_Final	Adapter_230030777 (97,35 %)	Adapter_230020947 (95,98 %)	Adapter_230030783 (95,64 %)	Adapter_230044372 (86,83 %)	20	 Adapter_230002358 (86,06 %)
							
 Latente Vektoren AE_ZW	Adapter_230030777 (96,61 %)	Adapter_230020947 (93,92 %)	Adapter_230030783 (92,70 %)	Adapter_230015330 (80,93 %)	67	 Adapter_230019956 (80,87 %)	
							
 Latente Vektoren AE_Final	Adapter_230030777 (96,55 %)	Adapter_230020947 (96,55 %)	Adapter_230030783 (94,74 %)	Adapter_230017533 (86,18 %)	22	 Cover_230009417 (85,98 %)	
							
 Latente Vektoren AE_ZW	Adapter_230030777 (96,80 %)	Adapter_230020947 (93,84 %)	Adapter_230030783 (92,70 %)	Adapter_230018664 (80,98 %)	71	 Adapter_230018287 (80,90 %)	
							

Abbildung Q-15: Ergebnisse auf latenten Vektoren von „AE_Final“ und „AE_ZW“ für Adapter 230021093. Gleiche Teile sind je Bauteilzustand farblich hervorgehoben.



















Modelle von verschiedenen Positionen der ähnlichsten Modelle						
Top 3 der ähnlichsten Modelle				Letztes Modell innerhalb	Erstes Modell außerhalb	
Bauteilzustand Komponente i		Housing_230007316 (94,38 %)	Adapter_230039371 (93,65 %)	Adapter_230002357 (93,25 %)	Oring_230018699 (81,17 %)	Housing_230005692 (80,94 %)
						
		Housing_230007316 (94,40 %)	Adapter_230003065 (94,12 %)	Adapter_230039371 (93,95 %)	Oring_231030813 (81,38 %)	Oring_230030605 (81,32 %)
						
		Adapter_230003065 (95,07 %)	Adapter_230039371 (94,00 %)	Housing_230007316 (93,49 %)	Cover_230046239 (80,52 %)	Oring_231030813 (80,50 %)
						

Abbildung Q-16: Ergebnisse der Suche über Metainformationen für Adapter 230021093. Dargestellt ist je Zwischenzustand der Komponenten i das Bauteil, dessen Position dem unähnlichsten Teil innerhalb und dem ersten Teil außerhalb $S_{Sim,i}$ basierend auf den latenten Vektoren von AE_ZW entspricht. Gleiche Teile sind farblich hervorgehoben.

Anhang Q.13 Häufigkeit der Distanzen zwischen den latenten Vektoren der finalen Modelle je Bauteilklasse

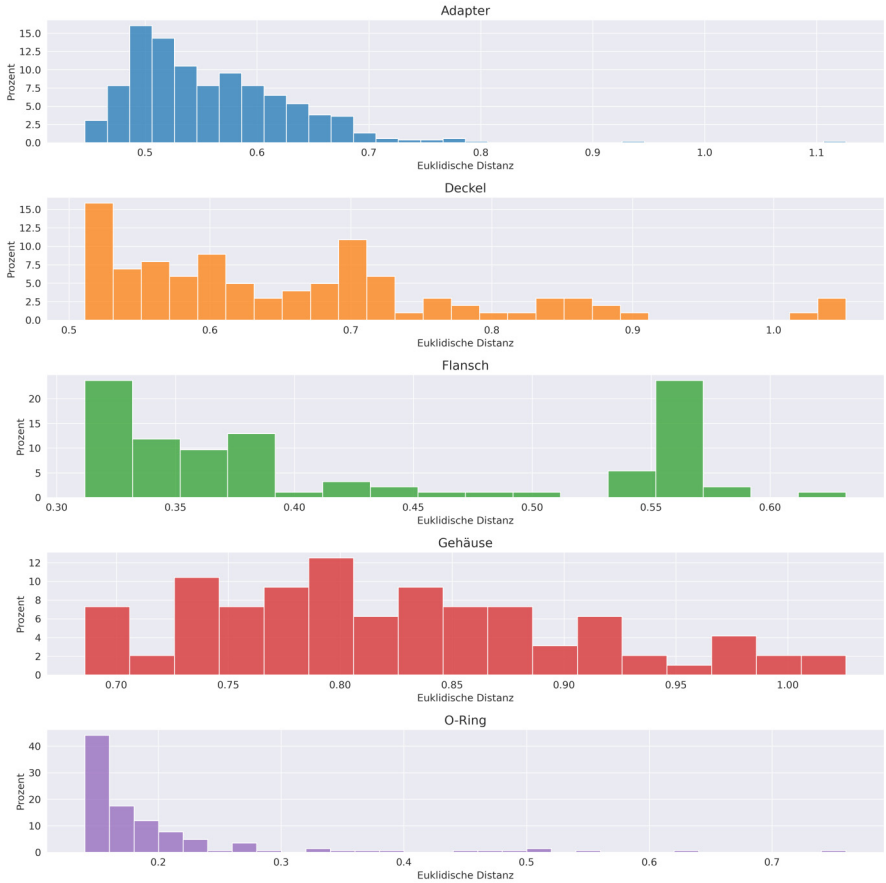



Abbildung Q-17: Prozentuale Häufigkeit der euklidischen Distanzen zwischen den latenten Vektoren der finalen Modelle je Bauteilklasse

Anhang Q.14 Ergebnisse der Ähnlichkeitssuche mit Metainformationen für O-Ring 230016675

Input-Komponente:  O-Ring 230016675
 Material: Stahl
 Abmessungen: 21 x 21 x 1,8 [mm]
 Volumen: 182 [mm³]






	Gewichtung	Top 3 der ähnlichsten Modelle				Unähnlichstes Modell aus $S_{sim,i}$	$ S_{sim,i} $
		O-Ring 231031993	O-Ring 231036104	O-Ring 230007805	O-Ring 230006863		
Umsetzungsstufe 1	$\alpha_{geo} = 1$ $\alpha_{mat} = 0$ $\alpha_{abm} = 0$ $\alpha_{voi} = 0$	Geometrie: 98,5% Material: Stahl 100% Abmessung: 21 x 21 x 1,8 100% Volumen: 182 100%	Geometrie: 98,4% Material: Kunststoff 0% Abmessung: 36 x 35 x 3 97,6% Volumen: 1022 99,9%	Geometrie: 98,3% Material: Stahl 100% Abmessung: 34 x 33 x 3 97,6% Volumen: 946 99,9%	Geometrie: 98,3% Material: Stahl 100% Abmessung: 27 x 26 x 2,2 97,6% Volumen: 415 99,9%		126
							
Umsetzungsstufe 2	$\alpha_{geo} = 0,25$ $\alpha_{mat} = 0,25$ $\alpha_{abm} = 0,25$ $\alpha_{voi} = 0,25$	Geometrie: 98,5% Material: Stahl 100% Abmessung: 21 x 21 x 1,8 100% Volumen: 182 100%	Geometrie: 97,7% Material: Stahl 100% Abmessung: 19 x 18 x 1,5 99,9% Volumen: 135 99,9%	Geometrie: 97,6% Material: Stahl 100% Abmessung: 25 x 24 x 3,3 99,3% Volumen: 392 99,9%	Geometrie: 83,6% Material: Kunststoff 0% Abmessung: 24 x 23 x 2,2 76,8% Volumen: 1342 99,9%		
		Gewichtete Ähnlichkeit: 99,6%	Gewichtete Ähnlichkeit: 99,3%	Gewichtete Ähnlichkeit: 99,2%	Gewichtete Ähnlichkeit: 65,1%		

Abbildung Q-18: Ergebnisse der Ähnlichkeitssuche auf den latenten Vektoren von „AE_ZW“ über Umsetzungsstufe 1 und Umsetzungsstufe 2 für O-Ring 230016675. Gleiche Teile sind farblich hervorgehoben.

Anhang Q.15 Ergebnisse der Ähnlichkeitssuche mit Metainformationen für Adapter 23005421 (mittlerer Zwischenzustand)

Input-Komponente:  Adapter 230005421 (Zwischenzustand)
 Material: Stahl
 Abmessungen: 50 x 34 x 34 [mm]
 Volumen: 20.150 [mm³]

Gewichtung	Top 3 der ähnlichsten Modelle				Unähnlichstes Modell aus $S_{sim,l}$	$ S_{sim,l} $
Umsetzungsstufe 1 $\alpha_{geo} = 1$ $\alpha_{mat} = 0$ $\alpha_{abm} = 0$ $\alpha_{vol} = 0$	Adapter 230005420	Adapter 230005419	Adapter 230038868	Adapter 230005531		
	Geometrie: 97,9% Material: Stahl 100% Abmessung: 50 x 34 x 34 100% Volumen: 19.199 99,9%	Geometrie: 96,3% Material: Stahl 100% Abmessung: 48 x 34 x 34 99,8% Volumen: 18.953 99,9%	Geometrie: 92,3% Material: Stahl 100% Abmessung: 45 x 34 x 34 100% Volumen: 17.408 99,9%	Geometrie: 81,0% Material: Stahl 100% Abmessung: 69 x 41 x 41 97,1% Volumen: 40.253 99,7%		
Umsetzungsstufe 2 $\alpha_{geo} = 0,2$ $\alpha_{mat} = 0,2$ $\alpha_{abm} = 0,5$ $\alpha_{vol} = 0,1$	Adapter 230005420	Adapter 230005419	Adapter 230005417	Adapter 230003665		34
	Geometrie: 97,3% Material: Stahl 100% Abmessung: 50 x 34 x 34 100% Volumen: 19.199 99,9%	Geometrie: 96,3% Material: Stahl 100% Abmessung: 48 x 34 x 34 99,8% Volumen: 18.953 99,9%	Geometrie: 92,0% Material: Stahl 100% Abmessung: 45 x 34 x 34 99,6% Volumen: 17.784 99,9%	Geometrie: 82,9% Material: Kunststoff 0% Abmessung: 98 x 63 x 60 90,5% Volumen: 77.083 99,2%		
	Gewichtete Ähnlichkeit: 99,6%	Gewichtete Ähnlichkeit: 99,2%	Gewichtete Ähnlichkeit: 98,2%	Gewichtete Ähnlichkeit: 71,75%		

Abbildung Q-19: Ergebnisse der Ähnlichkeitssuche auf den latenten Vektoren von „AE_ZW“ über Umsetzungsstufe 1 und Umsetzungsstufe 2 für Adapter 23005421 (mittlerer Zwischenzustand)

Aus Darstellungszwecken sind in Abbildung Q-19 keine Nachkommastellen der Metainformationen angegeben. Oben dargestellt sind die Suchergebnisse, bei denen rein die Geometrie für die Ähnlichkeit ausschlaggebend ist, unten erfolgt die Suche anhand der gewichteten Ähnlichkeit von Geometrie, Material, Abmessungen und Volumen. Die beiden geometrisch ähnlichsten Modelle sind auch bezüglich der gewichteten Gesamtähnlichkeit am ähnlichsten. Der ursprünglich auf Platz 3 positionierte Adapter wird jedoch auf Rang 24 verschoben. Die neue Position 3 war zuvor auf Rang 5.

Anhang Q.16 Weitere Adapter aus Kunststoff neben Adapter 230021093



Abbildung Q-20: Weitere Adapter aus Kunststoff im betrachteten Datensatz neben Adapter 230021093

Anhang Q.17 Bestimmung von ϵ_{DB} für die Bewertung produktionsrelevanter Produkteigenschaften

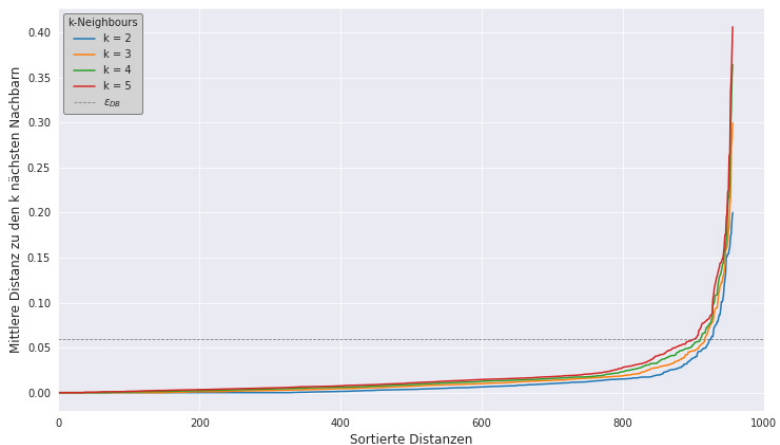


Abbildung Q-21: Bestimmung von ϵ_{DB} nach Ester et al. (1996) für die Bewertung der produktionsrelevanten Produkteigenschaften in Form von Abmessungen und Dichte auf Basis der euklidischen Distanz

Anhang R Zusammenhang Fenstergröße mit Häufigkeit eines Zwischenzustandes

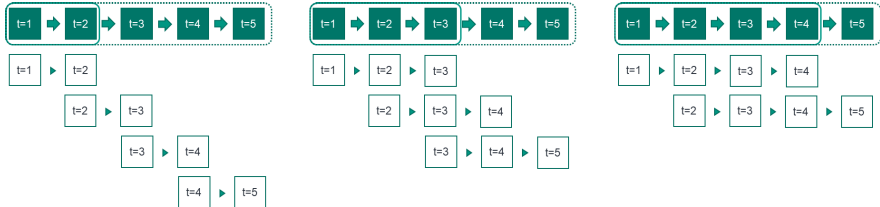


Abbildung R-1: Anzahl der Sequenzabschnitte in Abhängigkeit der Fenstergröße sowie Häufigkeit eines Zwischenzustandes in Abhängigkeit seiner Position sowie der Fenstergröße

Anhang S Sequenzanalyse mittels RNN

Anhang S.1 Optimale Parametrisierung des RNN

Tabelle S-1: Optimale Parametrisierungen der Hyperparameter des RNNs

Breite	Tiefe	Zelltyp	Metainformationen
512	1	GRU	Ja

Anhang S.2 Beispielhafter Adapter mit kürzerer Extrusion

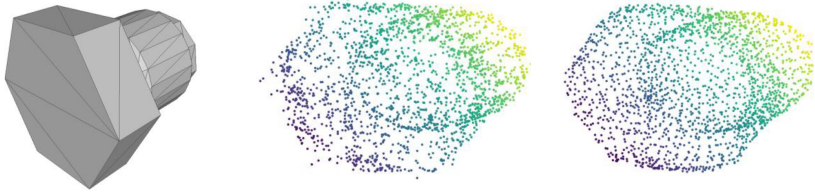


Abbildung S-1: Beispielhafter Adapter mit kürzerer Extrusion im zweiten Zwischenzustand als CAD-Modell (a), generierte Punktwolke (b) und durch den Autoencoder rekonstruierte Punktwolke (c)

Anhang S.3 Visualisierung Vorhersage Gehäuse 230044255




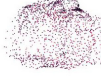










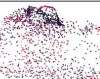
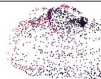
Zeitpunkt t	Tatsächlicher Zwischenzustand zum Zeitpunkt t			Vorhersage des Zwischenzustandes für Zeitpunkt t
	CAD-Modell	Generierte Punktwolke	Rekonstruierte Punktwolke	
27				
28				
29				
30				

Abbildung S-2: Visualisierung der Vorhersage für Gehäuse 230044255

Anhang T Bereiche produktionsrelevanter Produkteigenschaften für Adapter 230005421

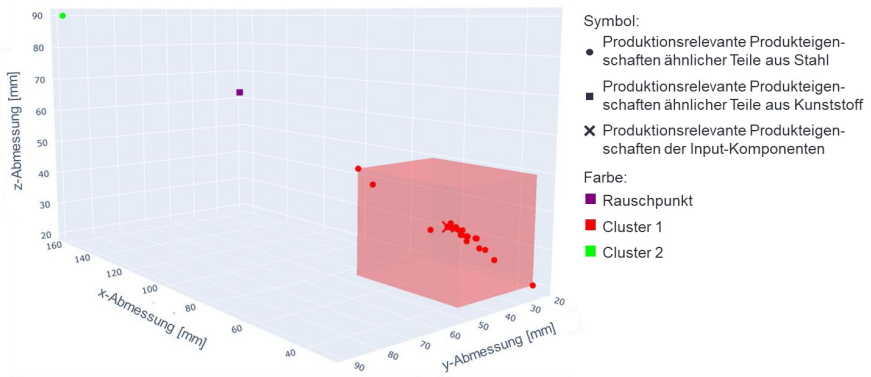


Abbildung T-1: Bereiche produktionsrelevanter Produkteigenschaften auf Basis der zwei gefundenen Cluster für Adapter 230005421 (Zwischenzustand 4)

Forschungsberichte aus dem **wbk**
Institut für Produktionstechnik
Karlsruher Institut für Technologie (KIT)

Bisher erschienene Bände:

Band 0

Dr.-Ing. Wu Hong-qi

Adaptive Volumenstromregelung mit Hilfe von drehzahleregelten Elektroantrieben

Band 1

Dr.-Ing. Heinrich Weiß

**Fräsen mit Schneidkeramik - Verhalten des System
Werkzeugmaschine-Werkzeug-Werkstück und Prozessanalyse**

Band 2

Dr.-Ing. Hans-Jürgen Stierle

Entwicklung und Untersuchung hydrostatischer Lager für die Axialkolbenmaschine

Band 3

Dr.-Ing. Herbert Hörner

Untersuchung des Geräuschverhaltens druckeregelter Axialkolbenpumpen

Band 4

Dr.-Ing. Rolf-Dieter Brückbauer

Digitale Drehzahlregelung unter der besonderen Berücksichtigung von Quantisierungseffekten

Band 5

Dr.-Ing. Gerhard Staiger

Graphisch interaktive NC-Programmierung von Drehteilen im Werkstattbereich

Band 6

Dr.-Ing. Karl Peters

Ein Beitrag zur Berechnung und Kompensation von Positionierfehlern an Industrierobotern

Band 7

Dr.-Ing. Paul Stauss

Automatisierte Inbetriebnahme und Sicherung der Zuverlässigkeit und Verfügbarkeit numerisch gesteuerter Fertigungseinrichtungen

Band 8

Dr.-Ing. Günter Möckesch

Konzeption und Realisierung eines strategischen, integrierten Gesamtplanungs- und -bearbeitungssystems zur Optimierung der Drehteilorganisation für auftragsbezogene Drehereien

Band 9

Dr.-Ing. Thomas Oestreicher

Rechnergestützte Projektierung von Steuerungen

Band 10

Dr.-Ing. Thomas Selinger

Teilautomatisierte werkstattnahe NC-Programmerstellung im Umfeld einer integrierten Informationsverarbeitung

Band 11

Dr.-Ing. Thomas Buchholz

Prozessmodell Fräsen, Rechnerunterstützte Analyse, Optimierung und Überwachung

Band 12

Dr.-Ing. Bernhard Reichling

Lasergestützte Positions- und Bahnvermessung von Industrierobotern

Band 13

Dr.-Ing. Hans-Jürgen Lesser

Rechnergestützte Methoden zur Auswahl anforderungsgerechter Verbindungselemente

Band 14

Dr.-Ing. Hans-Jürgen Lauffer

Einsatz von Prozessmodellen zur rechnerunterstützten Auslegung von Räumwerkzeugen

Band 15

Dr.-Ing. Michael C. Wilhelm

Rechnergestützte Prüfplanung im Informationsverbund moderner Produktionssysteme

Band 16

Dr.-Ing. Martin Ochs

Entwurf eines Programmsystems zur wissensbasierten Planung und Konfigurierung

Band 17

Dr.-Ing. Heinz-Joachim Schneider

Erhöhung der Verfügbarkeit von hochautomatisierten Produktionseinrichtungen mit Hilfe der Fertigungsleittechnik

Band 18

Dr.-Ing. Hans-Reiner Ludwig

Beanspruchungsanalyse der Werkzeugschneiden beim Stirnplanfräsen

Band 19

Dr.-Ing. Rudolf Wieser

Methoden zur rechnergestützten Konfigurierung von Fertigungsanlagen

Band 20

Dr.-Ing. Edgar Schmitt

Werkstattsteuerung bei wechselnder Auftragsstruktur

Band 21

Dr.-Ing. Wilhelm Enderle

Verfügbarkeitssteigerung automatisierter Montagesysteme durch selbsttätige Behebung prozessbedingter Störungen

Band 22

Dr.-Ing. Dieter Buchberger

Rechnergestützte Strukturplanung von Produktionssystemen

Band 23

Prof. Dr.-Ing. Jürgen Fleischer

Rechnerunterstützte Technologieplanung für die flexibel automatisierte Fertigung von Abkantteilen

Band 24

Dr.-Ing. Lukas Loeffler

Adaptierbare und adaptive Benutzerschnittstellen

Band 25

Dr.-Ing. Thomas Friedmann

Integration von Produktentwicklung und Montageplanung durch neue rechnergestützte Verfahren

Band 26

Dr.-Ing. Robert Zurrin

Variables Formhonen durch rechnergestützte Hornprozesssteuerung

Band 27

Dr.-Ing. Karl-Heinz Bergen

Langhub-Innenrundhonen von Grauguss und Stahl mit einem elektromechanischem Vorschubsystem

Band 28

Dr.-Ing. Andreas Liebisch

Einflüsse des Festwalzens auf die Eigenspannungsverteilung und die Dauerfestigkeit einsatzgehärteter Zahnräder

Band 29

Dr.-Ing. Rolf Ziegler

Auslegung und Optimierung schneller Servopumpen

Band 30

Dr.-Ing. Rainer Bartl

Datenmodellgestützte Wissensverarbeitung zur Diagnose und Informationsunterstützung in technischen Systemen

Band 31

Dr.-Ing. Ulrich Golz

Analyse, Modellbildung und Optimierung des Betriebsverhaltens von Kugelgewindetrieben

Band 32

Dr.-Ing. Stephan Timmermann

Automatisierung der Feinbearbeitung in der Fertigung von Hohlformwerkzeugen

Band 33

Dr.-Ing. Thomas Noe

Rechnergestützter Wissenserwerb zur Erstellung von Überwachungs- und Diagnoseexpertensystemen für hydraulische Anlagen

Band 34

Dr.-Ing. Ralf Lenschow

Rechnerintegrierte Erstellung und Verifikation von Steuerungsprogrammen als Komponente einer durchgängigen Planungsmethodik

Band 35

Dr.-Ing. Matthias Kallabis

Räumen gehärteter Werkstoffe mit kristallinen Hartstoffen

Band 36

Dr.-Ing. Heiner-Michael Honeck

Rückführung von Fertigungsdaten zur Unterstützung einer fertigungsgerechten Konstruktion

Band 37

Dr.-Ing. Manfred Rohr

Automatisierte Technologieplanung am Beispiel der Komplettbearbeitung auf Dreh-/Fräszellen

Band 38

Dr.-Ing. Martin Steuer

Entwicklung von Softwarewerkzeugen zur wissensbasierten Inbetriebnahme von komplexen Serienmaschinen

Band 39

Dr.-Ing. Siegfried Beichter

Rechnergestützte technische Problemlösung bei der Angebotserstellung von flexiblen Drehzellen

Band 40

Dr.-Ing. Thomas Steitz

Methodik zur marktorientierten Entwicklung von Werkzeugmaschinen mit Integration von funktionsbasierter Strukturierung und Kostenschätzung

Band 41

Dr.-Ing. Michael Richter

Wissensbasierte Projektierung elektrohydraulischer Regelungen

Band 42

Dr.-Ing. Roman Kuhn

Technologieplanungssystem Fräsen. Wissensbasierte Auswahl von Werkzeugen, Schneidkörpern und Schnittbedingungen für das Fertigungsverfahren Fräsen

Band 43

Dr.-Ing. Hubert Klein

Rechnerunterstützte Qualitätssicherung bei der Produktion von Bauteilen mit frei geformten Oberflächen

Band 44

Dr.-Ing. Christian Hoffmann

Konzeption und Realisierung eines fertigungsintegrierten Koordinatenmessgerätes

Band 45

Dr.-Ing. Volker Frey

Planung der Leittechnik für flexible Fertigungsanlagen

Band 46

Dr.-Ing. Achim Feller

Kalkulation in der Angebotsphase mit dem selbsttätig abgeleiteten Erfahrungswissen der Arbeitsplanung

Band 47

Dr.-Ing. Markus Klaiber

Produktivitätssteigerung durch rechnerunterstütztes Einfahren von NC-Programmen

Band 48

Dr.-Ing. Roland Minges

Verbesserung der Genauigkeit beim fünffachsigen Fräsen von Freiformflächen

Band 49

Dr.-Ing. Wolfgang Bernhart

Beitrag zur Bewertung von Montagevarianten: Rechnergestützte Hilfsmittel zur kostenorientierten, parallelen Entwicklung von Produkt und Montagesystem

Band 50

Dr.-Ing. Peter Ganghoff

Wissensbasierte Unterstützung der Planung technischer Systeme: Konzeption eines Planungswerkzeuges und exemplarische Anwendung im Bereich der Montagesystemplanung

Band 51

Dr.-Ing. Frank Maier

Rechnergestützte Prozessregelung beim flexiblen Gesenkbiegen durch Rückführung von Qualitätsinformationen

Band 52

Dr.-Ing. Frank Debus

Ansatz eines rechnerunterstützten Planungsmanagements für die Planung in verteilten Strukturen

Band 53

Dr.-Ing. Joachim Weinbrecht

Ein Verfahren zur zielorientierten Reaktion auf Planabweichungen in der Werkstattregelung

Band 54

Dr.-Ing. Gerd Herrmann

Reduzierung des Entwicklungsaufwandes für anwendungsspezifische Zellenrechnersoftware durch Rechnerunterstützung

Band 55

Dr.-Ing. Robert Wassmer

Verschleissentwicklung im tribologischen System Fräsen: Beiträge zur Methodik der Prozessmodellierung auf der Basis tribologischer Untersuchungen beim Fräsen

Band 56

Dr.-Ing. Peter Uebelhoer

Inprocess-Geometriemessung beim Honen

Band 57

Dr.-Ing. Hans-Joachim Schelberg

Objektorientierte Projektierung von SPS-Software

Band 58

Dr.-Ing. Klaus Boes

Integration der Qualitätsentwicklung in featurebasierte CAD/CAM-Prozessketten

Band 59

Dr.-Ing. Martin Schreiber

Wirtschaftliche Investitionsbewertung komplexer Produktionssysteme unter Berücksichtigung von Unsicherheit

Band 60

Dr.-Ing. Ralf Steuernagel

Offenes adaptives Engineering-Werkzeug zur automatisierten Erstellung von entscheidungsunterstützenden Informationssystemen

Band 62

Dr.-Ing. Uwe Schauer

Qualitätsorientierte Feinbearbeitung mit Industrierobotern: Regelungsansatz für die Freiformflächenfertigung des Werkzeug- und Formenbaus

Band 63

Dr.-Ing. Simone Loeper

Kennzahlengestütztes Beratungssystem zur Verbesserung der Logistikleistung in der Werkstattfertigung

Band 64

Dr.-Ing. Achim Raab

Räumen mit hartstoffbeschichteten HSS-Werkzeugen

Band 65,

Dr.-Ing. Jan Erik Burghardt

Unterstützung der NC-Verfahrenskette durch ein bearbeitungs-elementorientiertes, lernfähiges Technologieplanungssystem

Band 66

Dr.-Ing. Christian Tritsch

Flexible Demontage technischer Gebrauchsgüter: Ansatz zur Planung und (teil-)automatisierten Durchführung industrieller Demontageprozesse

Band 67

Dr.-Ing. Oliver Eitrich

Prozessorientiertes Kostenmodell für die entwicklungsbegleitende Vorkalkulation

Band 68

Dr.-Ing. Oliver Wilke

Optimierte Antriebskonzepte für Räummaschinen - Potentiale zur Leistungssteigerung

Band 69

Dr.-Ing. Thilo Sieth

Rechnergestützte Modellierungsmethodik zerspantechnologischer Prozesse

Band 70

Dr.-Ing. Jan Linnenbuenger

Entwicklung neuer Verfahren zur automatisierten Erfassung der geometrischen Abweichungen an Linearachsen und Drehschwenkköpfen

Band 71

Dr.-Ing. Mathias Klimmek

Fraktionierung technischer Produkte mittels eines frei beweglichen Wasserstrahlwerkzeuges

Band 72

Dr.-Ing. Marko Hartel

Kennzahlenbasiertes Bewertungssystem zur Beurteilung der Demontage- und Recyclingeignung von Produkten

Band 73

Dr.-Ing. Jörg Schaupp

Wechselwirkung zwischen der Maschinen- und Hauptspindeltriebsdynamik und dem Zerspanprozess beim Fräsen

Band 74

Dr.-Ing. Bernhard Neisius

Konzeption und Realisierung eines experimentellen Telemanipulators für die Laparoskopie

Band 75

Dr.-Ing. Wolfgang Walter

Erfolgsversprechende Muster für betriebliche Ideenfindungsprozesse. Ein Beitrag zur Steigerung der Innovationsfähigkeit

Band 76

Dr.-Ing. Julian Weber

Ein Ansatz zur Bewertung von Entwicklungsergebnissen in virtuellen Szenarien

Band 77

Dr.-Ing. Dipl. Wirtsch.-Ing. Markus Posur

Unterstützung der Auftragsdurchsetzung in der Fertigung durch Kommunikation über mobile Rechner

Band 78

Dr.-Ing. Frank Fleissner

Prozessorientierte Prüfplanung auf Basis von Bearbeitungsobjekten für die Kleinserienfertigung am Beispiel der Bohr- und Fräsbearbeitung

Band 79

Dr.-Ing. Anton Haberkern

Leistungsfähigere Kugelgewindetriebe durch Beschichtung

Band 80

Dr.-Ing. Dominik Matt

Objektorientierte Prozess- und Strukturinnovation (OPUS)

Band 81

Dr.-Ing. Jürgen Andres

Robotersysteme für den Wohnungsbau: Beitrag zur Automatisierung des Mauerwerkbaus und der Elektroinstallation auf Baustellen

Band 82

Dr.-Ing. Dipl.Wirtschaftsing. Simone Riedmiller

Der Prozesskalender - Eine Methodik zur marktorientierten Entwicklung von Prozessen

Band 83

Dr.-Ing. Dietmar Tilch

Analyse der Geometrieparameter von Präzisionsgewinden auf der Basis einer Least-Squares-Estimation

Band 84

Dr.-Ing. Dipl.-Kfm. Oliver Stiefbold

Konzeption eines reaktionsschnellen Planungssystems für Logistikketten auf Basis von Software-Agenten

Band 85

Dr.-Ing. Ulrich Walter

Einfluss von Kühlschmierstoff auf den Zerspansprozess beim Fräsen: Beitrag zum Prozessverständnis auf Basis von zerspantechnischen Untersuchungen

Band 86

Dr.-Ing. Bernd Werner

Konzeption von teilautonomer Gruppenarbeit unter Berücksichtigung kultureller Einflüsse

Band 87

Dr.-Ing. Ulf Osmers

Projektieren Speicherprogrammierbarer Steuerungen mit Virtual Reality

Band 88

Dr.-Ing. Oliver Doerfel

Optimierung der Zerspantechnik beim Fertigungsverfahren Wälzstossen: Analyse des Potentials zur Trockenbearbeitung

Band 89

Dr.-Ing. Peter Baumgartner

Stufenmethode zur Schnittstellengestaltung in der internationalen Produktion

Band 90

Dr.-Ing. Dirk Vossmann

Wissensmanagement in der Produktentwicklung durch Qualitätsmethodenverbund und Qualitätsmethodenintegration

Band 91

Dr.-Ing. Martin Plass

Beitrag zur Optimierung des Honprozesses durch den Aufbau einer Honprozessregelung

Band 92

Dr.-Ing. Titus Konold

Optimierung der Fünffachsfräsbearbeitung durch eine kennzahlenunterstützte CAM-Umgebung

Band 93

Dr.-Ing. Jürgen Brath

Unterstützung der Produktionsplanung in der Halbleiterfertigung durch risikoberücksichtigende Betriebskennlinien

Band 94

Dr.-Ing. Dirk Geisinger

Ein Konzept zur marktorientierten Produktentwicklung

Band 95

Dr.-Ing. Marco Lanza

Entwurf der Systemunterstützung des verteilten Engineering mit Axiomatic Design

Band 96

Dr.-Ing. Volker Hüntrup

Untersuchungen zur Mikrostrukturierbarkeit von Stählen durch das Fertigungsverfahren Fräsen

Band 97

Dr.-Ing. Frank Reinboth

Interne Stützung zur Genauigkeitsverbesserung in der Inertialmesstechnik: Beitrag zur Senkung der Anforderungen an Inertialsensoren

Band 98

Dr.-Ing. Lutz Trender

Entwicklungsintegrierte Kalkulation von Produktlebenszykluskosten auf Basis der ressourcenorientierten Prozesskostenrechnung

Band 99

Dr.-Ing. Cornelia Kafka

Konzeption und Umsetzung eines Leitfadens zum industriellen Einsatz von Data-Mining

Band 100

Dr.-Ing. Gebhard Selinger

Rechnerunterstützung der informellen Kommunikation in verteilten Unternehmensstrukturen

Band 101

Dr.-Ing. Thomas Windmüller

Verbesserung bestehender Geschäftsprozesse durch eine mitarbeiterorientierte Informationsversorgung

Band 102

Dr.-Ing. Knud Lembke

Theoretische und experimentelle Untersuchung eines bistabilen elektrohydraulischen Linearantriebs

Band 103

Dr.-Ing. Ulrich Thies

Methode zur Unterstützung der variantengerechten Konstruktion von industriell eingesetzten Kleingeräten

Band 104

Dr.-Ing. Andreas Schmälzle

Bewertungssystem für die Generalüberholung von Montageanlagen – Ein Beitrag zur wirtschaftlichen Gestaltung geschlossener Facility- Management- Systeme im Anlagenbau

Band 105

Dr.-Ing. Thorsten Frank

Vergleichende Untersuchungen schneller elektromechanischer Vorschubachsen mit Kugelgewindetrieb

Band 106

Dr.-Ing. Achim Agostini

Reihenfolgeplanung unter Berücksichtigung von Interaktionen: Beitrag zur ganzheitlichen Strukturierung und Verarbeitung von Interaktionen von Bearbeitungsobjekten

Band 107

Dr.-Ing. Thomas Barrho

Flexible, zeitfenstergesteuerte Auftragseinplanung in segmentierten Fertigungsstrukturen

Band 108

Dr.-Ing. Michael Scharer

Quality Gate-Ansatz mit integriertem Risikomanagement

Band 109

Dr.-Ing. Ulrich Suchy

Entwicklung und Untersuchung eines neuartigen Mischkopfes für das Wasser Abrasivstrahlschneiden

Band 110

Dr.-Ing. Sellal Mussa

Aktive Korrektur von Verlagerungsfehlern in Werkzeugmaschinen

Band 111

Dr.-Ing. Andreas Hühsam

Modellbildung und experimentelle Untersuchung des Wälzschälprozesses

Band 112

Dr.-Ing. Axel Plutowsky

Charakterisierung eines optischen Messsystems und den Bedingungen des Arbeitsraums einer Werkzeugmaschine

Band 113

Dr.-Ing. Robert Landwehr

Konsequent dezentralisierte Steuerung mit Industrial Ethernet und offenen Applikationsprotokollen

Band 114

Dr.-Ing. Christoph Dill

Turbulenzreaktionsprozesse

Band 115

Dr.-Ing. Michael Baumeister

Fabrikplanung im turbulenten Umfeld

Band 116

Dr.-Ing. Christoph Gönzheimer

Konzept zur Verbesserung der Elektromagnetischen Verträglichkeit (EMV) in Produktionssystemen durch intelligente Sensor/Aktor-Anbindung

Band 117

Dr.-Ing. Lutz Demuß

Ein Reifemodell für die Bewertung und Entwicklung von Dienstleistungsorganisationen: Das Service Management Maturity Modell (SMMM)

Band 118

Dr.-Ing. Jörg Söhner

Beitrag zur Simulation zerspanungstechnologischer Vorgänge mit Hilfe der Finite-Element-Methode

Band 119

Dr.-Ing. Judith Elsner

Informationsmanagement für mehrstufige Mikro-Fertigungsprozesse

Band 120

Dr.-Ing. Lijing Xie

Estimation Of Two-dimension Tool Wear Based On Finite Element Method

Band 121

Dr.-Ing. Ansgar Blessing

Geometrischer Entwurf mikromechatronischer Systeme

Band 122

Dr.-Ing. Rainer Ebner

Steigerung der Effizienz mehrachsiger Fräsprozesse durch neue Planungsmethoden mit hoher Benutzerunterstützung

Band 123

Dr.-Ing. Silja Klinkel

Multikriterielle Feinplanung in teilautonomen Produktionsbereichen – Ein Beitrag zur produkt- und prozessorientierten Planung und Steuerung

Band 124

Dr.-Ing. Wolfgang Neithardt

Methodik zur Simulation und Optimierung von Werkzeugmaschinen in der Konzept- und Entwurfsphase auf Basis der Mehrkörpersimulation

Band 125

Dr.-Ing. Andreas Mehr

Hartfeinbearbeitung von Verzahnungen mit kristallinen diamantbeschichteten Werkzeugen beim Fertigungsverfahren Wälzstoßen

Band 126

Dr.-Ing. Martin Gutmann

Entwicklung einer methodischen Vorgehensweise zur Diagnose von hydraulischen Produktionsmaschinen

Band 127

Dr.-Ing. Gisela Lanza

Simulative Anlaufunterstützung auf Basis der Qualitätsfähigkeiten von Produktionsprozessen

Band 128

Dr.-Ing. Ulf Dambacher

Kugelgewindetrieb mit hohem Druckwinkel

Band 129

Dr.-Ing. Carsten Buchholz

Systematische Konzeption und Aufbau einer automatisierten Produktionszelle für pulverspritzgegossene Mikrobauteile

Band 130

Dr.-Ing. Heiner Lang

Trocken-Räumen mit hohen Schnittgeschwindigkeiten

Band 131

Dr.-Ing. Daniel Nesges

Prognose operationeller Verfügbarkeiten von Werkzeugmaschinen unter Berücksichtigung von Serviceleistungen

Im Shaker Verlag erschienene Bände:

Band 132

Dr.-Ing. Andreas Bechle

Beitrag zur prozesssicheren Bearbeitung beim Hochleistungsfertigungsverfahren Wälzschäl

Band 133

Dr.-Ing. Markus Herm

Konfiguration globaler Wertschöpfungsnetzwerke auf Basis von Business Capabilities

Band 134

Dr.-Ing. Hanno Tritschler

**Werkzeug- und Zerspanprozessoptimierung beim Hartfräsen
von Mikrostrukturen in Stahl**

Band 135

Dr.-Ing. Christian Munzinger

**Adaptronische Strebe zur Steifigkeitssteigerung
von Werkzeugmaschinen**

Band 136

Dr.-Ing. Andreas Stepping

**Fabrikplanung im Umfeld von Wertschöpfungsnetzwerken und
ganzheitlichen Produktionssystemen**

Band 137

Dr.-Ing. Martin Dyck

**Beitrag zur Analyse thermische bedingter Werkstückdeformationen
in Trockenbearbeitungsprozessen**

Band 138

Dr.-Ing. Siegfried Schmalzried

**Dreidimensionales optisches Messsystem für eine effizientere
geometrische Maschinenbeurteilung**

Band 139

Dr.-Ing. Marc Wawerla

Risikomanagement von Garantieleistungen

Band 140

Dr.-Ing. Ivesa Buchholz

**Strategien zur Qualitätssicherung mikromechanischer Bauteile
mittels multisensorieller Koordinatenmesstechnik**

Band 141

Dr.-Ing. Jan Kotschenreuther

**Empirische Erweiterung von Modellen der Makrozerspannung
auf den Bereich der Mikrobearbeitung**

Band 142

Dr.-Ing. Andreas Knödel

Adaptronische hydrostatische Drucktascheneinheit

Band 143

Dr.-Ing. Gregor Stengel

Fliegendes Abtrennen räumlich gekrümmter Strangpressprofile mittels Industrierobotern

Band 144

Dr.-Ing. Udo Weismann

Lebenszyklusorientiertes interorganisationelles Anlagencontrolling

Band 145

Dr.-Ing. Rüdiger Pabst

Mathematische Modellierung der Wärmestromdichte zur Simulation des thermischen Bauteilverhaltens bei der Trockenbearbeitung

Band 146

Dr.-Ing. Jan Wieser

Intelligente Instandhaltung zur Verfügbarkeitssteigerung von Werkzeugmaschinen

Band 147

Dr.-Ing. Sebastian Haupt

Effiziente und kostenoptimale Herstellung von Mikrostrukturen durch eine Verfahrenskombination von Bahnerosion und Laserablation

Band 148

Dr.-Ing. Matthias Schlipf

Statistische Prozessregelung von Fertigungs- und Messprozess zur Erreichung einer variabilitätsarmen Produktion mikromechanischer Bauteile

Band 149

Dr.-Ing. Jan Philipp Schmidt-Ewig

Methodische Erarbeitung und Umsetzung eines neuartigen Maschinenkonzeptes zur produktflexiblen Bearbeitung räumlich gekrümmter Strangpressprofile

Band 150

Dr.-Ing. Thomas Ender

Prognose von Personalbedarfen im Produktionsanlauf unter Berücksichtigung dynamischer Planungsgrößen

Band 151

Dr.-Ing. Kathrin Peter

**Bewertung und Optimierung der Effektivität von Lean Methoden
in der Kleinserienproduktion**

Band 152

Dr.-Ing. Matthias Schopp

Sensorbasierte Zustandsdiagnose und -prognose von Kugelgewindetrieben

Band 153

Dr.-Ing. Martin Kipfmüller

Aufwandsoptimierte Simulation von Werkzeugmaschinen

Band 154

Dr.-Ing. Carsten Schmidt

**Development of a database to consider multi wear mechanisms
within chip forming simulation**

Band 155

Dr.-Ing. Stephan Niggeschmidt

**Ausfallgerechte Ersatzteilbereitstellung im Maschinen- und Anlagenbau
mittels lastabhängiger Lebensdauerprognose**

Band 156

Dr.-Ing. Jochen Conrad Peters

**Bewertung des Einflusses von Formabweichungen in der
Mikro-Koordinatenmesstechnik**

Band 157

Dr.-Ing. Jörg Ude

**Entscheidungsunterstützung für die Konfiguration
globaler Wertschöpfungsnetzwerke**

Band 158

Dr.-Ing. Stefan Weiler

Strategien zur wirtschaftlichen Gestaltung der globalen Beschaffung

Band 159

Dr.-Ing. Jan Rühl

Monetäre Flexibilitäts- und Risikobewertung

Band 160

Dr.-Ing. Daniel Ruch

Positions- und Konturerfassung räumlich gekrümmter Profile auf Basis bauteilimmanenter Markierungen

Band 161

Dr.-Ing. Manuel Tröndle

Flexible Zuführung von Mikrobauteilen mit piezoelektrischen Schwingförderern

Band 162

Dr.-Ing. Benjamin Viering

Mikroverzahnungsnormal

Band 163

Dr.-Ing. Chris Becke

Prozesskrafttrichtungsangepasste Frässtrategien zur schädigungsarmen Bohrungsbearbeitung an faserverstärkten Kunststoffen

Band 164

Dr.-Ing. Patrick Werner

Dynamische Optimierung und Unsicherheitsbewertung der lastabhängigen präventiven Instandhaltung von Maschinenkomponenten

Band 165

Dr.-Ing. Martin Weis

Kompensation systematischer Fehler bei Werkzeugmaschinen durch self-sensing Aktoren

Band 166

Dr.-Ing. Markus Schneider

Kompensation von Konturabweichungen bei gerundeten Strangpressprofilen durch robotergestützte Führungswerkzeuge

Band 167

Dr.-Ing. Ester M. R. Ruprecht

Prozesskette zur Herstellung schichtbasierter Systeme mit integrierten Kavitäten

Band 168

Dr.-Ing. Alexander Broos

Simulationsgestützte Ermittlung der Komponentenbelastung für die Lebensdauerprognose an Werkzeugmaschinen

Band 169

Dr.-Ing. Frederik Zanger

Segmentspanbildung, Werkzeugverschleiß, Randschichtzustand und Bauteileigenschaften: Numerische Analysen zur Optimierung des Zerspanungsprozesses am Beispiel von Ti-6Al-4V

Band 170

Dr.-Ing. Benjamin Behmann

Servicefähigkeit

Band 171

Dr.-Ing. Annabel Gabriele Jondral

Simulationsgestützte Optimierung und Wirtschaftlichkeitsbewertung des Lean-Methodeneinsatzes

Band 172

Dr.-Ing. Christoph Ruhs

Automatisierte Prozessabfolge zur qualitätssicheren Herstellung von Kavitäten mittels Mikrobahnerosion

Band 173

Dr.-Ing. Steven Peters

Markoffsche Entscheidungsprozesse zur Kapazitäts- und Investitionsplanung von Produktionssystemen

Band 174

Dr.-Ing. Christoph Kühlewein

Untersuchung und Optimierung des Wälzschälverfahrens mit Hilfe von 3D-FEM-Simulation – 3D-FEM Kinematik- und Spanbildungssimulation

Band 175

Dr.-Ing. Adam-Mwanga Dieckmann

Auslegung und Fertigungsprozessgestaltung sintergefügter Verbindungen für μ MIM-Bauteile

Band 176

Dr.-Ing. Heiko Hennrich

Aufbau eines kombinierten belastungs- und zustandsorientierten Diagnose- und Prognosesystems für Kugelgewindetriebe

Band 177

Dr.-Ing. Stefan Herder

Piezoelektrischer Self-Sensing-Aktor zur Vorspannungsregelung in adaptiven Kugelgewindetriebe

Band 178

Dr.-Ing. Alexander Ochs

Ultraschall-Strömungsgreifer für die Handhabung textiler Halbzeuge bei der automatisierten Fertigung von RTM-Bauteilen

Band 179

Dr.-Ing. Jürgen Michna

Numerische und experimentelle Untersuchung zerspanungsbedingter Gefügeumwandlungen und Modellierung des thermo-mechanischen Lastkollektivs beim Bohren von 42CrMo4

Band 180

Dr.-Ing. Jörg Elser

Vorrichtungsfreie räumliche Anordnung von Fügepartnern auf Basis von Bauteilmarkierungen

Band 181

Dr.-Ing. Katharina Klimscha

Einfluss des Fügspalts auf die erreichbare Verbindungsqualität beim Sinterfügen

Band 182

Dr.-Ing. Patricia Weber

Steigerung der Prozesswiederholbarkeit mittels Analyse akustischer Emissionen bei der Mikrolaserablation mit UV-Pikosekundenlasern

Band 183

Dr.-Ing. Jochen Schädel

Automatisiertes Fügen von Tragprofilen mittels Faserwickeln

Band 184

Dr.-Ing. Martin Krauß

Aufwandsoptimierte Simulation von Produktionsanlagen durch Vergrößerung der Geltungsbereiche von Teilmodellen

Band 185

Dr.-Ing. Raphael Moser

Strategische Planung globaler Produktionsnetzwerke

Bestimmung von Wandlungsbedarf und Wandlungszeitpunkt mittels multikriterieller Optimierung

Band 186

Dr.-Ing. Martin Otter

Methode zur Kompensation fertigungsbedingter Gestaltabweichungen für die Montage von Aluminium Space-Frame-Strukturen

Band 187

Dr.-Ing. Urs Leberle

Produktive und flexible Gleitförderung kleiner Bauteile auf phasenflexiblen Schwingförderern mit piezoelektrischen 2D-Antriebselementen

Band 188

Dr.-Ing. Johannes Book

Modellierung und Bewertung von Qualitätsmanagementstrategien in globalen Wertschöpfungsnetzwerken

Band 189

Dr.-Ing. Florian Ambrosy

Optimierung von Zerspanungsprozessen zur prozesssicheren Fertigung nanokristalliner Randschichten am Beispiel von 42CrMo4

Band 190

Dr.-Ing. Adrian Kölmel

Integrierte Messtechnik für Prozessketten unreifer Technologien am Beispiel der Batterieproduktion für Elektrofahrzeuge

Band 191

Dr.-Ing. Henning Wagner

Featurebasierte Technologieplanung zum Preforming von textilen Halbzeugen

Band 192

Dr.-Ing. Johannes Gebhardt

**Strukturoptimierung von in FVK eingebetteten metallischen
Lasteinleitungselementen**

Band 193

Dr.-Ing. Jörg Bauer

**Hochintegriertes hydraulisches Vorschubsystem für die Bearbeitung kleiner
Werkstücke mit hohen Fertigungsanforderungen**

Band 194

Dr.-Ing. Nicole Stricker

Robustheit verketteter Produktionssysteme

Robustheitsevaluation und Selektion des Kennzahlensystems der Robustheit

Band 195

Dr.-Ing. Anna Sauer

**Konfiguration von Montagelinien unreifer Produkttechnologien am Beispiel der
Batteriemontage für Elektrofahrzeuge**

Band 196

Dr.-Ing. Florian Sell-Le Blanc

Prozessmodell für das Linearwickeln unrunder Zahnspulen

Ein Beitrag zur orthozyklischen Spulenwickeltechnik

Band 197

Dr.-Ing. Frederic Förster

**Geregeltes Handhabungssystem zum zuverlässigen und energieeffizienten
Handling textiler Kohlenstofffaserzuschnitte**

Band 198

Dr.-Ing. Nikolay Boev

**Numerische Beschreibung von Wechselwirkungen zwischen Zerspanprozess und
Maschine am Beispiel Räumen**

Band 199

Dr.-Ing. Sebastian Greinacher

**Simulationsgestützte Mehrzieloptimierung schlanker und ressourceneffizienter
Produktionssysteme**

Band 200

Dr.-Ing. Benjamin Häfner

Lebensdauerprognose in Abhängigkeit der Fertigungsabweichungen bei Mikroverzahnungen

Band 201

Dr.-Ing. Stefan Klotz

Dynamische Parameteranpassung bei der Bohrungsherstellung in faserverstärkten Kunststoffen unter zusätzlicher Berücksichtigung der Einspannsituation

Band 202

Dr.-Ing. Johannes Stoll

Bewertung konkurrierender Fertigungsfolgen mittels Kostensimulation und stochastischer Mehrzieloptimierung

Anwendung am Beispiel der Blechpaketfertigung für automobiler Elektromotoren

Band 203

Dr.-Ing. Simon-Frederik Koch

Fügen von Metall-Faserverbund-Hybridwellen im Schleuderverfahren ein Beitrag zur fertigungsgerechten intrinsischen Hybridisierung

Band 204

Dr.-Ing. Julius Ficht

Numerische Untersuchung der Eigenspannungsentwicklung für sequenzielle Zerspanungsprozesse

Band 205

Dr.-Ing. Manuel Baumeister

Automatisierte Fertigung von Einzelblattstapeln in der Lithium-Ionen-Zellproduktion

Band 206

Dr.-Ing. Daniel Bertsch

Optimierung der Werkzeug- und Prozessauslegung für das Wälzschälen von Innenverzahnungen

Band 207

Dr.-Ing. Kyle James Kippenbrock

Deconvolution of Industrial Measurement and Manufacturing Processes for Improved Process Capability Assessments

Band 208

Dr.-Ing. Farboud Bejnoud

Experimentelle Prozesskettenbetrachtung für Räumbauteile am Beispiel einer einsatzgehärteten PKW-Schiebemuffe

Band 209

Dr.-Ing. Steffen Dosch

Herstellungsübergreifende Informationsübertragung zur effizienten Produktion von Werkzeugmaschinen am Beispiel von Kugelgewindetrieben

Band 210

Dr.-Ing. Emanuel Moser

Migrationsplanung globaler Produktionsnetzwerke

Bestimmung robuster Migrationspfade und risiko-effizienter Wandlungsbefähiger

Band 211

Dr.-Ing. Jan Hochdörffer

Integrierte Produktallokationsstrategie und Konfigurationssequenz in globalen Produktionsnetzwerken

Band 212

Dr.-Ing. Tobias Arndt

Bewertung und Steigerung der Prozessqualität in globalen Produktionsnetzwerken

Band 213

Dr.-Ing. Manuel Peter

Unwuchtminimale Montage von Permanentmagnetrotoren durch modellbasierte Online-Optimierung

Band 214

Dr.-Ing. Robin Kopf

Kostenorientierte Planung von Fertigungsfolgen additiver Technologien

Band 215

Dr.-Ing. Harald Meier

**Einfluss des Räumens auf den Bauteilzustand in der Prozesskette
Weichbearbeitung – Wärmebehandlung – Hartbearbeitung**

Band 216

Dr.-Ing. Daniel Brabandt

**Qualitätssicherung von textilen Kohlenstofffaser-Preforms mittels
optischer Messtechnik**

Band 217

Dr.-Ing. Alexandra Schabunow

**Einstellung von Aufnahmeparametern mittels projektionsbasierter Qualitäts-
kenngrößen in der industriellen Röntgen-Computertomographie**

Band 218

Dr.-Ing. Jens Bürgin

Robuste Auftragsplanung in Produktionsnetzwerken

Mittelfristige Planung der variantenreichen Serienproduktion unter Unsicherheit
der Kundenauftragskonfigurationen

Band 219

Dr.-Ing. Michael Gerstenmeyer

**Entwicklung und Analyse eines mechanischen Oberflächenbehandlungs-
verfahrens unter Verwendung des Zerspanungswerkzeuges**

Band 220

Dr.-Ing. Jacques Burtcher

**Erhöhung der Bearbeitungsstabilität von Werkzeugmaschinen durch
semi-passive masseneinstellbare Dämpfungssysteme**

Band 221

Dr.-Ing. Dietrich Berger

**Qualitätssicherung von textilen Kohlenstofffaser-Preforms mittels prozess-
integrierter Wirbelstromsensor-Arrays**

Band 222

Dr.-Ing. Fabian Johannes Ballier

Systematic gripper arrangement for a handling device in lightweight production processes

Band 223

Dr.-Ing. Marielouise Schäferling, geb. Zaiß

Development of a Data Fusion-Based Multi-Sensor System for Hybrid Sheet Molding Compound

Band 224

Dr.-Ing. Quirin Spiller

Additive Herstellung von Metallbauteilen mit dem ARBURG Kunststoff-Freiformen

Band 225

Dr.-Ing. Andreas Spohrer

Steigerung der Ressourceneffizienz und Verfügbarkeit von Kugelgewindetrieben durch adaptive Schmierung

Band 226

Dr.-Ing. Johannes Fisel

Veränderungsfähigkeit getakteter Fließmontagesysteme

Planung der Fließbandabstimmung am Beispiel der Automobilmontage

Band 227

Dr.-Ing. Patrick Bollig

Numerische Entwicklung von Strategien zur Kompensation thermisch bedingter Verzüge beim Bohren von 42CrMo4

Band 228

Dr.-Ing. Ramona Pfeiffer, geb. Singer

Untersuchung der prozessbestimmenden Größen für die anforderungsgerechte Gestaltung von Pouchzellen-Verpackungen

Band 229

Dr.-Ing. Florian Baumann

Additive Fertigung von endlosfaserverstärkten Kunststoffen mit dem ARBURG Kunststoff-Freiform Verfahren

Band 230

Dr.-Ing. Tom Stähr

Methodik zur Planung und Konfigurationsauswahl skalierbarer Montagesysteme – Ein Beitrag zur skalierbaren Automatisierung

Band 231

Dr.-Ing. Jan Schwennen

Einbringung und Gestaltung von Lasteinleitungsstrukturen für im RTM-Verfahren hergestellte FVK-Sandwichbauteile

Band 232

Dr.-Ing. Sven Coutandin

Prozessstrategien für das automatisierte Preforming von bebinderten textilen Halbzeugen mit einem segmentierten Werkzeugsystem

Band 233

Dr.-Ing. Christoph Liebrecht

Entscheidungsunterstützung für den Industrie 4.0-Methodeneinsatz
Strukturierung, Bewertung und Ableitung von Implementierungsreihenfolgen

Band 234

Dr.-Ing. Stefan Treber

Transparenzsteigerung in Produktionsnetzwerken
Verbesserung des Störungsmanagements durch verstärkten Informationsaustausch

Band 235

Dr.-Ing. Marius Dackweiler

Modellierung des Fügewickelprozesses zur Herstellung von leichten Fachwerkstrukturen

Band 236

Dr.-Ing. Fabio Echsler Minguillon

Prädiktiv-reaktives Scheduling zur Steigerung der Robustheit in der Matrix-Produktion

Band 237

Dr.-Ing. Sebastian Haag

Entwicklung eines Verfahrensablaufes zur Herstellung von Batteriezellstapeln mit großformatigem, rechteckigem Stapelformat und kontinuierlichen Materialbahnen

Band 238

Dr.-Ing. Raphael Wagner

Strategien zur funktionsorientierten Qualitätsregelung in der Serienproduktion

Band 239

Dr.-Ing. Christopher Ehrmann

Ausfallfrüherkennung von Ritzel-Zahnstangen- Trieben mittels Acoustic Emission

Band 240

Dr.-Ing. Janna Hofmann

Prozessmodellierung des Fünf-Achs-Nadelwickelns zur Implementierung einer trajektoriebasierten Drahtzugkraftregelung

Band 241

Dr.-Ing. Andreas Kuhnle

Adaptive Order Dispatching based on Reinforcement Learning
Application in a Complex Job Shop in the Semiconductor Industry

Band 242

Dr.-Ing. Andreas Greiber

Fertigung optimierter technischer Oberflächen durch eine Verfahrenskombination aus Fliehkraft-Tauchgleitschleifen und Laserablation
Prozesseinflüsse und Prozessauslegung

Band 243

Dr.-Ing. Jan Niclas Eschner

Entwicklung einer akustischen Prozessüberwachung zur Porenbestimmung im Laserstrahlschmelzen

Band 244

Dr.-Ing. Sven Roth

Schädigungsfreie Anbindung von hybriden FVK/Metall-Bauteilen an metallische Tragstrukturen durch Widerstandspunktschweißen

Band 245

Dr.-Ing. Sina Kathrin Peukert

Robustheitssteigerung in Produktionsnetzwerken mithilfe eines integrierten Störungsmanagements

Band 246

Dr.-Ing. Alexander Jacob

Hochiterative Technologieplanung

Rekursive Optimierung produkt- und fertigungsbezogener Freiheitsgrade am Beispiel der hybrid-additiven Fertigung

Band 247

Dr.-Ing. Patrick Moll

Ressourceneffiziente Herstellung von Langfaser-Preforms im Faserblasverfahren

Band 248

Dr.-Ing. Eric Thore Segebadé

Erhöhung der Verschleißbeständigkeit von Bauteilen aus Ti-6Al-4V mittels simulationsgestützter Zerspanung und mechanischer Mikrotextrurierung

Band 249

Dr.-Ing. Shun Yang

Regionalized implementation strategy of smart automation within assembly systems in China

Band 250

Dr.-Ing. Constantin Carl Hofmann

Vorausschauende und reaktive Mehrzieloptimierung für die Produktionssteuerung einer Matrixproduktion

Band 251

Dr.-Ing. Paul Ruhland

Prozesskette zur Herstellung von hybriden Faser-Metall-Preforms
Modellbildung und Optimierung des Binderauftrags und der Drapierung für stabförmige Bauteile

Band 252

Dr.-Ing. Leonard Schild

Erzeugung und Verwendung von Anwendungswissen in der industriellen Computertomographie

Band 253

Dr.-Ing. Benedikt Klee

Analyse von Phaseninformationen in Videodaten zur Identifikation von Schwingungen in Werkzeugmaschinen

Band 254

Dr.-Ing. Bruno Vargas

Wälzschalen mit kleinen Achskreuzwinkeln

Prozessgrenzen und Umsetzbarkeit

Band 255

Dr.-Ing. Lucas Bretz

Function-oriented in-line quality assurance of hybrid sheet molding compound

Band 256

Dr.-Ing. Bastian Rothaupt

Dämpfung von Bauteilschwingungen durch einstellbare Werkstückdirektspannung mit Hydrodehnspanntechnik

Band 257

Dr.-Ing. Daniel Kupzik

Robotic Swing Folding of three-dimensional UD-tape-based Reinforcement Structures

Band 258

Dr.-Ing. Bastian Verhaelen

(De-)Zentralisierung von Entscheidungen in globalen Produktionsnetzwerken

Strategie- und komplexitätsorientierte Gestaltung der Entscheidungsautonomie

Band 259

Dr.-Ing. Hannes Wilhelm Weinmann

Integration des Vereinzelungs- und Stapelbildungsprozesses in ein flexibel und kontinuierlich arbeitendes Anlagenmodul für die Li-Ionen Batteriezellfertigung

Band 260

Dr.-Ing. Florian Stamer

Dynamische Lieferzeit-Preisgestaltung in variantenreicher Produktion

Ein adaptiver Ansatz mithilfe von Reinforcement Learning

Band 261

Dr.-Ing. Patrick Neuenfeldt

Modellbildung des Tauchgleitschleifens zur Abtrag- und Topografievorhersage an komplexen Geometrien

Band 262

Dr.-Ing. Boris Matuschka

Energieeffizienz in Prozessketten: Analyse und Optimierung von Energieflüssen bei der Herstellung eines PKW-Getriebebauteils aus 16MnCr5

Band 263

Dr.-Ing. Tobias Schlagenhauf

Bildbasierte Quantifizierung und Prognose des Verschleißes an Kugelgewindetriebspindeln

Ein Beitrag zur Zustandsüberwachung von Kugelgewindetrieben mittels Methoden des maschinellen Lernens

Band 264

Dr.-Ing. Benedict Stampfer

Entwicklung eines multimodalen Prozessmodells zur Oberflächenkonditionierung beim Außenlängsdrehen von 42CrMo4

Band 265

Dr.-Ing. Carmen Maria Krahe

KI-gestützte produktionsgerechte Produktentwicklung

Automatisierte Wissensextraktion aus vorhandenen Produktgenerationen