A Multitask Latent Space Learning Framework for Modeling and Inversion of Structure–Property Relationships

Ein Multitask Learning Framework für die Erzeugung von latenten Repräsentationen zur Modellierung und Invertierung von Struktur–Eigenschafts–Beziehungen

Von der Fakultät für Georessourcen und Materialtechnik der

Rheinisch-Westfälischen Technischen Hochschule Aachen

zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

genehmigte Dissertation

vorgelegt von

Tarek Iraki, M.Sc.

Berichter: Univ.-Prof. Dr. Stefan Sandfeld
Univ.-Prof. Dr.-Ing. Sebastian Münstermann

Tag der mündlichen Prüfung: 09.03.2026

Diese Dissertation ist auf den Internetseiten der Universitätsbibliothek online verfügbar.

For Lena

Abstract

The process–structure–property (PSP) linkage forms the foundation of modern materials science. It describes how manufacturing processes shape microstructural features, which in turn govern a material's properties. Characteristics such as strength, ductility, hardness, and anisotropy define the functional suitability of a component, and tailoring these properties to the intended application is essential. This leads to the concept of inverse design, which seeks to map target material properties onto microstructural evolutions and ultimately onto optimal process parameters. Achieving this goal is highly challenging, as material properties depend on complex and often nonlinear relationships with the underlying microstructure. Effective process optimization therefore requires controlling microstructural evolution through process parameter trajectories, which typically operates in a high-dimensional microstructure space. For instance, if microstructures are represented by orientation distribution functions (ODFs), the resulting complexity can be computationally intensive.

Despite decades of experimental and modeling studies, fully exploiting structure–property relationships for predictive and inverse design remains a significant challenge. Machine learning offers powerful and continually advancing capabilities to address this issue by learning high-dimensional, nonlinear structure–property linkages directly from data and extracting near-optimal process paths through systematic exploration and optimization. A prerequisite for such optimization is the specification of goal microstructures that fulfill the desired properties. This identification of microstructures corresponding to desired properties is commonly referred to as the inverse design of structure–property relationships.

This work advances data-driven methodologies for modeling and optimization of structure–property relationships in materials science, with the dual objectives of enabling inverse design and reducing the dimensionality of the microstructure space to support efficient optimization. The set of microstructures resulting from inverse design should not only exhibit the desired properties but also satisfy two additional requirements: they must be physically producible through the process, and they should encompass sufficient diversity to provide a broad range of design options. Integrating these requirements into the inverse design framework constitutes a key contribution of this work, ensuring that all microstructures in the solution set are feasible and that process optimization can identify near-optimal paths.

The main contribution of this work is the development of a multitask latent space learning approach that provides a low-dimensional representation for microstructures. In this framework, a shared encoder maps microstructures onto the latent space. The required inverse mapping from desired properties to a diverse set of feasible microstructures is performed by an optimizer with a specifically designed objective function, also developed in this work. The diversity requirement for the microstructure set is quantified through the mutual distances among its members. Since microstructures are represented by ODFs, a dedicated distance measure between ODFs is introduced.

The resulting efficiency gain in process optimization represents the main advancement over the state of the art, demonstrated in this work through applications of the developed framework to dedicated processes. By interpreting microstructures more generally as process state variables, the multitask framework—with its low-dimensional latent space, reconstruction capability, and property mappings—can be transferred to other processes. This generalizability is illustrated using the example of resistance spot welding.

Zusammenfassung

Die Verbindung zwischen Prozess, Struktur und Eigenschaft (PSP) bildet die Grundlage der modernen Materialwissenschaft. Sie beschreibt, wie Fertigungsprozesse mikrostrukturelle Merkmale beeinflussen, die wiederum die Eigenschaften eines Materials bestimmen. Eigenschaften wie Festigkeit, Duktilität, Härte und Anisotropie bestimmen die funktionale Eignung eines Bauteils, und es ist unerlässlich, diese Eigenschaften auf die beabsichtigte Anwendung abzustimmen. Dies führt zum Konzept des inversen Designs, das darauf abzielt, die gewünschten Materialeigenschaften auf mikrostrukturelle Entwicklungen und letztlich auf optimale Prozessparameter abzubilden. Das Erreichen dieses Ziels ist eine große Herausforderung, da die Materialeigenschaften von komplexen und oft nichtlinearen Beziehungen zur zugrunde liegenden Mikrostruktur abhängen. Eine effektive Prozessoptimierung erfordert daher die Steuerung der mikrostrukturellen Entwicklung durch Prozessparameterverläufe, die typischerweise in einem hochdimensionalen Mikrostrukturraum ablaufen. Wenn beispielsweise Mikrostrukturen durch Orientierungsverteilungsfunktionen (ODFs) dargestellt werden, kann die daraus resultierende Komplexität rechenintensiv sein.

Trotz jahrzehntelanger experimenteller und modellbasierter Studien bleibt die vollständige Nutzung von Struktur-Eigenschafts-Beziehungen für prädiktives und inverses Design herausfordernd. Maschinelles Lernen eröffnet hier neue Möglichkeiten, da es hochdimensionale, nichtlineare Zusammenhänge direkt aus Daten erlernen und nahezu optimale Prozesspfade durch systematische Exploration und Optimierung ableiten kann. Voraussetzung ist die Definition von Zielmikrostrukturen, die die gewünschten Eigenschaften erfüllen. Diese Identifizierung wird allgemein als inverses Design von Struktur-Eigenschafts-Beziehungen bezeichnet.

Diese Arbeit entwickelt datengesteuerte Methoden zur Modellierung und Optimierung solcher Beziehungen. Ziel ist es, inverses Design zu ermöglichen und gleichzeitig die Dimensionalität des Mikrostrukturraums zu reduzieren, um eine effiziente Optimierung zu unterstützen. Die ermittelten Mikrostrukturen müssen nicht nur die geforderten Eigenschaften aufweisen, sondern auch physikalisch herstellbar und hinreichend vielfältig sein, um eine breite Palette an Designoptionen zu bieten. Die Integration dieser Anforderungen in das inverse Design Framework stellt einen wesentlichen Beitrag dar, da sie sicherstellt, dass alle Lösungen realisierbar sind und die Prozessoptimierung robuste Wege identifizieren kann.

Der Hauptbeitrag dieser Arbeit ist die Entwicklung eines Multitask-Latent-Space-Learning-Ansatzes, der eine niedrigdimensionale Darstellung für Mikrostrukturen liefert. Ein gemeinsamer Encoder bildet Mikrostrukturen auf den Latent Space ab. Die inverse Abbildung von Eigenschaften auf vielfältige, realisierbare Mikrostrukturen erfolgt durch einen Optimierer mit einer speziell entwickelten Zielfunktion. Diversität wird dabei über Abstände zwischen den Mikrostrukturen quantifiziert. Da diese durch ODFs dargestellt werden, wird ein spezielles Distanzmaß zwischen ODFs eingeführt.

Der daraus resultierende Effizienzgewinn bei der Prozessoptimierung stellt den wichtigsten Fortschritt gegenüber dem Stand der Technik dar, der in dieser Arbeit durch die Anwendung des entwickelten Ansatzes auf spezielle Prozesse demonstriert wird. Durch die allgemeinere Interpretation von Mikrostrukturen als Prozesszustandsvariablen kann das Multitask-Framework – mit seinem niedrigdimensionalen latenten Raum, seiner Rekonstruktionsfähigkeit und seinen Eigenschaftsabbildungen – auf andere Prozesse übertragen werden. Diese Verallgemeinerbarkeit wird am Beispiel des Widerstandspunktschweißens veranschaulicht.

Acknowledgments

Firstly, I would like to express my deepest gratitude to Prof. Dr. Stefan Sandfeld for supervising this thesis and for giving me the opportunity to be part of his research institute, which has been greatly beneficial. His guidance and support throughout my research were instrumental in the development of my academic career.

I am also deeply grateful to Prof. Dr. Norbert Link for providing me with the invaluable opportunity to be part of his research group and for supporting my academic development as a mentor.

I would also like to express my sincere gratitude to Prof. Dr.-Ing. Sebastian Münstermann for serving as a reviewer of this thesis and to Prof. Dr. Jochen Michael Schneider for serving as the chair of the examination committee.

Furthermore, I would like to express my profound gratitude to all those individuals who have accompanied me over the past few years and provided me with their invaluable support (in alphabetical order): Tarek Allam, Robin Baumann, Anita Bender, Melina Bolle, Bei Chen, Aytekin Demirci, Johannes Dornheim, Nadine Filipovic, Hariprasath Ganesan, Abril Azocar Guzman, Hamed Hemati, Dirk Helm, Hamed Hemati, Katharina Immel, Dennis Janka, Alicia Janz, Bashir Kazimi, Shivangi Kirti, Peter Konijnenberg, Marion Köster, Lucas Lamparter, Astrid Laubenheimer, Alexander Melde, Lukas Morand, Heather More, Pavlo Potapenko, Martin Redlof, Gabriele Strauch, Hui Min Teh, Rohan Nolson Thekkekunnel, Christian Wernet, Johannes Wetzel, Le Yang, Samuel Zeitvogel, Hu Zhao from Intelligent Systems Research Group (ISRG), Institute for Advanced Simulations - Materials Data Science and Informatics (IAS-9), Institute of Energy Materials and Devices - Structure and Function of Materials (IMD-1), RWTH Aachen - Institute of Metal Forming (ibf), RWTH Aachen - Materials Chemistry, and Fraunhofer Institute for Mechanics of Materials (IWM).

The completion of this work would not have been possible without the support of my family. I would like to express my sincere gratitude to my parents, Christina and Ahmad, for their unwavering and unconditional support throughout this journey. I am also deeply grateful to my parents-in-law, Maria-Luise and Wolfgang, and particularly appreciate their help with the preparation and organization of the celebration following my dissertation defense. The support, effort, and generosity of both my parents and my parents-in-law helped turn this milestone into a truly memorable day.

Finally, I would like to express my deepest gratitude to my wife, Lena, for her boundless patience and invaluable support throughout the demanding period of this work. I also sincerely thank her for carefully reviewing numerous figures in this thesis and for providing thoughtful and critical feedback ("And you may find yourself in a beautiful house, with a beautiful wife, and you may ask yourself, 'Well, how did I get here?'"[1]).

[1] Lyrics from *Once in a Lifetime* by Talking Heads (written by David Byrne, Brian Eno, Chris Frantz, Tina Weymouth, and Jerry Harrison), released in 1980 on the album *Remain in Light*.

Prior publications

This work is based in part on research performed by the author during their time at the Intelligent Systems Research Group (ISRG) at Karlsruhe University of Applied Sciences, Karlsruhe, Germany, and at the Institute for Advanced Simulations - Materials Data Science and Informatics (IAS-9) at Forschungszentrum Jülich, Jülich, Germany between May 2019 and April 2023 and May 2023 and August 2025.

Major parts of this work have appeared in previous publications and are reproduced here in restructured, modified, and/or extended form, with this work being based primarily on the author's peer-reviewed journal articles. Chapters 1, 2, and 3 are derived on the introductions and abstracts of the publications listed below. Chapters 4, 5, and 6 reproduce the introduction, methodology, results, and discussion from the publications (Iraki and Link, 2022), (Iraki et al., 2024a), and (Iraki et al., 2024b), respectively. Chapter 7 is primarily derived from (Morand et al., 2024), where the introduction is reproduced, while the methodology, results, and discussion have been extended through the inclusion of an additional use case beyond the original publication. Chapter 8 builds upon the findings of the listed publications and synthesizes them into a comprehensive summary and conclusion, highlighting their collective contributions. Figures and tables reproduced from these publications are referenced accordingly.

For each of the peer-reviewed journal publications listed below, the author T. Iraki contributed in the following areas: writing (original draft, review, and editing), visualization, validation, software, methodology, investigation, and conceptualization:

- **T. Iraki**, N. Link. Generative models for capturing and exploiting the influence of process conditions on process curves. Journal of Intelligent Manufacturing 33, 473–492 (2022). https://doi.org/10.1007/s10845-021-01846-4

  The author T. Iraki developed the multitask learning approach as well as the transformation approach and applied both to the problem examined in this study.

- **T. Iraki**, L. Morand, J. Dornheim, N. Link, D. Helm. A multi-task learning-based optimization approach for finding diverse sets of microstructures with desired properties. Journal of Intelligent Manufacturing 35, 1887–1903 (2024). https://doi.org/10.1007/s10845-023-02139-8

  The author T. Iraki developed the Siamese multitask learning approach as well as the optimization strategy for generating diverse sets of microstructures. Both approaches were applied to the problem examined in this study.

- **T. Iraki**, L. Morand, N. Link, S. Sandfeld, D. Helm. Accurate distances measures and machine learning of the texture-property relation for crystallographic textures represented by one-point statistics. Modelling and Simulation in Materials Science and Engineering, Volume 32, Number 5 (2024). https://doi.org/10.1088/1361-651X/ad4c81

DeepL was used to assist in rephrasing and correcting grammatical errors. This tools was employed solely to enhance the clarity and readability of the text. Some of the figures presented in this thesis were prepared with the assistance of ChatGPT (OpenAI), which was employed as a programming tool to generate and refine code for `matplotlib` and `TikZ`.

The author T. Iraki developed accurate distance measures for crystallographic textures. In addition, he implemented machine learning approaches to model the texture–property relationship, which represent key contributions to this study.

- L. Morand, **T. Iraki**, J. Dornheim, S. Sandfeld, N. Link, D. Helm. Machine learning for structure-guided materials and process design. Materials & Design, Volume 248, (2024), https://doi.org/10.1016/j.matdes.2024.113453

  The author T. Iraki developed the Siamese multitask learning optimization (SMTLO) approach and applied both the SMTLO and the multi-equivalent goal structure-guided processing path optimization (MEG-SGGPO) approaches to the problem analyzed in this study. Furthermore, he extended the MEG-SGGPO approach by using the Sinkhorn distance measure.

Nomenclature

Acronyms

Acronym	Description	Page
CNN	Convolutional Neural Networks	18
cGAN	Conditional Generative Adversarial Networks	136
CVAE	Conditional Variational Autoencoder	134
EMD	Earth Mover's Distance	5
GSH	Generalized Spherical Harmonics	32
GAN	Generative Adversarial Network	40
ICME	Integrated Computational Materials Engineering	2
LF	Latent Features	26
ML	Machine Learning	2
MDS	Multidimensional Scaling	22
MLP	Multilayer Perceptron	16
MAE	Mean Absolute Error	77
MSE	Mean Squared Error	23
MTL	Multitask Learning	3
OCSVM	One-class Support Vector Machine	44
ODF	Orientation Distribution Function	5
PCA	Principal Component Analysis	10
PSP	Process-Structure-Property	1
ReLU	Rectified Linear Unit	17
RL	Reinforcement Learning	11
SSC	Steel Sheet Combinations	51
SMTL	Siamese multitask learning	4
SMTLO	Siamese multitask learning optimization	5
SVM	Support Vector Machine	8
TCN	Temporal Convolutional Network	40
VAE	Variational Autoencoder	3

Symbols for the neural network models for the general multitask learning approach

Symbol	Description	Equation	Page
$\mathcal{D}$	Dataset		28
$\boldsymbol{x} \in \mathbb{R}^D$	Input features with dimension D		28
$\boldsymbol{y}_{\text{cls}_t} \in \mathbb{Z}^K$	Ground truth target labels for the t-th classification task with K classes		28
$\boldsymbol{y}_{\text{est}_t} \in \mathbb{R}^P$	Ground truth target continuous values for the t-th estimation task with P values		28
Θ	Compound parameter vector		29
Θ_{enc}	Parameter vector for the encoder network	(2.26)	28
Θ_{dec}	Parameter vector for the decoder network	(2.27)	28
Θ_{cls_t}	Parameter vector for the t-th classification network	(2.28)	28
Θ_{est_t}	Parameter vector for the t-th estimation network	(2.29)	29
$f_{\text{enc}}(\boldsymbol{x}, \Theta_{\text{enc}})$	Encoder function delivers latent feature $\boldsymbol{z}$	(2.26)	28
$f_{\text{dec}}(\boldsymbol{z}, \Theta_{\text{dec}})$	Decoder function	(2.27)	28
$f_{\text{cls}_t}(\boldsymbol{z}, \Theta_{\text{cls}_t})$	Classification function for the t-th classification	(2.28)	28
$f_{\text{est}_t}(\boldsymbol{z}, \Theta_{\text{est}_t})$	Estimation function for the t-th estimation	(2.29)	29
λ_{dec}	Weight of the decoder loss	(2.31)	29
λ_{cls_t}	Weight of the t-th classification loss	(2.31)	29
λ_{est_t}	Weight of the t-th estimation loss	(2.31)	29
$\boldsymbol{z} \in \mathbb{R}^Z$	Low-dimensional latent feature representation of the input features $\boldsymbol{x}$	(2.26)	28
$\hat{\boldsymbol{x}} \in \mathbb{R}^D$	Reconstruction of $\boldsymbol{x}$	(2.27)	28
$\hat{\boldsymbol{y}}_{\text{cls}_t} \in \mathbb{Z}^K$	Prediction of the t-th classification task	(2.28)	28
$\hat{\boldsymbol{y}}_{\text{est}_t} \in \mathbb{R}^P$	Prediction of the t-th estimation task	(2.29)	29
$\mathcal{J}_{\text{dec}}(\boldsymbol{x}, \hat{\boldsymbol{x}})$	Loss for reconstruction	(2.32)	29
$\mathcal{J}_{\text{cls}_t}(\boldsymbol{y}_{\text{cls}_t}, \hat{\boldsymbol{y}}_{\text{cls}_t})$	Loss for the t-th classification task	(2.33)	29
$\mathcal{J}_{\text{est}_t}(\boldsymbol{y}_{\text{est}_t}, \hat{\boldsymbol{y}}_{\text{est}_t})$	Loss for the t-th estimation task	(2.34)	29

Symbols for the crystallographic texture

Symbol	Description	Equation	Page
g	Orientation (point in $SO(3)$)	(2.36)	31
$f(g)$	Orientation distribution function	(2.36)	31
$f_{\mathrm{o}}(g)$	Approximation of the ODF as orientation histogram	(2.41)	33
$f_{\mathrm{p}}(g)$	Approximation of the ODF as polefigure		33
$f_{\mathrm{g}}(g)$	Approximation of the ODF on the basis of generalized spherical harmonics	(2.37)	32
$\boldsymbol{F}$	Deformation gradient	(2.44)	34
$\boldsymbol{T}^{(\mathrm{i})}$	Cauchy stress of ith crystal	(2.43)	34
$\boldsymbol{L}_{\mathrm{p}}$	Velocity gradient	(2.46)	34
E_{00}, E_{45}, E_{90}	Young's moduli in 0, 45, and 90 degree to rolling direction		75
R_{00}, R_{45}, R_{90}	R-values in 0, 45, and 90 degree to rolling direction		75
E_{11}, E_{22}, E_{33}	Young's moduli in three orthogonal directions		113
$\widetilde{R}_{23}, \widetilde{R}_{12}, \widetilde{R}_{13}$	Describes the plastic anisotropy at the material point in three orthogonal directions		113
$\mathscr{D}_{\chi^2}(.,.)$	Chi-Squared distance between two orientation histograms $f_{\mathrm{o}}(g), f_{\mathrm{o}}(g)$	(2.42)	33
$\mathscr{D}_{\mathrm{sh}}(.,.)$	Sinkhorn distance between two orientation histograms $f_{\mathrm{o}}(g), f_{\mathrm{o}}(g)$	(6.12)	88
$\mathscr{D}_{\mathrm{p}}(.,.)$	Distance between two pole figures $f_{\mathrm{p}}^{(1)}(g), f_{\mathrm{p}}^{(2)}(g)$	(6.13)	89
$\mathscr{D}_{\mathrm{g}}(.,.)$	Distance between two orientation densities expressed as GSH $f_{\mathrm{g}}^{(1)}(g), f_{\mathrm{g}}^{(2)}(g)$	(6.6)	87

Symbols for resistance spot welding

Symbol	Description	Equation	Page
$\boldsymbol{s}$	process state		50
$\boldsymbol{s}_0$	initial process state		50
$\boldsymbol{u}$	external process driving forces		50
$\boldsymbol{b}$	physical process boundary conditions		50
$\boldsymbol{i}$	properties of the process input		50
$\boldsymbol{M}$	machine characteristics		50
$\boldsymbol{p}$	control parameters		50
$\boldsymbol{c}$	process conditions		50
$\boldsymbol{c}', \boldsymbol{c}''$	process condition instances		50
$\boldsymbol{q}$	quality measures		50
$\boldsymbol{m}(\mathrm{t}_i)$	measured values of observables at time t_i		50

Contents

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Problem statement . . . 3
- 1.3 Contribution . . . 4
- 1.4 Thesis outline . . . 5

2 Background **7**
- 2.1 Fundamentals of machine learning . . . 7
  - 2.1.1 Overview of machine learning . . . 7
  - 2.1.2 Neural networks . . . 12
  - 2.1.3 Dimensionality reduction . . . 21
  - 2.1.4 Gradient-free optimization . . . 24
- 2.2 Approach for the learning of an universal low dimensional latent feature space . . . 26
- 2.3 Introduction to the most important materials aspects relevant to this work . . . 30
  - 2.3.1 The crystallographic texture . . . 31
  - 2.3.2 Taylor-type material model . . . 33
  - 2.3.3 Resistance spot welding . . . 36

3 State of the art **39**
- 3.1 Time-series analysis of process data . . . 39
- 3.2 Materials and process design . . . 41
- 3.3 Measuring texture distances . . . 45

4 The impact of process conditions on material behavior **49**
- 4.1 Methods . . . 50
  - 4.1.1 Semantic process curve transformation and hyper-modeling . . . 50
  - 4.1.2 Obtaining the latent feature space for the spot welding application . . . 51
  - 4.1.3 Derivation of class-conditional density functions and generator models . . . 54
  - 4.1.4 Topologies of the neural networks . . . 55
- 4.2 Results . . . 56
  - 4.2.1 Data set for the spot welding application . . . 57
  - 4.2.2 Validation of the neural network model . . . 57
  - 4.2.3 Evaluation results with generalized features . . . 59
  - 4.2.4 Latent feature space representation and conditional generative and transformation models . . . 60
- 4.3 Discussion . . . 65

5 Multitask learning in materials design **67**
5.1 Methods . . . 67
5.1.1 Materials design via siamese multitask learning and optimization . . . 67
5.1.2 Materials science fundamentals . . . 74
5.2 Results . . . 75
5.2.1 Texture-property data set . . . 75
5.2.2 Validation of the siamese multitask learning model . . . 77
5.2.3 Rolling texture identification . . . 78
5.3 Discussion . . . 82

6 Measuring texture distances and learning texture-property relations **85**
6.1 Methods . . . 86
6.1.1 Distance measures for texture representations . . . 86
6.1.2 Texture representations for learning texture-property relations . . . 89
6.1.3 Texture data sets for evaluation . . . 91
6.2 Results . . . 93
6.2.1 Comparing measures of texture distances . . . 93
6.2.2 Learning texture-property relations . . . 101
6.3 Discussion . . . 102

7 Holistic optimization of the process-structure-property linkage **105**
7.1 Methods . . . 106
7.1.1 Structure-guided materials and process design . . . 106
7.1.2 Application case . . . 111
7.2 Results . . . 113
7.2.1 Materials and process design task . . . 113
7.2.2 Validation of the siamese multi task learning model . . . 113
7.2.3 Materials design using siamese multi task learning optimization . . . 115
7.2.4 Process design using multi-equivalent goal structure-guided processing path optimization . . . 119
7.3 Discussion . . . 121

8 Summary and Conclusion **123**

9 Outlook **127**

A Appendix **131**
A.1 Supporting information for Section 2.1 . . . 131
A.1.1 Neural networks . . . 131
A.1.2 Dimensionality reduction . . . 131
A.1.3 Generative approaches . . . 133
A.1.4 Anomaly detection . . . 136
A.1.5 Gradient-free optimization . . . 138
A.2 Supporting information for Chapter 4 . . . 140
A.2.1 Encoder topologies . . . 140
A.2.2 Feature processing networks . . . 142

A.2.3 Results of further transformations 143
A.3 Supporting information for Chapter 5 146
A.3.1 One twin part of the siamese multitask learning model 146
A.4 Supporting information for Chapter 6 148
A.4.1 Generalized spherical harmonics distance 148
A.4.2 Details of structure-property-mapping 149
A.4.3 Overview of the methods of the generation the orientation distribution function and measuring the distances 152
A.5 Supporting information for Chapter 7 153
A.5.1 Visualization of the optimizer evolution in the property space 153
A.5.2 Visualization of the validity and regression results 154
A.5.3 Distance preservation 155

Bibliography **157**

List of Figures

2.1 Schematic representation of a perceptron . . . 13
2.2 Plot of the Heaviside step function. . . . 13
2.3 Example of linear separable data with a decision boundary. . . . 14
2.4 Representation of the non-linearly separable XOR function. . . . 15
2.5 Schematic representation of a multilayer perceptron. . . . 16
2.6 Plots of the sigmoid, tanh, and ReLU activation functions. . . . 17
2.7 Visualization of siamese neural networks. . . . 21
2.8 Representation of an autoencoder. . . . 23
2.9 Schematic illustration of the workflow of the differential evolution algorithm. . . . 24
2.10 General multitask learning architecture. . . . 27
2.11 Example of an orientation histogram. . . . 33
2.12 Example of an pole figure. . . . 34
2.13 Schematic representation of an resistance spot welding arrangement. . . . 36
2.14 Schematic representation of an resistance spot welding procedure. . . . 36
2.15 (a) Voltage, (b) current and (c) power curves of a welding process. . . . 37

4.1 Multitask learning architecture for process curves. . . . 52
4.2 Comparison of the results for different numbers of latent features of the top-ranking topology CNN-VA2. . . . 59
4.3 Comparison of the results for numbers of latent features of 8 of the investigated topologies. . . . 60
4.4 Comparison of representations in feature spaces learned with autoencoder-only learning and with multitask learning. . . . 62
4.5 Estimated density functions. . . . 62
4.6 Transformed density and contour plot for welding guns. . . . 64
4.7 Reconstructed curves. . . . 64

5.1 General concept of the multitask learning based optimization approach. . . . 68
5.2 Schematic illustration of the structure space including the valid region and unknown microstructures. . . . 69
5.3 Multitask learning architecture for orientation histograms. . . . 70
5.4 Architecture of the Siamese multitask learning approach. . . . 72
5.5 Section of the orientation distribution function to visualize the cold rolled DC04 steel texture. . . . 76
5.6 Histograms of the anomaly scores for the data from the test set and the set of artificially generated anomalies. . . . 78

5.7 Projection of the training set showing different planes of the property space. 79
5.8 Histogram of pairwise distances of the set of identified textures and the baseline set for Target Region 1. 80
5.9 Texture that yields properties which are closest to the center of Target Region 1. . . 81
5.10 Histogram of mutual distances of the set of identified textures and the baseline set for Target Region 2. 81
5.11 Two exemplary textures from the set of identified textures. 82

6.1 Representation of the orientation distribution function by two-dimensional pole figures. 90
6.2 Representation of the orientation distribution function with one single orientation by two-dimensional pole figures. 94
6.3 Representation of the orientation distribution function with 100 orientations by two-dimensional pole figures. 95
6.4 Representation of the orientation distribution function with 512 orientations by two-dimensional pole figures. 96
6.5 Distances over process step curves for the textures with 512 orientations, with 100 orientations, and with one single orientation. 97
6.6 Distances over process step curves for the textures with 512 orientations for the GSH distance and Sinkhorn distance. 98
6.7 Representation of the orientation distribution function for the experimentally measured textures by two-dimensional pole figures. 99
6.8 Orientation distribution of the experimentally measured textures as sections through the Euler space. 100

7.1 Process-structure-properties linkage. 106
7.2 General concept for structure-guided materials and process design. 107
7.3 The neural networks-based Siamese multitask learning model and the optimization part. 108
7.4 Visualization of the diversity of microstructure sets within the microstructure space. 110
7.5 Visualization of the loss curves of the Siamese multitask learning model training for the test dataset and the training dataset. 114
7.6 Projections of the properties space showing the data distribution and the Target Region 1 and Target Region 2, respectively. 116
7.7 Density of pairwise distances between textures belonging to the sets of identified textures and between textures belonging to the benchmark sets. 117
7.8 Projections of the properties space showing the Target Region 1 and the calculated properties of the ten chosen goal textures. 118
7.9 Projections of the properties space showing the Target Region 2 and the calculated properties of the ten chosen goal textures. 118
7.10 Four exemplary textures of the ten chosen goal textures depicted as pole figure plots. 119
7.11 Sinkhorn texture distance between the produced and the goal texture over the episodes. 120
7.12 Pole figure plots of the targeted goal texture and the produced texture. 121

A.1 Conception of the variational autoencoder. 133
A.2 Depiction of the conditional variational autoencoder. 135
A.3 Visualization of the generative adversarial network. 135

A.4 Neural network that presents the conditional generative adversarial network. 136
A.5 Visualization of the Mahalanobis distance. 137
A.6 Visualization of the one-class support vector machine. 138
A.7 Instantiated encoder topologies for MLP. 140
A.8 Instantiated encoder topologies for CNN-VA1. 140
A.9 Instantiated encoder topologies for CNN-VA2. 140
A.10 Instantiated encoder topologies for CNN-VB1. 141
A.11 Latent feature space processing networks for classifier welding gun type. 142
A.12 Latent feature space processing networks for classifier steel sheet combination. . . . 142
A.13 Latent feature space processing networks for estimator welding spot diameter. 142
A.14 Latent feature space processing networks for decoder. 143
A.15 Investigated transformation: (Gun1, SSC7) → (Gun2, SSC7). 144
A.16 Investigated transformation: (Gun1, SSC7) → (Gun1, SSC8). 144
A.17 Investigated transformation: (Gun2, SSC7) → (Gun2, SSC8). 145
A.18 One twin part of the Siamese multitask learning model. 146
A.19 Topologies of the neural networks for learning the texture-property relations by the texture representation as orientation histograms. 150
A.20 Topologies of the neural networks for learning the texture-property relations by the texture representation as GSH for L=10. 150
A.21 Topologies of the neural networks for learning the texture-property relations by the texture representation as GSH for L=16. 150
A.22 Topologies of the neural networks for learning the texture-property relations by the texture representation as Pole figures. 151
A.23 Overview of the methods of the generation the orientation distribution functions and measuring the distances by the corresponding distance functions. 152
A.24 Projections of the properties space showing the Target Region 1 and the evolution of the optimization for the generations 0, 10, and 100. 153
A.25 Projections of the properties space showing the Target Region 2 and the evolution of the optimization for the generations 0, 10, and 100. 154
A.26 Results of the Siamese multitask learning model for predicting the properties. 155
A.27 Distribution of the validity loss and the regression loss, as well as Two-Dimensional UMAP spaces. 155
A.28 Visualization of the Sinkhorn distances and the l_1 distances. 156

List of Tables

2.1 Representation of the XOR function. 15

4.1 Welding process parameters. 57
4.2 Evaluation results of the performance of the models for obtaining an low dimensional latent feature space. 61

5.1 Definition of the fibers of bcc rolling textures. 74
5.2 Definition of the parameters D_i of the texture model. 75
5.3 Material dependent parameters used in this study. 75
5.4 Parameter ranges for D_i. 76
5.5 Used hyperparamters for training the Siamese multitask learning model. 77
5.6 Results for varying numbers of latent features of the texture-property mapping and the distance preservation. 78
5.7 Center points of the two target regions. 79

6.1 Material dependent parameters used in this study. 91
6.2 Results of the measured distances between each of the experimentally measured textures. 101
6.3 Results of the investigated textures for learning the texture-property relations. . . . 101
6.4 Distribution of the property space for the underlying data set. 102

7.1 The results for the properties prediction. 114
7.2 Distribution of the property space for the underlying data set. 115
7.3 Center points of the two target regions. 115
7.4 Properties of the produced crystallographic textures and the distance to their respective target regions. 120

A.1 Instantiated encoder topologies. 141
A.2 Latent feature space processing networks. 143
A.3 Topologies of the developed multi task learning model. 147
A.4 Topologies of the developed neural networks models for learning the structure-property relations. 151

Chapter 1

Introduction

1.1 Motivation

The design and optimization of materials is a central challenge across the sciences and engineering, as materials are fundamental to modern technology and infrastructure. The ability to design materials with tailored properties is crucial for meeting the demands of current applications, as exemplified by structural alloys such as steels, aluminum, and titanium in transportation, construction, and energy systems. A central concept in this context is the process–structure–property (PSP) linkage, originally proposed by (Olson, 1997) as a framework for computational materials design. Its broad application in polycrystal plasticity is thoroughly presented in (Roters et al., 2010), which describes how processing conditions influence grain-scale microstructural responses, affecting macroscopic material behavior. In addition, (Kalidindi et al., 2011) emphasized its relevance for linking microstructural data science and informatics methods to property prediction. Together, these studies establish the PSP linkage as a systematic foundation for understanding how processing conditions lead to microstructural features and how these features govern material properties.

In the context of metals, the PSP linkage begins with the processing of raw ores or recycled alloys, which are subjected to metallurgical operations such as melting, casting, rolling, forming, welding, or heat treatment. These processes modify the grain structure, influence crystallographic texture (Kocks et al., 1998), and control phase transformations (Porter et al., 2021). Collectively, these microstructural modifications determine key mechanical properties such as strength, ductility, toughness, and fatigue resistance. While this work focuses on metals, the PSP linkage is universal across material classes; for instance, ceramics such as alumina, produced by powder compaction and sintering, form dense microstructures with minimal porosity that result in exceptional hardness and wear resistance in cutting tools (Senthil Kumar et al., 2006; Grigoriev et al., 2019).

Yet, despite decades of progress, the predictive control of PSP linkages remains a fundamental scientific and engineering challenge. (Kuehmann and Olson, 2009) showed that traditional computational materials design often struggles with the vastness of design spaces, while (Pollock, 2010) highlighted the limitations of alloy development when pursued through incremental approaches. More recently, (Ramprasad et al., 2017) pointed out that the nonlinear and multiscale character of these linkages makes conventional strategies—whether empirical trial-and-error or high-fidelity simulations—resource-intensive and impractical for exploring large design spaces.

This difficulty arises because small variations in processing conditions can propagate across multiple length scales, leading to unpredictable changes in microstructure and properties (Agrawal and Choudhary, 2016). Consequently, exhaustive exploration of processing–structure combinations becomes computationally prohibitive and experimentally infeasible, slowing the pace of new materials discovery. (Lookman et al., 2019) further emphasized that such methods are insufficient for systematic exploration, underscoring the need for data-driven strategies that can efficiently capture nonlinear dependencies, thereby complementing conventional simulation and experimental approaches.

To address these challenges, the Integrated Computational Materials Engineering (ICME) paradigm has emerged as a systematic methodology that tightly integrates materials modeling and simulation tools across multiple length and time scales with process optimization and performance design. The vision of ICME was proposed by the National Research Council (National Research Council, 2008), which emphasized its potential to transform materials development by embedding computational methods directly into engineering practice. Building on this foundation, (Ghosh and Dimiduk, 2011) described how ICME frameworks link processing models with microstructure evolution and property prediction across multiple scales. (Allison, 2011) further highlighted that such integration enables concurrent, rather than sequential, design of materials and processes, thereby reducing development time.

Nevertheless, despite its significant impact, ICME workflows still face major limitations. They often rely on physics-based models that are computationally demanding, and they require extensive calibration with experimental data. As (Pablo et al., 2019) emphasized, these constraints limit scalability and make it difficult to efficiently explore the vast design spaces inherent to PSP linkages. A further complication arises from the growing demand for multifunctional materials, which frequently requires balancing multiple and potentially competing property requirements. Designing microstructures that simultaneously optimize, for example, strength and ductility or stiffness and toughness remains an active research challenge.

Machine learning (ML) has emerged as a transformative approach to address these challenges. By learning representations directly from data, ML models can capture nonlinear and multiscale PSP linkages that are difficult to model explicitly. (Liu et al., 2017) reviewed how ML can assist in addressing this challenge by enabling the prediction of material properties across complex design spaces. Building on this, (Noh et al., 2020) demonstrated an inverse design strategy for solid-state materials using continuous latent representations, showing how target properties can guide the generation of candidate structures. In a similar vein, (Xie et al., 2021) developed graph-based networks to learn atomic-scale structure–property relationships, further advancing the capability of inverse design frameworks. (Butler et al., 2018) highlighted how ML can accelerate molecular and materials discovery by enabling efficient prediction of properties that are otherwise expensive to compute. (Agrawal and Choudhary, 2019) provided one of the earliest comprehensive surveys of deep learning applications in materials science, establishing the foundations of the field. Building on this, (Guo et al., 2021) offered a broader perspective on the role of artificial intelligence and ML in the design of mechanical materials, emphasizing their potential for handling high-dimensional design problems. (Ling et al., 2017) demonstrated the use of Gaussian process regression as a surrogate modeling technique for high-dimensional materials design spaces, while (Ward et al., 2018) introduced a general-purpose ML framework capable of predicting a wide range of inorganic material properties. (Schmidt et al., 2019) further reviewed applications of ML in solid-state materials science, showcasing recent progress in data-driven

approaches to microstructure–property prediction. Taken together, these studies demonstrate how inverse design approaches can uncover processing conditions and microstructures that yield desired properties. These approaches depend on efficient models capable of both predicting and inverting PSP linkages, thus facilitating the exploration of feasible solutions across vast and complex design spaces.

Within this context, latent space learning has become particularly important. (Kingma and Welling, 2014) introduced the variational autoencoder (VAE), which enables the embedding of high-dimensional data into structured latent representations. (Bengio et al., 2013) provided a comprehensive perspective on representation learning, underscoring its importance for extracting meaningful latent features that support interpretability, optimization, and interpolation across design spaces.

A key challenge, however, is that single-task learning, in which a model is trained to solve only one prediction task in isolation, often fails to generalize across the diverse objectives inherent in PSP linkages. Multitask learning (MTL) provides a way forward by jointly learning across related prediction tasks, thereby exploiting shared structures in the data and producing richer, domain-aware latent space features. (Caruana, 1997) first established the principle of MTL in ML, showing how shared representations improve generalization across tasks. Building on this foundation, (Ruder, 2017) provided a comprehensive overview of deep learning–based MTL, emphasizing its effectiveness in domains with limited training data. More recently, (Standley et al., 2020) investigated which tasks should be learned together, demonstrating that the selection of related tasks is critical for achieving performance gains in MTL frameworks.

Such latent features are crucial not only for forward prediction but also for inverse design. (Noh et al., 2020) illustrated this by developing a continuous representation for solid-state materials that enables property-driven inverse design. Complementing this, (Kusampudi and Diehl, 2023) combined generative machine learning models with Bayesian optimization to design dual-phase steel microstructures, further showcasing how latent-space optimization can yield microstructures aligned with targeted mechanical properties.

1.2 Problem statement

In materials science, a central challenge lies in understanding how processing conditions shape microstructures, which in turn govern material properties. Process engineering adds further complexity, as dynamic systems evolve over time and require precise control of multiple variables to achieve desired outcomes. Traditional approaches—such as trial-and-error experimentation or computationally intensive simulations—have provided valuable insights but remain resource-demanding and insufficient for addressing the growing complexity of modern engineering problems. In this context, ML offers a transformative perspective by uncovering patterns in large datasets and modeling complex, non-linear interactions. These capabilities are particularly relevant within the PSP linkage, which provides the foundational framework for linking processes, microstructures, and material properties.

Despite decades of experimental and modeling efforts, the systematic exploitation of structure–property relationships for predictive and inverse design remains a significant challenge. This work addresses this challenge by focusing on three central aspects: (i) representing high-dimensional microstructure data in compact forms suitable for optimization, (ii) developing ro-

bust distance measures to quantify texture similarity, and (iii) enabling inverse design methods that identify feasible process paths leading to targeted microstructures with desired properties.

Research Hypothesis

This work proceeds from the hypothesis that MTL frameworks can learn shared latent representations by simultaneously addressing prediction tasks such as classification and regression. These shared latent representations reduce dimensionality while preserving task-relevant features, thereby providing a systematic basis for efficient inverse design of microstructures and process paths. In this way, the learned latent spaces enable both property-driven microstructure generation and process optimization, resulting in greater efficiency compared to traditional approaches such as trial-and-error experimentation or computationally intensive simulations.

To investigate this hypothesis, the thesis examines the potential of MTL-derived latent spaces for improving shared latent representation, achieving dimensionality reduction, and enabling inverse design in materials and process optimization. The research contributions are organized into four complementary studies that collectively address different aspects of materials and process design, with a focus on both forward modeling and inverse design.

1.3 Contribution

The **first contribution** of this work introduces a convolutional neural network (CNN)–based multitask learning (MTL) framework for extracting compact and condition-aware latent feature spaces from process curves. By combining reconstruction, classification, and regression tasks within a unified architecture, the approach ensures that the latent representation captures both stochastic variability and systematic dependencies on process conditions. Applied to resistance spot welding, the method demonstrates the ability to accurately reconstruct welding curves, classify tool and material configurations, and estimate welding spot diameters from a three-dimensional latent space. Compared to linear baselines such as principal component analysis, the CNN-based latent representation provides superior non-linear feature extraction, while also enabling probabilistic transformations of process curve distributions across different conditions. This capability establishes the foundation for hyper modeling of manufacturing processes, where conditional dependencies can be leveraged to predict properties of previously unseen process configurations.

The **second contribution** extends the MTL framework to a Siamese multitask learning (SMTL) framework, specifically designed to preserve distance-based relationships in the latent feature space. This extension addresses the enhancement of the initial approach by introducing a distance-preservation loss, ensuring that pairwise similarities between microstructures in the input space are approximately maintained in the learned representation. The SMTL framework jointly performs reconstruction, property prediction, validity assessment, and distance preservation, thereby creating a latent space in which optimization can be carried out efficiently. Leveraging this representation, an optimization strategy was developed to generate diverse sets of microstructures that satisfy specified property constraints while explicitly avoiding convergence to single solutions. The method was validated in crystallographic texture optimization, where it successfully identified multiple distinct textures aligned with targeted property requirements. This demonstrated its potential to support structure-guided processing strategies and

to provide input for process control approaches that demand both accuracy and diversity in solution sets.

The **third contribution** focuses on the fundamental challenge of quantifying distances between crystallographic textures, which are central to understanding and modeling anisotropic material behavior. Since texture similarity plays a critical role in process design and in the optimization of texture–property relationships, this work developed and systematically evaluated several distance measures for orientation distribution functions (ODFs). Approaches included a generalized spherical harmonics (GSH)–based distance, the Sinkhorn distance (efficient approximation of the Earth Mover's Distance (EMD)) and Chi-Squared distance for orientation histogram representations, and a pole-figure-based distance. Comparative analyses revealed that the Sinkhorn distance provided higher sensitivity for small differences, while the GSH distance was more effective for capturing larger differences, thereby offering complementary perspectives. Application to experimental data from hot rolling, cold rolling, and heat treatment demonstrated that all measures captured the evolution of process states for non-spiky ODFs. Additionally, the suitability of different texture representations for structure-property modeling was investigated, showing that GSH-based representations delivered accuracy comparable to pole figures at lower computational cost, while both significantly outperformed orientation histograms.

The **fourth contribution** presents a holistic inverse design framework that integrates the Siamese multitask learning optimization (SMTLO) approach from the second contribution with the Sinkhorn distance measure developed from the third contribution. This integration enabled end-to-end optimization of PSP relationships, explicitly in complex manufacturing scenarios. A case study on metal forming, involving a combinatorial design space of 25^7 possible processing paths, demonstrated the scalability of the approach. Using only 76,980 samples, the SMTLO method successfully identified diverse sets of near-optimal crystallographic textures that fulfilled targeted property requirements. Subsequent process optimization via the multi-equivalent goal structure-guided processing path optimization (MEG-SGGPO) approach steered the forming process to the best reachable texture within just 400 episodes. By explicitly exploiting the non-uniqueness of both materials and process design problems, the approach not only produced multiple valid microstructures but also determined the most reachable one under process optimization constraints.

1.4 Thesis outline

The remainder of this work is structured as follows: Chapter 2 provides an introduction to the fundamentals that form the foundation of this work. This includes the basics of ML and material science. Chapter 3 provides a literature review of the state of the art regarding machine learning within materials science. Chapter 4 focuses on understanding and modeling how process conditions influence the evolution of process curves, which are represented as time series. A latent feature space, derived from the process data, is learned through multitask learning (MTL). To preserve the distances from the original space to the latent feature space, the concept of MTL for obtaining latent features is extended in Chapter 5, which is utilized by using Siamese neural networks. The resulting latent feature space is used by an optimizer that generates target microstructures for desired properties. Chapter 6 outlines the importance of distance measurement for both process design and material design. In Chapter 7, the most important

results of distance-preserving feature extraction and the use of suitable texture descriptors and distance measurements are applied to the application of the holistic optimization approach. In Chapter 8 a conclusion and summary is given, Chapter 9 provides perspective for future work.

Chapter 2

Background

This chapter provides a comprehensive foundation for understanding the key concepts and methods underlying this work. It is divided into three main sections. Section 2.1 introduces the basic principles of machine learning, including an overview of the field, neural networks, dimensionality reduction techniques, and gradient-free optimization. Further topics in machine learning, such as generative approaches and anomaly detection, are presented in Section A.1 of the Appendix. Furthermore, in Section 2.2 an approach for learning of an universal low dimensional latent feature space is presented. These topics lay the foundation for the computational techniques used later in the work. Section 2.3 shifts the focus to the realm of materials science, covering essential concepts such as crystallographic texture, Taylor-type material models, and resistance spot welding. These principles are critical to understanding the materials and processes studied in this work.

2.1 Fundamentals of machine learning

2.1.1 Overview of machine learning

Machine learning (ML) is a subfield of artificial intelligence (AI) that aims to develop algorithms capable of learning patterns from data and making decisions or predictions without being explicitly programmed for specific tasks. In his book, Mitchell (Mitchell, 1997) provides one of the foundational definitions of ML:

> "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

The task T defines the scope of the problem to be solved and represents the set of activities or objectives assigned to the learning system. These tasks may vary in nature, encompassing classification, regression, clustering, or decision-making, and they provide the context in which learning takes place. By formally specifying the task, the definition ensures that the purpose of the learning process is clearly defined.

The experience E denotes the source of information through which the system acquires knowledge. It encompasses training data, observations, or interactions with an environment, depending on the learning paradigm. In supervised learning, E typically consists of labeled

datasets that guide the program in associating inputs with correct outputs. In unsupervised learning, E may be expressed as unlabeled data from which the system must discover hidden patterns or structures. In reinforcement learning, E corresponds to feedback from the environment in the form of rewards or penalties following specific actions. Thus, experience represents the mechanism by which the system adapts, and its nature fundamentally shapes the kind of knowledge that can be acquired.

The performance measure P provides the criterion by which progress in learning is evaluated. It transforms the abstract notion of improvement into a quantifiable quantity, such as accuracy, mean squared error, likelihood, or cumulative reward. Without such a measure, the concept of learning would remain vague, as one could not rigorously determine whether the system's behavior had improved over time. Importantly, the performance measure must be chosen with respect to the specific task and application domain, since different measures may yield different judgments about the adequacy of the learned behavior.

Together, these three components form a rigorous framework that distinguishes learning from mere programming. A program can be said to have learned when its ability to perform a defined task improves as a result of accumulating relevant experience. This improvement must be assessed by an appropriate performance measure, which provides a rigorous criterion for evaluating the extent of learning.

As the quantity of data and computational power has increased, ML has become an indispensable method across numerous domains, including healthcare, finance, engineering, and scientific research. In essence, ML enables systems to enhance their performance through experience derived from data, thereby significantly advancing decision-making processes in complex, data-driven environments.

ML methodologies can be broadly categorized into five main types: supervised learning, unsupervised learning, semi-supervised learning, self-supervised learning, and reinforcement learning. Each of these approaches serves distinct purposes and addresses different types of problems. This section will provide an in-depth discussion of these ML paradigms, exploring their core principles, key algorithms, and applications, as well as their advantages and limitations.

Supervised learning

Supervised learning is one of the most prevalent forms of ML. The term supervised learning is used to describe a learning process whereby a model is provided with input-output pairs, with the objective of mapping inputs to corresponding outputs with a high degree of accuracy. The system is trained on a dataset comprising inputs, which are features or attributes, and corresponding outputs, which are referred to as labels. The objective of the model is to generalize the mapping in order to predict the output for previously unseen data. In this sense, supervised learning can be viewed as a form of function approximation (Goodfellow et al., 2016). Supervised learning tasks can be further classified into two subcategories: classification and regression.

In the context of classification tasks, the objective is to assign each input instance to one of a predefined set of categories. Examples of classification tasks include image recognition, in which images are classified into predefined categories (e.g., dog, cat, etc.), and email filtering, in which emails are classified as either spam or non-spam. The most commonly utilized algorithms for classification tasks include decision trees, random forests, support vector machines (SVMs), and the k-nearest neighbors algorithm (k-NN). The fundamental process underlying the operation

of decision trees is recursive partitioning of data into subsets, based on the most informative feature at each step, forming a tree structure (Breiman et al., 1984). As outlined by (Breiman, 2001), random forests represent an extension of the concept of decision trees. An ensemble of decision trees is constructed, with each tree trained on a distinct random subset of the data. The final prediction is then generated by aggregating the predictions of all trees, which is frequently achieved through voting or averaging. SVMs (Cortes and Vapnik, 1995) represent a robust classification technique that identifies the optimal hyperplane for differentiating between data points belonging to disparate classes, thereby maximizing the margin between them. The non-parametric k-NN algorithm (Cover and Hart, 1967; Fix and Jr., 1989) operates on the principle of instance-based learning, where the classification method identifies the k nearest training samples and assigns the most common label among them to new data points. The value of the parameter k is typically specified by the user, and the appropriate distance metric between data points is selected based on the characteristics of the dataset in question (Duda et al., 2000). This may be Euclidean, Manhattan (Riesz, 1910), or Minkowski (Minkowski, 1910). As outlined in Section 2.1.2, neural networks can also be employed for classification and regression purposes.

In regression tasks, the objective is to predict a continuous value based on the input data, which is typically represented as a set of numerical values. Regression tasks may be exemplified by the prediction of housing prices based on features such as location, size, and amenities, or the prediction of temperature given a set of environmental variables. The most commonly utilized algorithms for regression include traditional methods such as linear regression (Seber and Lee, 2012; Weisberg, 2005) and polynomial regression (Draper and Smith, 1998; Aiken and West, 1991). These methods model the relationship between input features and the target variable as either a linear or nonlinear function, respectively. These methods are often straightforward to interpret and apply; however, they may encounter difficulties in addressing complex relationships in the data. In addition to these, more sophisticated ML techniques, such as support vector regression (SVR) and XGBoost, have gained prominence in the field of regression analysis. SVR (Drucker et al., 1996) represents an extension of SVMs that is designed to identify the optimal hyperplane within a specified margin of tolerance, with the objective of predicting continuous outcomes. SVR is particularly effective when the relationship between the input features and the target variable is complex, yet the data set is not excessively high-dimensional. XGBoost (Chen and Guestrin, 2016), an abbreviation for eXtreme Gradient Boosting, represents another efficacious and pervasive approach to regression. XGBoost is based on gradient boosting, a method whereby models are constructed sequentially, with each subsequent model rectifying the errors of the previous one. It is renowned for its scalability and efficiency, rendering it especially advantageous in real-world applications involving large datasets and intricate feature interactions, such as forecasting stock prices or customer demand.

Unsupervised learning

The distinction between unsupervised and supervised learning lies on the manner in which the data is presented to the model. In unsupervised learning, the data is not labeled, and the objective is not to predict specific outcomes, but to identify the underlying patterns, structures, or relationships within the data set. Unsupervised learning is a technique that is frequently employed in a multitude of tasks, including clustering and dimensionality reduction. Anomaly detection will be discussed in Section A.1.4.

Clustering is one of the most frequently employed unsupervised learning techniques. The objective of clustering is to identify groups of data points (clusters) that reflect the underlying similarities and dissimilarities within the data. One of the most frequently utilized clustering algorithms is k-means, which partitions the data into k clusters by minimizing the variance within each cluster. Despite its effectiveness and computational efficiency, the k-means algorithm, as originally described by (MacQueen, 1967), requires the number of clusters k to be predefined and may encounter difficulties in identifying clusters of non-spherical shapes. Another prevalent approach is hierarchical clustering (Johnson, 1967), which generates a hierarchical structure of clusters that can be represented as a dendrogram, elucidating the interrelationships between disparate data points. In contrast to the k-means algorithm, hierarchical clustering does not require the a priori specification of the number of clusters, thereby conferring enhanced flexibility for certain applications. In addition to k-means and hierarchical methods, DBSCAN (Density-Based Spatial Clustering of Applications with Noise) (Ester et al., 1996) has emerged as a highly effective clustering algorithm, particularly for datasets exhibiting complex structures. In contrast to the k-means algorithm, the DBSCAN method does not require the predefinition of the number of clusters. In contrast to the aforementioned techniques, DBSCAN identifies clusters based on the density of data points within a given region. The algorithm groups points that are closely packed together, while treating points in low-density regions as noise. This makes it particularly useful for identifying clusters of arbitrary shapes and handling outliers.

Dimensionality reduction techniques are of considerable significance in the domain of unsupervised learning, particularly in the context of high-dimensional datasets. The objective of these techniques is to reduce the number of features in a dataset while maintaining its intrinsic structure. Principal component analysis (PCA), t-distributed stochastic neighbor embedding (t-SNE), and the Uniform Manifold Approximation and Projection (UMAP) algorithm represent three prominent techniques for dimensionality reduction. PCA, initially proposed by (Hotelling, 1933), is a technique that projects data into a lower-dimensional space through the identification of the directions (or principal components) along which the variance in the data is maximized. In contrast, t-SNE (Maaten and Hinton, 2008) is a commonly utilized technique for the visualization of high-dimensional data, whereby the data is projected into two or three dimensions, thus facilitating the detection of clusters or groupings within the data set. T-SNE operates by minimizing the divergence between two probability distributions. One method assesses the pairwise similarities of the data points in the high-dimensional space, while the other assesses them in the low-dimensional space. This results in the effective preservation of local structure, thereby enabling the visualization of complex patterns. The UMAP algorithm (McInnes et al., 2018) is a dimensionality reduction technique that preserves both global structure and local relationships in high-dimensional data. In contrast to linear techniques such as PCA, UMAP is capable of modeling non-linear structures, rendering it an effective approach for visualizing intricate datasets. The algorithmic process is composed of two discrete stages. In the initial stage, the manifold structure is approximated through the construction of a k-nearest-neighbor graph. Subsequently, a low-dimensional representation is optimized through the application of stochastic gradient descent (McInnes et al., 2018).

Semi-supervised learning

Semi-supervised learning represents a ML paradigm that incorporates aspects of both supervised and unsupervised learning. It is especially beneficial in situations where obtaining labeled data is expensive or time-consuming, yet a significant amount of unlabeled data is available (Zhu, 2005). Semi-supervised learning employs a limited labeled dataset to direct the learning process, while leveraging a copious unlabeled data set to elucidate the intrinsic structure of the data distribution (Chapelle et al., 2009). The process typically begins with the training of an initial model on the labeled data, which is then used to predict pseudo-labels for the unlabeled dataset. These pseudo-labeled examples are incorporated in subsequent training iterations, thereby enabling the model to refine its predictions and generalize more effectively. The integration of unlabeled data into semi-supervised learning can enhance the model's generalization capabilities, particularly in scenarios where labeled samples are limited (Engelen and Hoos, 2020).

Self-supervised learning

Self-supervised learning occupies a position between the supervised and unsupervised learning paradigms. In contrast to traditional supervised learning, which requires manually labeled datasets, self-supervised learning eliminates the need for labeled data by automatically generating supervisory labels from the data itself. This is achieved by leveraging the intrinsic structures and relationships within the data to generate pseudo-labels, thereby enabling the model to learn meaningful representations without the necessity for external annotations (Jing and Tian, 2021). A number of approaches have emerged as prominent techniques in the field of self-supervised learning across diverse domains. One of the most successful methods is contrastive learning, which has demonstrated particular efficacy in the domain of computer vision. In contrastive learning, models learn representations by contrasting positive pairs, which are different augmented views of the same image, with negative pairs, which are samples from different images. SimCLR represents a notable example of this approach, employing techniques such as cropping, rotation, and color distortion to create positive pairs. The objective is to train the model to cluster positive pairs in close proximity within the learned representation space, while separating negative pairs from one another (Chen et al., 2020). This framework has been demonstrated to yield state-of-the-art results when applied to large-scale image datasets.

Reinforcement learning

Reinforcement learning (RL) represents a discrete ML paradigm that prioritizes the acquisition of knowledge through interaction with an external environment. In RL, an agent learns to make decisions by taking actions within an environment and receiving feedback in the form of rewards or penalties. The objective of the agent is to maximize the cumulative reward over time by identifying the optimal policy, which maps states of the environment to actions. In contrast to supervised learning, RL does not necessitate labeled data; rather, it learns through trial and error (Sutton and Barto, 2018). The solution to RL problems is frequently expressed as a markov decision process (Bellman, 1957; Puterman, 1994), wherein the agent interacts with an environment that is defined by states, actions, and rewards. As time progresses, the agent proceeds to explore the environment by undertaking actions and subsequently observing the resulting outcomes. By balancing exploration (testing novel actions to ascertain their effects)

and exploitation (utilizing known actions that maximize reward), the agent gradually enhances its policy. The principal algorithms in RL include Q-learning (Watkins and Dayan, 1992), policy gradients (Sutton et al., 2000), and actor-critic methods (Konda and Tsitsiklis, 2000). Q-learning is a model-free algorithm that seeks to ascertain the value of each action in a given state, with estimates being updated based on the rewards received. In contrast, policy gradient methods directly optimize the agent's policy by adjusting the probability distribution over actions in a given state, as opposed to learning a value function. Actor-critic methods integrate the benefits of both approaches, whereby the actor updates the policy and the critic assesses the actions taken (Mnih et al., 2015).

2.1.2 Neural networks

This section presents an examination of the fundamental and advanced architectures of neural networks, which form the basis of modern deep learning. A neural network is a computational model that draws inspiration from the structure and function of the human brain. It is designed to process data and identify patterns. The model consists of interconnected layers of nodes (neurons) that transform input data through weighted connections and activation functions to produce an output. The discussion begins with the perceptron, the most fundamental neural network model, and subsequently progresses to the multilayer perceptron, which extends the perceptron into a more sophisticated and versatile architecture. The role of activation functions, which are indispensable for introducing non-linearity and enabling networks to learn complex patterns, is also examined. Moreover, the section addresses the convolutional neural network, which is a specialized architectural configuration for processing structured data, such as images. The discussion then concludes with an overview of Siamese neural networks, which are particularly adept at similarity learning and comparison tasks.

Perceptron

The perceptron algorithm, initially proposed by Frank Rosenblatt in (Rosenblatt, 1958), represents one of the earliest models of artificial neural networks and constitutes a pivotal milestone in the history of ML. The perceptron was developed as a theoretical model to investigate the potential functionality of biological neurons. The objective of the perceptron was to classify input data into binary categories through a process of supervised learning. Despite its simplicity, the perceptron model established the foundation for subsequent, more sophisticated neural networks and continues to be a pivotal concept in understanding the evolution of ML. The development of the perceptron occurred during a period when researchers were beginning to explore the possibility of replicating human cognitive abilities, such as pattern recognition and decision-making, in machines. Building on the theoretical framework proposed by McCulloch and Pitts (McCulloch and Pitts, 1943), who developed a model for artificial neurons based on binary logic, Rosenblatt sought to create a practical algorithmic solution. His work was motivated by the question of whether machines could learn from examples in a manner analogous to humans, which ultimately contributed to the broader fields of cybernetics and artificial intelligence. Nevertheless, despite this enthusiasm, the perceptron was constrained by both technical and theoretical limitations that would only be addressed in subsequent decades.

The perceptron is fundamentally a linear classifier. It is a single-layer neural network that employs a set of weights to map input data to an output decision. The perceptron is designed to

perform binary classification, whereby input vectors are assigned to one of two classes based on their linear separability. In the perceptron model (Fig. 2.1), the input vector $\boldsymbol{x} = [x_1, x_2, \ldots, x_n]$, the label $y \in \{0, 1\}$, and the weight vector $\boldsymbol{w} = [w_1, w_2, \ldots, w_n]$ are central to its function and learning process. The input vector $\boldsymbol{x}$ consists of features for a single data instance, where each feature x_i contributes to the perceptron's classification decision. The ground truth label y indicates the target class (typically 0 or 1). The weight vector $\boldsymbol{w}$ holds the parameters that the perceptron learns during training. Each weight w_i determines the influence of the corresponding feature x_i on the decision. Initially set to small values, the weights are adjusted iteratively using the perceptron learning rule to reduce classification errors.

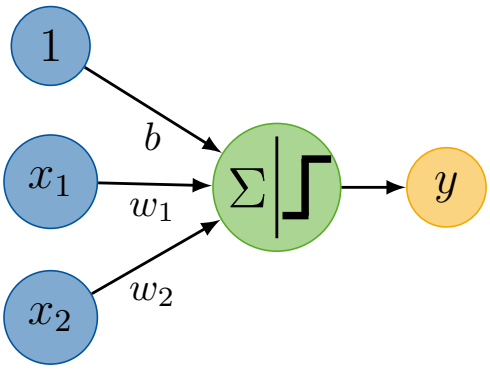

Figure 2.1: Schematic representation of a perceptron.

Given this fundamental basement, the perceptron computes a weighted sum z, which is the linear combination of the input vector $\boldsymbol{x}$ and the the weight vector $\boldsymbol{w}$. This is expressed as:

$$z = \boldsymbol{w}^\top \cdot \boldsymbol{x} + b = \sum_{i=1}^{n} w_i x_i + b, \tag{2.1}$$

where b is the bias term, which serves to displace the decision boundary from its original position. Subsequently, the perceptron applies a step activation function $\phi(z)$, which is also referred to as the Heaviside function (Whittaker and Watson, 1927), to transform the weighted sum z into a binary output, that is the prediction $\hat{y}$. If the weighted sum z surpasses a specified threshold θ, the perceptron outputs 1, signifying the designated class. Conversely, if the weighted sum falls below the threshold, the perceptron outputs 0, indicating the alternative class. This is illustrated in Fig. 2.2 and described as follows:

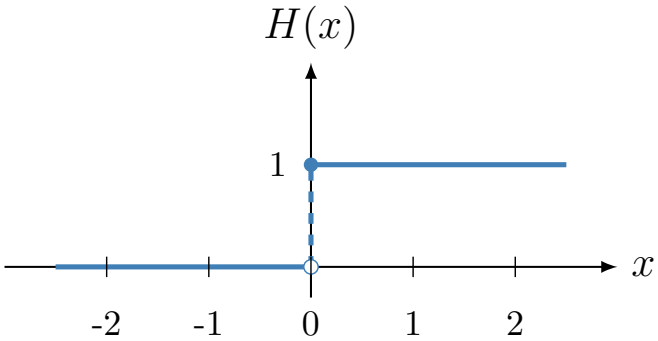

Figure 2.2: Plot of the Heaviside step function.

$$\hat{y} = \phi(z) = \begin{cases} 1 & \text{if } z > \theta, \\ 0 & \text{if } z \leq \theta. \end{cases} \tag{2.2}$$

The perceptron makes its decision based on the linear combination of the input features and their associated weights. However, the key challenge is to determine the appropriate weights that correctly classify the training data. This is where the perceptron's learning rule comes into play. The perceptron learning rule is an iterative process that adjusts the weights and bias based on the error between the predicted output $\hat{y}$ and the target label y for each input data. Specifically, the perceptron updates its weights after each training sample, where the weight update rule for the perceptron is given by:

$$\Delta w_i = \eta \cdot (y - \hat{y}) \cdot x_i, \tag{2.3}$$

where Δw_i is the change in the weight w_i, η is the learning rate, a small positive constant that controls the size of the weight updates. The weights are updated as follows:

$$w_i^{(t+1)} = w_i^{(t)} + \eta \cdot (y - \hat{y}) \cdot x_i, \tag{2.4}$$

where $w_i^{(t)}$ is the current weight for feature x_i after iteration t, $w_i^{(t+1)}$ represents the updated weight for feature x_i after iteration $t + 1$. The bias term b is also updated similarly:

$$b^{(t+1)} = b^{(t)} + \eta \cdot (y - \hat{y}). \tag{2.5}$$

In essence, if the perceptron makes a correct prediction, the weights remain unchanged. However, if the perceptron prediction makes an incorrect prediction ($y \neq \hat{y}$), the weights are adjusted in the direction that will reduce the error, bringing the output closer to the correct classification in subsequent iterations. The iterative nature of this learning rule allows the perceptron to gradually converge to a set of weights that minimizes classification errors for linearly separable data (c.f. Fig. 2.3).

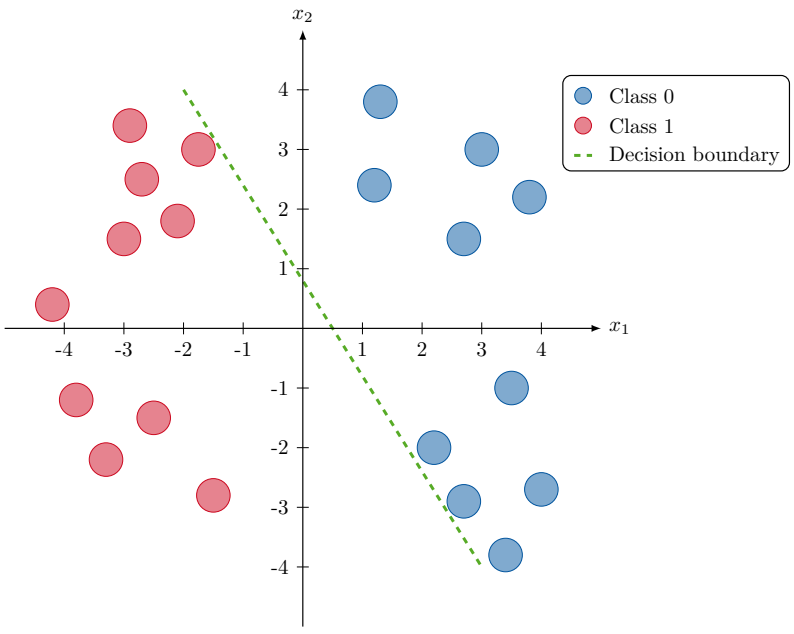

Figure 2.3: Example of linear separable data with a decision boundary.

A fundamental attribute of the perceptron learning rule is its guaranteed convergence for datasets that can be linearly separated. If a solution exists - that is, if there is a hyperplane that perfectly separates the two classes - the perceptron will find it in a finite number of iterations (Minsky and Papert, 1969). In the case of more complex problems where the classes are not linearly separable, the perceptron is unable to identify a suitable decision boundary. For example, the exclusive OR (XOR) function is not linearly separable, as its output cannot be separated into two distinct regions by a single straight line in two-dimensional space. This limitation was famously elucidated by Minsky and Papert in their seminal work, Perceptrons (Minsky and Papert, 1969). The researchers demonstrated that a single-layer perceptron was unable to solve problems that required the construction of non-linear decision boundaries, where the XOR problem has become a paradigmatic example of this limitation. In certain tasks, XOR is a binary classification problem, where the inputs and corresponding outputs are shown in Table 2.1.

x_1	x_2	XOR (x_1, x_2)
0	0	0
0	1	1
1	0	1
1	1	0

Table 2.1: Representation of the XOR function.

Upon visualizing the points corresponding to the XOR table in a two-dimensional plane (see Fig. 2.4), it becomes evident that they are not linearly separable. Given that the XOR function necessitates two distinct boundaries (one separating instances where $x_1 = x_2$ from those where $x_1 \neq x_2$), it is evident that a perceptron is an inadequate approach for addressing this task.

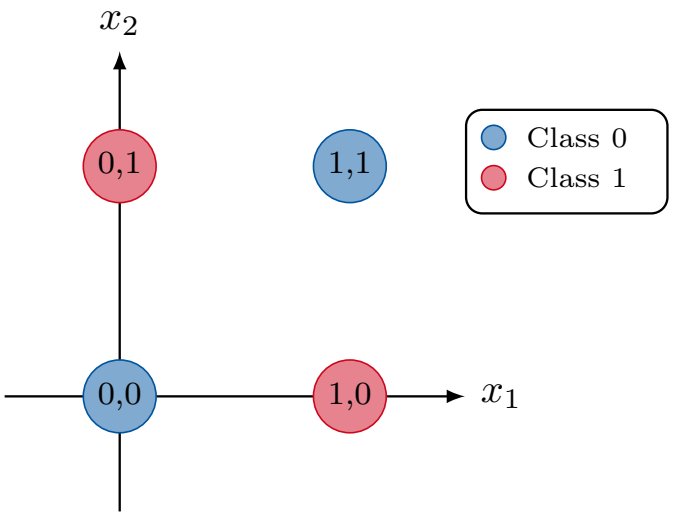

Figure 2.4: Representation of the non-linearly separable XOR function.

The inability of single-layer perceptrons to solve non-linearly separable problems like XOR constituted a pivotal factor in the evolution of MLP. The incorporation of an additional layer (hidden layer) between the input and output layers enables MLP to model more intricate and non-linear relationships. The introduction of a hidden layer enables the model to transform the

input data through non-linear activation functions, thereby allowing it to construct non-linear decision boundaries and to solve problems that a single-layer perceptron is unable to.

Multilayer perceptron

A multilayer perceptron (MLP) is a type of feedforward neural network comprising multiple layers of neurons, wherein each layer is fully connected to the subsequent one (see Fig. 2.5). An MLP typically comprises three types of layers: an input layer, one or more hidden layers, and an output layer. The neurons in each layer perform weighted summations of their inputs and subsequently apply a non-linear activation function. The ability of MLPs to approximate complex, non-linear mappings between inputs and outputs is enabled by the stacking of multiple layers.

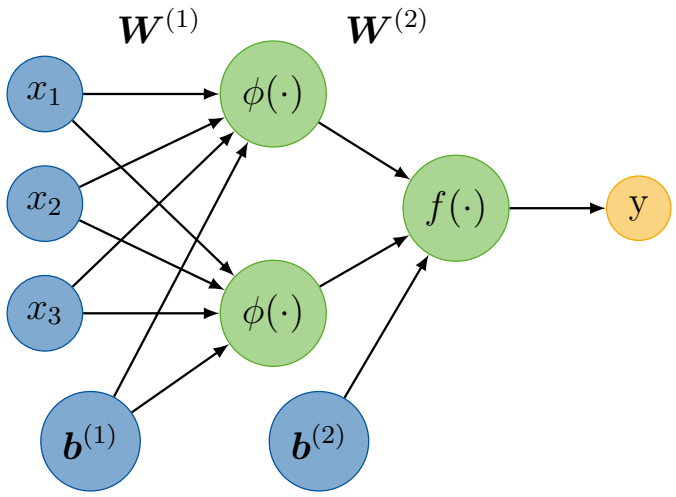

Figure 2.5: Schematic representation of a multilayer perceptron.

The perceptron can be extended to an MLP with one hidden layer as follows: Let $\boldsymbol{W}^{(1)} \in \mathbb{R}^{m \times n}$ denote the weight matrix connecting the input layer with n features to the the hidden layer with m neurons. Similarly, let $\boldsymbol{W}^{(2)} \in \mathbb{R}^{1 \times m}$ denote the weight matrix connecting the hidden layer to the output layer. The bias terms are represented by the vectors $\boldsymbol{b}^{(1)} \in \mathbb{R}^m$ for the hidden layer and $\boldsymbol{b}^{(2)} \in \mathbb{R}$ for the output layer. The output of the hidden layer, designated as $\boldsymbol{h}$, is calculated according to the following equation:

$$\boldsymbol{h} = \phi(\boldsymbol{W}^{(1)} \cdot \boldsymbol{x} + \boldsymbol{b}^{(1)}), \tag{2.6}$$

where $\phi(\cdot)$ is a non-linear activation function. The output of the network $\hat{y}$ is then:

$$\hat{y} = f(\boldsymbol{W}^{(2)} \cdot \boldsymbol{h} + \boldsymbol{b}^{(2)}), \tag{2.7}$$

where $f(\cdot)$ maps the hidden layer's outputs to the final prediction. In binary classification problems, this is typically the sigmoid function (c.f. Eq. (2.9)). A defining feature of MLPs is the presence of multiple hidden layers, which collectively contribute to the overall complexity of the model. The network is capable of learning hierarchical representations of the input data through the process of stacking layers, whereby deeper layers are able to capture increasingly abstract features. This is achieved through the network's ability to learn through the process of stacking layers, whereby deeper layers are capable of capturing increasingly abstract features.

The key breakthrough that made MLPs practical for ML tasks was the development of the backpropagation algorithm (Rumelhart et al., 1986). Backpropagation is a supervised learning technique that enables MLPs to learn by adjusting weights and biases based on errors in their predictions. It is a widely used method for training neural networks, enabling efficient computation of gradients required to minimize a network's loss function. Introduced by (Rumelhart et al., 1986), backpropagation calculates the partial derivatives of the loss function with respect to each parameter, allowing gradient-based optimization algorithms, such as gradient descent, to iteratively update the model's parameters and effectively reduce the loss.

The backpropagation algorithm is composed of three principal phases: the forward pass, the backward pass, and the gradient descent step. In the forward pass, the input variable $\boldsymbol{x}$ is conveyed through each layer of the network, yielding the network's prediction $\hat{y}$. Once the forward pass is completed, the loss function is usually defined as the squared difference as:

$$\mathcal{J}(y, \hat{y}) = (y - \hat{y})^2, \tag{2.8}$$

which can be expressed as a measure of the discrepancy between the predicted values, represented by $\hat{y}$, and the ground truth target values, represented by y. In the backward pass, the algorithm calculates the gradients of the loss function with respect to each weight and bias, layer by layer, moving from the output layer back to the input layer. This is achieved using the chain rule of calculus. By recursively applying the calculations from the output layer to the input layer, the gradients of the loss with respect to all weights and biases in the network are obtained. Once the gradients have been computed, the final stage is the gradient descent step. Here, each parameter is adjusted in the direction that reduces the loss function, using the calculated gradients.

By repeating the forward pass, backward pass, and gradient descent steps over multiple epochs, the backpropagation algorithm iteratively reduces the loss, enabling the neural network to learn and make increasingly accurate predictions. This combination of forward propagation to make predictions, backpropagation to calculate gradients, and gradient descent to update parameters is the foundation of training in deep learning.

Activation functions

Activation functions are of crucial importance in the context of neural networks, as they facilitate the introduction of non-linearity, thereby enabling the model to capture complex patterns in data. The most commonly used activation functions are the sigmoid, hyperbolic tangent (tanh), and rectified linear unit (ReLU) functions (see Fig. 2.6), each of which possesses distinctive properties and implications for neural network training.

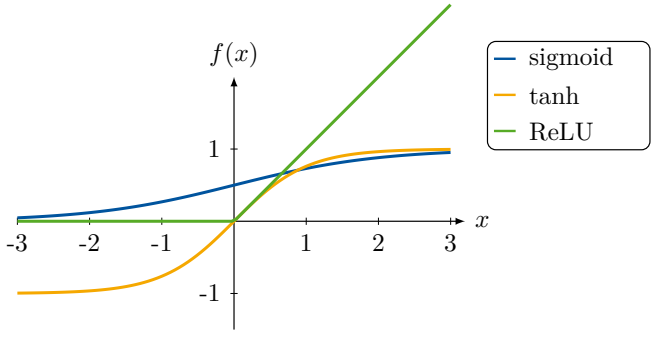

Figure 2.6: Plots of the sigmoid, tanh, and ReLU activation functions.

The sigmoid function σ, defined as:

$$\sigma(\boldsymbol{x}) = \frac{1}{1 + e^{-\boldsymbol{x}}}, \tag{2.9}$$

maps input values to a range between 0 and 1, rendering it suitable for use in output layers in binary classification tasks, where a probabilistic interpretation is required (Rumelhart et al., 1986). However, due to its asymptotic behavior, the sigmoid function is susceptible to the vanishing gradient problem. When the inputs are either exceedingly large or exceedingly small, the gradients approach zero, which can impede weight updates during training (Bengio et al., 1994). This phenomenon can render deep networks with numerous layers slow to train when sigmoid activations are employed, due to the aforementioned issues. The tanh function, expressed as:

$$\tanh(\boldsymbol{x}) = \frac{e^{\boldsymbol{x}} - e^{-\boldsymbol{x}}}{e^{\boldsymbol{x}} + e^{-\boldsymbol{x}}}, \tag{2.10}$$

maps inputs to values between -1 and 1. As with the sigmoid function, tanh introduces non-linearity, but with outputs centered around zero, which helps to mitigate issues related to scale and offsets (LeCun et al., 2012). The zero-centered output can facilitate faster convergence during training in comparison to the sigmoid function, particularly in deeper networks. Nevertheless, despite these advantages, the tanh function still encounters the vanishing gradient problem for large input magnitudes, which can impede learning in deep networks (Bengio et al., 1994). The ReLU function is applied element-wise and expressed as

$$\mathrm{ReLU}(\boldsymbol{x}) = (\max(0, x_1), \max(0, x_2), \ldots, \max(0, x_n)), \tag{2.11}$$

has become the most prevalent activation function, particularly in the context of hidden layers (Nair and Hinton, 2010). The ReLU function performs a binary mapping of inputs, assigning a value of zero to all inputs with a negative value and retaining only those inputs with a positive value. This simplicity circumvents the vanishing gradient issue commonly observed in sigmoid and tanh functions, as the gradient for positive inputs remains consistently non-zero, facilitating more rapid and efficient training (Glorot et al., 2011). However, the use of ReLU introduces a distinct challenge, known as the dying ReLU problem. This phenomenon occurs when neurons become inactive if they consistently receive negative inputs, resulting in zero gradients that impede their ability to learn. To address this issue, variants such as leaky ReLU have been proposed, which allow for a small gradient for negative inputs (Maas et al., 2013).

Convolutional neural networks

Convolutional neural networks (CNNs) represent a specialized class of artificial neural networks, designed to process data with a grid-like topology. CNNs are highly effective at capturing spatial hierarchies in data through the application of convolutional layers, rendering them well-suited for applications such as image processing, time series forecasting, and natural language processing. A significant benefit of CNNs is their capacity to automatically discern spatial characteristics, thereby eliminating the necessity for manual feature extraction.

The evolution of CNNs can be traced back to the 1980s, when Kunihiko Fukushima introduced the neocognitron, a hierarchical, self-organising neural network model inspired by the

visual cortex of animals (Fukushima, 1980). This foundational work provided the basis for subsequent advancements in convolutional architectures. Nevertheless, it was not until the early 1990s that CNNs began to gain traction, particularly with the introduction of LeNet by Yann LeCun and his colleagues. The objective of LeNet was to facilitate the recognition of handwritten digits within the MNIST dataset. The model was successful in applying convolutional layers for feature extraction, max-pooling layers for dimensionality reduction, and fully connected layers for classification (LeCun et al., 1998). CNNs achieved widespread popularity in 2012, when AlexNet, developed by Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, demonstrated groundbreaking performance in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC). The AlexNet model demonstrated that deep CNNs trained using graphics processing units (GPUs) could outperform traditional ML techniques in large-scale image classification tasks (Krizhevsky et al., 2012). The following section will present an overview of both one-dimensional and two-dimensional CNNs.

One-dimensional convolutional neural networks: A one-dimensional (1D) CNN is designed to process sequential data, including time series data, audio signals, and textual data. In a 1D CNN, the convolution operation is applied across the temporal or sequential dimension of the data. The convolution operation enables the capture of local dependencies within the sequence, rendering 1D CNNs highly effective in the detection of patterns such as peaks, trends, or periodicities. The mathematical expression for the 1D convolution operation is as follows:

$$s(t) = (\boldsymbol{x} * \boldsymbol{w})(t) = \sum_{i=1}^{k} w_i x_{t+i-1}, \tag{2.12}$$

where $s(t)$ is the result of the convolution at position t, $\boldsymbol{x} = [x_1, x_2, \ldots, x_n]$ is the input sequence vector of lenght n, $\boldsymbol{w} = [w_1, w_2, \ldots, w_n]$ is the convolutional filter (kernel) of size k. The filter $\boldsymbol{w}$ is slid across the input sequence $\boldsymbol{x}$, and at each position a dot product is computed between the filter and the corresponding local window of the input sequence. Here, the index t is an integer, representing the discrete position in the sequence at which the convolution is evaluated. This operation yields a feature map that elucidates the most salient patterns within the sequence. Subsequently, the feature map is subjected to a non-linear activation function. In addition to the convolutional layer, 1D CNNs frequently comprise a pooling layer, such as max pooling, with the objective of reducing the dimensionality of the feature maps. Max pooling involves the selection of the maximum value within a local region of the feature map, thereby enhancing the network's resilience to minor shifts in the input. The pooling operation for 1D CNNs is defined as follows:

$$y_t = \max\{s(t), s(t+1), \ldots, s(t+k-1)\}, \tag{2.13}$$

where y_t is the pooled output at position t, and $s(t)$ denotes the scalar convolution output at position t. The operation takes the maximum value over a local window of size k, starting at index t. By applying multiple layers of convolutions and pooling operations, 1D CNNs can learn increasingly complex hierarchical features from sequential data. 1D CNNs are a ubiquitous method in domains where data is inherently sequential. This enables the identification of underlying patterns, such as trends or seasonality, which can then be incorporated into a predictive model (Borovykh et al., 2019).

Two-dimensional convolutional neural networks: Two-dimensional (2D) CNNs are designed for the processing of spatial data, particularly images. In a 2D CNN, the input is represented as a two-dimensional grid of pixel values, and the convolution operation is applied across the entire image, including both the height and width. The objective of a 2D CNN is to facilitate the learning of spatial hierarchies. Subsequently, the data undergo a process of feature extraction, whereby characteristics such as edges, textures, and shapes are identified. The 2D convolution operation is defined as:

$$s(i,j) = (\boldsymbol{X} * \boldsymbol{W})(i,j) = \sum_{m=1}^{k_h} \sum_{n=1}^{k_w} \boldsymbol{W}(m,n) \cdot \boldsymbol{X}(i+m-1, j+n-1), \tag{2.14}$$

where $s(i,j)$ denotes the scalar output of the convolution at position (i,j). The input image is represented by the matrix $\boldsymbol{X} \in \mathbb{R}^{H \times W}$ with height H and width W, and the convolutional kernel is given by the matrix $\boldsymbol{W} \in \mathbb{R}^{k_h \times k_w}$, where k_h and k_w specify the height and width of the kernel. The output of the convolution is the matrix $\boldsymbol{S}$, whose entries are the values $s(i,j)$ obtained by sliding the kernel across the input image. As with 1D CNNs, the output of the convolutional layer is passed through a non-linear activation function. This introduces non-linearity, thereby enabling the network to model complex relationships in the data. A pooling layer, such as max pooling, is often utilized to reduce the spatial dimensions of the feature maps while preserving the most salient features. In a 2D CNN, max pooling is applied to reduce the spatial dimensions of the feature map while retaining its most salient information. The operation is performed over a local window of the feature map, typically of size 2×2 . At each pooling step, the maximum value within the window is selected. For a pooling region of size 2×2 , the operation can be written as

$$y(i,j) = \max\{s(i,j), s(i+1,j), s(i,j+1), si+1, j+1)\}, \tag{2.15}$$

where $y(i,j)$ denotes the scalar pooled output at position (i,j) and $s(i,j)$ denotes the scalar activation of the feature map at position (i,j). The stacking of multiple convolutional and pooling layers enables CNNs to learn increasingly complex and abstract features, which are subsequently employed for classification tasks through the use of fully connected layers.

2D CNNs have had a profound impact on the field of computer vision, becoming the foundation of numerous cutting-edge image processing models. In addition to image classification, CNNs are also widely employed in object detection, which involves identifying and localizing objects within an image. Methods such as Faster R-CNN employ CNNs for feature extraction in conjunction with region proposal networks to achieve real-time object detection (Ren et al., 2017). In the healthcare domain, CNNs are employed in medical imaging tasks, including the detection of tumors in magnetic resonance imaging or computed tomography scans, the identification of lesions in X-rays, and the segmentation of organs for surgical planning. The application of CNNs has been demonstrated to markedly enhance the precision and expediency of medical diagnoses (Litjens et al., 2017).

Siamese neural networks

Siamese neural networks (SNNs) represent a class of neural network architectures designed with the objective of comparing and measuring the similarity between two inputs. First introduced by (Bromley et al., 1993) for the purpose of signature verification, SNNs have since been widely

adopted across a number of different domains, including image recognition, natural language processing, and time series analysis. Their capacity to learn meaningful latent features for comparison tasks renders them highly efficacious for applications such as verification, matching, and clustering. A typical Siamese neural network is composed of two identical subnetworks, each of which processes one of the two input data points. The subnetworks are configured to share weights and parameters, thereby ensuring comparability of the feature representations. Subsequently, the aforementioned feature representations are compared using an appropriate similarity measure, such as the Euclidean distance. Given inputs $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, the subnetwork f produces latent features $\boldsymbol{z}_1 = f(\boldsymbol{x}_1; \Theta)$ and $\boldsymbol{z}_2 = f(\boldsymbol{x}_2; \Theta)$, with shared parameters Θ (see Fig. 2.7). Further applications of SNNs are shown in Appendix A.1.1.

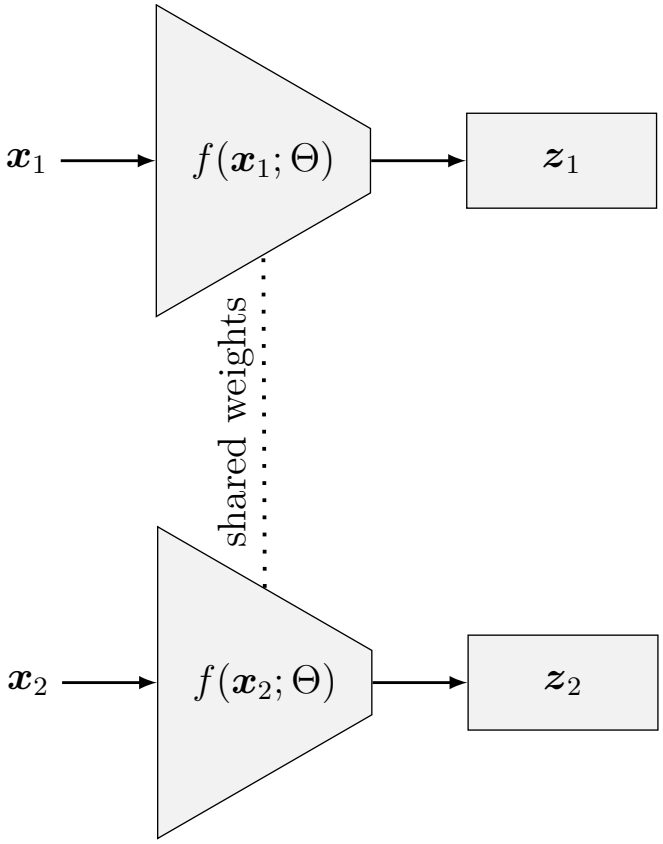

Figure 2.7: Visualization of siamese neural networks.

2.1.3 Dimensionality reduction

Dimensionality reduction enables the condensation of high-dimensional data into a format that is more readily comprehensible and interpretable. The objective is to transform data from a high-dimensional space into a lower-dimensional space while maintaining crucial attributes, such as variance or distances between data points. This section presents a motivation for dimensionality reduction and both linear and non-linear approaches. A general review of dimensionality reduction techniques can be found in (Maaten et al., 2009).

The motivation behind the pursuit of dimensionality reduction can be attributed to the multitude of challenges inherent to high-dimensional data. First, high-dimensional datasets frequently comprise redundant or irrelevant features, which can impair the efficacy of ML algo-

rithms. Secondly, as the dimensionality of the data set increases, the computational complexity grows exponentially, resulting in inefficiencies in both memory and processing time. This phenomenon is commonly referred to as the curse of dimensionality, a term first introduced by Richard Bellman (Bellman, 1957). It describes the exponential growth in computational complexity and data sparsity as the number of dimensions increases. In high-dimensional spaces, data points frequently exhibit sparsity and equidistance, rendering traditional distance-based algorithms (such as k-nearest neighbors) less effective. Consequently, learning algorithms may encounter difficulties in generalizing to new data, which may result in overfitting.

Mathematically, the curse of dimensionality can be expressed by examining the volume of an n-dimensional space. The volume V of an n-dimensional hypercube with side length r is given by the following equation:

$$V = r^n. \tag{2.16}$$

As the value of n increases, the volume of the space in question grows exponentially, while the density of the data points decreases. This sparsity can result in suboptimal generalization in ML models, as the number of data points required to sufficiently cover the space in question grows exponentially with the number of dimensions. Dimensionality reduction techniques are designed to address this challenge by reducing the number of dimensions while retaining the most informative features.

Linear dimensionality reduction techniques

One of the earliest techniques for reducing the dimensionality of data sets is principal component analysis (PCA), which was first introduced by (Pearson, 1901) and subsequently formalized by (Hotelling, 1933). PCA is a technique that transforms data with a high number of dimensions into a lower-dimensional space. This is accomplished by identifying orthogonal principal components that capture the maximum variability. The PCA technique comprises a series of steps, including mean centering, normalization (if necessary), computation of a covariance matrix, and computation of an eigenvalue decomposition. Subsequently, the components are ordered according to the proportion of variance they explain, and the data are projected onto those components that retain the desired proportion of variance. PCA is a prominent tool in the field of computer vision, with applications including the eigenfaces technique used in face recognition (Turk and Pentland, 1991). PCA is employed for the purpose of extracting principal components, or eigenfaces, from facial images. This process captures distinctive features while simultaneously reducing noise and redundancy. By projecting a face image onto the lower-dimensional eigenface space, it can be represented and compared in an efficient manner. This demonstrates the capacity of PCA to diminish the dimensionality of high-dimensional image data while preserving the essential information. The mathematical decomposition technique of singular value decomposition is discussed in Section A.1.2.

Non-linear dimensionality reduction techniques

Multidimensional scaling (MDS) is a technique that prioritizes the preservation of the pairwise distances between data points when projecting them into a lower-dimensional space. The objective of MDS is to minimize the differences between the original distances in the high-dimensional

space and the distances in the reduced space. Given a set of points with pairwise distances d_{ij} in the original space, MDS identifies a lower-dimensional configuration of points such that the distances d'_{ij} in the lower-dimensional space approximate the original distances. The objective function for MDS is known as Stress and is typically defined as follows:

$$\text{Stress} = \sqrt{\frac{\sum_{i<j}(d_{ij} - d'_{ij})^2}{\sum_{i<j} d_{ij}^2}}. \tag{2.17}$$

MDS is particularly useful for visualizing high-dimensional data, as it preserves the relative distances between points, thereby facilitating the interpretation of relationships between data points in a lower-dimensional space (Borg and Groenen, 2005).

Autoencoders are a specific type of artificial neural network that is employed for the purpose of non-linear dimensionality reduction. An autoencoder is comprised of two essential components: an encoder and a decoder (see Fig. 2.8). The encoder maps the input data to a lower-dimensional latent space, whereas the decoder is responsible for reconstructing the original data from the latent representation. The architecture of an autoencoder can be described as follows: Let $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ be the input data, where n is the number of samples and d is the number of features. The encoder function $f(\boldsymbol{x}; \Theta)$ maps the input $\boldsymbol{x}$ to a lower-dimensional latent space $\boldsymbol{z} = f(\boldsymbol{x}; \Theta)$, where Θ represents the parameters of the encoder. The decoder function $g(\boldsymbol{z}; \phi)$ reconstructs the input from the latent representation, producing $\hat{\boldsymbol{x}} = g(\boldsymbol{z}; \phi)$, where ϕ represents the parameters of the decoder. The objective of training an autoencoder is to minimize the reconstruction error, which is typically measured using the mean squared error (MSE):

$$\mathcal{J}_{\text{ae}}(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \frac{1}{n} \sum_{i=1}^{n} (x_i - \hat{x}_i)^2. \tag{2.18}$$

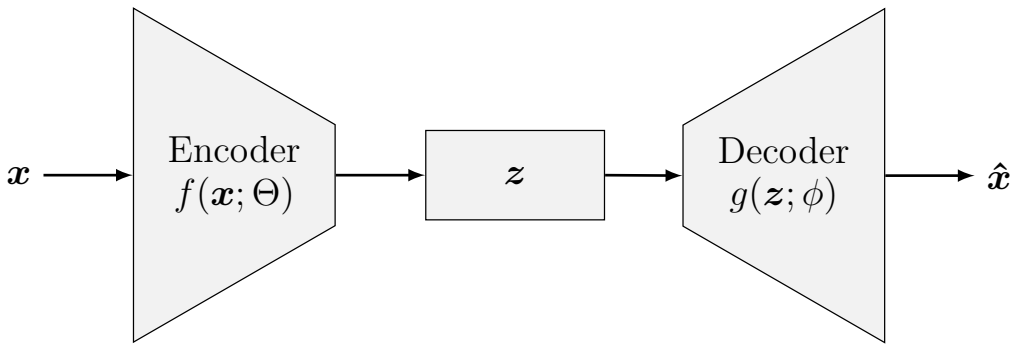

Figure 2.8: Representation of an autoencoder with the encoder $f(\boldsymbol{x})$ and the decoder $g(\boldsymbol{z})$.

Autoencoders are particularly useful for compressing high-dimensional data into a lower-dimensional representation, which can then be used for tasks such as anomaly detection, data visualization, and feature extraction. Autoencoders have been successfully applied in a number of fields, including image processing, where they are utilized for tasks such as image denoising and dimensionality reduction in deep learning models (Hinton and Salakhutdinov, 2006). Further approaches on non-linear dimensionality reduction techniques are discussed in Section A.1.2

2.1.4 Gradient-free optimization

Gradient-free optimization methods are particularly well-suited for addressing problems where gradient information is unavailable, unreliable, or computationally expensive to compute. These techniques are particularly advantageous when applied to objective functions that are non-differentiable, noisy, or discontinuous. Among the various approaches, population-based optimization methods are notable for their capacity to navigate high-dimensional and multimodal search spaces. These methods operate by maintaining a population of candidate solutions that evolve iteratively through mechanisms inspired by biological or social behaviors. Three prominent population-based techniques are differential evolution, genetic algorithms, and particle swarm optimization. Each of these techniques employs a distinct strategy for optimizing complex objective functions. The technique of differential evolution will be discussed in the following, whereas the techniques of genetic algorithms and particle swarm optimization are discussed in Appendix A.1.5.

As outlined by (Storn and Price, 1997), differential evolution (DE) represents a sophisticated and resilient population-based optimization technique, particularly adept at addressing complex, nonlinear, and multimodal objective functions. The method operates by maintaining a population of candidate solutions and iteratively improving them through a series of steps, including initialization, mutation, crossover, and selection (see Fig. 2.9).

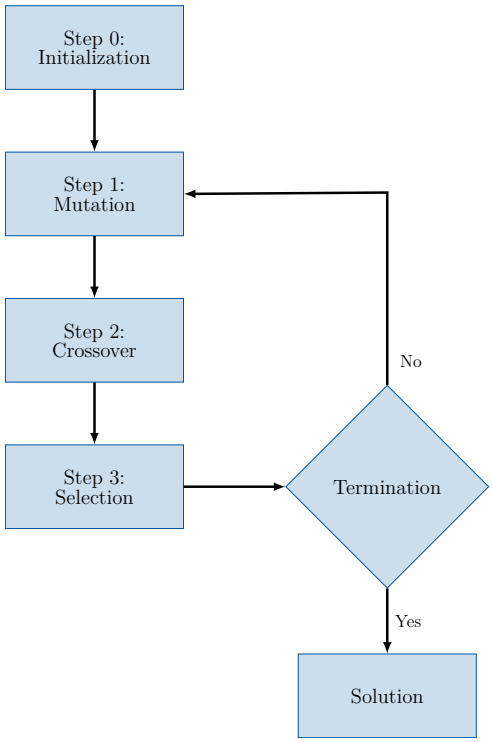

Figure 2.9: Schematic illustration of the workflow of the differential evolution algorithm.

The initialization phase (step 0) of the algorithm begins by initializing a population P of candidate solutions. Let the population size be denoted by N_p and the dimensionality of the optimization problem by D. Each individual $\boldsymbol{x}_i$ in the population corresponds to a candidate solution represented as a D-dimensional real-valued vector: $\boldsymbol{x}_i = [x_{i,1}, x_{i,2}, \ldots, x_{i,D}]$ where $i = 1, 2, \ldots, N_p$ and $x_{i,j}$ denotes the value of the j-th variable of the i-th individual. For each

dimension j the search space is bounded by a lower bound lower_j and an upper bound upper_j.

The initialization process assigns each component $x_{i,j}$ a value drawn uniformly at random from its respective interval $[\text{lower}_j, \text{upper}_j]$. This is expressed as:

$$x_{i,j} = \text{lower}_j + r \cdot (\text{upper}_j - \text{lower}_j), \tag{2.19}$$

where $r \sim U(0,1)$ is a uniformly distributed random variable. This guarantees that the initial population is diverse and effectively spans the search space.

Subsequent to initialization, the algorithm proceeds to the mutation step (step 1), wherein a mutant vector $\boldsymbol{v}_i$ is generated for each individual $\boldsymbol{x}_i$ by leveraging the disparities between randomly selected population members. Specifically, three distinct individuals, denoted as $\boldsymbol{x}_{r1}$, $\boldsymbol{x}_{r2}$, and $\boldsymbol{x}_{r3}$, are selected such that $r1 \neq r2 \neq r3 \neq i$. The mutant vector is then computed using the following equation:

$$\boldsymbol{v}_i = \boldsymbol{x}_{r1} + F \cdot (\boldsymbol{x}_{r2} - \boldsymbol{x}_{r3}), \tag{2.20}$$

where F represents the scaling factor that governs the magnitude of the difference vector $(\boldsymbol{x}_{r2} - \boldsymbol{x}_{r3})$. This mutation strategy introduces variation into the population while preserving information from existing solutions and is known as 'DE/rand/1'. Further strategies exist like 'DE/rand/2': $\boldsymbol{v}_i = \boldsymbol{x}_{r1} + F \cdot (\boldsymbol{x}_{r2} - \boldsymbol{x}_{r3} + \boldsymbol{x}_{r4} - \boldsymbol{x}_{r5})$, which takes five distinct individuals, and 'DE/best/1': $\boldsymbol{v}_i = \boldsymbol{x}_{r\text{best}} + F \cdot (\boldsymbol{x}_{r2} - \boldsymbol{x}_{r3})$, which takes the best individual as the first individual.

Following the completion of the mutation step, the algorithm applies crossover (step 2). The objective of this is to enhance diversity by assimilating information from both the mutant vector $\boldsymbol{v}_i$ and the original individual $\boldsymbol{x}_i$. The trial vector $\boldsymbol{u}_i$ is derived using binomial crossover, wherein each component j is assigned according to:

$$\begin{cases} v_{i,j} & \text{if } r_j \leq \text{CR or } j = j_{\text{rand}}, \\ x_{i,j} & \text{otherwise.} \end{cases} \tag{2.21}$$

In this context, CR denotes the crossover probability, which is typically selected within the range [0.1, 0.9], and $r_j \sim U(0,1)$ represents a random number. Furthermore, the index j guarantees that at least one component of the trial vector is derived from the mutant vector. Consequently, this prevents the trial vector from being identical to the original individual.

Subsequent to the crossover step, the selection step (step 3) determines whether the newly generated trial vector $\boldsymbol{u}_i$ replaces its corresponding target vector $\boldsymbol{x}_i$. This determination is made according to a greedy selection criterion, wherein the vector with the lower objective function $f(.)$ value is retained in the subsequent generation. Mathematically, this can be expressed as follows:

$$\boldsymbol{x}_i^* = \begin{cases} \boldsymbol{u}_i & \text{if } f(\boldsymbol{u}_i) < f(\boldsymbol{x}_i), \\ \boldsymbol{x}_i & \text{otherwise.} \end{cases} \tag{2.22}$$

This process is intended to guarantee that the population progresses towards enhanced solutions across successive generations. The iterative process of mutation, crossover, and selection is continued until a predefined stopping criterion is met. Common termination conditions include reaching a maximum number of generations, achieving convergence within a specified tolerance, or observing stagnation in the optimization process.

A variety of enhancements have been proposed to address the limitations of DE in terms of convergence speed, adaptability, and robustness in complex optimization problems. One such advancement is JADE (a self-adaptive variant of DE) introduced by (Zhang and Sanderson, 2009). JADE overcomes the limitations of traditional DE by incorporating two key modifications: an adaptive control mechanism for the scaling factor F and crossover probability CR, and a more effective mutation strategy based on an external archive. In contrast to the static or heuristically chosen parameters in conventional DE, JADE dynamically updates F and CR throughout the evolutionary process, allowing the algorithm to balance exploration and exploitation more efficiently. Moreover, the mutation strategy in JADE employs a differential mutation scheme referred to as 'DE/current-to-pbest/1' where the base vector is not randomly selected but is instead drawn from the best-performing individuals in the current population. This strategy, when coupled with an external archive that stores previously discarded solutions, enhances diversity and prevents premature convergence. By integrating these self-adaptive mechanisms, JADE has demonstrated improved convergence speed and robustness across a wide range of optimization problems, making it a more reliable choice for high-dimensional and multimodal landscapes compared to classical DE.

2.2 Approach for the learning of an universal low dimensional latent feature space

General architecture

The method for the determination of the latent feature space corresponds to the application of a ML methodology called multitask learning (MTL) (Caruana, 1997). MTL is a method to train a single, composed neural network for the simultaneous solution of different tasks. The MTL processing scheme must consist at least of (a) an encoder network, which extracts common features and transforms the high-dimensional input feature space to a lower dimensional feature space (latent features (LF)), and (b) subsequent task processing networks, each for a different task. MTL is pursuing the approach of training all tasks in parallel with each other at the same time to allow a solution structure for one task to use the information of the solution structures for the other tasks. The tasks are of one of the following categories: (1) classification in the case of conditions and outcomes of discrete values, (2) estimation in the case of conditions and outcomes of continuous values, and (3) the reconstruction (decoding) of the input data to enable the feature extraction via auto-encoding (see Fig. 2.10).

The layers above the latent feature space have common structures which are reused. The activations of the latent feature neurons are the feature values representing the input, resulting from the encoding by the preceding layers and are then used as input to the successive tasks. The layers below the latent features are attached to the encoder network and represent separate specializations for the different tasks. The training of the MTL network yields a lower dimensional representation space with features, which are sensitive to the learned tasks. Choosing a low-dimensional feature space forces the generalization of the features and additionally lends itself to visualization. The proposed MTL scheme of Fig. 2.10 allows to reveal and describe the influence of conditions and outcomes on the input data by means of feature space analysis.

In the context of multitask learning, the approach of hard parameter sharing has been identified as a fundamental technique. In this method, the model's parameters are shared across

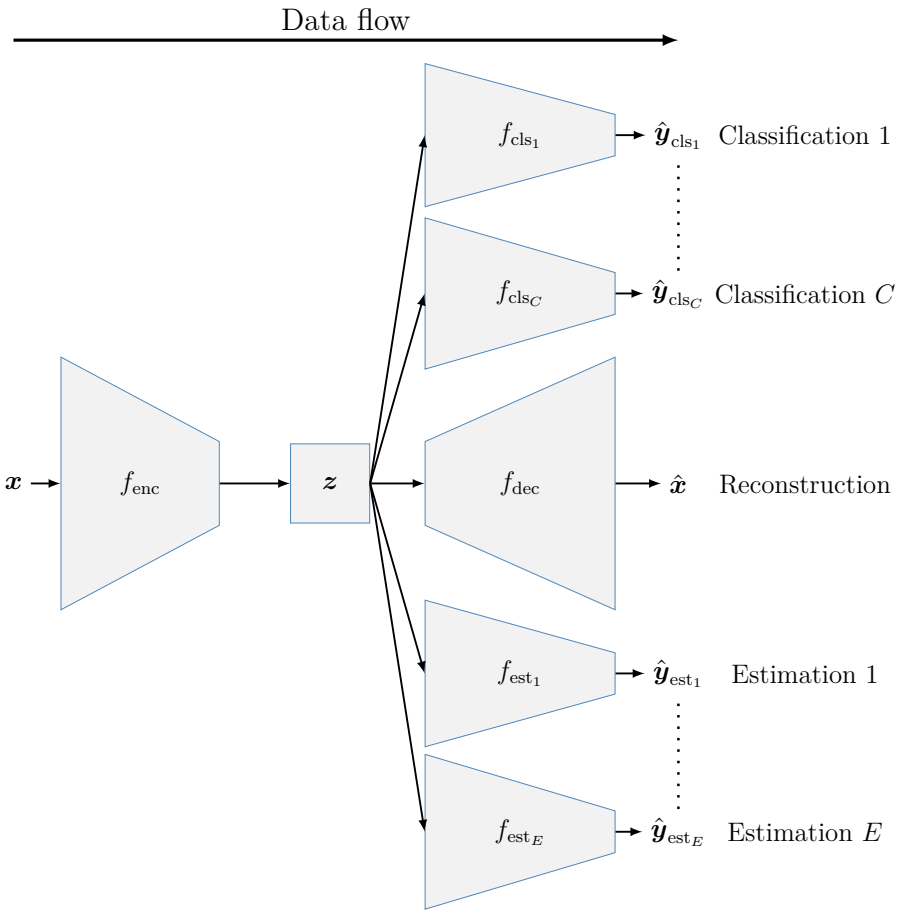

Figure 2.10: General multitask learning architecture with the three categories of tasks: (1) C classification tasks, (2) E estimation tasks, and (3) reconstruction.

multiple tasks, typically in the lower layers of a neural network (Ruder, 2017). The efficacy of this approach has been demonstrated in enhancing the model's generalization capability by facilitating the acquisition of shared representations that encapsulate the commonalities among tasks. This effect functions as a regularizer, thereby mitigating the risk of overfitting.The utilization of shared knowledge across tasks, facilitated by hard parameter sharing, results in enhanced computational efficiency, reduced memory requirements, and particular advantages for tasks with limited data or closely related objectives.

Multitask learning objective function

In MTL we have the following mapping function

$$f : \mathcal{X} \to \mathcal{Y}, \tag{2.23}$$

where $\mathcal{X}$ is the input feature space and

$$\mathcal{Y} = \prod_{t \in \mathcal{T}} \mathcal{Y}_t. \tag{2.24}$$

is the output space, as a product of distinct output spaces from a given set of tasks $\mathcal{T} = \{t_1, t_2, \ldots, t_k\}$. We assume that each task t can be either a classification task $\mathcal{T}_{\text{cls}}$ or an estimation

task $\mathcal{T}_{\text{est}}$, therefore $\mathcal{T} = \mathcal{T}_{\text{cls}} \cup \mathcal{T}_{\text{est}}$ and $\mathcal{T}_{\text{cls}} \cap \mathcal{T}_{\text{est}} = \varnothing$. For example in an MTL setting with two classification tasks and one estimation task, we have $\mathcal{T} = \{\text{cls}_1, \text{cls}_2, \text{est}_1\}$.

For the t-th classification task cls_t we assume $\mathcal{Y}_{\text{cls}_t} \subseteq \mathbb{Z}^K$, where K is the number of target classes, and for the t-th estimation task est_t, we assume $\mathcal{Y}_{\text{est}_t} \subseteq \mathbb{R}^P$, where P is the dimensionality of the continuous output. Furthermore, we suppose there is an underlying probability distribution defined over the input space $\mathcal{X}$ denoted by $p(\mathcal{X})$, and a joint conditional probability distribution defined over $\mathcal{Y}$, indicated by $p(\mathcal{Y}|\mathcal{X})$. Hence an MTL dataset $\mathcal{D}$ is defined as:

$$\mathcal{D} = \left\{ (\boldsymbol{x}_n, \boldsymbol{y}_n) \right\}_{n=1}^{N}, \tag{2.25}$$

where $\boldsymbol{x}_n \sim p(\mathcal{X}) \in \mathbb{R}^D$ represents the input features for the n-th sample with $n = 1, 2, \ldots, N$ and Dimension D , and $\boldsymbol{y}_n \sim p(\mathcal{Y}|\mathcal{X} = \boldsymbol{x}_n)$ is the ground-truth for the n-th sample. Consequently, depending on the MTL setting, the structure of $\boldsymbol{y}_n$ can change. For example for the previous example setting with two classification tasks and one estimation task, we have $\boldsymbol{y}_n = [\boldsymbol{y}_{n,\text{cls}_1}, \boldsymbol{y}_{n,\text{cls}_2}, \boldsymbol{y}_{n,\text{est}_1}]$.

The proposed approach shown in Fig. 2.10 automatically derives low-dimensional latent features, which allow the optimal reconstruction of the input data and also capture maximum information about the corresponding tasks. The latent features can therefore represent the effects of different influences of the input data.

The encoder transformation of the data into the low-dimensional latent feature space is learned via MTL together with the involved models of the decoder, classifier and estimation tasks, which process the data in the latent feature space. Each of the neural network mapping functions of Fig. 2.10 is parametrized by its weight values, which forms the parameter vector Θ.

The input features $\boldsymbol{x}_n \in \mathbb{R}^D$ are transformed by the encoder network into a low-dimensional latent feature representation $\boldsymbol{z}_n \in \mathbb{R}^Z$ according to the learned function $f_{\text{enc}} : \mathbb{R}^D \to \mathbb{R}^Z$

$$\boldsymbol{z}_n = f_{\text{enc}}(\boldsymbol{x}_n; \Theta_{\text{enc}}), \tag{2.26}$$

in which the encoder network is parameterized by its weight values Θ_{enc}. The MTL model consists of $C + E + 1$ networks:

1. The decoder network reconstructs the input features $\boldsymbol{x}_n$ based on the latent feature representation $\boldsymbol{z}_n$ back into approximations $\hat{\boldsymbol{x}}_n \in \mathbb{R}^D$ of the original input features according to the learned function $f_{\text{dec}} : \mathbb{R}^Z \to \mathbb{R}^D$

$$\hat{\boldsymbol{x}}_n = f_{\text{dec}}(\boldsymbol{z}_n; \Theta_{\text{dec}}) = f_{\text{dec}}(f_{\text{enc}}(\boldsymbol{x}_n, \Theta_{\text{enc}}), \Theta_{\text{dec}}), \tag{2.27}$$

   where the decoder network is parameterized by its weight values Θ_{dec}.

2. There are C classification networks each responsible for learning to predict class labels from the latent representation $\boldsymbol{z}_n$ for the t-th classification task according to the learned function $f_{\text{cls}_t} : \mathbb{R}^Z \to \mathbb{Z}^K$

$$\hat{\boldsymbol{y}}_{n,\text{cls}_t} = f_{\text{cls}_t}(\boldsymbol{z}_n; \Theta_{\text{cls}_t}) = f_{\text{cls}_t}(f_{\text{enc}}(\boldsymbol{x}_n, \Theta_{\text{enc}}), \Theta_{\text{cls}_t}), \tag{2.28}$$

   where the t-th classifier network is parameterized by its weight values Θ_{cls_t}.

3. There are E estimation networks each responsible for learning to predict real-valued outputs from the latent representation $\boldsymbol{z}_n$ for the t-th estimation task according to the learned function $f_{\text{est}_t} : \mathbb{R}^Z \rightarrow \mathbb{R}^P$

$$\hat{\boldsymbol{y}}_{n,\text{est}_t} = f_{\text{est}_t}(\boldsymbol{z}_n; \Theta_{\text{est}_t}) = f_{\text{est}_t}(f_{\text{enc}}(\boldsymbol{x}_n, \Theta_{\text{enc}}), \Theta_{\text{est}_t}), \tag{2.29}$$

where the t-th estimation network is parameterized by its weight values Θ_{est_t}.

The parameters of all networks are simultaneously optimized, which means, that for the combined parameter vector $\Theta = [\Theta_{\text{enc}}, \Theta_{\text{dec}}, \Theta_{\text{cls}_t}, \Theta_{\text{est}_t}]^\top$ the optimum Θ^* is seeked for all N input data samples $\boldsymbol{x}_n$ to deliver the corresponding $\boldsymbol{y}_{n,\text{cls}_t}$ and $\boldsymbol{y}_{n,\text{est}_t}$ values and to reproduce $\boldsymbol{x}_n$ as good as possible, which is expressed by the search for the minimum of a loss function $\mathcal{J}$ for the data set $\mathcal{D}$:

$$\begin{aligned} \Theta^* &= \underset{\Theta}{\text{argmin}}\, \mathcal{J}(\Theta) \\ \mathcal{J}(\Theta) &= \underset{\Theta}{\text{argmin}} \sum_{n=1}^{N} \mathcal{J}_n(\Theta) \end{aligned} \tag{2.30}$$

The loss function is composed of the weighted loss terms of all tasks, where λ_{dec} is the weight of the decoder loss, λ_{cls_t} is the weight of the t-th classification loss and λ_{est_t} is the weight of the t-th estimation loss:

$$\mathcal{J}_n = \lambda_{\text{dec}} \cdot \mathcal{J}_{n,\text{dec}} + \lambda_{\text{cls}_t} \cdot \mathcal{J}_{n,\text{cls}_t} + \lambda_{\text{est}_t} \cdot \mathcal{J}_{n,\text{est}_t} \tag{2.31}$$

The reconstruction loss of the decoder network for the n-th sample is the MSE between the input features $\boldsymbol{x}_n$ and the reconstructions $\hat{\boldsymbol{x}}_n$:

$$\mathcal{J}_{n,\text{dec}}(\boldsymbol{x}_n, \hat{\boldsymbol{x}}_n) = \frac{1}{D} \sum_{i=1}^{D} (\boldsymbol{x}_{n_i} - \hat{\boldsymbol{x}}_{n_i})^2 \tag{2.32}$$

The t-th classification loss for the n-th sample is the difference between the ground truth classes $\boldsymbol{y}_{n,\text{cls}_t}$ and the predicted classes $\hat{\boldsymbol{y}}_{n,\text{cls}_t}$:

$$\mathcal{J}_{n,\text{cls}_t}(\boldsymbol{y}_{n,\text{cls}_t}, \hat{\boldsymbol{y}}_{n,\text{cls}_t}) = (\boldsymbol{y}_{n,\text{cls}_t} - \hat{\boldsymbol{y}}_{n,\text{cls}_t})^2 \tag{2.33}$$

The t-th estimation loss for the n-th sample is the MSE between the ground truth continuous values $\boldsymbol{y}_{n,\text{est}_t}$ and the predicted continuous values $\hat{\boldsymbol{y}}_{n,\text{est}_t}$:

$$\mathcal{J}_{n,\text{est}_t}(\boldsymbol{y}_{n,\text{est}_t}, \hat{\boldsymbol{y}}_{n,\text{est}_t}) = \frac{1}{P} \sum_{i=1}^{P} (\boldsymbol{y}_{n,\text{est}_{t_i}}, \hat{\boldsymbol{y}}_{n,\text{est}_{t_i}})^2 \tag{2.34}$$

An optimizer is seeking the optimal set of all parameters Θ^*:

$$\begin{aligned} [\Theta^*_{\text{enc}}, \Theta^*_{\text{dec}}, \Theta^*_{\text{cls}_t} \Theta^*_{\text{est}_t}] = \underset{\Theta}{\text{argmin}} \sum_{n=1}^{N} \\ \lambda_{\text{dec}} \Big(\boldsymbol{x}_n - f_{\text{dec}}(f_{\text{enc}}(\boldsymbol{x}_n, \Theta_{\text{enc}}), \Theta_{\text{dec}}) \Big)^2 \\ + \lambda_{\text{cls}_t} \Big(\boldsymbol{y}_{n,\text{cls}_t} - f_{\text{cls}_t}(f_{\text{enc}}(\boldsymbol{x}_n, \Theta_{\text{enc}}), \Theta_{\text{cls}_t}) \Big)^2 \\ + \lambda_{\text{est}_t} \Big(\boldsymbol{y}_{n,\text{est}_t} - f_{\text{est}_t}(f_{\text{enc}}(\boldsymbol{x}_n, \Theta_{\text{enc}}), \Theta_{\text{est}_t}) \Big)^2 \end{aligned} \tag{2.35}$$

Criteria for meaningful latent spaces in multitask learning

A good and meaningful latent space is characterized by several important properties. It should capture the underlying structure of the data in a compact yet expressive form, such that relevant features for all tasks are preserved. The latent space should also preserve essential relationships between data points, for instance by maintaining similarity structures so that samples that are close in the input space remain close in the latent space. This property ensures that the latent space enables task-specific learning while supporting joint learning across tasks within a shared representation. The quality of a latent space, however, is not universal but depends on the context: it must be specifically adapted to the dataset and the tasks under consideration. For example, a latent space that organizes features such as shape and line thickness in handwritten digit images may be highly effective for digit classification and stroke thickness prediction, but it would not necessarily transfer to tasks involving medical images. Finally, the practical usefulness of a latent space can be evaluated by the learnability of the tasks: if multiple tasks can be solved with sufficient accuracy based on the shared representation, this provides confirmation that the latent space is well-structured and meaningful. The importance of such structured representations is emphasized by (Standley et al., 2020), who demonstrate that task compatibility is directly linked to the quality of the shared latent representation and that poor alignment can hinder performance across tasks.

In contrast, a bad latent space lacks these properties. It may fail to encode the essential task-relevant features, leading to poor task performance or biased generalization. Similarly, if the latent space collapses important variations in the data, for example by mapping two distinct points in the input space to the same position in the latent space, task-relevant distinctions are lost. Such collapse reduces the representation's capacity to separate meaningful features, leading to ambiguity and low performance. Likewise, if the latent space introduces distortions that mask similarities, it will undermine the ability to learn knowledge across tasks. In such cases, even if the model appears to learn, the latent space contributes little to effective knowledge sharing across tasks, thereby reducing its value for MTL.

2.3 Introduction to the most important materials aspects relevant to this work

This section introduces fundamental concepts in materials science that are essential for understanding the microstructural and mechanical behavior of materials in manufacturing processes. The section begins with an examination of crystallographic texture, with a particular focus on the characterization of grain orientations through the use of the orientation distribution function, generalized spherical harmonic functions, orientation histograms, and pole figures. These methods are fundamental to the analysis and representation of anisotropy in material properties. The discussion then proceeds to the Taylor-type material model, which provides a theoretical framework for establishing a link between microstructural features and mechanical behavior. In conclusion, this subchapter provides an overview of resistance spot welding, a joining technique in industrial manufacturing.

2.3.1 The crystallographic texture

In polycrystalline materials, the fundamental building blocks of the microstructure are grains, which are regions of crystalline order separated by grain boundaries. Each grain is characterized by a distinct crystallographic orientation, and together, the distribution of these orientations across all grains defines the material's crystallographic texture. Grains can vary significantly in size, shape, and orientation, and these variations strongly influence macroscopic material properties such as strength, ductility, hardness, and anisotropy. For example, the collective arrangement of orientations gives rise to preferred directions of deformation (Kocks et al., 1998).

The importance of grains and textures is underscored by a large body of literature spanning metallurgy, ceramics, and functional materials (Dieter, 1986; Cullity and Stock, 2001; Kocks et al., 1998; Randle and Engler, 2009). These works demonstrate how microstructural features rooted in grains control performance across applications ranging from structural alloys to metal forming processes. In the latter, the evolution of grain structure and crystallographic texture governs anisotropy, drawability, and overall formability (Wenk and Van Houtte, 2004; Ray et al., 1994). In this context, grains and their textures emerge as central factors of structure–property relationships in materials science. This role has been further reinforced by the development of advanced methods for measuring and quantifying texture, which provide comprehensive methods to characterize orientation distributions at multiple length scales.

Orientation distribution function

Grain orientations can be parameterized in different ways, see (Hansen et al., 1978). Most frequently used in experimentation are Euler angles (Bunge, 1982; Pospiech, 1972), which describe an orientation as three subsequently executed rotations around defined axes of a Cartesian coordinate system. However, due to singularities, the Euler space is distorted, in general. To overcome this issue, the axis-angle description (Hansen et al., 1978) or Rodrigues vectors (Frank, 1988) can be used, which describe orientations as a rotation around a 3D vector (the length of the vector yields the rotation). A general drawback of these descriptions, however: for doing calculus, signs and special conventions have to be respected and the amount of permutations that follows from it is relatively high. For this reason, unit quaternions, which are unique and well-defined, have been introduced for the description of orientations (Morawiec and Pospiech, 1989).

The crystallographic texture of materials is typically described by the orientation distribution function (ODF), which is defined by

$$f(g)\mathrm{d}g = \frac{\mathrm{d}V}{V}, \tag{2.36}$$

for an orientation g (a point in $SO(3)$) and the volume $V(g)$ in $SO(3)$. The ODF $f(g)$ often underlies specific symmetry conditions, for which various regions in $SO(3)$ are equivalent. Although this work is not limited by specific symmetry conditions, cubic symmetry conditions are assumed, as the focus is particularly on steel materials. Therefore, depending on the symmetries, orientations can be mapped into an elementary region of $SO(3)$, the so-called fundamental zone, see (Hansen et al., 1978). The ODF on the basis of the orientations mapped into the fundamental zone is indistinguishable from the original ODF.

For expressing an orientation g, several representations exist (cf. (Hansen et al., 1978)). For this work, however, two typically used representations are of importance, which are the Euler

angle representation following (Bunge, 1982) and the representation using unit quaternions, see for example (Morawiec and Pospiech, 1989; Frank, 1988). The Euler angles $\varphi_1, \phi, \varphi_2$ define subsequent rotations around the axes X, Z, X of a Cartesian coordinate system, while quaternions q_0, q_i, q_j, q_k form a vector in a four-dimensional vector space (Morawiec and Pospiech, 1989). In the following, commonly used texture descriptions are presented.

Generalized spherical harmonic functions

The ODF $f(g)$ can be approximated by a series expansion on the basis of generalized spherical harmonics (GSHs) (Bunge, 1982)

$$f_{\mathrm{gsh}}(g) = \sum_{l=0}^{\infty} \sum_{m=-l}^{+l} \sum_{n=-l}^{+l} C_l^{mn} T_l^{mn}(g), \tag{2.37}$$

with the orthonormal basis functions

$$T_l^{mn}(\varphi_1, \phi, \varphi_2) = e^{im\varphi_2} P_l^{mn}(\cos(\phi)) e^{in\varphi_1}, \tag{2.38}$$

in which g is expressed as Euler angles and P_l^{mn} denotes generalizations of the associated Legendre functions. The coefficients of the series expansion C_l^{mn} are often used as descriptors for crystallographic texture, as they encode the shape of the ODF, see for example (Kalidindi et al., 2004; Lyon and Adams, 2004; Proust and Kalidindi, 2006). In these works, however, symmetries of the ODF are taken into account, yielding a more compact form of Eq. (2.37) as certain functions $T_l^{mn}(g)$ can be transformed or eliminated, depending on the symmetry (Bunge, 1982). In this work, GSHs are generated using the software PyMKS (Brough et al., 2017).

Orientation histograms

Orientation histograms have been introduced in (Dornheim et al., 2022) to facilitate formulating statistical distance measures. The basis of the approach is the quaternion distance $\Phi(q_1, q_2)$ between two unit quaternions q_1 and q_2 as given in (Huynh, 2009):

$$\Phi(q_1, q_2) = \min(|q_1 - q_2|, |q_1 + q_2|). \tag{2.39}$$

To construct an orientation histogram, a set O of j nearly uniformly distributed orientations o_j in $SO(3)$ (or the fundamental zone, if symmetry conditions hold) is needed. The set O forms the bin centers of the orientation histograms and is derived using the algorithm introduced in (Quey et al., 2018), as is implemented in the software *neper* (Quey et al., 2011).

For a set of orientations G that describes the crystallographic texture of a material, assignment vectors $\boldsymbol{w}_g$ are calculated for each element $g \in G$ by

$$w_{\mathrm{g}}^{(j)} = \begin{cases} \frac{\Phi(g, o_j)}{\sum_{o_i \in N_l} \Phi(g, o_i)}, & \text{if } \; o_j \in N_l, \\ 0, & \text{else}, \end{cases} \tag{2.40}$$

with $w_{\mathrm{g}}^{(j)}$ denoting the j-th component of $\boldsymbol{w}_g$. Furthermore, N_l is a set of l nearest neighbor orientations of g in terms of the orientation distance Eq. (2.39). This so-called soft-assignment approach (with soft assignment factor l) guarantees that orientations that do not lie perfectly in the bin center are assigned proportionally to the neighbor bins $o_i \in N_l$. Finally, a volume

average over the assignment vectors of the individual orientations $\boldsymbol{w}_g$ yields the orientation histogram

$$f_{\mathrm{o}}(g) = \frac{1}{V} \sum_{g \in G} V(g) \boldsymbol{w}_g. \tag{2.41}$$

An example of an orientation histogram is given in Fig. 2.11. On the basis of this construction, the distance between two orientation histograms $f_{\mathrm{o}}^{(1)}(g)$ and $f_{\mathrm{o}}^{(2)}(g)$ can be measured via any kind of histogram-based distance measure, such as the Chi-Squared distance (Pele and Werman, 2010):

$$\mathscr{D}_{\chi^2}(f_{\mathrm{o}}^{(1)}(g), f_{\mathrm{o}}^{(2)}(g))) = \sum_{j=1}^{J} \frac{(f_{\mathrm{o}_j}^{(1)}(g) - f_{\mathrm{o}_j}^{(2)}(g))^2}{f_{\mathrm{o}_j}^{(1)}(g) + f_{\mathrm{o}_j}^{(2)}(g)}. \tag{2.42}$$

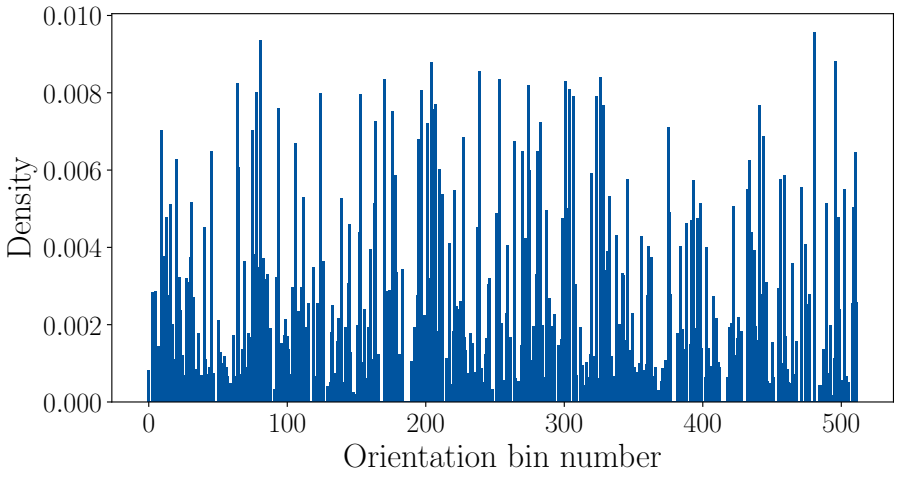

Figure 2.11: Example of an orientation histogram with 512 bins.

Pole figures

A pole figure is a graphical representation used in materials science and crystallography to display the orientation of polycrystals. It is based on a stereographic projection that maps three-dimensional information on a two-dimensional plane (Kocks et al., 1998). When using pole figures, the neighborhood relations of the orientations are represented by the neighboring pixels in the image. In this work, pole figures are generated using the software *mtex* (Bachmann et al., 2010). An example of a pole figure is given in Fig. 2.12.

2.3.2 Taylor-type material model

In this work, a Taylor-type material model is adopted to describe the mechanical response of polycrystals. The formulation relies on the deformation gradient, decomposed into elastic and plastic parts, with plastic flow governed by the plastic velocity gradient expressed through shear rates on crystallographic slip systems. The Cauchy stress tensor is computed using the fourth-order elastic stiffness tensor, linking anisotropic elasticity with plastic deformation. Together, these operators provide a consistent description of the interaction between elasticity and plasticity at the crystal level. The basic form of the Taylor-type material model used in this work

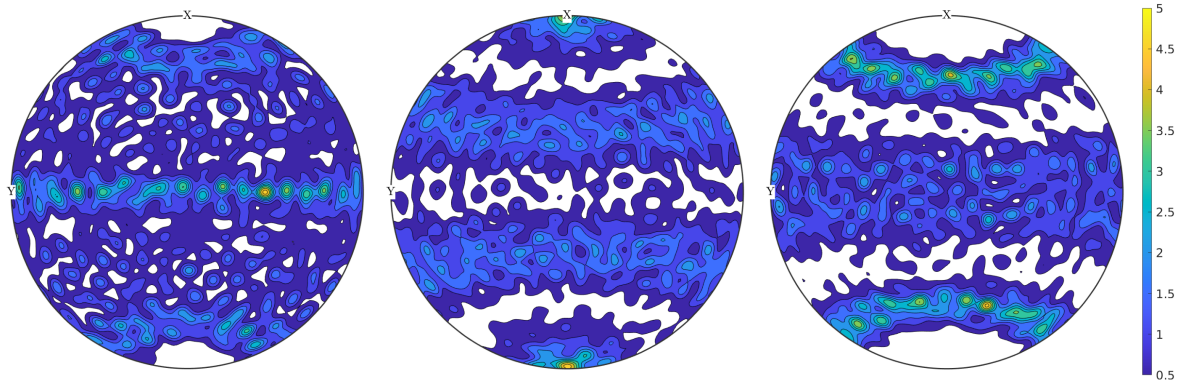

Figure 2.12: Example of an pole figure for the Miller's indices (001) (left), (110) (center) and (111) (right).

originates from the description in (Kalidindi et al., 1992). The operators $\cdot$ and : denote dot and the double dot product.

For n crystals, the volume averaged stress is defined by

$$\overline{\boldsymbol{T}} = \frac{1}{V} \sum_{i=1}^{n} \boldsymbol{T}^{(i)} V^{(i)}, \tag{2.43}$$

with $\boldsymbol{T}^{(i)}$ being the Cauchy stress tensor of a single crystal i with volume $V^{(i)}$. The Cauchy stress tensor $\boldsymbol{T}$ is derived from the stress tensor in the intermediate configuration using a multiplicative decomposition of the deformation gradient in its elastic and plastic part ($\boldsymbol{F} = \boldsymbol{F}_\mathrm{e} \cdot \boldsymbol{F}_\mathrm{p}$):

$$\boldsymbol{T}^* = \frac{1}{2}\, \mathbb{C} : (\boldsymbol{F}_\mathrm{e}^\top \cdot \boldsymbol{F}_\mathrm{e} - \boldsymbol{I}). \tag{2.44}$$

Here, $\boldsymbol{I}$ denotes the second order identity tensor and $\mathbb{C}$ denotes the forth order elastic stiffness tensor. When applying to cubic crystal symmetry, $\mathbb{C}$ is composed of three independent parameters, which are C_{11}, C_{12}, and C_{44}. To transform $\boldsymbol{T}$ into $\boldsymbol{T}^*$, the following relation is used:

$$\boldsymbol{T}^* = \boldsymbol{F}_\mathrm{e}^{-1} \cdot (\det(\boldsymbol{F}_\mathrm{e})\ \boldsymbol{T}) \cdot \boldsymbol{F}_\mathrm{e}^{-\top}. \tag{2.45}$$

The operator $\det(\boldsymbol{F}_\mathrm{e})$ takes the determinant of the second order tensor $\boldsymbol{F}_\mathrm{e}$. The plastic velocity gradient $\boldsymbol{L}_\mathrm{p}$, which is used to described the evolution of the plastic deformation, is defined as the sum of the shear rates $\dot{\gamma}^{(\eta)}$ on the slip systems η (Rice, 1971)

$$\boldsymbol{L}_\mathrm{p} = \dot{\boldsymbol{F}}_\mathrm{p} \cdot \boldsymbol{F}_\mathrm{p}^{-1} = \sum_{\eta} \dot{\gamma}^{(\eta)} \boldsymbol{m}^{(\eta)} \otimes \boldsymbol{n}^{(\eta)}. \tag{2.46}$$

The evolution over time of the plastic deformation is denoted as $\dot{\boldsymbol{F}}_\mathrm{p}$, where this derivative describes the rate of plastic deformation as a second order tensor. The slip systems are defined by the slip plane normal $\boldsymbol{n}^{(\eta)}$ and the slip direction $\boldsymbol{m}^{(\eta)}$. The tensor product of two the vectors $\boldsymbol{n}^{(\eta)}$ and $\boldsymbol{m}^{(\eta)}$ is denoted by $\boldsymbol{n}^{(\eta)} \otimes \boldsymbol{m}^{(\eta)}$. For the purpose of this work, the slip system families for body centered cubic (bcc) crystals are used ($\{110\}\langle 111\rangle$ and $\{112\}\langle 111\rangle$ in Miller index notation).

A phenomenological power law, as is for example described in (Asaro and Needleman, 1985), is used to determine the slip rates:

$$\dot{\gamma}^{(\eta)} = \dot{\gamma}_0 \left| \frac{\tau^{(\eta)}}{r^{(\eta)}} \right|^{1/m} \mathrm{sign}(\tau^{(\eta)}), \tag{2.47}$$

where $r^{(\eta)}$, $\dot{\gamma}_0$, and m are material dependent parameters for the slip resistance, reference shear rate, and the shear rate sensitivity, respectively. Furthermore, the resolved shear stress, denoted by $\tau^{(\eta)}$, is defined by Schmid's law

$$\tau^{(\eta)} = ((\boldsymbol{F}_\mathrm{e}^\top \cdot \boldsymbol{F}_\mathrm{e}) \cdot \boldsymbol{T}^*) : (\boldsymbol{m}^{(\eta)} \otimes \boldsymbol{n}^{(\eta)}). \tag{2.48}$$

The evolution of slip resistance is described by

$$\dot{r}^{(\eta)} = \frac{\mathrm{d}\hat{\tau}^{(\eta)}}{\mathrm{d}\Gamma} \sum_\zeta \hat{q}_{\eta\zeta} |\dot{\gamma}^{(\zeta)}|, \tag{2.49}$$

with an interaction matrix $\hat{q}_{\eta\zeta}$ taking into account self and latent hardening. This matrix is composed of diagonal elements equal to 1.0 and off-diagonal elements q_1 and q_2 (cf. (Baiker et al., 2014)). In addition, Γ denotes the accumulated plastic shear, which is defined by

$$\Gamma = \int_0^t \sum_\eta \left|\dot{\gamma}^{(\eta)}\right| \mathrm{d}t. \tag{2.50}$$

Finally, an extended Voce-type model is used to describe the hardening behavior (Tomé et al., 1984):

$$\hat{\tau}^{(\eta)} = \tau_0 + (\tau_1 + \vartheta_1 \Gamma)(1 - e^{-\Gamma\vartheta_0/\tau_1}), \tag{2.51}$$

as is used for example in (Pagenkopf et al., 2016; Baiker et al., 2014). The used model contains four material dependent parameters τ_0, τ_1, ϑ_0, and ϑ_1, which are calibrated to DC04 steel.

The reorientation of the crystals is calculated based on a rigid body rotation $\boldsymbol{R}_\mathrm{e}$ (Ling et al., 2005):

$$\boldsymbol{R}_1 = \boldsymbol{R}_\mathrm{e} \boldsymbol{R}_0, \tag{2.52}$$

with the rotation matrices $\boldsymbol{R}_0$ describing the original crystal orientation and $\boldsymbol{R}_1$ describing the orientation after deformation. $\boldsymbol{R}_\mathrm{e}$ is derived using the polar decomposition of $\boldsymbol{F}_\mathrm{e}$:

$$\boldsymbol{R}_\mathrm{e} = \boldsymbol{F}_\mathrm{e} \cdot \boldsymbol{U}_\mathrm{e}^{-1}, \tag{2.53}$$

with the elastic part of the right Cauchy-Green tensor $\boldsymbol{U}_\mathrm{e}$. The orientation of a crystal is therefore expected to change smoothly for an applied load. For tensile load increments, the distances between the initial texture and the evolving texture are expected to increase monotonically.

The presented crystal plasticity model involves solving the constitutive equations using an implicit time integration scheme and an iterative two-stage method to determine the unknown stress tensor in the intermediate configuration $\boldsymbol{T}^*$ and the slip resistance $r^{(\eta)}$ (Kalidindi et al., 1992). Further implementation details can be found in (Pagenkopf, 2019). In (Pagenkopf et al., 2016), it was shown that the model can reproduce experimentally measured rolling textures of a dual phase steel on the basis of a spatially resolved microstructure model. In contrast, the Taylor framework used in this work is known to typically overestimate texture evolution and produce sharper textures (Kocks et al., 1998). Nevertheless, for the purposes of this work, the computationally inexpensive Taylor-type material model is sufficient. Although material parameters for DC04 steel are used in this work, it is to remark that the described Taylor-type crystal plasticity model and the texture generation approach can be applied to any kind of metallic material with bcc crystal structure.

2.3.3 Resistance spot welding

Resistance spot welding is used to connect metal components. In the present application, the components are steel sheets. The steel sheets are squeezed locally by the welding gun electrode tips and form a contact area. The latter is heated by an electric current with simultaneously continued electrode force until the sheet material starts melting in the contact area. When the current is shut off, the material re-solidifies again and forms a welding spot, which joins the two (or more) sheets together. The resulting welding spot usually has the form of an ellipsoid and is characterized by its maximum equatorial diameter. The basic parameters governing the welding process are (1) the time, for which current flow and electrode force are maintained (welding time), (2) the level of the current between the electrodes (welding current) (3) and the squeezing force exerted on the sheet compound by the electrodes (electrode force). Figure 2.13 and 2.14 show the principle resistance spot welding arrangement and procedure.

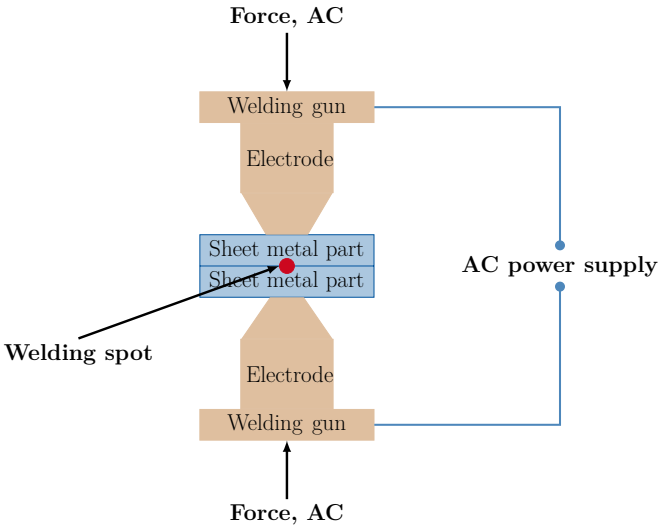

Figure 2.13: Schematic representation of an resistance spot welding arrangement. AC is the abbreviation for alternating current.

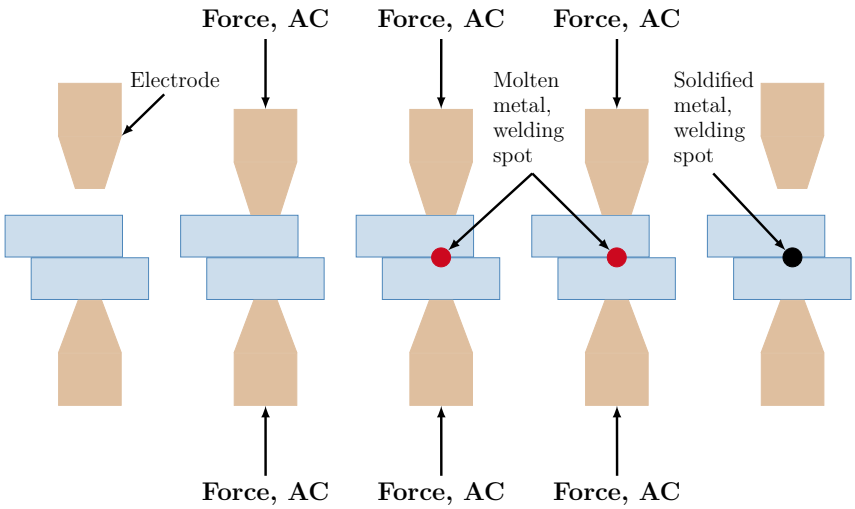

Figure 2.14: Schematic representation of an resistance spot welding procedure. AC is the abbreviation for alternating current.

The process data consists of samples of the measurements of electrode voltage (V) (see Fig. 2.15 (a)) and welding current (kA) (see Fig. 2.15 (b)) measured within a given (varying) welding time. These values can be used to calculate the power curve (kW) (see Fig. 2.15 (c)).

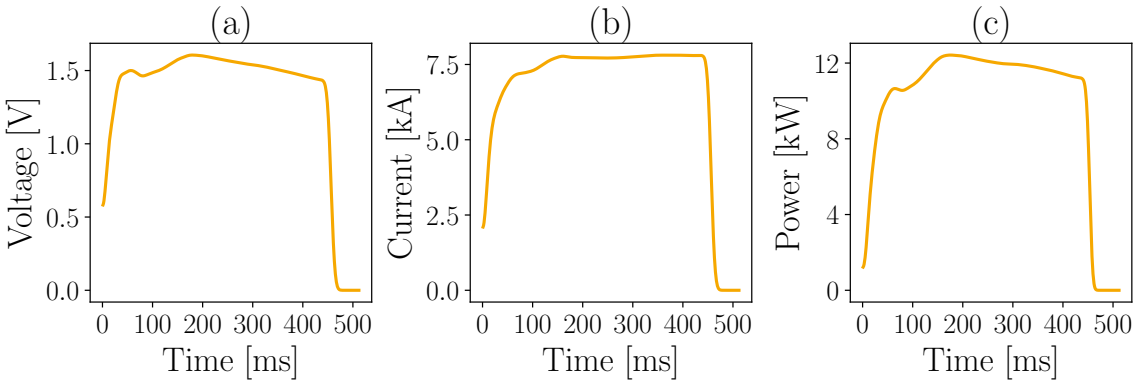

Figure 2.15: (a) Voltage, (b) current and (c) power curves of a welding process.

Chapter 3

State of the art

This section presents a review of recent advancements in the application of machine learning (ML) techniques to materials science, with a particular focus on the key areas where data-driven methods have transformed traditional approaches. The section begins with a time-series analysis of process data, with a particular focus on the use of ML models to improve manufacturing processes. Subsequently, the discussion turns to the domain of materials and process design, in which predictive and generative models are employed to examine and optimize the relationships between processing parameters, microstructures, and material properties. In conclusion, the section examines techniques for measuring texture distances, emphasizing how ML allows for a quantitative comparison of crystallographic textures, thereby facilitating a more profound comprehension of their impact on material behavior.

3.1 Time-series analysis of process data

Process data are by nature sequential and are captured as time series. Suitable representation methods and process models are not only important in manufacturing, but for processes in general such as weather, finance or speech. Thus, distance-based methods with a predefined distance measure such as Dynamic Time Warping (DTW) (Sakoe and Chiba, 1978) and hidden Markov models (Baum and Eagon, 1967; Baum et al., 1970) operate directly with the data to either find similarities with given model templates (DTW) (Keogh and Ratanamahatana, 2005) or to find a most probable state sequence as a cause of an observed time series (Ghahramani and Jordan, 1997). The combination of DTW and the k-nearest neighbors algorithm (k-NN) allows to recognize time-series patterns (Rakthanmanon et al., 2012). In contrast, feature-based methods extract features that represent the pattern of the time series. Methods include, for example, quantizing the features to form a bag-of-words (Lin et al., 2007), or extracting features of different scales to form a bag-of-features (Baydogan et al., 2013). However, these methods require very elaborate manual feature engineering based on expert knowledge.

The use of neural networks for time-series classification, especially by deep structures, allows to automatically find appropriate features, represented by the outputs of dedicated hidden layers. In (Wang et al., 2017) the classification of time series using deep neural networks is investigated, based on the UCR Time Series Classification Archive (Chen et al., 2015) from different application domains, such as electrocardiogram (ECG), Natural Language Processing (NLP) and so on. Convolutional neural networks (CNNs) with filter size decreasing from the

input layer are applied in (Dai et al., 2017) to time series, which represent acoustic signals by raw waveform data. These deep CNNs are used to generate features of the waveform input data for subsequent classification and serve to replace manual feature engineering, for end-to-end learning of the classification task. The input data are sequences with a vector of length 32,000 from which a lower number of features is extracted, which allow sufficient classification quality. (Yan et al., 2020) proposed the use of Temporal Convolutional Networks (TCNs) for forecasting the El Niño–Southern Oscillation (ENSO), thereby addressing the limitations of conventional statistical and dynamical models. TCNs effectively capture long-term dependencies through the utilization of causal and dilated convolutions, resulting in enhanced accuracy and lead-time predictions in comparison to alternative methodologies. The proposed approach offers a simplified modeling alternative to Long Short-Term-Memory (LSTMs) and attention-based models, while achieving comparable or superior results.

Recurrent neural networks (RNNs) (Hopfield, 1982; Williams and Zipser, 1989), namely LSTM networks (Hochreiter and Schmidhuber, 1997) have been established in modeling generators of (time) sequences of data (Graves, 2013). An LSTM can generate process curves from initial values and a distribution of curves from adding noise to the initial and intermediate states. The models are directly represented by the weight values of the LSTMs, which form the representation space of the models. RNNs are enrolled into a finite number of time steps, where the weight values determine the dynamical behavior in the corresponding interval. If important temporal relations extend beyond this interval, the dynamics is not fully captured and generated sequences are not valid.

Variational autoencoders (VAEs) represent similar objects as being generated by a dedicated normal distribution (characterized by a dedicated centroid and diagonal covariance matrix) in a latent space with chosen (low) dimensionality. Objects that form different similar groups are represented by multiple normal distributions. The optimization eventually results in distinct normal distributions with different semantics, which is found from the latent space representation of the condition-labeled sample data. Frequently, the distinct normal distributions are labeled accordingly, and linear interpolation in the latent space between the centroids of the distributions is used to generate objects (by the decoder), which are semantically interpolated (Hou et al., 2017). This is only possible, if the similarities between the objects in the original space reflects the semantics, because no other information than the structure of data itself is used for the training of VAEs. Neither the objective function used for training the VAEs nor the generic VAE structure enforce the capability to semantically interpolate in the latent space (Berthelot et al., 2019).

Only few ML approaches exist, which learn generative models of time series, according to the the architecture of generative adversarial networks (GANs) (Goodfellow et al., 2014). In (Yoon et al., 2019) the joint probability of time series and conditions is learned in the proposed framework. It consists of networks for embedding and recovery (auto-encoding) of the time series data, and a generator (operating on the latent space) and a discriminator for the sequences. This requires the learning of the joint probabilities of the occurrence of time series and conditions simultaneously. The sampling of the condition-time series product space requires a huge amount of data, which are rarely accessible with process data. Another time series generative model approach is presented in (Esteban et al., 2017), which also uses a GAN architecture in combination with a RNN. These RNNs are conditioned on auxiliary information. The latent space in this approach only represents the evidence (unconditioned probability density), from

which the LSTM recurrent network generates conditioned time series by using extra inputs representing the conditions. The goal was to generate training data for medical staff in ICUs. The condition-unspecific latent space does not allow to morph between generators under different conditions. The transfer of the learned knowledge to new conditions is therefore not possible.

3.2 Materials and process design

Pioneering work has been done in the field of materials and process design, however, typically either focusing on solving materials design problems (without taking into account processing) or focusing on solving process design problems for given desired properties (not taking into account the structure of materials). The following discusses works that describe approaches for solving materials and process design problems, with an emphasis on the application of ML.

A generic approach to solve materials design problems is the microstructure sensitive design (MSD) approach introduced in (Adams et al., 2001). Following (Fullwood et al., 2010), MSD can be described by seven steps. First, the properties of interest as well as candidate materials have to be defined. After that, a suitable microstructure definition is applied for these materials yielding a microstructure design space. On this basis, relevant homogenization relations are identified and applied over the whole design space. The resulting property closure can be used to select desired properties, which are then mapped back to the microstructure design space in order to identify optimal microstructures. The last step of MSD aims to determine processes and processing routes needed to produce the identified microstructure. To the authors knowledge, there are only few works that show how to set up process-structure-property linkages for crystallographic texture optimization within the framework of MSD. One approach involves modeling texture evolution as fluid flow in the orientation space. On this basis, so-called processing streamlines are used to guide from one point to another (Fullwood et al., 2010; Li et al., 2007).

The works by Adams (Adams et al., 2001) and Kalidindi (Kalidindi et al., 2004) instantiate the MSD approach for texture optimization. The first one describes how optimal crystallographic textures can be identified in order to improve the deformation behavior of a compliant beam. In the latter, a similar approach is shown to optimize the crystallographic texture for the design of an orthotropic plate. The core of both approaches lies in the usage of a lower dimensional spectral representation of the orientation distribution, cf. (Bunge, 1982). Alternatively, so-called texture evolution networks have been developed within the context of MSD, using a priory sampled processing paths to create a directed tree graph where microstructures are represented by the nodes on the graph (Shaffer et al., 2010). Graph search algorithms are then used to find optimal processing paths from an initial to a targeted microstructures. For more complex microstructure representations, like two-point correlations, feature extraction methods can be applied to reduce the dimensionality. Methods that are used in the context of materials design are principal component analysis (PCA) (Paulson et al., 2017; Gupta et al., 2015) and multidimensional scaling (MDS) (Jung et al., 2019a) for example.

Besides the MSD approach, also ML-based approaches exist that solve crystallographic texture optimization problems in terms of materials design, however, without solving a corresponding optimal processing problem. (Liu et al., 2015) and (Paul et al., 2019) describe iterative sampling approaches that interact with crystal plasticity simulations aiming to identify crystallographic textures for given desired properties. Therefore, an initial set of texture-property

tuples (crystallographic textures and corresponding macroscopic properties) is generated. Via supervised learning, significant features of the parameterized orientation distribution (and in (Liu et al., 2015) also significant regions) are identified that yield optimal or near-optimal solutions. Based on the identified features and regions, new texture-property data points are sampled in order to get closer to the optima.

Another approach for identifying optimal textures is described in (Kuroda and Ikawa, 2004). Therein, a real-coded genetic algorithm (Goldberg, 1991) is described that interacts with a crystal plasticity model in order to find optimal combinations of typical rolling texture components of face-centered cubic metal (Cu, Brass, S, Cube and Goss) for given desired properties. The algorithm starts with an initial set of textures consisting of different fractions of these components. The set of textures evolves iteratively by combining them using operators such as mutation, crossover and selection (Herrera et al., 1998).

Surrogate-based optimization has also been explored in several studies, such as for handling uncertainties in materials design (Balachandran et al., 2016). Alternatively, probabilistic modeling approaches can be used to directly solve the inverse identification problem (Tran and Wildey, 2021).

Recent works (i.e., (Jung et al., 2020) and (Kamijyo et al., 2022)) use Bayesian optimization for microstructure design. In (Kamijyo et al., 2022), a deep neural network is used for the estimation of mechanical properties. On this basis, Bayesian optimization is used to determine optimal volume fractions of texture components of aluminum (cf. (Kuroda and Ikawa, 2004)) for a desired formability. For designing complex microstructure models, in (Jung et al., 2020), the use of the latent space of a convolutional autoencoder as a low dimensional design space is proposed. Within this design space, Bayesian optimization is adopted to search for optimal dual phase microstructures for given desired properties (i.e, tensile strength).

Predicting dual phase microstructure properties using CNNs is also used in (Mann and Kalidindi, 2022), however, to explore the properties space defined by the material stiffness. The CNN architecture was developed for approximating the highly non-linear microstructure-property linkage, while using also two-point spatial correlation functions of the microstructure as input. A further convolutional approach is described in (Tan et al., 2020), in which a deep convolutional generative adversarial network (DCGAN) and a CNN is proposed for the design of materials. Therein, the CNN links the micostructures to its properties and acts as a surrogate model, whereas the DCGAN generates design candidates for a desired compliance tensor.

Recent advancements in inverse design have markedly enhanced the capacity to optimize material properties and structures through data-driven and computational approaches, representing a significant advancement in the field. In a recent study, (Raßloff et al., 2025) demonstrated the efficacy of Bayesian optimization in refining spinodoid structures and efficiently navigating complex design spaces with minimal computational resources. Building on this, (Kumar et al., 2020) employed inverse design in the context of spinodoid metamaterials, thereby illustrating the potential of these materials for attaining optimized mechanical properties that are tailored to specific applications. (Bompas and Sandfeld, 2023) introduced a generative model with novel embeddings for continuous variables, thereby significantly accelerating the inverse modeling process and broadening its applicability. Similarly, (Nobari et al., 2021) developed PcDGAN, a conditional GAN, which enables the generation of diverse material structures that meet targeted property requirements, even in high-dimensional design spaces. (Seibert et al., 2022) underscored the significance of descriptor-based gradient optimization in the reconstruc-

tion of three-dimensional microstructures, emphasizing the importance of efficient representation techniques for optimizing material properties.

Recent advances in generative modeling have increasingly applied diffusion-based approaches to microstructure reconstruction and design. (Lyu et al., 2024) employed denoising diffusion probabilistic models (DDPMs) to reconstruct both two- and three-dimensional microstructures, including chessboard, spinodal, fractal, and porous patterns. They further enabled permeability-controlled conditional generation of three-dimensional porous architectures, which was validated through lattice Boltzmann simulations. (Lee and Yun, 2024) introduced a versatile diffusion-based reconstruction framework applicable across a wide range of microstructure types, demonstrating the generalizability of diffusion models for handling complex morphologies. In related work, (Düreth et al., 2023) applied diffusion models to real-world microstructure data, achieving high realism as assessed by descriptor-based error metrics and Fréchet Inception Distance (FID) (Heusel et al., 2017) scores, and highlighting the potential for extending such approaches to three-dimensional representations. Expanding these ideas, (Baishnab et al., 2024) proposed a latent diffusion model capable of generating three-dimensional microstructures for organic photovoltaics, allowing user control over statistical and topological attributes and even inferring corresponding process parameters. Finally, (Choi et al., 2025) developed an image-guided, closed-loop framework that integrates diffusion-based generation with quantitative Scanning Electron Microscopy (SEM) analysis and particle swarm optimization, enabling both forward prediction and inverse design of battery cathode precursor microstructures. This provides an important context for the multi-method machine learning analysis of porous copper microstructures carried out by (Wijaya et al., 2024)

Summarized, for the solution of microstructure design problems, a linkage from properties to microstructures is required. Such a linkage is often achieved by genetic or optimization algorithms that interact with numerical simulations. However, as these algorithms generally need a lot of function evaluations, it is not reasonable to apply them to complex numerical simulations directly. Instead, the performance can be increased by using numerically simpler surrogate models, see for example (Simpson et al., 2001). Typically, these are supervised learning models that learn the input-output relations of the numerical simulation under consideration. To run optimization algorithms in combination with supervised learning models it is necessary to limit the region in which they operate to the region, which is covered by the training data. One way to achieve this is by training unsupervised learning models on the input data, as it is done in (Jung et al., 2019b) for example using support vector machines (SVMs). From a ML perspective such an approach can be seen as anomaly detection.

Anomaly detection aims to separate data that is characteristically different from the known data of the sample data set, which has been used for training. An extensive overview of anomaly detection methods is given in (Chandola et al., 2009). Furthermore, the works of (Ruff et al., 2021; Chalapathy and Chawla, 2019) offer a comprehensive survey of contemporary deep learning-based methodologies for anomaly detection. Among these, neural network-based autoencoders (Hinton and Salakhutdinov, 2006) are particularly noteworthy, as they seamlessly integrate into multitask learning (MTL) (Caruana, 1997) schemes, in contrast to SVMs. Autoencoder approaches assume that features of a data set can be mapped into a lower dimensional latent feature space, in which the known data points differ substantially from unknown data points. By backmapping into the original space, anomalies can be identified by evaluating the reconstruction error, see for example (Sakurada and Yairi, 2014). In (Sakurada and Yairi, 2014)

it is also shown that autoencoder networks are able to detect subtle anomalies, which cannot be detected by linear methods like PCA. Furthermore, autoencoder networks require less complex computations compared to a nonlinear kernel-based PCA.

Recent developments in anomaly detection include deep learning approaches, like the deep support vector data description method (Deep SVDD) (Ruff et al., 2018). Deep SVDD is an unsupervised anomaly detection method, which is inspired by kernel-based one-class classification and minimum volume estimation, and can be traced back to traditional methods, which are one-class support vector machine (OCSVM) (Schölkopf et al., 2001) and support vector data description (SVDD) (Tax and Duin, 2004). In contrast to autoencoder approaches, Deep SVDD is based on an anomaly detection objective, rather than relying on the reconstruction error. By using Deep SVDD, a neural network is trained while minimizing the volume of a hypersphere that encloses the network representations of the data. By minimizing the objective, Deep SVDD aims to find a preferably small data-enclosing hypersphere and learns to extract the common factors of variation of the data distribution. The aim of the approach is that representations of the normal data lie inside the hypersphere, while anomalous data points lie outside the hypersphere. Thereby, anomalies can be detected based on their distance to the centroid of the hypersphere.

An extension of Deep SVDD is given by the method Improved AutoEncoder for Anomaly Detection (IAEAD) (Cheng et al., 2021) by combining Deep SVDD with autoencoders. The autoencoder is used for the embedding of the features and to preserve the local structure of the data generating distribution, whereas Deep SVDD detects anomalies in the feature space. This is achieved by adding the Deep SVDD loss as a regularization term to the original autoencoder optimization objective (i.e. the minimization of the reconstruction error). However, instead of using the reconstruction error, IAEAD uses the distance to the centroid in feature space for anomaly detection like the original Deep SVDD approach.

Another recently developed approach uses an autoencoder at the example of learning image data by minimizing the reconstruction error (defined as the loss function) (Kwon et al., 2020). The trained model is used for anomaly detection by evaluating the gradients of the reconstruction error with respect to the neural network weights. Gradients are generated through backpropagation to train neural networks by minimizing designed loss functions (Rumelhart et al., 1986). While feeding new input data into to the neural network, the gradients originating from normal data cause only slight changes with respect to the neural network weights, whereas the gradients from anomalous data cause more drastic changes. Thus, anomalies can be detected by measuring how much of the input data does not correspond to the learned information of the network in terms of the gradients.

Regarding process design, ML-based and data mining-based approaches have been proposed by several works, however, without solving the corresponding materials design problem in beforehand. One approach involves the use of principle component analysis to represent one-step and two- to three-step deformation processes and to identify the deformation sequences required for reaching target crystallographic textures (Acar and Sundararaghavan, 2016; Acar and Sundararaghavan, 2018). Alternatively, a database approach can be used that stores microstructure representations and corresponding processing paths (Sundararaghavan and Zabaras, 2005). The database can be searched for desired textures yielding optimal process paths. These process paths can then be fine-tuned using gradient-based optimization. Another option is to store microstructure representations in a lower dimensional feature space generated by a variational

autoencoder (Sundar and Sundararaghavan, 2020). In this lower dimensional feature space, optimal processing paths can be identified using a suitable distance measure.

In a further expansion beyond the conventional limits of classical materials science, the integration of ML into chemistry has brought about a paradigm shift in inverse design and structure-property relationship modeling, thereby enabling the efficient discovery and optimization of novel materials. Inverse design, which entails the identification of molecular structures that exhibit the desired properties, has derived substantial benefits from ML-driven approaches. (Han, 2025) offers a thorough review of the advancements in AI-driven inverse design, emphasizing how generative models, such as VAEs and GANs, facilitate the exploration of vast chemical spaces by predicting structures that meet predefined property constraints. In a similar vein, (Noh et al., 2020) underscores the potential of ML to enable the inverse design of inorganic solid materials. The study demonstrates that data-driven approaches can circumvent the limitations of conventional trial-and-error synthesis by proposing viable compounds based on electronic and structural criteria. Building upon this foundation, (Vogel and Weber, 2025) presents an inverse design methodology tailored for copolymers, integrating stoichiometric and architectural constraints to direct the synthesis of polymeric materials with customized optoelectronic and mechanical properties. In the domain of quantum chemistry, (Fallani et al., 2024) investigate inverse mapping techniques that connect quantum properties to molecular structures, showcasing how machine learning can systematically navigate chemical space to identify novel functional molecules with optimized electronic characteristics.

Beyond inverse design, understanding structure-property relationships remains a fundamental challenge in materials science, particularly in predicting and tailoring material behaviors. Deep learning models, particularly those with interpretable architectures, have played a crucial role in unveiling complex correlations between molecular structures and their macroscopic properties (Vu et al., 2023). Attention-based neural networks have been demonstrated to extract meaningful patterns from material data, offering insights into the key structural features that govern electronic and mechanical performance. Complementing these advancements, (Rittig et al., 2024) has demonstrated the efficacy of graph neural networks (GNNs) in predicting molecular structure-property relationships. By leveraging the inherent topological representation of molecular graphs, GNNs enhance predictive accuracy. This enhancement is achieved by capturing intricate bonding patterns and molecular interactions, enabling data-driven modeling of diverse chemical systems and facilitating property predictions with high reliability.

3.3 Measuring texture distances

Crucial for measuring texture distances is the way how the crystallographic texture, i.e. the orientation distribution function (ODF), is represented. Having grain orientation data, the ODF can be approximated basically by using different methods, representing the ODF under different aspects: (i) on the basis of a series expansion using generalized spherical harmonics (GSHs) as described in (Bunge, 1982) or hyperspherical harmonics (Mason and Schuh, 2008; Mason and Schuh, 2009), (ii) on the basis of orientation histograms (Dornheim et al., 2022), and (iii) by parameterizing characteristic texture components and fibers, see (Delannay et al., 1999) and (Kocks et al., 1998). Also experimental pole figures and inverse pole figures (Kocks et al., 1998) can be used to represent the ODF.

As the ODF is a probability density function, the similarity between two ODFs can be expressed by many density-related distance and divergence measures from probability theory textbooks, such as the four most important measures Kullback-Leibler Divergence (Kullback and Leibler, 1951), Hellinger distance (Hellinger, 1909), Bhattacharyya distance (Bhattacharyya, 1946), and Wasserstein metric (Vaserstein, 1969). Typically used distance measure in material science is defined like in (Adams et al., 2001), where neither the distance nor the neighborhood relations between the corresponding orientations are reflected by this measure.

Instead of expanding into a finite GSH series, the density is approximated by PCA (Pearson, 1901) in (Paulson et al., 2017) to reduce the representation dimension of the density functions (representing texture representative volume elements) (Kalidindi et al., 2011; Niezgoda et al., 2011). The distance is then measured by the Euclidean distance between the PCA coefficient vectors (see (Niezgoda et al., 2013)). The PCA coefficients are ordered according to ascending represented variance of the principal components. In texture analysis, higher variance components are more important, but by using the Euclidean distance all components are weighted equally.

A different distance measure is introduced in (Sundararaghavan and Zabaras, 2005) for fcc textures (e.g. goss, brass, copper, etc. (Kocks et al., 1998)) that are described by their typical components in Rodrigues space. (Sundararaghavan and Zabaras, 2005) uses pole-density functions of essential orientation fibers in the fundamental zone for decomposition. The ODF is then described by a feature vector containing the expansion coefficients as entries. The distance between two ODFs is then measured using the l_2 norm between their two feature vectors. This approach, however, is only applicable for textures that show such characteristic fibers and is therefore not generalizable.

Alternatively, in (Engler et al., 1994) and (Moreau et al., 1994), the ODF distance is measured on the basis of pole figures. This is done by calculating the sum of absolute or squared distance, respectively, between discretized pole figures of the ODFs. Another distance measure based on pole figures was introduced in (Tarasiuk and Wierzbanowski, 1996): The two values of the pole figures of two ODFs are taken at the same position of the discretization grid and stored as entries of a two dimensional vector and then plotted as a point cloud in a 2-d diagram. This scatter plot is then analyzed by performing a linear regression. The regression parameters *offset* and *slope* as well as the *correlation factor* are determined, where the latter is used as a similarity score between the two ODFs. However, this measure ignores the neighborhood relations of the orientations represented in the pole figures.

The idea of using orientation histograms to measure similarities of textures is used by (Eisenlohr and Roters, 2008). This work analyzes the reconstruction of textures by a finite number of orientations with equal weight. In (Eisenlohr and Roters, 2008), the similarity between original and reconstructed ODFs are quantified by the measure proposed in (Tarasiuk and Wierzbanowski, 1996), but also by comparing orientation histograms in the Euler space by the root mean squared deviation of the bin entries of both histograms. The use of histogram bin entries does not consider neighborhood relations between orientations. Furthermore, operating in the Euler space suffers from its distortion.

Dornheim et al. (Dornheim et al., 2022) introduced an alternative approach that uses the Chi-Squared distance (Pele and Werman, 2010) applied to histogram-based texture representations in the space of unit quaternions. By using unit quaternions to describe orientations, distances between individual orientations can be evaluated in a straightforward manner (Huynh,

2009). Moreover, the Chi-Squared distance measure has the particular advantage that it is statistically consistent and minimum and maximum bounds exists. But as before, also only bin entries are compared. Neglecting the underlying orientation distances leads to a distance measure, which does not represent texture distances well, especially in the case of sparse histograms.

A histogram based measure also taking into account the distances between the representatives of the bins is the Earth Mover's Distance (EMD) (Rubner et al., 1997; Rubner et al., 2000). Therefore it is better suited to represent texture distances even with sparse histograms.

Chapter 4

The impact of process conditions on material behavior

Manufacturing processes are subject to variations from fluctuating operating conditions, such as changes in tool (e. g. welding guns) and material properties (e. g. steel sheet combinations). These variations affect the progression of process states, typically captured as time series of quantifiable variables known as process curves. Understanding the relationship between process conditions and process curves is key to optimizing of process control and ensuring quality. This chapter presents a generative modeling approach that uses machine learning to extract and represent the statistical influence of conditions on process curves, yielding a low-dimensional representation that enables efficient modeling, interpretation, and transformation under varying conditions.

The methodology derives a latent feature space using multitask learning (MTL), providing a compact representation of process curves that preserves essential information while emphasizing condition-sensitive features. The MTL framework integrates an autoencoder for reconstructing process curves with condition detectors for classifying representations, thereby capturing salient features within the latent space. Each process execution maps to a distinct point in this space, with condition variations aligned along dedicated regions. Generative models are then built in the latent space using conditional probability functions within a Bayesian framework, enabling probabilistic modeling of process curves across conditions. Transformations of the latent space further allow process curve distributions under one condition to be mapped onto another, supporting the transfer of generative models to new conditions without extensive experimental data.

The proposed approach reduces the experimental workload for modeling process curves under varying conditions. For example, in resistance spot welding, process curves are measured in real time, while weld quality requires destructive testing. By estimating transformations between process curve distributions, quality metrics for new conditions can be inferred without direct measurement, requiring only process curve data. The method also enables generating curves for unobserved conditions: interpolating between transformation models allows creation of generative models for unknown conditions, supporting zero-shot learning (Larochelle et al., 2008; Socher et al., 2013) and hyper-modeling (Link et al., 2016; Reis et al., 2017). This capability broadens the scope of applicability of the proposed approach to intelligent process control, predictive quality assessment, and adaptive manufacturing, where process conditions and requirements may undergo dynamic evolution.

4.1 Methods

A process can be described as a sequence of process states. In manufacturing, the process changes the state of a workpiece and terminates in a final state in which the workpiece should have the desired properties. The time evolution of such processes is, in principle, governed by the fundamental dynamical equations—mathematical formulations such as ordinary or partial differential equations that express conservation of mass, momentum, and energy, together with appropriate constitutive laws. These equations describe how the process state $\boldsymbol{s}$ evolves from an initial state $\boldsymbol{s}_0$ (e.g., the input material), under the influence of process driving forces $\boldsymbol{u}$ (e.g., external loads or heat input) and boundary conditions $\boldsymbol{b}$. In practice, however, the quantities required to fully specify and solve these equations are often only partially known. Instead of the complete initial state, typically only certain properties $\boldsymbol{i}$ of the input (e.g., material type or geometry) are known.

Also instead of the boundary conditions only machine characteristics $\boldsymbol{M}$ (such as tool type) are available in real processes and the process control adjusts only rough control parameters $\boldsymbol{p}$ instead of controlling directly the precise forces driving the process. The input properties and the machine characteristics represent the accessible fixed conditions governing a process, but are ambiguous with respect to the real process boundary conditions. This may lead to different process evolutions and outcomes under the same conditions. The outcomes are represented by some quality measures $\boldsymbol{q}$. The values of the process state variables are usually not directly measurable, but are only represented by a corresponding time series of measurements $\boldsymbol{m}(\mathrm{t}_i)$, the process curve, represented by the vector of sampled values $\boldsymbol{x} = [\boldsymbol{m}(\mathrm{t}_0), ..., \boldsymbol{m}(\mathrm{t}_N)]^\top$. An emprical surrogate process model has therefore to be used in place of the precise dynamical model, relating the available condition quantities $\boldsymbol{c}$ with the state evolution represented by the process curve.

The measured process curve reflects the evolution of the process state, but the exact correspondence between measurements and state variables is often theoretically unknown or ambiguous. However, in certain restricted process domains, the process variables may span only a limited subset of possible states, making it feasible to establish a one-to-one relationship between process curves and state variables. In this case any representative sub-space of the process curves is also representative of the original process. The restriction of the process sub-domain is expressed by limited and small range of conditions.

4.1.1 Semantic process curve transformation and hyper-modeling

The generation of process curves $\boldsymbol{x}_{\mathrm{pc}}$ is subject to a stochastic process. The curve probability density $p(\boldsymbol{x}_{\mathrm{pc}}|\boldsymbol{p}, \boldsymbol{c})$ represents the curve-generating stochastic process and is determined by the parameters $\boldsymbol{p}$ (controlling the forces on the process; usually adjusted by the process machine) and the conditions $\boldsymbol{c}$ simultaneously. In many practical cases the knowledge of the general influence of the conditions on the generator models of process curves is required, regardless of the chosen machine parameters $\boldsymbol{p}$. The condition influence is represented by the law of the transformation of the generator models (densities) according to a change in the condition values. The machine parameters are then considered as stochastic variables and the process curve densities are solely governed by the conditions: $p(\boldsymbol{x}_{\mathrm{pc}}|\boldsymbol{c})$. The process curve densities $p(\boldsymbol{x}_{\mathrm{pc}}|\boldsymbol{c}')$ with a certain condition $\boldsymbol{c}'$ are transformed into a density with new condition $\boldsymbol{c}''$ via

$p(\boldsymbol{x}_{\mathrm{pc}}|\boldsymbol{c}'') = f(p(\boldsymbol{x}_{\mathrm{pc}}|\boldsymbol{c}'), \boldsymbol{c}'')$. This is referred to as a semantic transformation of process curve densities. An application example is how the density of process curves of a reference process (determined in the laboratory) transforms into a new density, when the process is executed under different operating conditions than in the lab.

The density transformation can be expressed as a transformation h of the representation space of the process curves by ${\boldsymbol{x}_{\mathrm{pc}}}^* = h(\boldsymbol{x}_{\mathrm{pc}})$, such that $p^*(\boldsymbol{x}^*|\boldsymbol{p}, \boldsymbol{c}') = p(\boldsymbol{x}_{\mathrm{pc}}|\boldsymbol{p}, \boldsymbol{c}'')$. Such transformtion models $h(\boldsymbol{x}_{\mathrm{pc}})$ are estimated in Section 4.2.4 in the latent space of the process curves. If instances of the transformation models between densities under different conditions are generalized into a general transformation $\hat{\boldsymbol{x}}_{\mathrm{pc}} = \hat{h}(\boldsymbol{x}_{\mathrm{pc}}, \hat{\boldsymbol{c}})$ of a reference density with a condition-representing quantity $\hat{\boldsymbol{c}}$ as a parameter, a hyper-model of the process is generated.

4.1.2 Obtaining the latent feature space for the spot welding application

Instead of training a generative model by means of a generator and a discriminator combined as opponents in a generative adversarial network (GAN) (Goodfellow et al., 2014), Bayesian inference can be used, if the proposed MTL structure is implemented. The advantage of the Bayesian approach is the resulting explicit density function. This can be achieved by an encoder-generated data representation in a latent feature space and a MTL processing scheme of the encoded data. The MTL processing scheme must consist at least of a decoder and one or several classifiers or estimators, all of them attached to the encoder network. The general architecture for obtaining a latent feature space is introduced in Section 2.2 and depicted in Fig. 2.10. The problem to be solved in the evaluation case is to find universal features describing all possible welding curves independently for all specific conditions. These are the welding gun types (reflecting geometry and type of drive) and steel sheet combinations (SCC) (determined by materials, thicknesses and number of sheets). All these conditions have influence on the shape of the welding curves, which has to be reflected by the extracted features. The following describes the MTL approach for process curves.

The proposed approach shown in Fig. 4.1 automatically derives features, which allow the optimal reconstruction of process curves and also capture maximum information about process conditions. The features can therefore represent the process curve deformation caused by different process conditions. Also the effect of different process outcomes on the shape of the process curves is represented by the features. This allows to reveal and describe the influence of conditions and outcomes on the process curves by means of feature space analysis. These components are:

- The encoder, which transforms the process curves to their low-dimensional feature space representations and therefore enables the probability density estimation,
- the condition classifiers to estimate the condition probability posteriors depending on the latent-space features,
- the estimators to incorporate the relation between latent-space features and process outcomes, and
- the decoder to allow the reconstruction of class-conditioned curve probability densities.

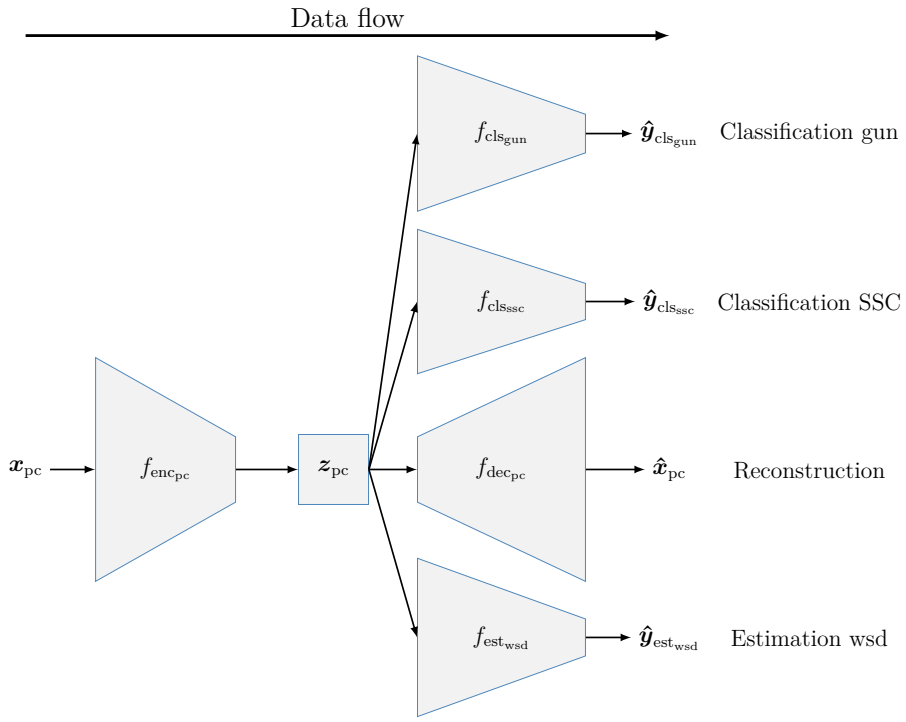

Figure 4.1: Multitask learning architecture for process curves.

Multitask learning objective function

The MTL model, as shown in Fig. 4.1 is optimized on the dataset $\mathcal{D}_{\text{pc}}$ with (1) process curves that are originally represented by vectors of sampled observable variable values $\boldsymbol{x}_{\text{pc}} \in \mathbb{R}^D$, (2) welding guns with different geometry/ drive combinations $\boldsymbol{y}_{\text{cls}_{\text{gun}}} \in \mathbb{Z}^K$, (3) SSC of different standard and high-strength steels $\boldsymbol{y}_{\text{cls}_{\text{ssc}}} \in \mathbb{Z}^K$, and (4) the continuous-valued quality measure welding spot diameter $\boldsymbol{y}_{\text{est}_{\text{wsd}}} \in \mathbb{R}^P$.

The encoder transformation of the data into the low dimensional latent feature space is learned via MTL together with the involved models of the decoder, classifier and estimator tasks, which process the data in the latent feature space. Each of the neural network functions of Fig. 4.1 is parametrized by its weight values, which forms the combined parameter vector $\Theta_{\text{pc}} = [\Theta_{\text{enc}_{\text{pc}}}, \Theta_{\text{dec}_{\text{pc}}}, \Theta_{\text{cls}_{\text{ssc}}}, \Theta_{\text{cls}_{\text{gun}}}, \Theta_{\text{est}_{\text{wsd}}}]^\top$. With the encoder function

$$\boldsymbol{z}_{\text{pc}} = f_{\text{enc}_{\text{pc}}}(\boldsymbol{x}_{\text{pc}}; \Theta_{\text{enc}_{\text{pc}}}), \tag{4.1}$$

we get the latent feature space representation of $\boldsymbol{x}_{\text{pc}}$. The latent feature space representation is used as input to obtain the outputs of the total MTL network, thereby enabling the definition of the loss functions for the four tasks:

1. The decoder network is responsible for the reconstruction and transforms the latent feature vector $\boldsymbol{z}_{\text{pc}}$ approximately back to the original process curve:

$$\hat{\boldsymbol{x}}_{\text{pc}} = f_{\text{dec}_{\text{pc}}}(\boldsymbol{z}_{\text{pc}}, \Theta_{\text{dec}_{\text{pc}}}) = f_{\text{dec}_{\text{pc}}}(f_{\text{enc}_{\text{pc}}}(\boldsymbol{x}_{\text{pc}}, \Theta_{\text{enc}_{\text{pc}}}), \Theta_{\text{dec}_{\text{pc}}}). \tag{4.2}$$

   The reconstruction loss of the decoder is the MSE between the original process curve $\boldsymbol{x}_{\text{pc}}$ and the reconstructed process curve $\hat{\boldsymbol{x}}_{\text{pc}}$:

$$\mathcal{J}_{\mathrm{dec_{pc}}}(\boldsymbol{x}_{\mathrm{pc}}, \hat{\boldsymbol{x}}_{\mathrm{pc}}) = \frac{1}{D}\sum_{i=1}^{D}(x_{\mathrm{pc}_i} - \hat{x}_{\mathrm{pc}_i})^2. \tag{4.3}$$

2. The prediction for the SSC is given by the learned function for the classifier network:

$$\hat{\boldsymbol{y}}_{\mathrm{cls_{ssc}}} = f_{\mathrm{cls_{ssc}}}(\boldsymbol{z}_{\mathrm{pc}}, \Theta_{\mathrm{cls_{ssc}}}) = f_{\mathrm{cls_{ssc}}}(f_{\mathrm{enc_{pc}}}(\boldsymbol{x}_{\mathrm{pc}}, \Theta_{\mathrm{enc_{pc}}}), \Theta_{\mathrm{cls_{ssc}}}), \tag{4.4}$$

   The classifier loss is chosen as soft-max cross entropy Λ between the ground truth class $\boldsymbol{y}_{\mathrm{cls_{ssc}}}$ and the predicted class $\hat{\boldsymbol{y}}_{\mathrm{cls_{ssc}}}$:

$$\mathcal{J}_{\mathrm{cls_{ssc}}}(\boldsymbol{y}_{\mathrm{cls_{ssc}}}, \hat{\boldsymbol{y}}_{\mathrm{cls_{ssc}}}) = \Lambda(\boldsymbol{y}_{\mathrm{cls_{ssc}}}, \hat{\boldsymbol{y}}_{\mathrm{cls_{ssc}}}) \tag{4.5}$$

3. The prediction for the welding gun is given by the learned function for the classifier network:

$$\hat{\boldsymbol{y}}_{\mathrm{cls_{gun}}} = f_{\mathrm{cls_{gun}}}(\boldsymbol{z}_{\mathrm{pc}}, \Theta_{\mathrm{cls_{gun}}}) = f_{\mathrm{cls_{gun}}}(f_{\mathrm{enc_{pc}}}(\boldsymbol{x}_{\mathrm{pc}}, \Theta_{\mathrm{enc_{pc}}}), \Theta_{\mathrm{cls_{gun}}}), \tag{4.6}$$

   The classifier loss is chosen as soft-max cross entropy Λ between the ground truth class $\boldsymbol{y}_{\mathrm{cls_{gun}}}$ and the predicted class $\hat{\boldsymbol{y}}_{\mathrm{cls_{gun}}}$:

$$\mathcal{J}_{\mathrm{cls_{gun}}}(\boldsymbol{y}_{\mathrm{cls_{gun}}}, \hat{\boldsymbol{y}}_{\mathrm{cls_{gun}}}) = \Lambda(\boldsymbol{y}_{\mathrm{cls_{gun}}}, \hat{\boldsymbol{y}}_{\mathrm{cls_{gun}}}) \tag{4.7}$$

4. The forward mapping of the latent feature vector $\boldsymbol{z}_{\mathrm{pc}}$ to the continuous-valued quality measure welding spot diameter $\hat{\boldsymbol{y}}_{\mathrm{est_{wsd}}}$ is represented by the learned function

$$\hat{\boldsymbol{y}}_{\mathrm{est_{wsd}}} = f_{\mathrm{est_{wsd}}}(\boldsymbol{z}_{\mathrm{pc}}, \Theta_{\mathrm{est_{wsd}}}) = f_{\mathrm{est_{wsd}}}(f_{\mathrm{enc_{pc}}}(\boldsymbol{x}_{\mathrm{pc}}, \Theta_{\mathrm{enc_{pc}}}), \Theta_{\mathrm{est_{wsd}}}) \tag{4.8}$$

   The esitmator loss is the squared difference between the predicted welding spot diameter $\hat{\boldsymbol{y}}_{\mathrm{est_{wsd}}}$ and the true welding spot diameter $\boldsymbol{y}_{\mathrm{est_{wsd}}}$:

$$\mathcal{J}_{\mathrm{est_{wsd}}}(\boldsymbol{y}_{\mathrm{est_{wsd}}}, \hat{\boldsymbol{y}}_{\mathrm{est_{wsd}}}) = (\mathrm{y}_{\mathrm{est_{wsd}}} - \hat{\mathrm{y}}_{\mathrm{est_{wsd}}})^2 \tag{4.9}$$

The loss function is composed of the weighted loss terms of all tasks, where $\lambda_{\mathrm{dec_{pc}}}$ is the weight of the decoder loss, $\lambda_{\mathrm{cls_{ssc}}}$ the weight of the condition ssc classifier loss, $\lambda_{\mathrm{cls_{gun}}}$ the weight of the condition gun classifier loss, and $\lambda_{\mathrm{est_{wsd}}}$ the weight of the outcome estimator loss.

$$\mathcal{J}_{\mathrm{pc}} = \lambda_{\mathrm{dec_{pc}}}\mathcal{J}_{\mathrm{dec_{pc}}} + \lambda_{\mathrm{cls_{ssc}}}\mathcal{J}_{\mathrm{cls_{ssc}}} + \lambda_{\mathrm{cls_{gun}}}\mathcal{J}_{\mathrm{cls_{gun}}} + \lambda_{\mathrm{est_{wsd}}}\mathcal{J}_{\mathrm{est_{wsd}}}. \tag{4.10}$$

The parameters of all networks are simultaneously optimized, which means, that for the combined parameter vector Θ_{pc} the optimum Θ^*_{pc} is seeked for all N process curve samples, which is expressed by the search for the minimum of the loss function (c.f. Eq. (2.35)). The optimization also comprises and yields the optimal parameters of the process curve feature space transformation $\Theta^*_{\mathrm{enc_{pc}}}$, giving $\boldsymbol{z}_{\mathrm{pc}} = f_{\mathrm{enc_{pc}}}(\boldsymbol{z}_{\mathrm{pc}}, \Theta^*_{\mathrm{enc_{pc}}})$.

4.1.3 Derivation of class-conditional density functions and generator models

The soft-max trained classifier can be interpreted to deliver the estimated posterior probability $\tilde{P}(\boldsymbol{y}_{\mathrm{cls}_t}|\boldsymbol{z}_{\mathrm{pc}})$ of having class $\boldsymbol{y}_{\mathrm{cls}_t}$, when latent feature vector $\boldsymbol{z}_{\mathrm{pc}}$ is observed. If the dimensionality Z of the latent feature space is small enough that the sample gives a good coverage of the underlying distribution, the probability density of the latent feature vectors $\boldsymbol{z}_{\mathrm{pc}}$ can be estimated as $\tilde{p}(\boldsymbol{z}_{\mathrm{pc}})$. This can be achieved by transforming all N sampled process curves (under all conditions) $\{\boldsymbol{x}_{\mathrm{pc}_n}\}_{n=1}^{N}$ to the latent feature space representations $\{\boldsymbol{z}_{\mathrm{pc}_n}\}_{n=1}^{N}$ and estimating an appropriate density function (i.e. multi-modal Gaussian distribution). The priors of the classes can be estimated as $\tilde{P}(\boldsymbol{y}_{\mathrm{cls}_t})$ by the relative frequencies of the class instances. Then the class-conditional densities $\tilde{p}(\boldsymbol{z}_{\mathrm{pc}}|\boldsymbol{y}_{\mathrm{cls}_t})$ can be estimated via Bayes inference:

$$\tilde{p}_{\boldsymbol{z}_{\mathrm{pc}}}(\boldsymbol{z}_{\mathrm{pc}}|\boldsymbol{y}_{\mathrm{cls}_t}) = \frac{\tilde{P}(\boldsymbol{y}_{\mathrm{cls}_t}|\boldsymbol{z}_{\mathrm{pc}})\tilde{p}(\boldsymbol{z}_{\mathrm{pc}})}{\tilde{P}(\boldsymbol{y}_{\mathrm{cls}_t})} \tag{4.11}$$

The density $\tilde{p}(\boldsymbol{z}_{\mathrm{pc}}|\boldsymbol{y}_{\mathrm{cls}_t})$ can be used to set up a generative model of instance of $\boldsymbol{z}_{\mathrm{pc}}$, when the cumulative distribution function calculated from $\tilde{P}(\boldsymbol{z}_{\mathrm{pc}}|\boldsymbol{y}_{\mathrm{cls}_t})$ is used to map an i.u.d. random variable r to get a sample vector $\boldsymbol{z}_{\mathrm{pc}_s} = \tilde{P}^{-1}(r)$ (Larson and Odoni, 1981).

But a generator of process curves $\boldsymbol{x}_{\mathrm{pc}}$ instead of their latent feature representations $\boldsymbol{z}_{\mathrm{pc}}$ is desired. By construction, a generator of the approximate process curves can be identified as $\hat{\boldsymbol{x}}_{\mathrm{pc}} \approx \boldsymbol{x}_{\mathrm{pc}}$ using the decoder, which is represented by the learned function $\hat{\boldsymbol{x}}_{\mathrm{pc}} = f_{\mathrm{dec}_{\mathrm{pc}}}(\boldsymbol{z}_{\mathrm{pc}})$, which transforms feature representations to the original process curve space.

$$\tilde{p}_{\hat{\boldsymbol{x}}_{\mathrm{pc}}}(\hat{\boldsymbol{x}}_{\mathrm{pc}}|\boldsymbol{y}_{\mathrm{cls}_t}) = \tilde{p}_{\boldsymbol{z}_{\mathrm{pc}}}(f_{\mathrm{dec}_{\mathrm{pc}}}(\boldsymbol{z}_{\mathrm{pc}})|\boldsymbol{y}_{\mathrm{cls}_t})\left|\frac{d\boldsymbol{z}_{\mathrm{pc}}}{df_{\mathrm{dec}_{\mathrm{pc}}}(\boldsymbol{z}_{\mathrm{pc}})}\right| \tag{4.12}$$

The autoencoder property allows for the approximation of

$$\frac{d\boldsymbol{z}_{\mathrm{pc}}}{df_{\mathrm{dec}_{\mathrm{pc}}}(\boldsymbol{z}_{\mathrm{pc}})} = \frac{d\boldsymbol{z}_{\mathrm{pc}}}{d\hat{\boldsymbol{x}}_{\mathrm{pc}}} \approx \frac{d\boldsymbol{z}_{\mathrm{pc}}}{d\boldsymbol{x}_{\mathrm{pc}}} = \frac{df_{\mathrm{enc}_{\mathrm{pc}}}(\boldsymbol{x}_{\mathrm{pc}})}{d\boldsymbol{x}_{\mathrm{pc}}} \tag{4.13}$$

And insert Eq. (4.13) into Eq. (4.12) to finally get

$$\tilde{p}_{\hat{\boldsymbol{x}}_{\mathrm{pc}}}(\hat{\boldsymbol{x}}_{\mathrm{pc}}|\boldsymbol{y}_{\mathrm{cls}_t}) = \tilde{p}_{\boldsymbol{z}_{\mathrm{pc}}}(f_{\mathrm{dec}_{\mathrm{pc}}}(\boldsymbol{z}_{\mathrm{pc}})|\boldsymbol{y}_{\mathrm{cls}_t})\left|\frac{df_{\mathrm{enc}_{\mathrm{pc}}}(\boldsymbol{x}_{\mathrm{pc}})}{d\boldsymbol{x}_{\mathrm{pc}}}\right| \tag{4.14}$$

The estimated and approximated density $\tilde{p}_{\hat{\boldsymbol{x}}_{\mathrm{pc}}}(\hat{\boldsymbol{x}}_{\mathrm{pc}}|\boldsymbol{y}_{\mathrm{cls}_t})$ is a generator model of process curves. This can be used to study the general influence of a process condition on the process curves. The influence of a continuous-valued condition or outcome can be revealed from data in a similar way only if the condition is discretized in intervals and a classifier for the condition-value intervals is used to find the present condition range.

The continuous condition estimator as shown in Fig. 4.1 can be used to derive the distribution of the continuous-valued quality measures $\boldsymbol{q}$, which is related with the distribution of process curves of a discrete condition (Eq. (4.11)), which is derived by Bayesian inference as described above. For the example of resistance spot welding, this might be the distribution of the quality measure welding spot diameter (continuous condition) of different welding guns and metal sheet combinations (discrete conditions).

With the estimator for a single quantity $\boldsymbol{y}_{\text{est}_{\text{wsd}}}$ represented by $\boldsymbol{y}_{\text{est}_{\text{wsd}}} = f_{\text{est}_{\text{wsd}}}(\boldsymbol{z}_{\text{pc}})$, the distribution of $\boldsymbol{q}$ can be found by

$$\tilde{p}_{\boldsymbol{y}_{\text{est}_{\text{wsd}}}}(\boldsymbol{y}_{\text{est}_{\text{wsd}}}|\boldsymbol{y}_{\text{cls}_t}) = \tilde{p}_{\boldsymbol{z}_{\text{pc}}}(f_{\text{est}_{\text{wsd}}}(\boldsymbol{z}_{\text{pc}})|\boldsymbol{y}_{\text{cls}_t}) \left| \frac{d\boldsymbol{z}_{\text{pc}}}{df_{\text{est}_{\text{wsd}}}(\boldsymbol{z}_{\text{pc}})} \right| \tag{4.15}$$

With the method presented so far the statistical properties of the process under different conditions can be modeled and the influence of such conditions be studied in terms of moments of the modeled distributions. The resulting density functions under different conditions can be used as well to estimate transformation functions of the latent space (and thereby of the curve representation space), which transform the densities from one into the other, as discussed in the previous section.

4.1.4 Topologies of the neural networks

The proposed concept of Section 4.1.2 is instantiated to reveal the influence of conditions and outcomes on the process curves of resistance spot welding processes. Different neural network topologies are investigated and compared with each other and with a PCA feature extraction as non-neural baseline. For the feature extraction, both, multilayer perceptrons (MLPs) and convolutional neural networks (CNNs), are used. The difference between MLP- and CNN based features is the locality of the feature properties in the original curves. CNN are better suited if local properties are more relevant, otherwise MLP are the better choice. The goal is to fulfill all tasks satisfactorily with only a few features. Two, three, four and eight features will be use in the evaluation experiments.

Temporal Convolutional Networks (TCNs) are proposed to be used for the encoder network because of the advantages of deep CNNs for feature extraction found by (Dai et al., 2017) and because of the fact that process curves can be thought of a composition of time-localized curve-shape primitives. Another motivation to propose the use of deep CNNs is their success in image analysis, where they have been able to significantly improve the results in segmentation, object detection and classification.

The most common CNN architectures align several convolutional layers, including the rectified linear unit (ReLU) activation (c.f. Eq. (2.11)), which are followed by pooling layers. This pattern is repeated until the input data is reduced to a small representation. In addition, a subsequent processing by fully connected layers is common. The last fully connected layer generates the output data.

Instantiated encoder topologies and parameters

The MLP topology (see Fig. A.7 and Tab. A.1) includes three fully hidden connected layers that halve the size of the preceding layer. The result of the third hidden layer is mapped to the fourth hidden layer in accordance with the dimensions of the latent feature space.

The layout of the CNN topologies (CNN-VA1, CNN-VA2 and CNN-VB1) in Fig. A.8, Fig. A.9, and Fig. A.10 follows the findings of Section 3.1. The first convolutional layer of each network has a filter kernel size of 25. This allows the detection of local structures of the power curves. Another common property is a stride of length one and thereby no sub-sampling in the convolutional layers. Sub-sampling is performed instead in the pooling layers with stride of length two, halving the pooling layer input. Each convolutional layer of all topologies contains a

ReLU activation function to process the convolution result. In each topology, the convolutional layers and the pooling layers are stacked multiple times and their output is vectorized. This high-dimensional vector is finally processed by a combination of two consecutive fully connected layers with ReLU activation function to produce the low-dimensional feature vector. The topologies differ in the number of the convolutional and pooling layers (variant A1, variant A2) and in their arrangement (variant B1).

CNN-VA1 (see Fig. A.8) consists of two consecutive blocks composed of one convolutional and one pooling layer. The result of the last pooling layer is vectorized and used for further processing by the fully-connected network combination. The Structure CNN-VA2 (see Fig. A.9) has one additional block of Conv-ReLU and Pooling layer and is otherwise like CNN-VA1. The effect of the additional block is two-fold: Higher order relations between curve features are considered and the size of the vector layer is halved due to the additional pooling layer. CNN-VB1 (see Fig. A.10) has the same general structure as CNN-V A1, but with different blocks, which consist of two consecutive convolutional layers, followed by one pooling layer. The final output of the last fully connected (plus ReLU) layer forms the feature vectors in each case. Every architecture is investigated with two, three, four and eight features.

In addition to the investigation of the proposed neural network structures, the linear feature extraction via PCA is used to create a performance baseline for the comparison. The number of output components equals two, three, four and eight, analogous to the feature vector size.

Latent feature space processing network topologies and parameters

The data of the feature vector layer are used as input data for the feature-vector processing networks. The processing network structures used for the classifications tasks (three welding gun types and eight steel sheet combinations), for the estimator task (of the welding spot diameter) and the decoder task (reconstruction of the input welding curve) are discussed in the following. For the classification (c.f. Fig. A.11 and Fig. A.12), the low-dimensional feature space input is first non-linearly transformed to a higher-dimensional space to enable the formation of a non-linear class-separation surface of the classifier. This is achieved by a fully connected layer with ReLU activation function. This is followed by another such layer, reducing again the dimensionality to deliver the final features for the Softmax classification layer, which assign confidence values between zero and one to the class outputs. The estimator (c.f. Fig. A.13) has the same first two processing layers, which feed a linear final regressor, giving a float output value for the welding spot diameter. The decoder (c.f. Fig. A.14) has a typical up-sampling layout by increasing the dimensionality from the feature vector size up to the number of sampling values of the process curve by three fully-connected layers with ReLU activation function and a subsequent linear layer. The number of neurons can be seen in Tab. A.2.

4.2 Results

The approach presented in Section 4.1.2 is evaluated in two steps. In the first step, for every task, the encoder block is connected only to one single processing task to check if the dimensionality of the feature vector space is in principal sufficient to bear the required information after training the corresponding single task. After this has been proven, the full MTL is performed on the data. The resulting generalized features are evaluated with all connected feature processing

networks to check, if they still carry the information about the conditions. Furthermore, the precision of the input reconstruction is analyzed for the generalized features.

4.2.1 Data set for the spot welding application

Welding process parameters

The available experimental data set $\mathcal{D}_{\text{pc}}$ comprise almost 4000 power curves $\boldsymbol{x}_{\text{pc}}$ with (1) three different welding guns with different geometry/ drive combinations $\boldsymbol{y}_{\text{cls}_{\text{gun}}}$ with guns 'Gun1', 'Gun2' and 'Gun3', (2) eight SSC of different standard and high-strength steels $\boldsymbol{y}_{\text{cls}_{\text{ssc}}}$, and (3) the continuous-valued quality measure welding spot diameter $\boldsymbol{y}_{\text{est}_{\text{wsd}}}$. The welding process parameters are summarized in Tab. 4.1. The curves are sampled by measuring the observables during process executions under different conditions. A representative sample requires that the condition space is sampled with sufficient density. After each process execution, the respective process outcomes are measured.

Table 4.1: Welding process parameters.

Parameter	Value
Gun types	Gun1, Gun2, Gun3
Material combinations	SSC1, SSC2, ..., SSC8
Welding time	Milliseconds (ms)
Welding current level	Kiloamps (kA)
Electrode force	Kilonewtons (kN)

Preprocessing

The following describes the preprocessing of the sampled raw data. The considered power curves are sampled at 1 ms intervals and last for 512 ms, where the power may be cut (be zero) after shorter periods. The power curves are determined by the total resistance of the steel sheets plus the resistance of the welding gun itself, which is constant and irrelevant for the welding process. Therefore the resistance R_0 is measured during a weld without steel sheets and the corresponding welding gun power contribution subtracted from the calculated power values of the welding curve. The calculated power curves are subject to measurement noise and smoothed via a discrete Gaussian filter with a standard deviation of 1.7 ms and a length of the Gaussian window of 6 ms. The curves are standardized to have zero mean and unit variance.

4.2.2 Validation of the neural network model

Evaluation methods and model training

In the context of hyper-parameter optimization of the neural networks, the method grid search (Bergstra and Bengio, 2012) is used. Based on the analysis of the hyper-parameters, the following decisions are made about the investigated hyper-parameters: The Xavier method (Glorot and Bengio, 2010) is used for weight initialization. The ReLU activation function is used due to the best results in terms of accuracy. The Adam optimizer (Kingma and Ba, 2015) was found to

be most efficient. A descending size of the feature maps in the encoder network was selected. Average-Pooling is used for sub-sampling in a Gaussian resolution pyramid.

The neural networks use the power curves as input data, which are subject to the preprocessing of 4.2.1. The data set $\mathcal{D}_{\text{pc}}$ is split in 15% test data $\mathcal{D}_{\text{pc}}^{\text{test}}$ and 85% training data $\mathcal{D}_{\text{pc}}^{\text{train}}$. The sample distributions of the different welding gun types and steel sheet combinations (SCC) are stratified in the data set. In addition, the data for the training data $\mathcal{D}_{\text{pc}}^{\text{train}}$ is divided into several batches. After each training epoch, the entire test data set $\mathcal{D}_{\text{pc}}^{\text{test}}$ is applied to the network to determine the respective quality measures:

1. Classification of the welding gun type and the SCC: The Softmax layer of the classifier delivers estimates of the Posterior of the classes. The class label with maximum Posterior is assigned to the input data. This results in true positive (TP), true negative (TN), false positive (FP) and false negative (FN) predictions, which are presented by the confusion matrix. The accuracy gives an insight into how many objects are correctly classified:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{FP} + \text{FN} + \text{TP} + \text{TN}} \tag{4.16}$$

2. The estimator quality is measured by comparing the estimated diameter $\hat{\boldsymbol{y}}_{\text{est}_{\text{wsd}}}$ and the ground truth diameter $\boldsymbol{y}_{\text{est}_{\text{wsd}}}$. The measure is the MSE between ground truth and prediction:

$$\text{MSE} = \frac{1}{\left|\mathcal{D}_{\text{pc}}^{\text{test}}\right|} \sum_{i=1}^{\left|\mathcal{D}_{\text{pc}}^{\text{test}}\right|} (\boldsymbol{y}_{\text{est}_{\text{wsd}\,i}} - \hat{\boldsymbol{y}}_{\text{est}_{\text{wsd}\,i}})^2 \tag{4.17}$$

3. The reconstruction error of the decoder is represented by the Mean Relative Absolute Deviation (MRAD). The sampled, measured power curve is represented by $\boldsymbol{x}_{\text{pc}}$ and the corresponding, reconstructed power curve by $\hat{\boldsymbol{x}}_{\text{pc}}$. The MRAD is then the mean of the relative, absolute differences between the $j = 1, \ldots, D$ vector components, averaged over all N samples.

$$\text{MRAD} = \frac{1}{\left|\mathcal{D}_{\text{pc}}^{\text{test}}\right|} \sum_{i=1}^{\left|\mathcal{D}_{\text{pc}}^{\text{test}}\right|} \frac{1}{D} \sum_{j=1}^{D} \frac{|x_{\text{pc}_{j,i}} - \hat{x}_{\text{pc}_{j,i}}|}{x_{\text{pc}_{j,i}}} \tag{4.18}$$

Based on these measures, early-stopping (Prechelt, 1998) is applied to save the best interim result in order to prevent overfitting. Early stopping should be used almost universally to prevent overfitting (Goodfellow et al., 2016) in MLPs and CNNs, why it is used in combination with L_2 regularization (Krogh and Hertz, 1991) in this work. The training data is stochastically re-sampled within each epoch. The training comprises 4000 epochs.

Each network is trained and evaluated ten times with differently sampled training and test data. From these ten evaluation results, the mean and the standard deviation are calculated to show the quality and robustness of the MTL results, where a higher standard deviation implies lower robustness of the topology w.r.t. the data.

4.2.3 Evaluation results with generalized features

The outcomes of all tasks, including welding gun classification, SCC classification, estimation of the welding spot diameter, and reconstruction of the welding curve, are presented for all architectural variants (MLP, CNN-VA1, CNN-VA2, CNN-VB1, as outlined in Section 4.1.4) and for PCA feature extraction (as a conventional baseline) for latent feature space dimensions of two, three, four, and eight. Subsequently, a comparison of the implemented architectures is presented based on the quality criteria.

The performance (quality and robustness) of the CNN encoder structures slightly outperform MLP in most cases and both neural structures are significantly superior to the linear PCA baseline. The performance improves with the number of latent features for CNN, MLP and PCA. This is shown in Fig. 4.2, which shows the dependency of the quality metrics for the tasks of welding gun and SCC classifications, of the estimation of the welding spot diameter and the reconstruction error of the process curves on the latent feature space size N of the topology CNN-VA2.

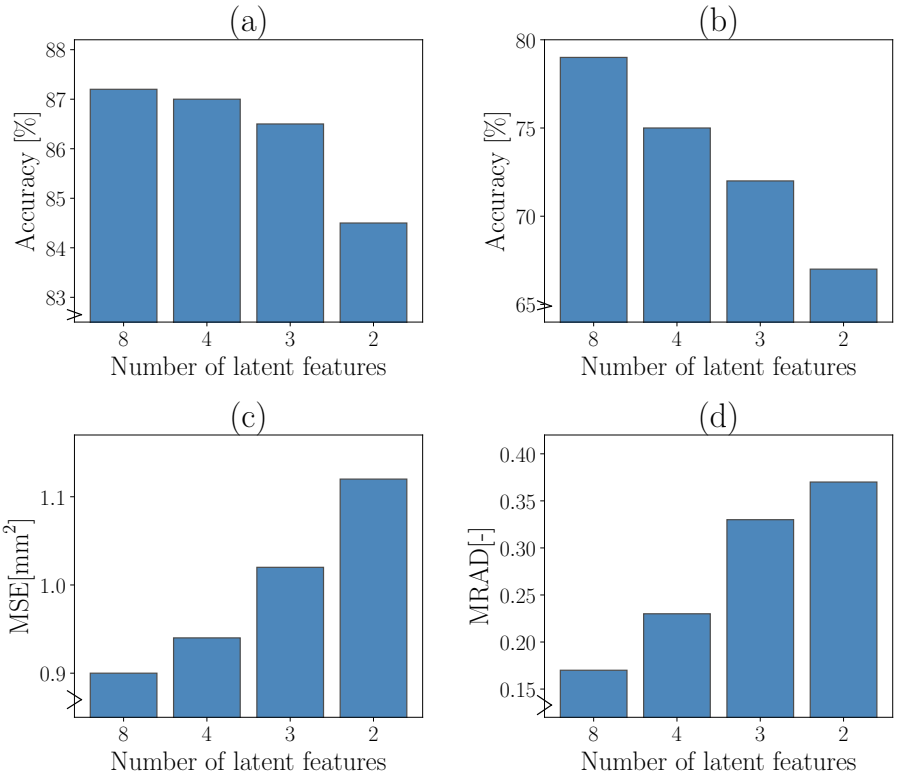

Figure 4.2: Comparison of the results for different numbers of latent features of the top-ranking topology CNN-VA2 for the tasks (a) welding gun classification, (b) steel sheet combination classification, (c) welding spot diameter estimation and (d) reconstruction.

The quality metrics of different topologies, all with the same number of latent features of eight are compared in Fig. 4.3. The different network topologies show different quality rank order for different tasks. The decision upon a dedicated topology is determined by the average over the weighted ranks of all tasks. The rank of a task is weighted by the relative importance of the respective task. In this work, no task preference is assumed and weight one used for all tasks. The resulting average rank values of all topologies (except the linear PCA) are comparable, while

CNN-VA2 shows the slightly best overall performance (including robustness), as shown in Fig. 4.2 and Fig. 4.3.

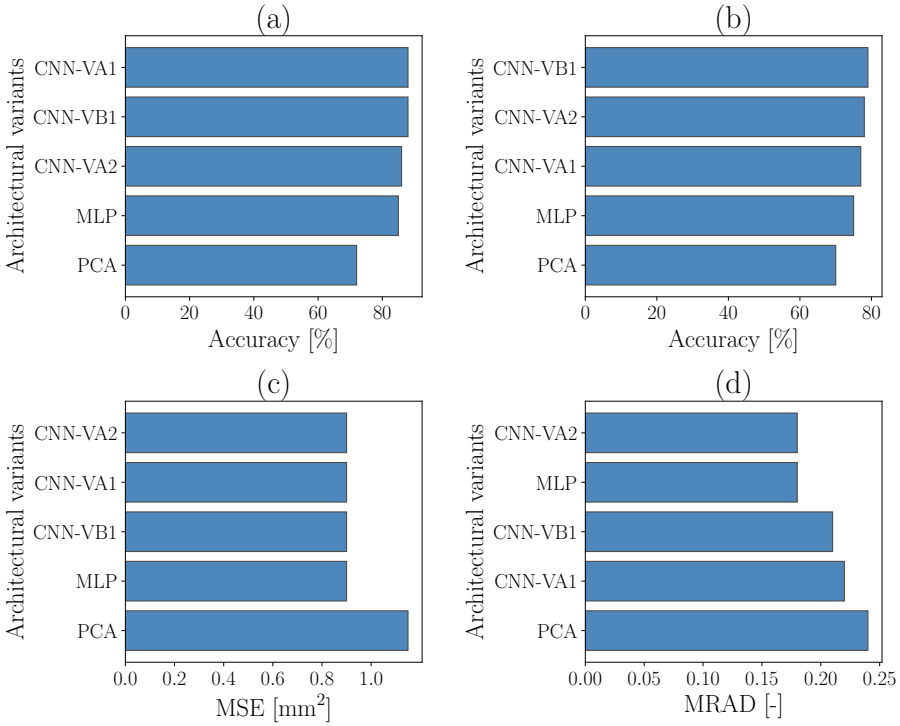

Figure 4.3: Comparison of the results for numbers of latent features of 8 of the investigated topologies for the tasks (a) welding gun classification, (b) steel sheet combination classification, (c) welding spot diameter estimation and (d) deconstruction.

The evaluation shows that even CNN with the number of latent features as small as three has smaller, but sufficient representation power. The performance of the representative top-ranking CNN-VA2 structure with the number of latent features (denoted as LF) eight and three are compared to PCA in the Tab. 4.2.

4.2.4 Latent feature space representation and conditional generative and transformation models

First, the welding curve representations are compared in the latent feature space, which results from only an auto-encoder, with a latent-space representation from MTL. Next the conditional densities are estimated in the MTL latent space from the encoded sample curves via Kernel-density (Rosenblatt, 1956) approximation. The density functions of different conditions (two different welding guns) are used to estimate a transformation of the latent space variables, which transforms one density on the other. The representations of individual welding curves from one welding gun are transformed accordingly to the other gun, then reconstructed by the decoder and compared to the original curve belonging to the nearest neighbour representation of the target gun. The experimental investigations below are restricted to the case of two welding guns only, because the third welding gun (which was used in MTL training) has only a low number of samples, preventing a meaningful estimation of a conditional density.

Table 4.2: Evaluation results. The accuracy is defined in Eq. (4.16). The measured ground truth of the spot diameter in the sample has a range between 2.5 mm and 8 mm and a standard deviation of 0.6 mm. The numbers of latent features are denoted as LF.

Welding gun classification	Accuracy	Standard deviation of accuracy
PCA, LF=8	72.27%	2.20%
CNN-VA2, LF=8	87.14%	1.64%
CNN-VA2, LF=3	86.51%	1.34%

Steel sheet combination classification	Accuracy	Standard deviation of accuracy
PCA, LF=8	71.62%	1.89%
CNN-VA2, LF=8	79.36%	1.74%
CNN-VA2, LF=3	70.86%	4.95%

Welding spot diameter estimation	MSE	Standard deviation of MSE
PCA, LF=8	1.11 mm^2	0.08 mm^2
CNN-VA2, LF=8	0.90 mm^2	0.03 mm^2
CNN-VA2, LF=3	1.01 mm^2	0.13 mm^2

Reconstruction	MRAD	Standard deviation of MRAD
PCA, LF=8	0.23	0.09
CNN-VA2, LF=8	0.18	0.06
CNN-VA2, LF=3	0.32	0.11

Influence of multitask learning on the latent feature space and densities

MTL with auto-encoder, classifier and estimator loss functions has the effect of separating the feature representations of the welding curves according to the respective semantics of the classifiers and estimators. This effect is shown in Fig. 4.4, where the distributions of the sample curve representations are compared in the latent feature space, resulting from MTL and autoencoder-only learning. The improvement in arranging the representations in semantically separated areas by MTL is clearly visible for the welding guns (Fig. 4.4 bottom left) and SSC (Fig. 4.4 bottom right), which are much better separated in the MTL case.

From these samples the corresponding conditional densities $p_{\text{est}}(\boldsymbol{z}_{\text{pc}}|\boldsymbol{y}_{\text{cls}_{\text{gun}}}, \boldsymbol{y}_{\text{cls}_{\text{ssc}}}), \boldsymbol{z}_{\text{pc}} \in$ latent space (conditions: class labels of welding gun type $\boldsymbol{y}_{\text{cls}_{\text{gun}}}$ and of SCC $\boldsymbol{y}_{\text{cls}_{\text{ssc}}}$) can be estimated from the condition-filtered latent-space sample points using e.g. the Kernel density method. The resulting two welding-gun conditional densities for the specific SSC8 $p_{\text{est}}(\boldsymbol{z}_{\text{pc}}|\boldsymbol{y}_{\text{cls}_{\text{gun1}}}, \boldsymbol{y}_{\text{cls}_{\text{ssc8}}})$ and $p_{\text{est}}(\boldsymbol{z}_{\text{pc}}|\boldsymbol{y}_{\text{cls}_{\text{gun2}}}, \boldsymbol{y}_{\text{cls}_{\text{ssc8}}})$ are shown in Fig. 4.5 with the sample point clouds, which were used to estimate the densities, in colour blue and red, respectively. The estimated densities represent conditional generative model, which allows the analysis of the effect of the conditions on the densities, which is studied in the next sub-section.

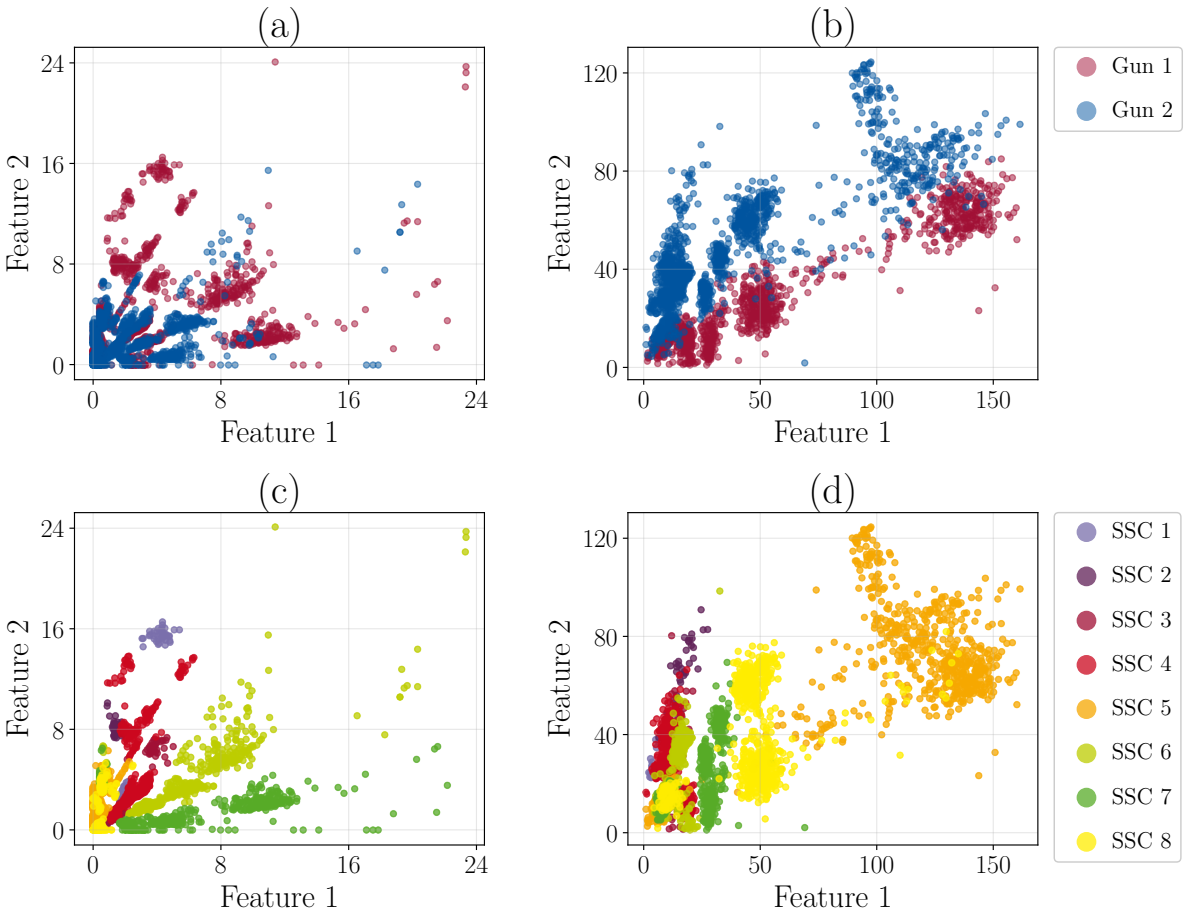

Figure 4.4: Comparison of representations in feature spaces learned with autoencoder-only learning (left column: (a) and (c)) and with multitask learning (right column: (b) and (d)). Top row: welding gun color labeled sample points (red: Gun 1, blue: Gun 2). Bottom row: SSC color labeled sample points (color code: see legend).

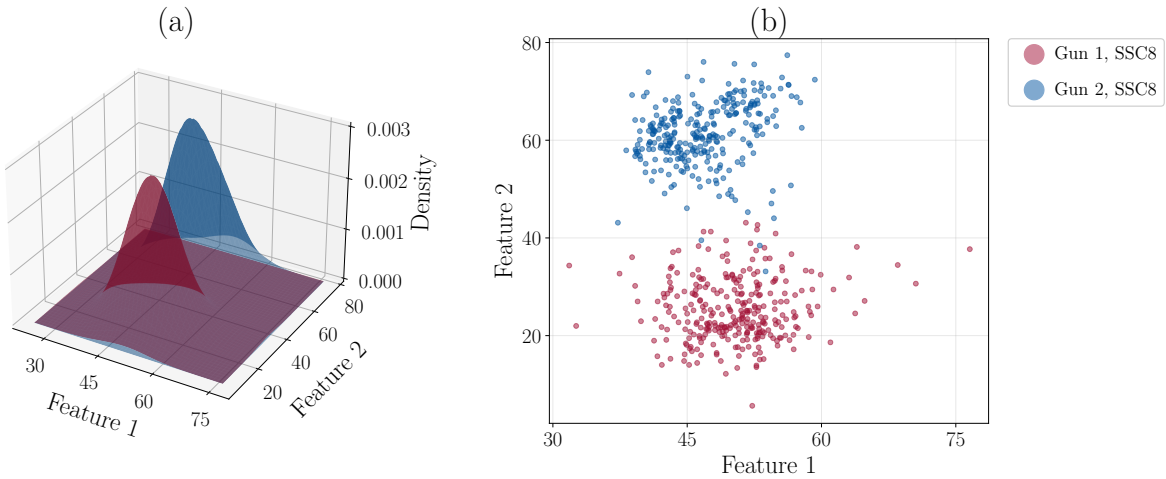

Figure 4.5: Estimated density functions (a) for welding gun 1 and welding gun 2, calculated from sample points shown in (b). For colors see legend.

Transformation models of latent feature space densities with different conditions

Density functions can be transformed one into the other by a transformation of the latent space variables, which represents the effect of the conditions on the latent space in the sense, that the stochastic generative process of one condition is transformed in a way that it generates samples, which obey the same distribution as for the other condition. In other words: The stochastic generative process under a certain condition is transformed into a stochastic generative process under a different condition. Models of such transformations could be estimated assuming a generic transformation (such as affine transformation) and estimating the transformation parameter values. From multiple such models (parameter value instances) a model of the dependency of the parameters on the conditions can be established, representing a process hyper-model, as introduced in the introduction. Similar ideas are applied in applications of VAEs (as discussed in Section A.1.3), where -instead of modeling condition-dependency of the densities- only linear interpolations between samples are applied directly in the latent space. The remaining part of the subsection shows the estimation of the latent-space transformation, which maps the density $p_{\text{est}}(\boldsymbol{z}_{\text{pc}}|\boldsymbol{y}_{\text{cls}_{\text{gun1}}}, \boldsymbol{y}_{\text{cls}_{\text{ssc8}}})$ on density $p_{\text{est}}(\boldsymbol{z}_{\text{pc}}|\boldsymbol{y}_{\text{cls}_{\text{gun2}}}, \boldsymbol{y}_{\text{cls}_{\text{ssc8}}})$ as an example of this procedure. A linear transformation matrix $\boldsymbol{A}$ of the latent-space coordinates $\tilde{\boldsymbol{z}}'_{\text{pc}} = \boldsymbol{A}\tilde{\boldsymbol{z}}_{\text{pc}}$ in homogeneous coordinates (to also include shift transformations) $\tilde{\boldsymbol{z}}_{\text{pc}} = [\boldsymbol{z}_{\text{pc}}, 1]^\top$ is used, and the squared difference loss function is minimized to yield the estimated transformation

$$\boldsymbol{A}^* = \text{argmin}_{\boldsymbol{A}}[p_{\text{est}}(\mathbf{A}\tilde{\boldsymbol{z}}_{\text{pc}}|\boldsymbol{y}_{\text{cls}_{\text{gun1}}}, \boldsymbol{y}_{\text{cls}_{\text{ssc8}}}) - p_{\text{est}}(\tilde{\boldsymbol{z}}_{\text{pc}}|\boldsymbol{y}_{\text{cls}_{\text{gun2}}}, \boldsymbol{y}_{\text{cls}_{\text{ssc8}}})]^2. \tag{4.19}$$

The minimization is performed by equidistantly sampling the (via Kernel-density) estimated densities in the original and transformed space and finding the optimal transformation. The optimization is performed with an Adam optimizer (Kingma and Ba, 2015). A regularization term is included in the loss function, which keeps the integral of the transformed density close to one, taking care of the normalization property also of the transformed density. This procedure is applied to the two densities as introduced above to reveal the influence of the welding gun in the case of SSC8.

The two original densities with fixed SSC8 condition and Gun1 and Gun2 conditions, estimated from the corresponding samples, are shown with the overlayed transformed gun1 density in Fig. 4.6. It can be seen that the chosen linear transform is sufficient in this case and the transformed Gun1 density is in good agreement with the Gun2 density.

When the transformation is applied directly to the Gun1 sample points, a good agreement with the point cloud of Gun2 is observed, as shown in Fig. 4.6 (c). To understand how a change in process conditions (such as tool types, workpiece material properties, and so on) affects the evolution of the process, as reflected by the process curve, a process curve transformation is estimated based on the condition change. The result in Fig. 4.6 (c) shows that the point cloud of the measured sample points under condition 1 almost coincides with the transformed point cloud of the sample points under condition 2, where each point was mapped by the transformation matrix, which was optimized to map the density function under condition 2 on the density function under condition 1. This nearly coincidence implies that a model for the transformation between the generating processes under both conditions was found. The comparison of the reconstructed curves of two transformed Gun1 sample points with the reconstruction of its nearest Gun2 neighbors in Fig. 4.7 shows a high similarity.

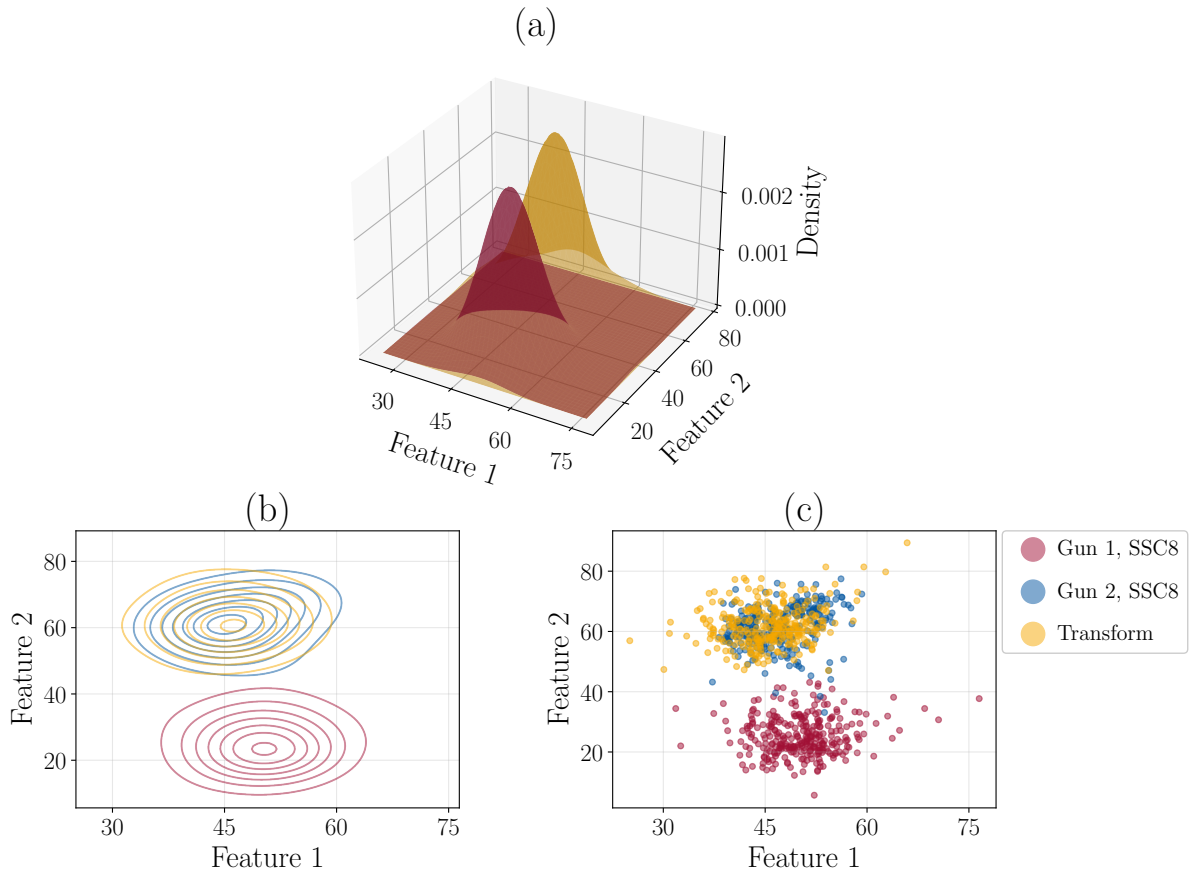

Figure 4.6: (a) Transformed density (yellow color) from estimated linear latent space transformation of welding gun 1 density (red color). (b) Contour plot of transformed densities as shown in figure (a) along with welding gun 2 density (blue color) for comparison. (c) Result (yellow) of the feature space transformation on the sample points of Gun1 (red), overlayed with Gun2 sample points (blue).

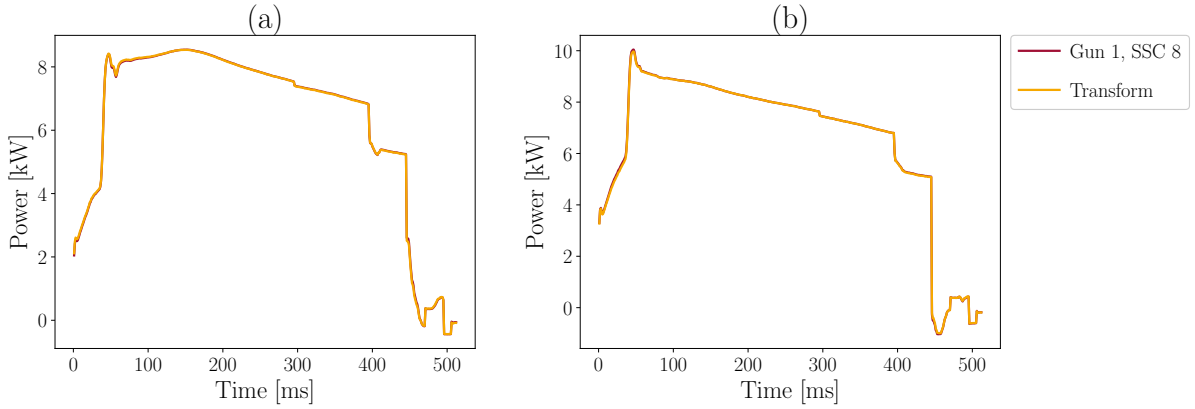

Figure 4.7: Reconstructed curves.

Additionally, three further transformations are investigated, exploring the condition space of welding guns and SCC of the sample application. The results are depicted in Fig. A.15, Fig. A.16 and Fig. A.17 in Appendix A.2.3. It can be seen that the previous findings for the condition transformation hold in general true for the sample application. An in-depth investigation in

hyper-modeling, generalizing the influence of conditions on the processes, which is revealed with the above methods, will be subject of future work.

4.3 Discussion

By reducing the representation space of process curves to a low dimensionality, the process curve space can be covered by smaller samples and the data efficiency be increased. As a positive side effect, visualization of the condition dependencies is possible as well. The evaluation results show that a low-dimensional, generalized feature space can be found via CNN-architectures. Those features do not only allow a sufficient reconstruction of the welding curves, but also bear the information on the process conditions and on the process result. This was proven by the classification quality with respect to the process conditions of the welding gun and SCC and the estimation quality, all of them based on the generalized features. The resulting feature space is not only universal with respect to reconstruction, process conditions and process result, but also low-dimensional, as required.

The training of CNN structures with a given number of latent features, was in some cases switching off some of the latent feature neurons by constantly setting their outputs to zero. This means that the feature space dimension was effectively smaller than the size after training. Some CNN structures with the number of latent features as three were setting one latent feature neuron to zero, thus creating an only two-dimensional latent space, which makes them especially well suited for studies including visualization. Therefore they were used in the subsequent latent feature space transformation despite their lower performance compared to CNN structures with higher latent feature space dimensions.

Going back to the transformed densities, all quantities, which can be derived from a density function can be transformed as well. This could be the average process curve, which is frequently used in quality measurements, but also the variance of process curves, which is frequently used as a bound for process controls. Any higher-order statistical moment, which has importance for control or quality assessment can also be derived.

Chapter 5

Multitask learning in materials design

The demand for more and more specific and individually designed products with certain performance requirements has become a driving force in the world of manufacturing. For this reason, the optimization along the process–structure–property (PSP) linkage (Olson, 1997) became a fast growing research topic in the field of Integrated Computational Materials Engineering (ICME) (Panchal et al., 2013). Nowadays, such optimization problems can be solved efficiently with the help of machine learning (ML) techniques (Ramprasad et al., 2017). Building on this background, a previous study investigated the use of reinforcement learning to find optimal processing routes in a simulated metal forming process, aiming to produce microstructures with targeted crystallographic textures. (Dornheim et al., 2022). To bridge the remaining gap between microstructures and desired properties, this study focuses on solving materials design problems. These are to identify appropriate material microstructures or microstructural features (e.g. the crystallographic texture) for given desired properties. It is thereby of particular importance to identify sets of near-optimal and preferably diverse microstructures in order to guarantee a robust design (McDowell, 2007).

5.1 Methods

5.1.1 Materials design via siamese multitask learning and optimization

General concept

First of all, the concept of the multitask learning (MTL)-based optimization approach is presented. The general architecture for obtaining a latent feature space is introduced in Section 2.2 and depicted in Fig. 2.10. The approach can be applied to general materials design problems and starts by defining the desired properties and corresponding tolerances. This in turn defines a target region, for which the approach is supposed to identify a diverse set of microstructures. The approach is schematically depicted in Fig. 5.1 and basically consists of three components: optimizer, microstructure-property mapping (m-p-m) and validity-prediction (v-p). The optimizer generates candidate microstructures that minimize the combined costs, which result from evaluations based on the m-p-m and v-p components.

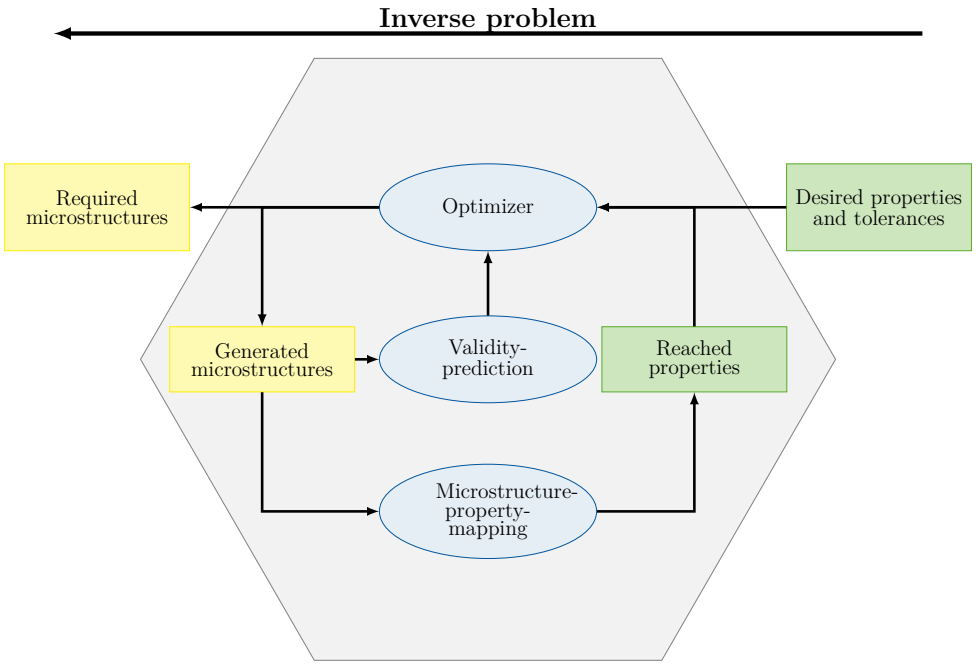

Figure 5.1: General concept of the multitask learning based optimization approach to solve materials design problems.

The *m-p-m* component assigns properties to a candidate microstructure. The deviation of the assigned properties to the target region determines the cost. In general, the *m-p-m* component can be realized by a numerical simulation. However, since numerical simulations are computationally expensive, a surrogate model is used instead. The surrogate model is realized by a regression model that learns the relations from a priori generated microstructure-property data.

The *v-p* component is realized by an anomaly detection method which determines the validity of a candidate microstructure by comparing it to the set of valid microstructures. The concept of the anomaly detection is illustrated in Fig. 5.2. The *v-p* component returns a value that can be seen as an estimate of a candidate microstructure being an element of the microstructure set under consideration. This is for example the set, which can be produced by a dedicated process (e.g. rolling). The value returned by the *v-p* component defines the validity cost and is supposed to drive the optimizer solution to a valid microstructure region. Besides, such a microstructure region can alternatively be identified by a further optimization loop that interacts with a numerical simulation of the dedicated process, which, however, suffers from high computational costs.

The two components *m-p-m* and *v-p* can be realized by training two separate ML models. However, when the training procedures are isolated from each other, the models are not able to mutually access information already learned by the other model. Therefore, the two components are combined as tasks into one MTL model (Caruana, 1997). Both tasks have a common backbone (the feature extraction part of a network) and different heads (feature processing part of a network) operating on the backbone output. The backbone output vectors form the so-called latent feature space.

The proposed MTL approach furthermore uses the backbone as an encoder network of an autoencoder, where the decoder is also attached to the latent feature space with the purpose to

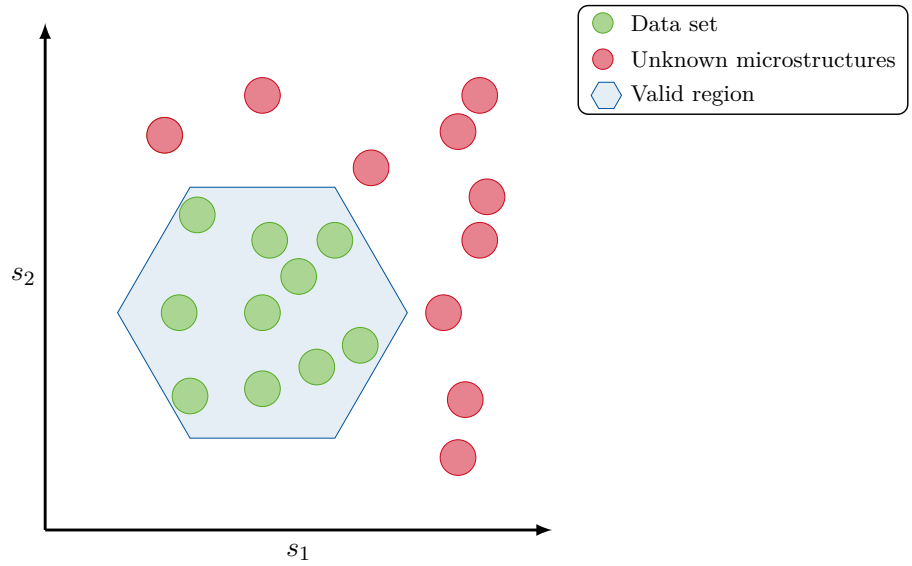

Figure 5.2: Schematic illustration of the structure space s_1, s_2 including the valid region and unknown microstructures.

reconstruct the input pattern of the backbone. This is achieved by adding the reconstruction of the microstructures from the latent feature space as a third task. In the MTL approach, all the three tasks are represented by a single neural network-based model. The neural weights of the model are trained simultaneously based on a combined loss function. After training the MTL model, the optimizer can operate very efficiently in the lower dimensional latent feature space.

However, it has some limitations, which are mentioned in the following. Since the concept is based on a data-driven modeling and optimization approach, an adequate data set is required. The described components which are learned within the concept are approximations of the numerical simulations and are accordingly not equally exact. The model quality of the components depends on the size and quality of the underlying data set. Therefore, the application of an efficient sampling strategy for exploring the microstructure and property space can be suitable (Morand et al., 2022). However, under the assumption of low model errors, the components can be efficiently used as surrogate models in the application of the concept (except for extrapolation).

The remainder of this section presents the optimization approach and the MTL approach in detail, as well as an extension based on Siamese neural networks (Bromley et al., 1993) to enforce the representation of microstructures in the latent feature space to preserve the microstructure distances in the original representation space.

Multitask learning

The MTL model, as shown in Fig. 5.3, is optimized on the dataset $\mathcal{D}_{\text{ohg}}$ with pairs of microstructures $\boldsymbol{x}_{\text{ohg}} \in \mathbb{R}^D$ and corresponding properties $\boldsymbol{y}_{\text{est}_{\text{prp}}} \in \mathbb{R}^P$. The input microstructures are transformed into latent features $\boldsymbol{z}_{\text{ohg}} \in \mathbb{R}^Z$. The individual outputs of the connected tasks are the estimated properties $\hat{\boldsymbol{y}}_{\text{est}_{\text{prp}}} \in \mathbb{R}^P$, the reconstructed microstructure $\hat{\boldsymbol{x}}_{\text{ohg}} \in \mathbb{R}^D$ and the reconstructed latent features $\hat{\boldsymbol{z}}_{\text{ohg}} \in \mathbb{R}^Z$. The following introduces the information processing scheme of the MTL model in detail.

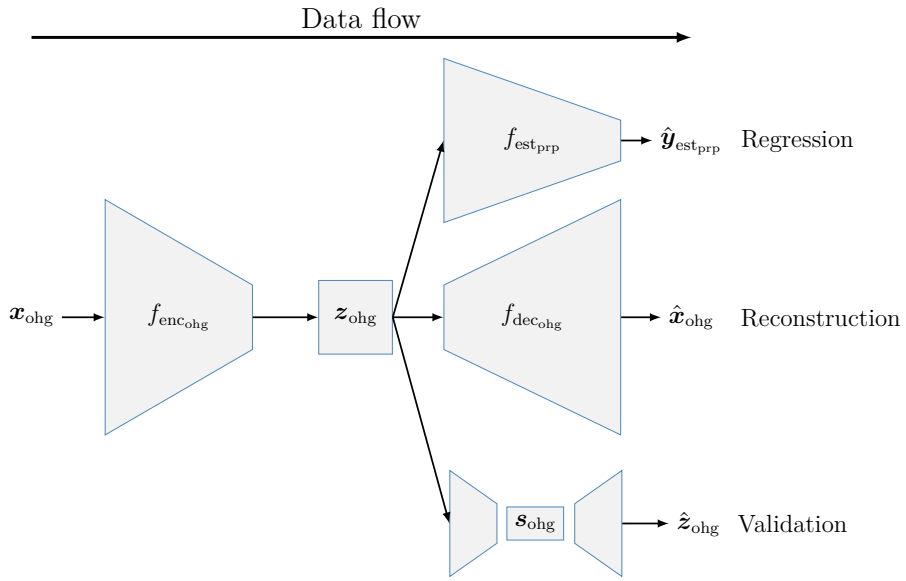

Figure 5.3: Multitask learning architecture for orientation histograms.

The processing scheme starts with an encoder network $f_{\text{enc}_{\text{ohg}}}$ which extracts significant features by mapping the microstructure space $\boldsymbol{x}_{\text{ohg}}$ into a lower dimensional latent feature space $\boldsymbol{z}_{\text{ohg}}$ via the learned function

$$\boldsymbol{z}_{\text{ohg}} = f_{\text{enc}_{\text{ohg}}}(\boldsymbol{x}_{\text{ohg}}, \Theta_{\text{enc}_{\text{ohg}}}), \tag{5.1}$$

in which the encoder network is parameterized by its weight values $\Theta_{\text{enc}_{\text{ohg}}}$. All three previously described tasks are attached to the encoder in the form of feedforward neural networks. Besides, the encoder can be easily adapted to higher dimensional microstructure representing data types, like images (Electron Backscatter Diffraction (EBSD) or micrograph images) or three dimensional microstructure data by using convolutional neural networks (CNNs) (see (Krizhevsky et al., 2012)). The latter is used in (Cecen et al., 2018) in the materials sciences domain for example.

To train the MTL model, a loss function that combines all the three tasks is needed. This is achieved by a function $\mathcal{J}$ (see Eq. (2.31)) that cumulates the loss terms of the three tasks $\mathcal{J}_{\text{est}_{\text{prp}}}$ (regression loss), $\mathcal{J}_{\text{dec}_{\text{ohg}}}$ (decoder loss) and $\mathcal{J}_{\text{est}_{\text{val}}}$ (validation loss), and weights them using $\lambda_{\text{est}_{\text{prp}}}$, $\lambda_{\text{dec}_{\text{ohg}}}$ and $\lambda_{\text{est}_{\text{val}}}$ to allow for prioritization. The total loss function is defined as

$$\mathcal{J}_{\text{MTL}} = \lambda_{\text{est}_{\text{prp}}}\mathcal{J}_{\text{est}_{\text{prp}}} + \lambda_{\text{dec}_{\text{ohg}}}\mathcal{J}_{\text{dec}_{\text{ohg}}} + \lambda_{\text{est}_{\text{val}}}\mathcal{J}_{\text{est}_{\text{val}}} + \gamma\mathcal{R}(\Theta_{\text{ohg}}), \tag{5.2}$$

where $\mathcal{R}(\Theta_{\text{ohg}})$ is a regularization term that is used to prevent overfitting with the hyperparameter γ defining the strength of the regularization (also known as weight decay, see (Krogh and Hertz, 1991) and (Hinton, 1987)). Each of the feedforward neural networks is parameterized by the respective weight values $\Theta_{\text{enc}_{\text{ohg}}}$, $\Theta_{\text{prp}_{\text{ohg}}}$, $\Theta_{\text{dec}_{\text{ohg}}}$ and $\Theta_{\text{val}_{\text{ohg}}}$, which are adjusted simultaneously during training and altogether form the weight vector Θ_{ohg}. The following introduces the three individual loss terms.

1. The decoder network, which is responsible for the reconstruction, transforms the latent feature vectors $\boldsymbol{z}_{\text{ohg}}$ back to the original microstructure space:

$$\hat{\boldsymbol{x}}_{\text{ohg}} = f_{\text{dec}_{\text{ohg}}}(\boldsymbol{z}_{\text{ohg}}, \Theta_{\text{dec}_{\text{ohg}}}) = f_{\text{dec}_{\text{ohg}}}(f_{\text{enc}_{\text{ohg}}}(\boldsymbol{x}_{\text{ohg}}, \Theta_{\text{enc}_{\text{ohg}}}), \Theta_{\text{dec}_{\text{ohg}}}). \tag{5.3}$$

The reconstruction loss is defined on the basis of a distance measure between two microstructural feature vectors $\mathscr{D}(\boldsymbol{x}_{\text{ohg}}, \hat{\boldsymbol{x}}_{\text{ohg}})$:

$$\mathcal{J}_{\text{dec}_{\text{ohg}}}(\boldsymbol{x}_{\text{ohg}}, \hat{\boldsymbol{x}}_{\text{ohg}}) = \mathscr{D}(\boldsymbol{x}_{\text{ohg}}, \hat{\boldsymbol{x}}_{\text{ohg}}). \tag{5.4}$$

The distance measure between depends on the microstructure representation and has to be chosen appropriately.

2. The forward mapping of the latent feature vector $\boldsymbol{z}_{\text{ohg}}$ to the properties vector $\hat{\boldsymbol{y}}_{\text{est}_{\text{prp}}}$ is represented by the learned function

$$\hat{\boldsymbol{y}}_{\text{est}_{\text{prp}}} = f_{\text{est}_{\text{prp}}}(\boldsymbol{z}_{\text{ohg}}, \Theta_{\text{prp}_{\text{ohg}}}) = f_{\text{est}_{\text{prp}}}(f_{\text{enc}_{\text{ohg}}}(\boldsymbol{x}_{\text{ohg}}, \Theta_{\text{enc}_{\text{ohg}}}), \Theta_{\text{prp}_{\text{ohg}}}). \tag{5.5}$$

The regression loss is given by the MSE between true properties $\boldsymbol{y}_{\text{est}_{\text{prp}}}$ and the predicted properties $\hat{\boldsymbol{y}}_{\text{est}_{\text{prp}}}$:

$$\mathcal{J}_{\text{est}_{\text{prp}}}(\boldsymbol{y}_{\text{est}_{\text{prp}}}, \hat{\boldsymbol{y}}_{\text{est}_{\text{prp}}}) = \frac{1}{P}\sum_{i=1}^{P}(\text{y}_{\text{est}_{\text{prp}_i}} - \hat{\text{y}}_{\text{est}_{\text{prp}_i}})^2, \tag{5.6}$$

where P denotes the number of properties.

3. On the basis of the latent feature space, an autoencoder network is set up transforming $\boldsymbol{z}_{\text{ohg}}$ into an even lower-dimensional feature sub-space $\boldsymbol{s}_{\text{ohg}} \in \mathbb{R}^{\text{S}}$ with $\text{S} < \text{Z}$ and transforming back to $\hat{\boldsymbol{z}}_{\text{ohg}}$ via

$$\hat{\boldsymbol{z}}_{\text{ohg}} = f_{\text{est}_{\text{val}}}(\boldsymbol{z}_{\text{ohg}}, \Theta_{\text{val}_{\text{ohg}}}) = f_{\text{est}_{\text{val}}}(f_{\text{enc}_{\text{ohg}}}(\boldsymbol{x}_{\text{ohg}}, \Theta_{\text{enc}_{\text{ohg}}}), \Theta_{\text{val}_{\text{ohg}}}). \tag{5.7}$$

The validity loss is defined by the MSE between $\boldsymbol{z}_{\text{ohg}}$ and $\hat{\boldsymbol{z}}_{\text{ohg}}$:

$$\mathcal{J}_{\text{est}_{\text{val}}}(\boldsymbol{z}_{\text{ohg}}, \hat{\boldsymbol{z}}_{\text{ohg}}) = \frac{1}{Z}\sum_{i=1}^{Z}(\text{z}_{\text{ohg}_i} - \hat{\text{z}}_{\text{ohg}_i})^2 \tag{5.8}$$

with Z dimensions of the latent feature space.

Distance preserving feature extraction using siamese neural networks

The above described MTL approach is used in combination with an optimizer that searches for candidate microstructures with desired properties in the latent feature space. However, the approach aims to identify a diverse set of microstructures with high diversity. For the diversity quantification a distance measure in the latent feature space is required. The MTL approach as defined above, is not able to preserve the distances of the original space in the latent feature space. To construct a distance preserving latent feature space, the MTL model is embedded in a Siamese neural network (Bromley et al., 1993; Chicco, 2021), which is described next.

Siamese neural networks consist of two identical networks, which share weights in the encoder part, see Fig. 5.4. Both networks embed different microstructures $\boldsymbol{x}_{L,\text{ohg}}$ and $\boldsymbol{x}_{R,\text{ohg}}$ as $\boldsymbol{z}_{L,\text{ohg}}$ and $\boldsymbol{z}_{R,\text{ohg}}$ in the latent feature space which is finally processed by two identical MTL networks. The distance preservation is enforced by defining a distance preservation loss $\mathcal{J}_{\text{prs}}$ that minimizes the difference between the distance of two different input microstructures in the original space

$\mathscr{D}(\boldsymbol{x}_{L,\mathrm{ohg}}, \boldsymbol{x}_{R,\mathrm{ohg}})$ and the corresponding distance in the latent feature space $\mathscr{D}(\boldsymbol{z}_{L,\mathrm{ohg}}, \boldsymbol{z}_{R,\mathrm{ohg}})$, with $\mathscr{D}(\boldsymbol{x}_{L,\mathrm{ohg}}) \neq \mathscr{D}(\boldsymbol{x}_{R,\mathrm{ohg}})$ (Utkin et al., 2017):

$$\mathcal{J}_{\mathrm{prs}} = \frac{1}{N}\Big(\mathscr{D}(\boldsymbol{x}_{L,\mathrm{ohg}}, \boldsymbol{x}_{R,\mathrm{ohg}}) - \mathscr{D}(\boldsymbol{z}_{L,\mathrm{ohg}}, \boldsymbol{z}_{R,\mathrm{ohg}})\Big)^2, \tag{5.9}$$

while N indicates the number of samples in the data set and the two distance measures $\mathscr{D}(\boldsymbol{x}_{L,\mathrm{ohg}}, \boldsymbol{x}_{R,\mathrm{ohg}})$ and $\mathscr{D}(\boldsymbol{z}_{L,\mathrm{ohg}}, \boldsymbol{z}_{R,\mathrm{ohg}})$ are not necessarily the same. Applying such loss terms leads to multi dimensional scaling, see (Kruskal, 1964) and (Cox and Cox, 2008). Using the distance preservation loss $\mathcal{J}_{\mathrm{prs}}$, the MTL loss function, defined in Eq. (5.2), is extended by the weighted preservation loss $\lambda_{\mathrm{prs}}\mathcal{J}_{\mathrm{prs}}$ to

$$\mathcal{J}_{\mathrm{SMTL}} = \lambda_{\mathrm{est_{prp}}}\mathcal{J}_{\mathrm{est_{prp}}} + \lambda_{\mathrm{dec_{ohg}}}\mathcal{J}_{\mathrm{dec_{ohg}}} + \lambda_{\mathrm{est_{val}}}\mathcal{J}_{\mathrm{est_{val}}} + \lambda_{\mathrm{prs}}\mathcal{J}_{\mathrm{prs}} + \gamma\mathcal{R}(\Theta). \tag{5.10}$$

The Siamese multitask learning (SMTL) approach delivers a function which can map a microstructure representation in the latent feature space on properties. Now, an optimizer can operate on a lower dimensional feature space to find microstructures with desired properties. The SMTL framework also allows to reconstruct the original represenation of microstructures, to asses the distances between them and to validate them in the latent feature space.

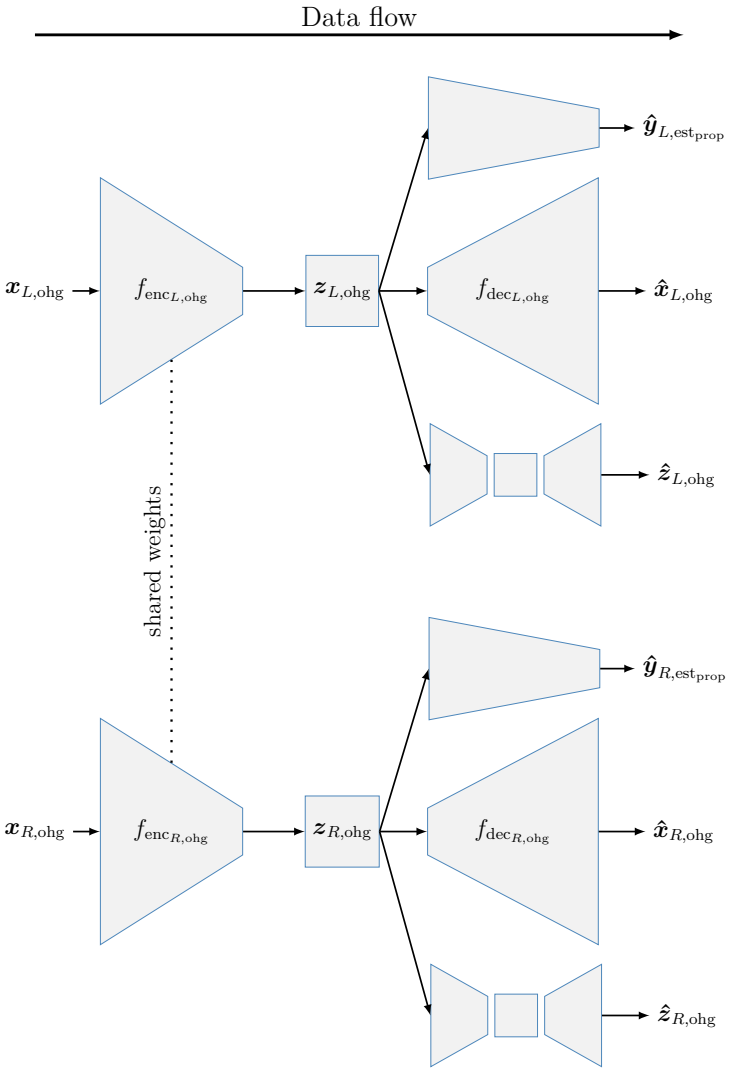

Figure 5.4: Architecture of the Siamese multitask learning approach. The dotted line between the encoders E_{L} and E_{R} indicates shared weights.

Microstructure optimizer

The microstructure optimization with respect to desired properties uses the distance preserving SMTL framework with the tasks microstructure-property mapping, validity-prediction and reconstruction. The optimization minimizes a cost function $\mathcal{F}$, which consists of the cost terms $\mathcal{F}_{\text{prp}}$, $\mathcal{F}_{\text{val}}$ and $\mathcal{F}_{\text{div}}$ and the corresponding weights ω_{prp}, ω_{val} and ω_{div}:

$$\mathcal{F} = \omega_{\text{prp}}\mathcal{F}_{\text{prp}} + \omega_{\text{val}}\mathcal{F}_{\text{val}} + \omega_{\text{div}}(1 + \mathcal{F}_{\text{div}}). \tag{5.11}$$

$\mathcal{F}_{\text{prp}}$, $\mathcal{F}_{\text{val}}$ and $\mathcal{F}_{\text{div}}$ denote the property, validity and diversity cost terms, respectively. While the property cost term drives the candidate microstructures to lie inside a specified target properties region, the validity cost aims that the optimizer operates inside the region of valid microstructures and the diversity cost ensures that candidate microstructures differ from each other. To minimize the loss function genetic algorithms are used, which generate a population set of $\mathcal{P}$ candidate microstructures $\boldsymbol{z}^*_{\text{ohg}}$ in the latent feature space in every iteration. The three cost terms are described in more detail in the following.

1. The property cost is defined by the MSE between the desired properties and the predicted properties from the SMTL regression model:

$$\mathcal{F}_{\text{prp}} = \frac{1}{P}\sum_{i=1}^{P}(\widetilde{\mathcal{F}}_{\text{prp}_i})^2. \tag{5.12}$$

   If one of the predicted properties lies inside the target region, the cost $\widetilde{\mathcal{F}}_{\text{prp}_i}$ equals 0. Otherwise, $\widetilde{\mathcal{F}}_{\text{prp}_i}$ equals the minimum squared distance from the predicted properties to the target region borders.

2. The validity prediction is used to asses whether an identified candidate microstructure is likely to be represented by the sample data set. The validity cost is defined by

$$\mathcal{F}_{\text{val}} = \max(\mathscr{A} - \xi_{\text{val}}, 0), \tag{5.13}$$

   in which ξ_{val} is a threshold to define the maximum tolerated reconstruction error for valid textures and $\mathscr{A}$ denotes the anomaly score

$$\mathscr{A} = \frac{1}{Z}\sum_{i=1}^{Z}(z^*_{\text{ohg}_i} - \hat{z}^*_{\text{ohg}_i})^2. \tag{5.14}$$

3. The diversity cost is based on the sum of the distances between the candidate microstructure $\boldsymbol{z}^*_{\text{ohg}}$ in the latent feature space and every other microstructure in the population $\mathcal{P}$:

$$\mathcal{F}_{\text{div}} = -\sum_{i=1}^{\mathcal{P}}\mathscr{D}(\boldsymbol{z}^*_{\text{ohg}_i}, \boldsymbol{z}^*_{\text{ohg}}), \tag{5.15}$$

   in which for $\mathscr{D}(\boldsymbol{z}^*_{\text{ohg}_i}, \boldsymbol{z}^*_{\text{ohg}})$ the same distance measure has to be used as for the latent feature vectors in Eq. (5.9).

5.1.2 Materials science fundamentals

Crystallographic texture of steel sheets

For the representation of crystallographic texture, 512 nearly uniform distributed orientations are sampled over the cubic fundamental zone, and a soft assignment of $l = 3$ (c.f. see Section 2.3.1) is chosen.

After rolling body centered cubic (bcc) materials, typically so-called fiber textures are formed. Following (Ray et al., 1994), these textures are composed of the five fibers α, γ, η, ϵ, and β, which are defined in detail in Tab. 5.1. Among these fibers, the α and γ fiber are most prominent (Kocks et al., 1998), whereas the presence of the β fiber is only reported from theoretical predictions (Schlippenbach et al., 1986). To provide an idea of how the fibers affect forming properties, reference is made to (Ray et al., 1994). Therein, it is found out that the γ fiber causes good deep drawability and the α fiber has a contrary effect.

Table 5.1: Definition of the fibers of bcc rolling textures following (Kocks et al., 1998).

Fiber	Location
α	from {001}<110> to {111}<1$\bar{1}$0>, parallel to RD
γ	from {111}<1$\bar{1}$0> to {111}<112>, parallel to ND
η	from {001}<100> to {011}<100>, parallel to RD
ϵ	from {001}<110> to {111}<112>, parallel to TD
β	from {112}<1$\bar{1}$0> to {$\overline{11}$ $\overline{11}$ 8}<4 4 $\overline{11}$ >

In order to generate a data base of (artificial) rolling textures, in this work, a 25-parameter model is used, as it is proposed in (Delannay et al., 1999) to describe steel sheet textures. The model is based on textures that are composed of the fibers α, γ, and η. Since the η-fiber is not always present in steel sheet textures, the focus is limited to textures consisting of an α and γ fiber. Therefore, 6 of the 25 parameters can be neglected.

The texture model describes the orientation distribution function (ODF) as a set of weighted Gaussian distributions placed along the fibers. The model parameters D_i are listed in Tab. 5.2 and define the standard deviations and the mean values of the distributions based on the fiber thickness and the shifts from their ideal positions. Furthermore, the model parameters define the weights of the distributions among each other based on the probability given by the ODF, referred to as fiber intensity in the following.

To construct the set of Gaussian distributions, the seven base distributions from Tab. 5.2 are placed at their ideal positions with respect to the shifts. Between these seven distributions, further distributions are placed with a distance of about 3° to each other, leading to overall 41 Gaussians. Their weights w_i and the values for the standard deviation σ_i and mean value μ_i are interpolated linearly based on the values of the two neighboring base distributions. This yields a set of Gaussian distributions $\mathcal{N}_1(\mu_1, \sigma_1), ..., \mathcal{N}_{41}(\mu_{41}, \sigma_{41})$. The ODF $f(g)$ is defined by the normalized sum of this set:

$$f(g) = \frac{1}{\sum_i w_i} \sum_{i=1}^{n} w_i \mathcal{N}_i(\mu_i, \sigma_i). \tag{5.16}$$

Table 5.2: Definition of the parameters D_i of the texture model, cf. (Delannay et al., 1999). The ideal position is given in Bunge Euler angles [°].

	Ideal pos.	Intens.	Std φ_2	Std φ_1	Shift φ_1	Shift ϕ	Shift φ_2
a_1	$0, 0, 45$	D_1	D_7	D_7	0	0	0
a_2	$0, 30, 45$	D_2	D_8	D_{13}	0	D_{15}	0
a_3	$0, 55, 45$	D_3	D_9	D_9	0	D_{16}	0
a_4	$0, 70, 45$	D_4	D_{10}	D_{10}	0	D_{17}	0
a_5	$0, 90, 45$	D_4	D_{10}	D_{10}	0	0	0
g_2	$15, 55, 45$	D_5	D_{11}	D_{11}	D_{14}	0	0
g_3	$30, 55, 45$	D_6	D_{12}	D_{12}	0	D_{18}	D_{19}

Based on this definition, discrete orientations can be sampled. In the following, the set of orientations is denoted as G. As $f(g)$ is defined in the cubic-orthorhombic fundamental zone, it is necessary to add the equivalent orientations regarding the orthorhombic sample symmetry to the set of discrete orientations. This is done by applying rotation operations g_s on each orientation g_i in G

$$g_i^{\text{equiv}} = g_s \; g_i. \tag{5.17}$$

The rotation operations g_s for orthorhombic sample symmetry can be found in (Hansen et al., 1978). Rolling textures, for example, underlie a cubic crystal and an orthorhombic sample symmetry, for which 96 elementary regions exist (Hansen et al., 1978).

Material model

The sheet metal properties focused on in this study are the Young's moduli and the R-values at 0, 45 and 90 degree to rolling direction, resulting in the six properties: E_{00}, E_{45}, E_{90}, R_{00}, R_{45}, R_{90}. In this study, the properties are calculated by applying uniaxial tension on a crystal plasticity-based material model. As time efficiency is essential for the generation of data, a material model of Taylor-type is implemented, as it is described in (Dornheim et al., 2022). A detailed description of the material model is provided in Section 2.3.2. The material dependent parameters that are used in this study are summarized in Tab. 5.3.

Table 5.3: Material dependent parameters used in this study (cf. (Dornheim et al., 2022)).

parameter	C_{11}	C_{12}	C_{44}	$\dot{\gamma}_0$	m	τ_0	τ_1	ϑ_0	ϑ_1	q_1	q_2
value	218	131	105	0.001	0.02	94	50	258	32	1.4	1.4
unit	GPa	GPa	GPa	s^{-1}	-	MPa	MPa	MPa	MPa	-	-

5.2 Results

5.2.1 Texture-property data set

For the training data set $\mathcal{D}_{\text{ohg}}^{\text{train}}$, $50,000$ sets of $2,000$ discrete orientations are sampled via Latin hypercube design (McKay et al., 1979), based on Eq. (5.16). In order to have an independent test data set $\mathcal{D}_{\text{ohg}}^{\text{test}}$, further $10,000$ sets are generated randomly. The ranges inside which the parameters of the texture model vary are adjusted manually such that typical bcc rolling textures

found in literature (Hölscher et al., 1991; Inagaki and Suda, 1972; Klinkenberg et al., 1993; Kocks et al., 1998; Kestens and Pirgazi, 2016; Pagenkopf et al., 2016; Das, 2017) are covered. The parameter ranges are listed in Tab. 5.4. In addition, to evaluate the anomaly detection, a set of artificial textures is needed, which slightly differ from the generated rolling textures. For this purpose, the data set $\mathcal{D}_{\text{ohg}}^{\text{anom}}$ with 10,000 anomalies are generated by shifting the α-fiber (i.e. the ideal position of a_1, a_2, a_4 and a_5) about 20 degrees in φ_1-direction.

Table 5.4: Parameter ranges for D_i.

Intensity	D_1	D_2	D_3	D_4	D_5	D_6	
min	1/6	1/6	1/6	0	1/6	1/6	
max	1/3	2/3	2/3	1/3	1	1	
Std	D_7	D_8	D_9	D_{10}	D_{11}	D_{12}	D_{13}
min	5/3	5/3	5/3	5/3	5/3	5/3	5/3
max	45/3	35/3	35/3	30/3	30/3	30/3	35/3
Shift	D_{14}	D_{15}	D_{16}	D_{17}	D_{18}	D_{19}	
min	−5	−10	−10	−5	−10	−10	
max	10	10	5	10	10	10	

Moreover, the texture-property mapping and the validity-prediction are validated using experimental data. For this purpose, an experimentally measured texture of cold rolled DC04 steel from (Schreijäg, 2013) is used. Based on this measurement, an ODF is approximated via the MATLAB toolbox mtex (Bachmann et al., 2010), rotated into its symmetry axis assuming orthorhombic sample symmetry and mirrored. To visualize the α- and γ-fiber of the orientation distribution, an intersection plot of the Euler space at $\varphi_2 = 45°$ is depicted in Fig. 5.5.

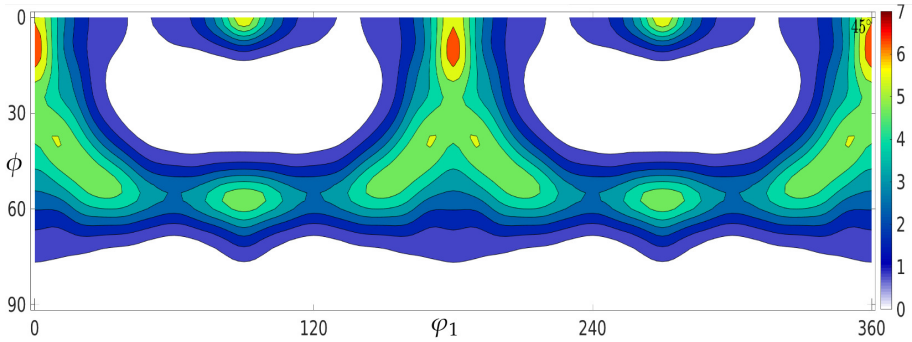

Figure 5.5: $\varphi_2 = 45°$ section of the orientation distribution function to visualize the γ-fiber of the cold rolled DC04 steel texture.

For the generated textures in the data sets $\mathcal{D}_{\text{ohg}}^{\text{train}}$ and $\mathcal{D}_{\text{ohg}}^{\text{test}}$, the corresponding Young's moduli and R-values in 0, 45, and 90 degree to rolling direction are determined using the Taylor-type crystal plasticity model described in Section 5.1.2. Both quantities, Young's modulus and especially R-values, are highly affected by the crystallographic texture, which is why these are chosen exemplary for the purpose of this study.

5.2.2 Validation of the siamese multitask learning model

In this study, the individual tasks of the SMTL model are realized via feedforward neural networks with *tanh* activation functions (c.f. Eq.(2.10)) to obtain features between -1 and $+1$ in the latent feature space. The SMTL model is implemented based on the Python TensorFlow API (Abadi et al., 2021). The base architecture of the Siamese neural network is illustrated in Fig. A.18. The Glorot Normal method (Glorot and Bengio, 2010) is used for weight initialization. In order to adjust the hyperparameters, a random search method (Bergstra and Bengio, 2012) is applied using 5-fold cross-validation. The best model configuration that was found is shown in Tab. 5.5.

Table 5.5: Used hyperparamters for training the Siamese multitask learning model.

Hyperparameter	Value
Optimizer	Adam (Kingma and Ba, 2015)
Learning rate	0.001
Weight decay	1e-6
Batch size	256

The Chi-Squared distance $\mathscr{D}_{\chi^2}$ introduced in Eq. (2.42) is used as distance measure in the input space. In the latent feature space, the sum of squared errors (SSE) between two vectors $\boldsymbol{z}_{\text{ohg}}^{(1)}$ and $\boldsymbol{z}_{\text{ohg}}^{(2)}$ is used as the distance measure

$$\mathscr{D}_{\text{SSE}}(\boldsymbol{z}_{\text{ohg}}^{(1)}, \boldsymbol{z}_{\text{ohg}}^{(2)}) = \sum_{i=1}^{Z} (z_{\text{ohg}_i}^{(1)} - z_{\text{ohg}_i}^{(2)})^2. \tag{5.18}$$

The SMTL model is trained for 200 epochs, while the best intermediate result of the test set is retained, which can be interpreted as early stopping (Prechelt, 1998). Before the model training is executed, the loss terms are scaled to values between 0 and 1 in order to make them comparable. The following weights for the scaled loss terms were based on hyper parameter optimization: $\lambda_{\text{est}_{\text{prp}}} = 0.05$, $\lambda_{\text{dec}_{f_{\boldsymbol{b}}(g)}} = 0.05$, $\lambda_{\text{est}_{\text{val}}} = 0.05$ and $\lambda_{\text{prs}} = 0.85$.

The results for the texture-property mapping and the distance preservation are shown in Tab. 5.6, in which the regression errors MAE_{E} and MAE_{R} denote the mean absolute error (MAE) between the true and predicted Young's moduli and R-values depending on the dimension of the latent feature space $\boldsymbol{z}_{\text{ohg}}$. The quality of the distance preservation is measured by the coefficient of determination R^2, between the distances of two input textures and their corresponding latent feature vectors

$$R^2\Big(\mathscr{D}_{\chi^2}(\boldsymbol{x}_{L,\text{ohg}}, \boldsymbol{x}_{L,\text{ohg}}), \mathscr{D}_{\text{SSE}}(\boldsymbol{z}_{L,\text{ohg}}, \boldsymbol{z}_{R,\text{ohg}})\Big). \tag{5.19}$$

It is shown that texture-property mappings with an adequate prediction quality can be achieved by extensively reducing the dimensionality of the latent feature space. However, regarding the distance preservation quality, a lower bound of at least 10 latent features can be identified, below which the distance preservation is unsatisfactory. Additionally, the texture-property mapping is evaluated on the experimentally measured texture and the corresponding properties. The results are listed in Tab. 5.6. It can be seen that a satisfactory prediction quality (Regr. $\text{MAE}_{\text{E}} \leq 1000$ MPa and Regr. $\text{MAE}_{\text{R}} \leq 0.1$) can only be achieved for at least 16 latent features.

Table 5.6: Results for varying numbers of latent features (LF) of the texture-property mapping and the distance preservation applied to the artificially generated textures and experimentally measured texture. Regr.(ession error) MAE_E is given in [MPa], Regr.(ession error) MAE_R in [-] and Pres.(erve quality) R^2 in [%].

LF	Artificial textures			Experimental texture	
	Regr. MAE_E	Regr. MAE_R	Pres. R^2	Regr. MAE_E	Regr. MAE_R
20	153	0.03	98.2	596	0.04
18	183	0.03	98.0	773	0.11
16	162	0.03	97.8	907	0.05
14	193	0.03	97.0	1282	0.07
12	215	0.04	95.4	1468	0.13
10	238	0.05	92.2	1463	0.13
8	335	0.06	85.7	1575	0.17
6	390	0.07	72.4	1554	0.15
4	664	0.10	34.2	2768	0.12

On the basis of this 16-dimensional feature space, the validity-prediction is evaluated. The anomaly scores for the textures in the test set and for the artificially generated anomalies are shown in Fig. 5.6. It can be seen, that the anomalies can be separated in a sufficient manner from the textures in the test set.

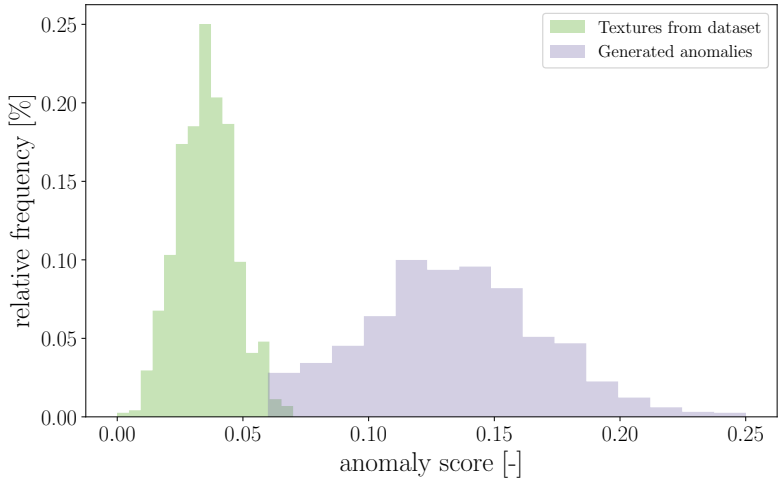

Figure 5.6: Histograms of the anomaly scores for the data from the test set and the set of artificially generated anomalies. The anomaly scores are based on the model that uses 16 latent features.

5.2.3 Rolling texture identification

To validate the texture identification, two target regions are defined in property space, see Fig. 5.7. The first one is defined by the properties of the experimentally measured texture, which lies in a sparsely populated region and is labeled as *Target Region 1*. As a consequence of its location in the sparsely populated region, the anomaly score of this texture is 0.0099 and lies in the transition zone shifted towards the generated anomalies (cf. Fig. 5.6). It is of interest if the optimizer is generally able to find a whole set of microstructures with properties in this region.

The second target region represents a densely populated region located near the center of the properties point cloud and is labeled as *Target Region 2*.

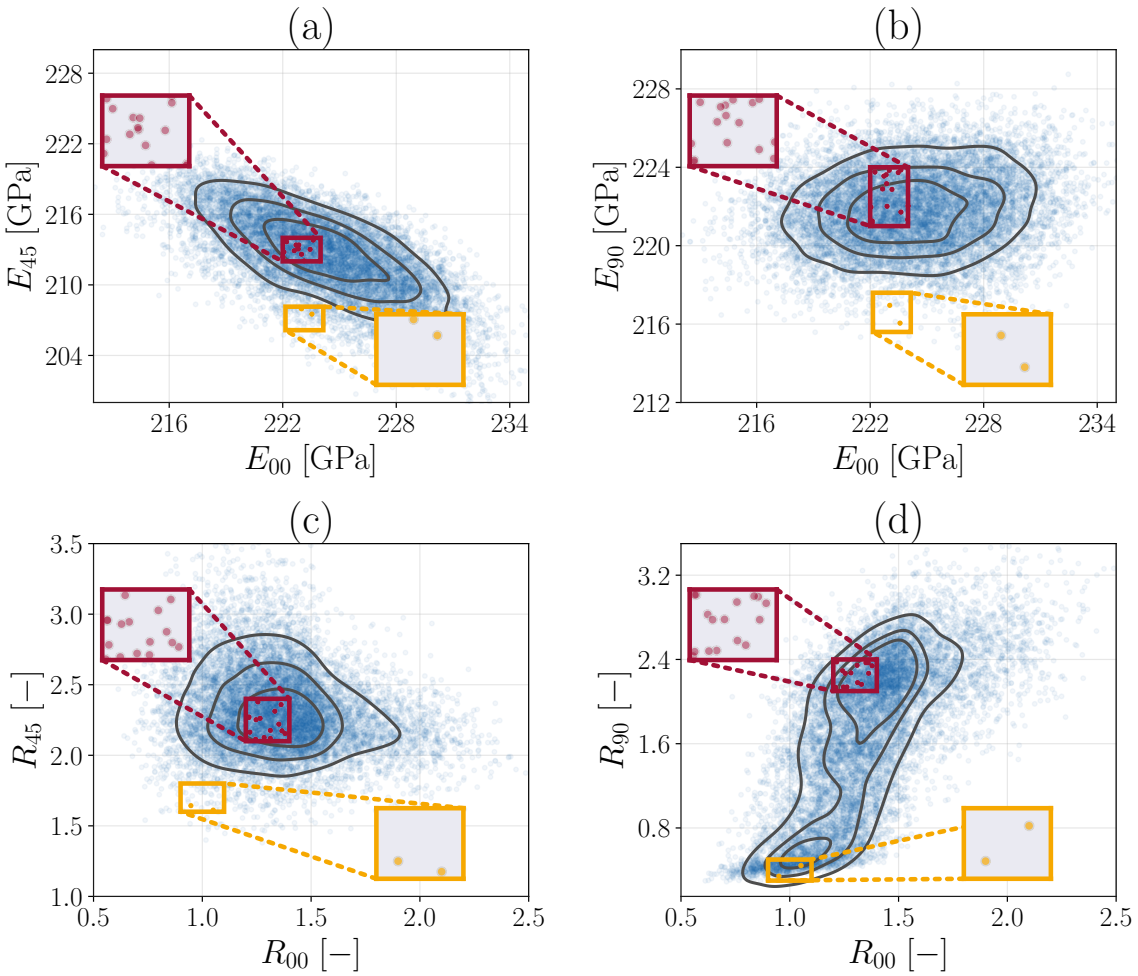

Figure 5.7: Projection of the training set showing different planes of the property space: the Young's modulus E at (a) 0 vs. 45 degree and (b) 0 vs. 90 degree for the R-values at (c) 0 vs. 45 degree and (d) 0 vs. 90 to rolling direction. The orange and red squares mark the projections of *Target Region 1* and *Target Region 2*, respectively. The dots show the projected samples from the training set that lie inside the target region. The black isolines indicate regions with the same point cloud density.

The center of each target region is listed in Tab. 5.7. The target regions are defined by adding a tolerance of ±1000 MPa to the Young's moduli and ±0.10 to the R-values, yielding a sufficiently small properties window from an engineering point of view. As a baseline, all data points from the training set that lie inside the target regions are collected. In *Target Region 1* only two textures can be found, whereas in *Target Region 2* 13 textures can be found.

Table 5.7: Center points of the two target regions (TR). The Young's moduli E are given in MPa, the R-values in [-].

TR	E_{00}	E_{45}	E_{90}	R_{00}	R_{45}	R_{90}
1	223145	207148	216599	1.01	1.7	0.4
2	223000	213000	222000	1.3	2.2	2.2

The optimization algorithm JADE (Zhang and Sanderson, 2009), an extension of the differential evolution algorithm (Storn and Price, 1997), is used to identify a diverse set of textures, while the external archive is not used in this study. Before starting the optimization via JADE, an initial population has to be selected: Therefore, 100 textures are sampled from the test set, which are approximately uniformly distributed over the property space. The cost function, defined in Eq. (5.11), is used with the weights $\omega_{\text{prp}} = 0.90$, $\omega_{\text{val}} = 0.03$ and $\omega_{\text{div}} = 0.07$ and $\mathcal{F}_{\text{prp}}$ and $\mathcal{F}_{\text{div}}$ is scaled to values between 0 and 1 based on the selected 100 initial textures. The threshold ξ_{val} is set to 0.01 based on the maximum anomaly score in the data set, cf. Fig. 5.6. The optimization is performed for 300 iterations with a fixed population size of 100. During the optimization, all valid textures that fulfill the target properties are collected, according to the texture-property mapping. Based on the results from the previous section, the trained SMTL model with a 16-dimensional latent feature space is utilized.

Target region 1

The approach is able to find a diverse set of textures that meet the property requirements of *Target Region 1*, according to the texture-property mapping. Fig. 5.8 depicts the mutual distances in the latent feature space between all the found textures and between the two baseline textures. It is shown, that the set of identified textures contains 221 diverse textures in contrary to only two in the baseline set.

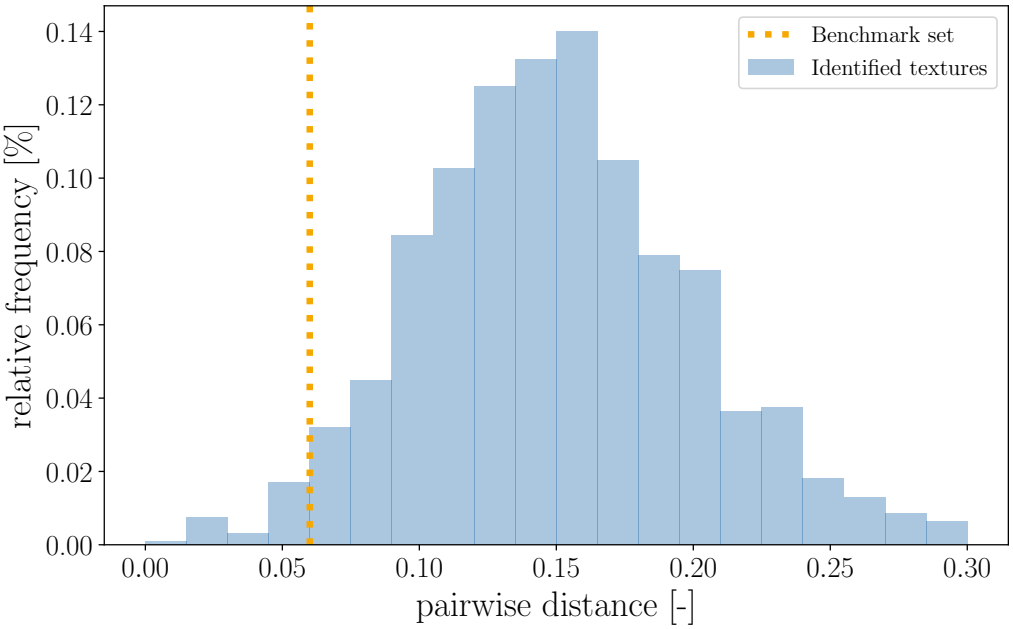

Figure 5.8: Histogram of pairwise SSE distances of the set of identified textures and the baseline set for *Target Region 1*. The distance between the two textures from the baseline set is indicated by the dashed line.

In order to compare the results to the experimentally measured texture, the closest texture to the center point of *Target Region 1* is depicted in Fig 5.9 as a section through the Euler space at $\varphi_2 = 45°$. By comparing the two textures, it can be seen that they are roughly the same in terms of the magnitude of the intensities and the shape of the α- and γ-fibers. However, they also show differences in terms of smoothness and the location of the intensity peaks.

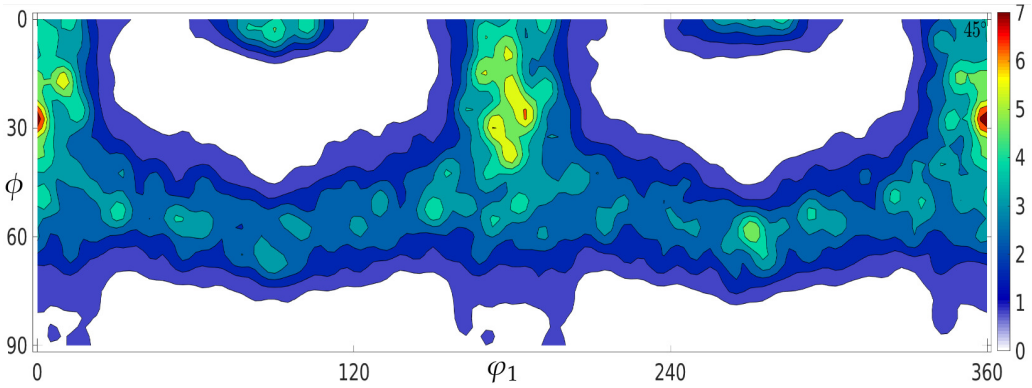

Figure 5.9: Texture that yields properties which are closest to the center of *Target Region 1*. The plot shows the $\varphi_2 = 45°$ section of the orientation distribution function.

Target region 2

Compared to *Target Region 1*, an even more diverse set of 1,315 textures can be identified for *Target Region 2*, which can be seen in the histogram of the mutual distances in Fig. 5.10. To get an idea of the differences between the textures, two exemplary textures are plotted in Fig. 5.11 as a section through the Euler space at $\varphi_2 = 45°$. It can be seen that the α- and γ-fiber of both textures differ significantly in terms of intensity. However, the locations of the intensity peaks and the thickness of the α- and γ-fiber are similar.

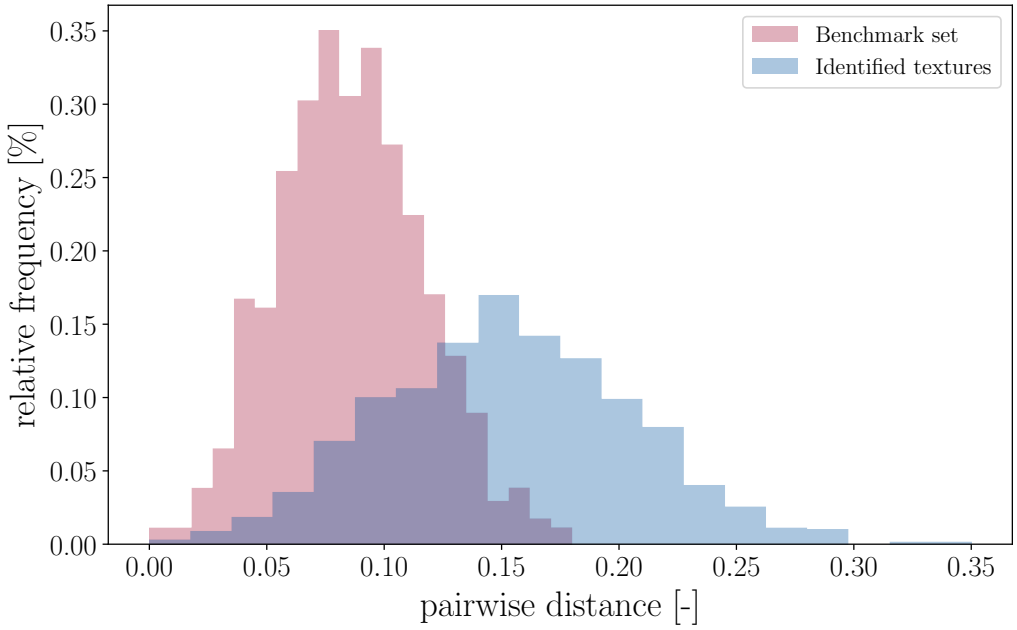

Figure 5.10: Histogram of mutual distances of the set of identified textures and the baseline set for *Target Region 2*.

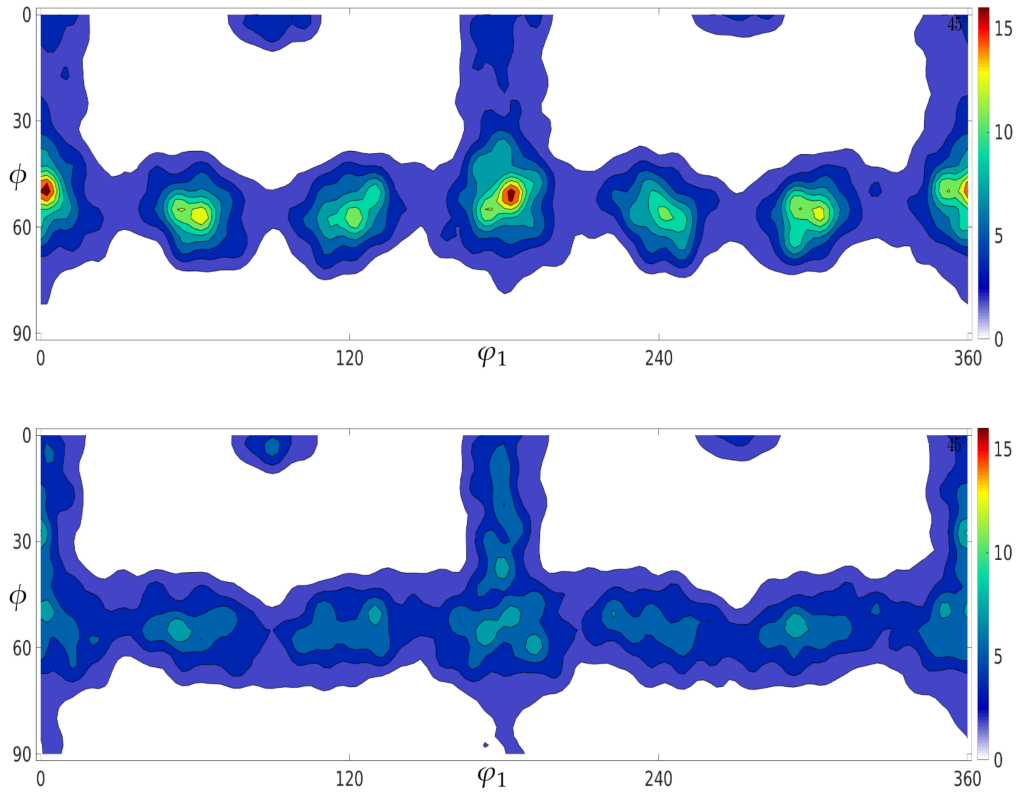

Figure 5.11: Two exemplary textures from the set of identified textures. Both plots show $\varphi_2 = 45°$ sections of the respective orientation distribution functions.

5.3 Discussion

The results presented in Section 5.2.2 show that the two tasks texture-property mapping and validity-prediction are solved by the Siamese multitask learning (SMTL) model. To achieve a sufficient prediction quality for both tasks in the test set as well as for the experimentally measured texture, a minimum dimensionality of the latent feature space is needed. Here, also the dimensionality requirements of the distance preservation goal has to be considered. 16 latent features were found to be sufficient for the example task regarding the texture of cold rolled bcc steel sheets.

However, the prediction error for the experimentally measured texture is higher than for the test set using the same latent feature space dimensionality. This can be explained by the fact that the corresponding property is in a texture space region with low sampling density and the model therefore is not well supported by data. This results also in an instability of the model quality depending on the dimensionality of the latent feature space in this region. This instability can be seen by studying the R-value in Tab. 5.6. By choosing the latent feature space size of 16, also the results for the experimentally measured texture are satisfactory, especially keeping in mind that the experimentally measured texture differs naturally from the artificially generated data and additionally lies in a sparsely sampled region, cf. *Target Region 1* in Fig. 5.7.

Due to the sparsity of *Target Region 1*, the identification of textures in this region is challenging. Nevertheless, the optimization approach is able to identify a set of textures that contains

more diverse individuals compared to the two baseline textures from the training set. Regarding the identified texture, which is closest to the experimentally measured texture in terms of properties, one can see that they are also similar in terms of crystallographic texture, what basically proofs the concept of the approach.

The most obvious difference between both textures is smoothness. The irregular distribution of intensity peaks of the identified texture is due to the resolution of the histogram-based texture descriptor. Also, the orthorhombic sample symmetry is not represented locally. However, by increasing the resolution, these two issues can be solved. Furthermore, a higher resolution of the descriptor decreases the descriptor error, which reflects the deviation between the properties of the original texture and the properties of the texture described by the descriptor. However, the choice of resolution is a trade-off between accuracy and descriptor complexity. Generally, with the use of the SMTL model and the incorporated feature extraction, the resolution is limited only by computational power.

Compared to *Target Region 1*, the identification task for *Target Region 2* seems to be less challenging as the target region is located in a densely sampled region. However, as there already exists a proper set of diverse textures in the baseline, the main challenge is to outperform the baseline set in terms of diversity. Fig. 5.10 shows that the materials design problem (the identification of multiple equivalent microstructures/ textures) is accomplished by the optimization approach. This is exemplary shown when comparing two of the identified textures in Fig. 5.11 with each other: similar properties can be reached by different microstructures. The identification of such a highly diverse set of microstructures with similar properties is an important precondition to construct robust optimizing process control algorithms, which need to choose among multiple optimal paths leading to desired properties.

Data availability

The data used to validate the SMTL approach is made available via the Fraunhofer repository Fordatis at `https://fordatis.fraunhofer.de/handle/fordatis/204` (Morand et al., 2023).

Chapter 6

Measuring texture distances and learning texture-property relations

The crystallographic texture is a microstructural feature that describes the non-uniform distribution of grain orientations in polycrystalline materials. In metallic materials, the texture is initiated via solidification and can be modified by thermal, mechanical, and thermo-mechanical processes like heat treatment, cold forming or hot forming. The existence of preferred grain orientations, however, leads to the typically anisotropic behavior of processed metallic products and is therefore an important aspect for the metal industry. Consequently, texture optimization along the process–structure–property (PSP) linkage (Olson, 1997) is of major importance. The measurement of the distance or also the similarity between crystallographic textures becomes essential when target textures have to be reached by a process. Then it is important to know how far the actual texture of the process path is away from the target (Dornheim et al., 2022). The knowledge of texture distances is also required for the design of textures for given desired properties (Iraki et al., 2024a).

In general, the distribution of microstructural features, such as the orientation distribution function (ODF), can be described by sets of n-point correlation functions (Torquato and Haslach, 2002). This work is focused on the first order approximation by the one-point correlation function (probability density function). This approximation is commonly used for many tasks in materials science and industry, see (Niezgoda et al., 2011) and (Kocks et al., 1998). One of the most important one-point correlation functions for many material properties in polycrystalline materials is the ODF, see (Kocks et al., 1998). It is used in this paper to demonstrate the concepts of representations and distance measurements of one-point correlation functions. In the literature, however, only few investigations in ODF distance measures can be found.

The commonly used distance measures take only into account the shape difference of the density functions, but do not consider the underlying orientation distances and correlations. This missing information is represented in the proposed new distance measures, which also represent orientation distances or even neighborhood correlations between orientations. The effect of incorporating or neglecting this information by using the various distance measures is discussed and demonstrated in this paper. Process path optimization and machine learning (ML) of microstructure-property relations are chosen as demonstration use cases.

6.1 Methods

The following section presents the development of distance metrics for the crystallographic textures relevant to this study. The following metrics are employed: the orientation distribution function, generalized spherical harmonic functions, orientation histograms, and pole figures. For a discussion of the textures, refer to Section 2.3.1.

6.1.1 Distance measures for texture representations

Orientation distribution function

In material science the commonly used distance measure between two ODFs $f^{(1)}(g)$ and $f^{(2)}(g)$ is defined in (Adams et al., 2001), Eq. (49) as:

$$\mathscr{D}_{\mathrm{odf}}(f^{(1)}(g), f^{(2)}(g)) = \int_g \left(f^{(1)}(g) - f^{(2)}(g)\right)^2 \mathrm{d}g, \tag{6.1}$$

with g being an orientation in the orientation space $SO(3)$. This distance measure represents the integral squared difference between the amplitudes of the density functions only. In Eq. (6.1) the ODFs are usually represented as truncated GSH series, but can also be approximated by orientation histograms. In (Adams et al., 2001), approximations of the ODF distance are calculated similarly to Eq. (6.1), but on the basis on two-point correlation functions. In this study, this approach is transferred to ODFs, which capture one-point correlation functions. As already mentioned in the Introduction, due to the property of Eq. (6.1), neither the distance nor the neighborhood relationships between the corresponding orientations are reflected by this distance measure. Instead, the Sinkhorn distance is proposed in the study.

Generalized spherical harmonic functions

The distance between two ODFs $f^{(1)}(g)$ and $f^{(2)}(g)$ can be expressed according to Eq. (6.1) as

$$\begin{aligned}\mathscr{D}_{\mathrm{odf}}(f^{(1)}(g), f^{(2)}(g)) = \frac{1}{8\pi^2} \int_0^{2\pi}\int_0^{\pi}\int_0^{2\pi} \\ \left(f^{(1)}(\varphi_1, \Phi, \varphi_2) - f^{(2)}(\varphi_1, \Phi, \varphi_2)\right)^2 \\ d\varphi_1 \sin(\Phi) d\Phi d\varphi_2.\end{aligned} \tag{6.2}$$

When the densities $f^{(1)}(g)$ and $f^{(2)}(g)$ are expanded in two series of generalized spherical harmonics (GSHs) base functions

$$f_{\mathrm{g}}^{(1)}(g) = \sum_{l=0}^{\infty} \sum_{m=-l}^{+l} \sum_{n=-l}^{+l} C_l^{mn} e^{im\varphi_2} P_l^{mn}(\cos(\Phi)) e^{in\varphi_1} \tag{6.3}$$

and

$$f_{\mathrm{g}}^{(2)}(g) = \sum_{l=0}^{\infty} \sum_{m=-l}^{+l} \sum_{n=-l}^{+l} \tilde{C}_l^{mn} e^{im\varphi_2} P_l^{mn}(\cos(\Phi)) e^{in\varphi_1}, \tag{6.4}$$

the distance measure of Eq. (6.2) becomes

$$\mathscr{D}_{\mathrm{gsh}}(f_{\mathrm{g}}^{(1)}(g), f_{\mathrm{g}}^{(2)}(g)) = \frac{1}{8\pi^2} \int_0^{2\pi} \int_0^{\pi} \int_0^{2\pi} \left(\sum_{l=0}^{\infty} \sum_{m=-l}^{+l} \sum_{n=-l}^{+l} (C_l^{mn} - \tilde{C}_l^{mn}) e^{im\varphi_2} P_l^{mn}(\cos(\Phi)) e^{in\varphi_1} \right)^2 d\varphi_1 \sin(\Phi) d\Phi d\varphi_2. \tag{6.5}$$

Considering the orthogonality of P_l^{mn}, the distance measure between orientation densities $f_{\mathrm{g}}^{(1)}(g)$ and $f_{\mathrm{g}}^{(2)}(g)$ reduces to

$$\mathscr{D}_{\mathrm{gsh}}(f_{\mathrm{gsh}}^{(1)}(g), f_{\mathrm{gsh}}^{(2)}(g)) = \sum_{l=0}^{\infty} \sum_{m=-l}^{+l} \sum_{n=-l}^{+l} \frac{1}{2l+1} (C_l^{mn} - \tilde{C}_l^{mn})^2. \tag{6.6}$$

For the sake of completeness, the full derivation of Eq. (6.6) is included in Appendix A.4.1.

Orientation histograms

On the basis of the orientation histograms, texture distances can be defined using any histogram-based distance measure. The Chi-Squared distance and the Earth Mover's Distance (EMD), as well as a faster implementation of the EMD, called the Sinkhorn distance (Cuturi, 2013), are introduced in the following.

Chi-Squared distance: In (Dornheim et al., 2022), the Chi-Squared distance measure (Pele and Werman, 2010) was shown to be suitable for measuring texture distances. The Chi-Squared distance $\mathscr{D}_{\chi^2}$ between two orientation histograms $f_{\mathrm{o}}^{(1)}(g)$ and $f_{\mathrm{o}}^{(2)}(g)$ is defined in Eq. (2.42) following (Pele and Werman, 2010). However, for comparing sparsely populated histograms, the applicability of the Chi-Squared distance measure is limited. In such cases, the Chi-Squared distance measure is very likely to show very high values (close to the maximum) and is therefore not sensitive for changes in the histograms. This is why the EMD (Rubner et al., 1997) is proposed for use.

Earth mover's distance: The EMD (Rubner et al., 1997; Rubner et al., 2000), going back to the works (Werman et al., 1985) and (Peleg et al., 1989), is based on a solution to the well-known transportation problem (Hitchcock, 1941). It is typically used for comparing distributions over a discrete region (e.g. the bins of a histogram as being used in image retrieval (Rubner et al., 2000)). EMD measures the minimum amount of work needed to transport the probability mass from one distribution to the other to make them equal. The work of transfer between two bins is the transported probability mass times the distance between the representatives of the bins. In this case, the minimum amount of work needed to transform one orientation histogram into the other is considered. Thereby, orientation distances between bins play an important role, compared to the Chi-Squared distance.

The computation of the EMD can be formalized as a linear programming problem with constraint conditions as described in (Rubner et al., 1997), which is used in this approach. The principle procedure of this approach is as follows: Given the following two sets of tuples: $\boldsymbol{t}^{(1)} = \{(q_1^{(1)}, f_{\mathrm{o}_1}^{(1)}(g)), (q_2^{(1)}, f_{\mathrm{o}_1}^{(1)}(g)), \ldots, (q_m^{(1)}, f_{\mathrm{o}_m}^{(1)}(g))\}$ and $\boldsymbol{t}^{(2)} = \{(q_1^{(2)}, f_{\mathrm{o}_1}^{(2)}(g)), (q_2^{(2)}, f_{\mathrm{o}_2}^{(2)}(g)),$

$\ldots, (q_n^{(2)}, f_{\mathrm{o}_n}^{(2)}(g))\}$ where $q_i^{(1)}$, $q_j^{(2)}$ are the histogram bin centers represented as unit quaternions and $f_{\mathrm{o}_i}^{(1)}(g)$, $f_{\mathrm{o}_j}^{(2)}(g)$ are the entries (weights) of the corresponding histogram. Respecting the crystal symmetry, the ground distance matrix $D = [d_{ij}]$ is calculated, where d_{ij} is the distance between the quaternion representations $q_i^{(1)}$ and $q_j^{(2)}$. The quaternion distance (Huynh, 2009) is used as the distance function:

$$d_{ij} = \arccos(|q_i^{(1)} \cdot q_j^{(2)}|), \tag{6.7}$$

where $\cdot$ denotes the product of the vector elements (not the quaternion multiplication, which results in another quaternion). The aim of solving the transportation problem is to find a flow matrix $F = [f_{ij}]$, where f_{ij} is the flow between $f_{\mathrm{o}_i}^{(1)}(g)$ and $f_{\mathrm{o}_j}^{(2)}(g)$, that minimizes the overall work

$$\mathrm{WORK}(\boldsymbol{t}^{(1)}, \boldsymbol{t}^{(2)}, F) = \sum_{i=1}^{m}\sum_{j=1}^{n} f_{ij} d_{ij}. \tag{6.8}$$

Once the transportation problem is solved, and thus the optimal flow F is found, the EMD is defined as the normalized distance between the sets $\boldsymbol{t}^{(1)}$ and $\boldsymbol{t}^{(2)}$:

$$\mathscr{D}_{\mathrm{emd}}(\boldsymbol{t}^{(1)}, \boldsymbol{t}^{(2)}) = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} f_{ij} d_{ij}}{\sum_{i=1}^{m}\sum_{j=1}^{n} f_{ij}}. \tag{6.9}$$

Sinkhorn distance: The solution of the transport problem as discussed above suffers from the increase in complexity with increasing number of histogram bins, which is also the dimensionality δ of the histogram representation space. If two histograms of dimension δ are compared, the costs for the calculation of the optimal transport distances scale at least in $O(\delta^3 log(\delta))$ (Cuturi, 2013; Pele and Werman, 2009). The transportation problem therefore becomes intractable when δ exceeds around 100. To overcome this issue, in (Cuturi, 2013) the Sinkhorn algorithm is proposed instead. In this work, an additional constrain is introduced, in order to ensure that the the KL divergence between the flow matrix F and $\pi_i\, \zeta_j^\top$ is smaller to the pre-defined parameter α:

$$\mathrm{KL}(F, \pi_i \zeta_j^\top) \leq \alpha \quad \text{with} \quad \zeta_j = \sum_i f_{i,j} \quad \text{and} \quad \pi_i = \sum_j f_{i,j}. \tag{6.10}$$

This results in the requirement that $s(F)$ should be as large as possible,

$$s(F) = \max\left(\sum_{i,j} f_{i,j} \log f_{i,j}\right). \tag{6.11}$$

The resulting (dual) Sinkhorn distance is

$$\mathscr{D}_{\mathrm{sh}} = \min\left(\sum_{i,j} f_{i,j} \cdot d_{i,j} - \frac{1}{\lambda} s(F)\right), \tag{6.12}$$

which is again a convex function. By forming the Lagrangian and searching for its minimum with respect to $f_{i,j}$, a strictly convex problem is obtained yielding a unique optimal solution. It is shown that the resulting optimum is also a distance. Since the Sinkhorn algorithm relies on matrix-vector products, the distance can be computed through Sinkhorn's matrix scaling algorithm, whereby it is several orders of magnitude faster than transport solvers.

In this approach, histogram distances are calculated using the highly efficient Sinkhorn algorithm ((Cuturi, 2013), Algorithm 1) with 100 iterations and setting the value of parameter λ to 0.01. Furthermore the algorithm can be integrated as a fully differentiable function in optimization frameworks such as being used for neural networks and implemented on Graphics Processing Units.

Pole figures

Distances between textures are measured by comparing two pole figures $f_{\mathrm{p}}^{(1)}(g)$ and $f_{\mathrm{p}}^{(2)}(g)$ pixel-wise by means of the absolute or l_1 loss:

$$\mathscr{D}_{\mathrm{pf}}(f_{\mathrm{p}}^{(1)}(g), f_{\mathrm{p}}^{(2)}(g)) = \frac{1}{M}\frac{1}{N}\sum_{m=1}^{M}\sum_{i=1}^{N}\left|f_{\mathrm{p}_{m,i}}^{(1)}(g) - f_{\mathrm{p}_{m,i}}^{(2)}(g)\right|, \tag{6.13}$$

where N denotes the number of pixels and M denotes the number of pole figures.

6.1.2 Texture representations for learning texture-property relations

When texture-property relations are learned, the textures are transformed into a latent space, which arranges the texture representations according to their properties. In the context of ML in materials design, the goal is to retrieve textures corresponding to given properties. For this purpose it is very important that the latent space representation also reflects the distance in the original texture space. This can be enforced by using e.g. Siamese neural networks (Iraki et al., 2024a). It still remains crucial that the original texture representation contains the essential information about the properties.

To investigate the ability of the discussed texture representations being suitable for carrying property information, neural network-based models are developed for learning the relation between textures and their corresponding properties, based on the *texture-property data set* described in Section 6.1.3. This forward mapping from textures to properties is modeled as a regression problem. The regression loss is given by the MSE between the true properties $\boldsymbol{y}_{\mathrm{est}_{\mathrm{prp}}}$ and the predicted properties $\hat{\boldsymbol{y}}_{\mathrm{est}_{\mathrm{prp}}}$:

$$\mathcal{J}_{\mathrm{est}_{\mathrm{prp}}}(\boldsymbol{y}_{\mathrm{est}_{\mathrm{prp}}}, \hat{\boldsymbol{y}}_{\mathrm{est}_{\mathrm{prp}}}) = \frac{1}{P}\sum_{i=1}^{P}(y_{\mathrm{est}_{\mathrm{prp}_i}} - \hat{y}_{\mathrm{est}_{\mathrm{prp}_i}})^2 + \gamma\mathcal{R}(\Theta_{\mathrm{reg}}), \tag{6.14}$$

where P denotes the number of properties, Θ_{reg} are the trainable parameters and $\mathcal{R}(\Theta_{\mathrm{reg}})$ is a regularization term that is used to prevent overfitting with the hyperparameter γ defining the strength of the regularization (also known as weight decay, see (Krogh and Hertz, 1991) and (Hinton, 1987)). The elastic and plastic material properties are described using E_{11}, E_{22}, E_{33}, $\widetilde{R}23$, $\widetilde{R}12$, and $\widetilde{R}_{13}$. To learn the relation between textures and their corresponding properties, the textures are represented in three different ways:

- as orientation histograms having the resolution $J = 512$ and $l \in \{1, 3\}$.
- as pole figures for the Miller's indices (001), (110), (111), where the polar coordinates are transformed into Cartesian coordinates with a resolution of 64×64 pixels. The resulting pole figure representation for one exemplary ODF is depicted in Fig. 6.1.

- as GSH textures for degree L=16 and L=10. For the mapping, only a subset of the GSH coefficients is used, as the first coefficient equals one and half of the remaining coefficients are complex conjugates of the other half (Tallman et al., 2019).

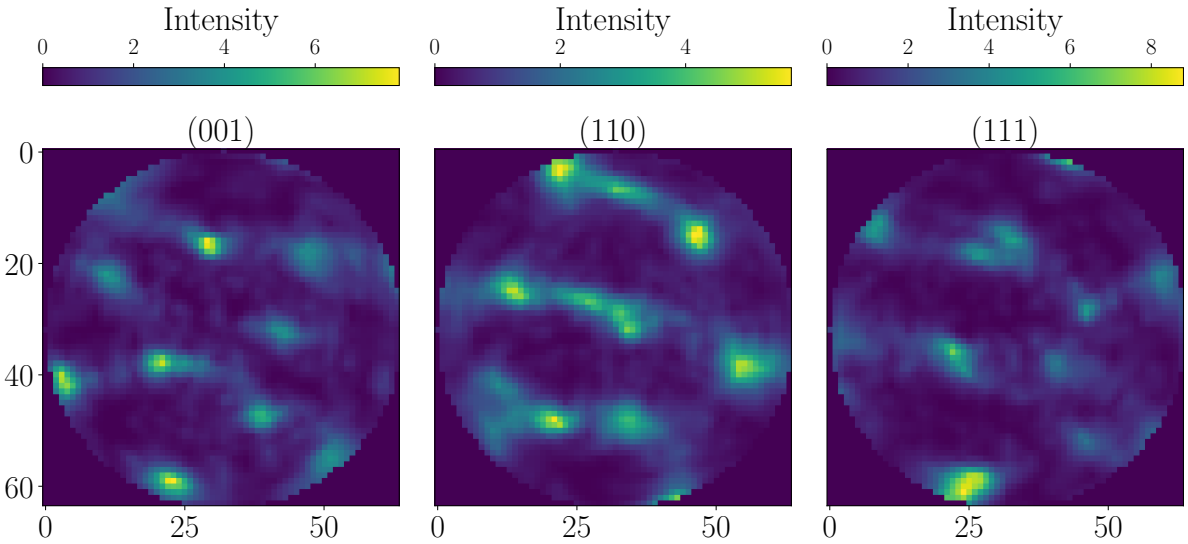

Figure 6.1: Representation of the orientation distribution function by two-dimensional pole figures for the Miller's indices (001) (left), (110) (center) and (111) (right) with a resolution of 64×64 pixels.

Fully connected neural networks are used for orientation histograms and GSH representations, convolutional neural networks (CNNs) are used for pole figure representations (Yamanaka et al., 2020; Koenuma et al., 2020). CNNs are well-studied deep learning methods for image data (LeCun et al., 1990; LeCun and Bengio, 1995; LeCun et al., 1999) and are particularly powerful for processing images, where the underlying neighborhood relations between pixels are exploited, as is the case with pole figures. The topologies of the neural network models are listed in Tab. A.4 and are explained in more detail in the following:

Orientations histograms: The topology for the neural network on the basis of orientation histogram (see Tab. A.4 (a) and Fig. A.19) consists of three hidden fully connected layers, where each of the layers halve the size of the previous layer. The output of the third hidden layer is followed by two processing layers which feed a final linear regression layer, giving a float output vector for the six predicted properties $\hat{\boldsymbol{y}}_{\mathrm{est_{prp}}}$.

GSH: The topology for GSH with L=10 (see Tab. A.4 (b) and Fig. A.20) half the size of the previous layer within the first three fully connected hidden layers. This is followed by the final linear regression layer for the six predicted properties $\hat{\boldsymbol{y}}_{\mathrm{est_{prp}}}$. The first fully connected hidden layer of the topology for GSH with L=16 (see Tab. A.4 (c) and Fig. A.21) reduces the input data to the size of 160 neurons. This is followed by four hidden fully connected layers, where each of the layers halve the size of the previous layer. This is followed by the final linear regression layer for the six predicted properties $\hat{\boldsymbol{y}}_{\mathrm{est_{prp}}}$.

Pole figures: The layout of the CNN topology can be seen in Tab. A.4 (d) and Fig. A.22. The convolutional layers have a filter kernel size of 3×3, for which an increasing size of the feature maps was selected. A common approach is a stride of length one with no sub-sampling in the convolutional layers. Instead, sub-sampling is performed in the pooling layers with stride of length two, halving the pooling layer input (in a Gaussian resolution pyramid). The convolutional layers and the pooling layers are stacked two times and their output is vectorized. This high-dimensional vector is processed by four consecutive fully connected layers to feeding the final linear regression layer for the six predicted properties $\hat{\boldsymbol{y}}_{\mathrm{est}_{\mathrm{prp}}}$.

Implementation details: In the context of hyperparameter optimization the random search method (Bergstra and Bengio, 2012) is applied using 5-fold cross-validation. Based on the analysis of the hyperparameters, the following decisions are made about the investigated hyperparameters: The Xavier method (Glorot and Bengio, 2010) is used for weight initialization. The Adam Optimizer (Kingma and Ba, 2015) was found to be most efficient, with the following parameters: learning rate = 0.001, weight decay = 10^{-6}, batch size = 32. The models are trained for 200 epochs, where the best intermediate result of the test set is retained to prevent over fitting. This can be considered as an application of early stopping (Prechelt, 1998). The *tanh* activation function (c.f. Eq. (2.10)) is used. The neural networks are implemented based on the Pytorch API (Paszke et al., 2019).

6.1.3 Texture data sets for evaluation

Texture evolution and properties calculation

The basis for the investigations in the present paper is a simulated metal forming process, which is described in (Dornheim et al., 2022) and for which a simulation framework has been published (Morand et al., 2021). The simulation is based on the Taylor-type crystal plasticity model introduced in (Kalidindi et al., 1992), which is modified in terms of hardening as described in (Baiker et al., 2014). A detailed description of the material model is provided in Section 2.3.2. The material dependent parameters that are used in this study are defined as in (Dornheim et al., 2022). These are summarized in Tab. 6.1.

Table 6.1: Material dependent parameters used in this study (cf. (Dornheim et al., 2022)).

parameter	C_{11}	C_{12}	C_{44}	$\dot{\gamma}_0$	m	τ_0	τ_1	ϑ_0	ϑ_1	q_1	q_2
value	226	140	116	0.001	0.02	90	32	250	60	1.4	1.4
unit	GPa	GPa	GPa	s^{-1}	-	MPa	MPa	MPa	MPa	-	-

Basically, the simulation framework allows for applying sequences of tension and compression operations on a point within a workpiece of the material under consideration, referred to as a material point in the following. These operations are defined by applying the deformation gradient

$$\widetilde{\boldsymbol{F}} = \widetilde{F}_{11}\boldsymbol{e}_1 \otimes \boldsymbol{e}_1 + \widetilde{F}_2\boldsymbol{e}_2 \otimes \boldsymbol{e}_2 + \widetilde{F}_{33}\boldsymbol{e}_3 \otimes \boldsymbol{e}_3, \tag{6.15}$$

with orthogonal basis vectors $\boldsymbol{e}_i$ and the outer product denoted by the operator $\otimes$. Expressed

as a matrix, the deformation gradient writes

$$\widetilde{\boldsymbol{F}} = \begin{bmatrix} \widetilde{F}_{11} & 0 & 0 \\ 0 & \widetilde{F}_{22} & 0 \\ 0 & 0 & \widetilde{F}_{33} \end{bmatrix}. \tag{6.16}$$

$\widetilde{F}_{11}$ is defined by the applied tensile load increment and both, $\widetilde{F}_{22}$ and $\widetilde{F}_{33}$, are determined such that the stress boundary conditions are fulfilled, cf. (Dornheim et al., 2022). Each of the individual tensile load increments $\widetilde{\boldsymbol{F}}$ can be oriented arbitrarily to the material point yielding $\hat{\boldsymbol{F}}$, which is a rotated form of the deformation gradient, following

$$\hat{\boldsymbol{F}} = \boldsymbol{R}\widetilde{\boldsymbol{F}}\boldsymbol{R}^{\top}, \tag{6.17}$$

with the rotation matrix $\boldsymbol{R}$. The framework is used to simulate texture evolution and generate different crystallographic textures for analysis. It is well known that Taylor-type mean-field models typically overestimate texture evolution (Kocks et al., 1998). For the purpose of this study, the mean-field model used is sufficient, as execution speed is prioritized over the prediction of highly realistic material behavior.

In addition, the simulation framework is used to calculate material properties.The focus is on properties such as the orientation-dependent Young's moduli and anisotropy similar to R-values, as in (Morand et al., 2022). R-values, however, are originally developed to express plastic anisotropy in sheet metals. On this basis, an R-values-like measure, $\widetilde{R}$, is used to described the plastic anisotropy at the material point in three space directions. To do so, tension tests are conducted in each space direction using the simulation framework. The $\widetilde{R}_i$-values are given by the relation between tensile and lateral strain. On the basis of the simulated tension tests, also, the Young's modulus E_i is calculated in three space directions.

Crystallographic texture data generation

In total, two data sets are generated: a first one, which serves to compare the different measures of texture distances and a second (larger) one to serve the demonstration use case of learning texture-property relations. The generation of both data sets is described in the following.

Distance measurement evaluation set: For this purpose, datasets are sought where the texture distance between subsequent elements increases continuously. Such an ODF dataset is generated by simulating a uniaxial tension test with 20 load steps and 4% applied strain. When repeatedly applying tension operations on a material with initial texture as in the example process, the distance from the initial texture is expected to be smoothly and monotonically increasing. Three such datasets are generated by starting the simulated tension test with different initial textures as follows:

- Initial texture 1: A grey initial texture is used that consists of 512 nearly uniformly distributed orientations ($n_{\text{oris}} = 512$) in the cubic-triclinic fundamental zone. The orientations are generated using the algorithm described in (Quey et al., 2018).
- Initial texture 2: 100 orientations ($n_{\text{oris}} = 100$) are randomly sampled from the 512 orientations of initial texture 1.
- Initial texture 3: The third initial texture consists of only one orientation ($n_{\text{oris}} = 1$).

Texture-property data set: For learning of a generalized texture-property mapping, a total of 76980 simulations were conducted, each by applying seven tension steps of 10% strain based on Eq. (6.17). The rotation matrix $\boldsymbol{R}$ is determined on the basis of one randomly chosen orientation from a set of 25 (also nearly uniformly) distributed orientations. The properties E_i and $\tilde{R}_i$ are then calculated for each texture of the thereby obtained texture dataset.

6.2 Results

The following demonstrates the effect of using different texture representations in the two investigated use cases.

6.2.1 Comparing measures of texture distances

The distances of crystallographic textures are usually measured in materials science by means of Eq. (6.1) with the texture ODFs represented as truncated GSH expansions. A more recent approach uses orientation histograms as discrete representations of the ODFs, while the distance is still calculated according to Eq. (6.1) in its discretized form (Chi-Squared distance $\mathscr{D}_{\chi^2}$ (Eq. (2.42))). The alternative approaches of using the Sinkhorn distance $\mathscr{D}_{\text{sh}}$ (Eq. (6.9)) between orientation histograms and of representing ODFs as pole figures to calculate the pole figure distance $\mathscr{D}_{\text{pf}}$ (Eq. (6.13)) capture orientation neighborhoods of and correlations between orientations. The non-GSH approaches are first compared among each other, followed by a comparison of the best among them, the Sinkhorn distance, to the classical GSH approach.

The effect of the various discussed ODF representations on the corresponding ODF distance measures are shown in the following. For this purpose, the *distance measurement evaluation set* $\{f^{(0)}(g), f^{(1)}(g), \ldots, f^{(19)}(g)\}$, described in Section 6.1.3, is used. This set results from the forming process, which consists of 20 tension steps. The distances from the initial texture $f^{(0)}(g)$ to each subsequent texture $f^{(i)}(g), i = 1, \ldots, 19 : \mathscr{D}(f^{(0)}(g), f^{(1)}(g)), \ldots, \mathscr{D}(f^{(0)}(g), f^{(19)}(g))$, are determined, where $\mathscr{D}$ represents the distance function. Such sequences of distance measure values are then compared using different texture representations. An overview of the methods of data generation and distance measurements is depicted in Fig. A.23 in Appendix A.4.3.

Orientation histogram and pole figure representations

The following, compare the distances, which result from using

1. the Chi-Squared distance $\mathscr{D}_{\chi^2}$ (Eq. (2.42)),
2. the Sinkhorn distance $\mathscr{D}_{\text{sh}}$ (Eq. (6.9)) and
3. the pole figure distance $\mathscr{D}_{\text{pf}}$ (Eq. (6.13)).

These measures are calculated for the textures resulting from the forming process starting at the three different initial textures. For each texture set, the orientation histograms and pole figures (depicted in Fig. 6.2, Fig. 6.3, and Fig. 6.4) are created as follows: The textures are represented by orientation histograms of different resolution, where the number of histogram bins $J \in \{128, 256, 512, 1024, 2048\}$. The histogram of each resolution is smoothed with three

different soft-assignment values $l \in \{1, 3, 5\}$. Pole figures are generated for the Miller's indices $(001), (110), (111)$ and the polar coordinates are transformed into Cartesian coordinates with a resolution of 64×64 pixels.

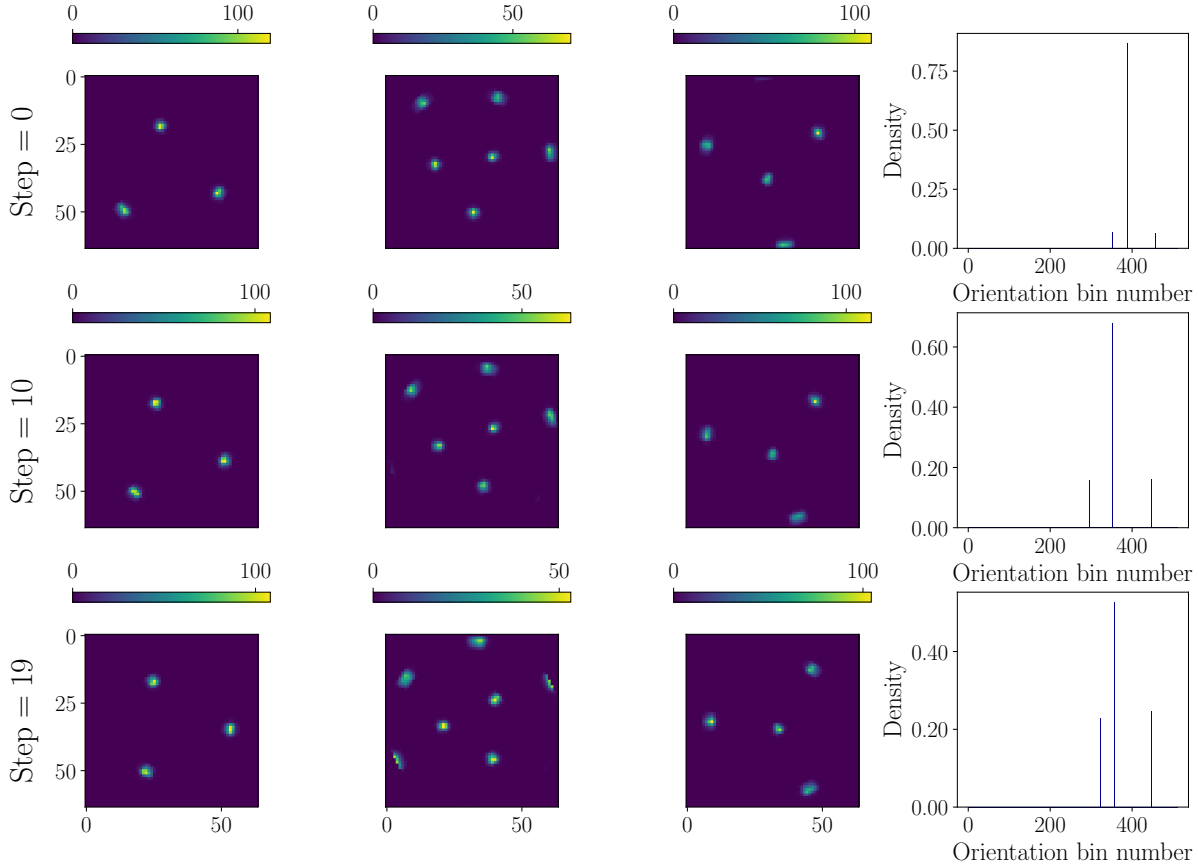

Figure 6.2: Representation of the orientation distribution function with one single orientation ($n_{\text{oris}} = 1$) by two-dimensional pole figures for the Miller's indices (001), (110), and (111) with a resolution of 64×64 pixels as well as by orientation histograms for number of bins $J = 512$ and soft-assignment $l = 3$. The first row shows the initial texture at process step 0, the second row shows the representations at process step 10, and the third row shows the representations at the final process step 19.

To ensure the distances are comparable regardless of the measure under consideration, the raw distances are normalized by dividing them by the maximum possible distance in each case. The maximum possible distance is the distance between two single crystals $f_{\text{sc}}^{(0)}(g)$ and $f_{\text{sc}}^{(1)}(g)$, which have the maximum distance in their orientations:

$$\mathscr{D} = \mathscr{D}\Big(f^{(0)}(g), f^{(k)}(g)\Big) / \mathscr{D}\Big(f_{\text{sc}}^{(0)}(g), f_{\text{sc}}^{(1)}(g)\Big). \tag{6.18}$$

The sequences of these normalized distances are calculated for the three different texture sets and plotted as distance curves over process step number (depicted in Fig. 6.5). They are discussed in the following.

Initial texture 1 ($n_{\text{oris}} = 512$): The distance curves of the Chi-Squared and Sinkhorn distance measures for the (J, l)-settings of the orientation histograms as well the distance curves of the

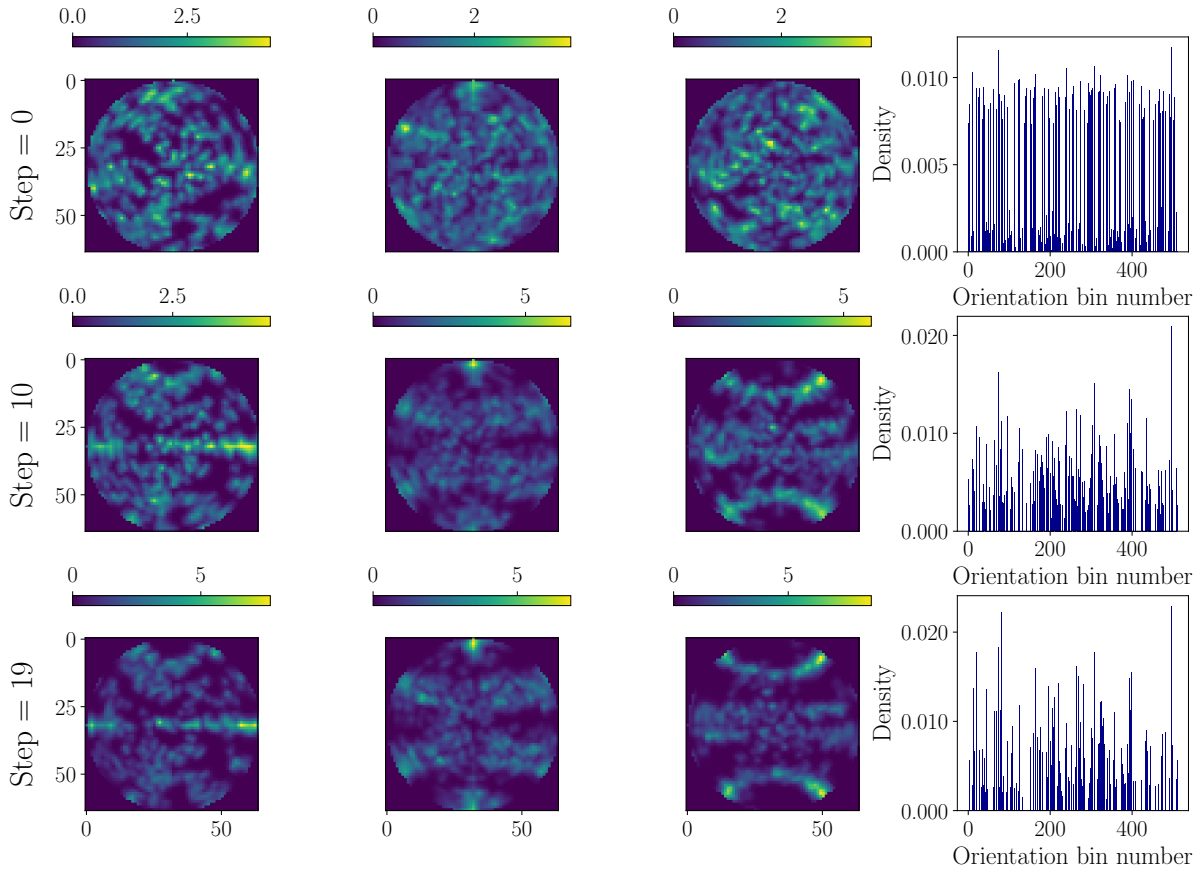

Figure 6.3: Representation of the orientation distribution function with 100 orientations ($n_{\text{oris}} = 100$) by two-dimensional pole figures for the Miller's indices (001), (110), and (111) with a resolution of 64 × 64 pixels as well as by orientation histograms for number of bins $J = 512$ and soft-assignment $l = 3$. The first row shows the initial texture at process step 0, the second row shows the representations at process step 10, and the third row shows the representations at the final process step 19.

l_1-distance measure for the pole figures are depicted in Fig. 6.5 (a) - (c). It can be seen that the Chi-Squared and Sinkhorn distance values for $J = 512$ and $l = 1$ do not increase until step 2 despite the expected texture evolution due to increasing load (Fig. 6.5 (a)). This is because in this special case, all texture orientations are located initially in the bin centers. The process then needs two steps to force the movement of the orientations into another bins. On the other hand, when applying smoothing by soft-assignment with $l = 3$ and $l = 5$, a small process-enforced movement from the bin center is enough to change the histogram and the distance value. Furthermore, the Chi-Squared distance curve is not smooth for $l = 1$ (Fig. 6.5 (a)) and erroneously not monotonic for $l = 1$ and $J = 2048$. At higher soft-assignment values $l = 3$ (Fig. 6.5 (b)) and $l = 5$ (Fig. 6.5 (c)) the Chi-Squared distance curve shows a monotonic behavior and is smoother. Another effect of the discretization parameters of the histograms on their distance curves is the variance of the distance values at a process step. One can see from Fig. 6.5 (a) - (c) that this variance is much higher for the Chi-Squared than for the Sinkhorn distance measure and therefore the Chi-Squared measure is much more sensitive against the choice of histogram parameter values. The Sinkhorn distance as well as the pole figure distance result in monotonic and smoother curves, while the Sinkhorn distance is less sensitive to the number of bins J and

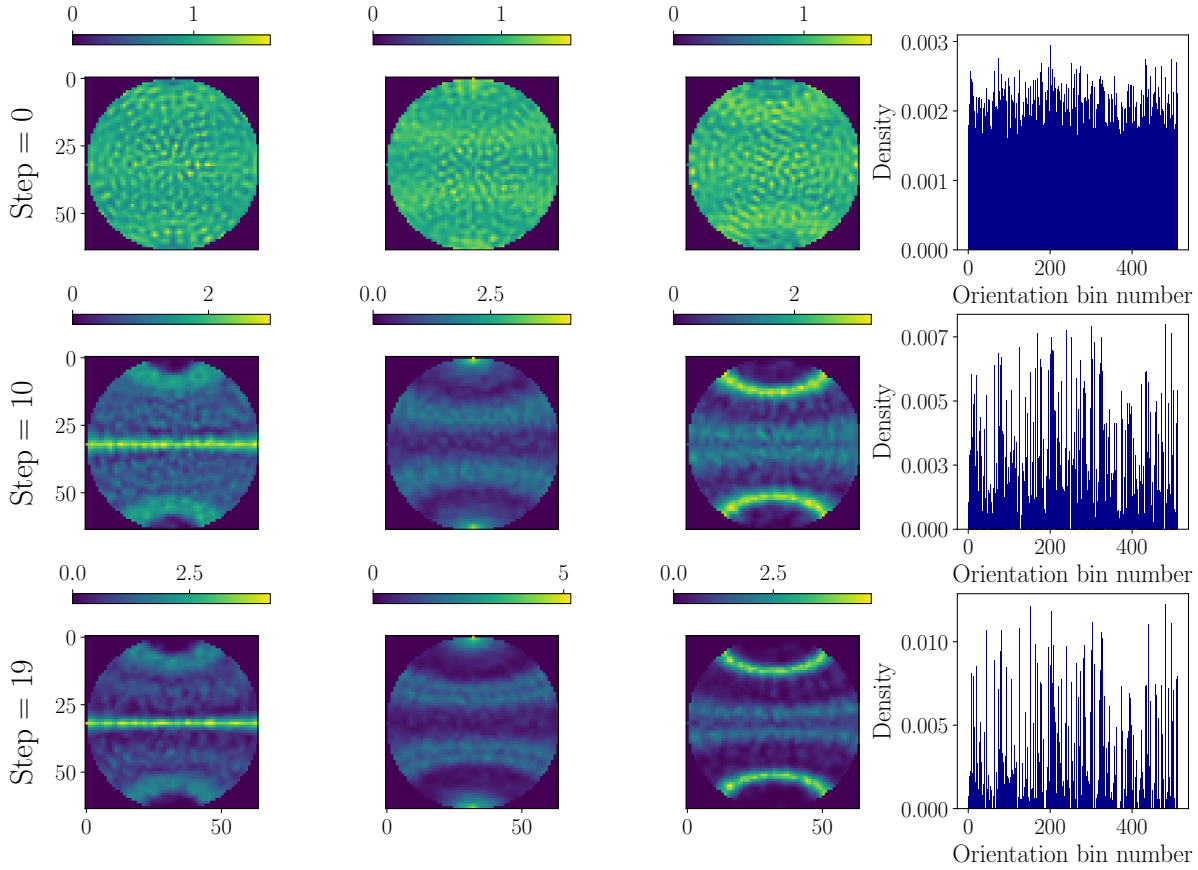

Figure 6.4: Representation of the orientation distribution function with 512 orientations ($n_{\text{oris}} = 512$) by two-dimensional pole figures for the Miller's indices (001), (110), and (111) with a resolution of 64 × 64 pixels as well as by orientation histograms for number of bins $J = 512$ and soft-assignment $l = 3$. The first row shows the initial texture at process step 0, the second row shows the representations at process step 10, and the third row shows the representations at the final process step 19.

soft-assignment l.

Initial texture 2 ($n_{\text{oris}} = 100$): The results of the investigated distances for the orientation histograms and pole figures are depicted in Fig. 6.5 (d) - (f). The general behavior of the curves is similar to the case $n_{\text{oris}} = 512$, but the curves are rougher for the Chi-Squared measure where also the maximum value is much higher compared to $n_{\text{oris}} = 512$ than for the Sinkhorn distance, the curves of which still remain smooth. The distance curve resulting for the pole figure measure remains almost unaffected.

Initial texture 3 ($n_{\text{oris}} = 1$): The results of the investigated distances for textures with only one orientation are shown in Fig. 6.5 (g) - (i). The simulated process delivers a sequence of single crystals where the texture is determined by their orientation quaternions. The quaternion distance Eq. (6.7) can be used in this case as the ground truth for the texture distance. The quaternion distance curve is strictly monotonic and almost linear. The Chi-Squared distance measure can not cope with this case because it changes its value only if the single crystal orientation moves into a different bin and then immediately takes on the maximum possible

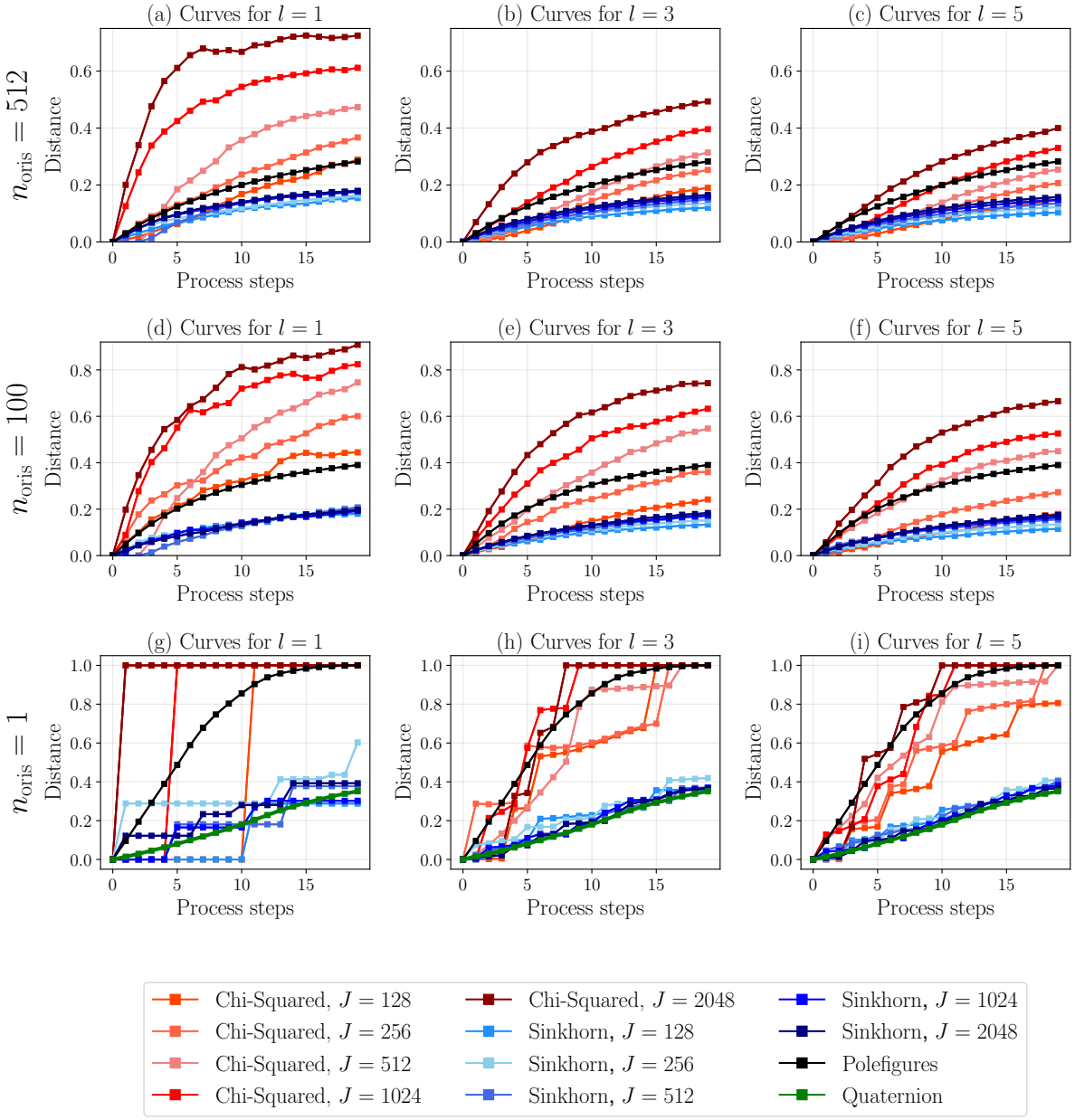

Figure 6.5: Distances over process step curves for the textures with 512 orientations ($n_{\text{oris}} = 512$, first row), with 100 orientations ($n_{\text{oris}} = 100$, second row) and with one single orientation ($n_{\text{oris}} = 1$, third row). The Chi-Squared distance (red) and Sinkhorn distance (blue) are represented for the soft-assignment $l = 1$ (a, d, g), $l = 3$ (b, e, h) and $l = 5$ (c, f, i). Each of these diagrams shows the curves for the different number of bins $J \in \{128, 256, 512, 1024, 2048\}$ and the pole figure distance curve in black. The Quaternion distance curve is shown in the single orientations diagrams (g, h, i) in green color.

value (saturation). Higher-order soft-assignments smoothen this step function behavior and let the curves saturate slower. The discontinuity problem of the Chi-Squared curves is less prominent for Sinkhorn distance curves because the quaternion distances between bin centers are also taken into account. This also forces the Sinkhorn distance curves to be closer to distance curve and also prevents saturation. The pole figure distance curve is monotonic and smooth, but deviates from the linear ground truth curve by forming a half sigmoid function which saturates smoothly with increasing step number.

Generalized spherical harmonic functions

Having established that the Sinkhorn distance provides the best results for histogram distance measures on textures composed of 512 orientations, these results are now compared to those obtained using the distance measure $\mathscr{D}_{\text{gsh}}$ (Eq. (6.6)), based on GSH texture representations with various GSH degrees. For this part of the study, the initial texture 1 set ($n_{\text{oris}} = 512$) from the *distance measurement evaluation set* described in Section 6.1.3 is used. The ODF is approximated by a GSH series expansion (Bunge, 1982) of the degrees L $\in \{4, 6, 8, 10, 16\}$ (Bunge, 1982). For comparison, the corresponding Sinkhorn distance is shown.

The value ranges of the different analyzed distances differ due to different representations. To compensate this effect the distances are normalized to the value of the distance at step 19 (maximum distance value):

$$\mathscr{D} = \mathscr{D}\Big(f^{(0)}(g), f^{(k)}(g)\Big) / \mathscr{D}\Big(f^{(0)}(g), f^{(19)}(g)\Big) \tag{6.19}$$

The results for the GSH distance are shown over process step numbers in Fig. 6.6. It can be seen that the Sinkhorn distance increases stronger in the first steps and then tends to saturate at later steps. In contrast, the GSH distance shows a small increase at the beginning and increases stronger after certain process steps, resulting in an approximately linear increase from step 8 onwards. In general, the GSH distance is strictly monotonic and smooth. The curves show that the GSH distance measure is less sensitive to small textures distances than the Sinkhorn distance measure, while the latter tends to saturate at higher distances.

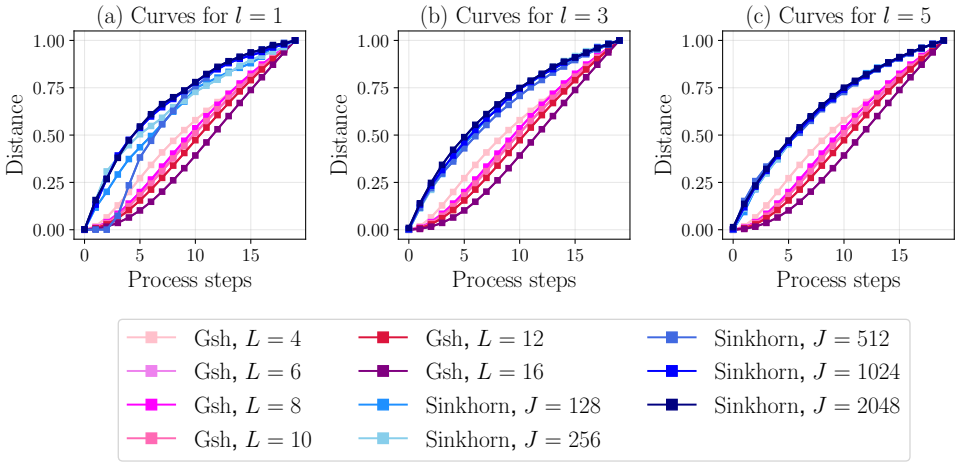

Figure 6.6: Distances over process step curves for the textures with 512 orientations ($n_{\text{oris}} = 512$). In each of the charts (a), (b) and (c) the GSH distance (red) is plotted for L $\in \{4, 6, 8, 10, 16\}$. The Sinkhorn distance curves (blue) for number of bins $J \in \{128, 256, 512, 1024, 2048\}$ are shown in (a) for the soft-assignment $l = 1$, in (b) for $l = 3$ and in (c) for $l = 5$.

Investigation of experimentally measured textures

Finally, the distance measures are evaluated using experimentally measured textures of a DC04 steel sheet at different processing steps. The data was obtained from Electron Backscatter Diffraction (EBSD) measurements conducted after (1) hot rolling, (2) cold rolling and (3) heat treatment in order to study the texture evolution (Schreijäg, 2013). The resulting textures are shown as pole figures and orientation histograms in Fig. 6.7. After hot rolling, a nearly gray texture can be observed, while cold rolling and heat treatment result in pronounced textures typical for these processing steps.

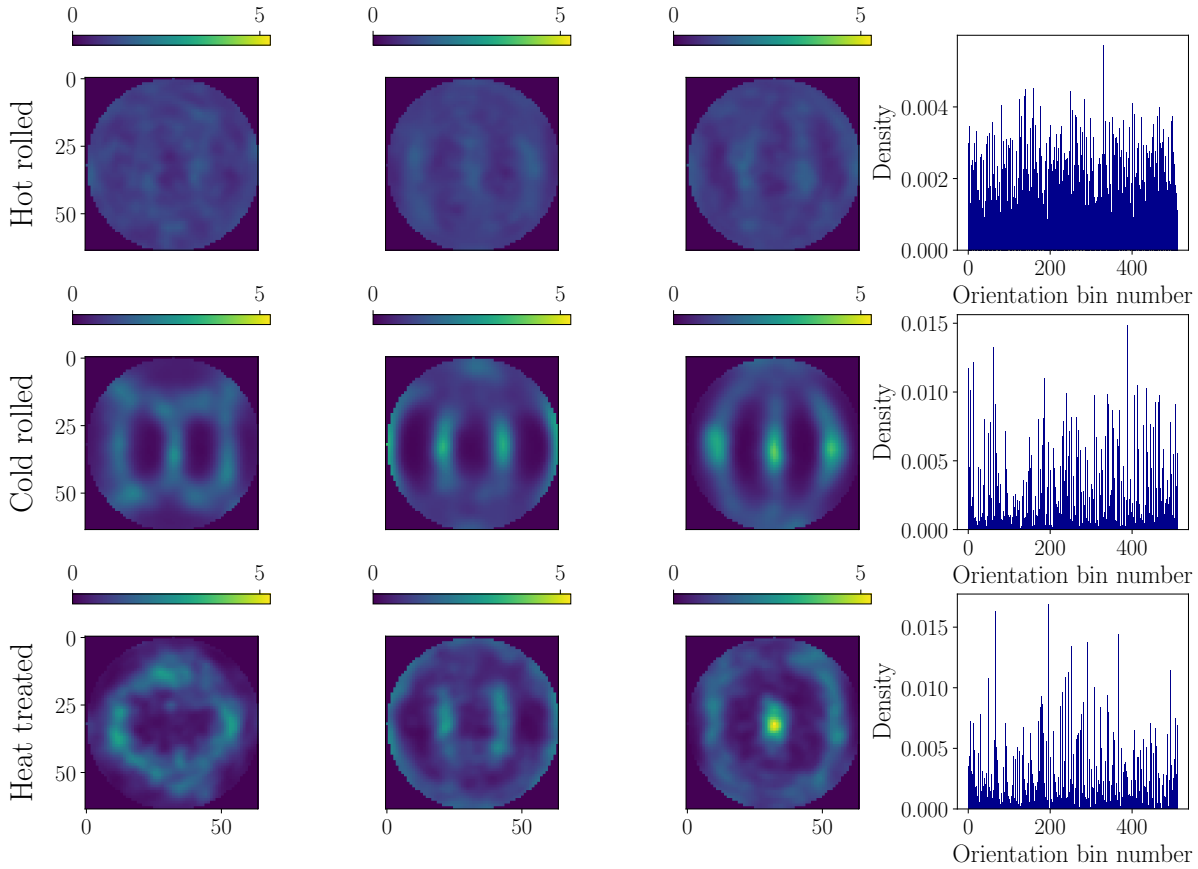

Figure 6.7: Representation of the orientation distribution function for the experimentally measured textures by two-dimensional pole figures for the Miller's indices (001), (110), and (111) with a resolution of 64 × 64 pixels as well as by orientation histograms for a number of bins $J = 512$ and soft-assignment $l = 3$. The first row shows the hot rolled texture, the second row shows cold rolled texture, and the third row shows the heat treated texture.

Additionally visualizations of each individual orientation distribution as $\varphi_2 = 45°$ sections through the Euler space are provided in Fig. 6.8. Therein, the typically pronounced α and γ fiber of cold rolling textures can be clearly seen. The α fiber disappears after heat treatment, while the γ fiber gets more pronounced (Pagenkopf, 2019).

The distances of the ODFs are calculated between textures after (1) hot rolling and (2) cold rolling, (2) cold rolling and (3) heat treatment, and (1) hot rolling and (3) heat treatment. For the experimental data it is not possible to compare GSH-based distance measures to histogram-based and pole-figure-based distance measures, because the prerequisites of normalization as in

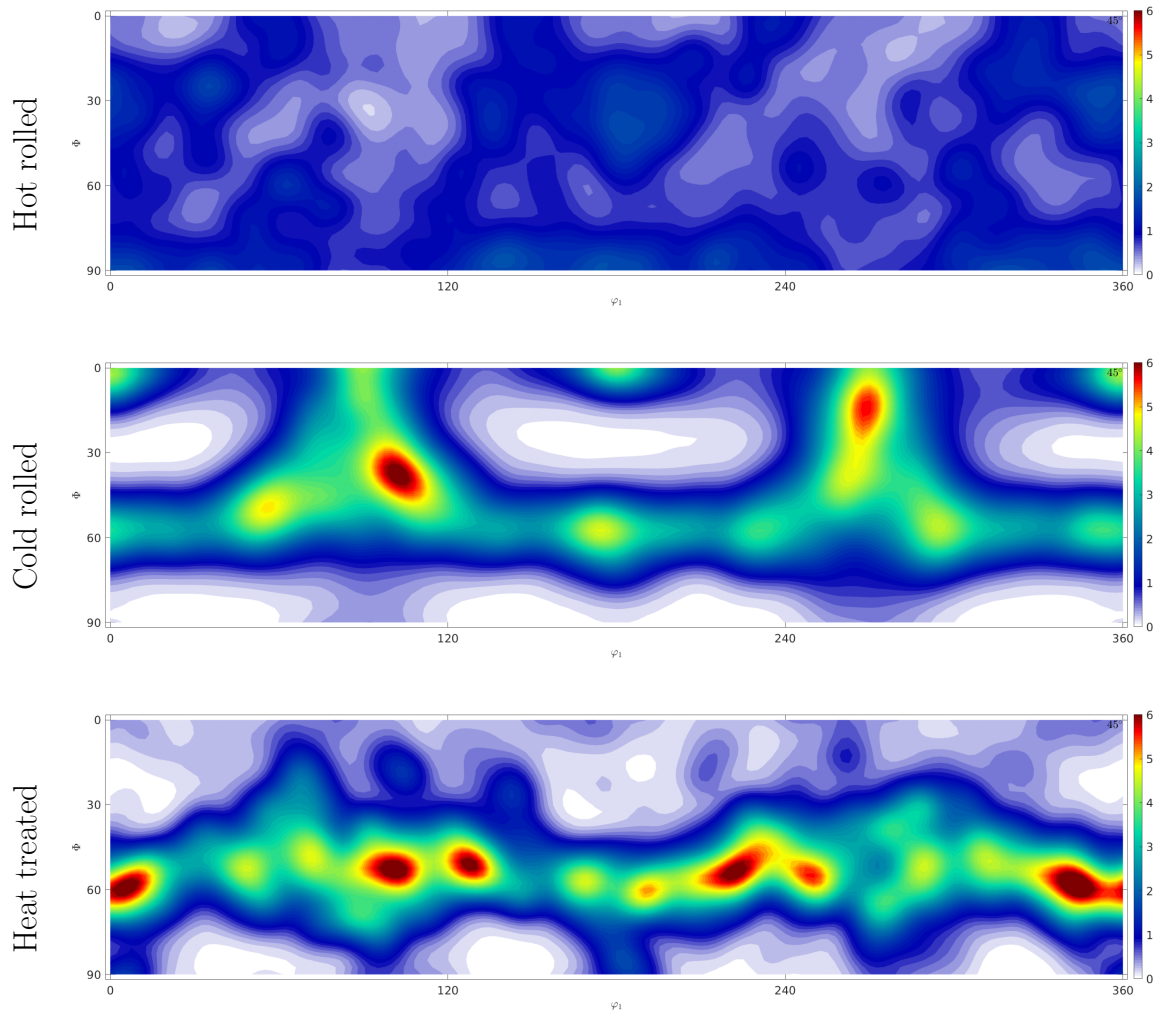

Figure 6.8: Orientation distribution of the experimentally measured textures as $\varphi_2 = 45°$ sections through the Euler space. The first row shows the hot rolled texture, the second row shows cold rolled texture, and the third row shows the heat treated texture.

the simulated process are not given: With only three textures, using one distance for normalization of the others is not meaningful, unlike the approach used with the distance from the last process step in the simulated 20-step process. To compare distances between the three experimental textures, the distances of the histogram-based and pole-figure-based measures were normalized by the distance between two single crystals (maximum possible), but which can not be calculated for GSH representations.

The observed qualitative behavior of the textures is reflected by all distance measures, which are shown in Tab. 6.2. The difference between the behaviors of the different distance measures is small, because all the experimental textures are not very sharp (sparse), which would induce distinct distance measure behaviors.

Table 6.2: Results of the measured distances between each of the experimentally measured textures. The results are shown for the orientation histograms (OH) with a fixed number of bins $J = 512$ and varying soft-assignments l, as well as for the pole figures.

Texture	Distance	Hot rolled vs. Cold rolled	Cold rolled vs. Heat treated	Hot rolled vs. Heat treated
OH for $l = 1$	$\mathscr{D}_{\chi^2}$	0.421	0.296	0.453
OH for $l = 3$	$\mathscr{D}_{\chi^2}$	0.368	0.209	0.340
OH for $l = 5$	$\mathscr{D}_{\chi^2}$	0.343	0.184	0.305
OH for $l = 1$	$\mathscr{D}_{\text{sh}}$	0.206	0.180	0.207
OH for $l = 3$	$\mathscr{D}_{\text{sh}}$	0.195	0.166	0.192
OH for $l = 5$	$\mathscr{D}_{\text{sh}}$	0.189	0.163	0.186
Pole figures	$\mathscr{D}_{\text{pf}}$	0.241	0.172	0.228

6.2.2 Learning texture-property relations

The results of the texture-property relations are summarized in Tab. 6.3. The coefficient of determination R^2 is used for comparison, as well as the regression errors MAE_{E} and $\text{MAE}_{\widetilde{\text{R}}}$, which denote the MAE between the true and predicted Young's moduli and $\widetilde{R}$-values. An MAE of Young's moduli ≤ 1000 and of $\widetilde{R}$ of ≤ 0.1 is sufficient in most applications. Additionally, the numbers of input features of the used texture representations are noted. It is shown that texture-property relations with a sufficient prediction quality can be achieved for all texture representations. By considering the orientation histograms, better results are obtained with the soft-assignment $l = 3$. Comparing all texture representations, the estimator quality for GSH for $L = 16$ is best considering R^2 while pole figures achieves the best result considering MAE. The distribution of the property values of the learning data set is shown in Tab. 6.4.

Table 6.3: Results of the investigated textures for learning the texture-property relations. Regression error MAE_{E} is given in [GPa], Regression error $\text{MAE}_{\widetilde{\text{R}}}$ in [-] and Regression score R^2 in [%]. The best results for the metrics are noted in bold.

Texture	Number of features	R^2	MAE_{E}	$\text{MAE}_{\widetilde{\text{R}}}$
Orientation histograms for $J = 512, l = 1$	512	98.2	0.329	0.022
Orientation histograms for $J = 512, l = 3$	512	99.5	0.125	0.012
GSH for $L = 10$	80	99.3	0.152	0.015
GSH for $L = 16$	280	**99.7**	0.087	**0.011**
Pole figures	$64 \times 64 \times 3$	99.6	**0.080**	**0.011**

Table 6.4: Distribution of the property space for the underlying data set. The E-Values are given in [GPa] and the $\widetilde{R}$-Values are given in [-].

Property	Min. Value	Max. Value	Range	Mean	Std.
E_{11}	204	231	27	216	3.14
E_{22}	206	234	28	218	3.42
E_{33}	204	229	25	217	3.27
$\widetilde{R}_{11}$	0.373	4.024	3.651	1.084	0.284
$\widetilde{R}_{22}$	0.342	2.882	2.540	0.964	0.167
$\widetilde{R}_{33}$	0.239	2.210	1.971	0.931	0.194

6.3 Discussion

The findings of this study will be discussed under three aspects: (1) ODF-based texture representations, (2) texture distances measured with representations as discussed in (1), and (3) effects of texture representations on learning of texture-property relations.

ODF-based texture representations: Pole figures are the richest representation of the investigated textures in this study, as they are represented as image data. In this representation, the neighborhood relations of the orientations are represented by the neighboring pixels in the image. Moreover, the visualization of the ODF by pole figures is interpretable by domain experts, which in turn allows them to interpret texture-property relations represented by neural regression networks. In contrast, neighborhood relations are not reflected by orientation histograms, since the order of bins does not reflect the order of orientations, cf. Section 6.1.1. For a comparison of the ODF represented as pole figures and orientation histograms see Fig. 6.2, Fig. 6.3 and Fig. 6.4. Orientation relationships are only represented by orientation histograms when the orientations of the bin centers are included. The representations of ODFs by GSH coefficients does not contain any information about orientation relationships. When using GSHs, neighborhood relations between orientations are implicitly represented by the base functions, which belong to the GSH coefficients. When expanding an ODF using GSH with only low order coefficients, sharp textures can not be represented very well. Besides pole figure images, plots showing sections through the Euler space are also often used to visualize the orientation distribution. However, due to the distortion of the Euler space, this representations is disadvantageous for representing and processing crystallographic texture data.

Texture distance measures: Distances between textures using the *distance measurement evaluation set* described in Section 6.1.3 are evaluated by means of a stepwise process, which pushes the texture further apart from the initial texture at each step. When constantly applying tensile load increments on a material with initial texture $f^{(0)}(g)$ as it is done in the example process (cf. Section 6.1.3), the distance between texture $f^{(0)}(g)$ and $f^{(k)}(g)$ is expected to be smoothly and monotonically increasing with step number k. The GSH-based distance measure as well as the distance measures based on pole figures show this behavior. The monotonic

requirement of the distance measure is also preserved by the Sinkhorn distance of the histogram representation, while it is violated by the Chi-Squared distance measure at soft-assignment $l = 1$ of the histogram representation. The Chi-Squared distance measure is often not smooth (see Fig. 6.5), independent of the soft-assignment parameter, while the Sinkhorn distance is smooth in all of the presented studies (except for the single crystal example shown in Fig. 6.5 (g)).

Another disadvantage of the Chi-Squared distance is its high sensitivity with respect to a shift of orientations from one bin to another one. This is less pronounced for dense histograms (i.e. where there is a non-zero number in each bin) than for sparse histograms. This is best seen by looking at extreme cases of three sparse histograms: Histogram $f_{\mathrm{o}}^{(1)}(g)$ has orientations only in one single bin o_1, otherwise all entries are zero. Histogram $f_{\mathrm{o}}^{(2)}(g)$ has orientations only in an immediate neighbor bin of o_1. Histogram $f_{\mathrm{o}}^{(3)}(g)$ has orientations only in any other bin. The Chi-Squared distances in both cases $\mathscr{D}_{\chi^2}(f_{\mathrm{o}}^{(1)}(g), f_{\mathrm{o}}^{(2)}(g)) = \mathscr{D}_{\chi^2}(f_{\mathrm{o}}^{(1)}(g), f_{\mathrm{o}}^{(3)}(g)) = 2$ and thus maximal (following Eq. (2.42)), no matter how close the unit quaternions of the bins are.

In general, it can be seen that the soft-assignment parameter has a strong effect on the Chi-Squared distance measure, as it incorporates local orientation distances by distributing histogram weights to several bins. This is artificially countering the disadvantage of sensibility against the shift of orientations. The Sinkhorn distance, in contrast, is not affected much by the soft-assignment parameter as it incorporates also the distances between the orientations of the bin representatives (cf. Eq (6.7)). This reflects the neighborhood relations of the orientation space over which the ODF is spanned. Nevertheless, larger values of soft-assignment l yield smoother distance curves. In the case of Chi-Squared distance the usage of $l > 1$ is mandatory for the above mentioned reasons.

In addition, the number of histogram bins of the orientation histogram has a significant effect on the Chi-Squared distance measure, which can be seen in Fig. 6.5 (a) - (c) and Fig. 6.5 (d) - (e): the higher the number of histogram bins, the lower the distance is. For higher number of histogram bins (e.g. $J = 2048$), the Chi-Squared distance even tends to converge towards the Sinkhorn distance. The Sinkhorn distance in-turn is much less sensitive to the number of histogram bins.

Regarding the single crystal texture study (initial texture 3 (n_{oris}=1)), the orientation histogram-based distance measures do not show a smooth behavior, because the crystal rotation after loading cannot be tracked adequately by the orientation histograms. Therefore, the Chi-Squared and the Sinkhorn distance jumps at certain load steps, independently of the number of histogram bins and soft-assignment parameter. However, because of the bin-wise comparison, the Chi-Squared distance reaches quickly it's maximum value. In contrast, the Sinkhorn distance does not reach the maximum value at all and converges almost to the same value for different number of histogram bins and soft-assignments. Moreover, the Sinkhorn distance is closer to the quaternion distance, which can be considered to be the ground truth in this case.

In contrast to the orientation histogram-based distance measures, the pole figure distance shows a rather smooth behavior over the loading steps, as the rotation of the single crystal is tracked by the pole figure. This is due to the generation of pole figures by using a Gaussian kernel which spreads the orientations over several pixels. The distance curves resulting from pole figures are monotonic and smooth. Also the GSH distance shows a smooth behavior in all cases, as it is based on a smooth function approximation (cf. Eq. (6.6)). However, in Fig. 6.6, it can be seen that the GSH distance behaves contrary to the Sinkhorn distance, as it has a slight increase at the beginning of the loading steps and a rather strong increase later. This

characteristic cannot be influenced by the degree L of the GSH distance and does not seem to have a strong effect in this example. It can be seen from Fig. 6.6 that the GSH distance measure has lower resolution power at small texture distance than the Sinkhorn distance measure while it is higher at larger texture distances than Sinkhorn.

When applying the distance measures on experimentally measured textures, naturally, the texture distances do not behave linearly in the chain hot rolling, cold rolling and heat treatment. Heat treated and cold rolled textures are similarly far from the hot rolled texture, what in the sense of the Sinkhorn distance means that the work to change a grey texture to one of these textures is similar.

Estimation of texture-property relations: Besides evaluating different texture representations for texture distance measurement their suitability for modeling texture-property relations was investigated in Section 6.1.2. The results show that sufficient prediction quality can be achieved for all texture representations (see Tab. 6.3), whereas the best results are obtained with the textures represented as pole figures (considering MAE) and GSH with $L = 16$ (considering R^2).

The pole figure representation allows the neural network models to exploit the pixel-represented neighborhood relationships within the textures through convolutional operations, which explains the superior prediction quality. However, the pole figure image data must be transformed by a backbone to a vectorized latent space by means of convolutional and pooling layers, before the neural network head can perform the regression to the property values. The additional backbone layers make processing of pole figures more complex than using histograms as input, which results in higher model training efforts.

In contrast, the orientation histograms can be directly used as input for the neural regression network, reducing the overall network complexity. Investigations were performed for the number of histogram bins $J = 512$ and soft-assignment $l = 1$ and $l = 3$, where the better result is achieved with $l = 3$. The improvement with $l = 3$ can be explained by the approximate consideration of neighborhood relationships by higher order smoothing via soft assignment. The number of bins of $J = 512$ was chosen as compromise between computational cost and representation quality.

The GSH representations can be directly used as input as well. The investigation was performed using GSH representations of the degrees $L = 16$ and $L = 10$, which comprise of 280 and 80 features respectively. A better prediction quality is achieved with $L = 16$, but with $L = 10$ sufficient results are already achieved with its lower number of features, requiring a less complex neural network model. Better prediction results than with orientation histograms can be achieved by GSH-based models with lower number of features due to the implicit representation of orientation neighborhood relations in the base functions of the GSHs.

Chapter 7

Holistic optimization of the process-structure-property linkage

Accelerated materials innovation has become a core research field in Integrated Computational Materials Engineering (ICME) and is pushed forward strongly in materials science and engineering (cf. materials genome initiative (Pablo et al., 2019), European advanced materials initiative (Advanced Materials Initiative 2030, 2022)). Essentially, the properties of such new sustainable, resilient, and high-performance materials depend on the structure of the material, which, in turn, is depending on the manufacturing process. Consequently, designing new materials without taking into account the manufacturing process does not add significant value (Grant, 2013). For an application in industry, groundbreaking technologies for modeling and optimization that cover the entire process-structure-properties linkage are required.

The process–structure–property (PSP) linkage, as depicted in Fig. 7.1, was originally introduced in (Olson, 1997) and describes fundamental relations in materials processing. A basic characteristic of this linkage is its modularity: Each individual link represents a specific identification problem, which includes identifying microstructures for given desired properties (materials design) and finding optimal processing paths for targeted microstructures (process design). These identification problems are typically not well-posed in the sense of Hadamard (Hadamard, 1902), which presents a significant challenge for solution approaches (Agrawal and Choudhary, 2016). However, the non-unique nature of these problems offers an important advantage for processing: It enables a more flexible production as processes can be efficiently guided to manufacture the best reachable microstructure from a set of equivalent microstructures with respect to their properties.

The identification problems mentioned above remain highly complex and are often high-dimensional. To handle this complexity, the usage of machine learning (ML) has shown to be suitable for materials and process design applications (Liu et al., 2017; Mozaffar et al., 2022; Bompas and Sandfeld, 2023). In this study, a novel ML framework is introduced that combines a reinforcement learning (RL)-based process design approach, namely multi-equivalent goal structure-guided processing path optimization (MEG-SGGPO) (Dornheim et al., 2022), with a materials design approach, namely Siamese multitask learning-based optimization (SMTLO) (Iraki et al., 2024a). The framework is tailored to the specifics of process-structure-property optimization problems and therefore constitutes a significant advancement towards accelerated process and materials design.

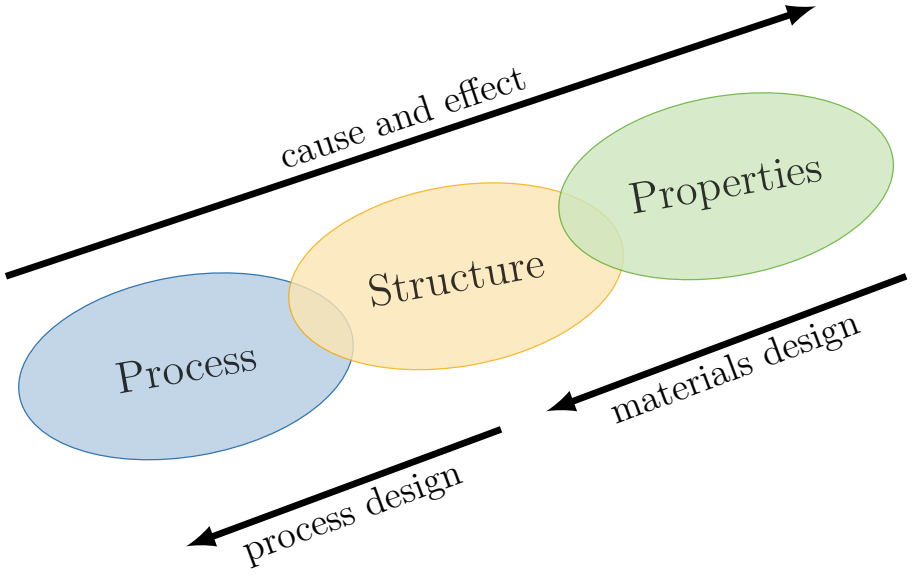

Figure 7.1: Process-structure-properties linkage following (Olson, 1997).

Although each of the approaches has been shown to work well for their individual design problems, the combined usage of both is not investigated yet. This is particularly the aim of the present work, while the approaches are enhanced by using a recently developed novel distance measure for one-point statistics microstructure representations: the Sinkhorn distance $\mathscr{D}_{\mathrm{sh}}$ (Iraki et al., 2024b). Specifically, the Sinkhorn distance highly suitable as it takes into account neighborhood information encoded in histogram-based microstructure representations. The importance of the distance measure can be seen in Fig. 7.2, which depicts the general concept of the approach. This study demonstrates the approach to manufacturing metallic materials with desired elastic and anisotropy properties, influenced by the crystallographic texture that evolves during forming.

7.1 Methods

The methods section is divided into two parts: First, the basics of the structure-guided materials and process design approach investigated in this study are introduced, and second, the domain-specific application case used to evaluate the approach is presented.

7.1.1 Structure-guided materials and process design

General concept

The approach presented in this paper combines two ML approaches for materials design, namely Siamese multitask learning-based optimization (SMTLO), and for process design, namely multi-equivalent goal structure guided processing path optimization (MEG-SGGPO), see Fig. 7.2. The approach starts in the properties space, where a target region of desired properties is defined. Then, the SMTLO approch is applied to identify a set of diverse microstructures that yield material properties inside the target region (Step 1). When having identified this set, MEG-SGGPO identifies the process path that leads to the best reachable microstrucutre (Step

2). Here, the structure space is linking process and properties, and, therefore, is of overall importance also for the approach. A suitable distance measure is one of the core ingredients the approach to work.

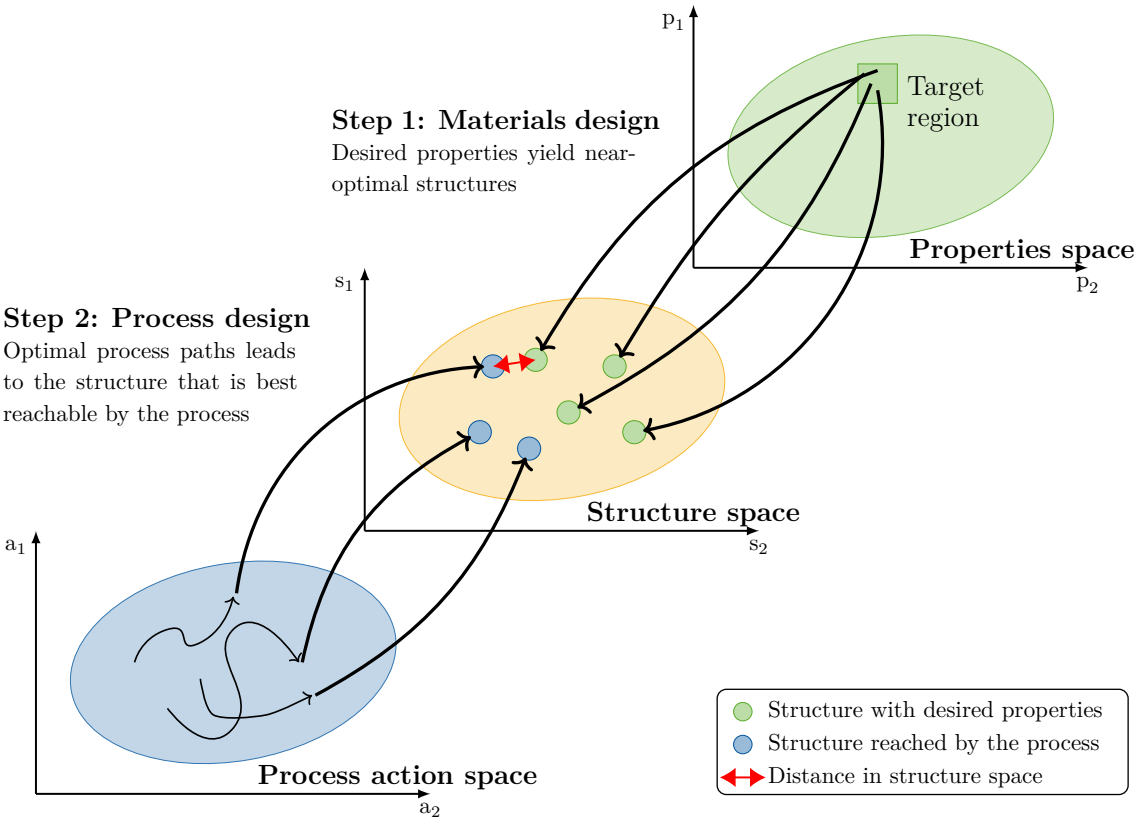

Figure 7.2: General concept for structure-guided materials and process design. The first step (materials design) is addressed by the SMTLO approach, while the second step (process design) is addressed by the MEG-SGGPO approach.

Siamese multitask learning-based optimization

The SMTLO approach described in (Iraki et al., 2024a) consists of a Siamese neural networks-based multitask learning (MTL) model and an optimizer. The latter uses the predictions of the learned neural network model to generate candidate microstructures that are supposed to (i) yield properties inside a defined target region and (ii) are reachable by the underlying manufacturing process, see Fig. 7.3. The general architecture for obtaining a latent feature space is introduced in Section 2.2 and depicted in Fig. 2.10. The details of the SMTLO approach is described in the following. Alternative approaches, that focus only the mapping of structures to properties can be found, for example in (Nguyen et al., 2024).

The MTL model is grounded on an encoder $f_{\text{enc}_{\text{ohg}}}$ that transforms a microstructure representation $\boldsymbol{x}_{\text{ohg}} \in \mathbb{R}^D$ into a lower dimensional latent feature space representation $\boldsymbol{z}_{\text{ohg}} \in \mathbb{R}^Z$

$$\boldsymbol{z}_{\text{ohg}} = f_{\text{enc}_{\text{ohg}}}(\boldsymbol{x}_{\text{ohg}}, \Theta_{\text{enc}_{\text{ohg}}}), \tag{7.1}$$

with the trainable parameters $\Theta_{\text{enc}_{\text{ohg}}}$. The encoder is equipped with three heads, each of which is designed to solve a specific learning task. These are

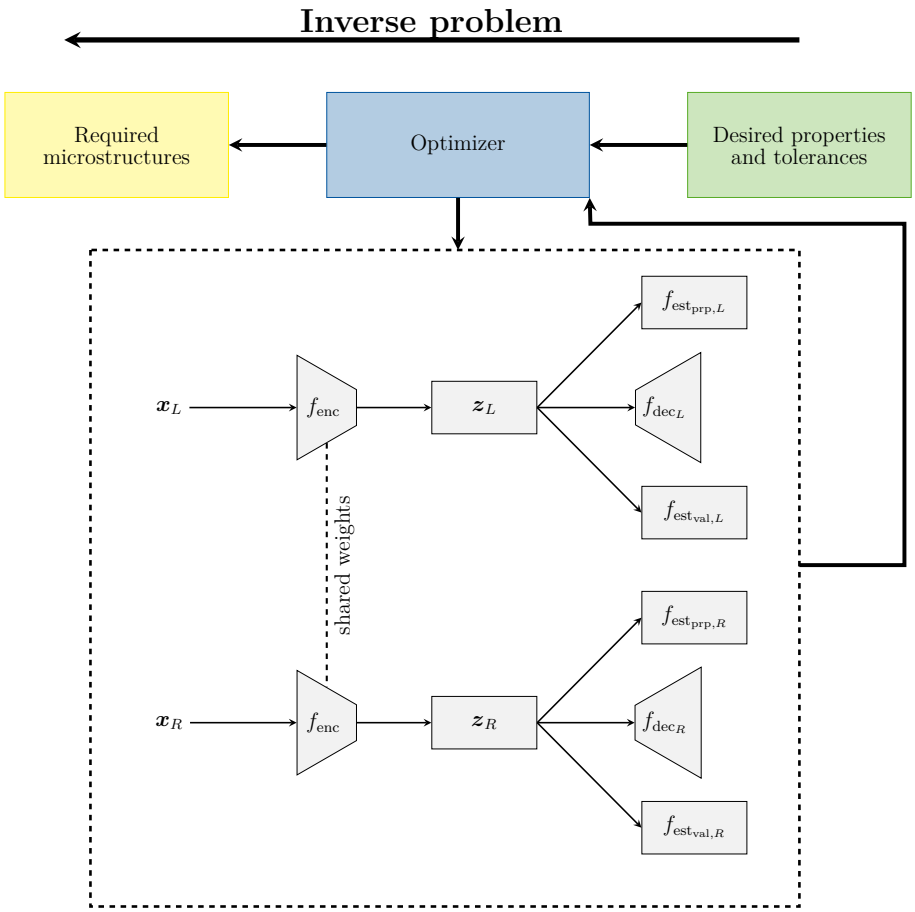

Figure 7.3: The neural networks-based Siamese multitask learning model (bottom) and the optimization part (top). Together, both parts build the SMTLO approach to solve materials design problems. The optimizer generates candidate microstructure $\boldsymbol{z}^*$ thar are to be evaluated using the SMTL model. Predictions and preserved distance in the latent feature space is used to evaluate microstructures in terms of property, validity and diversity.

1. a decoder head $f_{\text{dec}_{\text{ohg}}}$ that reconstructs the microstructure representations $\boldsymbol{x}_{\text{ohg}}$ from $\boldsymbol{z}_{\text{ohg}}$

$$\hat{\boldsymbol{x}}_{\text{ohg}} = f_{\text{dec}_{\text{ohg}}}(\boldsymbol{z}_{\text{ohg}}, \Theta_{\text{dec}_{\text{ohg}}}), \tag{7.2}$$

with the trainable parameters $\Theta_{\text{dec}_{\text{ohg}}}$. The encoder-decoder part is trained using a loss term that minimizes the Sinkhorn distance $\mathscr{D}_{\text{sh}}$ between the original microstructure $\boldsymbol{x}_{\text{ohg}}$ and its reconstruction $\hat{\boldsymbol{x}}_{\text{ohg}}$ (Iraki et al., 2024b)

$$\mathscr{J}_{\text{dec}_{\text{ohg}}}(\boldsymbol{x}_{\text{ohg}}, \hat{\boldsymbol{x}}_{\text{ohg}}) = \mathscr{D}_{\text{sh}}(\boldsymbol{x}_{\text{ohg}}, \hat{\boldsymbol{x}}_{\text{ohg}}) \tag{7.3}$$

2. a prediction head $f_{\text{est}_{\text{prp}}}$ to infer material properties $\hat{\boldsymbol{y}}_{\text{est}_{\text{prp}}}$

$$\hat{\boldsymbol{y}}_{\text{est}_{\text{prp}}} = f_{\text{est}_{\text{prp}}}(\boldsymbol{z}_{\text{ohg}}, \Theta_{\text{prp}_{\text{ohg}}}), \tag{7.4}$$

with the trainable parameters $\Theta_{\text{prp}_{\text{ohg}}}$. The sample-wise loss term is defined by the MSE between true properties $\boldsymbol{y}_{\text{est}_{\text{prp}}}$ and predicted properties $\hat{\boldsymbol{y}}_{\text{est}_{\text{prp}}}$:

$$\mathscr{J}_{\text{est}_{\text{prp}}}(\boldsymbol{y}_{\text{est}_{\text{prp}}}, \hat{\boldsymbol{y}}_{\text{est}_{\text{prp}}}) = \frac{1}{P}\sum_{i=1}^{P}(\text{y}_{\text{est}_{\text{prp}_i}} - \hat{\text{y}}_{\text{est}_{\text{prp}_i}})^2, \tag{7.5}$$

where P denotes the number of properties.

3. an auto-encoder head $f_{\text{est}_{\text{val}}}$ that serves as an anomaly detector to estimate whether a microstructure in its latent representation belongs to the set of known microstructures (defined by the training data) or not:

$$\hat{\boldsymbol{z}}_{\text{ohg}} = f_{\text{est}_{\text{val}}}(\boldsymbol{z}_{\text{ohg}}, \Theta_{\text{val}_{\text{ohg}}}), \tag{7.6}$$

with the trainable parameters $\Theta_{\text{val}_{\text{ohg}}}$. For this auto-encoder, a MSE loss function is used:

$$\mathcal{J}_{\text{est}_{\text{val}}}(\boldsymbol{z}_{\text{ohg}}, \hat{\boldsymbol{z}}_{\text{ohg}}) = \frac{1}{Z}\sum_{i=1}^{Z}(z_{\text{ohg}_i} - \hat{z}_{\text{ohg}_i})^2. \tag{7.7}$$

with Z dimensions of the latent feature space.

The neural networks-based MTL model that is used to solve the above mentioned learning tasks is trained using a combined loss function

$$\mathcal{J}_{\text{MTL}} = \lambda_{\text{est}_{\text{prp}}}\mathcal{J}_{\text{est}_{\text{prp}}} + \lambda_{\text{dec}_{\text{ohg}}}\mathcal{J}_{\text{dec}_{\text{ohg}}} + \lambda_{\text{est}_{\text{val}}}\mathcal{J}_{\text{est}_{\text{val}}} \tag{7.8}$$

with individually weighted loss terms using the parameters $\lambda_{\text{est}_{\text{prp}}}$, $\lambda_{\text{dec}_{\text{ohg}}}$ and $\lambda_{\text{est}_{\text{val}}}$.

To ensure that the original microstructure distance is preserved in the lower dimensional latent feature space, the MTL model is embedded in a Siamese neural network model (Bromley et al., 1993) and is defined as SMTL. This involves training two twin models simultaneously, with shared weights in the encoder, and adding a further loss term that applies to the latent feature space. Note that the two twin models are trained using different input and output vectors, denoted with the subscripts $_L$ and $_R$ in the following. The preservation loss leads to multidimensional scaling (MDS) (see (Kruskal, 1964) and (Cox and Cox, 2008)) and writes

$$\mathcal{J}_{\text{prs}} = \frac{1}{N}\Big(\mathscr{D}_{\text{sh}}(\boldsymbol{x}_{L,\text{ohg}}, \boldsymbol{x}_{R,\text{ohg}}) - \mathscr{D}_{l_1}(\boldsymbol{z}_{L,\text{ohg}}, \boldsymbol{z}_{R,\text{ohg}})\Big)^2, \tag{7.9}$$

with N indicates the number of samples in the data set and the absolute distance $\mathscr{D}_{l_1}$ in the latent feature space

$$\mathscr{D}_{l_1}(\boldsymbol{z}_{L,\text{ohg}}, \boldsymbol{z}_{R,\text{ohg}}) = \frac{1}{Z}\sum_{i}^{Z}|\boldsymbol{z}_{L,\text{ohg}_i} - \boldsymbol{z}_{R,\text{ohg}_i}| \tag{7.10}$$

with Z dimensions of the latent feature space. The subscripts $_L$ and $_R$ are used to distinguish between the two simultaneously trained twins parts. The overall loss function writes

$$\mathcal{J} = \lambda_{\text{est}_{\text{prp}}}\mathcal{J}_{\text{est}_{\text{prp}}} + \lambda_{\text{dec}_{\text{ohg}}}\mathcal{J}_{\text{dec}_{\text{ohg}}} + \lambda_{\text{est}_{\text{val}}}\mathcal{J}_{\text{est}_{\text{val}}} + \lambda_{\text{prs}}\mathcal{J}_{\text{prs}} + \gamma\mathcal{R}(\Theta). \tag{7.11}$$

with the regularization term $\gamma\mathcal{R}(\Theta)$, and the weight λ_{prs} for the preservation loss.

After training the SMTL, a genetic optimizer (Zhang and Sanderson, 2009) is used to search for candidate microstructures in the obtained latent feature space. The optimizer minimizes three cost terms:

$\mathcal{F}_{\text{prp}}$, $\mathcal{F}_{\text{val}}$ and $\mathcal{F}_{\text{div}}$, which are explained in detail in (Iraki et al., 2024a). Briefly described, $\mathcal{F}_{\text{prp}}$ describes the distance between the predicted properties $\hat{\boldsymbol{y}}_{\text{est}_{\text{prp}}}$ for a candidate microstructure $\boldsymbol{z}^*_{\text{ohg}}$ to the target region. $\mathcal{F}_{\text{val}}$ aims to penalize candidate microstructures that are not inside the region of known microstructures. $\mathcal{F}_{\text{div}}$ aims to enforce candidate to microstructures

be as far away as possible from each other in the actual population (in other words as diverse as possible). The overall objective function writes

$$\mathcal{F} = \omega_{\text{prp}}\mathcal{F}_{\text{prp}} + \omega_{\text{val}}\mathcal{F}_{\text{val}} + \omega_{\text{div}}(1 + \mathcal{F}_{\text{div}}), \tag{7.12}$$

with individual weights ω_{prp}, ω_{val}, and ω_{div}.

The generation of a diverse set of microstructures within the microstructure space is a critical factor in the optimization of processes. The presence of a broad and varied set of microstructures ensures that the optimization process is more likely to identify feasible pathways to achieve the desired material properties. When the generated microstructures span a wide range of morphologically distinct solutions, the process optimizer can navigate the process action space more effectively, increasing the probability of reaching at least one of these structures through suitable process parameter adjustments (c.f. Fig. 7.4 (b)). This diversity mitigates the risk of the optimizer being confined to a narrow or suboptimal region of the microstructure space, thereby enhancing the robustness and flexibility of the optimization process (c.f. Fig. 7.4 (a)). Ensuring high diversity in generated microstructures facilitates a more reliable and efficient optimization, enabling targeted microstructure design tailored to specific property requirements.

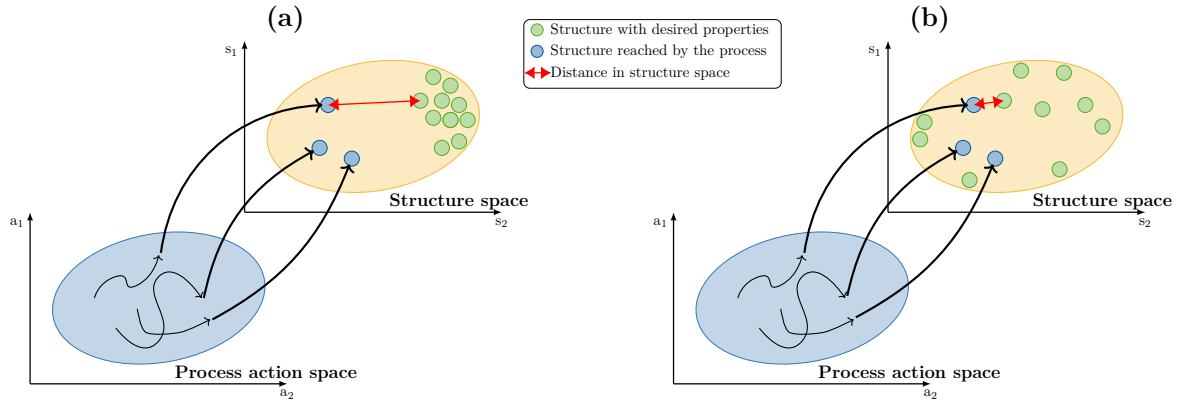

Figure 7.4: Visualization of the diversity of microstructure sets within the structure space. (a) Indicates a narrow region of generated microstructure in the structure space, where (b) presents of a broad and varied set of microstructures.

Multi-equivalent goal structure-guided processing path optimization

In RL, an agent learns to take optimal decisions by interacting with its environment in a sequence of discrete time steps t. The MEG-SGGPO approach (Dornheim et al., 2022) is tailored for efficient learning in the case of multiple equivalent target microstructures. Given a set $\mathcal{G}$ of near-optimal microstructures $\breve{g}^i \in \mathcal{G}$, an initial microstructure g_0, and a microstructure generating process $p(g_t, a_t) = g_{t+1}$, the goal of the approach is to find a sequence of processing parameters $a_t \in A$ that guide the process towards manufacturing the best reachable microstructure $g^* \in \mathcal{G}$.

MEG-SGGPO is based on deep Q-networks (Mnih et al., 2015) and its extensions (Hasselt et al., 2016; Schaul et al., 2016; Wang et al., 2016). Deep Q-networks are derived from Q-learning, where the so-called Q-function is approximated from data gathered by the RL agent while following a policy $a = \pi(g)$. The Q-function models the expected sum of future reward signals for state action pairs (g_t, a_t). In MEG-SGGPO the Q-function is generalized to also take

the goal microstructure into account and is, in its recursive form, defined as

$$\mathcal{Q}(g_t, a_t, \breve{g}^i) = \mathbb{E}_{P,\pi}\Big[R(g_t, g_{t+1}, \breve{g}^i) + \max_{a_{t+1}\in A} \mathcal{Q}(g_{t+1}, a_{t+1}, \breve{g}^i)\Big], \tag{7.13}$$

where $R(g_t, g_{t+1}, \breve{g}^i)$ is a per goal microstructure pseudo reward function

$$R(g_t, g_{t+1}, \breve{g}^i) = \frac{1}{\mathscr{D}(g_{t+1}, \breve{g}^i)} - \frac{1}{\mathscr{D}(g_t, \breve{g}^i)}, \tag{7.14}$$

which is based on a microstructure distance function $\mathscr{D}$.

The process design task is a two-fold optimization problem to (i) identify the best reachable goal microstructure $\breve{g}^{i*}$ and (ii) learn the optimal policy π^*. MEG-SGGPO addresses this by using the generalized functions defined above in a nested RL procedure. At the beginning of each episode (i.e. an execution of the process during learning), the agent uses the generalized Q-function to identify the targeted microstructure $\breve{g}^{i'}$ from $\mathcal{G}$ and determines the processing parameters during the episode. In a first step, an estimation of the best reachable goal microstructure $\breve{g}^{i*}$ is extracted from the current $\mathcal{Q}$ estimation by

$$\breve{g}^{i*} = \arg\max_{\breve{g}^i\in\mathcal{G}} \Big[\mathcal{V}(g_0, \breve{g}^i) + \frac{1}{\mathscr{D}(g_0, \breve{g}^i)}\Big], \tag{7.15}$$

where the generalized state-value function $\mathcal{V}$ is defined as

$$\mathcal{V}(g, \breve{g}^i) = \max_{a\in A} \mathcal{Q}(g, a, \breve{g}^i). \tag{7.16}$$

The targeted microstructure is chosen in an ϵ-greedy approach (Sutton et al., 1992), where the agent chooses random targets and processing parameters in an ϵ fraction of the cases ($0 \leq \epsilon \leq 1$) and optimizes targets and processing parameters in the remaining cases. The optimal processing path is identified simultaneously in an inner loop by using the Q-learning approach.

7.1.2 Application case

Crystallographic texture optimization

To study it's functionality, the structure-guided materials and process design approach is applied to a texture optimization problem in a metal forming process. The process–structure–property (PSP) linkage adapts to cold forming operations that change the crystallographic texture of a material, which significantly affects elastic and plastic properties.The underlying metal forming process simulation is described in the following section, as well as the determination of the elastic and plastic properties to be targeted.

Metal forming process simulation

This study utilizes the metal forming process simulation described in (Dornheim et al., 2022). The simulation applies a deformation $\hat{\boldsymbol{F}}$

$$\hat{\boldsymbol{F}} = \boldsymbol{R}\widetilde{\boldsymbol{F}}\boldsymbol{R}^\top, \tag{7.17}$$

with $\boldsymbol{R}$ being a rotation matrix that describes one out of 25 possible loading directions. The deformation $\widetilde{\boldsymbol{F}}$ is defined using orthogonal basis vectors $\boldsymbol{e}_i$ and the operator $\otimes$ for the outer product

$$\widetilde{\boldsymbol{F}} = \widetilde{F}_{11}\boldsymbol{e}_1 \otimes \boldsymbol{e}_1 + \widetilde{F}_{22}\boldsymbol{e}_2 \otimes \boldsymbol{e}_2 + \widetilde{F}_{33}\boldsymbol{e}_3 \otimes \boldsymbol{e}_3 \tag{7.18}$$

with $\widetilde{F}_{11}$ corresponding to 10% strain increments. $\widetilde{F}_{22}, \widetilde{F}_{33}$ are adjusted such that the stresses are in balance. The metal forming process consists of seven subsequent loading steps, each in a separate loading direction. For this study, 76,980 random process paths were conducted to generate the training data for the SMTLO approach.

The underlying material model is a crystal plasticity model of Taylor-type (Kalidindi et al., 1992). The volume averaged stress for n_{oris} crystals with different orientations is calculated by

$$\overline{\boldsymbol{T}} = \frac{1}{V} \sum_{i}^{n_{\text{oris}}} \boldsymbol{T}^{(i)} V^{(i)}, \tag{7.19}$$

with the total volume V, the individual volume of each crystal $V^{(i)}$, and the Cauchy stress tensor $\boldsymbol{T}^{(i)}$.

With the multiplicative decomposition of the deformation gradient in its elastic and plastic part

$$\boldsymbol{F} = \boldsymbol{F}_{\text{e}} \cdot \boldsymbol{F}_{\text{p}}, \tag{7.20}$$

and the conversion formula for the stress tensor in the intermediate configuration $\boldsymbol{T}^*$

$$\boldsymbol{T}^* = \boldsymbol{F}_{\text{e}}^{-1} \cdot (\det(\boldsymbol{F}_{\text{e}})\boldsymbol{T}) \cdot \boldsymbol{F}_{\text{e}}^{-\top}, \tag{7.21}$$

the Cauchy stress tensor is derived using

$$\boldsymbol{T}^* = \frac{1}{2}\mathbb{C} : (\boldsymbol{F}_{\text{e}}^{\top} \cdot \boldsymbol{F}_{\text{e}} - \boldsymbol{I}), \tag{7.22}$$

where $\boldsymbol{I}$ denotes the second order identity tensor and $\mathbb{C}$ the fourth order elastic stiffness tensor.

The evolution of the plastic deformation is described using the plastic part of the velocity gradient

$$\boldsymbol{L}_{\text{p}} = \dot{\boldsymbol{F}}_{\text{p}} \cdot \boldsymbol{F}_{\text{p}}^{-1} = \sum_{\alpha} \dot{\gamma}^{(\alpha)} \boldsymbol{m}^{(\alpha)} \otimes \boldsymbol{n}^{(\alpha)}, \tag{7.23}$$

with the slip rates $\dot{\gamma}^{(\alpha)}$ on slip system α that is defined by the slip plane normal $\boldsymbol{n}^{(\alpha)}$ and the slip direction $\boldsymbol{m}^{(\alpha)}$. The slip rates are calculated by a phenomenological power law. The crystal reorientation is calculated by applying a rigid body rotation derived from the polar decomposition of $\boldsymbol{F}_{\text{e}}$ to the original orientation, see (Ling et al., 2005; Iraki et al., 2024b). For the metal forming process simulation, the same material model parameters have been used as in (Dornheim et al., 2022).

For the purpose of this study, the above described material model is used to evaluate the Young's modulus E and an anisotropic property $\widetilde{R}$ in three orthogonal directions after a process run. The Young's modulus is calculated using the slope of the stress-strain curve in the elastic regime, that results after applying uniaxial tension. $\widetilde{R}$ is inspired by the Lankford coefficients in sheet metal forming and is calculated as the ratio between transverse strain and the strain in the loading direction in the plastic regime, also after applying uniaxial loading.

Crystallographic texture representation and texture distance measure

Texture is represented by using the histogram-based description introduced by Dornheim et al. (Dornheim et al., 2022). The orientation histogram (here, a soft-assignment factor of 3 is used) consists of 512 orientation bins that are nearly uniformly distributed in the cubic-triclinic fundamental zone. The bin orientations were created using the software *neper* (Quey et al.,

2011; Quey et al., 2018). In contrast to the original MEG-SGGPO and SMTLO approach, in this work, crystallographic texture distances are measured using the Sinkhorn distance that is applied to the histogram representations (Iraki et al., 2024b). Specifically, the Sinkhorn distance is an efficient implementation of the EMD that measures the least amount of work necessary to transform one histogram into the other (Rubner et al., 2000; Cuturi, 2013). Therefore, local orientation distances (Huynh, 2009) encoded in the histograms are taken into account, in contrast to the originally proposed Chi-Squared distance, which is basically a bin-wise comparison of two histograms.

7.2 Results

7.2.1 Materials and process design task

In the following sections, it is shown how near-optimal crystallographic textures are identified for given elastic and anisotropy properties using the Siamese multitask learning-based optimization (SMTLO) approach. The properties are in particular the Young's moduli E_i and anisotropy measures $\widetilde{R}_i$, both in three orthogonal directions. E_i is given by the slope in the elastic regime, while $\widetilde{R}_i$ is calculated by the ratio between transverse and tensile strain in the plastic regime when applying uniaxial tension. Both E_i and $\widetilde{R}_i$ are determined in three orthogonal directions at the material point at the end of the process, resulting in the six properties: E_{11}, E_{22}, E_{33}, $\widetilde{\mathrm{R}}_{23}, \widetilde{\mathrm{R}}_{12}, \widetilde{\mathrm{R}}_{13}$. The basis for applying the SMTLO approach is a data set of 76,980 samples, composed of textures and corresponding properties.

7.2.2 Validation of the siamese multi task learning model

The Siamese multitask learning (SMTL) model is realized via feedforward neural networks with *tanh* activation functions (c.f. Eq. (2.10)) and are implemented based on the Pytorch API (Paszke et al., 2019). For hyperparameter optimization the random search method (Bergstra and Bengio, 2012) is utilized using 5-fold cross-validation. The following hyperparameters have been chosen: The Glorot Normal method (Glorot and Bengio, 2010) is used for weight initialization and the Adam optimizer (Kingma and Ba, 2015) is used with the following parameters: learning rate = 0.001, weight decay = 10^{-6}, batch size = 128. The SMTL model is build on the basis of a 16-dimensional latent feature space. The SMTL model is trained for 100 epochs, where the best intermediate result of the test set is retained to prevent over fitting and to apply early stopping (Prechelt, 1998). Before the model is trained, the loss terms are scaled to values between 0 and 1 in order to make them comparable. The following weights for the scaled loss terms in Eq. (7.11) were chosen based on hyper parameter optimization: $\lambda_{\mathrm{dec}_{\mathrm{ohg}}} = 0.20$, $\lambda_{\mathrm{est}_{\mathrm{prp}}} = 0.06$, and $\lambda_{\mathrm{est}_{\mathrm{val}}} = 0.04$, and $\lambda_{\mathrm{prs}} = 0.70$. The loss values resulting from the model training are collected for both the test and the training dataset for each individual epoch and are shown for each task separately in Fig. 7.5. It can be seen that for tasks (a) Validity, (b) Reconstruction, and (c) Regression the largest amount of the loss is minimized up to epoch 50. On the other hand, for task (d) Distance Preservation the loss is still minimized after epoch 50.

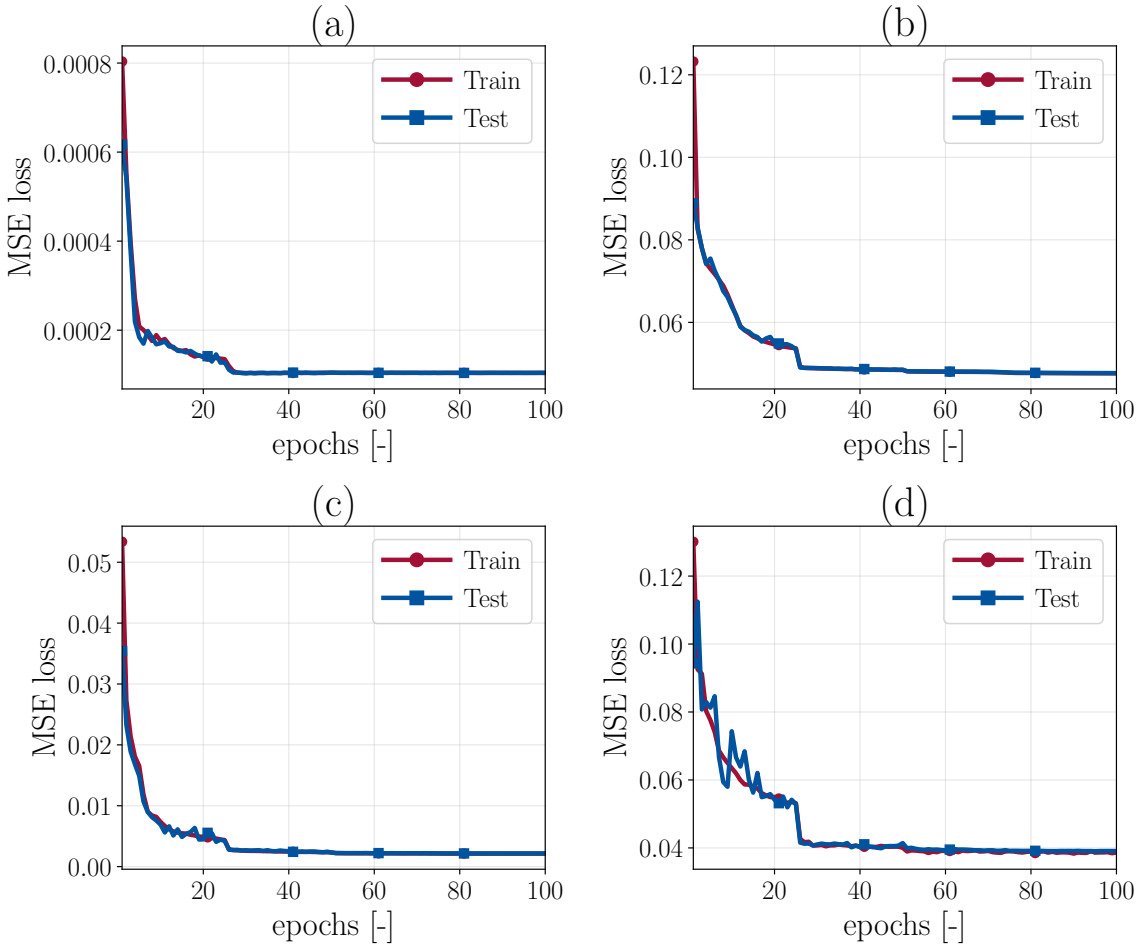

Figure 7.5: Visualization of the loss curves of the Siamese multitask learning model training for the test (indicated in blue) and the training dataset (indicated in red) individually for the tasks (a) validity, (b) reconstruction, (c) regression, and (d) distance preservation. The loss values are represented on the logarithmic scale.

The results for the properties prediction are given by the the MAE between the true and predicted Young's moduli and $\widetilde{R}$-values: $\text{MAE}_{\text{E}} = 0.368$ [GPa] and $\text{MAE}_{\widetilde{\text{R}}} = 0.032$ [-]. Tab. 7.1 depicts the deviation for all properties separately, with the overall data set distribution highlighted in Tab. 7.2. Fig. A.26 illustrates the equality between the true and predicted values. The results indicate that the $\widetilde{R}$-values exhibit a slightly higher R^2 score, making them more challenging to train.

Table 7.1: The results for the properties prediction. The Young's moduli E are given in GPa, the anisotropy measures $\widetilde{R}$ in [−].

Property	E_{11}	E_{22}	E_{33}	$\widetilde{R}_{23}$	$\widetilde{R}_{12}$	$\widetilde{R}_{13}$
MAE	368.75	360.44	375.56	0.040	0.030	0.026
R^2	97.78	98.11	97.84	95.73	95.45	95.14

Table 7.2: Distribution of the property space for the underlying data set. The E-Values are given in [GPa] and the $\widetilde{R}$-Values are given in [−].

Property	E_{11}	E_{22}	E_{33}	$\widetilde{R}_{23}$	$\widetilde{R}_{12}$	$\widetilde{R}_{13}$
Min. Value	204	206	204	0.373	0.342	0.239
Max. Value	231	234	229	4.024	2.882	2.210
Range	27	28	25	3.651	2.540	1.971
Mean	216	218	217	1.084	0.964	0.931
Std.	3.14	3.42	3.27	0.284	0.167	0.194

The result of the distance preservation is measured by the coefficient of determination R^2, between the Sinkhorn distance of two input textures and the l_1 distance of their corresponding latent feature vectors: $R^2\big(\mathscr{D}_{\text{sh}}(\boldsymbol{x}_{L,\text{ohg}}, \boldsymbol{x}_{R,\text{ohg}}), \mathscr{D}_{l_1}(\boldsymbol{z}_{L,\text{ohg}}, \boldsymbol{z}_{R,\text{ohg}})\big) = 90.22[\%]$. The result of the distance preservation is visualized in Fig. A.28 for distances in the original space and distances of their corresponding feature vectors. It can be seen that the l_1 distances in the z-space are slightly lower at certain points, but the distances can generally be preserved approximately in the latent feature space. A projection of the latent feature space to a two-dimensional space was conducted using the UMAP algorithm (McInnes et al., 2018). The resulting distribution is presented in Fig. A.27, with the data color-coded by the validity loss and regression loss. Additionally, the distributions of the validity loss and regression loss are also represented in Fig. A.27.

7.2.3 Materials design using siamese multi task learning optimization

For the cost function of the optimizer (defined in Eq. (7.12)) is configured with the weights $\omega_{\text{prp}} = 0.75$, $\omega_{\text{val}} = 0.05$, and $\omega_{\text{div}} = 0.20$. The terms $\mathcal{F}_{\text{prp}}$ and $\mathcal{F}_{\text{div}}$ are scaled to values between 0 and 1 based on the initial population. The optimization is performed for 100 generations with a fixed population size of 25.

2D projections of the training data distribution, generated by the metal forming process simulation, are shown in Fig. 7.6 including the regions delineating the desired properties (target regions). The objective of this study is to undertake the material design and process design steps for two target regions, which have been defined in different regions. The first one lies in a sparsely populated region and is labeled as *Target Region 1*. The second one lies in a more densely populated region and is labeled as *Target Region 2*. The center of each target region is listed in Tab. 7.3.

Table 7.3: Center points of the two target regions (TR). The Young's moduli E are given in GPa, the anisotropy measures $\widetilde{R}$ in [−].

TR	E_{11}	E_{22}	E_{33}	$\widetilde{R}_{23}$	$\widetilde{R}_{12}$	$\widetilde{R}_{13}$
1	214.0	214.0	221.0	0.650	0.685	0.885
2	217.0	218.5	219.0	1.000	0.900	0.900

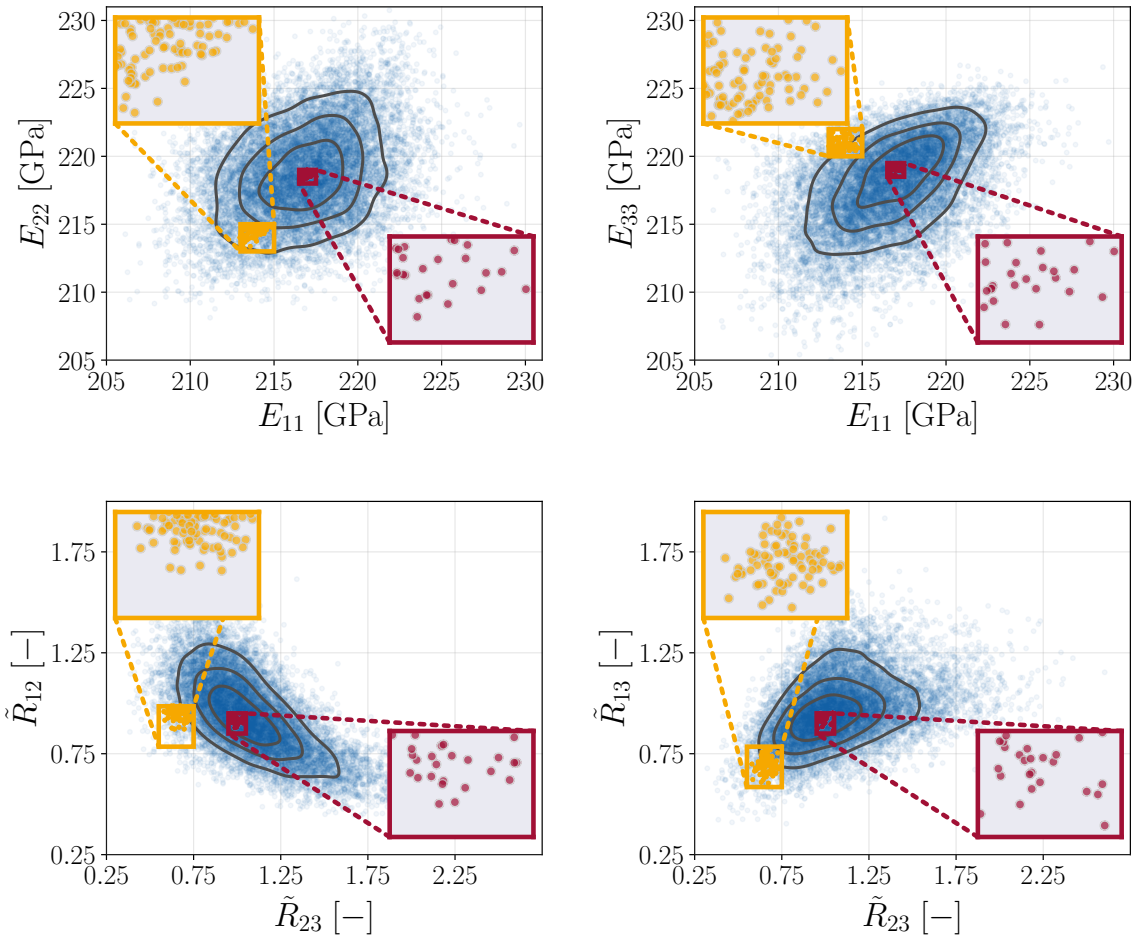

Figure 7.6: Projections of the properties space showing the data distribution and the Target Region 1 (indicated in orange color) and Target Region 2 (indicated in red color), respectively. The distribution of the underlying point cloud is displayed in blue, while the orange and red dots mark data points that are already located inside the target regions. The black isolines indicate regions with the same point cloud density.

The target regions are defined by the addition of a tolerance, which yields a sufficiently small properties window from an engineering perspective. Target Region 1 is defined by the addition of a tolerance of ± 1 GPa for the Young's moduli and ± 0.1 for the $\widetilde{R}$-values. For Target Region 2 a tolerance of ± 0.5 GPa for the Young's moduli and ± 0.05 is added to the center, cause it is located in a more densely populated region. It is of particularly interest to ascertain whether the optimizer is capable of identifying a set of microstructures within a smaller target region. In Fig. 7.6, training data points that are already located inside the target regions are highlighted. These sets are used as benchmark sets for the materials design approach. The benchmark sets for Target Region 1 and Target Region 2 contain 80 and 25 textures, respectively.

When the SMTLO approach is applied (utilizing the trained SMTL model as previously described), the objective in Eq. (7.12) is optimized by the optimizer over the entirety of the generations. The optimizer is initialized with structures that are widely distributed in the property space (generation 0, see Fig. A.24 for Target Region 1 and Fig. A.25 for Target Region 2). In these figures, the distance to the target regions (property loss) is represented by the color bar. The optimization of the structures can be observed in subsequent generations, as

the properties approach the target region and the degree of property loss is reduced. It is evident that after merely 10 generations, the properties have exhibited a notable convergence with the target regions. A detailed examination of generation 100 reveals that a significant proportion of the properties are already present within the designated target regions. The remaining properties are distributed in close proximity to this target region. During the optimization over the entirety of the generations, the generated structures are collected. Only those candidate solutions from the population that fulfill the following two requirements are added to this collection: (a) they have the desired properties, that is, they are within the target region, and (b) they are reachable by the process, that is, they do not exceed the validity threshold.

Upon completion of the optimization process and collecting the generated structures that satisfy the requisite criteria, sets of 175 (for Target Region 1) and of 87 (for Target Region 2) near-optimal textures were identified. To identify a preferably diverse set of textures, their pairwise distances are depicted in Fig. 7.7 and compared to the benchmark sets. As one can easily see, the SMTLO approach is able to find a set of textures that differ more to each other than the benchmark set for both target regions.

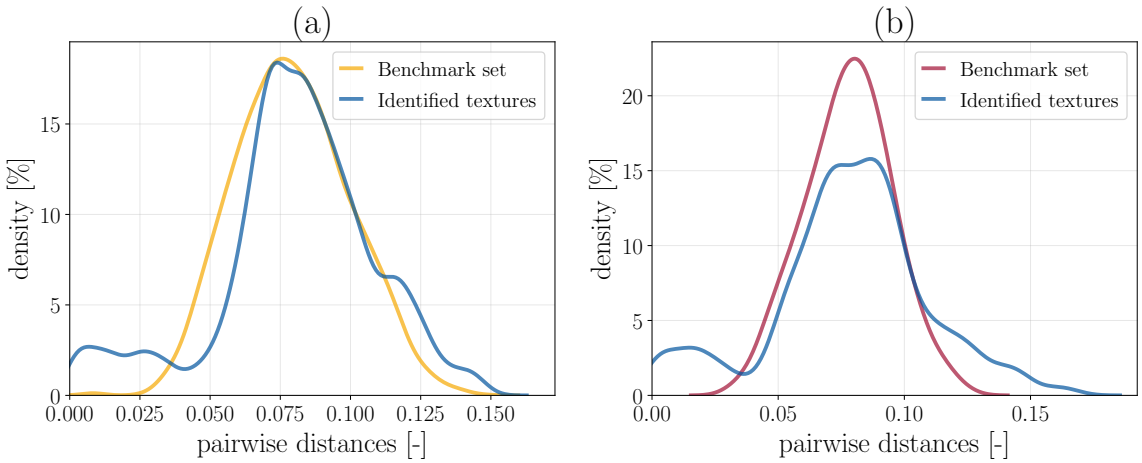

Figure 7.7: Density of pairwise distances between textures belonging to the sets of identified textures and between textures belonging to the benchmark sets. (a): Comparison of Target Region 1, where the identified textures are indicated in blue color an the textures of the benchmark sets are indicated in orange color. (b): Comparison of Target Region 2, where the identified textures are indicated in blue color an the textures of the benchmark sets are indicated in red color.

For the subsequent process design step, ten goal textures were selected from the sets of identified textures that differ strongly from each other (Bauckhage et al., 2021) for Target Region 1 and Target Region 2 separately. The distribution of the chosen goal textures in properties space is shown for Target Region 1 in Fig. 7.8 and for Target Region 2 in Fig. 7.9. In case of both target regions, the properties of the goal textures are mainly located inside the target region, with some exceptions that are tolerated for now and explain later in Section 7.3. To show the differences between the goal textures, four textures are depicted exemplary as pole figure plots in Fig 7.10 for Target Region 1 and Target Region 2. While the pole figure intensities are similar for all textures, the shape of the represented orientation distribution differs strongly regarding the target regions.

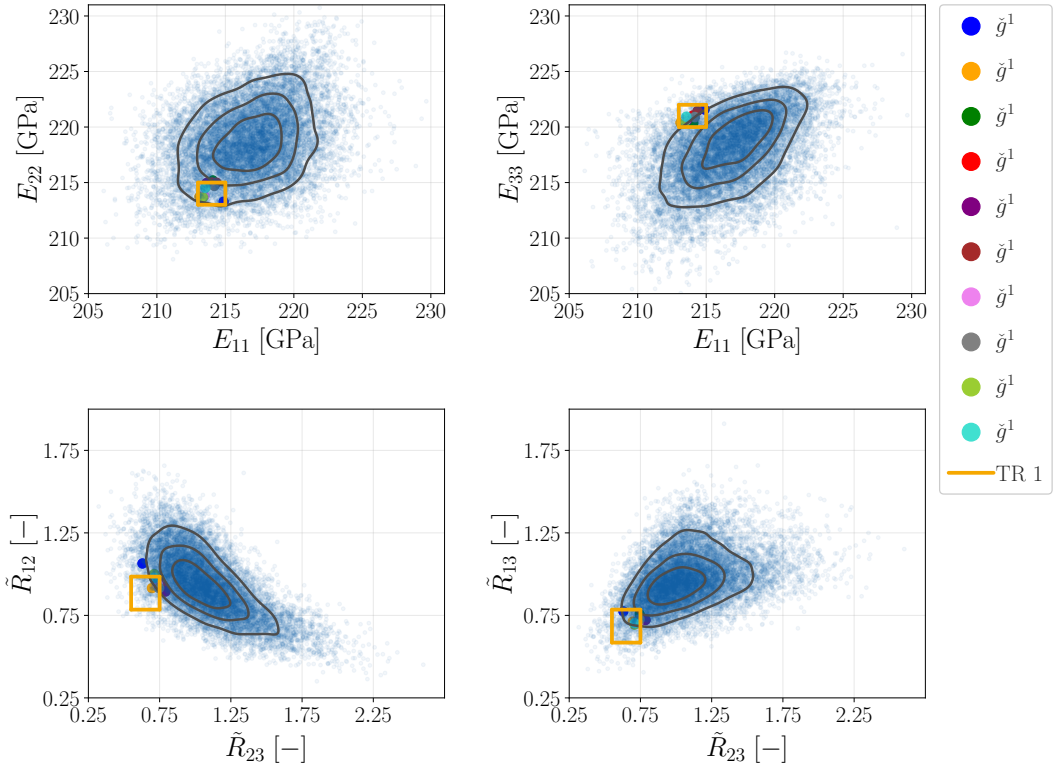

Figure 7.8: Projections of the properties space showing the Target Region 1 (TR 1) and the calculated properties of the ten chosen goal textures $\breve{g}^0$ to $\breve{g}^9$.

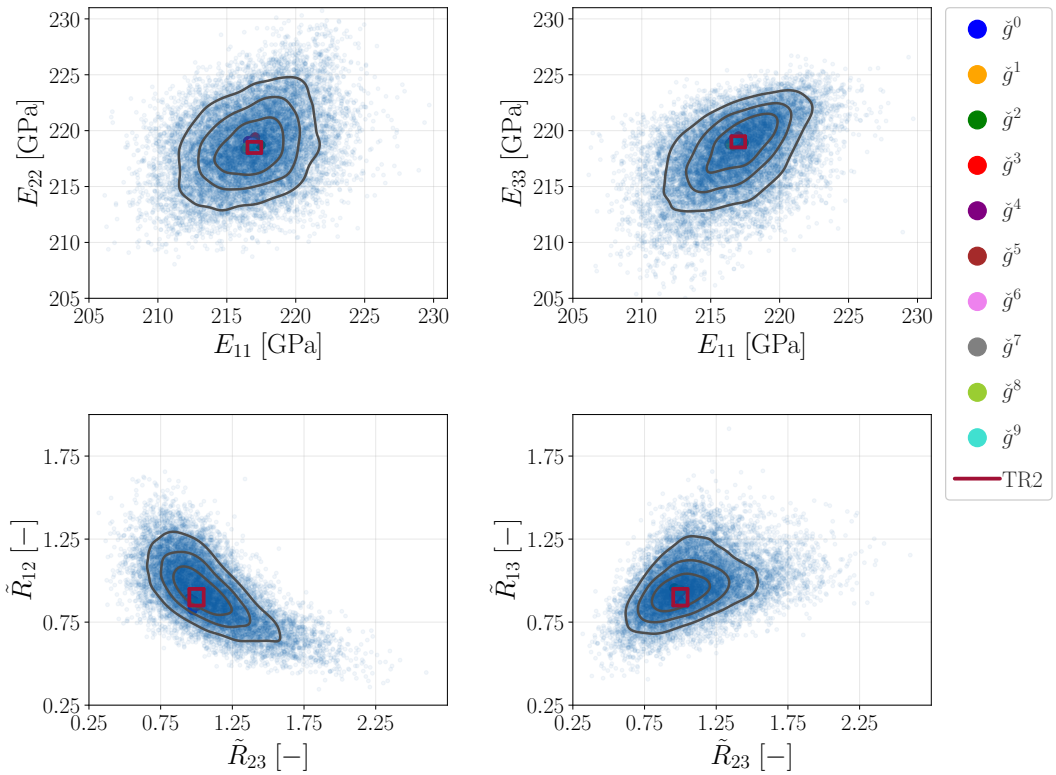

Figure 7.9: Projections of the properties space showing the Target Region 2 (TR 2) and the calculated properties of the ten chosen goal textures $\breve{g}^0$ to $\breve{g}^9$.

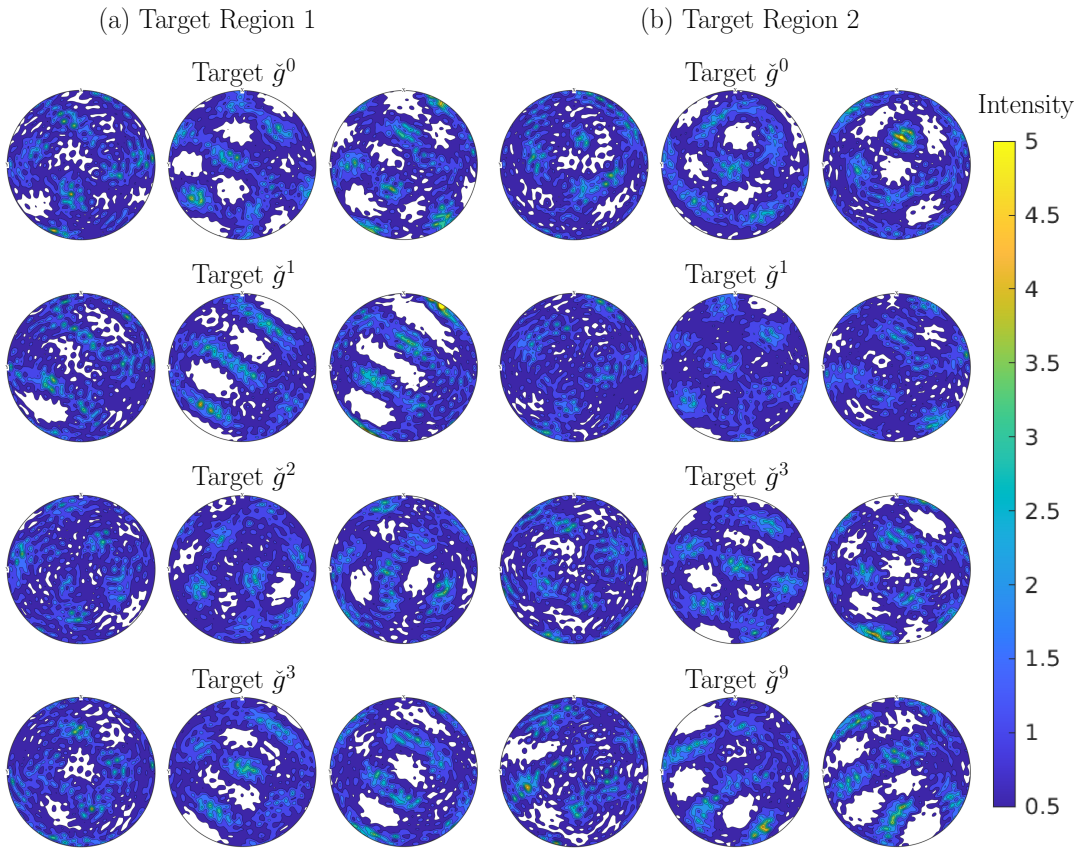

Figure 7.10: Four exemplary textures of the ten chosen goal textures depicted as pole figure plots: (001), (110), (111) for (a) Target Region 1 (b) and Target Region 2.

7.2.4 Process design using multi-equivalent goal structure-guided processing path optimization

Experiments are conducted with the following hyperparameters: Deep Q-learning as described in Section 7.1.1 as basic algorithm where the target-network is updated every $n_\theta = 50$ time-steps. Q-networks with hidden layer sizes of [128, 256, 256, 128], layer normalization and rectified linear unit (ReLU) activation functions. The learning process starts after 100 control-steps. The networks are trained after each control-step with a mini-batch of size 32. The Adam optimizer (Kingma and Ba, 2015) is used for neural network training, with a learning rate of $5e^{-4}$. An ϵ-greedy policy, with an initial exploration rate $\epsilon_0 = 0.5$ and the final exploration-rate $\epsilon_f = 0.0$, with $n_\epsilon = 390$.

Once a diverse set of crystallographic textures has been identified, multi-equivalent goal structure-guided processing path optimization (MEG-SGGPO) is used to guide the metal forming process to the best reachable texture. For solving the identification problem, the MEG-SGGPO approach is allowed to conduct 400 process runs, so-called episodes. The evolution of the distance between the produced and the chosen goal texture is depicted over the episodes in Fig. 7.11 for Target Region 1 and Target Region 2. Regarding Target Region 1, the reinforcement learning (RL) agent initially attempts to identify processing paths that target randomly selected goal textures. As the learning process progresses, the agent focuses increasingly on producing goal texture $\check{g}^2$. In the case of Target Region 2, the agent initially directs its atten-

tion towards the production of goal $\check{g}^0$. As the learning process unfolds, it discerns superior processing pathways for producing of goal texture $\check{g}^1$, and subsequently, goal texture $\check{g}^3$.

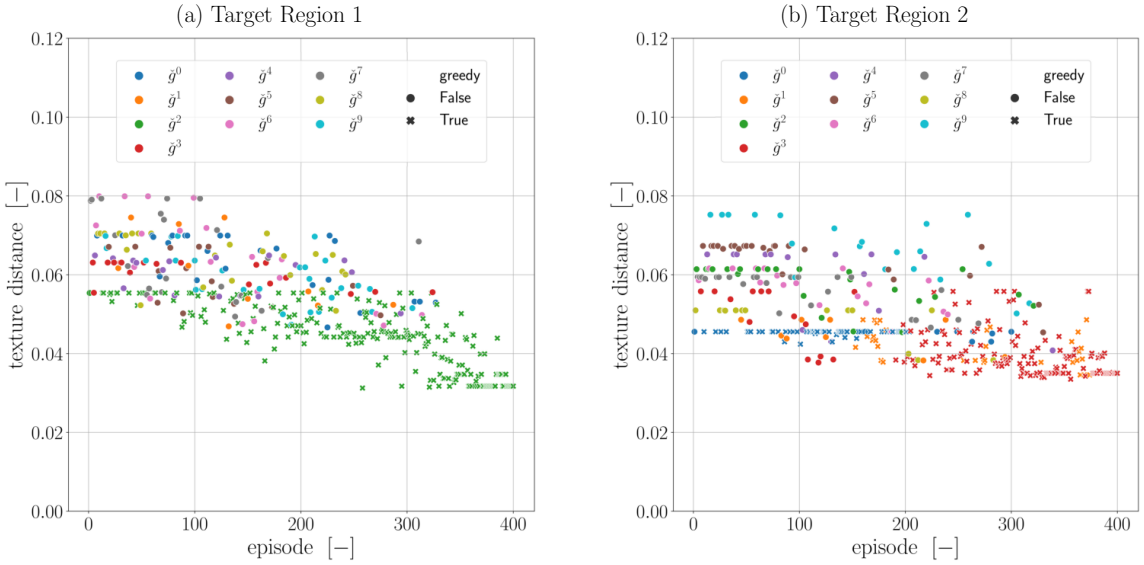

Figure 7.11: Sinkhorn texture distance $\mathscr{D}_{\mathrm{sh}}$ between the produced and the goal texture over the episodes of the MEG-SGGPO approach. The color indicates the targeted texture. The evolution is shown for Target Region 1 (left) and Target Region 2 (right).

For comparison, Fig. 7.12 presents pole figure plots of the goal texture $\check{g}^2$ and the produced texture $\check{g}^{2^*}$ for Target Region 1 and the goal texture $\check{g}^3$ and the produced texture $\check{g}^{3^*}$ after 400 episodes. Visually, for both target regions, the produced texture is highly similar to the goal texture with intensity peaks at the same positions and of a similar magnitude. This also holds for areas not covered by orientations at all. To quantify the effectiveness of the approach, the properties resulting from the produced crystallographic textures are computed and compared with the desired ones (c.f. Tab. 7.4). The results for Target Region 1 show that five of the properties (E_{11}, E_{22}, E_{33}, $\widetilde{R}_{12}$ and $\widetilde{R}_{13}$) are very close to the target region, while $\widetilde{R}_{23}$ lies inside. For Target Region 2, the properties E_{22}, E_{33} and $\widetilde{R}_{23}$ lie inside, while the remaining are very close to the target region.

Table 7.4: Properties of the produced crystallographic textures $\check{g}^{2^*}$ and $\check{g}^{3^*}$ and the distance to their respective Target Region 1 and Target Region 2. The Young's moduli E are given in GPa, the anisotropy measures $\widetilde{R}$ in [–].

	E_{11}	E_{22}	E_{33}	$\widetilde{R}_{23}$	$\widetilde{R}_{12}$	$\widetilde{R}_{13}$
TR 1						
Property Value	217.5	216.6	222.2	0.744	0.831	1.058
Distance to TR	2.5	1.6	0.3	0	0.046	0.073
TR 2						
Property Value	219.8	218.4	218.8	0.990	1.056	1.059
Distance to TR	2.3	0	0	0	0.106	0.109

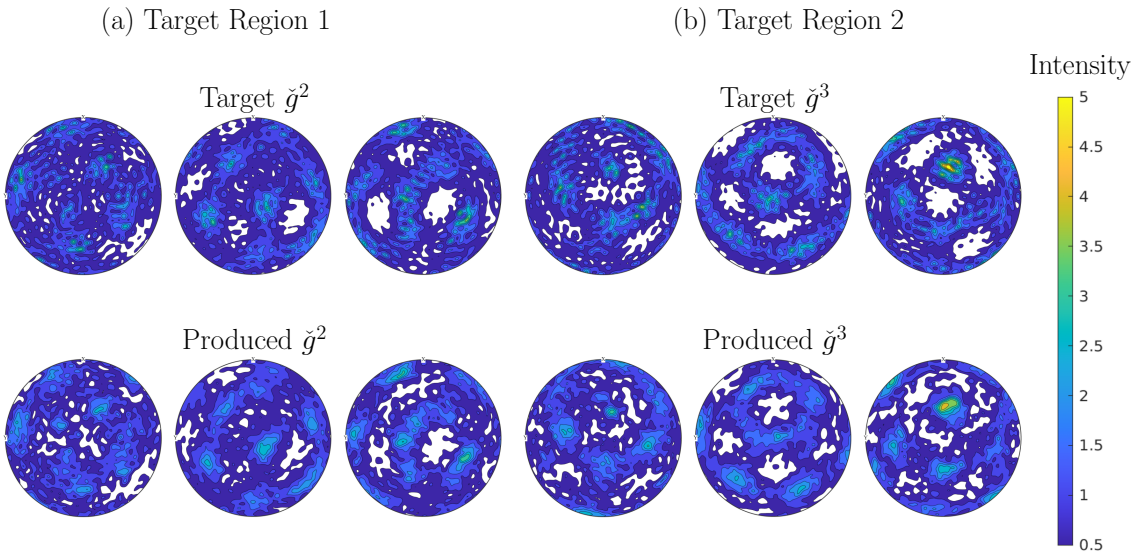

Figure 7.12: Pole figure plots (001), (110), (111) of the targeted goal texture (top) and the produced texture (bottom), for (a) Target Region 1 and (b) Target Region 2.

In summary, the sequential application of the SMTLO and the MEG-SGGPO approach successfully produced a crystallographic texture with desired properties in a simulated metal forming process. The resulting properties fall within the acceptable range from an engineering point of view, albeit at the border of the target region defined in properties space.

7.3 Discussion

The results presented demonstrate the effectiveness of the SMTLO approach in identifying sets of near-optimal crystallographic textures that exhibit desired properties in a given target region. The set of textures identified by the approach is shown to be more diverse than the benchmark set obtained from the training data set. However, the properties of the identified textures are not completely inside the target region, as can be seen in Fig. 7.8 and Fig. 7.9.

There are two reasons for this: First, the formulation of the optimization objective in the SMTLO approach allows for a trade-off between (i) finding crystallographic textures with properties inside the target region, (ii) identifying possibly diverse textures and, (iii) guaranteeing that texture can be produced by the process under consideration. Second, due to prediction errors of the underlying ML model, the SMTLO approach identifies textures whose predicted properties lie inside the target region, while some of the true properties (calculated by the numerical simulation on the basis of the reconstructed textures) slightly lie outside. Both of these issues can be addressed by modifying the objective function of the SMTLO optimizer, defined by Eq. (7.12). For instance, targeting the center of the properties region, instead of its bounds can mitigate both issues. Additionally, the second issue can be addressed by enhancing the ML model, for example, by increasing the amount of training data.

Taking a diverse subset of the identified textures as goal textures, the MEG-SGGPO approach guides the metal forming process closely to one of the chosen goal textures. Although the RL agent was not able to reproduce one of the goal textures exactly (which is challenging due to the many possible processing paths), the measured distance in case of Target Region

1 between the targeted texture $\check{g}^2$ and the produced one $\check{g}^{2^*}$ is sufficiently low ($\mathscr{D}_{\mathrm{sh}} = 0.031$) when relating it to the distance to the nearest neighbor in the training data set ($\mathscr{D}_{\mathrm{sh}} = 0.033$) and to the furthest data point ($\mathscr{D}_{\mathrm{sh}} = 0.141$). When regarding Target Region 2, the measured distance between the targeted texture $\check{g}^3$ and the produced texture $\check{g}^{3^*}$ is again sufficiently low ($\mathscr{D}_{\mathrm{sh}} = 0.034$). This perspective is derived from the comparison to the distance to the nearest neighbor in the training data set ($\mathscr{D}_{\mathrm{sh}} = 0.038$) and to the furthest data point ($\mathscr{D}_{\mathrm{sh}} = 0.139$). It is noteworthy that the RL agent required only 400 episodes to achieve this result, in contrast to the baseline set based on the generation of 76,980 random samples. The RL algorithm can therefore be seen as being data efficient.

Nevertheless, in this study, it seems that a lower bound is existing that is difficult to overcome by the MEG-SGGPO approach. In general, this can have two reasons: First, the forming process is unable to produce textures that are closer to the goal texture identified by the SMTLO approach. Yet, the SMTLO approach already addresses this issue by enforcing microstructures to remain within the region delineated by the known microstructures from the training data set via the term $\mathcal{F}_{\mathrm{val}}$ in Eq. (7.12). A stronger enforcement was expected to lead to textures that are better reachable by the process. This, however, can lead to a lower accuracy in terms of obtaining the desired material properties, due to the change in the other weights in the objective function of the SMTLO optimizer. Second, the MEG-SGGPO approach identified a local optimum and got stuck. This can be mitigated by longer MEG-SGGPO runs with optimized hyperparameters. Longer runs, however, have not shown to yield significant improvements in the presented study. As future work, it is desirable to incorporate the knowledge contained in the generated training samples for SMTLO into MEG-SGGPO a priori to enhance its performance.

Chapter 8

Summary and Conclusion

This work addressed the challenge of advancing data-driven methodologies for modeling, optimization, and inversion of structure–property relationships in materials science. The field is characterized by the high complexity arising from interdependencies between processing conditions, resulting microstructures, and material properties. Capturing and exploiting these interdependencies required the development of high-quality latent representations that served as a universal, low-dimensional latent space in which heterogeneous types of data could be expressed and optimized. The focus of this work was on proposing a multitask latent space learning framework that integrated multiple objectives to produce domain-aware latent features, tailored to capture the complexity of these relationships. Across the presented contributions, novel methods were developed to construct and exploit latent feature spaces that preserved domain-specific dependencies. These latent spaces were established as a central element for enabling efficient optimization and inverse design.

The **first contribution** proposed a convolutional neural network (CNN)–based multitask learning (MTL) framework for extracting a compact, domain-aware latent feature space from process curves. The approach ensured that the learned latent representation captured both the variability of the curves and their systematic dependencies on process conditions. This was achieved by attaching three complementary objectives to the encoder output: a decoder for reconstructing process curves, a classifier for discrete process conditions, and an estimator for continuous conditions. The resulting latent space at the CNN output represented the process curves in a manner that remains sensitive to the underlying process conditions.

The validity of the approach was demonstrated using resistance spot welding as an application example. Different CNN and multilayer perceptron (MLP) architectures were explored and compared. A latent space of only three dimensions was identified, which enabled accurate reconstruction of welding curves, classification of the welding gun in use and the steel sheet combination, as well as the estimation of the welding spot diameter with sufficient accuracy. A comparison with a latent space derived from three principal component analysis (PCA) features showed the superiority of non-linear feature extraction with CNN and MLP, with CNN exhibiting a slight advantage over MLP. The latent space obtained from the proposed approach further enabled the transformation of stochastic processes (their probability densities) that generated process curve instances to other process conditions. This capability constituted an important prerequisite for hyper modeling of the processes, where the dependencies on varying conditions were captured and subsequently used to predict the properties of previously unseen processes.

In the **second contribution**, an approach for addressing materials design problems was presented. The approach was based on an optimization strategy that incorporated machine learning models. Building on the foundation of the first contribution, the MTL framework was extended to a Siamese multitask learning (SMTL) framework. This extension was motivated by the additional requirement to explicitly preserve distances in the latent feature space, thereby enabling an optimizer to efficiently identify microstructures with desired properties. While the approach from the first contribution ensured that the latent space captured both the underlying structure of the input and its dependencies on process conditions, it did not explicitly preserve distance-based relationships between samples in the latent space. This limitation was addressed by incorporating a distance-preservation loss term that approximately maintained the pairwise distances between microstructures from the input space in the learned latent representation. The SMTL framework employed a neural network to jointly perform four key tasks: (1) reconstructing microstructures, (2) mapping microstructures to their corresponding properties, (3) assessing the validity of microstructures with respect to the underlying data distribution, and (4) preserving pairwise distances of microstructures in the latent space.

Building upon the latent space representation from the SMTL framework, an optimization strategy was employed to operate efficiently and directly within this space. This approach enabled the generation of diverse sets of microstructures that satisfied specified property constraints. Operating in the latent space, combined with the ability to measure distances within it, allowed for direct assessment of diversity. In this way, the optimization process was prevented from converging to a single solution and instead explored multiple valid microstructures.

The approach was validated in crystallographic texture optimization, where it successfully identified multiple distinct textures with desired properties. This provided a practical basis for structure-guided processing strategies. Two optimization use cases were examined, targeting regions of the property space with sparse and dense data coverage. In both scenarios, the approach generated not only a greater number of solutions but also more diverse ones than those contained in the baseline data, demonstrating that such sets of textures could serve as input for optimal processing control approaches.

The **third contribution** addressed the challenge of quantifying distances between crystallographic textures, a key microstructural feature responsible for anisotropic behavior. The measurement of texture distances is essential for tasks in which knowledge about texture similarity is required. Examples include the design of processes intended to achieve a target texture and the modeling of texture–property relations for materials design, where the objective is to optimize textures for given properties.

Therefore, this challenge was investigated in more detail for crystallographic textures, which are commonly described by the orientation distribution function (ODF). Several methods for measuring texture distances were developed and evaluated for textures represented by the ODF, including: (1) an explicit distance measure for ODFs expanded in generalized spherical harmonics (GSH), (2) the Sinkhorn distance for ODFs represented by orientation histograms, (3) the Chi-Squared distance for orientation histograms, and (4) a distance measure for ODFs represented as pole figures.

By evaluating the texture distances resulting from an example process, the advantages and drawbacks of the considered texture representations and distance measures were identified. The comparative evaluation revealed that the Sinkhorn distance provided higher resolution for small differences, whereas the GSH distance was preferable for larger differences. Application to ex-

perimental hot rolling, cold rolling, and heat treatment process data showed that all investigated measures effectively captured process state evolution for non-spiky ODFs.

Furthermore, the suitability of the considered texture representations for learning texture-property relations was investigated. The evaluation of texture-property modeling showed that GSH representations achieved a quality comparable to pole figures at lower computational cost, with both outperforming orientation histograms.

The **fourth contribution** presented a holistic inverse design approach that enabled accelerated materials development by explicitly accounting for the manufacturing process. It integrated the Siamese multitask learning framework and optimization strategy (Siamese multitask learning optimization, SMTLO) from the second contribution with the Sinkhorn distance measure from the third contribution. This integration demonstrated that the previously proposed approaches were capable of supporting end-to-end optimization of process–structure–property relationships in complex manufacturing scenarios.

The case study focused on a metal forming process with a combinatorial total of 25^7 possible processing paths—a scale that posed a substantial challenge for conventional design methods. Despite the vast design space, the SMTLO approach successfully solved the materials design problem using only 76,980 samples. It identified diverse sets of near-optimal textures that satisfied the specified desired properties. The crystallographic textures were considered near-optimal in the sense that each exhibited different but satisfactory macroscopic properties, and at the time of identification it was not known which one would be most attainable by the process. After solving the materials design problem, 400 episodes were sufficient for the multi-equivalent goal structure-guided processing path optimization (MEG-SGGPO) approach to successfully guide the forming process to the best reachable crystallographic texture and thereby solved the process design problem.

By explicitly exploiting the non-uniqueness of the materials and process design problems, the approach identified a diverse set of microstructures for given property requirements and subsequently produced the best reachable microstructure. This constituted a significant advantage when transferring the approach to real manufacturing systems, where constraints often exist that may exclude certain microstructures from being producible. This work advanced the field of data-driven materials design by addressing key challenges in integrating methodologies for modeling, optimization, and inverse design of structure–property relationships in materials science.

Collectively, these contributions established a unified framework for multitask latent space learning that spanned representation learning and diversity-aware optimization. Furthermore, robust texture distance measures—including the Sinkhorn distance—were developed and evaluated, which were essential for tasks requiring the assessment of texture similarity. Examples included the design of processes intended to achieve a given target texture and the modeling of texture–property relations for materials design, where the objective was to optimize textures for specified properties. The methods introduced in this work not only enabled forward prediction and inverse design of structure–property relationships but also were integrated into manufacturing contexts, providing a pathway toward data-driven process and materials design with improved efficiency.

Chapter 9

Outlook

Future research should focus on further developing the methodologies presented in this work, with the objective of transferring them to other process and manufacturing techniques as well as to different texture representations. Although the present studies employed data from mean-field simulations, it would be appealing to apply these methods to spatially resolved data from full-field simulations in subsequent studies. Such approaches, for instance the spectral method of (Eisenlohr et al., 2013), have demonstrated the ability to capture microstructural heterogeneity with higher fidelity. Likewise, recent developments in simulation platforms such as DAMASK highlight the potential of full-field methods to provide insights into deformation mechanisms and damage phenomena across scales (Roters et al., 2019). This would facilitate a more detailed comprehension of microstructural evolution and its impact on material properties.

Moreover, it would be desirable to evaluate the proposed methods in actual manufacturing systems to ensure their practical applicability. A key challenge for such validation is the large amount of training data required, which contrasts with the limited availability of high-quality experimental datasets in production workflows. As (Schmidt et al., 2019) emphasize, data scarcity remains a central bottleneck for applying machine learning methods in solid-state materials science.

To address this limitation, future research could explore the pre-training of machine learning models with data from numerical simulations that serve as digital twins of the material and process. Recent surveys underline their broader role as enablers of model generalization and transferability, offering effective pathways to bridge between simulation data and scarce experimental measurements (Rasheed et al., 2020). The use of digital twins would thus facilitate the effective generalization of models prior to fine-tuning on limited experimental data.

In addition, future research endeavors may concentrate on the advancement of methods that integrate Bayesian optimization with active learning to further improve materials design, process optimization, and inverse modeling. A particularly important direction concerns the refinement of constraint-aware optimization frameworks. Building on the entropy-based formulation of (Khatamsaz et al., 2023), such approaches could extend the efficiency of design space exploration while simultaneously ensuring manufacturability and feasibility. Complementary to this, the development of self-correcting Bayesian optimization strategies, as introduced by (Hvarfner et al., 2023), offers a pathway to dynamically adjust the exploration–exploitation balance in response to evolving model uncertainties.

A further domain of exploration lies in the formal convergence of Bayesian optimization and

active learning. (Di Fiore et al., 2024) provide a theoretical perspective on how adaptive sampling strategies can be designed to maximize information gain while minimizing computational expense. Earlier surveys such as (Settles, 2009) laid the groundwork by defining the principles of active learning, while more recent work by (Kirsch et al., 2019) introduced efficient algorithms for deep learning settings. Within materials science specifically, (Lookman et al., 2019) demonstrated how Bayesian optimization combined with active learning can accelerate the discovery of materials with targeted properties, illustrating the practical benefits of such strategies. Beyond methodological refinements, it will also be crucial to enhance the quality and interpretability of training datasets. In this context, (Swayamdipta et al., 2020) proposed dataset cartography as a structured means of diagnosing dataset composition and identifying the most informative and representative samples. Other approaches, such as confident learning by (Northcutt et al., 2021), further expand these capabilities by systematically detecting label errors in training data.

Further research could investigate the potential of auto-decoders and latent space interpolation techniques to enhance the flexibility and robustness of machine learning models in materials science. Auto-decoders, introduced for example in the DeepSDF framework by (Park et al., 2019), map latent representations directly to outputs, providing an alternative to traditional encoder–decoder architectures by focusing exclusively on the decoding of latent variables into target outputs. This architecture enables a more direct manipulation of the latent space, thereby facilitating the exploration and optimization of material microstructures.

Latent space interpolation itself is a vital element of this research direction. Its objective is to identify meaningful transitions between data points and to reveal intermediate solutions within the latent space. (White, 2016) has shown how different interpolation strategies—linear, spherical, or geodesic—yield qualitatively different results, underscoring the need for principled interpolation approaches. Advances such as DeepSDF (Park et al., 2019) illustrate how continuous representations can capture smooth transitions, enabling interpretable interpolations between target states. Applied to materials design, these principles could support the discovery of novel microstructures and process parameter configurations that lie between known solutions.

The geometry of latent spaces is another critical consideration. (Arvanitidis et al., 2017) introduced the concept of latent Riemannian geometry in generative models, demonstrating how curvature affects interpolation trajectories and distances. In a related field, (Shao et al., 2018) explored Riemannian metrics in generative models to ensure that latent transitions are meaningful and consistent with underlying data structures. These perspectives are particularly relevant for materials science, where avoiding unrealistic or non-physical interpolations is essential for meaningful structure–property exploration.

To further refine latent space interpolations, regularization strategies play a central role. β-variational autoencoders (β-VAEs) (Higgins et al., 2017) introduced disentanglement through stronger latent regularization, improving interpretability and controllability of latent traversals. More recent methods, such as adversarial regularizers (Berthelot et al., 2019) and interventional explorations (Leeb et al., 2022), provide additional mechanisms to enforce smoothness and coherence of interpolations while maintaining consistency with physical and statistical constraints.

Another apporach for research is the utilization of advanced materials informatics tools and semantic data orchestration frameworks to improve data accessibility and integration. The adoption of the FAIR principles (Wilkinson et al., 2016), which emphasize findability, accessibility, interoperability, and reusability, has already begun to shape materials-science data infrastructures. Within this context, (Scheffler et al., 2022) demonstrated how these principles can be

operationalized for heterogeneous materials datasets, building on earlier initiatives such as the NOMAD Laboratory (Draxl and Scheffler, 2018). Large-scale resources such as the Materials Project (Jain et al., 2013) further exemplify the value of FAIR-aligned databases in accelerating discovery. Beyond data management, semantic orchestration frameworks provide another route to leverage distributed knowledge. (Nahshon et al., 2025) showcased their application in steel and copper, while (Pizzi et al., 2016) emphasized the importance of provenance-tracking infrastructures such as AiiDA. In parallel, advances in materials informatics have introduced increasingly powerful descriptors and machine-learning frameworks for property prediction and design. (Ward et al., 2018) demonstrated how feature engineering can embed domain knowledge into predictive pipelines, while (Sivan et al., 2024) surveyed recent methodological progress in the field. At the same time, (Bukkapatnam, 2023) presented a framework for autonomous materials discovery.

Looking ahead, multimodal machine learning is meant to play an important role in the next stage of machine learning research. The central idea of multimodality is to integrate heterogeneous information sources—such as text, images, audio, video, and sensor data—into a unified representational space. This ability to connect and reason across modalities opens the door to more general and context-aware ML systems, capable of understanding complex relationships that are not accessible to unimodal approaches. Recent advances already demonstrate this potential: models such as CLIP align language and vision to enable zero-shot generalization to new tasks (Radford et al., 2021), while systems like Flamingo extend this capability to few-shot learning across diverse visual-linguistic scenarios (Alayrac et al., 2022).

In parallel with these developments, foundation models are shaping the broader trajectory of machine learning. These models are trained on massive and diverse datasets and subsequently adapted to a wide range of downstream tasks with minimal additional training. Their emergence reflects a shift away from narrowly specialized architectures toward flexible systems capable of generalization across domains. Transformer-based architectures such as the Vision Transformer (Dosovitskiy et al., 2021) exemplify this trend, demonstrating how pretraining on large-scale image datasets enables transfer to diverse vision tasks with remarkable efficiency. More recently, models such as DINO (Caron et al., 2021) and SAM (Kirillov et al., 2023) illustrate how foundation models extend beyond language, providing general-purpose feature extractors and adaptable tools for image understanding and segmentation

Alongside these advances, large language models have rapidly become central to the progress of machine learning. Trained on vast text corpora, these models learn statistical and semantic patterns of language that enable them to perform a wide spectrum of tasks, from translation and summarization to reasoning and code generation, often without explicit task-specific training. The development of autoregressive transformers such as GPT-3 (Brown et al., 2020) demonstrated that scaling model size and training data leads to emergent abilities, including few-shot and zero-shot generalization. More recently, models like GPT-4 (OpenAI, 2023) and Gemini (Gemini Team et al., 2023) extend this trend, achieving increasingly sophisticated reasoning, problem-solving, and interaction capabilities. This trajectory reflects a shift from narrow task-oriented NLP systems toward general-purpose language-based reasoning engines.

Appendix A

Appendix

A.1 Supporting information for Section 2.1

A.1.1 Neural networks

Siamese neural networks

Siamese neural networks (SNNs) can be trained using for example a contrastive loss function or a triplet loss function, depending on the desired outcome. The objective of these functions is to learn latent features that maximize inter-class separation while minimizing intra-class distances. In contrastive loss, as initially proposed by (Hadsell et al., 2006), the objective is to minimize the distance between the latent features of similar pairs while maximizing the distance between the latent features of dissimilar pairs, up to a margin

$$\mathcal{J}_{\text{contr}} = \boldsymbol{y} \cdot (||\boldsymbol{z}_1 - \boldsymbol{z}_2||_2)^2 + (1 - \boldsymbol{y}) \cdot \max(0, m - ||\boldsymbol{z}_1 - \boldsymbol{z}_2||_2)^2. \tag{A.1}$$

In this context, $||\boldsymbol{z}_1 - \boldsymbol{z}_2||_2$ is the Euclidean distance between the latent features of $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$. The binary label $\boldsymbol{y}$ assumes the value of 1 for similar and 0 for dissimilar observations. The margin m guarantees that dissimilar pairs are adequately separated.

In the context of face recognition, (Schroff et al., 2015) proposed triplet loss, which considers three inputs: an anchor, represented by $\boldsymbol{x}_{\text{a}}$, a positive example, represented by $\boldsymbol{x}_{\text{p}}$, which is similar to the anchor, and a negative example, represented by $\boldsymbol{x}_{\text{n}}$, which is dissimilar to the anchor. The objective is to guarantee that the distance between the anchor and the positive is less than that between the anchor and the negative by a margin of at least α:

$$\mathcal{J}_{\text{triplet}} = \max(0, (||\boldsymbol{z}_{\text{a}} - \boldsymbol{z}_{\text{p}}||_2)^2 - (||\boldsymbol{z}_{\text{a}}, \boldsymbol{z}_{\text{n}}||_2)^2 + \alpha). \tag{A.2}$$

A.1.2 Dimensionality reduction

Linear dimensionality reduction techniques

Singular value decomposition (SVD) is a mathematical decomposition technique that serves as the foundation for a multitude of dimensionality reduction methods, including principal component analysis (PCA). The origins of SVD can be traced back to the work of (Eckart and Young, 1936) on approximating matrices, which were subsequently generalized (Golub and Reinsch, 1970) for numerical applications. SVD decomposes a data matrix $\boldsymbol{X}$ into three matrices:

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\top}, \tag{A.3}$$

where $\boldsymbol{U}$ contains the left singular vectors (orthogonal), $\boldsymbol{\Sigma}$ is a diagonal matrix of singular values (representing the magnitude of variance along each dimension), and $\boldsymbol{V}^{\top}$ contains the right singular vectors (orthogonal), corresponding to the principal axes of the data. In contrast to PCA, SVD operates directly on the data matrix, thereby obviating the need for the computation of a covariance matrix. This renders it a computationally efficient method for the analysis of large datasets.

Non-linear dimensionality reduction techniques

Non-linear dimensionality reduction techniques are designed to handle data that lies on a non-linear manifold in the original high-dimensional space. These methods are trying to represent data in a lower dimensional space, while preserving either the topology (TP) of the data (in mathematical sense), or the geometry (GP) reflected by the distance or angle between data. TP methods are self-organizing maps (SOM) (Kohonen et al., 2001), locally linear embedding (LLE) (Roweis and Saul, 2000) and Laplacian Eigenmaps (LE) (Belkin and Niyogi, 2001), while multidimensional scaling (MDS) (Cox and Cox, 2008), Kernel PCA (Schölkopf et al., 1997) and isomap (Tenenbaum et al., 2000) belong to the GP class.

All the aforementioned methods assume that the data reside on a single manifold in a subspace of the original data space. If the data populate multiple manifolds (eventually in different dimensions and possibly overlapping and intersecting), multi-manifold learning is required, such as k Manifolds (Souvenir and Pless, 2005) or DC-Isomap (Gao and Liang, 2013), which decompose the manifold into intersecting-only and separated-only sub-manifolds respectively. All of these methods tend to find features, which optimally fulfil the method-specific criteria, but which are not necessarily uncorrelated or do not allow back-projection into the original space.

Kernel versions of the partial least squares (PLS) method allow the extraction of non-linear features and also maximize the correlation with some regression output variables (Rosipal and Trejo, 2001). Kernel-PLS as a regression method would be able to create a latent space reflecting the influence of continuous quantities, but can not treat conditions with discrete values. Principal function approximators (PFAs) (Senn, 2013) are especially constructed to realize a simultaneous invertible, non-linear dimensionality reduction of state variables and a regression of the extracted features with observables, which extends the linear methods of partial least squares regression (Vinzi et al., 2010) and principal component regression (Merz and Pazzani, 1999). Other than with (Kernel-)PLS, which builds upon variance, PFA features are formed by the latent feature space of an auto-encoder-like neural network and are not ordered according to relevance. The property of relevance-ordering is incorporated in (Fischer et al., 2015) in neural auto-encoding via a hierarchy of single-neuron-bottleneck neural networks (HSB-NN), where each auto-encoder in the hierarchy represents the reconstruction residuum of its predecessor, and can also be trained to maximize the correlation with observables. As being regressors like Kernel-PLS, these methods cannot treat processes with discrete-valued conditions.

Sammon mapping is a non-linear dimensionality reduction technique that aims to preserve the intrinsic structure of the data set by minimizing the distortion of pairwise distances during the projection (Sammon, 1969). Similarly to MDS, Sammon mapping strives to maintain the distances between points, but it places greater emphasis on preserving small distances. The

objective function for Sammon mapping is given by the following equation, which is often referred as Sammon error E:

$$E = \frac{1}{\sum_{i<j} d_{ij}} \sum_{i<j} \frac{(d_{ij} - d'_{ij})^2}{d_{ij}}, \tag{A.4}$$

where d_{ij} is the distance between points i and j in the original space, and d'_{ij} is the distance in the lower-dimensional space. By weighting the errors in distance preservation in accordance with the original distances, Sammon mapping ensures the maintenance of the local structure of the data, thereby preserving the intrinsic organization of the data set.

A.1.3 Generative approaches

Variational autoencoders (VAEs)) (Kingma and Welling, 2014) are a type of generative model that aim to learn a probabilistic latent feature variable representation of data. A fundamental distinction between VAEs and traditional autoencoders lies in their respective assumptions regarding the nature of the latent variables $\boldsymbol{z}$. In autoencoders, these latent variables are assumed to follow a deterministic distribution, whereas in VAEs, the assumption is made that the latent variables $\boldsymbol{z}$ follow a probabilistic distribution. This probabilistic assumption enables the generation of new data samples and the structured encoding of information.

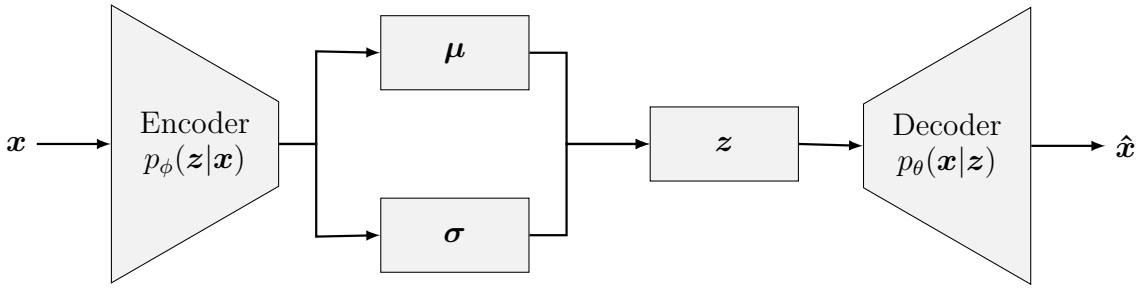

Figure A.1: Conception of the variational autoencoder.

The encoder in a VAE (see Fig. A.1) maps input data $\boldsymbol{x}$ to a distribution in the latent space $\boldsymbol{z}$. The expression

$$p_\phi(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{\mu}, \mathrm{diag}(\boldsymbol{\sigma^2})) \tag{A.5}$$

defines the approximate posterior distribution over the latent variables $\boldsymbol{z}$ conditioned on the input data $\boldsymbol{x}$. The distribution is modeled as a multivariate Gaussian with parameters $\boldsymbol{\mu}$ and $\boldsymbol{\sigma^2}$. $\mathcal{N}$ denotes a multivariate Gaussian distribution with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. Specifically, $\boldsymbol{\mu} \in \mathbb{R}^Z$ denotes the mean vector of the Gaussian distribution, representing the central tendency of the latent variable distribution in a Z-dimensional latent space. The term $\boldsymbol{\sigma^2} \in \mathbb{R}^Z$ represents the variance for each dimension, and $\mathrm{diag}(\boldsymbol{\sigma^2})$ constructs a diagonal covariance matrix, implying that the latent dimensions are assumed to be uncorrelated. The parameters $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are learned functions of the input $\boldsymbol{x}$, parameterized by ϕ, such that the encoder network outputs these values to characterize the posterior distribution. This representation ensures that the latent space is probabilistic, thereby allowing variability in the generated data.

During the training phase, the latent variable $\boldsymbol{z}$ is sampled from the distribution $p_\phi(\boldsymbol{z}|\boldsymbol{x})$. However, the process of directly sampling is not differentiable, which poses a significant challenge for the backpropagation algorithm. To address this challenge, VAEs utilize the reparameterization trick, where the sampled $\boldsymbol{z}$ is expressed as $\boldsymbol{z} = \boldsymbol{\mu} + \epsilon\boldsymbol{\sigma}$, with $\epsilon \sim \mathcal{N}(0, \mathrm{I})$ allowing gradients to flow through $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ during optimization. Subsequently, the decoder utilizes this information to reconstruct the input data $\boldsymbol{x}$ from the latent variable $\boldsymbol{z}$. The reconstructed data is modeled as a distribution, such as a Bernoulli or Gaussian distribution, depending on the nature of the input data. For instance, the reconstruction likelihood might be represented as $p_\theta(\boldsymbol{x}|\boldsymbol{z})$. The decoder's objective is to maximize this likelihood, thereby ensuring that the reconstructed data closely resembles the original input.

The goal of the VAE is to optimize the Evidence Lower Bound (ELBO), which balances reconstruction accuracy and regularization of the latent space. The ELBO is given by:

$$\mathcal{J}_{\text{vae}} = \mathbb{E}_{p_\theta(\boldsymbol{z}|\boldsymbol{x})}[\log p(\boldsymbol{x}|\boldsymbol{z})] - \mathrm{KL}(p_\phi(\boldsymbol{z}|\boldsymbol{x}) \,\|\, p(\boldsymbol{z})), \tag{A.6}$$

where the first term contains the expectation operator $\mathbb{E}[\cdot]$, which denotes the expected value of a random variable under a given probability distribution. The expectation in this case therefore quantifies, how well the decoder distribution $p(\boldsymbol{x}|\boldsymbol{z})$, parameterized by θ, can reconstruct the observed data $\boldsymbol{x}$ from its latent representation $\boldsymbol{z}$. The second term involves the Kullback–Leibler (KL) divergence, denoted as $\mathrm{KL}(\cdot \,\|\, \cdot)$, which regularizes the latent distribution $p_\phi(\boldsymbol{z}|\boldsymbol{x})$ to be close to a prior $p(\boldsymbol{z})$, often chosen as $\mathcal{N}(0, \mathrm{I})$. The vertical double bar symbol $\|$ in the argument of the KL divergence is read as "with respect to" and indicates the ordering of the two distributions: the divergence measures how much $p_\phi(\boldsymbol{z}|\boldsymbol{x})$ diverges from $p(\boldsymbol{z})$, but not vice versa, since the KL divergence is not symmetric.

The model reduces the negative ELBO, which is equivalent to maximizing the log likelihood of the data under the generative model while constraining the latent space. The optimization ensures a balance between accurate reconstructions and a structured latent space, leading to both effective compression and realistic data generation. This probabilistic framework facilitates the generation of robust data. The generation of new data is achieved through the sampling of a latent vector $\boldsymbol{z}$ from the prior $p(\boldsymbol{z})$, followed by its decoding using the decoder to yield a new data point $\boldsymbol{x}^* \sim p(\boldsymbol{x}|\boldsymbol{z})$. But neither the objective function used during training nor the standard VAE architecture inherently guarantees the ability to perform semantic interpolation within the latent space.

The problem of forcing VAEs to generate semantically conditioned object representations is addressed by the method of conditional variational autoencoders (CVAE) introduced by (Sohn et al., 2015). CVAEs (see Fig. A.2) arrange the representation space of process curves such that representations under the same conditions form clusters and can be interpreted w.r.t. the conditions. The CVAEs enhance the object data by including the class memberships of the objects in their original description. This is achieved by representing the class-membership as a one-hot condition vector $\boldsymbol{c}$ and concatenating this vector with the objects original data vector. In the resulting extended representation space the patterns of each class are represented in their own sub-space. Regular VAEs (structure and loss function) are then applied in this extended representation space to finally create the CVAEs. The resulting class-conditional densities in the latent space allow generating class-conditional object representations (by switching between the classes), but the densities are not spatially separated in the latent space. Therefore, it is

not possible to interpolate between classes in the latent space.

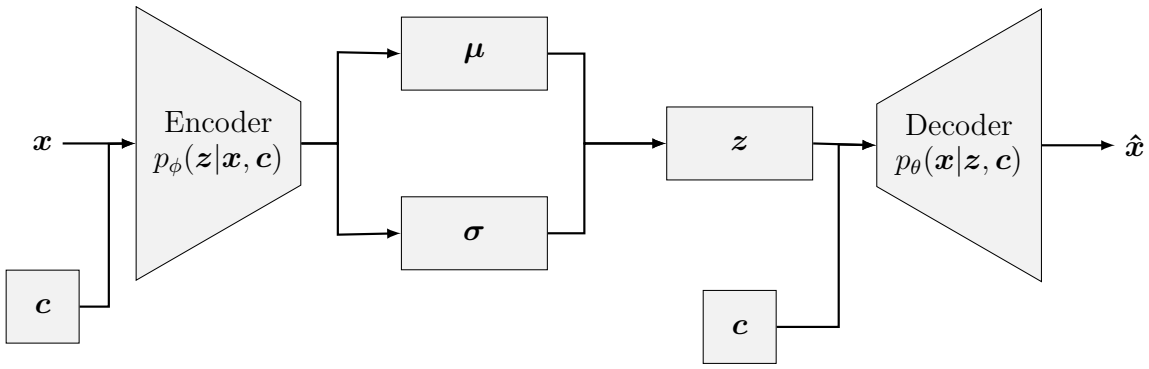

Figure A.2: Depiction of the conditional variational autoencoder.

Generative adversarial networks (GANs), introduced by (Goodfellow et al., 2014), have emerged as a powerful class of deep generative models designed to learn the underlying distribution of data. GANs comprise two neural networks: the generator and the discriminator (see Fig. A.3). The two networks are trained concurrently in a game-theoretic framework and collaborate to generate realistic data samples by leveraging their adversarial relationship. The generator, denoted as G, is responsible to sample random noise $\boldsymbol{z}$ from a prior distribution (such as a Gaussian or uniform distribution) to produce synthetic samples (e.g., images, text, or audio) that closely resembles the distribution of real data. On the other hand, the discriminator, denoted as D, is a binary classifier and designed to distinguish between real samples and the generator's output. The training of GANs is formulated as a minimax game between G and D, with the following objective function:

$$\mathcal{J}_{\text{gan}} = \min_G \max_D \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}}[\log(1 - D(G(\boldsymbol{z})))]. \tag{A.7}$$

In this context, $\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[\log D(\boldsymbol{x})]$ denotes the discriminator's capacity to accurately classify real samples, while $\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}}[\log(1 - D(G(\boldsymbol{z})))]$ signifies its ability for discerning synthetic samples generated by the generator. The generator's objective is to minimize this term, thereby deceiving the discriminator by generating data that closely mirrors the true data distribution. This adversarial process drives the generator to create samples that is increasingly realistic.

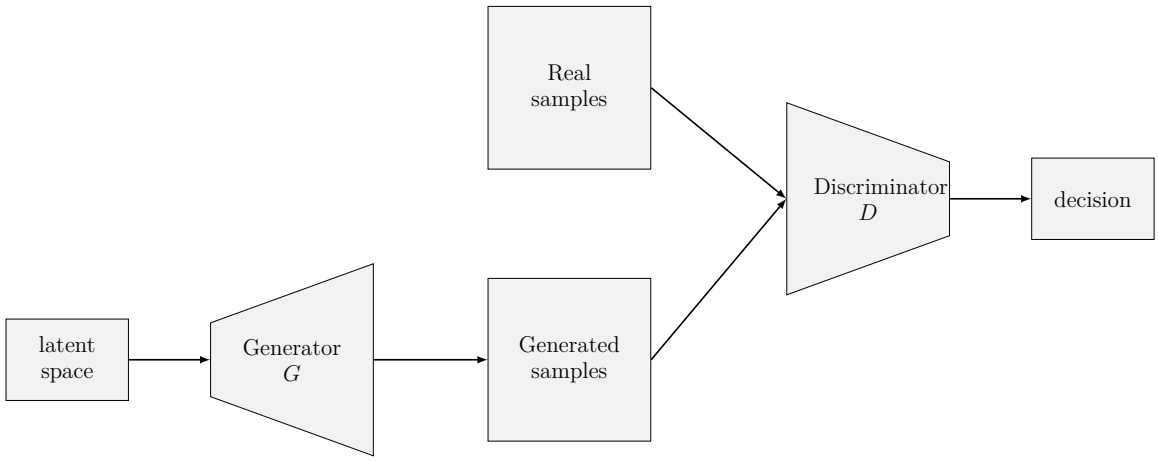

Figure A.3: Visualization of the generative adversarial network.

A significant advancement in the field of GANs was the introduction of conditional generative adversarial networks (cGANs) by (Mirza and Osindero, 2014). This innovative approach incorporates supplementary information, such as class labels $\boldsymbol{c}$ or input data $\boldsymbol{x}$, with the objective of enhancing the generation process. In cGANs, both the generator and discriminator are conditioned on this auxiliary information, thereby enabling the generation of data that adheres to specific characteristics or constraints (see Fig. A.4). To illustrate, in the context of image generation, cGANs are capable of generating images that correspond to specific classes. This is demonstrated by the generation of digits from 0 to 9 based on the input labels. The conditioning mechanism provides enhanced control over the generative process, rendering cGANs highly useful in applications such as image-to-image translation (Isola et al., 2017), text-to-image generation (Reed et al., 2016), and style transfer (Zhu et al., 2017).

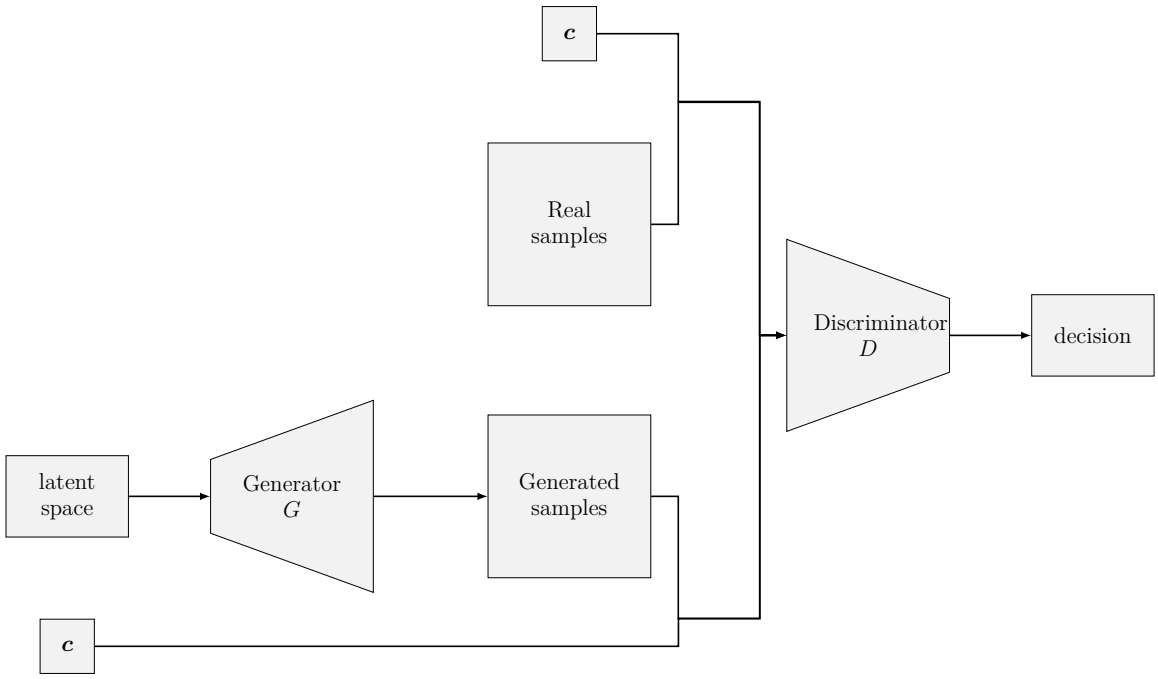

Figure A.4: Neural network that presents the conditional generative adversarial network.

A.1.4 Anomaly detection

Anomaly detection in the role of unsupervised learning is the process of identifying instances in data that deviate from the expected patterns and are therefore classified as anomalous. The sporadic and diverse nature of anomalies often renders it impractical to obtain labeled data for all potential anomalous scenarios, given the inherent limitations of human labeling capabilities. This renders unsupervised learning techniques an optimal choice for addressing this challenge. Anomaly detection is a technique that is commonly employed in a number of fields, including fraud detection, network security, manufacturing quality control, and medical diagnostics. A frequently utilized metric in such methodologies is the Mahalanobis distance $\mathscr{D}_M$, as defined by (Mahalanobis, 1936). The formula for the Mahalanobis distance between a vector $\boldsymbol{x}$ and the mean vector $\boldsymbol{\mu}$ of a distribution is as follows:

$$\mathscr{D}_M(\boldsymbol{x}) = \sqrt{(\boldsymbol{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})}, \tag{A.8}$$

where $\boldsymbol{\mu}$ is the mean vector of the distribution, $\boldsymbol{\Sigma}$ is the covariance matrix of the data, and $\boldsymbol{\Sigma}^{-1}$ is the inverse of the covariance matrix. This metric quantifies the distance between a data point and the center of a distribution, accounting for the correlations between features. In contrast to the Euclidean distance, the Mahalanobis distance considers the shape of the data distribution, rendering it highly effective for identifying anomalies in multivariate datasets where the features are not independent. Points with considerable Mahalanobis distances from the center of the distribution are identified as potential outliers (see Fig. A.5).

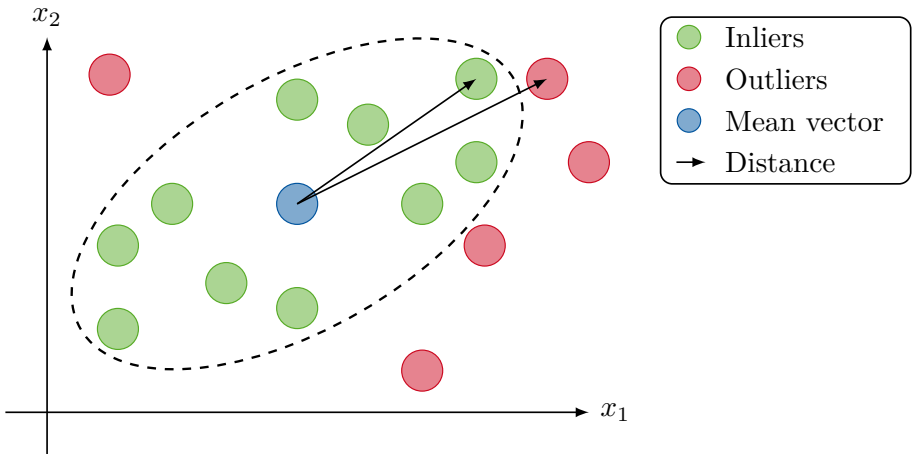

Figure A.5: Visualization of the Mahalanobis distance.

Another effective method for anomaly detection is the one-class support vector machine (OCSVM), which has been specifically designed to separate normal data (inliers) from anomalies (outliers) in high-dimensional data (Schölkopf et al., 2001). OCSVM models are trained to identify a decision boundary that encompasses the majority of data points, which are presumed to represent typical, non-anomalous observations. Therefore, data points that fall outside this boundary are classified as anomalous (see Fig. A.6). As with standard support vector machines (SVMs), OCSVM has the capacity to utilize kernels (e.g., linear, RBF, polynomial) to map input data to a higher-dimensional feature space, thereby enabling the system to process nonlinear patterns. The system identifies a hyperplane in the feature space that maximizes the margin between the origin (or a reference point) and the data points while minimizing the number of points classified as outliers. The decision function $f(\boldsymbol{x})$ is used to evaluate whether a data point $\boldsymbol{x}$ is within the normal region:

$$f(\boldsymbol{x}) = \boldsymbol{w}^\top \phi(\boldsymbol{x}) - \rho, \tag{A.9}$$

where $\phi(\boldsymbol{x})$ represents the mapping to the feature space, $\boldsymbol{w}$ is the weight vector defining the hyperplane, and ρ is the offset parameter. This method is particularly well-suited for anomaly detection because it is capable of capturing complex boundaries around the normal data set, rendering it highly effective in scenarios where anomalies are rare and do not form well-defined clusters.

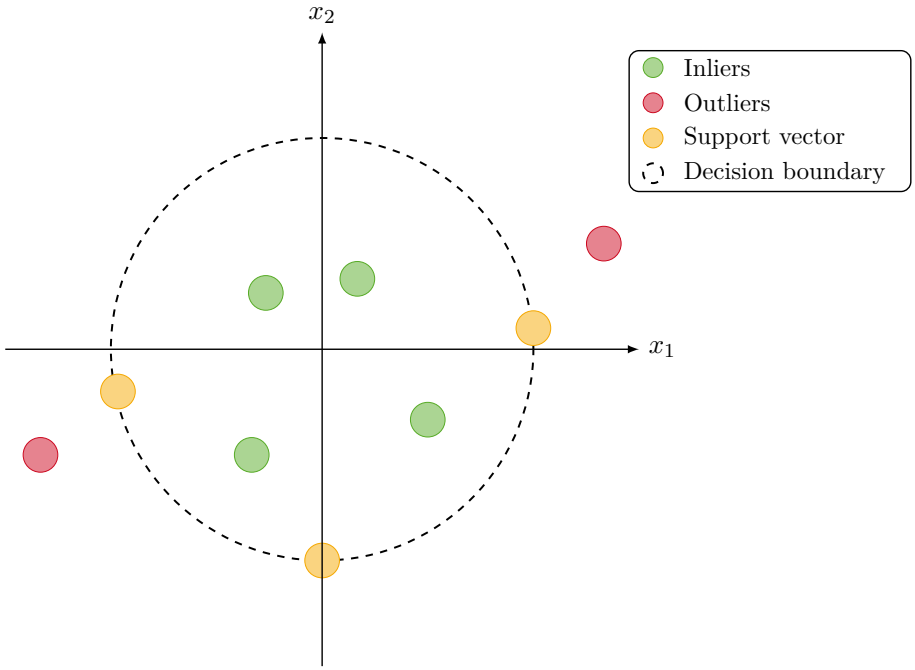

Figure A.6: Visualization of the one-class support vector machine.

A.1.5 Gradient-free optimization

The genetic algorithm (GA), initially proposed by (Holland, 1975), is based on the principles of natural selection and genetics. The algorithm evolves a population of candidate solutions through processes analogous to biological evolution, including selection, crossover, and mutation. In each generation, individuals exhibiting superior fitness are more likely to be selected for reproduction. Crossover combines the genetic information of two parent solutions to produce offspring, while mutation introduces minor random alterations to ensure diversity within the population. The fitness function provides a framework for guiding the optimization process and evaluating the quality of solutions. GAs are particularly effective for combinatorial optimization problems, including scheduling, traveling salesperson problems, and feature selection in ML. The versatility of these algorithms allows for their application to a diverse array of problems. Nevertheless, it is often necessary to conduct precise parameter tuning in order to achieve optimal performance. A more detailed discussion of this topic can be found in (Holland, 1975; Goldberg, 1989).

Particle swarm optimization (PSO), as initially proposed by (Kennedy and Eberhart, 1995), draws inspiration from the social behaviors observed in flocking birds and schooling fish. Each particle within the swarm represents a potential solution, navigating the search space in accordance with its velocity, which is shaped by its own historical position and the collective position of the swarm:

$$\boldsymbol{v}_i^{(t+1)} = \omega \cdot \boldsymbol{v}_i^{(t)} + c_1 r_1 (\boldsymbol{p}_i - \boldsymbol{x}_i^{(t)} + c_2 r_2 (\boldsymbol{g} - \boldsymbol{x}_i^{(t)})), \tag{A.10}$$

$$\boldsymbol{x}_i^{(t+1)} = \boldsymbol{x}_i^{(t)} + \boldsymbol{v}_i^{(t+1)}, \tag{A.11}$$

where $\boldsymbol{v}_i^{(t)}$ is the velocity vector of particle i at time t, $\boldsymbol{x}_i^{(t)}$ is the position vector of particle i at time t, $\boldsymbol{p}_i$ is the particle's best-known position vector particle i, $\boldsymbol{g}$ is the global best-known position vector, ω is the inertia weight, and c_1, c_2, r_1, and r_2 are acceleration coefficients and random numbers. The principal strength of PSO lies in its capacity to maintain a dynamic equilibrium between exploration and exploitation.

A.2 Supporting information for Chapter 4

A.2.1 Encoder topologies

The encoder topologies developed in this study are shown in Fig. A.7 for MLP, Fig. A.8 for CNN-VA1, Fig. A.9 for CNN-VA2, and Fig. A.10 for CNN-VB1. Furthermore they are summarized in Tab. A.1. The MLP topology comprises three hidden layers that successively reduce in size, with the final layer mapped to the latent feature space. The CNN architectures CNN-VA1, CNN-VA2, and CNN-VB1 share a common design of convolutional layers with kernel size 25 and stride one, pooling layers with stride two, ReLU activations, and two fully connected layers producing the feature vector. CNN-VA1 employs two convolution–pooling blocks, CNN-VA2 extends this with an additional block to capture higher-order relations and reduce vector size, while CNN-VB1 arranges its blocks with two convolutional layers before each pooling step. All architectures were evaluated with latent feature spaces of two, three, four, and eight dimensions.

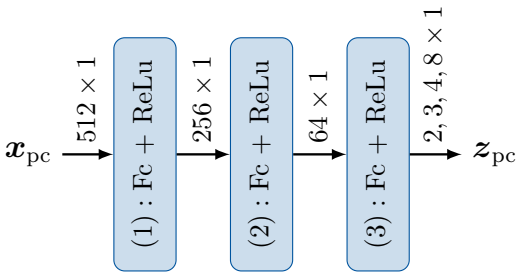

Figure A.7: Instantiated encoder topologies for MLP

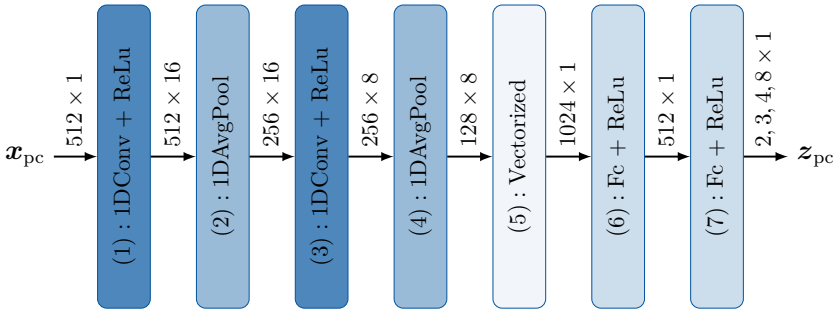

Figure A.8: Instantiated encoder topologies for CNN-VA1.

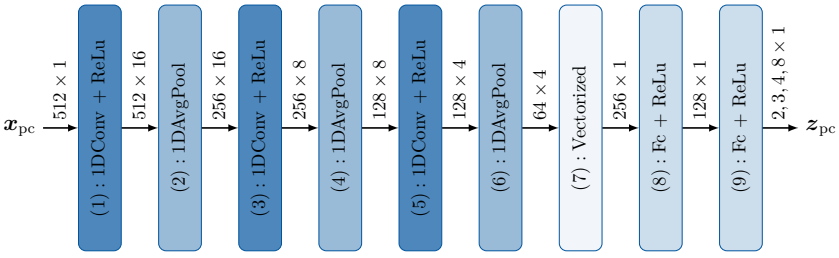

Figure A.9: Instantiated encoder topologies for CNN-VA2.

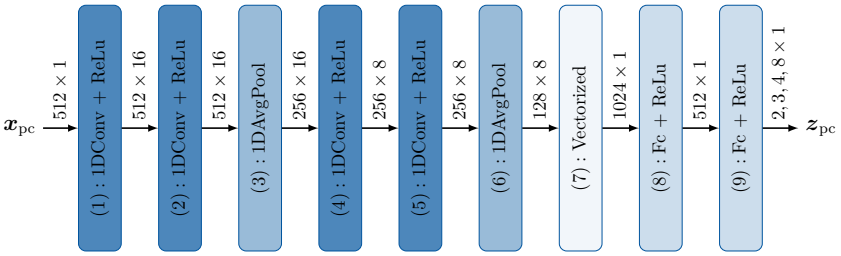

Figure A.10: Instantiated encoder topologies for CNN-VB1.

Table A.1: Instantiated encoder topologies.

Model	Layer	Layer type	Output dim.
MLP	0	Input layer	512×1
	1	Fc + ReLu	256×1
	2	Fc + ReLu	64×1
	3	Fc + ReLu	$2, 3, 4, 8 \times 1$
CNN-VA1	0	Input layer	512×1
	1	1DConv 1×25 + ReLu	512×16
	2	1DAvgPool	256×16
	3	1DConv 1×7 + ReLu	256×8
	4	1DAvgPool	128×8
	5	Vectorized	1024×1
	6	Fc + ReLu	512×1
	7	Fc + ReLu	$2, 3, 4, 8 \times 1$
CNN-VA2	0	Input layer	512×1
	1	1DConv 1×25 + ReLu	512×16
	2	1DAvgPool	256×16
	3	1DConv 1×7 + ReLu	256×8
	4	1DAvgPool	128×8
	5	1DConv 1×7 + ReLu	128×4
	6	1DAvgPool	64×4
	7	Vectorized	256×1
	8	Fc + ReLu	128×1
	9	Fc + ReLu	$2, 3, 4, 8 \times 1$
CNN-VB1	0	Input layer	512×1
	1	1DConv 1×25 + ReLu	512×16
	2	1DConv 1×7 + ReLu	512×16
	3	1DAvgPool	256×16
	4	1DConv 1×7 + ReLu	256×8
	5	1DConv 1×7 + ReLu	256×8
	6	1DAvgPool	128×8
	7	Vectorized	1024×1
	8	Fc + ReLu	512×1
	9	Fc + ReLu	$2, 3, 4, 8 \times 1$

A.2.2 Feature processing networks

The various feature processing task network topologies developed in this study are shown in Fig. A.11 for welding gun classification, Fig. A.12 for stell sheet combinations classification, Fig. A.13 for welding spot diameter estimation, and Fig. A.14 for the reconstruction of the process curves. Furthermore they are summarized in Tab. A.2. For classification of welding gun types and steel sheet combinations, the feature vector is first expanded to a higher-dimensional space to enable non-linear class separation, then reduced again before a Softmax layer assigns class probabilities. The estimator for welding spot diameter uses the same initial two layers, followed by a linear regressor outputting a continuous value. The decoder employs an up-sampling scheme, progressively expanding the feature vector through three fully connected ReLU layers and a final linear layer to reconstruct the original process curve.

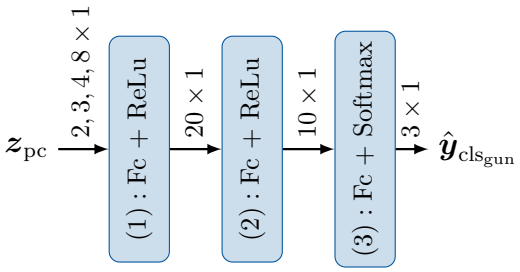

Figure A.11: Latent feature space processing networks for classifier welding gun type.

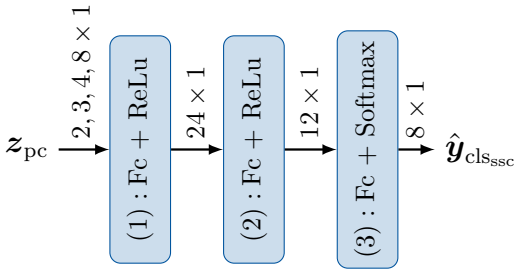

Figure A.12: Latent feature space processing networks for classifier steel sheet combination.

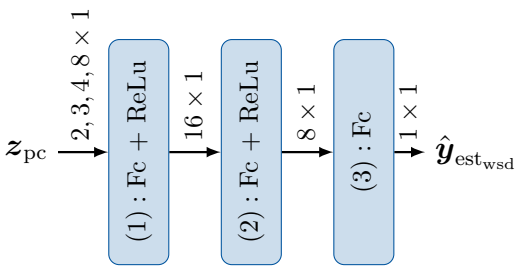

Figure A.13: Latent feature space processing networks for estimator welding spot diameter.

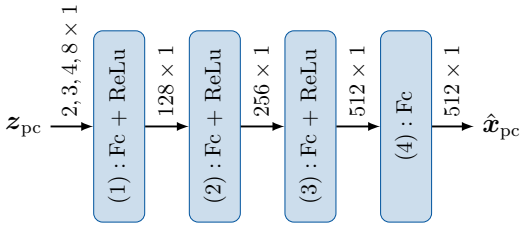

Figure A.14: Latent feature space processing networks for decoder.

Table A.2: Latent feature space processing networks.

Task	Layer	Layer type	Output dim.
Classifier welding gun type	0	Input layer	$2, 3, 4, 8 \times 1$
	1	Fc + ReLu	20×1
	2	Fc + ReLu	10×1
	3	Fc + Softmax	3×1
Classifier steel sheet combination	0	Input layer	$2, 3, 4, 8 \times 1$
	1	Fc + ReLu	24×1
	2	Fc + ReLu	12×1
	3	Fc + Softmax	8×1
Estimator welding spot diameter	0	Input layer	$2, 3, 4, 8 \times 1$
	1	Fc + ReLu	16×1
	2	Fc + ReLu	8×1
	3	Fc	1×1
Decoder	0	Input layer	$2, 3, 4, 8 \times 1$
	1	Fc + ReLu	128×1
	2	Fc + ReLu	256×1
	3	Fc + ReLu	512×1
	4	Fc	512×1

A.2.3 Results of further transformations

Further transformations are investigated to explore the condition space defined by the welding guns and steel sheet combinations of the sample application. The corresponding results are shown in Fig. A.15, Fig. A.16, and Fig. A.17. Each result presents (a) contour plots of the original densities (red and blue) together with the transformed density (green), (b) the transformed sample points (green) compared with the original points in feature space, and (c)–(d) reconstructed process curves for two selected samples. In the latter, the first curve (red) is reconstructed from an original sample under condition 1, while the second curve (green) is reconstructed from the nearest transformed neighbor under condition 2, based on the previously estimated transformation between the two conditions.

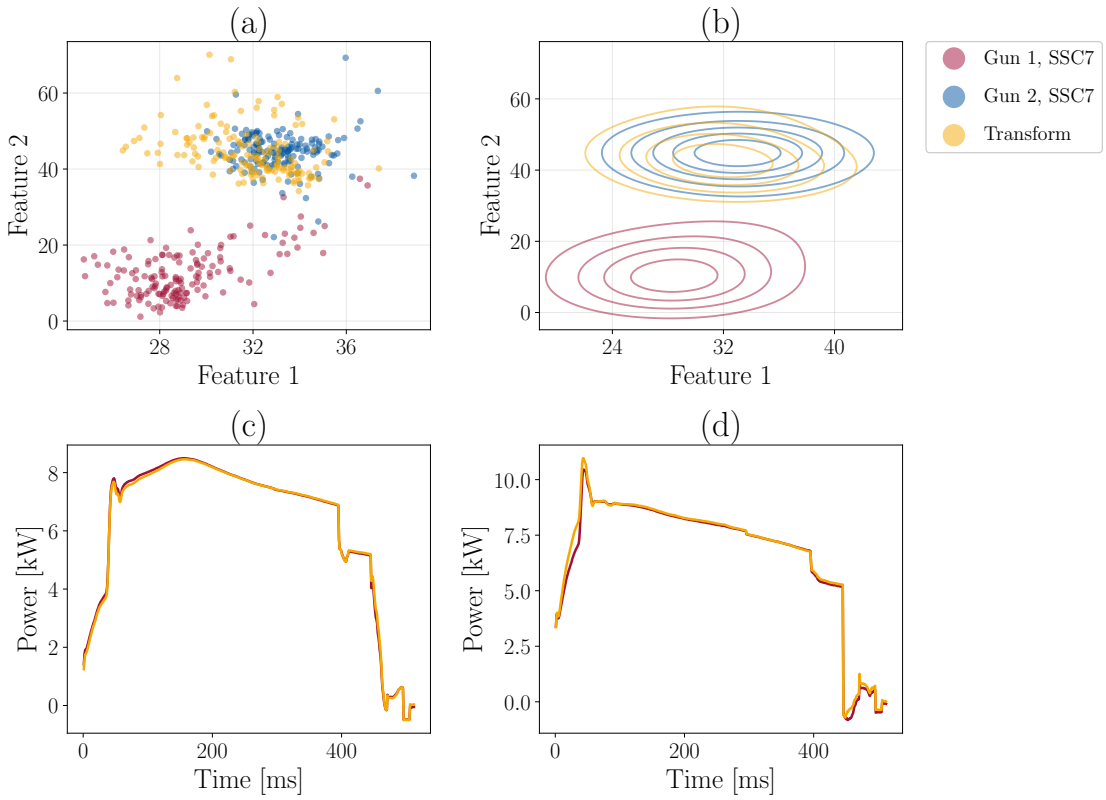

Figure A.15: Investigated transformation: (Gun1, SSC7) → (Gun2, SSC7).

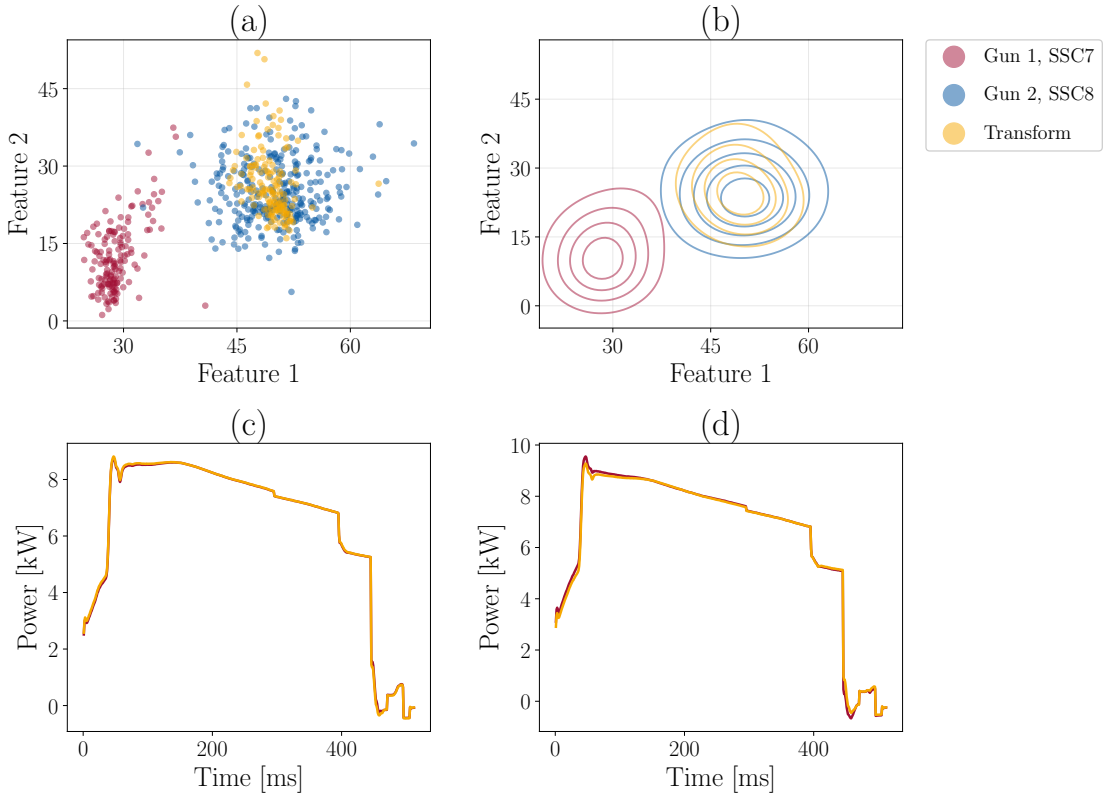

Figure A.16: Investigated transformation: (Gun1, SSC7) → (Gun1, SSC8).

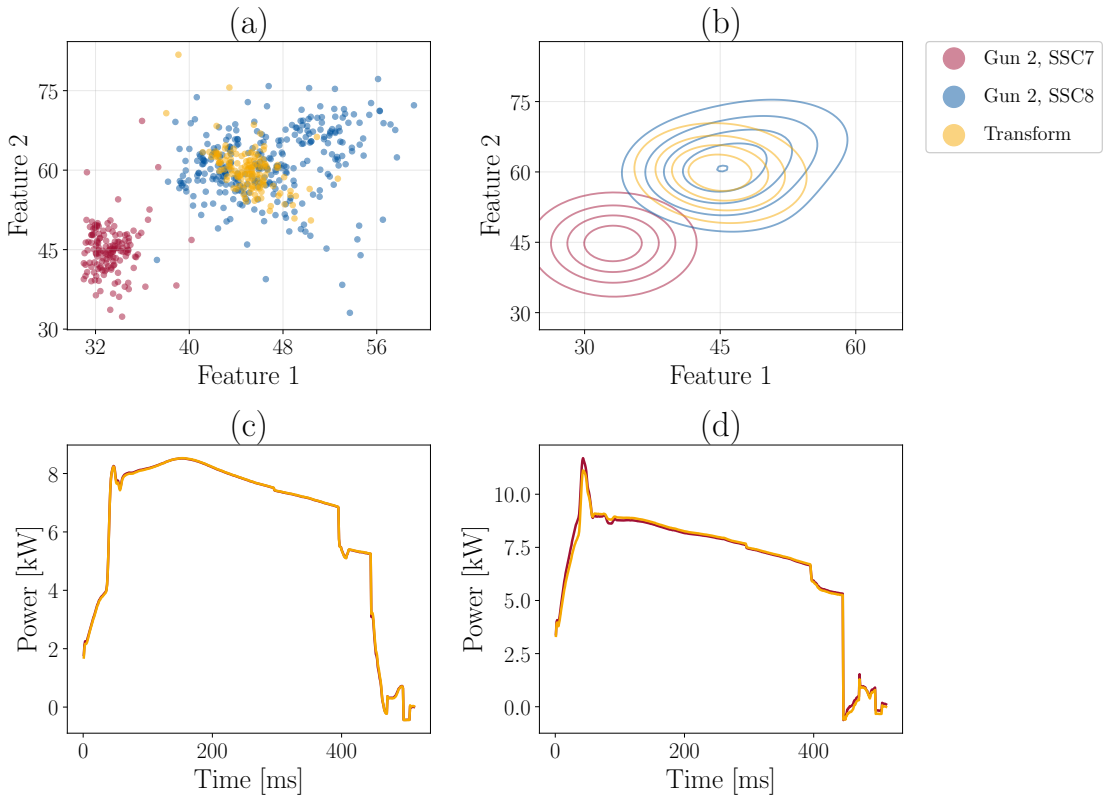

Figure A.17: Investigated transformation: (Gun2, SSC7) → (Gun2, SSC8)

A.3 Supporting information for Chapter 5

A.3.1 One twin part of the siamese multitask learning model

One twin part of the Siamese multitask learning model is shown in Fig. A.18 and Tab. A.3, which serves as the basis for the Siamese multitask learning approach. The architecture is organized around a shared encoder that derives a compact latent representation from the high-dimensional input, which is then processed by task-specific networks. The encoder reduces the dimensionality of the 512 input features stepwise through fully connected layers with Tanh activations, first to 256, then to 128, then to 64, and finally to a latent space of size 16.

From this shared representation, three distinct task networks branch off. The decoder reconstructs the original input by progressively expanding the latent space through three fully connected layers with Tanh activations, reaching 64, 128, and 256 dimensions, before a final linear layer restores the original dimensionality of 512. The regression head predicts a six-dimensional property vector. It first reduces the latent representation to 12 neurons, expands it to 25, reduces it again to 12, and finally produces the six outputs through a linear layer. The validity head evaluates process reachability by transforming the latent space through successive fully connected layers with Tanh activations of sizes 10, 6, and 2, followed by an expansion back to 6 and 10 neurons, before a final linear layer reconstructs a 16-dimensional output.

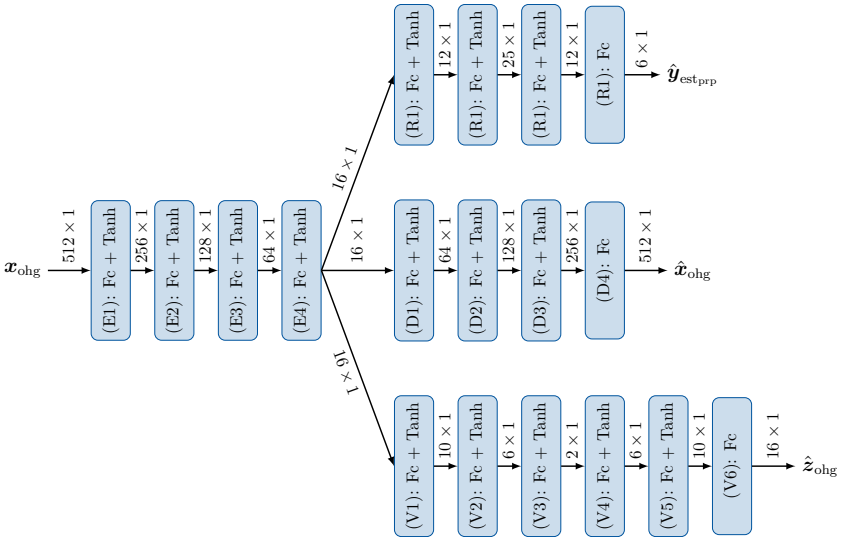

Figure A.18: One twin part of the Siamese multitask learning model with the annotation of the dimension size of the layers. *Fc* denotes fully-connected layers and *tanh* denotes hyperbolic tangent activation function.

Table A.3: Topologies of the developed multi task learning model.

Model part	Layer	Layer type	Output dim.
Encoder	0	Input layer	512×1
	1	Fc + Tanh	256×1
	2	Fc + Tanh	128×1
	3	Fc + Tanh	64×1
	4	Fc + Tanh	16×1
Decoder	0	Input layer	16×1
	1	Fc + Tanh	64×1
	2	Fc + Tanh	128×1
	3	Fc + Tanh	256×1
	4	Fc	512×1
Regression	0	Input layer	16×1
	1	Fc + Tanh	12×1
	2	Fc + Tanh	25×1
	3	Fc + Tanh	12×1
	4	Fc	6×1
Validity	0	Input layer	16×1
	1	Fc + Tanh	10×1
	2	Fc + Tanh	6×1
	3	Fc + Tanh	2×1
	4	Fc + Tanh	6×1
	5	Fc + Tanh	10×1
	6	Fc	16×1

A.4 Supporting information for Chapter 6

A.4.1 Generalized spherical harmonics distance

The distance between two ODFs $f^{(1)}(g)$ and $f^{(2)}(g)$ can be expressed according to Eq. (6.1) as

$$\begin{aligned}\mathscr{D}_{\mathrm{odf}}(f^{(1)}(g), f^{(2)}(g)) = \frac{1}{8\pi^2} \int_0^{2\pi}\int_0^{\pi}\int_0^{2\pi} \\ \left(f^{(1)}(\varphi_1, \Phi, \varphi_2) - f^{(2)}(\varphi_1, \Phi, \varphi_2)\right)^2 \\ d\varphi_1 \sin(\Phi) d\Phi d\varphi_2.\end{aligned} \tag{A.12}$$

When the densities $f^{(1)}(g)$ and $f^{(2)}(g)$ are expanded in two series of GSH base functions

$$f_{\mathrm{g}}^{(1)}(g) = \sum_{l=0}^{\infty} \sum_{m=-l}^{+l} \sum_{n=-l}^{+l} C_l^{mn} e^{im\varphi_2} P_l^{mn}(cos(\Phi)) e^{in\varphi_1} \tag{A.13}$$

and

$$f_{\mathrm{g}}^{(2)}(g) = \sum_{l=0}^{\infty} \sum_{m=-l}^{+l} \sum_{n=-l}^{+l} \tilde{C}_l^{mn} e^{im\varphi_2} P_l^{mn}(cos(\Phi)) e^{in\varphi_1}. \tag{A.14}$$

The distance measure of Eq. (A.12) becomes

$$\begin{aligned}\mathscr{D}_{\mathrm{gsh}}(f_{\mathrm{g}}^{(1)}(g), f_{\mathrm{g}}^{(2)}(g)) = \frac{1}{8\pi^2} \int_0^{2\pi}\int_0^{\pi}\int_0^{2\pi} \\ \left(\sum_{l=0}^{\infty} \sum_{m=-l}^{+l} \sum_{n=-l}^{+l} (C_l^{mn} - \tilde{C}_l^{mn}) e^{im\varphi_2} P_l^{mn}(\cos(\Phi)) e^{in\varphi_1}\right)^2 \\ d\varphi_1 \sin(\Phi) d\Phi d\varphi_2.\end{aligned} \tag{A.15}$$

Substituting $u = cos(\Phi)$ yields $du = -sin(\Phi)d\Phi$ and

$$\begin{aligned}\mathscr{D}_{\mathrm{gsh}}(f_{\mathrm{g}}^{(1)}(g), f_{\mathrm{g}}^{(2)}(g)) = \frac{1}{8\pi^2} \int_0^{2\pi}\int_{-1}^{+1}\int_0^{2\pi} \\ \left(\sum_{l=0}^{\infty} \sum_{m=-l}^{+l} \sum_{n=-l}^{+l} (C_l^{mn} - \tilde{C}_l^{mn}) e^{im\varphi_2} P_l^{mn}(u) e^{in\varphi_1}\right)^2 \\ d\varphi_1 du d\varphi_2.\end{aligned} \tag{A.16}$$

All mixed terms contain expressions of

$$\begin{aligned}&\int_0^{2\pi}\int_{-1}^{+1}\int_0^{2\pi} e^{i(m+v)\varphi_2} P_l^{mn}(u) P_p^{vw}(u) e^{i(n+w)\varphi_1} d\varphi_1 du d\varphi_2 \\ &\qquad\qquad \text{with} \quad m \neq v, n \neq u, l \neq w \\ &= \int_0^{2\pi} e^{i(m+v)\varphi_2} d\varphi_2 \int_0^{2\pi} e^{i(n+w)\varphi_1} d\varphi_1 \int_{-1}^{+1} P_l^{mn}(u) P_p^{vw}(u) du = 0 \\ &\qquad\qquad \text{for} \quad m \neq v, n \neq w, l \neq p.\end{aligned} \tag{A.17}$$

The mixed terms therefore all vanish due to orthogonality and only the squared terms remain

$$\mathscr{D}_{\mathrm{gsh}}(f_{\mathrm{g}}^{(1)}(g), f_{\mathrm{g}}^{(2)}(g)) = \frac{1}{8\pi^2} \int_0^{2\pi} \int_{-1}^{+1} \int_0^{2\pi} \left[\sum_{l=0}^{\infty} \sum_{m=-l}^{+l} \sum_{n=-l}^{+l} (C_l^{mn} - \tilde{C}_l^{mn})^2 (e^{im\varphi_2})^2 (P_l^{mn}(u))^2 (e^{in\varphi_1})^2 \right] d\varphi_1 du d\varphi_2. \tag{A.18}$$

The integration of the terms containing the integrands gives

$$\begin{aligned} &\int_0^{2\pi} \int_{-1}^{+1} \int_0^{2\pi} (e^{im\varphi_2})^2 (P_l^{mn}(u))^2 (e^{in\varphi_1})^2 d\varphi_1 du \varphi_2 \\ &= \int_0^{2\pi} (e^{im\varphi_2})^2 d\varphi_2 \int_0^{2\pi} (e^{in\varphi_1})^2 d\varphi_1 \int_{-1}^{+1} (P_l^{mn}(u))^2 du \end{aligned} \tag{A.19}$$

The integrads $(e^{im\varphi_2})^2$ and $(e^{in\varphi_1})^2$ are always equal to 1.
Therefore

$$\int_0^{2\pi} (e^{im\varphi_2})^2 d\varphi_2 \int_0^{2\pi} (e^{in\varphi_1})^2 d\varphi_1 = 4\pi^2 \tag{A.20}$$

The integral over the Legendre polynomials is

$$\int_{-1}^{+1} (P_l^{mn}(u))^2 du = \frac{2}{2l+1} \tag{A.21}$$

Inserting all terms into

$$\mathscr{D}_{\mathrm{gsh}}(f_{\mathrm{g}}^{(1)}(g), f_{\mathrm{g}}^{(2)}(g)) = \frac{1}{8\pi^2} \int_0^{2\pi} \int_{-1}^{+1} \int_0^{2\pi} \left(\sum_{l=0}^{\infty} \sum_{m=-l}^{+l} \sum_{n=-l}^{+l} (C_l^{mn} - \tilde{C}_l^{mn})^2 (e^{im\varphi_2})^2 (P_l^{mn}(u))^2 (e^{in\varphi_1})^2 \right) d\varphi_1 du d\varphi_2. \tag{A.22}$$

The distance measure between orientation densities $f_{\mathrm{g}}^{(1)}(g)$ and $f_{\mathrm{g}}^{(2)}(g)$, when expressed as series expansion of generalized spherical harmonics is then finally

$$\mathscr{D}_{\mathrm{gsh}}(f_{\mathrm{g}}^{(1)}(g), f_{\mathrm{g}}^{(2)}(g)) = \sum_{l=0}^{\infty} \sum_{m=-l}^{+l} \sum_{n=-l}^{+l} \frac{1}{2l+1} (C_l^{mn} - \tilde{C}_l^{mn})^2. \tag{A.23}$$

A.4.2 Details of structure-property-mapping

The details of the neural network topologies developed in this study for structure-property mappings are summarized in Tab. A.4 and detailed in Figs. A.19–A.22. The orientation histogram model (Tab. A.4 (a), Fig. A.19) employs three fully connected hidden layers that successively halve in size, followed by two processing layers and a final linear regression layer

predicting six properties $\hat{\boldsymbol{y}}_{\text{est}_{\text{prp}}}$. The GSH model with L=10 (Tab. A.4 (b), Fig. A.20) uses three halving fully connected layers and a final regression layer for $\hat{\boldsymbol{y}}_{\text{est}_{\text{prp}}}$, while the GSH model with L=16 (Tab. A.4 (c), Fig. A.21) first reduces the input to 160 neurons, then applies four halving hidden layers before the regression output. The pole figure CNN model (Tab. A.4 (d), Fig. A.22) consists of convolutional layers with 3×3 kernels and stride one, pooling layers with stride two in a Gaussian resolution pyramid, two stacked convolution–pooling blocks, and a vectorization step feeding four fully connected layers before the regression output.

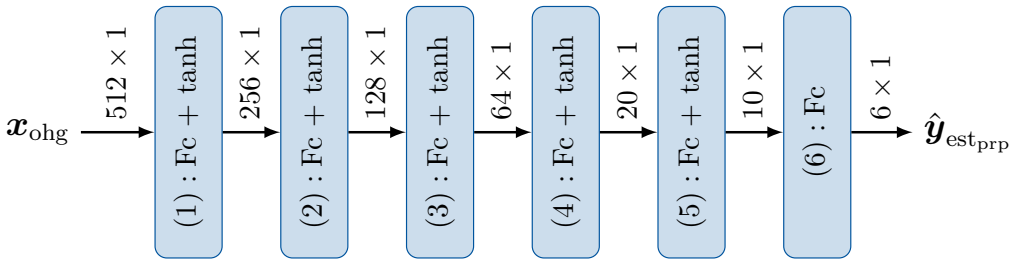

Figure A.19: Topologies of the neural networks for learning the texture-property relations by the texture representation as orientation histograms.

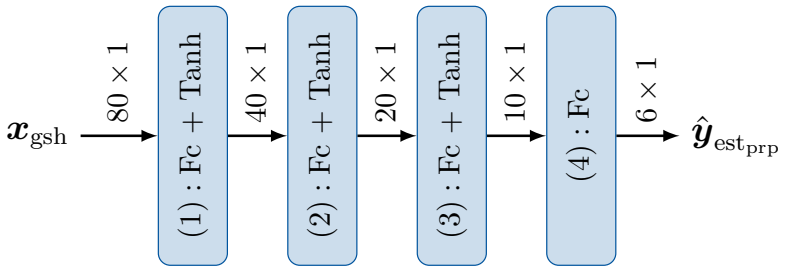

Figure A.20: Topologies of the neural networks for learning the texture-property relations by the texture representation as GSH for L=10.

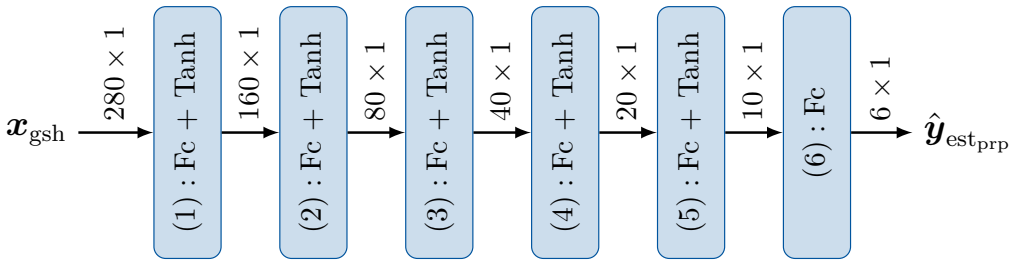

Figure A.21: Topologies of the neural networks for learning the texture-property relations by the texture representation as GSH for L=16.

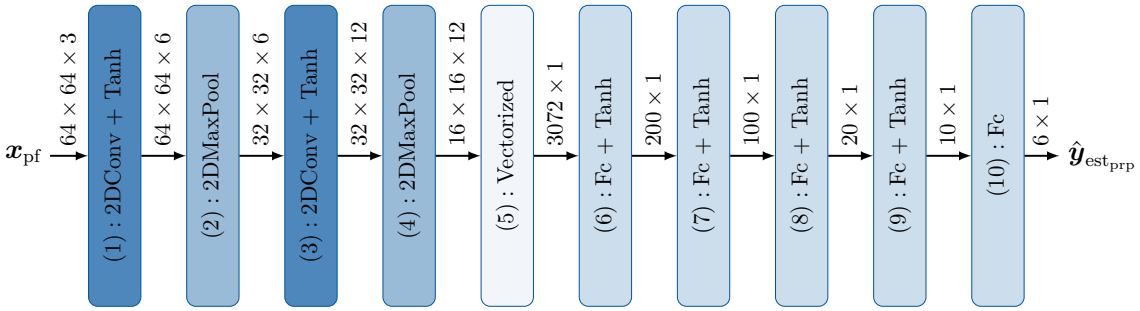

Figure A.22: Topologies of the neural networks for learning the texture-property relations by the texture representation as Pole figures.

Table A.4: Topologies of the developed neural networks models for learning the structure-property relations.

Texture	Layer	Layer type	Output dim.
(a) Orientation histogram	0	Input layer	512 × 1
	1	Fc + Tanh	256 × 1
	2	Fc + Tanh	128 × 1
	3	Fc + Tanh	64 × 1
	4	Fc + Tanh	20 × 1
	5	Fc + Tanh	10 × 1
	6	Fc	6 × 1
(b) GSH, L=10	0	Input layer	80 × 1
	1	Fc + Tanh	40 × 1
	2	Fc + Tanh	20 × 1
	3	Fc + Tanh	10 × 1
	4	Fc	6 × 1
(c) GSH, L=16	0	Input layer	280 × 1
	1	Fc + Tanh	160 × 1
	2	Fc + Tanh	80 × 1
	3	Fc + Tanh	40 × 1
	4	Fc + Tanh	20 × 1
	5	Fc + Tanh	10 × 1
	6	Fc	6 × 1
(d) Pole figures	0	Input layer	64 × 64 × 3
	1	2DConv 3 × 3 + Tanh	64 × 64 × 6
	2	2DMaxPool	32 × 32 × 6
	3	2DConv 3 × 3 + Tanh	32 × 32 × 12
	4	2DMaxPool	16 × 16 × 12
	5	Vectorized	3072 × 1
	6	Fc + Tanh	200 × 1
	7	Fc + Tanh	100 × 1
	8	Fc + Tanh	20 × 1
	9	Fc + Tanh	10 × 1
	10	Fc	6 × 1

A.4.3 Overview of the methods of the generation the orientation distribution function and measuring the distances

An overview of the methods of data generation and distance measurements is depicted in Fig. A.23.

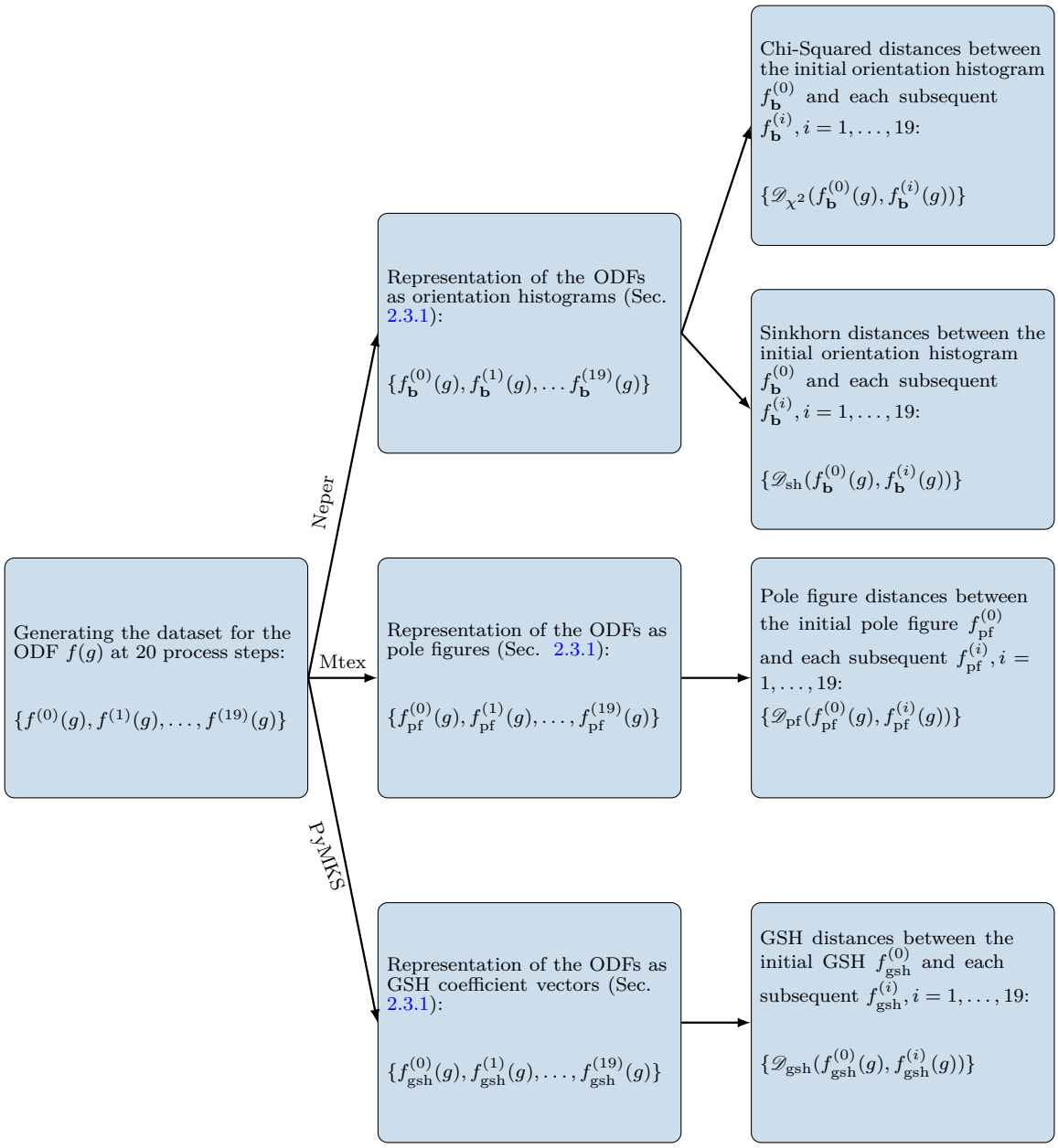

Figure A.23: Overview of the methods of the generation the orientation distribution functions and measuring the distances by the corresponding distance functions. The green squares indicate the used software for the approximation of the orientation distribution functions. The set of orientation distribution functions represents the textures resulting from the simulated forming process at different process steps (see Section 6.1.3).

A.5 Supporting information for Chapter 7

A.5.1 Visualization of the optimizer evolution in the property space

When the SMTLO approach is applied using the trained SMTL model, the optimizer iteratively minimizes the objective in Eq. (7.12) across generations. The initial population (generation 0) is widely distributed in the property space, as shown in Fig. A.24 for Target Region 1 and Fig. A.25 for Target Region 2, where property loss relative to the target regions is indicated by the color bar. Subsequent generations demonstrate a steady convergence toward the target regions, with clear improvements already visible after 10 generations and a majority of properties lying within the target regions by generation 100, while the remainder are positioned nearby. Throughout the optimization, candidate structures are collected only if they both satisfy the property requirements (fall within the target region) and remain process-reachable by not exceeding the validity threshold.

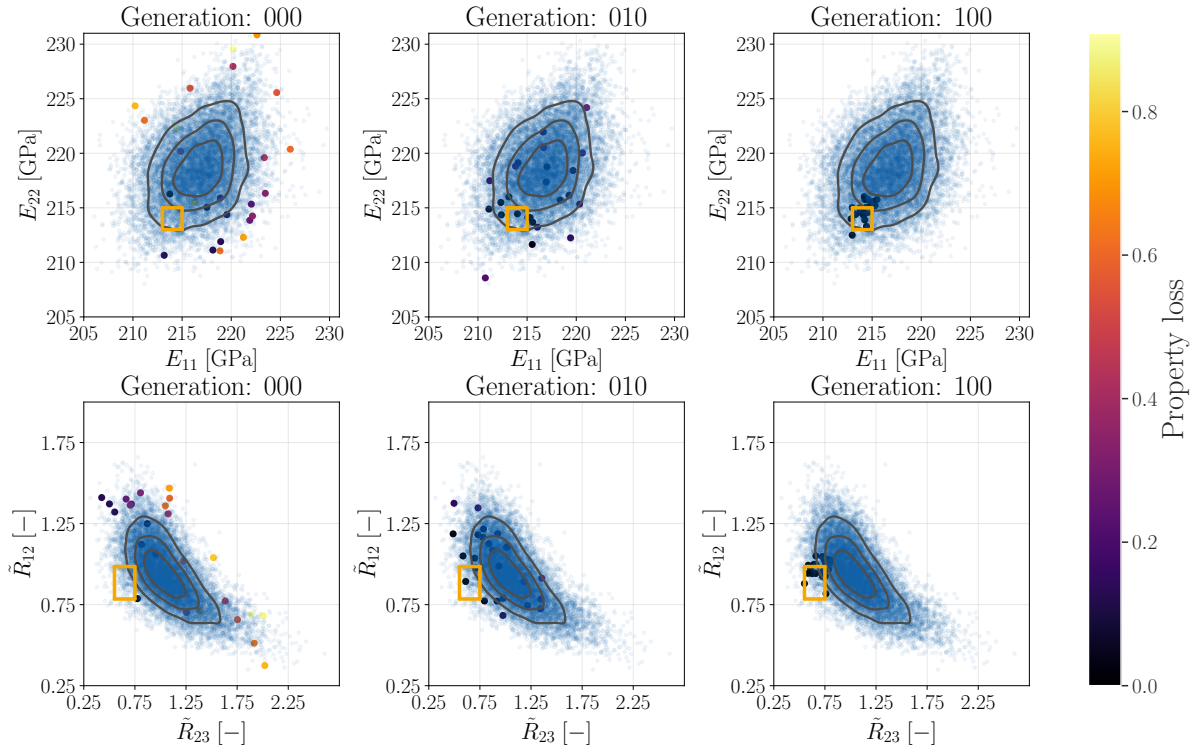

Figure A.24: Projections of the properties space showing the Target Region 1 and the evolution of the optimization for the generations 0, 10, and 100.

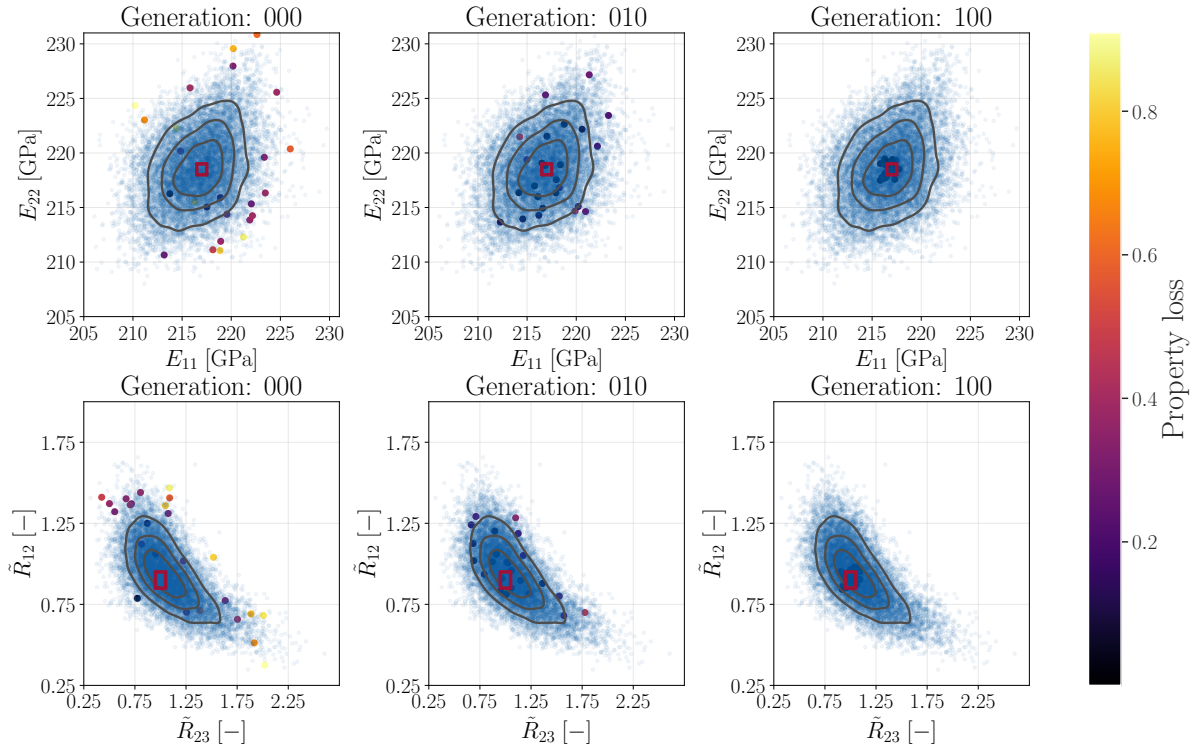

Figure A.25: Projections of the properties space showing the Target Region 2 and the evolution of the optimization for the generations 0, 10, and 100.

A.5.2 Visualization of the validity and regression results

The property prediction results yield MAE values of 0.368 [GPa] for Young's modulus and 0.032 [-] for $\widetilde{R}$-values. As shown in Fig. A.26, predicted and true values align closely, with R-values achieving a slightly higher R^2 score, indicating greater training difficulty. Furthermore, a two-dimensional projection of the latent feature space was obtained using the UMAP algorithm (McInnes et al., 2018). The resulting distribution is shown in Fig. A.27, where data points are color-coded by validity loss (Fig. A.27 (b)) and regression loss (Fig. A.27 (d)), with the corresponding distributions of these losses displayed as histograms (Fig. A.27 (a) and Fig. A.27 (c)).

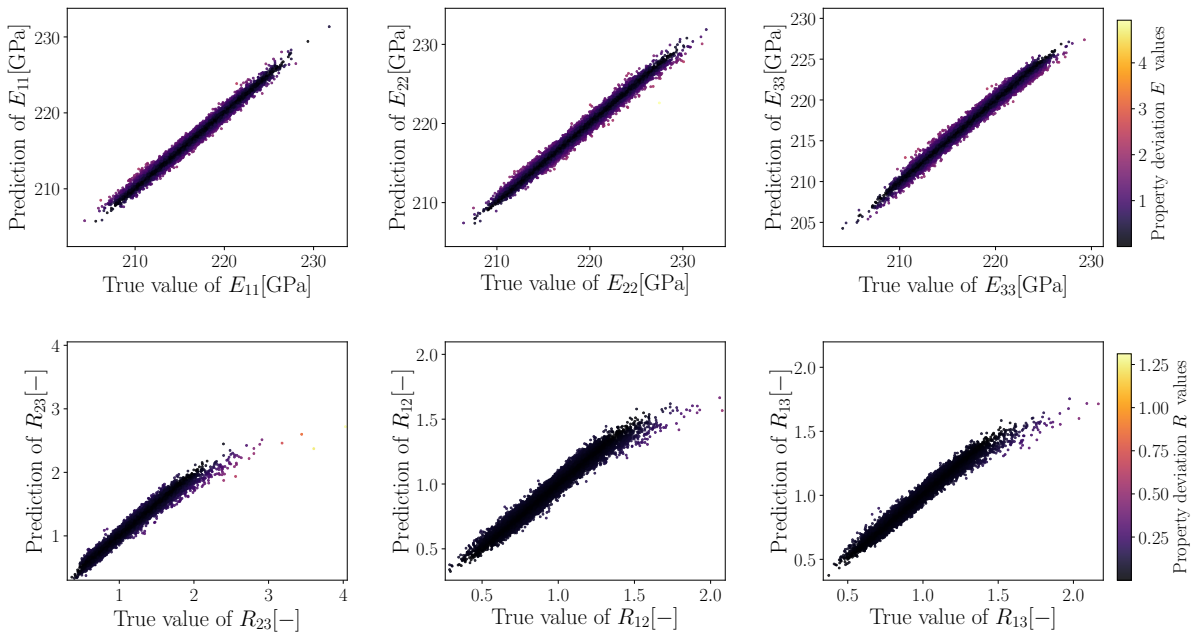

Figure A.26: Results of the Siamese multitask learning model for predicting the properties.

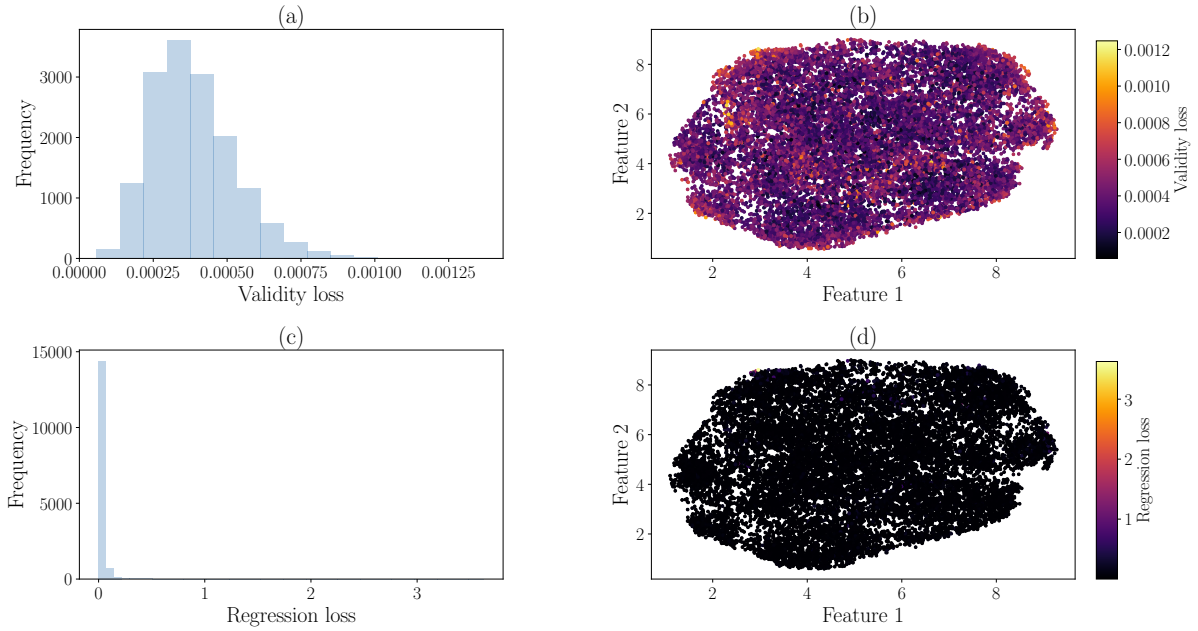

Figure A.27: (a) Distribution of the validity loss, (b) Two-Dimensional UMAP space, (c) Distribution of the regression loss, (d) Two-Dimensional UMAP space.

A.5.3 Distance preservation

The quality of distance preservation is evaluated using the coefficient of determination R^2 between the Sinkhorn distance of input textures and the l_1 distance of their latent feature vectors, yielding a value of 90.22 [%]. As shown in Fig. A.28, the l_1 distances in the latent space z are slightly lower at some points but overall closely preserve the distances from the original space.

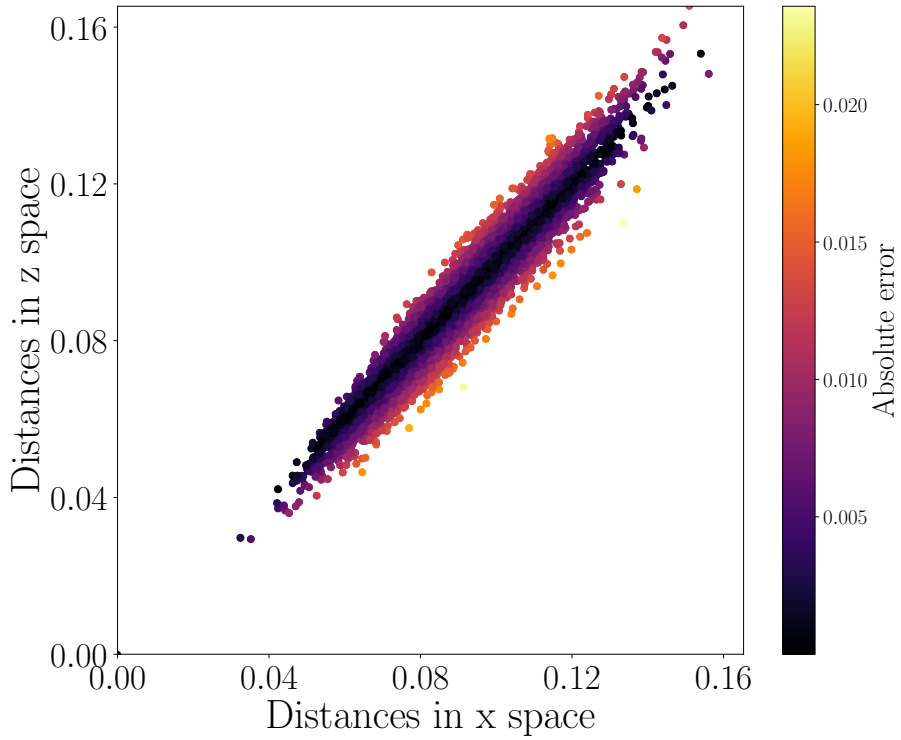

Figure A.28: Visualization of the Sinkhorn distances $\mathscr{D}_{\text{sh}}$ (x-axis) and the l_1 distances $\mathscr{D}_{l_1}$ (y-axis). The distances are color coded due to the absolute deviation, where a dark color indicates a lower deviation and a bright color indicates a higher deviation.

Bibliography

M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng (2021). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Version 2.5.0. DOI: 10.5281/zenodo.4724125.

P. Acar and V. Sundararaghavan (2016). „Linear Solution Scheme for Microstructure Design with Process Constraints“. In: *AIAA Journal* 54.12, pp. 4022–4031. DOI: 10.2514/1.J055247.

P. Acar and V. Sundararaghavan (2018). „Reduced-Order Modeling Approach for Materials Design with a Sequence of Processes“. In: *AIAA Journal* 56.12, pp. 5041–5044. DOI: 10.2514/1.J057221.

B. L. Adams, A. Henrie, B. Henrie, M. Lyon, S. R. Kalidindi, and H. Garmestani (2001). „Microstructure-Sensitive Design of a Compliant Beam“. In: *Journal of the Mechanics and Physics of Solids* 49.8, pp. 1639–1663. DOI: 10.1016/S0022-5096(01)00016-3.

Advanced Materials Initiative 2030 (2022). *The Materials Roadmap 2030*. URL: https://www.ami2030.eu/roadmap/ (visited on 08/27/2025).

A. Agrawal and A. Choudhary (2016). „Perspective: Materials informatics and big data: Realization of the “fourth paradigm” of science in materials science“. In: *APL Materials* 4.5, p. 053208. DOI: 10.1063/1.4946894.

A. Agrawal and A. Choudhary (2019). „Deep materials informatics: Applications of deep learning in materials science“. In: *MRS Communications* 9.3, pp. 779–792. DOI: 10.1557/mrc.2019.73.

L. S. Aiken and S. G. West (1991). *Multiple Regression: Testing and Interpreting Interactions*. SAGE Publications. ISBN: 978-0-7619-0712-1.

J.-B. Alayrac, J. Donahue, P. Luc, A. Miech, I. Barr, Y. Hasson, K. Lenc, A. Mensch, K. Millican, M. Reynolds, E. Rutherford, S. Borgeaud, K. Millican, J. Borgeaud, et al. (2022). „Flamingo: a Visual Language Model for Few-Shot Learning“. In: *NIPS'22: Proceedings of*

the 36th International Conference on Neural Information Processing Systems, pp. 23716–23736. DOI: 10.48550/arXiv.2204.14198.

J. Allison (2011). „Integrated computational materials engineering: A perspective on progress and future steps“. In: *JOM: the journal of the Minerals, Metals & Materials Society* 63, pp. 15–18. DOI: 10.1007/s11837-011-0053-y.

G. Arvanitidis, L. K. Hansen, and S. Hauberg (2017). „Latent Space Oddity: on the Curvature of Deep Generative Models“. In: *International Conference on Learning Representations (ICLR)*. DOI: 10.48550/arXiv.1710.11379.

R. J. Asaro and A. Needleman (1985). „Overview No. 42: Texture Development and Strain Hardening in Rate Dependent Polycrystals“. In: *Acta Metallurgica* 33.6, pp. 923–953. DOI: 10.1016/0001-6160(85)90188-9.

F. Bachmann, R. Hielscher, and H. Schaeben (2010). „Texture Analysis with MTEX – Free and Open Source Software Toolbox“. In: *Solid State Phenomena* 160, pp. 63–68. DOI: 10.4028/www.scientific.net/SSP.160.63.

M. Baiker, D. Helm, and A. Butz (2014). „Determination of Mechanical Properties of Polycrystals by Using Crystal Plasticity and Numerical Homogenization Schemes“. In: *Steel Research International* 85.6, pp. 988–998. DOI: 10.1002/srin.201300202.

N. Baishnab, E. Herron, A. Balu, S. Sarkar, A. Krishnamurthy, and B. Ganapathysubramanian (2024). „Latent diffusion models for 3D microstructure generation with process–structure linkages“. In: *AI4Mat-2024: NeurIPS 2024 Workshop on AI for Accelerated Materials Design*. DOI: 10.48550/arXiv.2503.10711.

P. V. Balachandran, D. Xue, J. Theiler, J. Hogden, and T. Lookman (2016). „Adaptive strategies for materials design using uncertainties“. In: *Scientific Reports* 6, p. 19660. DOI: 10.1038/srep19660.

C. Bauckhage, F. Beaumont, and S. Müller (2021). *ML2R Coding Nuggets: Hopfield Nets for Max-Sum Diversification*. Tech. rep. MLAI, University of Bonn.

L. E. Baum and J. A. Eagon (1967). „An Inequality with Applications to Statistical Estimation for Probabilistic Functions of Markov Processes and to a Model for Ecology“. In: *Bulletin of the American Mathematical Society* 73.3, pp. 360–363. DOI: 10.1090/S0002-9904-1967-11751-8.

L. E. Baum, T. Petrie, G. Soules, and N. Weiss (1970). „A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains“. In: *The Annals of Mathematical Statistics* 41.1, pp. 164–171. DOI: 10.1214/aoms/1177697196.

M. G. Baydogan, G. Runger, and E. Tuv (2013). „A Bag-of-Features Framework to Classify Time Series“. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.11, pp. 2796–2802. DOI: 10.1109/TPAMI.2013.72.

M. Belkin and P. Niyogi (2001). „Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering“. In: *NIPS'01: Proceedings of the 15th International Conference on Neural Information Processing Systems: Natural and Synthetic*, pp. 585–591. DOI: 10.7551/mitpress/1120.003.0080.

R. Bellman (1957). *Dynamic Programming*. Princeton University Press. ISBN: 9780691079516. DOI: 10.1515/9781400835386.

Y. Bengio, A. Courville, and P. Vincent (2013). „Representation learning: A review and new perspectives“. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.8, pp. 1798–1828. DOI: 10.1109/TPAMI.2013.50.

Y. Bengio, P. Simard, and P. Frasconi (1994). „Learning Long-Term Dependencies with Gradient Descent is Difficult“. In: *IEEE Transactions on Neural Networks* 5.2, pp. 157–166. DOI: 10.1109/72.279181.

J. Bergstra and Y. Bengio (2012). „Random Search for Hyper-Parameter Optimization“. In: *Journal of Machine Learning Research* 13, pp. 281–305.

D. Berthelot, C. Raffel, A. Roy, and I. Goodfellow (2019). „Understanding and Improving Interpolation in Autoencoders via an Adversarial Regularizer“. In: *Proceedings of the 7th International Conference on Learning Representations (ICLR 2019)*. DOI: 10.48550/arXiv.1807.07543.

A. Bhattacharyya (1946). „On a Measure of Divergence between Two Multinomial Populations“. In: *Sankhyā: The Indian Journal of Statistics (1933-1960)* 7.4, pp. 401–406.

S. Bompas and S. Sandfeld (2023). „A Generative Model for Accelerated Inverse Modelling Using a Novel Embedding for Continuous Variables“. In: *Proceedings of the AI for Accelerated Materials Design (AI4Mat) Workshop at the 37th Conference on Neural Information Processing Systems (NeurIPS)*. DOI: 10.48550/arXiv.2311.11343.

I. Borg and P. J. F. Groenen (2005). *Modern Multidimensional Scaling: Theory and Applications*. 2nd. Springer. ISBN: 978-0387251509. DOI: 10.1007/0-387-28981-X.

A. Borovykh, S. Bohte, and C. W. Oosterlee (2019). „Dilated Convolutional Neural Networks for Time Series Forecasting“. In: *Journal of Computational Finance* 22.4, pp. 73–101. DOI: 10.21314/JCF.2019.358.

L. Breiman (2001). „Random Forests“. In: *Machine Learning* 45.1, pp. 5–32. DOI: 10.1023/A:1010933404324.

L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone (1984). *Classification and Regression Trees*. Taylor & Francis Group. ISBN: 978-1315139470.

J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah (1993). „Signature Verification Using a "Siamese" Time Delay Neural Network“. In: *NIPS'93: Proceedings of the 7th International*

Conference on Neural Information Processing Systems. Vol. 7, pp. 737–744. DOI: 10.1142/9789812797926_0003.

D. B. Brough, D. Wheeler, and S. R. Kalidindi (2017). „Materials Knowledge Systems in Python—a Data Science Framework for Accelerated Development of Hierarchical Materials“. In: *Integrating Materials and Manufacturing Innovation* 6, pp. 36–53. DOI: 10.1007/s40192-017-0089-0.

T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. (2020). „Language Models are Few-Shot Learners“. In: *NIPS'20: Proceedings of the 34th International Conference on Neural Information Processing Systems* 34, pp. 1877–1901. DOI: 10.48550/arXiv.2005.14165.

S. T. Bukkapatnam (2023). „Autonomous Materials Discovery and Manufacturing (AMDM): A Review and Perspectives“. In: *IISE Transactions* 55.1, pp. 75–93. DOI: 10.1080/24725854.2022.2089785.

H.-J. Bunge (1982). *Texture Analysis in Materials Science: Mathematical Methods*. Elsevier. ISBN: 978-0-408-10642-9. DOI: 10.1016/C2013-0-11769-2.

K. T. Butler, D. W. Davies, H. Cartwright, O. Isayev, and A. Walsh (2018). „Machine learning for molecular and materials science“. In: *Nature* 559, pp. 547–555. DOI: 10.1038/s41586-018-0337-2.

M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin (2021). „Emerging Properties in Self-Supervised Vision Transformers“. In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9650–9660. DOI: 10.1109/ICCV48922.2021.00951.

R. Caruana (1997). „Multitask Learning“. In: *Machine Learning* 28, pp. 41–75. DOI: 10.1023/A:1007379606734.

A. Cecen, H. Dai, Y. C. Yabansu, S. R. Kalidindi, and L. Song (2018). „Material Structure-Property Linkages Using Three-Dimensional Convolutional Neural Networks“. In: *Acta Materialia* 146, pp. 76–84. DOI: 10.1016/j.actamat.2017.11.053.

R. Chalapathy and S. Chawla (2019). „Deep Learning for Anomaly Detection: A Survey“. In: *arXiv preprint*. DOI: 10.48550/arXiv.1901.03407.

V. Chandola, A. Banerjee, and V. Kumar (2009). „Anomaly Detection: A Survey“. In: *ACM Computing Surveys (CSUR)* 41.15, pp. 1–58. DOI: 10.1145/1541880.1541882.

O. Chapelle, B. Schölkopf, and A. Zien (2009). „Semi-Supervised Learning“. In: *IEEE Transactions on Neural Networks* 20.3. DOI: 10.1109/TNN.2009.2015974.

T. Chen and C. Guestrin (2016). „XGBoost: A Scalable Tree Boosting System“. In: *KDD '16: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794. DOI: 10.1145/2939672.2939785.

T. Chen, S. Kornblith, M. Norouzi, and G. Hinton (2020). „A Simple Framework for Contrastive Learning of Visual Representations“. In: *ICML'20: Proceedings of the 37th International Conference on Machine Learning*, pp. 1597–1607. DOI: 10.48550/arXiv.2002.05709.

Y. Chen, E. Keogh, B. Hu, N. Begum, A. Bagnall, A. Mueen, and G. Batista (2015). *The UCR Time Series Classification Archive*. URL: https://www.cs.ucr.edu/~eamonn/time_series_data_2018/ (visited on 08/27/2025).

Z. Cheng, S. Wang, P. Zhang, S. Wang, X. Liu, and E. Zhu (2021). „Improved Autoencoder for Unsupervised Anomaly Detection“. In: *International Journal of Intelligent Systems* 36.12, pp. 7103–7125. DOI: 10.1002/int.22582.

D. Chicco (2021). „Siamese Neural Networks: An Overview“. In: *Artificial Neural Networks. Methods in Molecular Biology*. Vol. 2190, pp. 73–94. DOI: 10.1007/978-1-0716-0826-5_3.

G. Choi, C. Lee, J. Kim, I. Ye, K. Jung, and I. Park (2025). „Image-Guided Microstructure Optimization using Diffusion Models: Validated with Li-Mn-rich Cathode Precursors“. In: *arXiv preprint*. DOI: 10.48550/arXiv.2505.07906.

C. Cortes and V. Vapnik (1995). „Support-Vector Networks“. In: *Machine Learning* 20, pp. 273–297. DOI: 10.1007/BF00994018.

T. M. Cover and P. E. Hart (1967). „Nearest Neighbor Pattern Classification“. In: *IEEE Transactions on Information Theory* 13.1, pp. 21–27. DOI: 10.1109/TIT.1967.1053964.

M. A. A. Cox and T. F. Cox (2008). „Multidimensional Scaling“. In: *Handbook of Data Visualization*. Springer, pp. 315–347. DOI: 10.1007/978-3-540-33037-0_14.

B. D. Cullity and S. R. Stock (2001). *Elements of X-ray Diffraction*. 3rd. Upper Saddle River, NJ: Prentice Hall. ISBN: 978-0201610918.

M. Cuturi (2013). „Sinkhorn Distances: Lightspeed Computation of Optimal Transport“. In: *Advances in Neural Information Processing Systems 26 (NIPS 2013)*. Vol. 26, pp. 2292–2300. DOI: 10.48550/arXiv.1306.0895.

W. Dai, C. Dai, S. Qu, J. Li, and S. Das (2017). „Very Deep Convolutional Neural Networks for Raw Waveforms“. In: *Proceedings of the 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 421–425. DOI: 10.1109/ICASSP.2017.7952190.

A. Das (2017). „Calculation of Crystallographic Texture of BCC Steels During Cold Rolling“. In: *Journal of Materials Engineering and Performance* 26, pp. 2708–2720. DOI: 10.1007/s11665-017-2695-6.

L. Delannay, P. V. Houtte, and A. V. Bael (1999). „New Parameter Model for Texture Description in Steel Sheets“. In: *Textures and Microstructures* 31.3, pp. 151–175. DOI: 10.1155/TSM.31.151.

F. Di Fiore, M. Nardelli, and L. Mainini (2024). „Active Learning and Bayesian Optimization: A Unified Perspective to Learn with a Goal“. In: *Archives of Computational Methods in Engineering* 31, pp. 2985–3013. DOI: 10.1007/s11831-024-10064-z.

G. E. Dieter (1986). *Mechanical Metallurgy*. 3rd. London: McGraw-Hill. ISBN: 978-0070168930.

J. Dornheim, L. Morand, S. Zeitvogel, T. Iraki, N. Link, and D. Helm (2022). „Deep Reinforcement Learning Methods for Structure-Guided Processing Path Optimization“. In: *Journal of Intelligent Manufacturing* 33, pp. 333–352. DOI: 10.1007/s10845-021-01805-z.

A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby (2021). „An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale“. In: *International Conference on Learning Representations (ICLR)*. DOI: 10.48550/arXiv.2010.11929.

N. R. Draper and H. Smith (1998). *Applied Regression Analysis*. 3rd. Wiley-Interscience. ISBN: 978-0471170822. DOI: 10.1002/9781118625590.

C. Draxl and M. Scheffler (2018). „NOMAD: The FAIR concept for big data-driven materials science“. In: *MRS Bulletin* 43, pp. 676–682. DOI: 10.1557/mrs.2018.208.

H. Drucker, C. J. C. Burges, L. Kaufman, A. Smola, and V. Vapnik (1996). „Support Vector Regression Machines“. In: *Advances in Neural Information Processing Systems 9 (NIPS 1996)*. Vol. 9, pp. 155–161.

R. O. Duda, P. E. Hart, and D. G. Stork (2000). *Pattern Classification*. 2nd. Wiley. ISBN: 978-0-471-05669-0.

C. Düreth, P. Seibert, D. Rücker, S. Handford, M. Kästner, and M. Gude (2023). „Conditional diffusion-based microstructure reconstruction“. In: *Materials Today Communications* 35, p. 105608. DOI: 10.1016/j.mtcomm.2023.105608.

C. Eckart and G. Young (1936). „The Approximation of One Matrix by Another of Lower Rank“. In: *Psychometrika* 1.3, pp. 211–218. DOI: 10.1007/BF02288367.

P. Eisenlohr, M. Diehl, R. A. Lebensohn, and F. Roters (2013). „A spectral method solution to crystal elasto-viscoplasticity at finite strains“. In: *International Journal of Plasticity* 46, pp. 37–53. DOI: 10.1016/j.ijplas.2012.09.012.

P. Eisenlohr and F. Roters (2008). „Selecting a Set of Discrete Orientations for Accurate Texture Reconstruction“. In: *Computational Materials Science* 42.4, pp. 670–678. DOI: 10.1016/j.commatsci.2007.09.015.

J. E. van Engelen and H. H. Hoos (2020). „A Survey on Semi-Supervised Learning“. In: *Machine Learning* 109.2, pp. 373–440. DOI: 10.1007/s10994-019-05855-6.

O. Engler, G. Gottstein, J. Pospiech, and J. Jura (1994). „Statistics, Evaluation and Representation of Single Grain Orientation Measurements“. In: *Materials Science Forum* 157-162, pp. 259–274. DOI: 10.4028/www.scientific.net/MSF.157-162.259.

C. Esteban, S. L. Hyland, and G. Rätsch (2017). „Real-valued (Medical) Time Series Generation with Recurrent Conditional GANs“. In: *Proceedings of the Machine Learning for Health Workshop at the 31st Conference on Neural Information Processing Systems*. DOI: 10.48550/arXiv.1706.02633.

M. Ester, H.-P. Kriegel, J. Sander, and X. Xu (1996). „A Density-Based Algorithm for Discovering Clusters in Large Spatial Datases with Noise“. In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, pp. 226–231.

A. Fallani, L. M. Sandonas, and A. Tkatchenko (2024). „Inverse Mapping of Quantum Properties to Structures for Chemical Space of Small Organic Molecules“. In: *Nature Communications* 15.1, pp. 1–11. DOI: 10.1038/s41467-024-50401-1.

S. Fischer, O. Hensgen, M. Elshaabiny, and N. Link (2015). „Generating Low-Dimensional Nonlinear Process Representations by Ordered Features“. In: *Proceedings of the 15th IFAC Symposium on Information Control in Manufacturing*. IFAC, pp. 1037–1042. DOI: 10.1016/j.ifacol.2015.06.220.

E. Fix and J. L. H. Jr. (1989). „Discriminatory Analysis: Nonparametric Discrimination: Consistency Properties“. In: *International Statistical Review* 57.3, pp. 238–247. DOI: 10.2307/1403797.

C. Frank (1988). „Orientation Mapping: 1987 MRS Fall Meeting Von Hippel Award Lecture“. In: *MRS Bulletin* 13.3, pp. 24–30. DOI: 10.1557/S0883769400066112.

K. Fukushima (1980). „Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position“. In: *Biological Cybernetics* 36.4, pp. 193–202. DOI: 10.1007/BF00344251.

D. T. Fullwood, S. R. Niezgoda, B. L. Adams, and S. R. Kalidindi (2010). „Microstructure Sensitive Design for Performance Optimization“. In: *Progress in Materials Science* 55.6, pp. 477–562. DOI: 10.1016/j.pmatsci.2009.08.002.

X. Gao and J. Liang (2013). „Manifold Learning Algorithm DC-ISOMAP of Data Lying on the Well-Separated Multi-Manifold with Same Intrinsic Dimension“. In: *Journal of Computer Research and Development* 50.8, pp. 1690–1699.

Gemini Team, R. Anil, Y. Bai, S. Borgeaud, T. Cai, A. Clark, R. Ring, et al. (2023). „Gemini: A Family of Highly Capable Multimodal Models“. In: *arXiv preprint*. DOI: 10.48550/arXiv.2312.11805.

Z. Ghahramani and M. I. Jordan (1997). „Factorial Hidden Markov Models“. In: *Machine Learning* 29, pp. 245–273. DOI: 10.1023/A:1007425814087.

S. Ghosh and D. M. Dimiduk (2011). *Computational Methods for Microstructure–Property Relationships*. Springer. ISBN: 978-1-4419-0642-7. DOI: 10.1007/978-1-4419-0643-4.

X. Glorot and Y. Bengio (2010). „Understanding the Difficulty of Training Deep Feedforward Neural Networks“. In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Vol. 9. Proceedings of Machine Learning Research. PMLR, pp. 249–256.

X. Glorot, A. Bordes, and Y. Bengio (2011). „Deep Sparse Rectifier Neural Networks“. In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. Vol. 15. PMLR, pp. 315–323.

D. E. Goldberg (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley. ISBN: 978-0201157673.

D. E. Goldberg (1991). „Real-Coded Genetic Algorithms, Virtual Alphabets, and Blocking“. In: *Complex Systems* 5.2, pp. 139–167.

G. H. Golub and C. Reinsch (1970). „Singular Value Decomposition and Least Squares Solutions“. In: *Numerische Mathematik* 14.5, pp. 403–420. DOI: 10.1007/BF02163027.

I. Goodfellow, Y. Bengio, and A. Courville (2016). *Deep Learning*. MIT Press. ISBN: 978-0262035613.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio (2014). „Generative Adversarial Nets“. In: *Advances in Neural Information Processing Systems 27 (NeurIPS 2014)*. Vol. 27, pp. 2672–2680. DOI: 10.48550/arXiv.1406.2661.

P. Grant (2013). *New and Advanced Materials*. Evidence Paper 10. Government Office for Science. URL: https://www.gov.uk/government/publications/future-manufacturing-new-and-advanced-materials (visited on 08/27/2025).

A. Graves (2013). „Generating Sequences With Recurrent Neural Networks“. In: *arXiv preprint*. DOI: 10.48550/arXiv.1308.0850.

S. N. Grigoriev, S. V. Fedorov, and K. Hamdy (2019). „Materials, properties, manufacturing methods and cutting performance of ceramic cutting inserts: A review“. In: *Manufacturing Review* 6. DOI: 10.1051/mfreview/2019016.

K. Guo, Z. Yang, C. Yu, and M. J. Buehler (2021). „Artificial intelligence and machine learning in design of mechanical materials“. In: *Materials Horizons* 8.4, pp. 1153–1172. DOI: 10.1039/D0MH01451F.

A. Gupta, A. Çeçen, S. Goyal, A. K. Singh, and S. R. Kalidindi (2015). „Structure–Property Linkages Using a Data Science Approach: Application to a Non-Metallic Inclusion/Steel Composite System“. In: *Acta Materialia* 91, pp. 239–254. DOI: 10.1016/j.actamat.2015.02.045.

J. Hadamard (1902). „Sur les problèmes aux dérivées partielles et leur signification physique“. In: *Princeton University Bulletin* 13, pp. 49–52.

R. Hadsell, S. Chopra, and Y. LeCun (2006). „Dimensionality Reduction by Learning an Invariant Mapping“. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1735–1742. DOI: 10.1109/CVPR.2006.100.

X.-Q. Han (2025). „AI-Driven Inverse Design of Materials: Past, Present, and Future“. In: *Chinese Physics Letters* 42.2, p. 027403. DOI: 10.1088/0256-307X/42/2/027403.

J. Hansen, J. Pospiech, and K. Lücke (1978). *Tables for Texture Analysis of Cubic Crystals*. Springer-Verlag. ISBN: 978-3-540-08689-5.

H. van Hasselt, A. Guez, and D. Silver (2016). „Deep Reinforcement Learning with Double Q-Learning“. In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)*. Vol. 30, pp. 2094–2100. DOI: 10.1609/aaai.v30i1.10295.

E. Hellinger (1909). „Neue Begründung der Theorie quadratischer Formen von unendlich vielen Veränderlichen“. In: *Journal für die reine und angewandte Mathematik (Crelle's Journal)* 136, pp. 210–271. DOI: 10.1515/crll.1909.136.210.

F. Herrera, M. Lozano, and J. L. Verdegay (1998). „Tackling Real-Coded Genetic Algorithms: Operators and Tools for Behavioural Analysis“. In: *Artificial Intelligence Review* 12.4, pp. 265–319. DOI: 10.1023/A:1006504901164.

M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter (2017). „GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium“. In: *NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems)*. Vol. 30. DOI: https://doi.org/10.48550/arXiv.1706.08500.

I. Higgins, L. Matthey, A. Pal, and et al. (2017). „beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework“. In: *Proceedings of the International Conference on Learning Representations (ICLR)*.

G. E. Hinton (1987). „Learning Translation Invariant Recognition in a Massively Parallel Network“. In: *PARLE: Parallel Architectures and Languages Europe*. Vol. 258. Lecture Notes in Computer Science. Springer, pp. 1–13. DOI: 10.1007/3-540-17943-7_117.

G. E. Hinton and R. R. Salakhutdinov (2006). „Reducing the Dimensionality of Data with Neural Networks“. In: *Science* 313.5786, pp. 504–507. DOI: 10.1126/science.1127647.

F. L. Hitchcock (1941). „The Distribution of a Product from Several Sources to Numerous Localities“. In: *Journal of Mathematics and Physics* 20.1-4, pp. 224–230. DOI: 10.1002/sapm1941201224.

S. Hochreiter and J. Schmidhuber (1997). „Long Short-Term Memory“. In: *Neural Computation* 9.8, pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

J. H. Holland (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press. ISBN: 978-0-262-58111-0. DOI: 10.7551/mitpress/1090.001.0001.

M. Hölscher, D. Raabe, and K. Lücke (1991). „Rolling and Recrystallization Textures of BCC Steels“. In: *Steel Research* 62.12, pp. 567–575. DOI: 10.1002/srin.199100451.

J. J. Hopfield (1982). „Neural Networks and Physical Systems with Emergent Collective Computational Abilities“. In: *Proceedings of the National Academy of Sciences* 79.8, pp. 2554–2558. DOI: 10.1073/pnas.79.8.2554.

H. Hotelling (1933). „Analysis of a Complex of Statistical Variables into Principal Components“. In: *Journal of Educational Psychology* 24, pp. 417–441, 498–520. DOI: 10.1037/h0071325.

X. Hou, L. Shen, K. Sun, and G. Qiu (2017). „Deep Feature Consistent Variational Autoencoder“. In: *Proceedings of the 2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1133–1141. DOI: 10.1109/WACV.2017.131.

D. Q. Huynh (2009). „Metrics for 3D Rotations: Comparison and Analysis“. In: *Journal of Mathematical Imaging and Vision* 35.2, pp. 155–164. DOI: 10.1007/s10851-009-0161-2.

C. Hvarfner, E. O. Hellsten, F. Hutter, and L. Nardi (2023). „Self-Correcting Bayesian Optimization through Bayesian Active Learning“. In: *NIPS '23: Proceedings of the 37th International Conference on Neural Information Processing Systems*. DOI: 10.48550/arXiv.2304.11005.

H. Inagaki and T. Suda (1972). „The Development of Rolling Textures in Low-Carbon Steels“. In: *Texture* 1, pp. 129–140. DOI: 10.1155/TSM.1.129.

T. Iraki and N. Link (2022). „Generative models for capturing and exploiting the influence of process conditions on process curves“. In: *Journal of Intelligent Manufacturing* 33.2, pp. 473–492. DOI: 10.1007/s10845-021-01846-4.

T. Iraki, L. Morand, J. Dornheim, N. Link, and D. Helm (2024a). „A multi-task learning-based optimization approach for finding diverse sets of microstructures with desired properties“. In: *Journal of Intelligent Manufacturing* 35, pp. 1887–1903. DOI: 10.1007/s10845-023-02139-8.

T. Iraki, L. Morand, N. Link, S. Sandfeld, and D. Helm (2024b). „Accurate distance measures and machine learning of the texture-property relation for crystallographic textures represented by one-point statistics“. In: *Modelling and Simulation in Materials Science and Engineering* 32.5, p. 055001. DOI: 10.1088/1361-651X/ad4c81.

P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros (2017). „Image-to-Image Translation with Conditional Adversarial Networks“. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1125–1134. DOI: 10.1109/CVPR.2017.632.

A. Jain, S. P. Ong, G. Hautier, W. Chen, W. D. Richards, S. Dacek, S. Cholia, D. Gunter, D. Skinner, G. Ceder, and K. A. Persson (2013). „Commentary: The Materials Project: A

materials genome approach to accelerating materials innovation“. In: *APL Materials* 1.1, p. 011002. DOI: 10.1063/1.4812323.

L. Jing and Y. Tian (2021). „Self-Supervised Visual Feature Learning with Deep Neural Networks: A Survey“. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43.11, pp. 4037–4058. DOI: 10.1109/TPAMI.2020.2992393.

S. C. Johnson (1967). „Hierarchical Clustering Schemes“. In: *Psychometrika* 32.3, pp. 241–254. DOI: 10.1007/BF02289588.

J. Jung, J. I. Yoon, H. K. Park, H. Jo, and H. S. Kim (2020). „Microstructure Design Using Machine Learning Generated Low Dimensional and Continuous Design Space“. In: *Materialia* 11, p. 100689. DOI: 10.1016/j.mtla.2020.100690.

J. Jung, J. I. Yoon, H. K. Park, J. Y. Kim, and H. S. Kim (2019a). „An Efficient Machine Learning Approach to Establish Structure-Property Linkages“. In: *Computational Materials Science* 156, pp. 17–25. DOI: 10.1016/j.commatsci.2018.09.034.

J. Jung, J. I. Yoon, S.-J. Park, J.-Y. Kang, G. L. Kim, Y. H. Song, S. T. Park, K. W. Oh, and H. S. Kim (2019b). „Modelling Feasibility Constraints for Materials Design: Application to Inverse Crystallographic Texture Problem“. In: *Computational Materials Science* 168, pp. 361–367. DOI: 10.1016/j.commatsci.2018.10.017.

S. R. Kalidindi, C. A. Bronkhorst, and L. Anand (1992). „Crystallographic Texture Evolution in Bulk Deformation Processing of FCC Metals“. In: *Journal of the Mechanics and Physics of Solids* 40.3, pp. 537–569. DOI: 10.1016/0022-5096(92)80003-9.

S. R. Kalidindi, J. R. Houskamp, M. Lyons, and B. L. Adams (2004). „Microstructure Sensitive Design of an Orthotropic Plate Subjected to Tensile Load“. In: *International Journal of Plasticity* 20.8-9, pp. 1561–1575. DOI: 10.1016/j.ijplas.2003.11.007.

S. R. Kalidindi, S. R. Niezgoda, and A. A. Salem (2011). „Microstructure Informatics Using Higher-Order Statistics and Efficient Data-Mining Protocols“. In: *JOM* 63.4, pp. 34–41. DOI: 10.1007/s11837-011-0057-7.

R. Kamijyo, A. Ishii, S. Coppieters, and A. Yamanaka (2022). „Bayesian Texture Optimization Using Deep Neural Network-Based Numerical Material Test“. In: *International Journal of Mechanical Sciences* 223, p. 107285. DOI: 10.1016/j.ijmecsci.2022.107285.

J. Kennedy and R. Eberhart (1995). „Particle Swarm Optimization“. In: *Proceedings of the IEEE International Conference on Neural Networks*. Vol. 4, pp. 1942–1948. DOI: 10.1109/ICNN.1995.488968.

E. Keogh and C. A. Ratanamahatana (2005). „Exact Indexing of Dynamic Time Warping“. In: *Knowledge and Information Systems* 7.3, pp. 358–386. DOI: 10.1007/s10115-004-0154-9.

L. A. I. Kestens and H. Pirgazi (2016). „Texture Formation in Metal Alloys with Cubic Crystal Structures“. In: *Materials Science and Technology* 32.13, pp. 1303–1315. DOI: 10.1080/02670836.2016.1231746.

D. Khatamsaz, B. Vela, P. Singh, D. D. Johnson, D. Allaire, and R. Arróyave (2023). „Bayesian optimization with active learning of design constraints using an entropy-based approach“. In: *npj Computational Materials* 9.1, p. 49. DOI: 10.1038/s41524-023-01006-7.

D. P. Kingma and J. Ba (2015). „Adam: A Method for Stochastic Optimization“. In: *3rd International Conference on Learning Representations (ICLR)*. DOI: 10.48550/arXiv.1412.6980.

D. P. Kingma and M. Welling (2014). „Auto-Encoding Variational Bayes“. In: *2nd International Conference on Learning Representations (ICLR)*. DOI: 10.48550/arXiv.1312.6114.

A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. Berg, W.-Y. Lo, et al. (2023). „Segment Anything“. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. DOI: 10.1109/ICCV51070.2023.00371.

A. Kirsch, J. van Amersfoort, and Y. Gal (2019). „BatchBALD: Efficient and Diverse Batch Acquisition for Deep Bayesian Active Learning“. In: *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pp. 7026–7037. DOI: 10.48550/arXiv.1906.08158.

C. Klinkenberg, D. Raabe, and K. Lücke (1993). „Influence of Volume Fraction and Dispersion Rate of Grain-Boundary Cementite on the Cold-Rolling Textures of Low-Carbon Steel“. In: *Steel Research* 64.6, pp. 287–293. DOI: 10.1002/srin.199200512.

U. F. Kocks, C. N. Tomé, and H.-R. Wenk (1998). *Texture and Anisotropy: Preferred Orientations in Polycrystals and Their Effect on Materials Properties*. Cambridge University Press. ISBN: 9780521465168.

K. Koenuma, A. Yamanaka, I. Watanabe, and T. Kuwabara (2020). „Estimation of Texture-Dependent Stress-Strain Curve and r-Value of Aluminum Alloy Sheet Using Deep Learning“. In: *Materials Transactions* 61.12, pp. 2276–2283. DOI: 10.2320/matertrans.P-M2020853.

T. Kohonen, M. Somervuo, and E. Oja, eds. (2001). *Self-Organizing Maps*. 3rd. Vol. 30. Springer Series in Information Sciences. Springer. ISBN: 978-3-540-41189-2.

V. R. Konda and J. N. Tsitsiklis (2000). „Actor-Critic Algorithms“. In: *Advances in Neural Information Processing Systems 12 (NIPS 1999)*. Vol. 12, pp. 1008–1014.

A. Krizhevsky, I. Sutskever, and G. E. Hinton (2012). „ImageNet Classification with Deep Convolutional Neural Networks“. In: *Communications of the ACM* 60.6, pp. 84–90. DOI: 10.1145/3065386.

A. Krogh and J. A. Hertz (1991). „A Simple Weight Decay Can Improve Generalization“. In: *Advances in Neural Information Processing Systems 4 (NIPS 1991)*. Vol. 4, pp. 950–957.

J. Kruskal (1964). „Multidimensional Scaling by Optimizing Goodness of Fit to a Nonmetric Hypothesis“. In: *Psychometrika* 29.1, pp. 1–27. DOI: 10.1007/BF02289565.

C. J. Kuehmann and G. B. Olson (2009). „Computational materials design and engineering“. In: *Materials Science and Technology* 25.4, pp. 472–478. DOI: 10.1179/174328408X371967.

S. Kullback and R. A. Leibler (1951). „On Information and Sufficiency“. In: *The Annals of Mathematical Statistics* 22.1, pp. 79–86. DOI: 10.1214/aoms/1177729694.

S. Kumar, S. Tan, L. Zheng, and D. M. Kochmann (2020). „Inverse-designed spinodoid metamaterials“. In: *npj Computational Materials* 6.1, p. 73. DOI: 10.1038/s41524-020-0341-6.

M. Kuroda and S. Ikawa (2004). „Texture Optimization of Rolled Aluminum Alloy Sheets Using a Genetic Algorithm“. In: *Materials Science and Engineering: A* 385.1-2, pp. 235–244. DOI: 10.1016/j.msea.2004.06.029.

N. Kusampudi and M. Diehl (2023). „Inverse design of dual-phase steel microstructures using generative machine learning model and Bayesian optimization“. In: *International Journal of Plasticity* 171, p. 103776. DOI: 10.1016/j.ijplas.2023.103776.

G. Kwon, M. Prabhushankar, D. Temel, and G. AlRegib (2020). „Backpropagated Gradient Representations for Anomaly Detection“. In: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 206–226. DOI: 10.1007/978-3-030-58589-1_13.

H. Larochelle, D. Erhan, and Y. Bengio (2008). „Zero-data Learning of New Tasks“. In: *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, pp. 646–651.

R. C. Larson and A. R. Odoni (1981). *Urban Operations Research*. Prentice-Hall. ISBN: 978-0139394478.

Y. LeCun and Y. Bengio (1995). „Convolutional Networks for Images, Speech, and Time Series“. In: *The Handbook of Brain Theory and Neural Networks*. Ed. by M. A. Arbib. Cambridge, MA: MIT Press, pp. 255–258. ISBN: 978-0-262-01197-6.

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel (1990). „Handwritten Digit Recognition with a Back-Propagation Network“. In: *Advances in Neural Information Processing Systems*. Vol. 2, pp. 396–404.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner (1998). „Gradient-Based Learning Applied to Document Recognition“. In: *Proceedings of the IEEE* 86.11, pp. 2278–2324. DOI: 10.1109/5.726791.

Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio (1999). „Object Recognition with Gradient-Based Learning“. In: *Shape, Contour and Grouping in Computer Vision*. Vol. 1681. Lecture Notes in Computer Science. Springer, pp. 319–345. DOI: 10.1007/3-540-46805-6_19.

Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller (2012). „Efficient BackProp“. In: *Neural Networks: Tricks of the Trade*. Vol. 7700. Lecture Notes in Computer Science. Springer, pp. 9–48. DOI: 10.1007/978-3-642-35289-8_3.

S. Lee and J. Yun (2024). „Microstructure reconstruction with diffusion models“. In: *Materials and Manufacturing Processes* 39.1, pp. 72–82. DOI: 10.1080/15376494.2023.2198528.

F. Leeb, A. Rügge, S. Ewert, K.-R. Müller, and G. Montavon (2022). „Exploring Interventions in Latent Space“. In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 35, pp. 30092–30105. DOI: 10.48550/arXiv.2106.16091.

D. S. Li, H. Garmestani, and S. Ahzi (2007). „Processing path optimization to achieve desired texture in polycrystalline materials“. In: *Acta Materialia* 55.2, pp. 647–654. DOI: 10.1016/j.actamat.2006.04.041.

J. Lin, E. Keogh, L. Wei, and S. Lonardi (2007). „Experiencing SAX: a novel symbolic representation of time series“. In: *Data Mining and Knowledge Discovery* 15.2, pp. 107–144. DOI: 10.1007/s10618-007-0064-z.

J. Ling, M. Hutchinson, and E. Antono (2017). „High-Dimensional Materials and Process Optimization Using Data-Driven Experimental Design with Well-Calibrated Uncertainty Estimates“. In: *Integrating Materials and Manufacturing Innovation* 6, pp. 207–217. DOI: 10.1007/s40192-017-0098-z.

X. Ling, M. F. Horstemeyer, and G. P. Potirniche (2005). „On the Numerical Implementation of 3D Rate-Dependent Single Crystal Plasticity Formulations“. In: *International Journal for Numerical Methods in Engineering* 63.4, pp. 548–568. DOI: 10.1002/nme.1289.

N. Link, J. Pollak, and A. Sarveniazi (2016). „Concept for Finding Process Models for New Classes of Industrial Production Processes“. In: *The Fifth International Conference on Intelligent Systems and Applications (INTELLI 2016)*, pp. 91–96.

G. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. W. M. van der Laak, B. van Ginneken, and C. I. Sánchez (2017). „A Survey on Deep Learning in Medical Image Analysis“. In: *Medical Image Analysis* 42, pp. 60–88. DOI: 10.1016/j.media.2017.07.005.

R. Liu, A. Kumar, Z. Chen, A. Agrawal, V. Sundararaghavan, and A. Choudhary (2015). „A Predictive Machine Learning Approach for Microstructure Optimization and Materials Design“. In: *Scientific Reports* 5. DOI: 10.1038/srep11551.

Y. Liu, T. Zhao, W. Ju, and S. Shi (2017). „Materials discovery and design using machine learning“. In: *Journal of Materiomics* 3.3, pp. 159–177. DOI: 10.1016/j.jmat.2017.08.002.

T. Lookman, P. V. Balachandran, D. Xue, and R. Yuan (2019). „Active learning in materials science with emphasis on adaptive sampling using uncertainties for targeted design“. In: *npj Computational Materials* 5, p. 21. DOI: 10.1038/s41524-019-0153-8.

M. Lyon and B. L. Adams (2004). „Gradient-Based Non-Linear Microstructure Design“. In: *Journal of the Mechanics and Physics of Solids* 52.11, pp. 2569–2586. DOI: 10.1016/j.jmps.2004.04.009.

W. Lyu, W. Zhang, F. Du, and S. Zhang (2024). „Microstructure reconstruction by denoising diffusion probabilistic models“. In: *Scientific Reports* 14.1, p. 54861. DOI: 10.1038/s41598-024-54861-9.

A. L. Maas, A. Y. Hannun, and A. Y. Ng (2013). „Rectifier Nonlinearities Improve Neural Network Acoustic Models“. In: *Proceedings of the 30th International Conference on Machine Learning (ICML)*, p. 3.

L. van der Maaten and G. Hinton (2008). „Visualizing Data using t-SNE“. In: *Journal of Machine Learning Research* 9, pp. 2579–2605.

L. van der Maaten, E. Postma, and J. van den Herik (2009). „Dimensionality Reduction: A Comparative Review“. In: *Journal of Machine Learning Research* 10, pp. 66–71.

J. MacQueen (1967). „Some Methods for Classification and Analysis of Multivariate Observations“. In: *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*. Vol. 1. University of California Press, pp. 281–297.

P. C. Mahalanobis (1936). „On the Generalized Distance in Statistics“. In: *Proceedings of the National Institute of Sciences of India* 2, pp. 49–55. DOI: 10.1007/s13171-019-00164-5.

A. Mann and S. R. Kalidindi (2022). „Development of a Robust CNN Model for Capturing Microstructure-Property Linkages and Building Property Closures Supporting Material Design“. In: *Frontiers in Materials* 9, p. 851085. DOI: 10.3389/fmats.2022.851085.

J. K. Mason and C. A. Schuh (2008). „Hyperspherical Harmonics for the Representation of Crystallographic Texture“. In: *Acta Materialia* 56.20, pp. 6141–6155. DOI: 10.1016/j.actamat.2008.08.031.

J. K. Mason and C. A. Schuh (2009). „Expressing Crystallographic Textures through the Orientation Distribution Function: Conversion between Generalized Spherical Harmonic and Hyperspherical Harmonic Expansions“. In: *Metallurgical and Materials Transactions A* 40.11, pp. 2590–2602. DOI: 10.1007/s11661-009-9936-8.

W. S. McCulloch and W. Pitts (1943). „A Logical Calculus of the Ideas Immanent in Nervous Activity“. In: *The Bulletin of Mathematical Biophysics* 5, pp. 115–133. DOI: 10.1007/BF02478259.

D. L. McDowell (2007). „Simulation-Assisted Materials Design for the Concurrent Design of Materials and Products“. In: *JOM* 59.9, pp. 21–25. DOI: 10.1007/s11837-007-0111-7.

L. McInnes, J. Healy, and J. Melville (2018). „UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction“. In: *arXiv preprint*. DOI: 10.48550/arXiv.1802.03426.

M. D. McKay, R. J. Beckman, and W. J. Conover (1979). „A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code“. In: *Technometrics* 21.2, pp. 239–245. DOI: 10.2307/1268522.

C. J. Merz and M. J. Pazzani (1999). „A Principal Components Approach to Combining Regression Estimates“. In: *Machine Learning* 36.1-2, pp. 9–32. DOI: 10.1023/A:1007507221352.

H. Minkowski (1910). *Geometrie der Zahlen*. Teubner.

M. Minsky and S. Papert (1969). *Perceptrons: An Introduction to Computational Geometry*. MIT Press. ISBN: 978-0-262-13043-1. DOI: 10.7551/mitpress/11301.001.0001.

M. Mirza and S. Osindero (2014). „Conditional Generative Adversarial Nets“. In: *arXiv preprint*. DOI: 10.48550/arXiv.1411.1784.

T. M. Mitchell (1997). *Machine Learning*. McGraw-Hill. ISBN: 978-0070428072.

V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis (2015). „Human-level control through deep reinforcement learning“. In: *Nature* 518.7540, pp. 529–533. DOI: 10.1038/nature14236.

L. Morand, J. Dornheim, J. Pagenkopf, and D. Helm (2021). „Simulation of Texture Evolution for a Multi-Step Metal Forming Process“. In: *Fordatis - Research Data Repository of Fraunhofer-Gesellschaft*. URL: http://dx.doi.org/10.24406/fordatis/127.2 (visited on 08/27/2025).

L. Morand, T. Iraki, J. Dornheim, J. Pagenkopf, and D. Helm (2023). „Artificially Generated Crystallographic Textures of Steel Sheets and Their Corresponding Properties Calculated by a Taylor-Type Crystal Plasticity Model“. In: *Fordatis - Research Data Repository of Fraunhofer-Gesellschaft*. URL: http://dx.doi.org/10.24406/fordatis/131 (visited on 08/27/2025).

L. Morand, T. Iraki, J. Dornheim, S. Sandfeld, N. Link, and D. Helm (2024). „Machine learning for structure-guided materials and process design“. In: *Materials & Design* 248, p. 113453. DOI: 10.1016/j.matdes.2024.113453.

L. Morand, N. Link, T. Iraki, J. Dornheim, and D. Helm (2022). „Efficient Exploration of Microstructure-Property Spaces via Active Learning“. In: *Frontiers in Materials* 8, p. 824441. DOI: 10.3389/fmats.2021.824441.

A. Morawiec and J. Pospiech (1989). „Some Information on Quaternions Useful in Texture Calculations“. In: *Texture, Stress, and Microstructure* 10, pp. 211–216. DOI: 10.1155/TSM.10.211.

B. Moreau, F. Wagner, and H. Göbel (1994). „Optimization of the Texture Determination of Thin Films from X-Ray Diffraction Measurements“. In: *Materials Science Forum* 157-162, pp. 159–166. DOI: 10.4028/www.scientific.net/MSF.157-162.159.

M. Mozaffar, S. Liao, X. Xie, S. Saha, C. Park, J. Cao, W. K. Liu, and Z. Gan (2022). „Mechanistic artificial intelligence (mechanistic-AI) for modeling, design, and control of advanced manufacturing processes: Current state and perspectives“. In: *Journal of Materials Processing Technology* 302, p. 117485. DOI: 10.1016/j.jmatprotec.2021.117485.

Y. Nahshon, L. Morand, M. Büschelberger, D. Helm, K. Kumaraswamy, P. Zierep, M. Weber, and P. de Andrés (2025). „Semantic Orchestration and Exploitation of Material Data: A Dataspace Solution Demonstrated on Steel and Copper Applications“. In: *Advanced Engineering Materials* 27. DOI: 10.1002/adem.202401448.

V. Nair and G. E. Hinton (2010). „Rectified Linear Units Improve Restricted Boltzmann Machines“. In: *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 807–814.

National Research Council (2008). *Integrated Computational Materials Engineering: A Transformational Discipline for Improved Competitiveness and National Security*. National Academies Press. DOI: 10.17226/12199.

B. D. Nguyen, P. Potapenko, A. Demirci, K. Govind, S. Bompas, and S. Sandfeld (2024). „Efficient Surrogate Models for Materials Science Simulations: Machine Learning-Based Prediction of Microstructure Properties“. In: *Machine Learning with Applications* 16, p. 100654. DOI: 10.1016/j.mlwa.2024.100544.

S. R. Niezgoda, A. K. Kanjarla, and S. R. Kalidindi (2013). „Novel Microstructure Quantification Framework for Databasing, Visualization, and Analysis of Microstructure Data“. In: *Integrating Materials and Manufacturing Innovation* 2, pp. 54–80. DOI: 10.1186/2193-9772-2-3.

S. R. Niezgoda, Y. C. Yabansu, and S. R. Kalidindi (2011). „Understanding and Visualizing Microstructure and Microstructure Variance as a Stochastic Process“. In: *Acta Materialia* 59.16, pp. 6387–6400. DOI: 10.1016/j.actamat.2011.06.051.

A. H. Nobari, W. Chen, and F. Ahmed (2021). „PcDGAN: A Continuous Conditional Diverse Generative Adversarial Network for Inverse Design“. In: *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 1–9. DOI: 10.1145/3447548.3467414.

J. Noh, G. H. Gu, S. Kim, and Y. Jung (2020). „Machine-Enabled Inverse Design of Inorganic Solid Materials: Promises and Challenges“. In: *Chemical Science* 11.19, pp. 4871–4881. DOI: 10.1039/D0SC00594K.

C. G. Northcutt, L. Jiang, and I. L. Chuang (2021). „Confident learning: Estimating uncertainty in dataset labels“. In: *Journal of Artificial Intelligence Research* 70, pp. 1373–1411. DOI: 10.1613/jair.1.12125.

G. B. Olson (1997). „Computational Design of Hierarchically Structured Materials“. In: *Science* 277.5330, pp. 1237–1242. DOI: 10.1126/science.277.5330.1237.

OpenAI (2023). „GPT-4 Technical Report“. In: *arXiv preprint*. DOI: 10.48550/arXiv.2303.08774.

J. J. de Pablo, N. E. Jackson, M. A. Webb, L.-Q. Chen, J. E. Moore, D. Morgan, R. Jacobs, T. Pollock, D. G. Schlom, E. S. Toberer, J. Analytis, I. Dabo, D. M. DeLongchamp, G. A. Fiete, G. M. Grason, G. Hautier, Y. Mo, K. Rajan, E. J. Reed, E. Rodriguez, V. Stevanovic, J. Suntivich, K. Thornton, and J.-C. Zhao (2019). „New frontiers for the materials genome initiative“. In: *npj Computational Materials* 5.1, p. 41. DOI: 10.1038/s41524-019-0173-4.

J. Pagenkopf, A. Butz, M. Wenk, and D. Helm (2016). „Virtual Testing of Dual-Phase Steels: Effect of Martensite Morphology on Plastic Flow Behavior“. In: *Materials Science and Engineering: A* 674, pp. 672–686. DOI: 10.1016/j.msea.2016.07.118.

J. Pagenkopf (2019). *Bestimmung der plastischen Anisotropie von Blechwerkstoffen durch ortsaufgelöste Simulationen auf Gefügeebene*. Fraunhofer Verlag. ISBN: 978-3-8396-1425-9. DOI: 10.24406/publica-fhg-282559.

J. H. Panchal, S. R. Kalidindi, and D. L. McDowell (2013). „Key Computational Modeling Issues in Integrated Computational Materials Engineering“. In: *Computer-Aided Design* 45.1, pp. 4–25. DOI: 10.1016/j.cad.2012.06.006.

J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove (2019). „DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation“. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 165–174. DOI: 10.1109/CVPR.2019.00025.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala (2019). „PyTorch: An Imperative Style, High-Performance Deep Learning Library“. In: *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*, pp. 8024–8035. DOI: 10.48550/arXiv.1912.01703.

A. Paul, P. Acar, W.-k. Liao, A. Choudhary, V. Sundararaghavan, and A. Agrawal (2019). „Microstructure Optimization with Constrained Design Objectives Using Machine Learning-Based Feedback-Aware Data-Generation“. In: *Computational Materials Science* 160, pp. 334–351. DOI: 10.1016/j.commatsci.2019.01.015.

N. H. Paulson, M. W. Priddy, D. L. McDowell, and S. R. Kalidindi (2017). „Reduced-Order Structure-Property Linkages for Polycrystalline Microstructures Based on 2-Point Statistics“. In: *Acta Materialia* 129, pp. 428–438. DOI: 10.1016/j.actamat.2017.03.009.

K. Pearson (1901). „On Lines and Planes of Closest Fit to Systems of Points in Space“. In: *Philosophical Magazine* 2.11, pp. 559–572. DOI: 10.1080/14786440109462720.

O. Pele and M. Werman (2009). „Fast and Robust Earth Mover's Distances“. In: *Proceedings of the IEEE 12th International Conference on Computer Vision (ICCV)*, pp. 460–467. DOI: 10.1109/ICCV.2009.5459199.

O. Pele and M. Werman (2010). „The Quadratic-Chi Histogram Distance Family“. In: *European Conference on Computer Vision (ECCV)*, pp. 749–762. DOI: 10.1007/978-3-642-15552-9_54.

S. Peleg, M. Werman, and H. Rom (1989). „A Unified Approach to the Change of Resolution: Space and Gray-Level“. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 11.7, pp. 711–719. DOI: 10.1109/34.192468.

G. Pizzi, A. Cepellotti, R. Sabatini, N. Marzari, and B. Kozinsky (2016). „AiiDA: automated interactive infrastructure and database for computational science“. In: *Computational Materials Science* 111, pp. 218–230. DOI: 10.1016/j.commatsci.2015.09.013.

T. M. Pollock (2010). „Weight loss with magnesium alloys“. In: *Science* 328.5981, pp. 986–987. DOI: 10.1126/science.1182848.

D. A. Porter, K. E. Easterling, and M. Y. Sherif (2021). *Phase Transformations in Metals and Alloys*. 4th ed. CRC Press. ISBN: 9781003011804. DOI: 10.1201/9781003011804.

J. Pospiech (1972). „Die Parameter der Drehung und die Orientierungsverteilungsfunktion (OVF)“. In: *Crystal Research and Technology* 7.9, pp. 1083–1090. DOI: 10.1002/crat.19720070908.

L. Prechelt (1998). „Early Stopping - But When“. In: *Neural Networks: Tricks of the Trade*. Ed. by G. B. Orr and K.-R. Müller. Vol. 1524. Lecture Notes in Computer Science. Springer, pp. 55–69. DOI: 10.1007/3-540-49430-8_3.

G. Proust and S. R. Kalidindi (2006). „Procedures for Construction of Anisotropic Elastic–Plastic Property Closures for Face-Centered Cubic Polycrystals Using First-Order Bounding Relations“. In: *Journal of the Mechanics and Physics of Solids* 54.8, pp. 1744–1762. DOI: 10.1016/j.jmps.2006.01.010.

M. L. Puterman (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley-Interscience. ISBN: 978-0471619772. DOI: 10.1002/9780470316887.fmatter.

R. Quey, P. R. Dawson, and F. Barbe (2011). „Large-Scale 3D Random Polycrystals for the Finite Element Method: Generation, Meshing and Remeshing“. In: *Computer Methods in Applied Mechanics and Engineering* 200.17-20, pp. 1729–1745. DOI: 10.1016/j.cma.2011.01.002.

R. Quey, A. Villani, and C. Maurice (2018). „Nearly Uniform Sampling of Crystal Orientations“. In: *Journal of Applied Crystallography* 51.4, pp. 1162–1173. DOI: 10.1107/S1600576718009019.

A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever (2021). „Learning Transferable Visual Models From Natural Language Supervision“. In: *Proceedings of the 38th International Conference on Machine Learning (ICML)*. DOI: 10.48550/arXiv.2103.00020.

T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh (2012). „Searching and Mining Trillions of Time Series Subsequences under Dynamic Time Warping“. In: *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD 2012)*, pp. 262–270. DOI: 10.1145/2339530.2339576.

R. Ramprasad, R. Batra, G. Pilania, A. Mannodi-Kanakkithodi, and C. Kim (2017). „Machine Learning in Materials Informatics: Recent Applications and Prospects“. In: *npj Computational Materials* 3. DOI: 10.1038/s41524-017-0056-5.

V. Randle and O. Engler (2009). *Introduction to Texture Analysis: Macrotexture, Microtexture, and Orientation Mapping*. 2nd. Boca Raton, FL: CRC Press. ISBN: 978-1420056722.

A. Rasheed, O. San, and T. Kvamsdal (2020). „Digital Twin: Values, Challenges and Enablers From a Modeling Perspective“. In: *IEEE Access* 8, pp. 21980–22012. DOI: 10.1109/ACCESS.2020.2970143.

A. Raßloff, P. Seibert, K. A. Kalina, and M. Kästner (2025). „Inverse Design of Spinodoid Structures Using Bayesian Optimization“. In: *Computational Mechanics*. DOI: 10.1007/s00466-024-02587-w.

R. K. Ray, J. J. Jonas, and R. E. Hook (1994). „Cold Rolling and Annealing Textures in Low Carbon and Extra Low Carbon Steels“. In: *International Materials Reviews* 39.4, pp. 129–172. DOI: 10.1179/imr.1994.39.4.129.

S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee (2016). „Generative Adversarial Text to Image Synthesis“. In: *Proceedings of The 33rd International Conference on Machine Learning*, pp. 1060–1069. DOI: 10.48550/arXiv.1605.05396.

J. Reis, G. Gonçalves, and N. Link (2017). „Meta-Process Modeling Methodology for Process Model Generation in Intelligent Manufacturing“. In: *Proceedings of the 43rd Annual Conference of the IEEE Industrial Electronics Society (IECON)*, pp. 3396–3402. DOI: 10.1109/IECON.2017.8216575.

S. Ren, K. He, R. Girshick, and J. Sun (2017). „Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks“. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.6, pp. 1137–1149. DOI: 10.1109/TPAMI.2016.2577031.

J. R. Rice (1971). „Inelastic Constitutive Relations for Solids: An Internal-Variable Theory and Its Application to Metal Plasticity“. In: *Journal of the Mechanics and Physics of Solids* 19.6, pp. 433–455. DOI: 10.1016/0022-5096(71)90010-X.

F. Riesz (1910). „Untersuchungen über Systeme integrierbarer Funktionen“. In: *Mathematische Annalen* 69, pp. 449–497. DOI: 10.1007/BF01457637.

J. G. Rittig, K. Gao, J. F. Sacha, J. R. Kreißl, J. A. Voigt, J. M. Großmann, and J. S. Sandfeld (2024). „Graph Neural Networks for the Prediction of Molecular Structure–Property Relationships“. In: *Machine Learning for Molecular Sciences*. Royal Society of Chemistry, pp. 159–182. DOI: 10.1039/BK9781837670178-00159.

F. Rosenblatt (1958). „The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain“. In: *Psychological Review* 65.6, pp. 386–408. DOI: 10.1037/h0042519.

M. Rosenblatt (1956). „Remarks on Some Nonparametric Estimates of a Density Function“. In: *The Annals of Mathematical Statistics* 27.3, pp. 832–837. DOI: 10.1214/aoms/1177728190.

R. Rosipal and L. J. Trejo (2001). „Kernel Partial Least Squares Regression in Reproducing Kernel Hilbert Space“. In: *Journal of Machine Learning Research* 2, pp. 97–123.

F. Roters, M. Diehl, P. Shanthraj, P. Eisenlohr, C. Reuber, S. Wong, T. Maiti, A. Ebrahimi, T. Hochrainer, H. O. Fabritius, et al. (2019). „DAMASK—the Düsseldorf Advanced Material Simulation Kit for modeling multi-physics crystal plasticity, thermal, and damage phenomena from the single crystal up to the component scale“. In: *Computational Materials Science* 158, pp. 420–478. DOI: 10.1016/j.commatsci.2018.04.030.

F. Roters, P. Eisenlohr, L. Hantcherli, D. D. Tjahjanto, T. R. Bieler, and D. Raabe (2010). *Crystal Plasticity Finite Element Methods*. Wiley-VCH. ISBN: 978-3-527-32447-7.

S. T. Roweis and L. K. Saul (2000). „Nonlinear Dimensionality Reduction by Locally Linear Embedding“. In: *Science* 290.5500, pp. 2323–2326. DOI: 10.1126/science.290.5500.2323.

Y. Rubner, L. J. Guibas, and C. Tomasi (1997). „The Earth Mover’s Distance, Multi-Dimensional Scaling, and Color-Based Image Retrieval“. In: *Proceedings of the ARPA Image Understanding Workshop*, pp. 661–668.

Y. Rubner, C. Tomasi, and L. J. Guibas (2000). „The Earth Mover’s Distance as a Metric for Image Retrieval“. In: *International Journal of Computer Vision* 40.2, pp. 99–121. DOI: 10.1023/A:1026543900054.

S. Ruder (2017). „An Overview of Multi-Task Learning in Deep Neural Networks“. In: *arXiv preprint*. DOI: 10.48550/arXiv.1706.05098.

L. Ruff, N. Görnitz, L. Deecke, S. A. Siddiqui, R. A. Vandermeulen, A. Binder, E. Müller, and M. Kloft (2018). „Deep One-Class Classification“. In: *Proceedings of the 35th International Conference on Machine Learning (ICML)*. Vol. 80, pp. 4393–4402.

L. Ruff, J. R. Kauffmann, R. A. Vandermeulen, G. Montavon, W. Samek, M. Kloft, T. G. Dietterich, and K.-R. Müller (2021). „A Unifying Review of Deep and Shallow Anomaly

Detection“. In: *Proceedings of the IEEE* 109.5, pp. 756–795. DOI: 10.1109/JPROC.2021.3052449.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams (1986). „Learning Representations by Back-Propagating Errors“. In: *Nature* 323.6088, pp. 533–536. DOI: 10.1038/323533a0.

H. Sakoe and S. Chiba (1978). „Dynamic Programming Algorithm Optimization for Spoken Word Recognition“. In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 26.1, pp. 43–49. DOI: 10.1109/TASSP.1978.1163055.

M. Sakurada and T. Yairi (2014). „Anomaly Detection Using Autoencoders with Nonlinear Dimensionality Reduction“. In: *Proceedings of the 2nd Workshop on Machine Learning for Sensory Data Analysis (MLSDA)*, pp. 4–11. DOI: 10.1145/2689746.2689747.

J. W. Sammon (1969). „A Nonlinear Mapping for Data Structure Analysis“. In: *IEEE Transactions on Computers* C-18.5, pp. 401–409. DOI: 10.1109/T-C.1969.222678.

T. Schaul, J. Quan, I. Antonoglou, and D. Silver (2016). „Prioritized Experience Replay“. In: *4th International Conference on Learning Representations (ICLR)*. DOI: 10.48550/arXiv.1511.05952.

M. Scheffler, M. Aeschlimann, M. Albrecht, T. Bereau, H.-J. Bungartz, C. Felser, M. Greiner, A. Groß, C. T. Koch, K. Kremer, W. E. Nagel, M. Scheidgen, C. Wöll, and C. Draxl (2022). „FAIR Data Enabling New Horizons for Materials Research“. In: *Nature* 604.7907, pp. 635–642. DOI: 10.1038/s41586-022-04501-x.

U. von Schlippenbach, F. Emren, and K. Lücke (1986). „Investigation of the Development of the Cold Rolling Texture in Deep Drawing Steels by ODF-Analysis“. In: *Acta Metallurgica* 34.6, pp. 1001–1010. DOI: 10.1016/0001-6160(86)90015-5.

J. Schmidt, M. R. Marques, S. Botti, and M. A. Marques (2019). „Recent advances and applications of machine learning in solid-state materials science“. In: *npj Computational Materials* 5.83. DOI: 10.1038/s41524-019-0221-0.

B. Schölkopf, J. C. Platt, J. C. Shawe-Taylor, A. J. Smola, and R. C. Williamson (2001). „Estimating the Support of a High-Dimensional Distribution“. In: *Neural Computation* 13.7, pp. 1443–1471. DOI: 10.1162/089976601750264965.

B. Schölkopf, A. Smola, and K.-R. Müller (1997). „Kernel Principal Component Analysis“. In: *Artificial Neural Networks - ICANN 1997*. Springer, pp. 583–588. DOI: 10.1007/BFb0020217.

S. Schreijäg (2013). *Microstructure and Mechanical Behavior of Deep Drawing DC04 Steel at Different Length Scales*. KIT Scientific Publishing. ISBN: 978-3-86644-967-1. DOI: 10.5445/KSP/1000032165.

F. Schroff, D. Kalenichenko, and J. Philbin (2015). „FaceNet: A Unified Embedding for Face Recognition and Clustering“. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 815–823. DOI: 10.1109/CVPR.2015.7298682.

G. A. F. Seber and A. J. Lee (2012). *Linear Regression Analysis*. 2nd. John Wiley & Sons. ISBN: 978-1-118-27442-2.

P. Seibert, A. Raßloff, M. Ambati, and M. Kästner (2022). „Descriptor-Based Reconstruction of Three-Dimensional Microstructures Through Gradient-Based Optimization“. In: *Acta Materialia* 227, p. 117667. DOI: 10.1016/j.actamat.2022.117667.

M. Senn (2013). *Optimale Prozessführung mit merkmalsbasierter Zustandsverfolgung*. Vol. 18. Schriftenreihe des Instituts für Angewandte Materialien, Karlsruher Institut für Technologie. KIT Scientific Publishing. ISBN: 978-3-7315-0004-9. DOI: 10.5445/KSP/1000034025.

A. Senthil Kumar, A. Raja Durai, and T. Sornakumar (2006). „Wear behaviour of alumina based ceramic cutting tools on machining steels“. In: *Tribology International* 39.3, pp. 191–197. DOI: 10.1016/j.triboint.2005.01.021.

B. Settles (2009). *Active Learning Literature Survey*. Tech. rep. 1648. University of Wisconsin–Madison.

J. B. Shaffer, M. Knezevic, and S. R. Kalidindi (2010). „Building texture evolution networks for deformation processing of polycrystalline FCC metals using spectral approaches: Applications to process design for targeted performance“. In: *International Journal of Plasticity* 26.8, pp. 1183–1194. DOI: 10.1016/j.ijplas.2010.03.010.

H. Shao, A. Kumar, and P. T. Fletcher (2018). „The Riemannian Geometry of Deep Generative Models“. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. DOI: 10.1109/CVPRW.2018.00071.

T. W. Simpson, J. D. Peplinski, P. N. Koch, and J. K. Allen (2001). „Metamodels for Computer-based Engineering Design: Survey and Recommendations“. In: *Engineering with Computers* 17.2, pp. 129–150. DOI: 10.1007/PL00007198.

D. Sivan, K. Satheesh Kumar, A. Abdullah, V. Raj, I. I. Misnon, and S. Ramakrishna (2024). „Advances in materials informatics: a review“. In: *Journal of Materials Science* 59.7, pp. 2602–2643. DOI: 10.1007/s10853-023-09312-5.

R. Socher, M. Ganjoo, H. Sridhar, O. Bastani, C. D. Manning, and A. Y. Ng (2013). „Zero-Shot Learning Through Cross-Modal Transfer“. In: *Advances in Neural Information Processing Systems 26 (NIPS 2013)*. Vol. 26, pp. 935–943. DOI: 10.48550/arXiv.1301.3666.

K. Sohn, H. Lee, and X. Yan (2015). „Learning Structured Output Representation using Deep Conditional Generative Models“. In: *Advances in Neural Information Processing Systems 28 (NIPS 2015)*, pp. 3483–3491.

R. Souvenir and R. Pless (2005). „Manifold Clustering“. In: *Proceedings of the Tenth IEEE International Conference on Computer Vision*, pp. 648–653. DOI: 10.1109/ICCV.2005.149.

T. Standley, A. R. Zamir, D. Chen, L. Guibas, J. Malik, and S. Savarese (2020). „Which Tasks Should Be Learned Together in Multi-task Learning?“ In: *Proceedings of the 37th Interna-*

tional Conference on Machine Learning (ICML), pp. 9120–9132. DOI: 10.48550/arXiv.1905.07553.

R. Storn and K. Price (1997). „Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces“. In: *Journal of Global Optimization* 11.4, pp. 341–359. DOI: 10.1023/A:1008202821328.

S. Sundar and V. Sundararaghavan (2020). „Database development and exploration of process–microstructure relationships using variational autoencoders“. In: *Materials Today Communications* 25, p. 101201. DOI: 10.1016/j.mtcomm.2020.101201.

V. Sundararaghavan and N. Zabaras (2005). „On the Synergy Between Texture Classification and Deformation Process Sequence Selection for the Control of Texture-Dependent Properties“. In: *Acta Materialia* 53.4, pp. 1015–1027. DOI: 10.1016/j.actamat.2004.11.001.

R. S. Sutton and A. G. Barto (2018). *Reinforcement Learning: An Introduction*. 2nd. MIT Press. ISBN: 978-0262039246.

R. S. Sutton, A. G. Barto, and R. J. Williams (1992). „Reinforcement Learning is Direct Adaptive Optimal Control“. In: *IEEE Control Systems Magazine* 12.2, pp. 19–22. DOI: 10.1109/37.126844.

R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour (2000). „Policy Gradient Methods for Reinforcement Learning with Function Approximation“. In: *Advances in Neural Information Processing Systems 12*. MIT Press, pp. 1057–1063.

S. Swayamdipta, R. Schwartz, N. Lourie, Y. Wang, H. Hajishirzi, N. A. Smith, and Y. Choi (2020). „Dataset Cartography: Mapping and Diagnosing Datasets with Training Dynamics“. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, pp. 9275–9293. DOI: 10.18653/v1/2020.emnlp-main.746.

A. E. Tallman, K. S. Stopka, L. P. Swiler, Y. Wang, S. R. Kalidindi, and D. L. McDowell (2019). „Gaussian-Process-Driven Adaptive Sampling for Reduced-Order Modeling of Texture Effects in Polycrystalline Alpha-Ti“. In: *JOM* 71.8, pp. 2646–2656. DOI: 10.1007/s11837-019-03553-1.

R. K. Tan, N. L. Zhang, and W. Ye (2020). „A Deep Learning–Based Method for the Design of Microstructural Materials“. In: *Structural and Multidisciplinary Optimization* 61.4, pp. 1417–1438. DOI: 10.1007/s00158-019-02424-2.

J. Tarasiuk and K. Wierzbanowski (1996). „Application of the Linear Regression Method for Comparison of Crystallographic Textures“. In: *Philosophical Magazine A* 73.4, pp. 1083–1091. DOI: 10.1080/01418619608243705.

D. M. J. Tax and R. P. W. Duin (2004). „Support Vector Data Description“. In: *Machine Learning* 54.1, pp. 45–66. DOI: 10.1023/B:MACH.0000008084.60811.49.

J. B. Tenenbaum, V. de Silva, and J. C. Langford (2000). „A Global Geometric Framework for Nonlinear Dimensionality Reduction“. In: *Science* 290.5500, pp. 2319–2323. DOI: 10.1126/science.290.5500.2319.

C. Tomé, G. R. Canova, U. F. Kocks, N. Christodoulou, and J. J. Jonas (1984). „The Relation Between Macroscopic and Microscopic Strain Hardening in F.C.C. Polycrystals“. In: *Acta Metallurgica* 32.10, pp. 1637–1653. DOI: 10.1016/0001-6160(84)90222-0.

S. Torquato and H. W. Haslach (2002). „Random Heterogeneous Materials: Microstructure and Macroscopic Properties“. In: *Applied Mechanics Reviews* 55.4, B62–B63. DOI: 10.1007/978-1-4757-6355-3.

A. Tran and T. Wildey (2021). „Solving stochastic inverse problems for property–structure linkages using data-consistent inversion and machine learning“. In: *JOM* 73, pp. 72–89. DOI: 10.1007/s11837-020-04432-w.

M. Turk and A. Pentland (1991). „Eigenfaces for Recognition“. In: *Journal of Cognitive Neuroscience* 3.1, pp. 71–86. DOI: 10.1162/jocn.1991.3.1.71.

L. V. Utkin, V. S. Zaborovsky, A. A. Lukashin, S. G. Popov, and A. V. Podolskaja (2017). „A Siamese Autoencoder Preserving Distances for Anomaly Detection in Multi-Robot Systems“. In: *2017 International Conference on Control, Artificial Intelligence, Robotics & Optimization (ICCAIRO)*, pp. 39–44. DOI: 10.1109/ICCAIRO.2017.17.

L. N. Vaserstein (1969). „Markov Processes over Denumerable Products of Spaces, Describing Large Systems of Automata“. In: *Problems of Information Transmission* 5.3, pp. 47–52.

V. E. Vinzi, W. W. Chin, J. Henseler, and H. Wang, eds. (2010). *Handbook of Partial Least Squares: Concepts, Methods and Applications*. Vol. 1. Springer Handbooks of Computational Statistics. Springer. ISBN: 978-3-540-32825-4. DOI: 10.1007/978-3-540-32827-8.

G. Vogel and J. M. Weber (2025). „Inverse Design of Copolymers Including Stoichiometry and Chain Architecture“. In: *Chemical Science* 16.3, pp. 1161–1178. DOI: 10.1039/D4SC05900J.

T.-S. Vu, M.-Q. Ha, D.-N. Nguyen, V.-C. Nguyen, Y. Abe, T. Tran, H. Tran, H. Kino, T. Miyake, K. Tsuda, and H.-C. Dam (2023). „Towards Understanding Structure–Property Relations in Materials with Interpretable Deep Learning“. In: *npj Computational Materials* 9.1, p. 215. DOI: 10.1038/s41524-023-01163-9.

Z. Wang, W. Yan, and T. Oates (2017). „Time Series Classification from Scratch with Deep Neural Networks: A Strong Baseline“. In: *Proceedings of the 2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 1578–1585. DOI: 10.1109/IJCNN.2017.7966039.

Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, and N. de Freitas (2016). „Dueling Network Architectures for Deep Reinforcement Learning“. In: *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, pp. 1995–2003. DOI: 10.48550/arXiv.1511.06581.

L. Ward, A. Agrawal, A. Choudhary, and C. Wolverton (2018). „A general-purpose machine learning framework for predicting properties of inorganic materials“. In: *npj Computational Materials* 4, p. 36. DOI: 10.1038/s41524-018-0067-x.

C. J. C. H. Watkins and P. Dayan (1992). „Q-learning“. In: *Machine Learning* 8.3–4, pp. 279–292. DOI: 10.1007/BF00992698.

S. Weisberg (2005). *Applied Linear Regression*. 3rd. Wiley-Interscience. ISBN: 978-0-471-66379-9.

H.-R. Wenk and P. Van Houtte (2004). „Texture and anisotropy“. In: *Reports on Progress in Physics* 67.8, pp. 1367–1428. DOI: 10.1088/0034-4885/67/8/R02.

M. Werman, S. Peleg, and A. Rosenfeld (1985). „A Distance Metric for Multidimensional Histograms“. In: *Computer Vision, Graphics, and Image Processing* 29.3, pp. 272–286. DOI: 10.1016/0734-189X(85)90055-6.

T. White (2016). „Sampling Generative Networks: Notes on a Few Effective Techniques“. In: DOI: https://doi.org/10.48550/arXiv.1609.04468.

E. T. Whittaker and G. N. Watson (1927). *A Course of Modern Analysis*. 4th. Cambridge University Press. DOI: 10.1017/CBO9780511608759.

A. Wijaya, J. Wagner, B. Sartory, and R. Brunner (2024). „Analyzing microstructure relationships in porous copper using a multi-method machine learning-based approach“. In: *Communications Materials* 5.1, p. 59. DOI: 10.1038/s43246-024-00493-5.

M. D. Wilkinson, M. Dumontier, I. J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.-W. Boiten, L. B. da Silva Santos, P. E. Bourne, et al. (2016). „The FAIR Guiding Principles for scientific data management and stewardship“. In: *Scientific Data* 3.1, p. 160018. DOI: 10.1038/sdata.2016.18.

R. J. Williams and D. Zipser (1989). „A Learning Algorithm for Continually Running Fully Recurrent Neural Networks“. In: *Neural Computation* 1.2, pp. 270–280. DOI: 10.1162/neco.1989.1.2.270.

T. Xie, A. France-Lanord, Y. Wang, Y. Shao-Horn, and J. C. Grossman (2021). „Graph dynamical networks for unsupervised learning of atomic scale dynamics in materials“. In: *Nature Communications* 10, p. 2667. DOI: 10.1038/s41467-019-10663-6.

A. Yamanaka, R. Kamijyo, K. Koenuma, I. Watanabe, and T. Kuwabara (2020). „Deep Neural Network Approach to Estimate Biaxial Stress-Strain Curves of Sheet Metals“. In: *Materials & Design* 195, p. 108970. DOI: 10.1016/j.matdes.2020.108970.

J. Yan, L. Mu, L. Wang, R. Ranjan, and A. Y. Zomaya (2020). „Temporal Convolutional Networks for the Advance Prediction of ENSO“. In: *Scientific Reports* 10.1, p. 8055. DOI: 10.1038/s41598-020-65070-5.

J. Yoon, D. Jarrett, and M. van der Schaar (2019). „Time-series Generative Adversarial Networks“. In: *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*, pp. 5508–5518.

J. Zhang and A. C. Sanderson (2009). „JADE: Adaptive Differential Evolution With Optional External Archive“. In: *IEEE Transactions on Evolutionary Computation* 13.5, pp. 945–958. DOI: 10.1109/TEVC.2009.2014613.

J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros (2017). „Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks“. In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2223–2232. DOI: 10.1109/ICCV.2017.244.

X. Zhu (2005). *Semi-Supervised Learning Literature Survey*. Tech. rep. 1530. University of Wisconsin-Madison, Department of Computer Sciences.