TECHNISCHE UNIVERSITÄT
KAISERSLAUTERN

# Know What You See

# -

# *Visual Analytics enabling Machine Learning Performance Evaluation*

Vom Fachbereich Informatik der Technischen Universität Kaiserslautern zur Verleihung des akademischen Grades
**Doktor der Ingenieurwissenschaften (Dr. -Ing.)**
genehmigte Dissertation
von

**Patrick RUEDIGER - FLORE**

*Datum der Aussprache:* **28.07.2021**
*Dekan:* Prof. Dr. Jens SCHMITT
*Prüfungskommission und Gutachter:*
Prof. Dr. Hans HAGEN, TUK
Prof. Dr. Heike LEITTE, TUK
Prof. Dr. Bernd HAMANN, UC Davis, USA

D 386

# *Abstract*

This work presents a visual analytics-driven workflow for an interpretable and understandable machine learning model. The model is driven by a reverse engineering task in automotive assembly processes. The model aims to predict the assembly parameters leading to the given displacement field on the geometries surface. The derived model can work on both measurement and simulation data. The proposed approach is driven by the scientific goals from visual analytics and interpretable artificial intelligence alike. First, a concept for systematic uncertainty monitoring, an object-oriented, virtual reference scheme (OOVRS), is developed. Afterward, the prediction task is solved via a regressive machine learning model using adversarial neural networks. A profound model parameter study is conducted and assisted with an interactive visual analytics pipeline. Further, the effects of the learned variance in displacement fields are analyzed in detail. Therefore a visual analytics pipeline is developed, resulting in a sensitivity benchmarking tool. This allows the testing of various segmentation approaches to lower the machine learning input dimensions. The effects of the assembly parameters are investigated in domain space to find a suitable segmentation of the training data set's geometry. Therefore, a sensitivity matrix visualization is developed. Further, it is shown how this concept could directly compare results from various segmentation methods, e.g., topological segmentation, concerning the assembly parameters and their impact on the displacement field variance. The resulting databases are still of substantial size for complex simulations with large and high-dimensional parameter spaces. Finally, the applicability of video compression techniques towards compressing visualization image databases is studied.

# *Zusamenfassung*

Diese Arbeit präsentiert einen visual-analytics Workflow für ein interpretierbares und verständliches Modell für maschinelles Lernen (ML). Das Modell ist motiviert von einem Reverse-Engineering-Problem in der Automobilmontage. Das Ziel des Modells ist die Vorhersage von Montageparametern, welche zu dem gegebenen Verschiebungsfeld auf der Geometrie geführt haben. Das abgeleitete Modell kann dabei sowohl Mess- als auch Simulationsdaten verarbeiten. Der vorgeschlagene Ansatz orientiert sich hierbei gleichermaßen an den wissenschaftlichen Zielen der visuellen Analyse und der interpretierbaren künstlichen Intelligenz. Hierfür wird zunächst ein Konzept zur systematischen Unsicherheitsüberwachung: ein objektorientiertes, virtuelles Referenzschema (OOVRS), entwickelt. Anschließend wird die Vorhersage über ein regressives ML-Modell unter Verwendung von neuronalen Netzen gelöst. Zur Validierung wird eine Modellparameterstudie durchgeführt und mit einer interaktiven visuellen Analysepipeline versehen. Anschließend werden die Auswirkungen der gelernten Varianz in den Verschiebungsfeldern analysiert und schließlich in ein Sensitivitäts- Benchmarking-Tool überführt. Dies ermöglicht das Testen verschiedener Segmentierungsansätze, um die Eingabedimensionen für das ML zu verringern. Die Auswirkungen der Montageparameter werden im Domänenraum untersucht, um eine geeignete Segmentierung der Geometrie im Trainingsdatensatz zu bestimmen. Zur besseren Darstellung wurde hierfür eine Sensitivitätsmatrix- Visualisierung entwickelt. In der Arbeit wurde gezeigt, wie dieses Konzept Ergebnisse verschiedener Segmentierungsmethoden, z. B. der topologischen Segmentierung, bezüglich der Montageparameter und deren Einfluss auf die Verschiebungsfeldvarianz direkt verglichen werden kann. Die resultierenden Datenbanken sind immer noch von beachtlicher Größe für komplexe Simulationen mit großen und hochdimensionalen Parameterräumen. Daher wird abschließend die Anwendbarkeit von Videokompressionstechniken für die Komprimierung von Visualisierungsbilddatenbanken untersucht.

# *Acknowledgements*

I wish to thank Prof. Dr. Hans Hagen for his constant encouragement and advisory way beyond the scope of my scientific career. To Prof. Dr. Heike Leitte, who showed me that the engagement in a good education of students is worth the struggle. A special thanks to Prof. Dr. Bernd Hamann and Prof. Dr. Christoph Garth for there critical discussions and opening up the view to different approaches. I also wish thank all members from the IRTG 2057 for their support, the great discussions and talks. Finally I want to thank Dr. Alexander Scherrer, who was my first advisor in a scientific project during my studies, for introducing me into the stunning world of scientific work.

# Contents

**Abstract** **iii**

**Acknowledgements** **v**

**1 Introduction** **1**
1.1 Machine Learning . . . 3
1.2 Visual Analytics . . . 5
1.3 Interpretable and Explainable Machine Learning . . . 6

**2 Uncertainties in Assembly Processes** **9**
2.1 Digital Twin and Uncertainty . . . 10
2.2 Validation Paradoxon . . . 12
2.3 Dealing with Uncertainties . . . 14
2.3.1 Impact Analysis . . . 16
2.3.2 Object-Oriented Virtual Reference Scheme . . . 17
2.4 Discussion . . . 20

**3 Effects of distance metrics on machine learning performance** **23**
3.1 Experimental Design . . . 24
3.2 Visual Analytics Pipeline . . . 25
3.3 Pairwise - Metrics . . . 26
3.3.1 Loss Function Metrics . . . 29
Fundamental Idea . . . 29
Mathematical Description . . . 30
Example . . . 30
Combined Angular-Magnitude Metric . . . 32
3.4 Learning Regression on Sheet Metal Deformation . . . 34
3.4.1 Reference Regression Model . . . 34
3.4.2 Visual Analysis . . . 35
3.4.3 Effects on Learning Performance . . . 37
3.5 Discussion . . . 40

**4 Visual Classifier Performance Evaluation** **43**
4.1 Experimental Design . . . . . 44
4.1.1 Selected Machine Learning Methods for Classification 45
Nearest Neighbor Classifications . . . . . 47
Support Vector Classifier . . . . . 47
Gaussian Process Classification (GPC) . . . . . 47
Decision Tree and Random Forest Methods . . . . . 47
Neural Network . . . . . 48
Naive Bayes Classifiers . . . . . 49
4.2 Visual Analytics Pipeline . . . . . 49
4.2.1 Parameter Distribution Analysis . . . . . 50
4.2.2 Correlation Analysis . . . . . 51
4.2.3 Dimension Reduction . . . . . 53
Hierarchical Linear Regression (HLR) . . . . . 53
Principal Component Analyis (PCA) . . . . . 54
4.3 Classifier Performance Analysis . . . . . 54
4.3.1 Confusion Matrix and Decision Boundary . . . . . 54
4.4 Results and Implications for the regression model . . . . . 57
4.4.1 Classifier Evaluation . . . . . 58
4.4.2 Correlation Results . . . . . 60
4.4.3 PCA Results . . . . . 62
4.4.4 Overall Observations . . . . . 62
4.5 Discussion . . . . . 64

**5 Impact Maps and Topological Segmentation** **67**
5.1 Sensitivity Analysis . . . . . 68
5.1.1 Statistical Properties . . . . . 69
5.1.2 Visualization Pipeline . . . . . 69
Example Results from the Sensitivity Analysis . . . . . 69
5.1.3 Sensitivity Matrix View . . . . . 71
5.2 Gradient and Topology-based Metrics . . . . . 74
5.2.1 k-Means Distance . . . . . 75
Fundamental Idea . . . . . 76
Mathematical Description . . . . . 76
5.2.2 k-Means Clustering . . . . . 76
5.2.3 Vorticity and Strain-Rate-Tensor . . . . . 77
Fundamental Idea . . . . . 77
Mathematical Description . . . . . 77

Example . . . . . 78
5.2.4 Okubo-Weiss-Criterion . . . . . 79
Fundamental Idea . . . . . 79
Mathematical Description . . . . . 80
Example . . . . . 80
Implications . . . . . 80
5.2.5 Angular Direction Changing Rate . . . . . 81
Fundamental Idea . . . . . 81
Mathematical Description . . . . . 83
Run-time Behavior, Scalability and Complexity . . . . 84
5.3 Topology-based Segmentation . . . . . 84
5.3.1 Morse Theory . . . . . 85
5.3.2 Topological Analysis of the Sheet Metal Ensemble . . . 86
5.4 Application - Measuring Turbulence in a Magnetic Field via Local Vector Field Similarity . . . . . 88
5.4.1 Visualization . . . . . 88
5.4.2 Derived Seeding Strategy for Streamlines and Stream-surfaces . . . . . 90
5.4.3 Results and Evaluation . . . . . 90
5.4.4 Domain Expert Evaluation . . . . . 92
5.4.5 Conclusions . . . . . 92
5.5 Discussion . . . . . 93

**6 Organizing Ensembles for Visual Analytics** **97**
6.1 Video Compression for Visualization Ensembles . . . . . 99
6.1.1 Motivation . . . . . 99
6.1.2 Experimental Setup . . . . . 100
Data Sets and Visualizations . . . . . 100
Image Database Linearization . . . . . 101
Depth Image Encoding . . . . . 103
Video Compression . . . . . 104
Image Retrieval . . . . . 105
6.1.3 From Scalars to pixel formats - Morton Quantization . 106
6.1.4 Evaluation Metrics . . . . . 107
Performance Metrics . . . . . 108
6.1.5 Results . . . . . 108
6.2 Discussion . . . . . 109

**7 Conclusion and Future Directions** **113**

**A Application Scenarios** **117**
A.1 Assembly Simulation for metal sheet parts . . . . . 117
A.1.1 Generic Metal Sheet Deformation . . . . . 117
A.1.2 Automotive Engine Hood . . . . . 117
A.2 Geodynamo - Simulating the emergence of earth's magnetic field 120
A.3 Viscous Fingering . . . . . 120
A.4 Jet Flow . . . . . 120

**B List of Publications** **121**

**Bibliography** **123**

# List of Figures

2.1 Simulation Validation Paradoxon . . . . . 13
2.2 Steps towards incorporating uncertainties in a digital twin . . 15
2.3 Exemplary use of an Ishikawa Diagram for a structural effect analysis on uncertainties in a process . . . . . 16
2.4 Concept of the Object-Oriented Virtual Reference Scheme . . . 18
2.5 Exemplary model of Accuracy and Uncertainty in the OOVRS 20
2.6 Exemplary use of the OOVRS . . . . . 21

3.1 General Study Design . . . . . 25
3.2 Visual Analytics Pipeline for Regression Study . . . . . 26
3.3 Examplary handling of Differences in Observation Pairs . . . 27
3.4 Effects of Absolute and Squared Difference on Vector Rotation 28
3.5 Comparison of Absolute and Squared Difference on Vector Rotation . . . . . 29
3.6 Distribution Analysis of Difference Pairs . . . . . 31
3.7 Amplification Effect for the Combined Angular-Magnitude Metric . . . . . 33
3.8 Distribution Analysis of the Combined Angular-Magnitude Metric . . . . . 33
3.9 Hierarchical Parameter Study . . . . . 35
3.10 Comparing Regression Performance for different Loss-Metrics 36
3.11 In depth Regression Performance Comparison for different Loss-Metrics. . . . . 37
3.12 Evaluating the Effect of Training Size . . . . . 39
3.13 Training Evolution for different Hidden-Layer sizes . . . . . 40
3.14 Evaluating the Effect of used Number of Neurons . . . . . 40
3.15 Training Evolution for different Loss-Metrics and Training Fractions . . . . . 42

4.1 Special Parameter Sets . . . . . 46
4.2 Confusion Analysis . . . . . 48
4.3 Visual Analytics Pipeline for the Inverse Classification . . . . . 49

4.4 Visual Ensemble Analysis . . . . . 50
4.5 Distribution patterns using violin plots . . . . . 51
4.6 Scatter plot matrix (SPLOM) with kernel-density estimator (KDE) 52
4.7 Biplot PCA . . . . . 55
4.8 Decision Boundaries for three selected use cases . . . . . 57
4.9 Classifier Comparison using two parameters . . . . . 58
4.10 Performance results (Sensitivity, Specificity, and Accuracy) for selected special cases . . . . . 60
4.11 Classifier Comparison based on performance indicators . . . . 61
4.12 Comparing Correlations with SPLOM . . . . . 62
4.13 ROC curves and classifier performance indicator distributions for the maximum load case (ID458 . . . . . 63
4.14 Different classification for symmetric boundary conditions for real car body part (sheet metal) . . . . . 64
4.15 Analysis of multi-dimensional output of a classifier . . . . . 65

5.1 Visualization Pipeline using TTK/VTK . . . . . 70
5.2 Results from the sensitivity analysis . . . . . 71
5.3 Results from the sensitivity analysis with two-parameter variations . . . . . 72
5.4 SPLOM for Sensitivity Analysis in Domain Space . . . . . 73
5.5 Transformation Concept for a SPLOM with image in-domain visualizations . . . . . 74
5.6 Investigating different summarizing metrics . . . . . 74
5.7 k-means Distance and Clustering . . . . . 75
5.8 Assembly of the Okubo-weiss-criterion . . . . . 79
5.9 The idea underlying our turbulence measure . . . . . 81
5.10 Similarity-based turbulence shown in a plane with four equidistant neighbors . . . . . 82
5.11 Special case, where the calculated similarity value is the same 82
5.12 Results from applying the Morse-Theory to a Scalar Field . . . 86
5.13 Combined Sensitivity Analysis and Topological Segmentation. We can directly compare the input scalar field correlations with the resulting topological segmentation with the matrix view. The segmentation patterns reveal a strong positive correlation between X1 and X2. . . . . 87
5.14 The chosen transfer function for different parameter values . . 89

5.15 Streamlines of the magnetic field seeded outside the turbulent areas . . . . . 91
5.16 Streamlines of the magnetic field seeded inside the turbulent areas . . . . . 94
5.17 High turbulence Examples . . . . . 95
5.18 Dipole Visualization . . . . . 96

6.1 Overview of the image database generation, compression, and decompression pipeline underlying our study . . . . . 101
6.2 Showing different ordering strategies for one data set . . . . . 102
6.3 Pixel format and Ico-sphere . . . . . 105
6.4 Encoding of a 24-bit depth image . . . . . 107
6.5 Results for the compression performance . . . . . 109
6.6 Results for the compression performance and constant rate factors . . . . . 109
6.7 Comparison of compression rate and retrieval time clustered by codec . . . . . 110
6.8 Depth Images before and after compression with the H.264 codec 111

A.1 Generic sheet metal example . . . . . 118
A.2 Sheet metal car body part used in automotive industry . . . . 119

# List of Tables

3.1 Exemplary Model Parameters as used in TensorFlow. . . . . 34

4.1 Dimension reduction results . . . . . 54

5.1 Example Queries for the Sensitivity analysis. . . . . 70

*Dedicated to my daughter and son, who showed me that the quest for knowledge is in our every nature and to my wife, for the consistent love and reminder to remain curious, because that's what a good scientists needs to be.*

# Chapter 1

# Introduction

Interpretable and understandable intelligent systems and especially machine learning is one of today's most pressing challenges in computer science [1]. The stunning results and fast development of deep learning techniques in the last decade enabled the transfer of machine learning in various application scenarios [2]–[4]. While these models' predictive quality reached a superior state compared to humans in many applications [5], such as image classification or fraud and anomaly detection, they remain a black box for most users [6], [7]. The mostly unknown learning process is not only a security issue, where utterly new research fields like *adversarial attacks* emerged [8] but also for many applications, where unknown behavior could lead to catastrophic consequences (e.g., medicine, power supply, financial transaction). On the other hand, these intelligent models come to age, where critical doubts slowly replace the initial excitement and risk-seeking with risk aversion [9]. It leads to an increased demand for interpretability and understandability throughout all applications.

In engineering and especially in quality control, the deployment of machine learning techniques took place before the current hype on machine learning [10]–[12]. Nonetheless, the models in use were designed explicitly for a narrow application field (e.g., visual object detection in a production line, for specific single our small group of predefined object features). Consequently, such a model's development is very time-consuming and only profitable when the number of daily decisions is justifying the initial and ongoing development costs, even under changing production parameters. Today's production systems have an increased variety in products [13], which makes those highly specialized models unprofitable in many applications, and a human workforce is preferred. On the other hand, the increasing demographic aging, especially in western societies [14] continuously lower the available amount of expert work-forces required for decision making in quality control. Consequently, the effectiveness of each workforce has to be increased.

Intelligent decision support tools, which incorporate machine learning, are the first choice here, but the requirements have changed. The main goal now is to enable the workforce to develop models independently, tackling the challenges from the variability while maintaining the ability to make an increased number of decisions with high reliability. This requirement consequently demands a model development pipeline, maintaining high interpretability and easy access.
In the presented thesis, we elaborate on the challenges and analyze results to meet those requirements. As an example, we choose a parameter prediction task from an automotive assembly line. Here the goal is to reliably predict the assembly parameters from a full face optical measurement based on a given set of simulation results. In the current state, the simulation model is used to solve an inverse optimization problem, which results in high computation times and costs for every prediction. Instead, we develop a machine learning model to solve the same optimization problem based on a set of simulation results. Hence the computational effort is only initially high, and additionally, we allow for incorporating real measured data in the continuous training of the model, which was not possible before.

Uncertainty is an everlasting challenge both for measurements and simulation models alike. Therefore assessing and documenting uncertainty in an assembly process is the first building block of this thesis (see Chapter 2). After describing and quantifying the uncertainty in our training data, we develop a machine learning model capable of predicting the parameter values for a given displacement field (see Chapter 3). The displacement field is either the result from an optical measurement to the planned CAD-model or, for more controlled testing, the result of a simulation with parameters not used for training. The resulting regression model thus is moderated by the uncertainty in the training data. The model then undergoes a parameter study focusing on the effects of distance metrics on loss-functions. Further, we analyze the impact of the training fraction size (the number of simulation results for the initial training) and the number of neurons in the first (and only) hidden layer. The goal is to create a minimalistic model [15]. The resulting optimized model is still overparameterized and, therefore, barely interpretable. In summary, it proposes two major research questions:

- Why is a continuous linear function sufficient for our prediction task?

- How to reduce the input shape, while maintaining the overall entropy of the training data?

In Chapter 4, we develop an analysis pipeline to analyze each simulation parameter's impact. More specifically, we map each parameter's impact on the resulting distances (the input for the ML-model prediction) between a target displacement field and the training data displacement fields. It results in a visual analysis tool, which allows the detection of dominant parameters and their pair-wise cross-correlations. Additionally, we derive a benchmark for checking the increase or decrease in entropy for segmentation of the input shape.
In Chapter 5, the investigated part's domain space is analyzed with regards to different simulation parameter combinations using the concept of sensitivity analysis. Based on these results, a sensitivity matrix view is derived. Together with the topology analysis results on the resulting deviation fields, we can explore possible segmentations on the machine learning model's input shape. Further, we show the applicability of topology-based metrics for the use in more turbulent vector fields, like the geodynamo.
Finally, we investigate video compression techniques to cover the challenges for storing and accessing large image databases resulting from in-situ visualizations (Chapter 6).

## 1.1 Machine Learning

The research field of machine learning (**ML**) is a broad topic and gained increasing importance with the availability of big data and cloud computing resource. In this thesis, we will focus on the branch of supervised learning. Machine Learning is now classified as a part of Artificial Intelligence (**AI**). Before that, machine learning was mostly considered a part of statistics and there, especially in inferential statistics. Thus it is not surprising that the first descriptions of machine learning approaches are already found in the late '50s due to the demand for more flexible statistical prediction methods.
First, there is no clear definition, and it changes with the advancements of technology. We choose to use the definition from Tom Mitchell, which is part of his famous textbook on machine learning, first released in 1997 :

> *"Machine learning is the study of computer algorithms that allow computer programs to automatically improve through experience."*[16]

Here the "learning through experience" part is what best separates machine learning from other AI methodologies. Now we can distinguish three ways to achieve this "learning through experience," which classifies the machine learning field into the three major paradigms: **supervised**, **unsupervised** and **reinforced**.
In **supervised** learning, which we also mainly consider here, the learning objective is known, and we call it "labeled." In contrast, in **unsupervised** learning, the object is unknown. Thus it is not clear at the beginning what exactly is learned. One could ask the question now: *" What is the purpose of learning if the objective is unknown?"*. The bottleneck of supervised learning gets clearer if we look at the challenges that big data proposes to data analytics. The amount of possible features (objectives) and their cross-dependencies is barely manageable. Supervised learning techniques demand a clear definition of input parameters and output features. In many big data applications, the definition of a clear objective is not possible or simply unknown. Here unsupervised learning can help find similar clusters in a data set, where traditional "exact" methods fail due to their computational complexity. However, beware that this "unknown" analysis proposes several paradoxes. The Simpson Paradoxon [17] is a classic example of false accusation, where the hypothesis is formulated after the data analysis resulting in a huge interpretation bias. Hence applied to a result from unsupervised learning, we tend to over-interpret the found similarities if we did not start our analysis with a profound hypothesis derivation.
Now there is also a third way to classify the learning objective. If the objective is known, we assumed that the objective could describe it via a label, a fixed value to say. If we want to learn an optimization task, on the other hand, the label is not set but described by minimizing or maximizing property. The paradigm of **reinforced** learning is used to solve these kinds of learning problems. Here the input is a parameterized function, and the output again is a function that evaluates the result from the input function, resulting in a score for the optimization problem. The model is now learning how to adapt and find the optimal parameters of the input function. The Braess Paradoxon [18], is very important to understand the challenges in reinforcement learning. In summary, it proposes the challenge of forcing the model to change its optimization strategy while it did not reach an equilibrium state. Typically, there is no incentive to change the strategy as long as the optimization function gradient decreases (for a minimization problem). Breass's example of network congestion showed that a change of strategy is

ultimately needed to solve the congestion.
Regarding Visual Analytics and the approaches developed here, the Accuracy Paradoxon [19], [20] is the challenge we focus to overcome in Chapter 4. The accuracy metric is often over-trusted in many machine learning applications. Precision (also denoted as Sensitivity) and Recall are much better suited for classifier performance evaluation.

## 1.2 Visual Analytics

The research field of **Visual Analytics** is the consequence of the ongoing advances in information and scientific visualization. It extends those fields by adding active reasoning as a fundamental part. The advances in both visualization fields, especially those that enabled near real-time interaction and modifications, enabled the field of Visual Analytics. A prominent definition of Visual Analytics is the one by J.J. Thomas:

> *"Visual analytics is the science of analytical reasoning facilitated by interactive visual interfaces."*[21]

The definition already states that Visual Analytics and Analytical Reasoning are inextricably linked with each other. As such, we can understand Visual Analytics as an enabler for Analytical Reasoning. The challenges that Visual Analytics has to overcome can be summarized into the fields of *Visual Representations and Interaction Technologies*, *Data Representations and Transformations* and *Production, Presentation, and Dissemination* as introduced by Thomas [21]. As a consequence, Thomas formulates the goals on which we develop the visual representations and interaction approaches. Those are:

- allow for continually growing collections of data of multiple types - (see Chapter 6 - Evaluating the use of Video Compression for Visualization Products);
- provide frameworks for spatial data - (see Chapter 5 - Impact Maps and Sensitivity Analysis );
- support the understanding of uncertain, incomplete, and often misleading information - (see Chapter 2)
- provide user- and task-adaptable guided representations that enable full situation awareness while supporting development of detailed actions - (see Chapter 4 Figure 4.15)

- support multiple levels of data and information abstraction, including integrating different types of information into a single representation. - (see Chapter 4 Figure 4.2)

Nevertheless, the task of visual analytics does not stop at the **visual** representation. Further, we have to consider the initial challenges proposed by the **data** representation and transformation:

- develop both theory and practice for transforming data into new scalable representations that faithfully represent the underlying data's relevant content. - ( see Chapter 4 - Experimental Design)
- create methods to synthesize different types of information from different sources into a unified data representation so users can focus on the data's meaning in the context of other relevant data, regardless of the data type. - (see Chapter 4 - Visual Analytics Pipeline)
- develop methods and principles for representing data quality, reliability, and certainty measures throughout the data transformation and analysis process. - (see Chapter 2- Object-Oriented Virtual Reference Scheme, Chapter 4 - Classifier Performance Analysis or Chapter 3 - Effects of Distance Metrics on Learning Performance)

## 1.3 Interpretable and Explainable Machine Learning

The research field of interpretable machine learning has grown at a fast pace over the last years. Regarding the recent publications in the IEEE Journal - Transactions on Computer Graphics and Visualizations and the Proceedings of the Visualization Conference in the years from 2015 - 2020, we observe a tremendous increase in the field of interpretable machine learning. Each year the number of related publications is nearly doubled in the past five years. Qin et al. [22] and Hohman et al. [23] provide a profound overview of the upcoming challenges and recent developments in visual analytics for explainable machine learning. Hani Hagras [24] formulated the needs and goals towards **human**-understandable artificial intelligence which also includes machine learning. As a consequence, Hagras formulated five focus areas: **Transparency - Causality - Bias - Fairness - Safety**.
Further Gunning [25] identified the four main building blocks explainable Artificial Intelligence has to address. These are:

- Model Performance
- Deep Explanation
- Interpretable Models
- Model Induction

The goal for *Model Performance* is to explain the relationship of changing performance under controlled conditions. This explanation can be achieved by a structured model parameter study like the one conducted in Chapter 3. As the name already suggests, *Deep Explanation* is a major concern in Deep Learning approaches. The goal here is to learn explainable features. *Interpretable Models* aim to keep them as simple as possible while monitoring their performance change. Thereby directly follows the concept of *Model Induction*, which directly focuses on general approaches, allowing the model deduction process recreation. Especially the last approach is what motivated this work the most. As such, our derived machine learning model is based on the concept of Yanez-Marquez[15] minimalist machine learning. For reverse engineering applications like the one we are investigating here, we think this approach will result in the best trade-off between performance and explainability. The very structured nature of engineering applications enables such a minimalistic approach, and the thesis shows this.

# Chapter 2

# Uncertainties in Assembly Processes

Uncertainty is a persistent challenge throughout all disciplines in science, and engineering [26]. In natural science, the complexity of our real-world defines uncertainty. We cannot fully control and measure all parameters and always have to deal with some uncertainty. Ideally, we want to describe only a subsystem of the world and, therefore, accept uncertainties when formulating a generalizing hypothesis. In engineering, we usually have more control over the whole process.
Nonetheless, the same mechanism introduces uncertainty: not fully controlling all parameters. Let us take a look at the light conditions in a factory hall. Even if we mainly use artificial light, some uncontrollable processes affect the lighting. Such a condition could be open doorways or windows from which daylight emits into the factory. The intensity and frequency of daylight are uncontrollable parameters [27].
A more common and dominating process parameter that introduces the most uncertainties in manufacturing and especially assembly is the human itself. At every point where a human being is involved in the process, we introduce uncertainty to it. Daily and hourly variations affect even a perfectly trained human. If more than one person is involved, the intermediate variations between them are inducing even more uncertainty. The variational effects hold both for manual and cognitive tasks.
Simulations, on the other hand, are always deterministic in their outcome based on the chosen parameters. Nonetheless, it makes sense to incorporate uncertainties in the simulation to validate and analyze the real-world, and simulated processes [28]. The benefit of simulations is clear nowadays [29], but it is not always clear why something should be simulated [30]. In the concept of a digital twin, incorporating these uncertainties plays a crucial role.
Regarding the overall research question, the uncertainty modeling method

strongly affects the learning model. It makes a massive difference if it is systematically defined or implicitly given in the training data. Only in the first case we can incorporate the uncertainty as an explicit additional parameter of the learned model. While in the latter case, we can only learn to distinguish it.

**Related Work** Zouaoui et.al. [31],[32] and Barton et. al. [33] are talking in detail on how to deal with uncertainties in the input model. While Zou et al. [34] and McKay et al. [35] are investigating the problem of analyzing and evaluating uncertainty. Du and Chen [36], and Hills [37] provide models on how to manage uncertainty in the process, and therefore also provide useful ideas on how to analyze uncertainty. A very advanced approach for its time was proposed by Su and Lee [38], where they analyzed the uncertainty of a process by controlled manipulation of uncertainty factors. A good resource and starting point for visualizing uncertainties is made by Potter et al. [39].

## 2.1 Digital Twin and Uncertainty

Tao et al. describe the general concept of a modern digital twin [40]. The main goal of these is an accurate copy (Twin) of a real-world process inside given boundaries with predefined target outcome data. Simulations are a vital component of most digital twins. The predefined target outcome is often not accurately specified. This misunderstanding elevates simulations as a replacement of a physical process, which leads to low acceptance of simulation development and results throughout the company. In engineering, this mainly occurs in the phases of quality and machine control [41]–[43].
Instead, communicating simulations to analyze processes with the necessary flexibility for upcoming changes is preferable. This flexibility is what demands the incorporation of uncertainty. Generalization introduces flexibility, and significance is the limiting factor. The significance ensures that the simulation matches achievable results in the real-world process. Those two factors have to be balanced out, which leads to the formulation of the following optimization problem for the development of simulations as digital twins:

$$Generalization := \{situations,\ that\ can\ be\ simulated\} \tag{2.1}$$

$$max|Generalization|,\ with\ significance > s_0 \tag{2.2}$$

Generalization consists of a set of simulatable situations. Maximizing the number of elements in this set is interpreted as maximizing the generalization of the simulation. On the other hand, the significance is hard to formalize, and so is setting a realistic threshold of $s_0$. As stated before, the significance ensures the realism of the simulation. In order to ensure realism, uncertainties have to be taken into account as well [44] .
Additionally, the simulation has to match a predefined precision to be useful at all. The somewhat fuzzy "significance" can be replaced with a combination of incorporated uncertainty and achieved precision. For an uncertain process, uncertainty moderates the achievable precision. As such, we can formalize the precision as a function of uncertainty. It figures out that using uncertainty instead of certainty suits better in most cases and better copes with the meaning for error in statistics, which is an excellent analogous for uncertainty in simulation validation with real process data. The optimization problem then changes to:

$$max|Generalization|,\ with\ precision(uncertainty) > p_0 \tag{2.3}$$

Now that the optimization problem is stated, the question arises how to check if the conditions hold. First, it may be useful to have a look at the definitions of precision and uncertainty from the 2007 International vocabulary of metrology [45].

**Definition 1 Measurement uncertainty** is a non-negative parameter characterizing the dispersion of the quantity values being attributed to a measurand, based on the information used
" *NOTE 1 Measurement uncertainty includes components arising from systematic effects, such as components associated with corrections and the assigned quantity values of measurement standards, as well as the definitional uncertainty. Sometimes estimated systematic effects are not corrected for but, instead, associated measurement uncertainty components are incorporated.*
*NOTE 2 The parameter may be, for example, a standard deviation called standard measurement uncertainty (or a specified multiple of it), or the half-width of an interval, having a stated coverage probability.*
*NOTE 3 Measurement uncertainty comprises, in general, many components. Some of these may be evaluated by Type A evaluation of measurement uncertainty from the statistical distribution of the quantity values from series of measurements and can be characterized by standard deviations. The other components, which may be*

*evaluated by Type B evaluation of measurement uncertainty, can also be characterized by standard deviations, evaluated from probability density functions based on experience or other information.*
*NOTE 4 In general, for a given set of information, it is understood that the measurement uncertainty is associated with a stated quantity value attributed to the measurand. A modification of this value results in a modification of the associated uncertainty.*"

**Definition 2 Measurement precision** is the closeness of agreement between indications or measured quantity values obtained by replicate measurements on the same or similar objects under specified conditions.
" *NOTE 1 Measurement precision is usually expressed numerically by measures of imprecision, such as standard deviation, variance, or coefficient of variation under the specified conditions of measurement.*
*NOTE 2 The 'specified conditions' can be, for example, repeatability conditions of measurement, intermediate precision conditions of measurement, or reproducibility conditions of measurement (see ISO 5725- 3:1994).*
*NOTE 3 Measurement precision is used to define measurement repeatability, intermediate measurement precision, and measurement reproducibility.*
*NOTE 4 Sometimes "measurement precision" is erroneously used to mean measurement accuracy.*" [45].

Summarizing, we can state that the amount of information (entropy) and how well the process is understood defines *uncertainty*. At the same time, *precision* is a quantifiable property that arises from repetition and systematic testing.

## 2.2 Validation Paradoxon

The validation of a digital twin simulation proposes several challenges and is often misguided due to the underlying validation process's dimensionality. Based on the previous observations, we can already define two parameters for the validation process: Generalization and precision. We already figured out that uncertainty moderates the precision of the real-world process itself. Additionally, we can always extend the validation with an economic viewpoint. The amount of effort (time, money, other resources) is limited for the validation and adds the third dimension to trade-off in the simulation validation. In real applications, we have to balance our effort, generalization, and

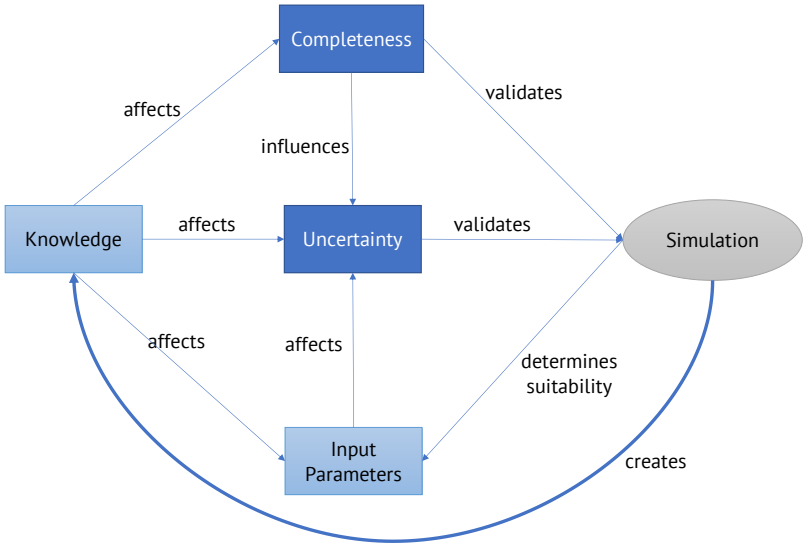



FIGURE 2.1: Simulation Validation Paradoxon. As the development of a simulation is a continuous process, it is hard to determine when a simulation is validated. With every newly generated simulation, new knowledge of the modeled process is gained. This new knowledge then affects the way we define completeness, uncertainty, and input parameters. Those three factors, on the other hand, are the key components the validation is built on. Therefore the validation of a simulation always depends on the current state of the development process.

precision when developing and validating the simulation. The real challenge now is not the three-dimensional optimization problem but the interference of such simulations' development process. Figure 2.1 illustrates this problem, which leads to the proposed Paradoxon.
The development of a simulation typically follows a looped cycle. We create an initial simulation, and from the analysis of the results, we create new knowledge about the real-world modeled process. The process's knowledge highly moderates the accuracy by describing the uncertainties in the process and how accurate the generalization parameters are modeled or extended. Both then positively affects the achievable precision as well as the current precision of the simulation itself.
The simultaneous change of the achievable and the current simulation precision leads to an ambiguous optimization problem. This process leads to a paradox where the simulation's target precision could be better and worse than the achievable precision of the real process simultaneously due to not incorporated uncertainties. These uncertainties depend on whether the newly generated knowledge reveals new uncertainties or better quantifies known ones in the real-world process. Suppose we model the simulation development process as a state-machine. Then, the current state defines the validation, and we can solve the ambiguity. The only real limiting factor, finally, is the available effort.

## 2.3 Dealing with Uncertainties

We have seen that uncertainties are a crucial part of the current strive for digitization in manufacturing [46]. Especially in assembly processes, much manual work is still used and therefore induces high amounts of uncertainties. The add-up tolerances and errors of the previous processing steps induce a high uncertainty in the final assembly stages. Quality Control (QC) is a key instrument to deal with these uncertainties. For now, it was sufficient for QC to maintain and prove predefined quality criteria without the need to specify them thoroughly. We showed that virtual models such as digital twins heavily rely on a profound specification of the occurring uncertainties throughout the whole process. Without them, the virtual model's precision will barely approach the demanded precision needed for a digital twin. The idea of a digital twin in quality control is to supplement high effort, or high-cost tasks [40]. High effort tasks typically involve a complex processing pipeline to be tested or where the measurement of quality criteria is only

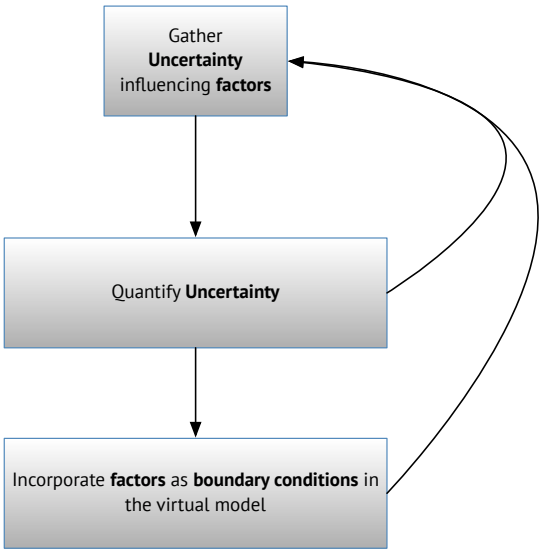



FIGURE 2.2: Steps towards incorporating uncertainties in a digital twin. The process follows a looped cycle. It starts with gathering all relevant factors and is followed by quantification of them. The quantification step could require additional factors, which results in a back-loop. The last step is the incorporation of the quantified factors as boundary conditions in the virtual model. In the virtual models' validation phase, new uncertainty factors could be identified and incorporated into a new cycle.

possible under tremendous efforts. We want to avoid high-cost tasks where the part is on purpose or potentially destroyed and is unusable afterward. In order to tackle the uncertainties for the usage in digital twins, we identified three steps:

1. Gather all uncertainty influencing factors in the process
2. Quantify the uncertainty of all factors
3. Incorporate the factors as boundary conditions in the virtual models

Remark that the process steps and their connection and dependencies to external sources are defining the influencing factors. To cover these dependencies already in the early stages, we use the Ishikawa method [42], [47] to gather and order the influencing factors. We introduce an object-oriented approach to store a digital reference scheme, which describes the current stage uncertainty of the virtual model.

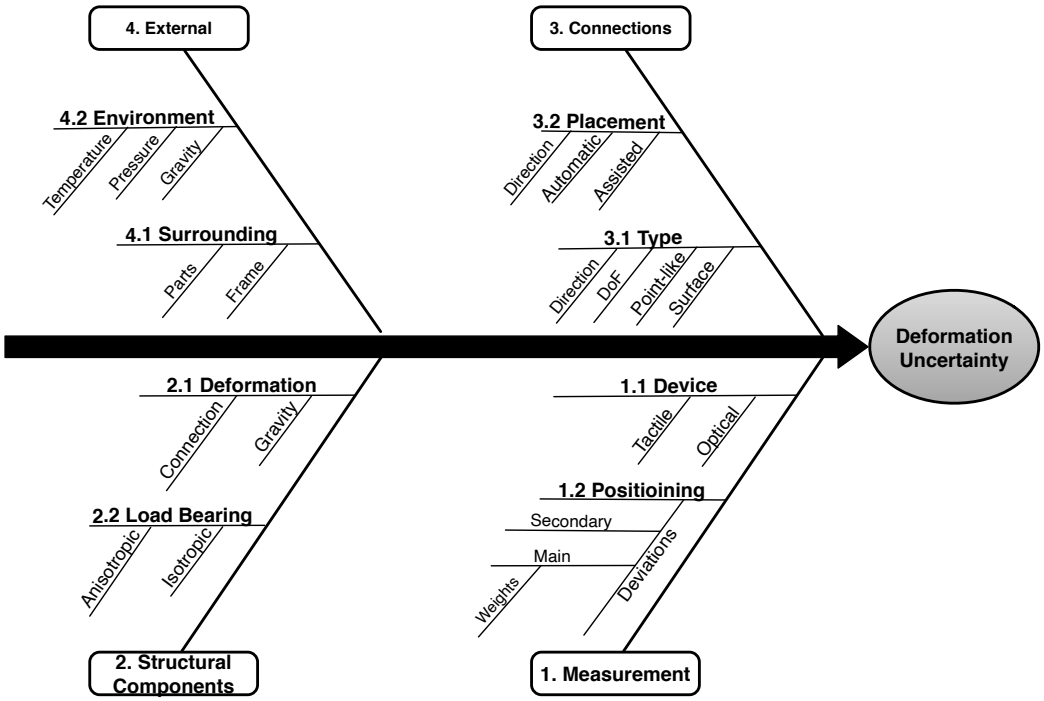



FIGURE 2.3: Exemplary use of an Ishikawa Diagram for a structural effect analysis on uncertainties in a process. We identified four main effect groups - Measurement, Structural Components, Connections, and External factors. Each of those is then subdivided into more detailed factors. E.g., the structural component consists of deformation and load-bearing effects. The structural components themselves bear deformation uncertainties, which could moderate the overall uncertainty of the assembled part. Different material properties lead to varying load-bearing behavior (especially directivity properties), affecting the deformation uncertainty.

### 2.3.1 Impact Analysis

Ishikawa diagrams, also called fishbone diagrams, herringbone diagrams, cause-and-effect diagrams, or Fishikawa, show cause and effect relations. The use of Ishikawa diagrams is widespread in product design and quality control (see VDI 2870 [48]). It is used in quality control to identify potential effects or influencing factors that might have an overall effect. Each of these identified factors is a source of variation in the underlying process. The factors are then typically grouped into major categories. In Figure 2.3 we show an example of an Ishikawa diagram used for the application scenario of the assembly of a car engine hood (A.1.2).

### 2.3.2 Object-Oriented Virtual Reference Scheme

To cover the complexity of digital twins' validation process, we developed an object-oriented approach for referencing dependencies on uncertainty and possible impacts on subsequent process steps. The object-oriented virtual reference scheme (OOVRS) is a tool to determine a digital process chain's reliability. The idea is to allow the definition of uncertainty budgets similar to tolerance chains [49] in the product development phase. The object-oriented method allows us to model hierarchies and dependencies with specific and abstract methods to calculate the uncertainty budget.
The idea is to make it possible to have specific methods modeling the dependency between two or more subsequent process steps and allow for general methods clustering steps that belong to a similar group of processes and therefore share standard features.
Following this abstraction the development of such a scheme should follow the **SOLID** principles [50]:

- Single-Responsibility Principle:
  "There should never be more than one reason for a class to change." [50]
- Open-Closed Principle:
  "Modules should be both open (for extension) and closed (for modification)." [50]
- Liskov Substitution Principle:
  "Let $q(x)$ be a property provable about objects $x$ of type $T$. Then $q(y)$ should be true for objects $y$ of type $S$ where $S$ is a subtype of $T$." [50]
- Interface Segregation Principle:
  "Clients should not be forced to depend upon interfaces that they do not use." [50]
- Dependency Inversion Principle:
  "A. High-level modules should not depend on low level modules. Both should depend on abstractions.
  B. Abstractions should not depend upon details. Details should depend upon abstractions." [50]

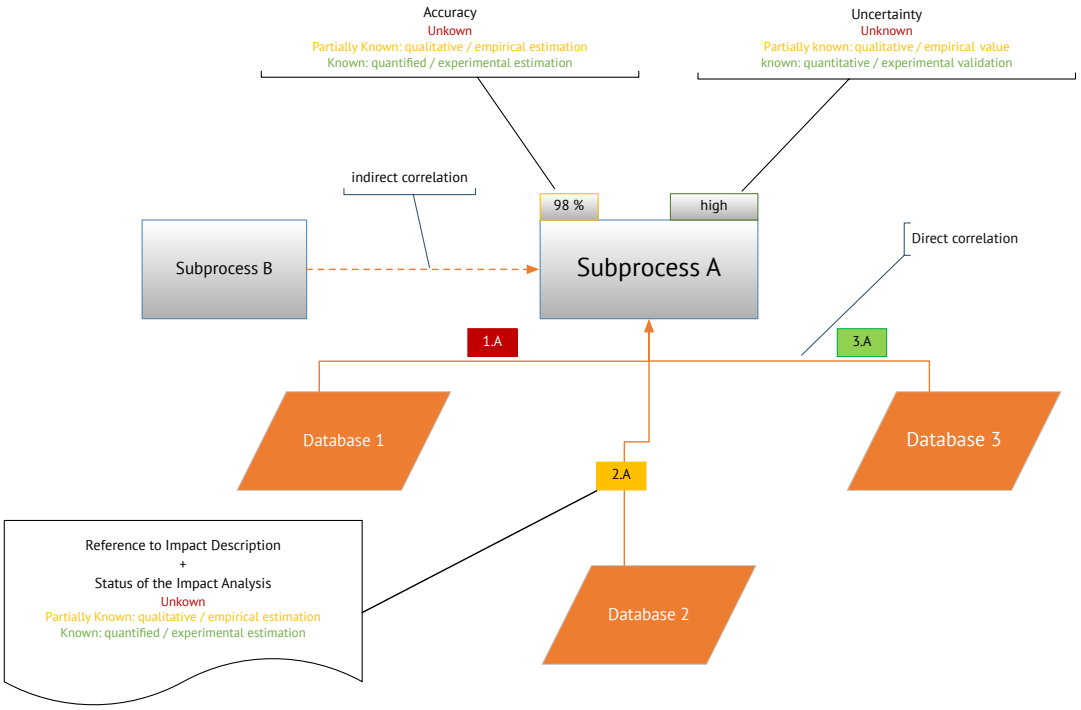



FIGURE 2.4: Concept of the Object-Oriented Virtual Reference Scheme. It consists of Processes (silver rectangle), Dependencies (arrows), and Data Sources (orange rhombus). The processes are extended with accuracy and uncertainty descriptions, while the dependencies bear a status annotation. Besides these restrictions, one is free to model the details of each object. The idea is to generate a quick overview of the relationships of accuracy and uncertainty throughout the whole process.

The idea of OOVRS is one of an inverse reference. The typical process would be as follows: We achieve an optimal result (physical or virtual) via calibrating all parameters until reaching a near-optimal state. However, this approach has a considerable disadvantage when the optimal result is not feasible or unknown, or the process itself is too complex (e.g., too many parameters) to solve. Instead, based on the uncertainties in the process, the optimal achievable result is derived from the system's current state. As such, improving the uncertainty is conditional for the improvement of the overall result.
For the implementation of such a scheme, we have to fulfill three constraints:

- A database for all process-relevant Data
- A formalized process definition that allows for mathematical operations (e.g., uncertainty budget)
- Definition of suitable references for the smallest process entities (physical and virtual).

The virtual reference scheme is an ongoing process itself, while new data and knowledge continuously enrich it.

Figure 2.4 is showing an abstract example of an OOVRS. Each class in the scheme belongs to one of three main types: **Process**, **Dependency** and **Data Source**.

- **Process** is a technologically, temporally, and locally determined interaction.
- **dependency** describes the conditional relationship between two processes or a process and a data source.
- **Data source** is a deterministic source of information throughout a process.

The selected Process A depends directly on three data sources Data1, Data2, and Data3. It indirectly depends on Sub-process B, which means that the uncertainty or accuracy of Process A is influenced by Sub-process B under certain constraints, which are defined in Process A specifically, which follows the "Dependency Inversion Principle." This principle means that Process A knows its dependency on Sub-process B and Data1, Data2, and Data3, but not vice versa. Each dependency is either a concrete entity or derived from the general dependency class, which holds an attribute on the status. The status models the knowledge of the dependency in three stages:

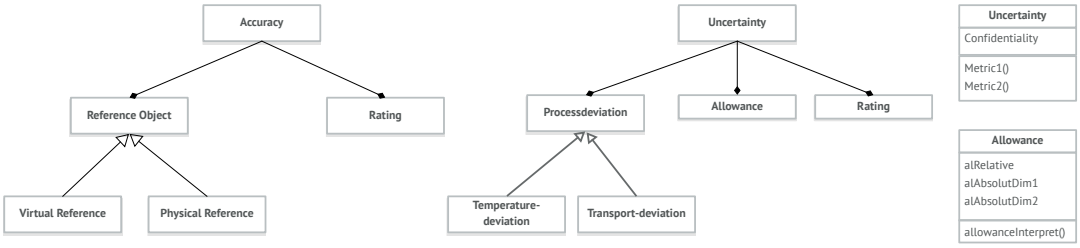



FIGURE 2.5: Exemplary model of Accuracy and Uncertainty in the OOVRS. The main difference between those two is that accuracy is based on a reference object, while uncertainty evaluates a process deviation. Mandatory for both is the rating object for the calculation of error chains. The modeling in UML-Style allows for direct software-assisted implementation.

- **known**: There exist a formal (mathematical) definition and method implementation.
- **partially known**: There exists a method, but it is not validated or applicable in all scenarios. (e.g., only for a certain accuracy range)
- **unspecific**: There is a dependency, but its mechanism is not known yet.

These stages indicate whether the dependency is useful for formal analysis like, e.g., uncertainty budgeting or not. Two main indicators then define each process:

1. Accuracy: It describes the mapping accuracy in the real-world process through the virtual process based on reference.
2. Uncertainty: It describes the uncertainty of the entropy with whom
   - the reference was developed
   - the method providing the accuracy of such reference

One is free to choose an ordinal or rational scale for accuracy or uncertainty, based on the application.

## 2.4 Discussion

We have demonstrated the influence of uncertainties in manufacturing processes and highlighted its importance for developing digital twins. Regarding our general research question, we could state:
Uncertainty is a property that can be learned by a machine learning model

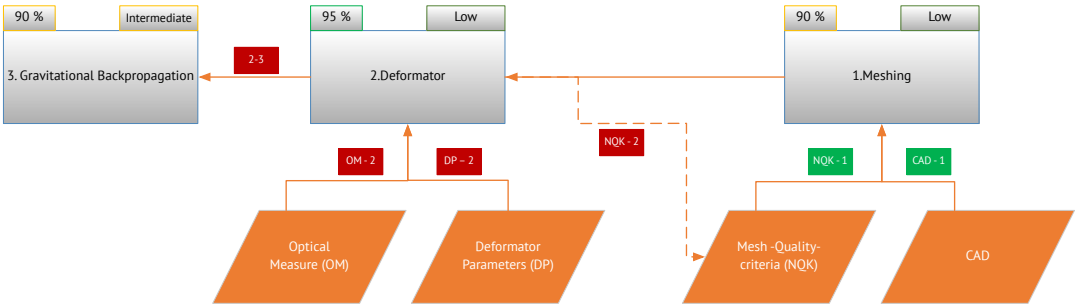



FIGURE 2.6: Exemplary use of the OOVRS. The OOVRS highlights a portion from the pre-processing step of application A.1.2. The gravitational back-propagation simulation depends on two previous processing steps: Deformator and Meshing. The meshing process is the starting point, where the initial simulation mesh is created and is mainly dependent on the two data sources, CAD, and Mesh-Quality criteria. Both sources are well understood and controlled. As such, the meshing uncertainty is low, and the accuracy is known and mainly quantified. There still exist some qualitative empirical values. As such, it is marked yellow. The deformation algorithm, which maps a measured displacement onto the simulation mesh, depends on the two data sources: Optical Measurement and Deformator Parameters. Both data sources' impact on the Deformator is not sufficiently studied yet(red label). Nonetheless, the overall outcome of the Deformator remains certain and accurate (green labels for both accuracy and uncertainty). The impact of the deformation result on the gravitational back-propagation simulation is currently unknown. As such, accuracy and uncertainty are only empirical estimates.

if the digital twin used for training incorporates it. Incorporating this uncertainty is shown to be a challenging multidimensional task. As such, the separation of uncertainty as a learned feature from the other ones is challenging too. Additionally, in such a system, the model can only learn those uncertainties well understood and adequately modeled. To support this task, we evaluated the use of Ishikawa's method [47] to determine the influencing factors and the concept of an Object-oriented Virtual Reference Scheme (OOVRS) for modeling those uncertainties across multiple process steps.
Suppose we want to incorporate uncertainty in a more general way into the model. In that case, we can augment the training data with real measurements, which inherently cover all uncertainties present in the process. With this approach, it is still impossible to determine what kind of uncertainty is learned, but it is easier separable from the other learned features. Additionally, the uncertainty is learned with the right amount of influence on the overall process if we carefully balance artificial and real measured data.
If we only want the model to learn linear elastic deformations accurately and incorporate real measurements, then the opposite effect occurs. We want to avoid training uncertainties, as these are process-relevant parameters, but not describing the correct physical behavior. Depending on the impact of those uncertainties, they can adumbrate the underlying physics, leading to a well-trained model for this process but poor performance when transferring it to another. As we can only validate the impact after the training and transfer to another process, it is crucial to analyze the uncertainties of the currently used process for training beforehand.

# Chapter 3

# Effects of distance metrics on machine learning performance

Distance metrics are the critical aspect for proper labeling of both test and training data in supervised learning. Comparing two deformation fields is a non-trivial task, and for different applications, different metrics exist. In general, we can distinguish between feature-based, value-based, and topology-based metrics. The value-based metrics use the actual values of the vector field without considering the topology. A prominent representative of this group is the mean-squared error. Feature-based metrics rely on comparing similar features in each set and, therefore, require an additional feature extraction step beforehand. Topology-based metrics put the actual values of the vector field in perspective to their neighborhood properties. These metrics typically rely on kernels or geometrical properties and are therefore very costly but lead to the most detailed results. This chapter investigates the value-based metrics' effect on the regression model's needed complexity for the given applications on the linear elastic deformation of a metal sheet.
The focus of this investigation is to understand and visually analyze the impact of each metric. We start with the basic differenced based metrics Mean-Absolute-Error (**MAE**) and Mean-Squared-Error (**MSE**). As a vector describes the displacement field, we also investigate the Cosine Similarity (**COS_SIM**), which uses the dot product as its main source of information. We then introduce a combined metric that uses the difference (length of the vector) and the dot product (angular change). We conducted a parameter study using a neural network regression with varying loss function metrics, training data sizes, and neurons to show practical relevance. A visual analytics approach assists the whole process of engineering an optimal model. We designed a specific test set, which allows us to summarize the regression's prediction results in distinct classes rather than a floating value range.

**Related Work** From the perspective of the end-user, machine learning is still mostly labeled as a black-box. Throughout the years, visual analytics is approaching this topic, intending to color this black box. In the work of Duch [51], he made some early attempts that highlight the four dominant effects in learning models, such as neural networks. These are the dynamics of neural learning, under and over-fitting effects, regularization effects, and differences between networks with the same performance. Tzeng et al. [52] extended this work intending to improve such networks' engineering. With the increased usage of deep neural networks (DNNs) in the last years, the black box coloring on the level of single neurons is no longer a feasible approach due to the immense number of neurons used here. As a consequence, the visual analytics approach switched more to the output side. Samek et.al. [53] for example introduced a direct visualization approach, which highlighted the learned features of images using the three techniques of a sensitivity analysis [54], Deconvolution Method [55] and the LRP Algorithm [56]. Inspired by the focus on the output performance, Smilkov et al. [57] introduced a direct-manipulation visualization approach, which communicates the method of neural networks via an interactive playground rather than coding. Focusing on the output, Zintgraf et al. [58] present the prediction difference analysis method for visualizing the response of a neural network. While this work is focused mainly on image classification, it contains the base idea of our visual analytics pipeline for the regression task. Finally, for the combination and comparison of multiple models and input data, Chatzimparmpas et al. [59] introduced StackGenVis.

## 3.1 Experimental Design

Our experimental approach systematically investigates the effects of the chosen metrics on the metal sheet example's learning performance. We choose a general model, where we train an adversarial neural net with all the simulation output cells. We investigate the metrics for the usage as loss functions. Here, we only choose the neural network as the learning model, as this method is the only one that can sufficiently handle such high dimensionality in the input shape. As distance metrics, we decided for the pairwise-metrics: the mean-squared error (MSE), the mean absolute error (MAE), the cosine similarity, and a custom combination of the MSE and cosine similarity the RMSECosine, which are applied either on the magnitude of the displacement vector or the vector itself. We focus on the root mean squared error

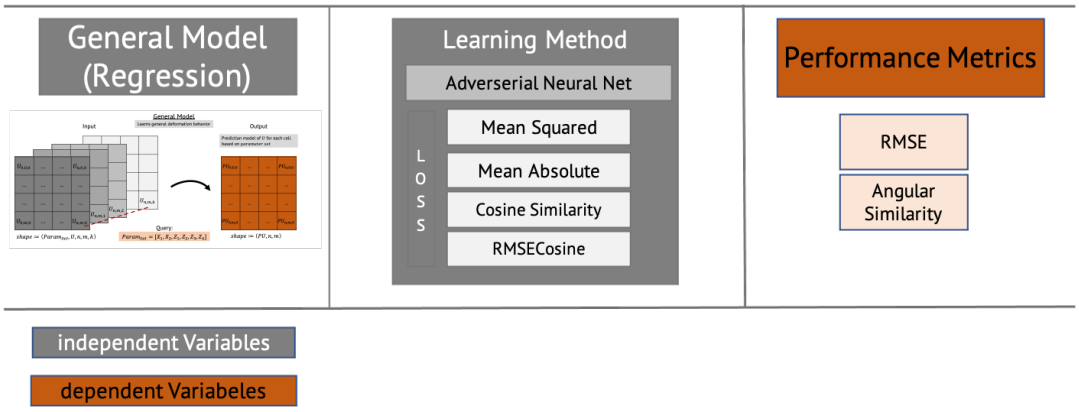



FIGURE 3.1: General Study Design. A general regression model using a fully connected one-layer neural network is chosen. For the model, different loss functions are evaluated with the use of two performance metrics. Additionally, the training fraction size and size of the first layer are varied.

(RMSE) and the angular similarity to capture the directional error when using the vector as an input for the learned model's performance metrics.

## 3.2 Visual Analytics Pipeline

For this study's visual analytics pipeline, a four-step approach is chosen (see Figure 3.2). At first, we investigate each metrics pairwise component's distribution and general behavior (Figure 3.6). We split each metric's sum into its component. We plotted them for each cell when choosing a target and reference data set (Figure 3.6, left) and the distribution of the summed value and all reference data sets (Figure 3.6, right). It gives us an overview of the expected performance of each metric. If there is a near equal distribution of the error for all reference data sets (training data), the metric poorly describes the differences. In the detailed view, we can observe the difference in how the metrics treat the same cells.
Second, we investigate the loss evolution throughout the training epochs (Figure 3.15, left) or directly compare the minimal error (Figure 3.15, right) for different training parameters. This visualization helps us to decide whether we need more training epochs or if we have to adjust the learning rate or activation if the loss evolution is oscillating.
Third, we evaluate the prediction performance across all parameters by using box-plots. Here we can use various grouping strategies based on the desired comparison (Figure 3.10). It allows for a direct comparison of the prediction


FIGURE 3.2: Visual Analytics Pipeline for Regression Study. This study's visual analytics pipeline consists of the four components: Loss Metric Distribution - Loss Evolution, while training - Prediction Performance, across the parameters and within the parameters. The pipeline is designed to meet the requirements on visual analytics described in Chapter 1.

performance in the value range of the parameters itself.
We further split the comparison for each parameter into three value categories (min, zero, max). It allows us to compare the prediction quality within the parameters' value range (Figure 3.11).

## 3.3 Pairwise - Metrics

Pairwise metrics are the most prominent metrics throughout data analytics applications. Due to their simplistic nature and fast computation, they are usually the first choice when comparing two numerical values. In contrast to topological metrics, the only information processed in these metrics are the paired-values, which allows the application on nearly every data set.
There are two general operations used for comparing a value-pair:

1. Difference: $\Delta_- : x - y$
2. Product: $\Delta_* x \cdot y$

The further processing is based on either one of these $\Delta$'s. The $\Delta_-$ gets encapsulated by another operation based on the chosen metrics. Typical choices are:

1. Absolute: $|x - y|$

2. Squared: $(x - y)^2$

If $x$ and $y$ are scalar values, we can show the effects via simple line plots, as depicted in Figure 3.3. We have two paired observations, $y1$, and $y2$. The signed difference is visualized using arrows to point in the direction of the difference when calculating $y2 - y1$. In the plots below, we depict the absolute and squared errors, which are no longer signed values. The absolute error represents the length of the arrow precisely, while the squared error amplifies larger ones. The root-mean-squared error (RMSE) and the mean absolute error (MAE) are only equal if the variance is $0.0$. Still, the RMSE is not directly linked to the variance but the variance of the frequency distribution of error magnitudes. It makes the RMSE harder to interpret on certain occasions.

If $X$ and $Y$ are vector values, the error metrics' effects are harder to inter-

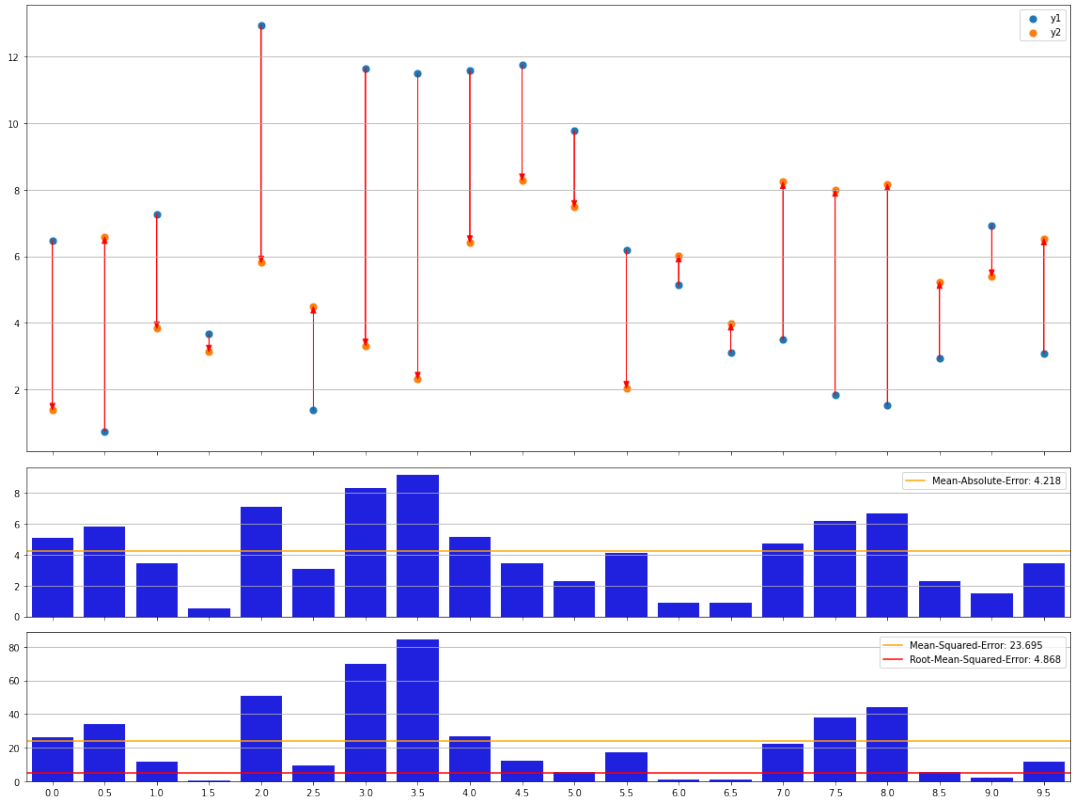



FIGURE 3.3: Examplary handling of Differences in Observation Pairs. In the top row, an example of paired observations is given. Additionally, a vector is added to indicate the direction of the change from one observation to another. In the bottom rows, the effects between using the absolute and squared errors are depicted. Based on the desired effect, one can amplify larger errors or handle all sizes equally.

pret. In Figure 3.4 we have depicted two vector-valued observations $\overrightarrow{A}$ and

$\overrightarrow{B}$. The difference $\Delta_-$ is then just the vector $\overrightarrow{\Delta_{MAE}}$ connecting the two endpoints of the vectors. So if we calculate the absolute difference, we compare the vector $\overrightarrow{\Delta_{MAE}}$ for each observation. If we now encapsulate the differences with the operations already used before in the scalar example, we get the vectors $\overrightarrow{\Delta_{MAE}}$ and $\overrightarrow{\Delta_{RMSE}}$. Now, the vectors' length gets amplified when using the squared difference, but the vector's direction changes according to each component's difference. In our given example the difference in $x$-direction is $2$ and in $y$-direction $3$, which results in a slightly more towards the $y$-axis tilted vector $\overrightarrow{\Delta_{MAE}}$. If we now use the squared error, we amplify this effect. In general, the squared operations amplify the rotations towards the dominating axis-component. In most cases, this is a desirable effect, as we want to weigh our metric based on the components, but it is tough to interpret how much of a rotation is incorporated in the resulting value. In our example we had a ratio of 3:2 for our two components which led to a rotation angle (measured from $\overrightarrow{A}$) of $56.9°$ for $\overrightarrow{\Delta_{MAE}}$ and of $66.0°$ for $\overrightarrow{\Delta_{RMSE}}$ (ratio= 0.86).

In Figure 3.5, a short overview is given. We can observe three exceptional

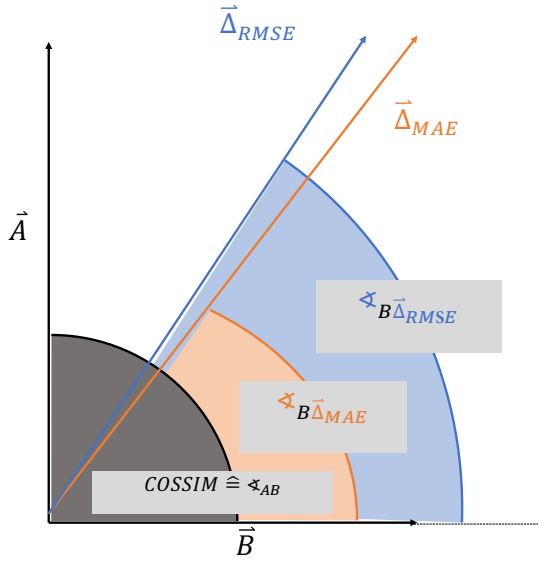



FIGURE 3.4: Effects of Absolute and Squared Difference on Vector Rotation. Schematic view on the effects on the resulting difference vector when using absolute or squared value pairs. While the length remains the same, the rotation angle is changing.

cases here. First, a pure rotation without a change in length (marked in orange) leads to a constant difference for both the absolute and the squared error. On the other hand, a constant angle but with varying length of the vectors leads to a change in the resulting $\Delta$-vectors (yellow marks in Figure 3.5 ). The third exception here is that a small angle ($18.43°$) together with a slightly

increased difference ($(1,2)$ vs. $(1,3)$) leads to greater deviations between the absolute and squared error than for the contrary example (blue squares in Figure 3.5). These are only some examples showing how difficult the interpretation of the metrics gets when applied to vectors. A prevalent metric for comparing vector-valued metrics is the **Cosine Similarity**, which uses the angle between the paired vectors. The multiplication in the dot product used to calculate the angle is an example of the alternative approach of just using the difference. Nonetheless, we can see in Figure 3.5 in the column Angle(A, B), that this metric does not take changes in the length of the vector into account. The optimal setting is a combination of both approaches, which we will investigate for the usage as a loss function in the next section.

| ID | A | | B | | Angle(A,B) (in degree) | DV | | D2V | | Angle(A,DV) (in degree) | Angle(A,D2V) (in degree) | Angle Ratio |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | x | y | x | y | | x | y | x | y | | | |
| 1 | 1 | 0 | 0 | 2 | 90 | 1 | 2 | 1 | 4 | 63.43 | 75.96 | 0.84 |
| 2 | 2 | 0 | 0 | 4 | 90 | 2 | 4 | 4 | 16 | 63.43 | 75.96 | 0.84 |
| 3 | 3 | 0 | 0 | 6 | 90 | 3 | 6 | 9 | 36 | 63.43 | 75.96 | 0.84 |
| 4 | 4 | 0 | 0 | 8 | 90 | 4 | 8 | 16 | 64 | 63.43 | 75.96 | 0.84 |
| 5 | 5 | 0 | 0 | 10 | 90 | 5 | 10 | 25 | 100 | 63.43 | 75.96 | 0.84 |
| 6 | 1 | 0 | 1 | 1 | 45 | 0 | 1 | 0 | 1 | 90.00 | 90.00 | 1.00 |
| 7 | 2 | 0 | 2 | 1 | 26.57 | 0 | 1 | 0 | 1 | 90.00 | 90.00 | 1.00 |
| 8 | 1 | 1 | 2 | 4 | 18.43 | 1 | 3 | 1 | 9 | 26.57 | 38.66 | 0.69 |
| 9 | 3 | 1 | 4 | 0 | 18.43 | 1 | 1 | 1 | 1 | 26.57 | 26.57 | 1.00 |
| 10 | 4 | 1 | 4 | 14 | 60.02 | 0 | 13 | 0 | 169 | 75.96 | 75.96 | 1.00 |
| | | | | | **COSSIM** | **MAE** | | **RMSE** | | | | |
| | | | | | 61.85 | 1.7 | 4.9 | 2.39 | 6.33 | | | |

FIGURE 3.5: Comparison of Absolute and Squared Difference on Vector Rotation. We defined ten exemplary use cases to show the challenges while interpreting the different metrics' results on vectors. The column Angle(A,B) uses the dot product, while DV refers to the absolute difference and D2V to the squared difference. The Angle Ratio is the ratio between Angle(A,DV) and Angle(A,D2V).

### 3.3.1 Loss Function Metrics

#### Fundamental Idea

The **loss function** is a fundamental part in neural networks. It calculates the prediction error of a neural net, and thus it is used for steering the network. The steering is done via gradients decent of the loss-function used to update the neurons' weights - the fundamental neural network training mechanism. To say what all neural networks have in common is an optimization method

that minimizes the prediction error via a gradient descent approach to find the network's optimal weights. The loss function is these optimizations input.

### Mathematical Description

$x_i$ and $y_i$ are scalar valued paired observations, while $X_i$ and $Y_i$ correspond to vector valued observation pairs. We observe the three commonly used metrics: Mean-Absolute-Error (**MAE**), Mean-Squared-Error (**MSE,RSME**) and the Cosine Similarity (**COSSIM**).

$$MAE = \frac{1}{n}\sum_{i=0}^{n}|x_i - y_i| \tag{3.1}$$

$$MSE = \frac{1}{n}\sum_{i=0}^{n}(x_i - y_i)^2 \tag{3.2}$$

$$RMSE = \sqrt{MSE} \tag{3.3}$$

$$COSSIM = \frac{1}{n}\sum_{i=0}^{n}\frac{X_i \cdot Y_i}{||X_i||||Y_i||} \tag{3.4}$$

### Example

Lets have a closer look on the behavior of the metrics itself. First we observe the input for the metric. The fundamental portion of information is the comparison of $x_k$ and $y_{k,i}$ for each cell $k$ over all ensemble members $i$, denoted as $\Delta(x_k, y_{k,i})$ . Where $x_k$ is the target value for the cell $k$ and $y_{k,i}$ is the prediction $i$ of cell $k$. For the MAE and MSE we use the magnitude of the vector, for the COSSIM we use the displacement vector itself (denoted with $X_k, Y_{k,i}$).

$$\Delta_{MAE}(x_k, y_{k,i}) = |x_k - y_{k,i}| \tag{3.5}$$

$$\Delta_{MSE}(x_k, y_{k,i}) = (x_k - y_{k,i})^2 \tag{3.6}$$

$$\Delta_{COSSIM}(X_k, Y_{k,i}) = \frac{X_k * Y_{k,i}}{||X_k||||Y_{k,i}||} \tag{3.7}$$

$$\overline{\Delta_{COSSIM}} = 1 - \frac{arccos(\Delta_{COSSIM})}{\pi} \tag{3.8}$$

We can now plot this information for all cells, which results in Figure 3.6. In the left column, we plot the metrics corresponding $\Delta$ and the result of the metric for a specified target-prediction pair. In the right column, we plot the

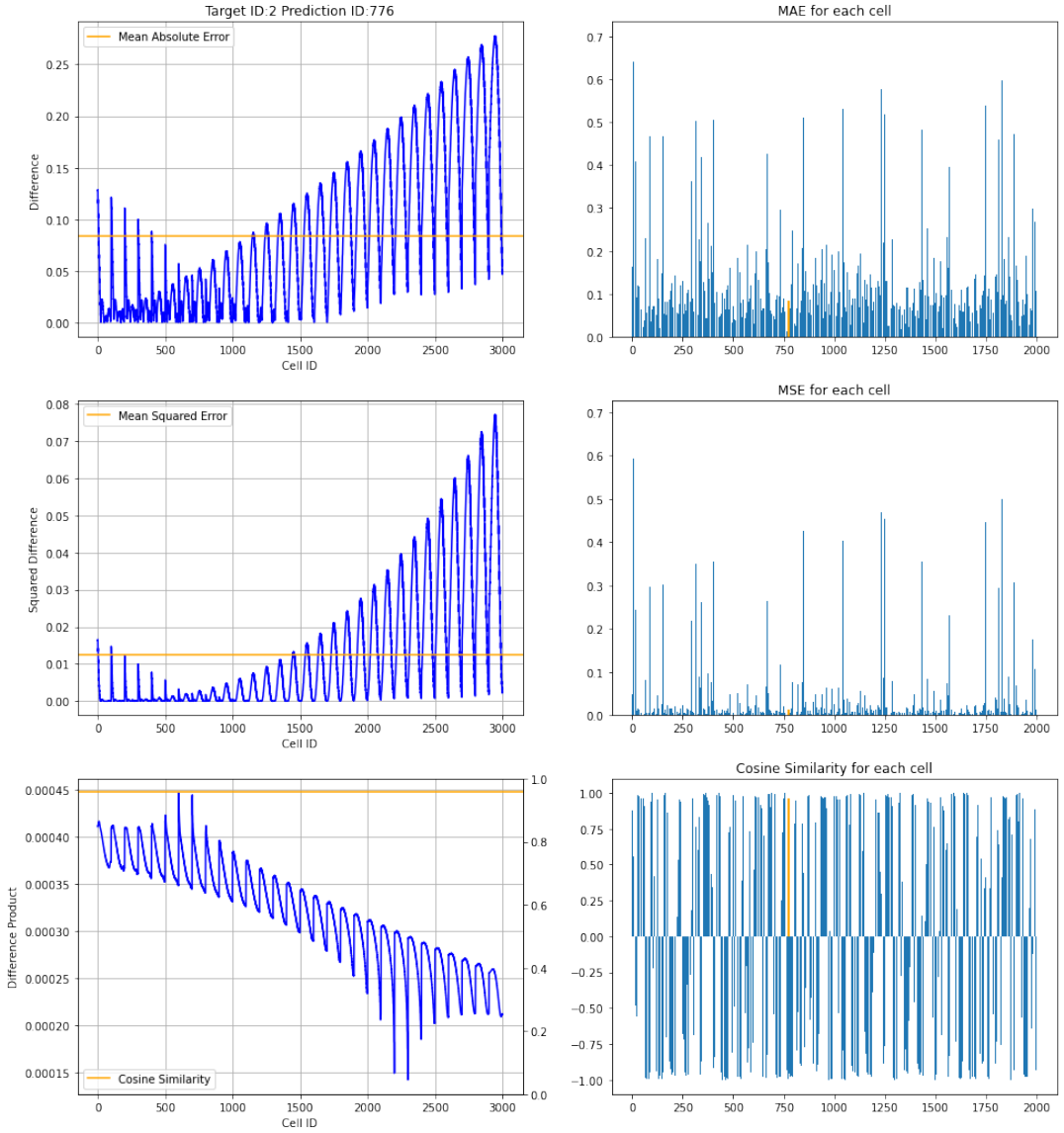



FIGURE 3.6: Distribution Analysis of Difference Pairs. In the left column, the metrics corresponding $\Delta$ and the result of the metric for a specified target-prediction pair is shown. In the right column, the metrics result of all ensemble members $i$ for the chosen target is shown. We can observe that the MSE is much more sensitive to outliers than the MAE, which results in a steeper gradient of the plotted values.

metrics result of all ensemble members $i$ for the chosen target. We can observe that the **Mean-Squared-Error (MSE)** is much more sensitive to outliers than the **Mean-Absolute-Error (MAE)**, which results in a steeper gradient of the plotted values. On the other hand, the **Mean-Absolute-Error (MAE)** applies less smoothing, which can be observed in the left part of the plots, where the MAE is more oscillating then the MSE. The same pattern can be seen for all ensemble members $k$ in the right column. Here we see that the

MSE amplifies extreme values and dampens lower differences.
The **Cosine-Similarity (COSSIM)** takes a different approach. Instead of subtracting the two values, they are multiplied as part of the dot-product operation. In contrast to the MAE and MSE, higher values refer to higher similarity and a better score. As such, we can observe that it amplifies the lower values and dampens the higher values, which is somewhat the mirrored case of the MSE. Compared to the MSE, this is a linear amplification instead of an exponential one, which leads to less smoothing. In the given example the cosine similarity is $0.95$ which leads to an intermediate angular difference of $18.19°$ ($cos^{-1}(COSSIM)$). Given a rather smooth displacement field, this is already a huge difference. Negative values indicate intermediate angular differences over $90°$. For a better comparison we normalize the COSSIM to the range $[0, 1]$ using the equation 3.8.
On the other hand, the combined view of MAE, MSE, and COSSIM could detect rotations. As such, a low MAE and MSE with a low $\overline{COSSIM}$, could be an indicator for a rotation, rather than a large displacement. On the other hand, a high $\overline{COSSIM}$ and high MAE/MSE value indicates a large removal but in the same direction. It leads us to the definition of a combined metric.

### Combined Angular-Magnitude Metric

We can now combine the metrics MSE on the magnitude with the COSSIM on the vector itself to incorporate rotation into the loss function. To keep the lengths up to scale, the RMSE is used.

$$\Delta(RMSE \circ \overline{COSSIM}) = RMSE \cdot (1 + e^{-2 \cdot \overline{COSSIM}} - \overline{COSSIM} e^{-2}) \quad (3.9)$$

Additionally, we introduce an optional weighting factor $\mu$, which helps to steer the angular dissimilarity penalty. A higher $\mu$ value corresponds to a higher penalty for larger angels and a lower penalty for smaller angels.

$$\Delta_\mu(RMSE \circ \overline{COSSIM}) = RMSE \cdot (1 + e^{-\mu \cdot \overline{COSSIM}} - \overline{COSSIM} e^{-\mu}), \mu \geq 2 \quad (3.10)$$

Figure 3.7 shows the amplification effect with different $\mu$. In general, this metric introduces an exponential amplification penalty with a higher angular difference. Analyzing the distribution we now achieve a similar behavior than for the RMSE, but with interpretable incorporation of rotational changes.

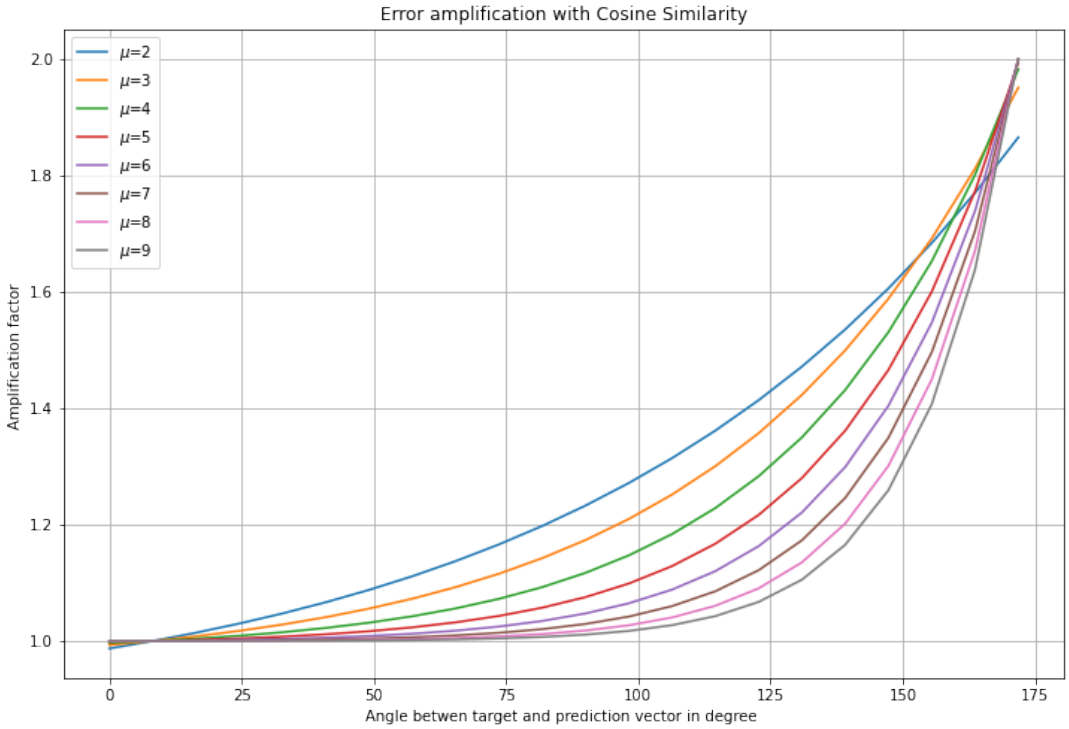



FIGURE 3.7: Amplification Effect for the Combined Angular-Magnitude Metric. With variying $\mu$ we can steer the exponential behavior from nearly linear ($\mu = 2$) to highly exponential. This way we can steer the penalty for larger angular deviations in the metric.

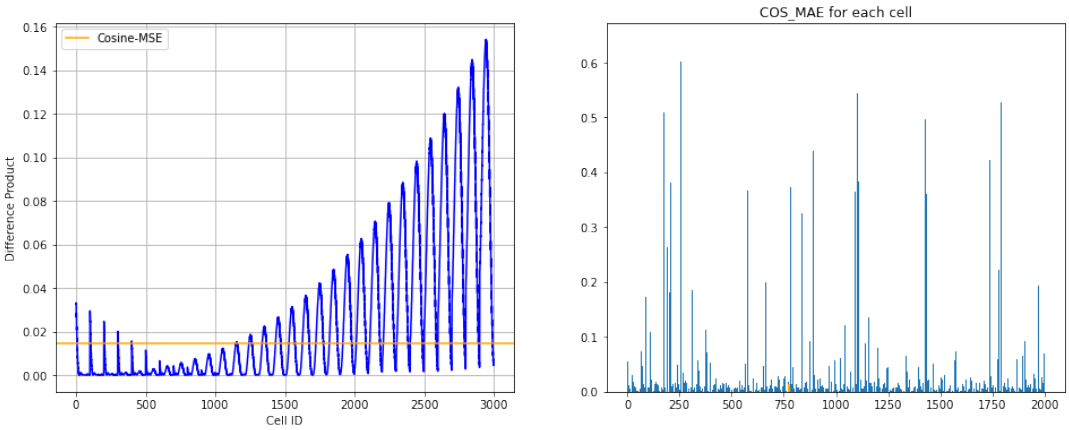



FIGURE 3.8: Distribution Analysis of the Combined Angular-Magnitude Metric. On the left, the metrics corresponding $\Delta$ (Equation 3.9) and the result of the metric for a specified target-prediction pair is shown. On the right, the metrics result of all ensemble members $i$ for the chosen target is shown.

| Model: "MAE_MAG" | | |
|---|---|---|
| Layer (type) | Output Shape | Param # |
| flatten (Flatten) | (None, 3000) | 0 |
| dense (Dense) | (None, 3000) | 9003000 |
| dense_1 (Dense) | (None, 6) | 18006 |
| Total params: 9,021,006 | | |
| Trainable params: 9,021,006 | | |
| Non-trainable params: 0 | | |

TABLE 3.1: Exemplary Model Parameters as used in TensorFlow.

## 3.4 Learning Regression on Sheet Metal Deformation

We will now investigate the effect of different loss metrics in a simple regression model, predicting the parameter set based on a displacement field. We start our initial training with 2000 simulations based on an equally sampled parameter set. We then vary the loss metrics. For the MAE and MSE, we investigate the difference between using the vector's magnitude and the complete vector itself. Afterwards we are decreasing the training size from 90% to 80%, 70%, 60%, and 50% respectively. We use one fully connected (dense) layer with the initial phase's data set's shape. Afterward, we decrease the number of neurons in the series of $[\frac{n}{2}, \frac{n}{4}, \frac{n}{8}, \frac{n}{16}, ...]$, with $n$ referring to the initial number of cells present in the data set. An overview of the experimental setup is given in Figure 3.9. We then introduce an interactive way to visually analyze the different models and test them with specific test data sets. We conclude this section with a general thought on the effects of metrics on this application's learning performance, which we will investigate further in the next chapter.

### 3.4.1 Reference Regression Model

We choose a fundamental model for our investigation, with only one fully connected (dense) layer and the output layer with the parameter set's dimension. With two layers, we are still in the class of continuous functions and can keep the method comparable to the others, as proven by Brutzkus et al. [60]. In general, a two-layer network can learn polynomial functions of degree $r$ over $d$-dimensional inputs [61].

In Table 3.1 a summary of such a model is given.

### 3.4.2 Visual Analysis

The general approach in evaluating the regression performance is by investigating the loss and error function during training as depicted in Figure 3.15. The information in this visualization is minimal and only gives us a rough idea of the model's correct tendency. Therefore we will mainly use this visualization for testing the initial setup and check if the loss function is reaching a saturation point or if we need to train more epochs or need to adapt the restart method to find additional local minima.
For our evaluation, we choose a hierarchical approach to minimize the combinatorial complexity. Therefore we are ordering our independent variables and choose only the best setting to proceed with in the next step. In this approach we end up with the experimental design depicted in Figure 3.9.
We start by evaluating the difference between MAE and MSE applied either

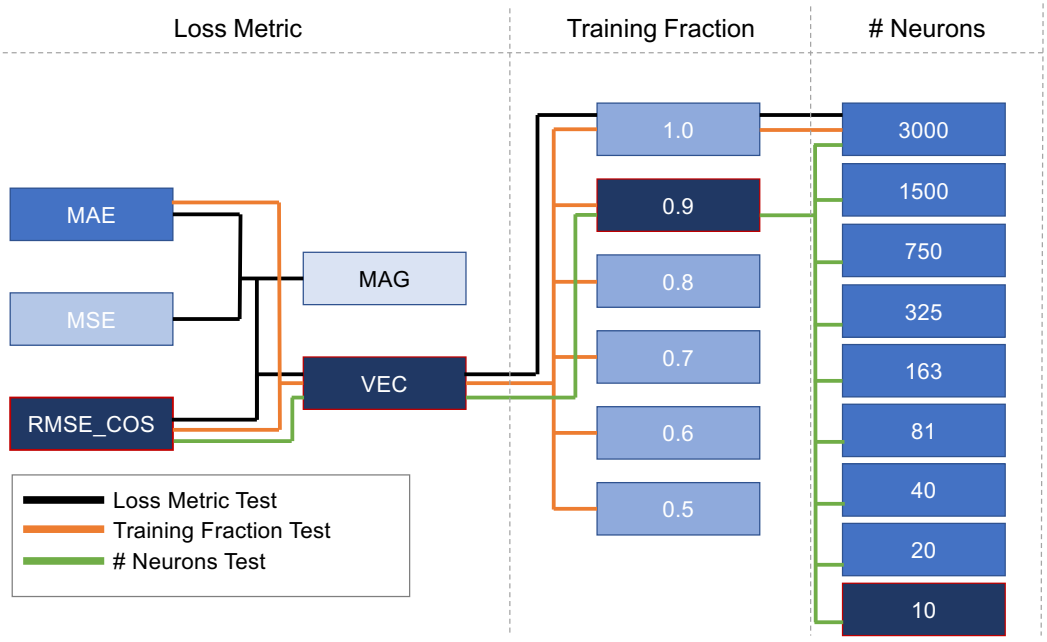



FIGURE 3.9: Hierarchical Parameter Study. First, we vary the loss metrics while keeping the training fraction and neurons fixed - Loss Metric Test. After finding the best-suited metrics, we proceed with the training fraction test. Finally, we vary the number of neurons while keeping the rest fix in an optimal state.

on the displacement vector's magnitude or on the vector itself, thus keeping the initial dimensions. For this investigation, we use the special sampling of our test data set. The parameter values examined can only assume the values $-0.1, 0.0$ or $0.1$, respectively. All combinations of those values for the six parameters are tested, which results in $6! = 720$ possibilities covering a wide

range of possible correlating effects. For the visual analysis, this allows us to focus on three groups of predicted values and enables the coupled visual analysis of each parameter, as shown in Figure 3.11. We split our analysis in a general view (see Figure 3.10), in which we observe the resulting absolute prediction error for each parameter and a detailed grid view for each parameter and each value. We developed an interactive approach, which allows us to compare multiple models directly with each other.

For the decision on whether to use the magnitude or each vector's component, we can observe that the prediction error is nearly $10$ times smaller for the vector approach. Putting it into context, when using the magnitude, inside the first quartiles, the error range ($[-0.084, 0.092]$) is about 88% of the whole value range used for training and testing. While using the vector, we can reduce the error to the range of $[-0.010, 0.003]$, which is about 6.5% of the total value range.

Using the cosine similarity on his own leads to prediction errors outside of

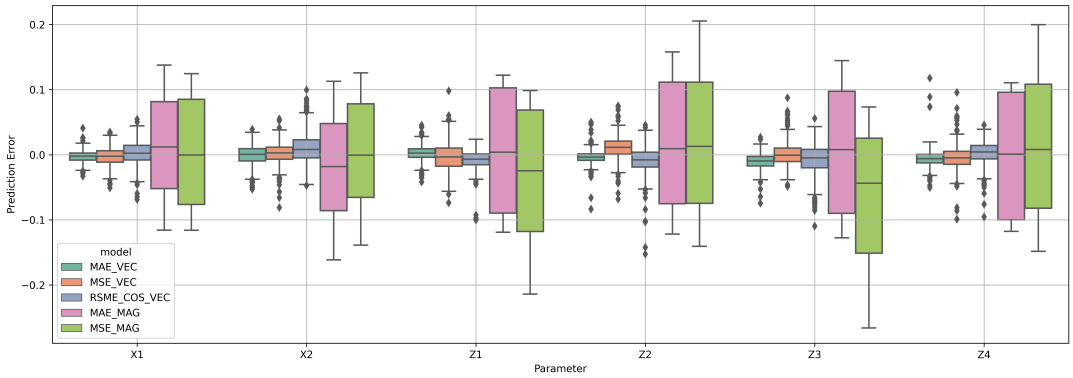



FIGURE 3.10: Comparing Regression Performance for different Loss-Metrics. We now observe the resulting prediction error for each predicted parameter, grouped by the used loss metric. Using the vector's magnitude instead of applying the model to each component leads to worse results.

the value range. As such, focusing purely on the directivity is not a feasible approach here. Nonetheless, it is surprising that the combined angular and value-based metric did not lead to overall better results. It seems that this loss metric is too sensitive in this setup. For the parameters X1, the MAE is the best performing, while for X2, the MSE and MAE are nearly equal. For Z1, the combined metric RMSE_COS is the best performing. For Z2, Z3, and Z4, the MAE is again leading. Overall we can observe that the difference between the three metrics is minimal. We will now investigate if this is true for all parameter values by looking at the detailed view in Figure 3.11.

In the detailed view, we can observe each parameter's differences in de-

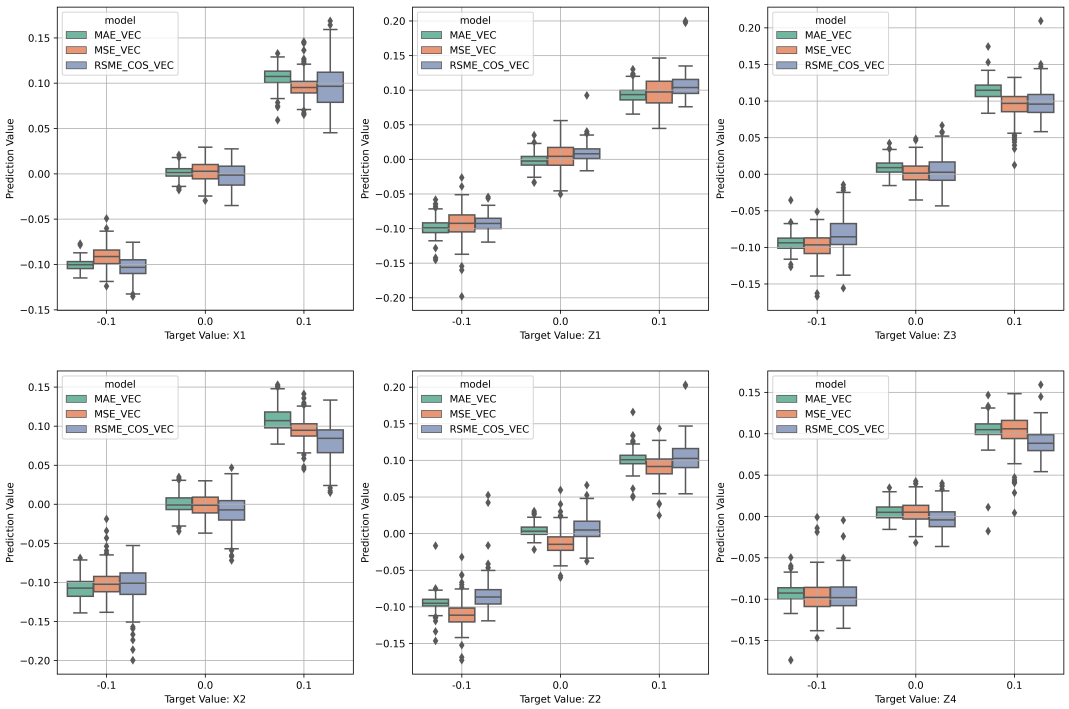



FIGURE 3.11: In depth Regression Performance Comparison for different Loss-Metrics. In this setup, we used a specially chosen test data set, which naturally results in the three groups of value ranges shown here. It allows for an in-depth prediction error analysis for each parameter and three main value ranges of the parameter.

tail using the three states of the parameter range: min, zero, and max (-0.1,0.0,0.1). Now looking at Z2, we can observe that the MSE-based prediction is tending towards a lower value. For now, it looks like the MSE metric is performing the worst out of the three vector-valued metrics. In our next analysis step, we will only focus on the vector-valued loss metrics (MAE-VEC, MSE-VEC, and RSME-COS-VEC). Here we investigate if the same holds if we are decreasing the training samples or reducing the neural networks weighting parameters.

### 3.4.3 Effects on Learning Performance

In this section, we want to show how to analyze the effects of different learning parameters visually. We are reusing the scheme from above with the overview and detail. This time, our grouping variable is either the training fraction or the number of neurons in the first layer.

At first, we look at the learning evolution depicted in Figure 3.15. We plotted the error evolution through 10 epochs of training for each of the investigated training fractions on the left. All of our investigated metrics are converging to a minimum after ten epochs. From these plots, we can derive that for the MSE metric, some more epochs or an adaption of the restart method is needed to end up in the global minimum. Comparing the three metrics, we can observe that the MAE metric is stable among the training fractions, while the MSE metric is starting to oscillate between local minima. On the right, we plotted the evolution of the trained minimal error. We can observe an exponential increase of the error with linear decreasing training fraction throughout all metrics. In general, our application's result here is that taking less than 1800 (90% of the initial training data) samples for training will result in a significant error increase. As such, the previously used 2000 samples were already a good choice. Typically one would choose the training fraction right before the exponential increase of the error as the optimal training sample size. For the MSE metric, we can observe the same oscillating behavior for the learning performance throughout the epochs. For the combined custom metric, we designed the RMSE_COS. We observe a less oscillating behavior throughout the epochs, resulting in a more stable learning process. Additionally, the error evolution throughout the training fractions is nearly stable until 60% of the training fraction. This result indicates our hypothesis that is incorporating both the angular and the absolute change into the loss function will result in a higher entropy, allowing us to use only 1200 samples to achieve the same results as with 1800 samples.

We will now compare the MAE and RMSE_COS metrics performance for each predicted parameter using the previous section's scheme. The results here for are depicted in Figure 3.12. For both of the examined metrics, we can observe significant differences in the prediction error parameters. In our high- level analysis from Figure 3.12, we would conclude that MAE is still the better performing metric. Now comparing each parameter's performance on our test set, we can observe that the RMSE_COSINE metric is performing better through all training fraction sizes. Especially for the fraction sizes 1.0 to 0.8, the prediction error variance is always lower than using the MAE metric. Additionally, we can observe that the MAE metric's median prediction error (indicated by the horizontal line in each box) is barely near 0.0, and the variance is shifted in either a positive or negative direction. There seems to be no explainable pattern for a directivity here. In our next evaluation step, we will only investigate the RMSE_COSINE metric for the training fraction

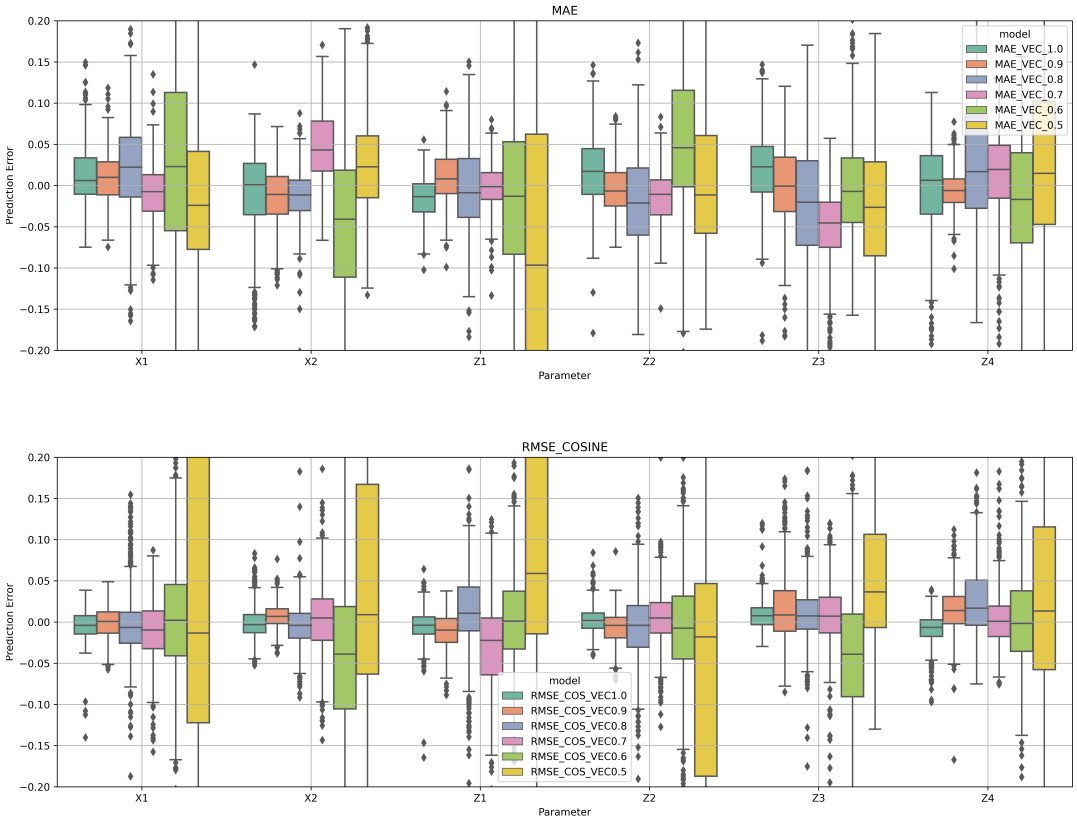



FIGURE 3.12: Evaluating the Effect of Training Size. For two loss metrics (top and bottom), the prediction error for each parameter (x-axis) and training fraction size (color coding) is shown. We can observe that the RMSE_COS has less variance for larger fraction sizes while the MAE median is offset.

of 0.9.

In our last test setting, we evaluate the impact of the number of neurons in the first hidden layer on the prediction error. First, we investigate the training behavior while decreasing the number of neurons, as depicted in Figure 3.13. We observe that more epochs are needed to achieve a stable state with a decreasing number of neurons. On the other hand, we can conclude that we can achieve the same overall prediction error or even better, even with fewer neurons. It means that our initial setup tends to overfit our model. Therefore we further decrease the number of neurons until the error rate is reaching a local minimum. On the right in Figure 3.13, we can see that the local minimum is between 163 and 20 neurons. Now, let's observe how the error behaves for each parameter when choosing the lowest number of neurons. Figure 3.14 is showing the prediction error for the models with 40,20 and 10 neurons, loss metric: RMSE_COS and 1800 training samples. Remark that the error is ten times smaller than in all previous test cases. Additionally,

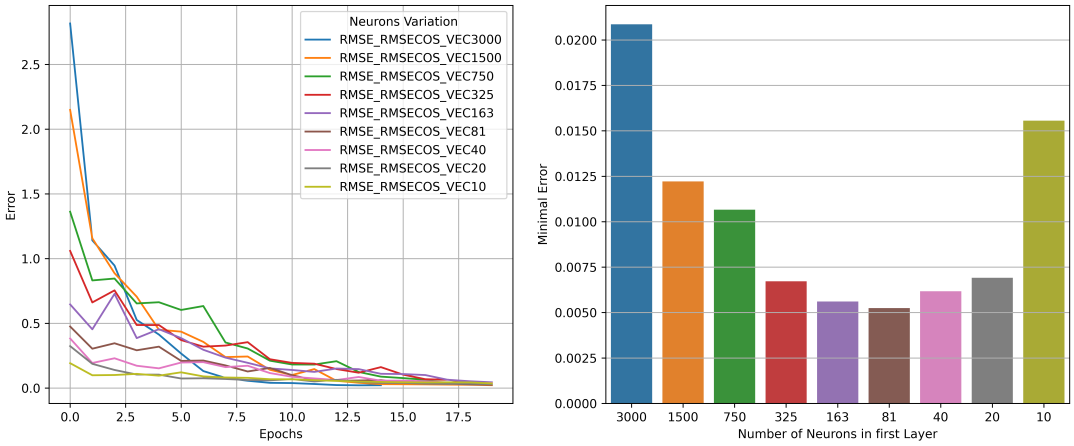



FIGURE 3.13: Training Evolution for different Hidden-Layer sizes. We evaluate the number of neurons needed to achieve a minimum prediction error. Here we observe that for more than 325 neurons, the model is overparameterized and thus performing worse. For a smaller number of neurons, more training epochs should be allowed.

we do not observe any differences between the parameters.

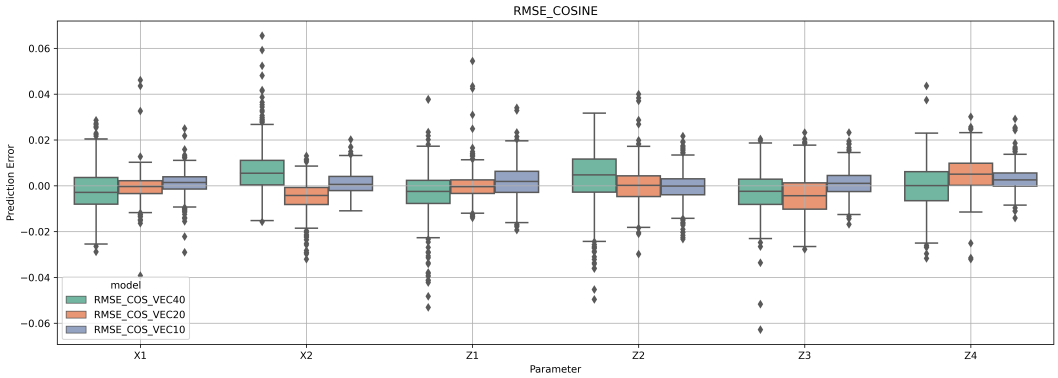



FIGURE 3.14: Evaluating the Effect of used Number of Neurons. Here we show the prediction error for a minimal number of neurons (10, 20, 40). We observe that for 10 neurons the best result are achieved, if we allow more then hundred training epochs.

## 3.5 Discussion

In this chapter, we investigated the use of visual analytic techniques alongside a regression study. We showed how we could subsequently narrow down the optimal set of model parameters. We start with varying the loss

function, then the training fractions, and finally, the number of trainable parameters steered by the number of neurons. While using a special test data setup, we can compare the difference along with each parameter and the overall error. We can even further narrow down our analysis to the value ranges of each predicted parameter. Therefore, the resulting analysis pipeline allows a seaming less switching between overview and detail throughout the different test stages. With our experimental design, we can show the chosen distance metric's effects on learning performance. The derived custom loss metric RMSE_COS allowed us to decrease the number of neurons down to 10 while achieving a prediction error of 2.5% (absolute error of 0.005) regarding the total parameter value range of [-0.1,0.1].

So far, we could show that a simple model with one hidden layer can solve our prediction task. As shown by Brutzkus et.al.[60] and Adoni et.al.[61], our model is still in the domain of linearly separable functions. Even in a setting with only ten neurons, we have 90,076 trainable parameters. In contrast to the 2000 observations, we still have an overparameterized setting. The proof that this still leads to reliable results was already observed by Neyshabur et.al.[62] and Zhang et.al.[63]. Nonetheless, for an explainable model, we do not want to accept this overparameterization. Reducing the hidden layer below ten will result in a less optimal model. As such, we can only reduce the input shape. A common approach here is to only use the $n$-maximal values instead of using all cells. For image-based strategies, convolutional layers are typically added. But all these approaches lack good interpretability, which we want to avoid.

We first want to investigate what the model is learning and how the parameters are influencing this. Therefore we analyze the resulting difference between a target field and the training data concerning its prediction parameters.

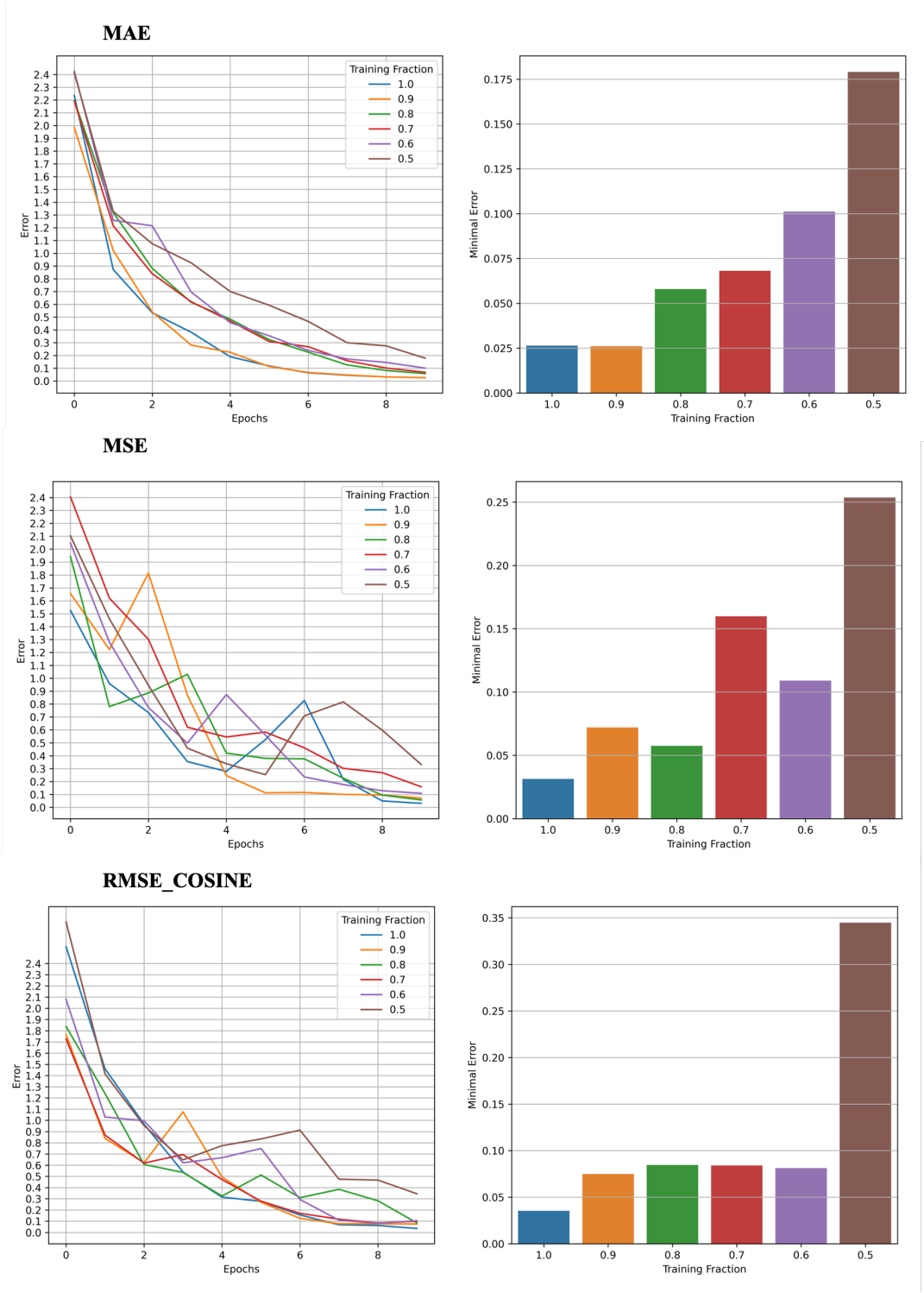



FIGURE 3.15: Training Evolution for different Loss-Metrics and Training Fractions. Here we combined two important views for monitoring the training process. On the left, the evolution of error prediction through the epochs is shown. Based on this view, we can decide if more epochs for training are need or if we have to adapt the restart function. On the right, the minimal achieved error for each training fraction is shown. It is used to decide what training size is needed.

# Chapter 4

# Visual Classifier Performance Evaluation

We investigated visual analytics to evaluate a regression model's performance using neural networks in our previous approach. What remains after this analysis is the black-box behavior of the model itself. Therefore we want to analyze the prediction parameters' influence on the resulting difference between a target deformation and the training deformation fields. Therefore we summarize the field's difference into one value using the root-mean-squared error on the magnitude of the resulting deformation difference field. It allows us to use the difference results for the whole training data set for our analysis, as analyzing 2000 $\times$ 3000 values would be impossible. We then analyze the resulting difference concerning the to be predicted simulation parameters. First, we normalize the resulting difference to a similarity value. We assume that the pairs with the highest similarity values to have the most significant impact on the learned model. Based on this assumption, we first want to define a threshold to separate the training data into two classes: an Acceptance class for $Sim_i > th$ and a Withdrawal class for $Sim_i \leq th$. Thus the first step of our analysis pipeline is targeted towards the definition of such a threshold. Afterward, we are investigating the resulting classes concerning the distribution of the parameters. We will use the concept of a scatter-plot matrix for this. Afterward, we apply dimension reduction techniques to narrow down the number of parameter pairs to the dominant ones. Finally, we want to analyze the parameter ranges in more detail. As we now have prepared our data to be represented by two classes, we can train binary classifiers to predict those classes based on the parameters. Those classifiers' performance could then be analyzed using either ROC-curves or with the concept of decision boundaries. The ROC-curves can be used to compare the models' sensitivity and various target- training pairs. At the same time, the decision boundaries visualize the sensitivity of the classification model in the parameter space. If

we are now changing the input dimension of the deformation field based on segmentation strategy, we could use the classification models' sensitivity as a benchmarking tool. The idea is to use the classifier model's sensitivity as an indicator for quantifying similar behavior in the segmented region.

**Related Work** The survey of Zhou et al. [64] classifies visualization methods based on the application scenario and industry sector. Our approach is driven by the needs of the automotive industry, more specifically, automobile preproduction. Regarding Zhou's taxonomy, the proposed method concerns the phase between the design and production phases. Xu et al. introduced ViDX - Visual Diagnostics for the analysis of assembly line performance [65]. Their tool combines historical and real-time data while focusing on anomaly detection and identifying inefficiencies. The system devised by Ramanujan et al. [66] analyzes similarities and associated performance metrics in computer-aided design (CAD) repositories. Thus it mainly evaluates the planned data. It can be useful for determining similar parts, which results in similar trained models. The approach developed by Wu et al. [67] concentrates on condition monitoring and is therefore targeted for machinery sensory data. A prominent example of the combined use of decision trees and dimension reduction techniques is included in Sun et al. [68], which motivated our analysis pipeline. Machine learning (ML) based analysis approaches pose new visualization challenges since interactive and visual ML systems are still in their infancy [69].

## 4.1 Experimental Design

Following the steps of parameter estimation, we can employ several methods. To quantify similarity for vector field ensemble elements, for example, Jarema et al. [70] proposed a visual analysis method combining statistical analysis and comparative visualization used in 2D space. Considering our application scenario, i.e., sheet metal deformation, we contemplate a deformation field on a surface as a 2D vector field.
Definition and labeling of clusters of ensemble elements can be carried out on the local scale using an entire surface geometry, using the method of Rieck et al. [71] that clusters elements based on persistent homology. Such an approach addresses the problems encountered with unlabeled data on a given surface. When a user labels classes, the level of confidence in decisions made

is an important aspect. Kumpf et al. [72] considers this aspect, helping with the understanding of different outcomes. At first, we investigate the impact on a summarizing high level. Therefore, we investigate the pairwise metrics used for difference calculation between the reference part and each ensemble member. The 1-dimensional summarizing metrics for the segmented cell cluster results in one result per segment per ensemble member. The distance metric is normalized, leading to a general similarity term using the following formula:

$$S_i = 1 - \frac{\Delta(U_i, \hat{U})}{max(U_i)}. \tag{4.1}$$

Here, $U_i$ refers to each displacement field of the training data set, and $\hat{U}$ refers to the target displacement field. $\Delta$ is synonymous for the chosen distance metric. In our use cases, we use the root-mean-squared (RMS) error on the displacement magnitude. The segments are then further classified into an acceptor and withdrawal class. For example, we could use the 80% quartile to define the classification, which means that 20% of the highest values belong to the acceptance class and the rest to the withdrawal class. To investigate the general applicability of our method, we then choose seven special parameter settings, on which we investigate the results. The chosen settings are described in Figure 4.1:

We choose the parameter settings to represent the boundary use cases and represent the lower bound of achievable results. We inspect pure translation cases either in one direction (ID5) or in all directions (ID365). In contrast to the pure translation, we chose a setting with pure rotation without deformation (ID730). To check the sensitivity of the metrics for symmetrical behavior, we chose two sets (ID2 and 7), which result in a nearly mirrored displacement field (See Figure.4.1).

### 4.1.1 Selected Machine Learning Methods for Classification

In the final step of parameter estimation, we use a set of supervised learning binary classification methods. All implementations used in this evaluation are from the "scikit-learn" [73]. The classifiers are chosen to represent a vast range of methodologies, considering the application scenario's special properties. Analyzing the commonalities and differences helps us to select the appropriate ones for further optimization. We developed an interactive system in which the user can smoothly switch between different classification methods and models. In the resulting view, we use ROC-curves and

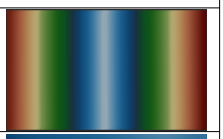
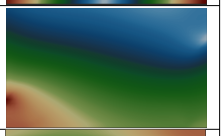
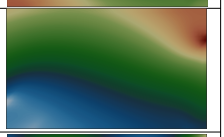
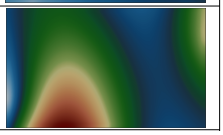
| ID | X1 | X2 | Z1 | Z2 | Z3 | Z4 | Property | Deformation Field |
|---|---|---|---|---|---|---|---|---|
| 5 | 0.1 | 0.1 | 0.0 | 0.0 | 0.0 | 0.0 | Pure translation in X | - |
| 365 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | Pure translation | - |
| 730 | 0.0 | 0.0 | 0.1 | - 0.1 | - 0.05 | 0.1 | Rotation w/o deformation |  |
| 2 | 0.1 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | Ambigious displacement – Symmetry of parameters |  |
| 7 | 0.0 | - 0.1 | 0.0 | 0.0 | 0.0 | 0.0 | |  |
| 458 | 0.1 | - 0.1 | - 0.1 | 0.1 | - 0.1 | 0.1 | Maximum Load |  |

FIGURE 4.1: Special Parameter Sets. Chosen parameter settings for the experimental design. The parameter sets are chosen to observe the model behavior in boundary cases.

decision boundaries to visualize the classifier's prediction performance (see Figure 4.2).

#### Nearest Neighbor Classifications

It is considered as a type of instance-based learning, i.e., it is not constructing or fitting any internal model but classifies the training data based on a majority vote [74]. The only parameters to choose is the number of neighbors for the query and a weight function. With increasing k, noise is reduced in favor of the distinctness of classification boundaries. In general, this method suffers from the "curse of dimensionality" [75], [76]. In our evaluation, $k = 3$ turned out to be a good tradeoff value, as we expect to have only a small degree of noise in the data. The weights are uniform, as our input space is uniform.

#### Support Vector Classifier

SVC's goal is to find a function $f(x)$ that has at most $\epsilon$ deviation from the obtained targets $y_i$ for all the training data, and at the same time, is as flat as possible' [77]. The main parameter for this method is the chosen kernel. In this evaluation, we choose the radial basis function to allow for more general shapes in the classification.

#### Gaussian Process Classification (GPC)

It uses a probabilistic approach for the classification. It uses the Gaussian Process [78] to classify the training data based on a given kernel. We chose the radial basis function for the kernel as well.

#### Decision Tree and Random Forest Methods

These methods follow the principle of learning a set of simple decision rules (if-then-else) from training data[79]. Decision trees' process resembles an experienced person's manual progress familiar with an assembly [68]. Decision trees are the only classifier in this evaluation that can be easily visualized even for higher dimensions. The decision tree has the highest number of parameters. First, the quality criterion for the split is chosen. We picked Gini impurity, as we did not observe any significant differences for $depth > 4$ in comparison to entropy (information gain). For the split strategy, we chose best over random, as our training data is nearly uniform. For all other tree

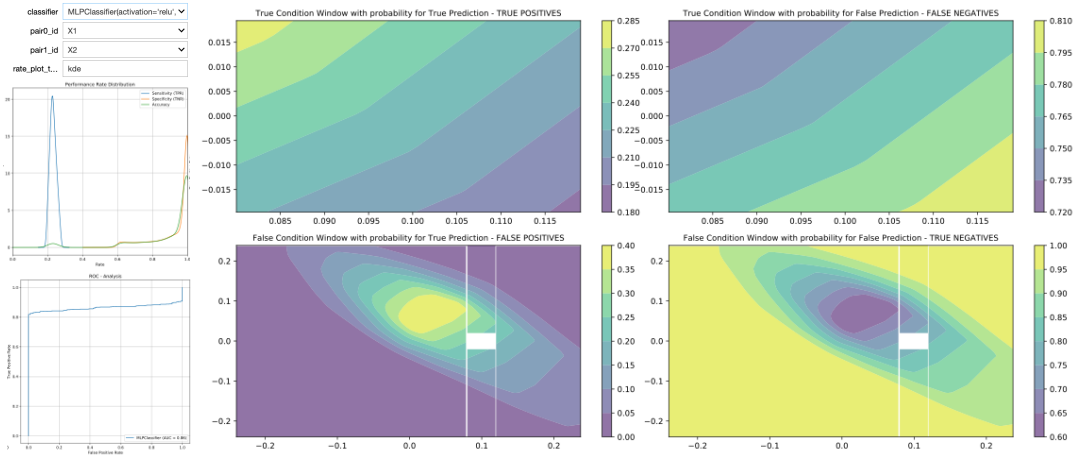



FIGURE 4.2: Confusion Analysis. The view for detailed classifier analysis. The user can select the parameter pairs of interest and investigate the classifier results in those two projections. The classifier's decision boundaries are plotted in a matrix style using a confusion or contingency matrix principle. Additionally, we can see the ROC curve and the distribution of the key performance indicators (Accuracy, Sensitivity, Specificity), which are also used in the overview. Depending on the application's goal, one or the other column/ row combination is more important.

parameters, we used default values, either minimum or maximum values. The simple decision tree tends to over-fit with these settings. We extended our approach using the random forest classifier, which fits several decision trees to various sub-samples and averages the results to address this shortcoming. We found out that $n = 10$ subdivisions are a fair tradeoff between computational time and increased predictive quality.

### Neural Network

We used a fundamental neural network definition [80] as a starting point for further investigations. A multi-layer perceptron classifier was chosen with rectified linear unit function for the activation, one hidden layers and a total number of 40 perceptrons ($6 \times 16 + 16 + 16 \times 2 = 144$ weights). The default L2 penalty (0.0001) was chosen with a constant learning rate. A small test series revealed that a maximum of 4000 iterations is a good choice as a termination criterion.

### Naive Bayes Classifiers

These classifiers are quite straightforward among rule-based classifiers [81]. They use the naive assumption of conditional independence between every pair of features in a dataset. We can use a maximum a posteriori estimation for classification. We assumed that the likelihood of our features was Gaussian-distributed for our evaluation, meaning that small changes in each parameter lead to relatively the same changes in target value. We do not expect to have sharp classification boundaries.

## 4.2 Visual Analytics Pipeline

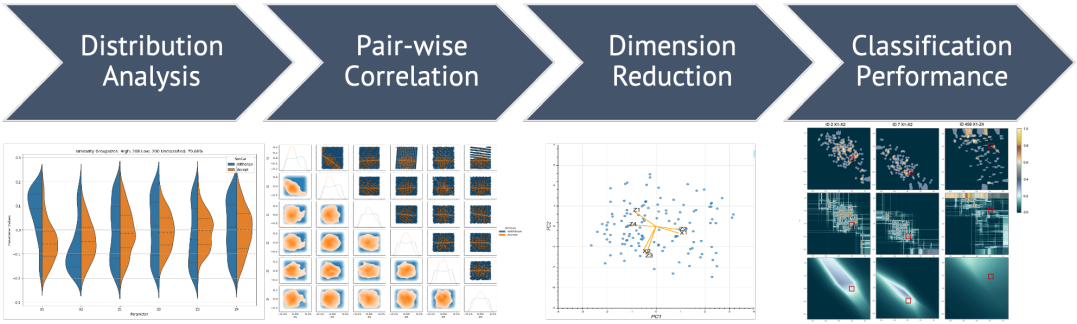



FIGURE 4.3: Visual Analytics Pipeline for the Inverse Classification. We start with a distribution analysis using violin-plots, followed by a correlation analysis using a scatterplot matrix. We then reduce the number of pairs in a dimension reduction step and finally train a classifier for pair-wise comparison of the parameters.

We start our evaluation with a rough analysis of the simulation ensemble. Therefore we calculated the statistical properties for both fields (Displacement "U," Mises-Stress "S") for each cell. The resulting field still preserved the original topology of the part and investigated the whole geometry distribution. In Figure 4.4 the standard deviation of both fields is visualized. Additionally, we used contours to highlight further the regions with a similar response to the induced loads.

The complete pipeline is depicted in Figure 4.3. We start with analyzing the parameter distribution and interactively setting an initial threshold for

our acceptor and withdrawal class. As we have two classes to compare, a spitted violin plot [82] is chosen. After investigating each parameter on its own, we take pairwise correlations into account in the next step. For the six assembly parameters, this results in 15 correlation pairs. The concept of a scatter plot matrix allows us to visualize these pairs compactly. We use the classification methods to learn the prediction probability for each assembly parameter in our final step. However, we can only adequately visualize the result in 2d- planes. Therefore we would have to compare 15 sections of our 6-dimensional hypercube to investigate each pair of parameters. We insert a dimension reduction step in our pipeline to reduce the visual complexity, focusing only on the relevant pairs. As dimension reduction techniques, we used hierarchical regression and the principal component analysis.

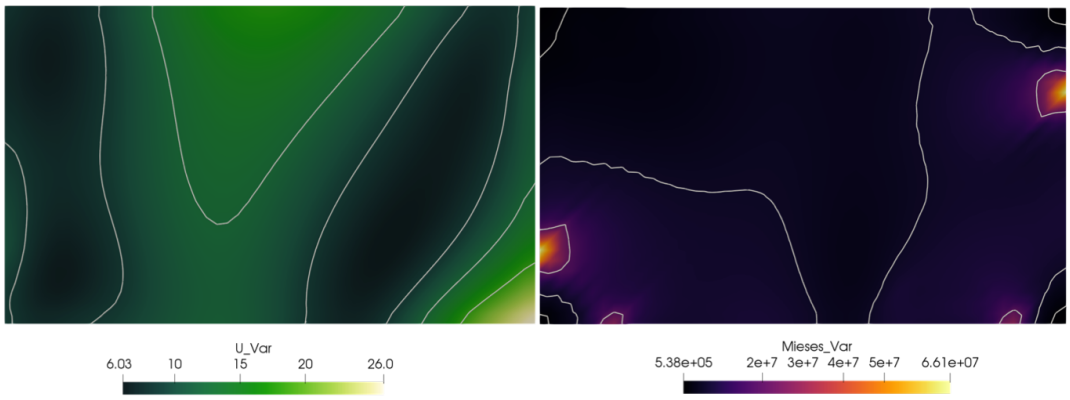



FIGURE 4.4: Visual Ensemble Analysis. The resulting field's standard deviation (Displacement U and Mises-1 Stress S) are calculated along with each cell. It allows visualization of the resulting deviations along with the whole geometry. Contours are used to highlight similar behaving regions. The displacement field shows the linking of the Z-Parameters, in the dark green areas, while the stress field indicates a higher impact for the X-parameters (yellow). See Figure A.1 for definitions of the parameters.

### 4.2.1 Parameter Distribution Analysis

Our first parameter space visualization uses violin plots [82], see Figure 4.5. We define a threshold value for similarity to separate our data set into two classes. For each class, we plot the distribution of each parameter. With this visual representation, we can compare parameter impact in terms of how often it is present in each class. We can relatively compare the mean and

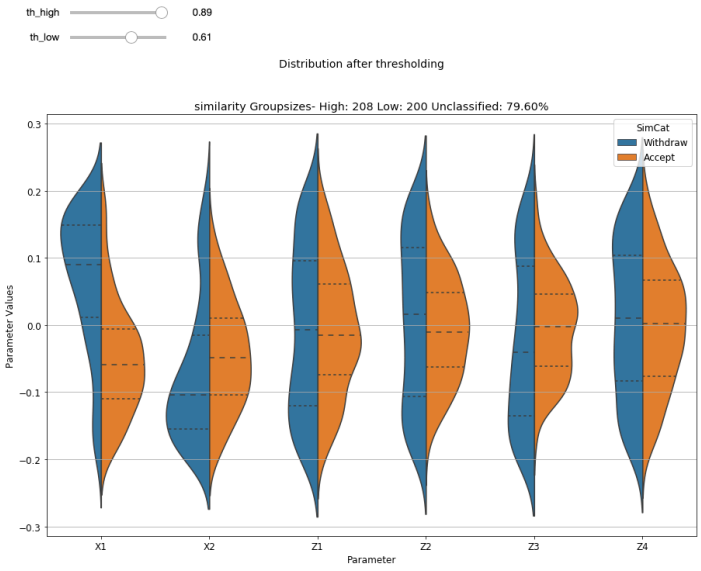



FIGURE 4.5: Distribution patterns using violin plots. An example of how violin plots can interactively spot patterns in the distributions for different similarity thresholds. The orange distribution refers to the acceptance class, the blue to the withdrawal class. For each of the six parameters, the distribution is shown. Given the example, the values of X1 are more frequently negative for higher similarity values and positive for lower values, which copes with the used test sample used here.

quartiles, but, more importantly, spot differences in the shape of the distribution. Of particular interest are distributions with multiple maxima or minima (multi-modal distributions), as these suggest some symmetry. We added interactive steering of the upper and lower bounds for each of the classes to our visualization system. It allows the user to investigate different thresholds and choose an appropriate one for his analysis.

At first, we analyze the distribution of similarity while setting the threshold value for acceptance and withdrawal. Figure 4.5 shows the results for one of the ambiguous cases (ID7).

### 4.2.2 Correlation Analysis

The predictive result for separately investigating each parameter is limited. Therefore, we extend the analysis by using scatter plots, kernel density estimates, and linear regression for each parameter pair and compare the distribution and correlation with similarity in more detail, see Figure4.6. The arrangement in a scatter-plot matrix (SPLOM) allows us to combine three types of visualizations in one view. In the lower diagonal, we plot each class's distribution based on the respective parameter values using a kernel density

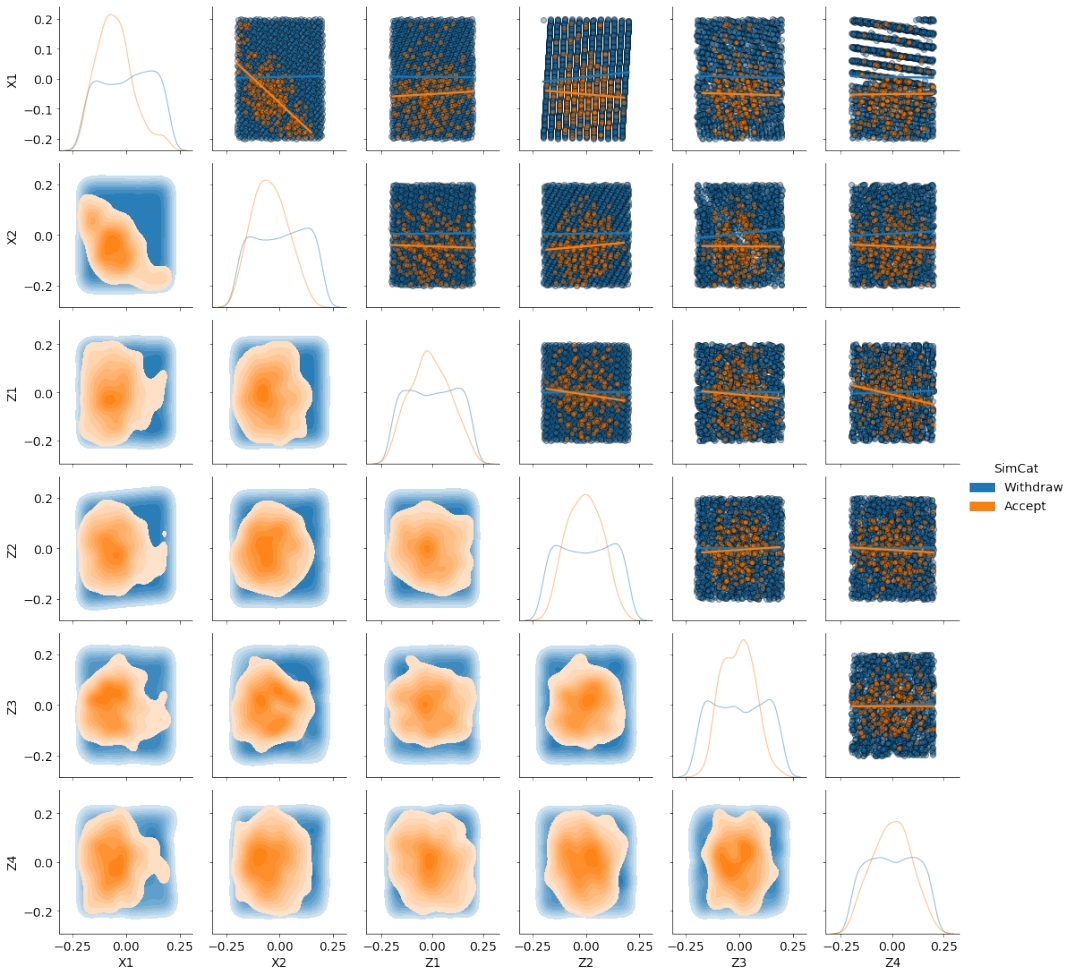



FIGURE 4.6: Scatter plot matrix (SPLOM) with kernel-density estimator(KDE). For complete pairwise correlation analysis, we use the concept of a SPLOM. We use a simple scatter plot in the upper diagonal, colored based on the set class labels and enhanced with linear regression curves for each class. On the lower diagonal, we used a KDE to visualize the shape of the two classes' distribution. This high-level view allows us to spot relevant correlations in addition to the ones already found in the ensemble analysis, e.g., row three (Z1) and column six (Z4) indicates a negative correlation.

estimator (KDE). It allows us to identify high-density regions for parameter combinations. In the upper diagonal, we use a scatter plot with linear regression for each class to highlight the interaction between the two parameters. The correlation from the upper diagonal plots allows us to determine in what direction to look for cluster separation. On the main diagonal, we plot the distribution density of the classes for each parameter. Correlation maps [83] and interactive regression lenses [84] are methods tackling this issue. Causal and correlated behaviors often coincide, and direct visualization methods rely on correlation coefficients only. The method of Wang et al. [85] solves this problem by including additional quality measures computed via statistical correlation in a visualization. For example, a visual analysis approach to evaluate the capabilities of epidemic prediction models was discussed by Bryan et al. [86].

### 4.2.3 Dimension Reduction

Projection-based methods, including the method described by Bui-Tahn et al. [87] and Gaggero et al. [88], are appropriate for inverse and optimal design. Combining these methods with machine learning (ML) and neural network techniques has produced promising results for parameter space prediction in inverse design applications [89]–[91]. At this point, we have analyzed the parameters individually and found correlations between them. In the next step, we try to reduce the dimensionality and focus on pairs of parameters or linear recombinations of our initial parameter space. We investigated two techniques, hierarchical linear regression (HLR) and principal component analysis (PCA)[92].

#### Hierarchical Linear Regression (HLR)

In the HLR approach, we use ordinary-least-square (OLS) as the model with one constant factor. First, we apply the linear regression model to each parameter individually and sort the output by explained variance (R-squared). We use this order to incrementally add each parameter in the model and observe its impact on R-squared. Based on the added value to R-squared in the model, we can determine what parameters to focus on. In general, this method allows us to use any model, and it has the highest potential to incorporate application knowledge into the system. Unfortunately, it restricts us to the original parameter space, and choosing a good model is a complicated task. First, we use hierarchical linear regression analysis to order parame-

| ID | HLR | | PCA | |
|---|---|---|---|---|
| | R2 | Order | R2 | Ordering |
| 5 | 22.15 | X2,Z3,Z2,Z4,Z1,X1 | 40.42 | X2,X1,Z1,Z3,Z2,Z4 |
| 365 | 25.18 | X2,Z3,Z1,Z2,Z4,X1 | 42.86 | Z1,X2,Z3,X1,Z4,Z2 |
| 730 | 13.84 | X2,X1,Z2,Z1,Z4,Z3 | 46.50 | X1,X2,Z3,Z1,Z4,Z2 |
| 2 | 14.61 | X2,X1,Z3,Z2,Z4,Z1 | 44.70 | X1,X2,Z1,Z2,Z4,Z3 |
| 7 | 12.73 | X1,X2,Z3,Z2,Z3,Z1 | 43.93 | X2,Z2,X1,Z4,Z1,Z3 |
| 458 | 12.73 | X1,X2,Z3,Z2,Z4,Z1 | 50.98 | X2,Z2,Z1,X1,Z4,Z3 |

TABLE 4.1: Dimension reduction results. Variance (R2) of hierarchical linear regression (HLR) is always lower than that of the first two principal components. The derived ordering varies among the two methods. A well-chosen or known statistical regression model could improve the results drastically. If no model is known, the PCA will be the better choice.

ters by impact. Depending on the particular case, the expressiveness of this ordering varies drastically, see Table 4.1.

**Principal Component Analyis (PCA)**

The Principal Component Analysis (PCA) finds linear recombinations of the original parameter space based on a correlation or covariance matrix. The only parameter we have to choose is the threshold value for similarity. The PCA is applied to each class. From the bi-plots, shown in Figure 4.7, we can determine what original parameters contribute to the principal derived components. The more they align with the axes of a principal component, the better we can separate our dimensions. The lengths of the arrows depict the relative impact of each parameter on the derived components. If multiple principal components have a similarly explained variance, we investigate each combination separately to determine good splits.

## 4.3 Classifier Performance Analysis

### 4.3.1 Confusion Matrix and Decision Boundary

To objectively compare the quality of the parameter estimation, we use the concept of confusion matrix [93]–[95]. Each method itself, and the combination of methods, leads to a prediction range in parameter space. The input test parameter set and a predefined acceptable deviation are defining the true conditions for the confusion matrix. The acceptable deviation depends strongly on the specific application's accuracy goal. In the case of the generic

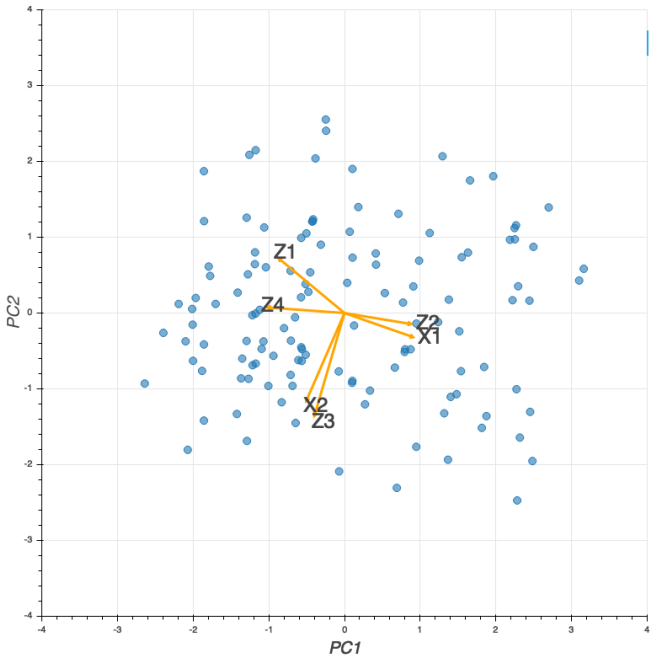



FIGURE 4.7: Biplot PCA. With a biplot's help, we can project the initial parameters concerning their variance portion on the derived first and second principal component (yellow arrows). The length of the arrows refers to the eigenvalues of the decomposition. Given an example, the first principal component mainly consists of X1, Z2, and Z4. The arrows opposite direction indicates a negative correlation between Z4 and X1, Z2. The second principal component consists mainly of X2 and Z3. Z1 has an equal influence on both components.

example, we chose 5% of the total parameter value range. For the real model, we chose 10%. In the confusion matrix we counted the number of true positives $tp$, true negatives $tn$, false positives $fp$ and false negatives $fn$ for our total size of the training data set $n$. For the evaluation and interpretation, we focus on these three ratios derived from those numbers:

$$Accuracy(ACC) = \frac{tp + tn}{n} \tag{4.2}$$

$$Sensitivity(TPR) = \frac{tp}{tp + fn} \tag{4.3}$$

$$Specificity(TNR) = \frac{tn}{tn + fp} \tag{4.4}$$

Accuracy describes the ratio of all correct predictions concerning the total

population. Accuracy does not provide information about the type of correct/incorrect predictions. Therefore, we also use the sensitivity (true positive rate), which describes the method's ability to correctly detect true parameter values. A high value of sensitivity reliably rules out negative predictions, while a positive result in high sensitivity prediction is not necessarily useful for ruling in the parameter set. Specificity deals with this shortcoming. A method with a high specificity value reliably handles positive parameter set predictions. Considering our application, one or the other rate is more important. In machine learning, high accuracy is generally preferred over sensitivity or specificity, but this leads to a series of drawbacks, typically denoted as the Accuracy Paradox [19], [20]. We consider two cases:

1. The preproduction phase and quality assurance benefit from a high sensitivity over specificity to detect anomalies and failures reliably.
2. Mass production and online steering applications favor a high specificity to minimize spoilage.

Once fitting the training data is accomplished, each classifier and its underlying data model can be used to predict the target class, given a set of parameter values. Typically, a classifier does not just return the best class label; it also generates a probability for each class in the model for specified values. More formally,

$$Predict(PS)_{CL} = [p(class_0), p(class_1), ..., p(class_i)] \tag{4.5}$$

$$p(class_k) = 1.0 - \sum p(class_j),\ for\ j \neq k, \tag{4.6}$$

where $PS$ determines the tested input parameter set, $CL$ describes the used classifier and $p(class_i)$ is the probability for predicting $class_i$.

In our experimental setting, the exact parameter values are known. We evaluate the predict function for values in a certain range, with a step size of $h$ used for visualization resolution. We plot results from the predict function as a heat map with value ranges between zero and one, see Figure 4.8 and 4.14. The red square is marking the acceptance window. The sensitivity is the summary result of the Acceptance Window and specificity of the rest; accuracy summarizes both. Based on these compelling visualizations, we could deduce a general workflow for classifier comparison, see Figure 4.3. For each classifier, we start with the summary performance indicators,

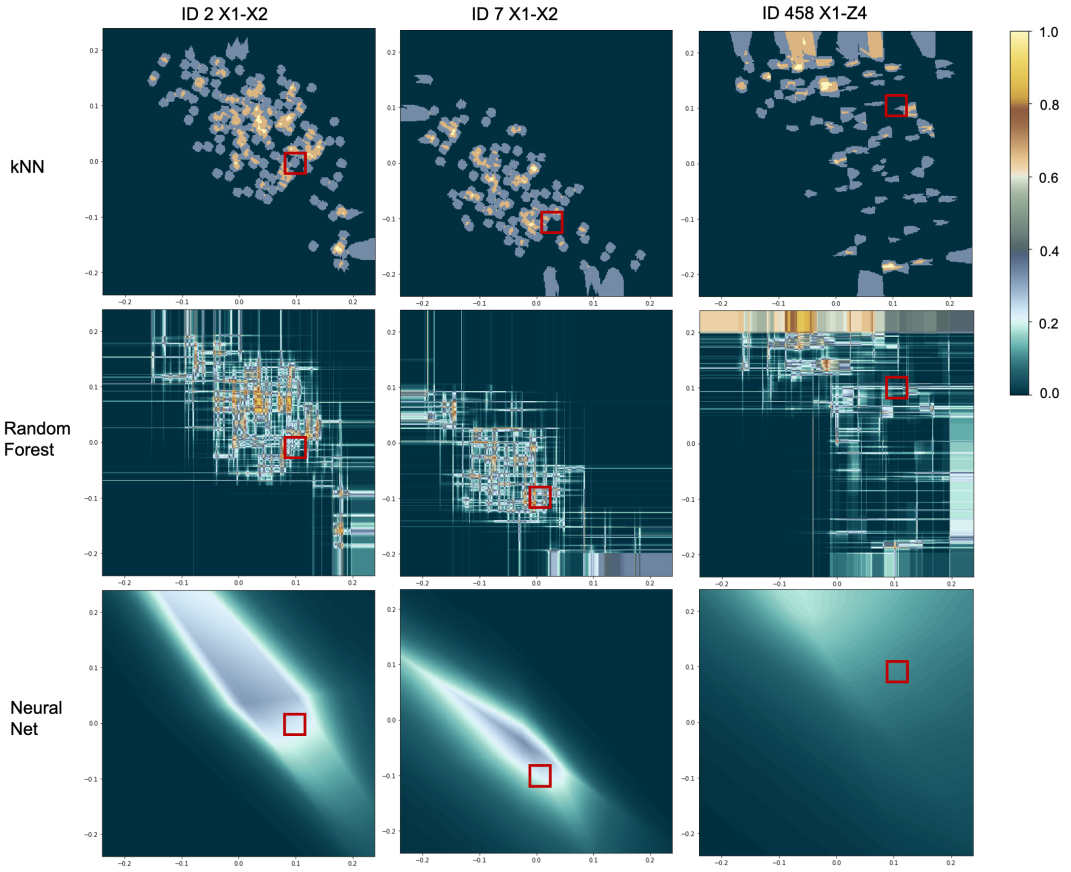



FIGURE 4.8: Decision Boundaries for three selected use cases. The heat maps are visualizing the prediction probability of each classifier. A five-step color map is used to point out sharp prediction boundaries. With higher values favoring the acceptance class and lower values the withdrawal class. The red squares are indicating the correct value ranges of the test data set. Instead of comparing the raw numbers, like, e.g., in ROC curves, this visualization allows us to compare the predictive model's shape between the classifiers. For the ambiguous cases (ID2, 7), the predictive quality is better than for the maximum load case (ID458). Also, the shape implies a more stable model compared to the maximum load case.

extend the analysis using ROC curves 4.13 and finally evaluate the shape of the decision boundary map (see Figure 4.8).

## 4.4 Results and Implications for the regression model

We found out that the 80% quartile is a good indicator for a threshold on the similarity for the metal sheet example from several analysis steps. For the real automotive body part, a lower value is suited better. Here we used the 75% quartile. In all further investigations, we will keep these values fixed.

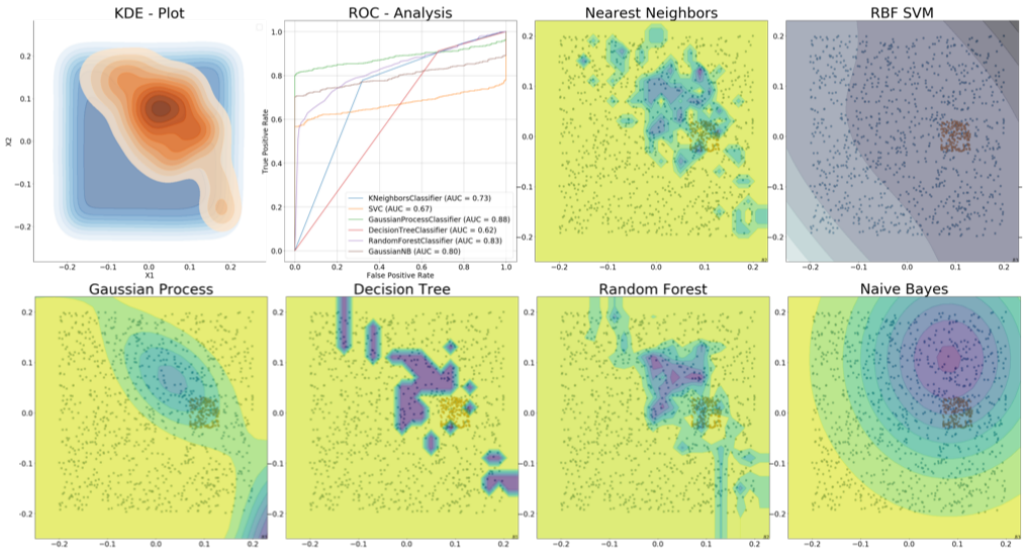



FIGURE 4.9: Classifier Comparison using two parameters. The KDE-plot shows the distribution of class labels. The ROC curve depicts the ratio between sensitivity and inverse specificity. The figure in the upper-left corner represents The optimal ratio. The additional plots show the decision boundary of classifiers. Higher values (dark) indicate higher class prediction probability. Regarding the SVM, colors imply different levels of predicted hyperplanes, separating classes in parameter space. Decision boundaries vary significantly between classifiers, leading to different prediction ranges. In the shown ambiguous symmetrical case, one would expect two prediction clusters, (X1, X2)=(0.1,0.0) and (X1, X2)=(0.0,-0.1). Instead, one cluster follows a diagonal and shifts in a direction opposite of the symmetry pair.

### 4.4.1 Classifier Evaluation

First, we evaluate the general applicability of each classifier on the ambiguous use case. In Figure 4.9 we visualized the resulting decision boundaries of the classifiers along with the corresponding KDE-plot of the X1-X2 pair and the ROC curves for each classifier. Figure 4.10 gives an overview of all exceptional cases' summarizing performance indicators. Here we already narrowed down the used classifiers to the k-nearest-neighbor, Random Forest, Naive Bayes, and Neural Network. We eliminated the Gaussian process due to its expensive computation costs, making it not feasible for usage in an interactive visual analytics tool. Further, the Random Forest is a generalization of the decision tree, which is less pruned to get stuck in the first encountered local minimum. The resulting hyper-planes of the support vector machine are hard to interpret, and therefore we opt them out in our visual analytics tool.

**k-nearest-neighbor Method** The k-nearest-neighbor (KNN) method performs well for cases with few dominant effects like the ambiguous cases 2 and 7 studied here, where only x1 and x2 are varied. In general, we can observe that the classifier performs well in the x1-x2 projection for all cases. The ROC curves for this approach vary significantly. In the sensitivity distribution, we can observe the absence of a low sensitivity cluster in case 7 compared to case 2. Analyzing the decision boundaries (see Figure 4.9, we find multiple small clusters with high prediction probabilities. If they spread out slightly, sensitivity drops drastically from 0.437 (right) to 0.292 (left). This effect shows that the *KNN* approach is very sensitive to outliers.

**Gaussian Process** The Gaussian Process performs well for the exceptional cases of translation and maximum load. The accuracy for all tested sets is between 0.74 (ambiguous displacement) and 0.89 (Translation), with a mean of 0.81. In the ambiguous displacement cases, the sensitivity is nearly 0, while the specificity is nearly 1. The same holds for the rotation. Additionally, the Gaussian process' derived model is the most complex one, leading to 100 times higher computation times in the evaluation.

**Random Forest** The random forest approach is, on average, the best performing classifier for sensitivity (mean=0.25, SD=0.11). It performs similarly to the KNN approach, but it is more stable in the presence of outliers, producing better results on average due to its random nature. We can observe this for the ROC curve as well, which shows slighter changes than for the KNN approach. There is only one dominant cluster in the distribution cluster, which stretches out over the three clusters from the KNN. Due to its weighting of local minima, it allows us to detect clusters better when compared to the KNN approach, see Figure 4.9. Nevertheless, the randomness makes it harder to predict the right interval.

**Neural Network** The rudimentary trained neural network performs poorly in most cases. The average sensitivity is 0.172, and the standard deviation is only 0.078, which makes it the most stable method. In the ROC analysis, on the other hand, the neural net achieved the overall best results. In the sensitivity distribution, there is only one dense cluster, but with low sensitivity. There are only minor variations between the cases, which confirms the low standard deviation. When analyzing the decision boundaries, we observe that the neural net forms one clear cluster around the target parameter set,

| ID | Property | | x1,x2 | | | | z1,z4 | | | | z2,z3 | | | | x1,z3 | | | | x2,z1 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | kNN | RF | NN | NB | kNN | RF | NN | NB | kNN | RF | NN | NB | kNN | RF | NN | NB | kNN | RF | NN | NB |
| 2 | Ambigious case | sens | 0.292 | 0.290 | 0.229 | 0.161 | 0.070 | 0.116 | 0.092 | 0.138 | 0.375 | 0.362 | 0.406 | 0.195 | 0.142 | 0.167 | 0.118 | 0.203 | 0.172 | 0.158 | 0.086 | 0.129 |
| | | spec | 0.936 | 0.930 | 0.929 | 0.942 | 0.939 | 0.936 | 0.908 | 0.930 | 0.940 | 0.934 | 0.935 | 0.942 | 0.937 | 0.938 | 0.928 | 0.950 | 0.973 | 0.974 | 0.943 | 0.971 |
| | | acc | 0.910 | 0.904 | 0.900 | 0.910 | 0.889 | 0.888 | 0.861 | 0.884 | 0.907 | 0.900 | 0.904 | 0.899 | 0.904 | 0.907 | 0.895 | 0.920 | 0.927 | 0.928 | 0.893 | 0.922 |
| 7 | | sens | 0.437 | 0.397 | 0.208 | 0.127 | 0.145 | 0.132 | 0.062 | 0.110 | 0.380 | 0.276 | 0.063 | 0.160 | 0.260 | 0.230 | 0.059 | 0.121 | 0.259 | 0.240 | 0.077 | 0.133 |
| | | spec | 0.935 | 0.933 | 0.931 | 0.928 | 0.953 | 0.960 | 0.937 | 0.955 | 0.965 | 0.964 | 0.941 | 0.971 | 0.942 | 0.934 | 0.913 | 0.937 | 0.958 | 0.950 | 0.908 | 0.944 |
| | | acc | 0.906 | 0.903 | 0.888 | 0.881 | 0.907 | 0.912 | 0.886 | 0.906 | 0.931 | 0.925 | 0.890 | 0.924 | 0.902 | 0.893 | 0.864 | 0.889 | 0.886 | 0.885 | 0.823 | 0.861 |
| 5 | Translation in X | sens | 0.565 | 0.556 | 0.355 | 0.434 | 0.053 | 0.064 | 0.088 | 0.112 | 0.272 | 0.310 | 0.092 | 0.180 | 0.361 | 0.356 | 0.410 | 0.285 | 0.197 | 0.222 | 0.157 | 0.267 |
| | | spec | 0.957 | 0.950 | 0.946 | 0.958 | 0.937 | 0.926 | 0.909 | 0.921 | 0.951 | 0.942 | 0.913 | 0.946 | 0.963 | 0.966 | 0.961 | 0.969 | 0.969 | 0.969 | 0.965 | 0.974 |
| | | acc | 0.941 | 0.933 | 0.922 | 0.936 | 0.885 | 0.877 | 0.861 | 0.873 | 0.911 | 0.906 | 0.865 | 0.902 | 0.939 | 0.940 | 0.938 | 0.941 | 0.937 | 0.938 | 0.932 | 0.945 |
| 365 | Pure Translation | sens | 0.474 | 0.454 | 0.284 | 0.375 | 0.203 | 0.169 | 0.158 | 0.198 | 0.277 | 0.278 | 0.217 | 0.316 | 0.288 | 0.308 | 0.213 | 0.318 | 0.262 | 0.292 | 0.224 | 0.300 |
| | | spec | 0.949 | 0.948 | 0.943 | 0.953 | 0.943 | 0.937 | 0.935 | 0.941 | 0.951 | 0.949 | 0.950 | 0.959 | 0.951 | 0.948 | 0.947 | 0.956 | 0.963 | 0.961 | 0.960 | 0.969 |
| | | acc | 0.930 | 0.928 | 0.916 | 0.929 | 0.913 | 0.906 | 0.904 | 0.911 | 0.924 | 0.922 | 0.920 | 0.933 | 0.924 | 0.922 | 0.917 | 0.930 | 0.934 | 0.934 | 0.930 | 0.942 |
| 730 | Rotation | sens | 0.448 | 0.428 | 0.261 | 0.143 | 0.185 | 0.235 | 0.193 | 0.255 | 0.176 | 0.206 | 0.232 | 0.277 | 0.343 | 0.391 | 0.283 | 0.190 | 0.203 | 0.211 | 0.162 | 0.184 |
| | | spec | 0.917 | 0.911 | 0.897 | 0.897 | 0.935 | 0.920 | 0.932 | 0.942 | 0.865 | 0.811 | 0.869 | 0.871 | 0.911 | 0.876 | 0.901 | 0.901 | 0.937 | 0.915 | 0.926 | 0.939 |
| | | acc | 0.890 | 0.883 | 0.860 | 0.853 | 0.904 | 0.892 | 0.902 | 0.914 | 0.796 | 0.751 | 0.805 | 0.811 | 0.878 | 0.847 | 0.865 | 0.859 | 0.895 | 0.874 | 0.882 | 0.895 |
| 458 | Maximum Load | sens | 0.317 | 0.304 | 0.109 | 0.163 | 0.112 | 0.111 | 0.090 | 0.093 | 0.071 | 0.070 | 0.056 | 0.048 | 0.025 | 0.062 | 0.060 | 0.060 | 0.106 | 0.089 | 0.091 | 0.131 |
| | | spec | 0.946 | 0.947 | 0.945 | 0.952 | 0.920 | 0.820 | 0.934 | 0.935 | 0.944 | 0.907 | 0.940 | 0.942 | 0.951 | 0.939 | 0.945 | 0.951 | 0.904 | 0.861 | 0.877 | 0.894 |
| | | acc | 0.920 | 0.919 | 0.910 | 0.920 | 0.838 | 0.748 | 0.847 | 0.848 | 0.910 | 0.874 | 0.905 | 0.907 | 0.915 | 0.905 | 0.910 | 0.916 | 0.822 | 0.782 | 0.797 | 0.816 |

FIGURE 4.10: Performance results (Sensitivity, Specificity, and Accuracy) for selected special cases, projections (x1,x2; z1,z4; z2,z3; x1,z3; x2,z1) and the four studied classifiers: Nearest neighbor (KNN), Random Forest (RF), Neural Network (NN) and Naive Bayes (NB). The best and worst-performing prediction for each entry is marked. Significant performance variation exists. Most classifiers have difficulties handling the ambiguous and the maximum load case. We observe that there is no overall best performing classifier. Even the rudimentary trained neural network sometimes outperforms the other approaches. A more detailed view is needed to understand the differences among and within the classifiers.

see Figure 4.9, but with low prediction probability. This observation is an example of misled deduction based on merely analyzing methods' higher-level performance characteristics, e.g., ROC curves.

**Naive Bayes** The naive Bayes approach typically overfits the "dominant optima" in parameter space. It performs best for cases where only one optimum exists (Case 5, 365), and noise or ambiguities are barely existing (Case 2, 7). The approach is very sensitive to the chosen projection in parameter space. The average sensitivity is 0.193, slightly higher than the one for the neural network, with a standard deviation of 0.058. Focusing on the ROC analysis, it performs worse. However, the distribution analysis also shows that it also has one clear and dense clustering like the neural net, but with lower sensitivity. Figure 4.10 summarizes each classifier's performance for each of our seven selected exceptional cases.

### 4.4.2 Correlation Results

Analyzing the parameter cross-correlations using the scatter-plot matrix, we found nearly the same pattern in the comparison of X1 and X2 as depicted in

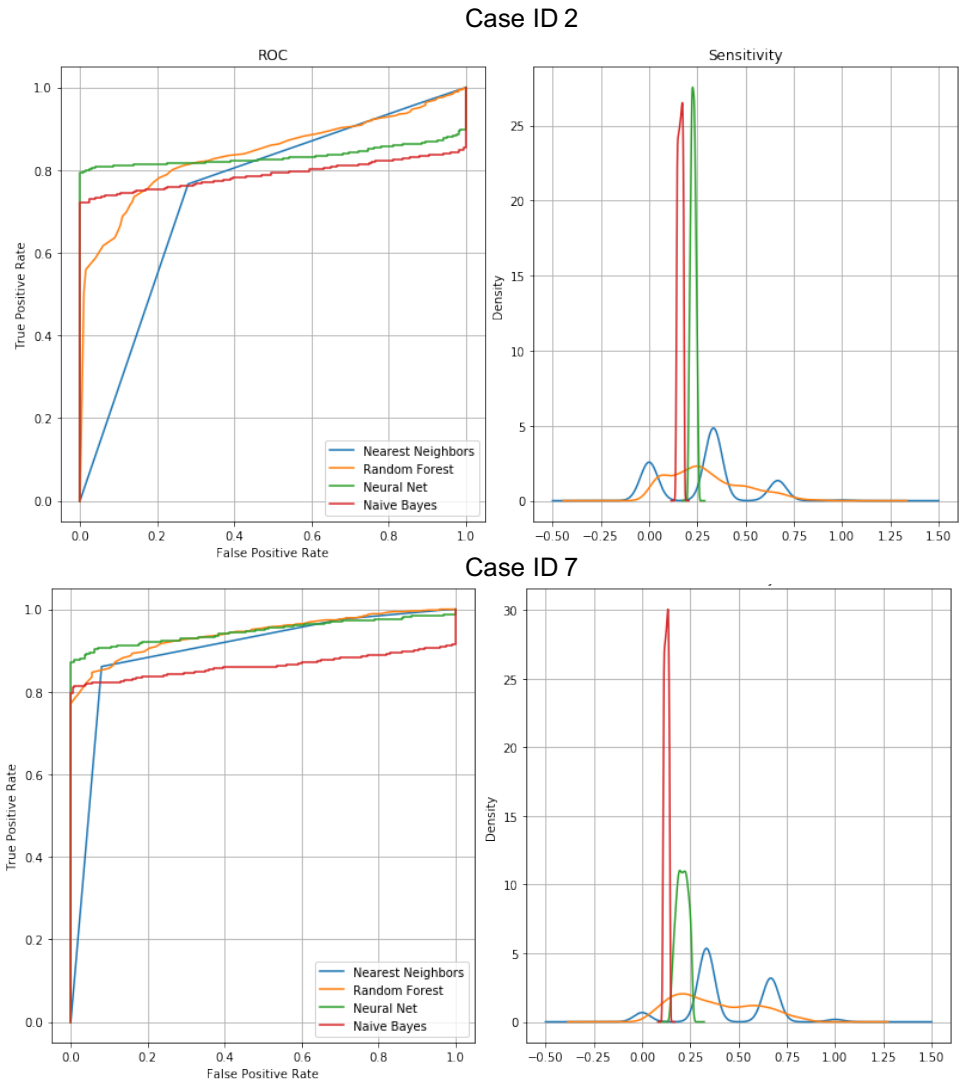



FIGURE 4.11: Classifier Comparison based on performance indicators. Comparing the results for the ambiguous cases for the k-nearest neighbor (KNN), Neural Network, naive Bayes, and random forest approaches. We would expect similar behavior for a stable classifier in this case. We observe differences in the ROC curves for all classifiers with overall better results for Case 7. The sensitivity distribution remains similar for the neural net and naive Bayes.

Figure 4.12. It shows that those parameters are highly correlated and imply focusing on selecting appropriate segmentation techniques.

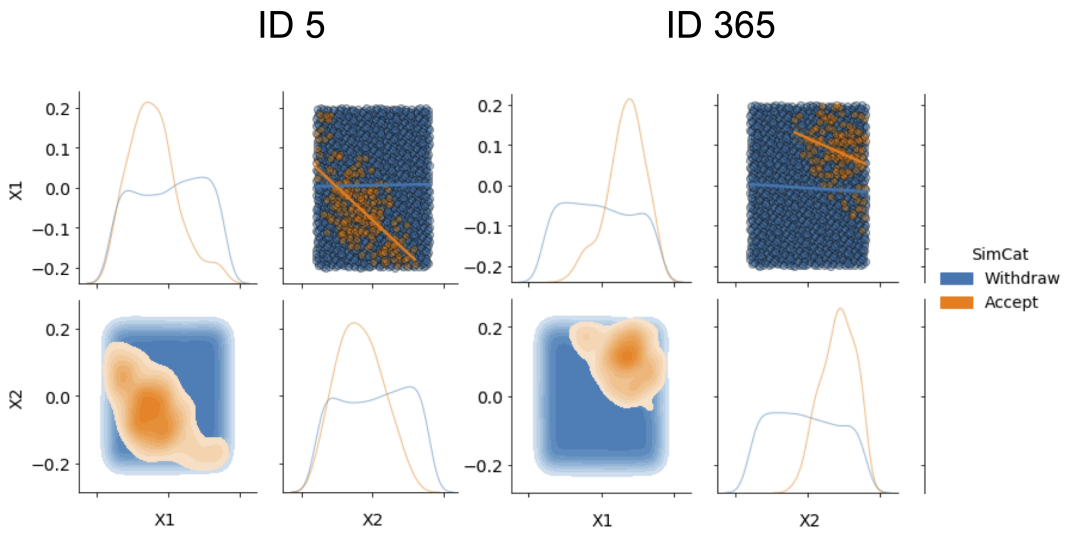



FIGURE 4.12: Comparing Correlations with SPLOM. We can use the SPLOM to analyze the ambiguity in the distribution shift. For both cases, we identify a strong negative correlation between X1 and X2, which results in the observed ambiguity in the distribution shifts.

### 4.4.3 PCA Results

To reduce the complexity, we use the PCA to summarize the parameter correlations into two principal components. The PCA turned out to be the most intuitive and reasonable method here. Using the bi-plot from Figure 4.7, we can form groups for (X1, Z2, Z4) and (X2, Z3). Z1 has an equal influence on both groups. The vector's opposite direction indicates a negative correlation between Z4 and X1, Z2. The vectors' length for X2 and Z3 further implies a more substantial influence on the components and influences the first component. These results cover our findings from the SPLOM analysis and show a strong correlation between X1 and X2. Additionally, it also arranges the other parameters accordingly, leading to a good separation, which we can use to narrow down the number of analysis steps for the segmentation task.

### 4.4.4 Overall Observations

The study's complete results can be found in [96]. We will summarize the results based on the dominant findings for the tuning of our regression model.

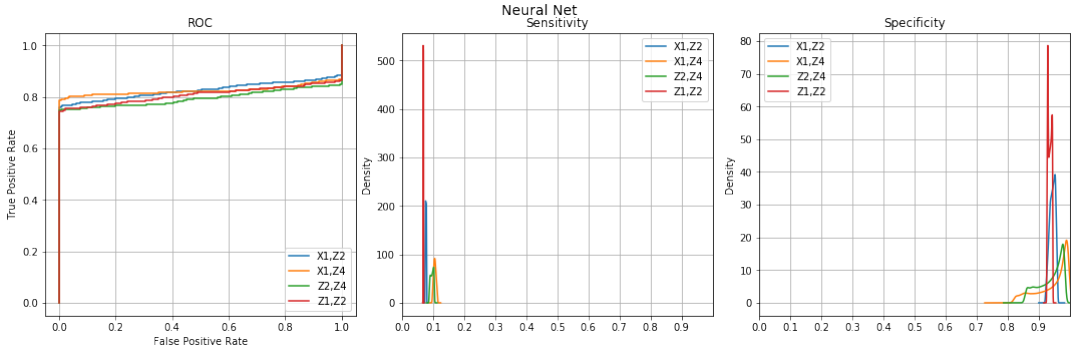



FIGURE 4.13: ROC curves and classifier performance indicator distributions for the maximum load case (ID458). For the selected boundary case, the classifiers are struggling to fit an adequate model. Only analyzing the ROC- curves would imply that the neural net seems to perform well. A closer look at the sensitivity and specificity reveals that this results from very high specificity values shadowing the very low sensitivity.

In general, we observe that there is no overall best way for classification regarding the exceptional cases, which copes with the "'No free Lunch'" theorem by Wolpert and Macready [97] and specifically holds for machine learning as well [98]. All classifications have low sensitivity, but the specificity is usually high. The classifiers vary from case to case and from one projection to the other. We can only merely explain the differences between the cases and the methodologies or projections by studying the performance indicators. Therefore the enhanced concept using the decision boundaries as depicted in Figure 4.8 improved the analysis task a lot. Using the Receiver-Operator Curves (ROC), we can compare different pairs of parameters. Nevertheless, for most cases, where all parameters have a personal impact, the dominating parameters are hard to identify with this visualization technique (see Figure 4.13). In general, the use of decision boundaries helps a lot understand the machine learning models' different behavior. We found out that the random forest achieves similar results to the neural network, while the KNN-approach fails for more complex cases. Further, we found an anomaly while testing the method for the real automotive body part. In the corresponding ambiguous load case, we spot a significant difference in the predictive quality (see Figure 4.14). We assume that the current labeling approach does not adequately cover the inherent asymmetric rigidity and distribution of the boundary conditions, requiring a solution at the preprocessing level, which further motivates the need for a good segmentation of the domain.

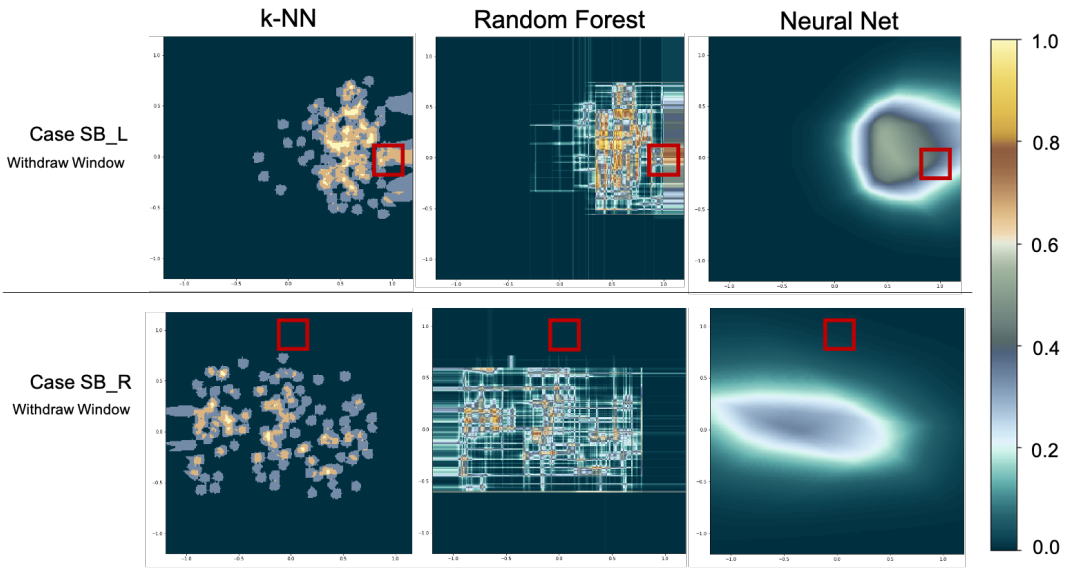



FIGURE 4.14: Different classification for symmetric boundary conditions for real car body part (sheet metal). When applying the methods to the sheet metal, we can observe large performance differences for symmetric boundary conditions. The left-side boundary condition (SB_L) is better predicted than the right-side (SB_R). The inherent asymmetric rigidity of the part is not captured by this simple approach. For more complex parts an a priori feature or cluster definition in the domain space would be needed.

## 4.5 Discussion

We have introduced a visual analytics pipeline to investigate the portions of information learned from the training data deformation field's differences to a target deformation field. This pipeline allowed us to determine the dominating parameters describing the target deformation field adequately and the correlation under these conditions. Further, we could show how different classifiers handled the exceptional load cases and revealed anomalies in real automotive part examples. Additionally, the concept of decision boundaries was beneficial as it summarizes each parameter's impact in its value range concerning any other. With this, we can combine the pipeline in a powerful analysis tool (Figure 4.15). In summary, this tool can help us investigate the training data's behavior on selected use cases in a globally summarizing scope. We can now use this tool to benchmark the expected performance increase of a segmentation method by investigating the classifiers' predictive quality increase. Therefore we expect to investigate significant changes in the predictive quality when we feed the system with a prior segmented portion

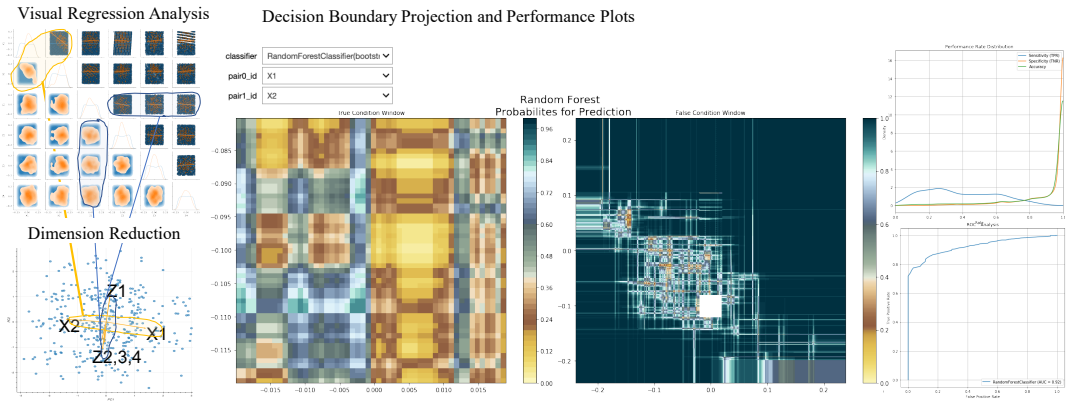



FIGURE 4.15: Analysis of multi-dimensional output of a classifier. The influence of parameter values on classifier output is studied visually to identify the most relevant parameters. Distribution plots make possible the interactive selection of appropriate values. Regression analysis is used to provide initial insight into dominant parameters. A matrix visualization shows correlations between pairs of parameters confirming the choices made for dimension/parameter reduction. Principal component analysis and hierarchical regression analysis provide additional means for estimation of parameter pair relevance. Parameter pairs are investigated using an enhanced confusion matrix with decision boundary contours and key performance indicator plots.

of the domain. Finding a good segmentation will be the thriving goal for our next chapter.

# Chapter 5

# Impact Maps and Topological Segmentation

The expressiveness of a globally summarizing distance metric is limited, and feature-based methods require profound domain knowledge and validation to ensure that all relevant properties are covered. So far, our predictive model is still over parameterized. To reduce the number of trainable parameters, we seek a reasonable and explainable segmentation of our domain. Additionally, we have developed a visual analysis toolkit that allows us to benchmark a segmentation before training the whole model. This way, we could increase the explainability of our approach a lot. The goal is to find a minimal number of segments, which can be represented by a summarizing value, such that our regression model's performance is still as high as for the overparameterized setting. This way, we achieve an explainable and reliable predictive model.
So far, we evaluated the prediction performance mainly in parameter space. It is necessary to analyze the impact of the simulation parameters in the domain space on the part's geometry to understand the prediction's learning process. At first, we investigate various approaches to map the impact of the parameters onto the part geometry. We start by calculating each cell's statistical properties along the training data set and visualize these via direct color mapping. We combine this technique with an interactive query, which allows us to steer the value ranges of a parameter and observe the changes. This technique already allows us to form clusters or regions of interest visually.
We then extend this view by investigating the pair-wise correlation of parameters. The idea is to transfer the results from Chapter 4 SPLOM Analysis into the domain space of the part geometry. We then use the resulting fields from the sensitivity analysis and apply a gradient or topology-based metrics. We then use these fields for topology-based segmentation and evaluate the found patterns. Finally, we show the application of a topology-based metric

for the usage in the analysis of a 3D vector field.

## 5.1 Sensitivity Analysis

"The importance of sensitivity analysis in engineering design cannot be overemphasized." Chen et al. [99]
Chen et al. put the importance of sensitivity analysis for understanding the parameters in a simulation. The same holds if we want to learn exactly these parameters in our machine learning model. For our machine learning model, we typically are not able to change the parameters to our liking. As such, we only can set up different queries on our training data for the sensitivity analysis. To analyze each parameter's impact, we are mapping each cell's statistical properties along with the training data in the part domain. Here we can distinguish between two views based on the desired analysis task.
First, we can operate only on the training data to evaluate the underlying simulation's general properties. Here we directly calculate the statistics on the deformation or stress field. It refers to the classical sensitivity analysis. Our primary focus is targeted towards the simulation's general behavior in this approach, thus revealing general behavior patterns.
Second, we can operate on the distance to a target deformation field. The idea here follows the inverse learning in Chapter 4, where we want to investigate the impact of the distance metrics in the domain space to learn about the possible features learned by our neural network. By picking specific target deformation fields, we can analyze the expected behavior of our learning model. Thus, our machine learning model should only learn those behaviors that we can observe here.

### 5.1.1 Statistical Properties

For our analysis we use the three basic statistical metrics: Mean, Median and Variance (plus the standard deviation).

$$mean = \overline{u} = \frac{1}{n}\sum_{i=0}^{n} u_i \tag{5.1}$$

$$median = \tilde{u} = \begin{cases} u_{\frac{n}{2}}\ ,\ if\, n\, odd \\ u_{\frac{n+1}{2}}\ else \end{cases} \tag{5.2}$$

$$variance = var(U) = \frac{1}{n}\sum_{i=0}^{n}(u_i - \overline{u})^2 = \frac{1}{n}\sum_{i=0}^{n} u_i^2 - \overline{u}^2 \tag{5.3}$$

$$standard\, deviation = \sqrt{var(U)} \tag{5.4}$$

The axis for our sums is the list of training data, where $n = \#Training\ Data$. For larger data sets it makes sense to re-index the training data, such that the block index for each simulation ensemble member represents the inner axis and the cells are the outer axis. This way the computation time of the statistics can make more use of parallelization capabilities. For our implementation in Paraview and TTK [100], we use this re-indexing strategy to allow a near real-time exploration in trade-off for exceeded memory usage.

### 5.1.2 Visualization Pipeline

The resulting visualization pipeline in Paraview uses the TTK extension for handling ensembles via the Cinema [101] approach. Depending on which analysis task we are following we first load the target deformation field for the distance computation or directly start with the *ttkCinemaReader*. We then use the *ttkCinemaQuery* to steer our input parameter ranges for the analysis. If needed the difference to the target deformation field for all ensemble members is computed. Afterwards we apply our statistics computation to the resulting *multiBlockDataSet*. Then we can directly visualize the statistical properties in the domain space.

#### Example Results from the Sensitivity Analysis

First, we look at some general behavior by investigating specific queries on the training data. In Table 5.1 we set up some queries for our investigation.

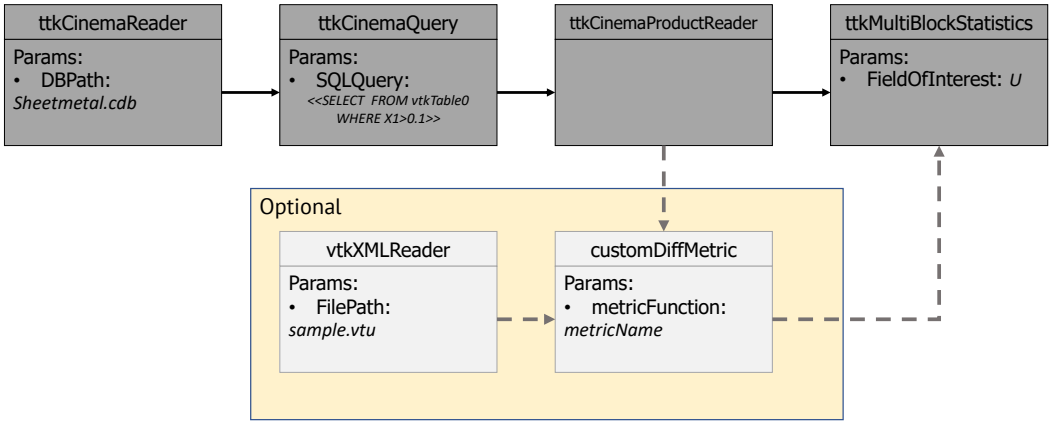


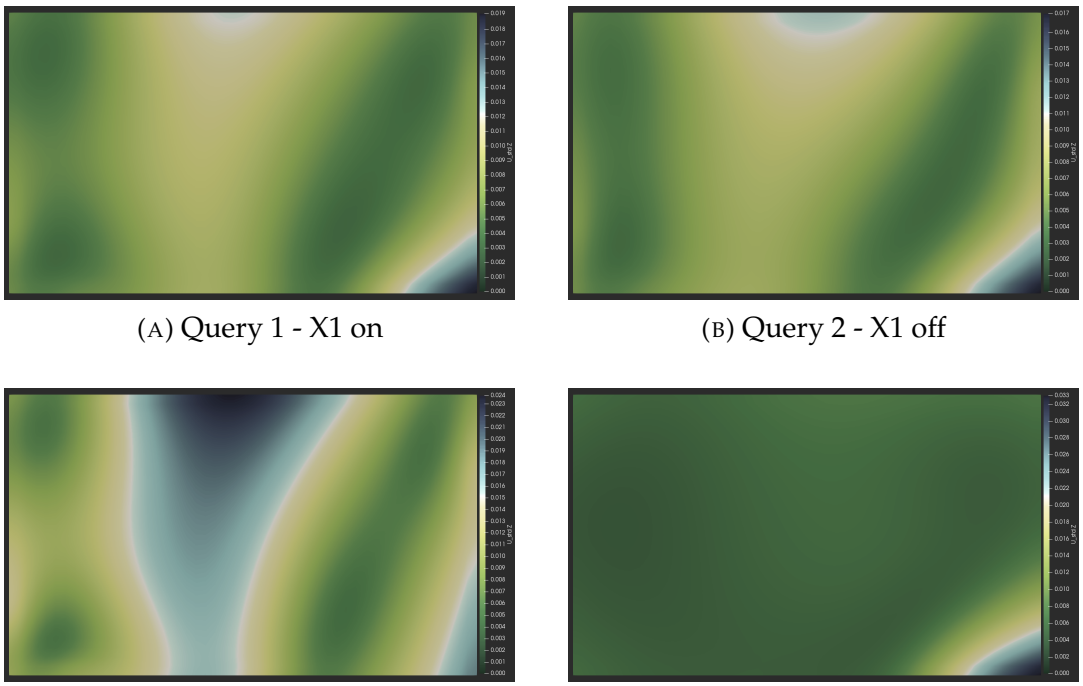

FIGURE 5.1: Visualization Pipeline using TTK/VTK. The cinema modules ares used to retrieve the data sets based on their parameters. This allows for a memory efficient access. On the resulting MultiBlock standard statistics or custom metrics are applied.

| QID | X1 | X2 | Z1 | Z2 | Z3 | Z4 | Note |
|---|---|---|---|---|---|---|---|
| 1 | \|v\|>0.1 | all | all | all | all | all | X1 above threshold |
| 2 | \|v\|<0.1 | all | all | all | all | all | X1 below threshold |
| 3 | >0.1 | all | all | all | all | all | X1 in positive direction |
| 4 | <-0.1 | all | all | all | all | all | X1 in negative direction |
| 5 | >0.1 | >0.1 | all | all | all | all | X1 AND X2 in positive direction |
| 6 | <-0.1 | >0.1 | all | all | all | all | X1 in negative direction AND X2 in positive direction |
| 7 | >0.1 | <-0.1 | all | all | all | all | X1 in positive direction AND X2 in negative direction |
| 8 | >0.1 | all | all | all | all | >0.1 | X1 AND Z4 in positive direction |

TABLE 5.1: Example Queries for the Sensitivity analysis.

First, we have an isolated look at the parameter X1. We then investigate the directivity of the parameter and then further investigate the interaction between X1 and X2 compared to X1 and Z4. In Figure 5.2 we see the results from varying the value ranges of X1 while keeping the rest untouched. We choose the z-component of the displacement field and the standard deviation here. For the query with the positive value range in X1 (Query ID 3), Figure 5.2c, we observe a significant difference in the resulting variation of the displacement field.

We can use this visualization pipeline to analyze combinations of parameter value ranges further (see Figure 5.3). Here we can observe that if X2 is also positive, the effect of X1 gets neutralized(Figure 5.3a). On the other hand, we can observe no effect (Figure 5.3b). Additionally, Z4 is also neutralizing



(A) Query 1 - X1 on

(B) Query 2 - X1 off

(C) Query 3 - X1 positive

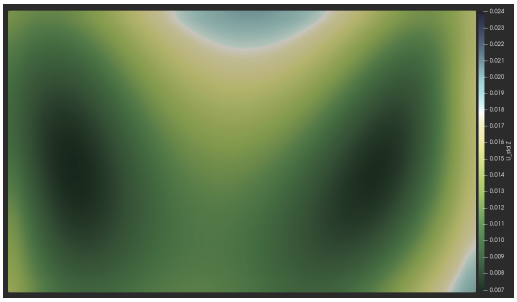
(D) Query 4 - X1 negative

FIGURE 5.2: Results from the sensitivity analysis. Showing the standard deviation in Z-Direction for the displacement field. For positive X1 values (C), the displacement field variation is increasing significantly.

the effect of X1, but more weakly. From the queries' visual analysis, we can deduce that there is a negative correlation between X1 and X2, and X1 and Z4 for positive X1. If we want to quantify this correlation, we would like to have a visualization that resembles the scatter plot matrix from Figure 4.6.

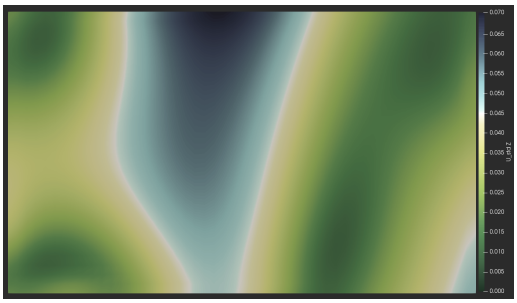
### 5.1.3 Sensitivity Matrix View

The scatter plot matrix is a powerful tool in visual analytics to compare pair-wise correlations of attributes. In Chapter 4, we have seen how they can be used to explain the strong correlation between X1 and X2. We now have geometry represented by cells and their topology in the domain space rather than a single data point. Thus a direct mapping of the concept is limited by the number of possible marks and channels available. We have to find a trade-off between keeping the domain information and providing an overview of pair-wise cross-correlation of the parameter values. To solve this, we reinterpret the marks in the scatter plot with color-coded domain visualization. As we are even more limited in space, we cannot use such an image for every parameter combination. Instead, we reuse the concept from

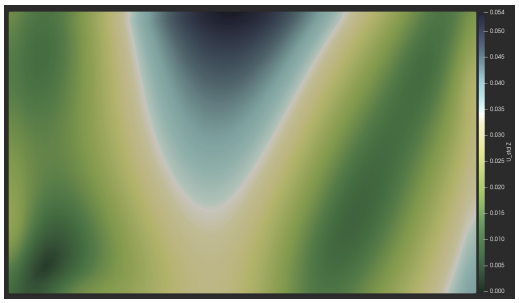


(A) Query 5 - X1 pos AND X2 pos

(B) Query 6 - X1 neg AND X2 pos



(C) Query 7 - X1 pos AND X2 neg



(D) Query 8 - X1 pos AND Z4 pos

FIGURE 5.3: Results from the sensitivity analysis with two-parameter variations. Showing the standard deviation in Z-Direction for the displacement field. The impact of X1 in the positive direction is neutralized when X2 is also positive (A). The impact of X2, when X1 is negative (B) is vanishing (compared to Figure 5.2d )

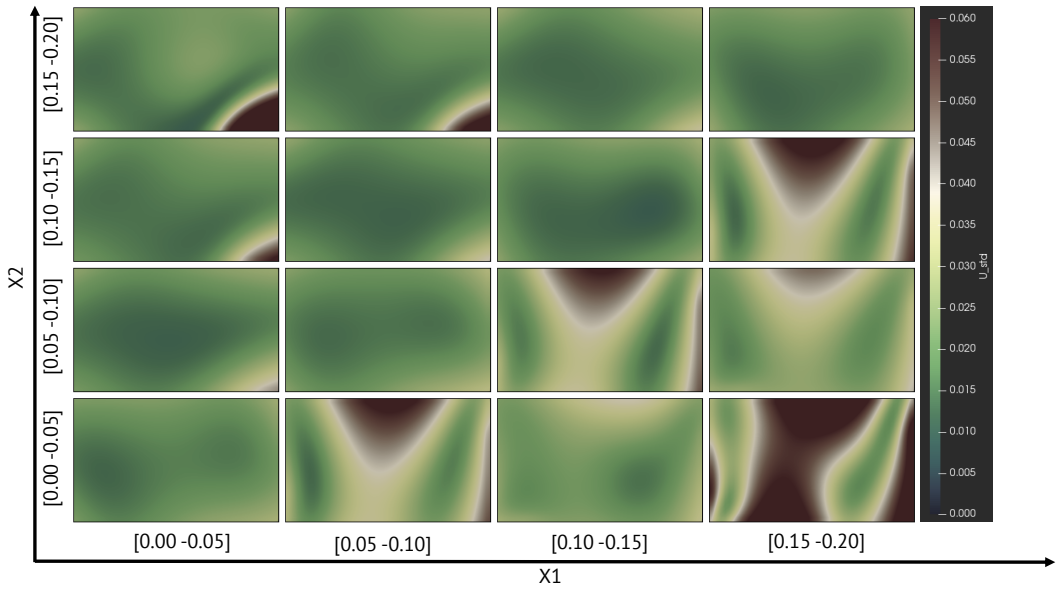
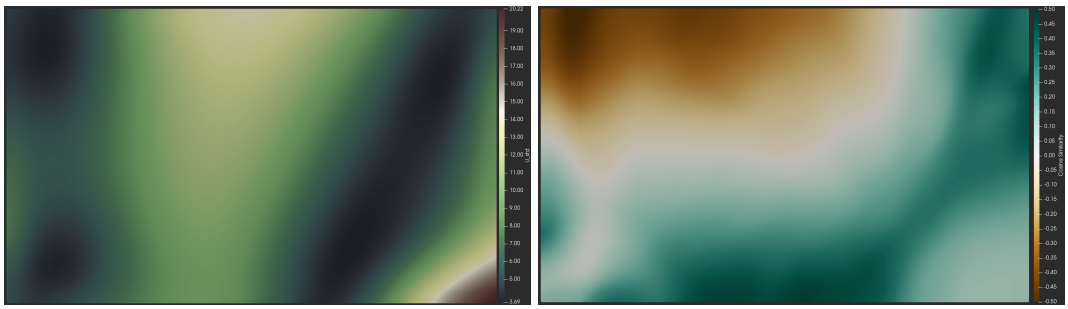



FIGURE 5.4: SPLOM for Sensitivity Analysis in Domain Space. We are reinterpreting the concept of a Scatterplot-matrix by visualizing pair-wise range queries in a matrix view. Here we see one element of the complete SPLOM for a selected pair. This visualization allows us to overview the whole pair-wise correlations in the domain of the geometry. Here we visualized the variance of the resulting query in each image. We can observe that with increasing X1, the variance is also increasing. If X2 is also increased, the resulting variance induced by X1 gets neutralized. Hence there is a positive correlation between X1 and X2.

the sensitivity analysis and use them as representatives for a range of parameters on the axis. We can steer the level of detail of our view by the number of ranges we define for our queries. In Figure 5.4, e.g., we chose four subdivisions of the positive parameter range and ordered the resulting visualizations for these from the sensitivity analysis in a matrix view. The two axes thus refer to the two queries on the respective parameters. This visualization then corresponds to one entry in the classical scatterplot matrix. We can now further recombine these to have a complete representation for all parameters. Technically this view is achieved by parameterizing the sensitivity analysis form above, such that we get a resulting color-coded visual representation of the chosen metric. In our case, we used the standard deviation on the Z-component of the deformation field. Afterward, we define the number of subdivisions that we want to investigate, render the visualizations, and store them in an image database with the corresponding parameter value ranges. We then define a matrix layout with proper labels and map the images from

our database to it. The transformation concept is summarized in Figure 5.5. Now that we have a compact visualization, we can use it as a general tool to analyze our data set based on various metrics (see Figure 5.6). In the next section, we will introduce some metrics used while studying the metal sheet example's behavior. The basic idea still is to find good segmentation of the domain for better training data.


FIGURE 5.5: Transformation Concept for a SPLOM with image in-domain visualizations. We start with defining queries that uniformly sample the parameter values in ranges. On those queries, we then compute general statistics or other comparative metrics. The following visualization pipeline is set up, and the rendered images are stored in a database with appropriate meta-data of the query. This meta-data is then used to layout the images properly, such that we achieve the look of a classical SPLOM.



(A) Displacement standard deviation (B) Cosine similarity of the displacement

FIGURE 5.6: Investigating different summarizing metrics. Besides using statistical metrics such as variance or standard deviation, we can also observe summarizing metrics like the one investigated for the loss functions in Section 3.3.1. In general one is free to use any metric, which is suitable for comparing alongside different parameter range queries.

## 5.2 Gradient and Topology-based Metrics

The general statistics introduced above lack the ability to incorporate topological information. As such, the achieved entropy is limited to the information in each paired observation. In simulations and measurements alike, each observation is coupled to a position in space or a geometry element. If

we only interpret them as a 1-dimensional series of observations, we drop the topological information, describing the neighboring connection of observations in space. As topological metrics, we define those that can use the neighboring information.
The idea is to use these metrics to describe the topology of the resulting fields, e.g., the ones from Figure 5.4. We then use these to define areas of interest, for which we can define a summarizing value and reuse the evaluation concept from Chapter 4, but on a specified area.
We start with the most common and straight forward metric, the k-Means. This method is widely used in many applications and form the baseline for our investigations. Further, we will look at more specialized metrics for vector fields, mainly using the neighboring cells' gradient as topological information. Those are the **vorticity** (curl), **strain** (normal and shear), the **okubo-weiss-criterion** and the **Angular Direction Changing Rate**, which we developed for filtering highly turbulent regions in earth's magnetic field (see Section 5.4.

### 5.2.1 k-Means Distance

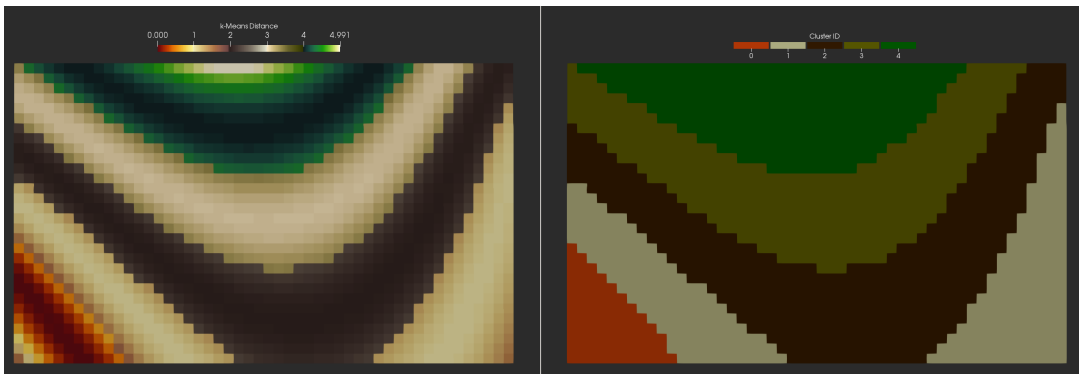



FIGURE 5.7: k-means Distance and Clustering. An example of using the k-means method for clustering and summarizing metric for the comparison task in the sensitivity analysis. The parabolic shape induces that polynomials of degree two can describe the found clusters.

The **k-means distance** is the metric that directly follows after applying k-means clustering to the data set. The k-means clustering aims to partition the observations into $k$ groups to minimize the within variance. The k-means distance is the resulting distance of each point to its cluster center.

**Fundamental Idea**

The k-means distance is a side-product from the k-means clustering, which minimizes the variance of distances to a cluster $i \in k$. This metric also produces clusters of distances with sharp borders if we use the information directly. In Figure 5.7 we show the results of such a distance metric with $k = 5$. In this representation, we used a 5 - wave color-map, to visualize each cluster's distances in a separate color scheme. The first clusters distances are between $[0, 1[$, the second between $[1, 2[$ and so on. To get a smooth representation like this over the whole domain, we add the cluster-ID as a constant. For increasing $k$, this metric's visualization gets increasingly complex, as we have to choose more and more different color maps.

**Mathematical Description**

The k-means clustering is an optimization of the following function:

$$\min_{S} \sum_{i=0}^{k} \sum_{x \in S_i} ||x - \mu_i||^2 \tag{5.5}$$

Here $S$ is the set of points to be clustered and $S_i$ is the $i$-th partition of the set with mean value $\mu_i$. After solving the optimization problem and finding the cluster centers for each partition $i$, the k-means distance of each point is the same as the squared variance to the mean value of its closest center. The first published algorithm is from Lloyd in 1982 [102], although it was developed in 1956. Nowadays, there exist fast and efficient methods for a couple of years [103].

$$\Delta_{kmeans} = ||x - \mu_{closest(x)}||^2 \tag{5.6}$$

To separate the clusters in the distance metric we use the root of the distances and add the cluster ID as a constant. If the value range is already normalized to the range $[0, 1]$ this results in a smooth distance metric for each cluster $i$ in the range of $[i, i + 1[$. In Figure 5.7 we applied a $5 - wave$ color-map to visualize the distance metric with a distinct color-map for each cluster.

$$\overline{\Delta}_{kmeans} = \sqrt{\Delta_{k-means}} + closest(i) \tag{5.7}$$

### 5.2.2 k-Means Clustering

The k-means clustering is solving an optimization problem. The goal is to partition the n observations (or cells) in $k$ sets, such that the sum of squares

(variance) is minimal. The mathematical description is denoted in Equation.5.5. Instead of the resulting distance to the centroid, we are now interested in the partition index. The segmentation is then based on the cells with the same partition index. In comparison to the other metrics discussed here. The k-means approach directly segments the data set.

### 5.2.3 Vorticity and Strain-Rate-Tensor

The vorticity and the strain-rate-tensor are metrics from the vector field analysis that describe a vector field's behavior concerning its direct neighborhood. Therefore both rely on partial derivatives. In a discrete vector field, these derivatives are derived via incorporating the directly neighboring cells. Therefore both metrics are classified as topological metrics. Compared to the k-means distance and the proposed angular direction changing rate, these metrics' topological scope is fixed to the direct neighborhood. Thus for increasing densely sampled discretizations, the topological information vanishes.

#### Fundamental Idea

The vorticity and divergence are forming the baseline derivatives for vector field metrics. The strain adds directivity about the element to this derivative. A combination of vorticity and strain creates the very popular Okubo-Weiss-Criterion for eddy detection. To better understand this criterion, we, therefore, look at each part of it separately. For the strain, we, therefore, look at the normal and shear strain.

#### Mathematical Description

The vorticity is defined as:

$$\overrightarrow{\omega} = \nabla \times \overrightarrow{v} \tag{5.8}$$

The strain is defined on the gradient of $v$ and yields to the tensor:

$$L = \frac{\partial v}{\partial x} \tag{5.9}$$

Here $x$ refers to the local frame element in a discrete vector field. We can now decompose this tensor in a symmetric and asymmetric part:

$$\mathbf{L} = \frac{1}{2}(\mathbf{L} + \mathbf{L}^T) + \frac{1}{2}(\mathbf{L} - \mathbf{L}^T) \tag{5.10}$$

$$\mathbf{L} = \mathbf{D} + \mathbf{W} \tag{5.11}$$

$\mathbf{D}$ is the deformation tensor or strain-rate tensor, while $\mathbf{W}$ is the spin tensor. From the deformation tensor, we then derive the normal and shear strain.

$$\mathbf{D} = \begin{bmatrix} \frac{\partial v_x}{\partial x} & \frac{1}{2}\left(\frac{\partial v_x}{\partial y} + \frac{\partial v_y}{\partial x}\right) & \frac{1}{2}\left(\frac{\partial v_x}{\partial z} + \frac{\partial v_z}{\partial x}\right) \\ \frac{1}{2}\left(\frac{\partial v_y}{\partial x} + \frac{\partial v_x}{\partial y}\right) & \frac{\partial v_y}{\partial y} & \frac{1}{2}\left(\frac{\partial v_y}{\partial z} + \frac{\partial v_z}{\partial y}\right) \\ \frac{1}{2}\left(\frac{\partial v_z}{\partial x} + \frac{\partial v_x}{\partial z}\right) & \frac{1}{2}\left(\frac{\partial v_z}{\partial y} + \frac{\partial v_y}{\partial z}\right) & \frac{\partial v_z}{\partial z} \end{bmatrix} \tag{5.12}$$

The normal strain $s_n$ are then the diagonal parts of the tensor:

$$s_n = \frac{\partial v_x}{\partial x} - \frac{\partial v_y}{\partial y} - \frac{\partial v_z}{\partial y} \tag{5.13}$$

The shear strain $\gamma_{xy}$ in $x, y$ direction is defined as :

$$\gamma_{xy} = \frac{\partial v_y}{\partial x} + \frac{\partial v_x}{\partial y} \tag{5.14}$$

Thus we can reformulate the Deformation tensor $\mathbf{D}$ as:

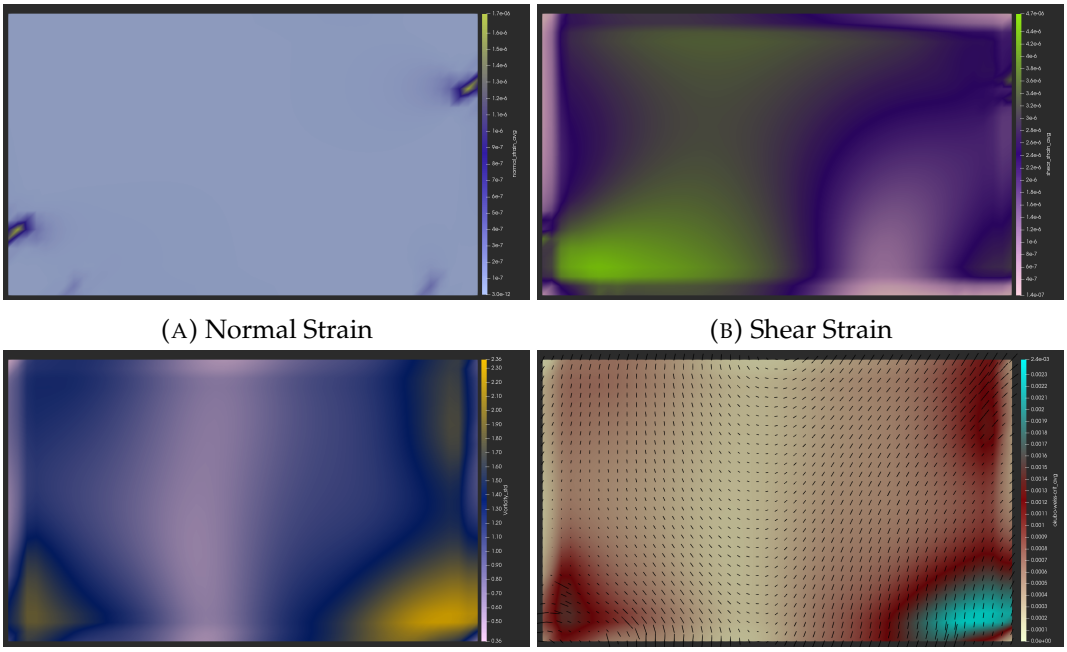
$$\mathbf{D} = \begin{bmatrix} \frac{\partial v_x}{\partial x} & \frac{1}{2}\gamma_{xy} & \frac{1}{2}\gamma_{xz} \\ \frac{1}{2}\gamma_{yx} & \frac{\partial v_y}{\partial y} & \frac{1}{2}\gamma_{yz} \\ \frac{1}{2}\gamma_{zx} & \frac{1}{2}\gamma_{zy} & \frac{\partial v_z}{\partial z} \end{bmatrix} \tag{5.15}$$

By comparing the two formulations of $\mathbf{D}$ it follows that: $\gamma_{xy} = \gamma_{yx}$, $\gamma_{yz} = \gamma_{zy}$ and $\gamma_{xz} = \gamma_{zx}$. As such we can write the shear strain $s_s$ as:

$$s_s = \begin{pmatrix} \gamma_{xy} \\ \gamma_{yz} \\ \gamma_{zx} \end{pmatrix} \tag{5.16}$$

**Example**

In Figure 5.8 we can get an idea of the metrics behavior. In the top row, the normal and shear strain is depicted. In the bottom row, we can see the vorticity magnitude. We observe two peak values for the normal strain that perfectly match the external load boundary conditions X1 and X2. We observe a



(A) Normal Strain (B) Shear Strain

(C) Vorticity (D) Okubo-weiss-criterion with field lines

FIGURE 5.8: Assembly of the Okubo-weiss-criterion [104], [105]. The criterion is assembled by the squared sum of the three input fields: (A),(B),(C) (see Equation 5.17. Here we can directly see, that the criterion is mainly influenced by the shear strain and vorticity. The two peak spots of the normal strain are no longer present in the okubo-weiss criterion. Additionally the direction of the vorticity field is indicated with small line glyphs. The brigther parts are more influenced by the vorticity, while the turquoise parts are mainly influenced by the strain components.

.

similar distribution pattern for the shear strain and the vorticity magnitude, but with different scales. In general, all gradient-based methods suffer from numerical instabilities near the domain boundaries.

### 5.2.4 Okubo-Weiss-Criterion

The Okubo-Weiss criterion is named after their founders Okubo [104] and Weiss [105] and gained prominence in the detection of eddies in ocean water movements[106]–[110].

#### Fundamental Idea

The Okubo-Weiss criterion was initially developed for eddy detection in oceanic movements. Nowadays, methods are usually building upon this criterion as their basis. The fundamental idea is to compare the velocity gradient for its

relative movement. Therefore the vector field is decomposed in the normal strain (pointing in the normal direction of the plane), the shear strain (relative movement in the plane), and the relative vorticity (relative circular momentum in the plane). The criterion is then defined as the difference between the strain and vorticity components as depicted in Equation 5.17. For eddy detection, a threshold is set on a negative value of $W$. It means the point at which the circular movement in the plane is dominating the relative shear movement.

#### Mathematical Description

The Okubbo-Weiss criterion is defined as:

$$W = s_n^2 + s_s^2 - \omega^2 \tag{5.17}$$

Here $\omega$ is the relative vorticity in the plane defined in $s_s$.

#### Example

Figure 5.8d shows an example of the Okubo-Weiss criterion on the displacement difference. We can observe that the vorticity and strain in the center axis are in balance, which means that most of the displacement here can be described via the strain components.

#### Implications

So far, we have investigated some primary gradient-based metrics on the resulting difference field. Incorporating a strain definition is especially suitable for our application in linear elastic load simulations. It enables us to see if we have to allow our neural network regression to incorporate the distinction between strain and vorticity into its learned model. This would lead to an increased combinatorial complexity as we would have to add topological information to our input model. Further, we are now able to determine if the gradient-based methods reveal any new patterns that were not already captured by one of the more basic ones depicted in Figure 5.8. Nonetheless, those metrics gain even more importance for more turbulent vector fields and mostly 3-dimensional ones, but they are hard to interpret in a 3D-domain. We took the best from the two worlds and developed a turbulence measure applicable and interpretable in three-dimensional space, but with less computational effort.

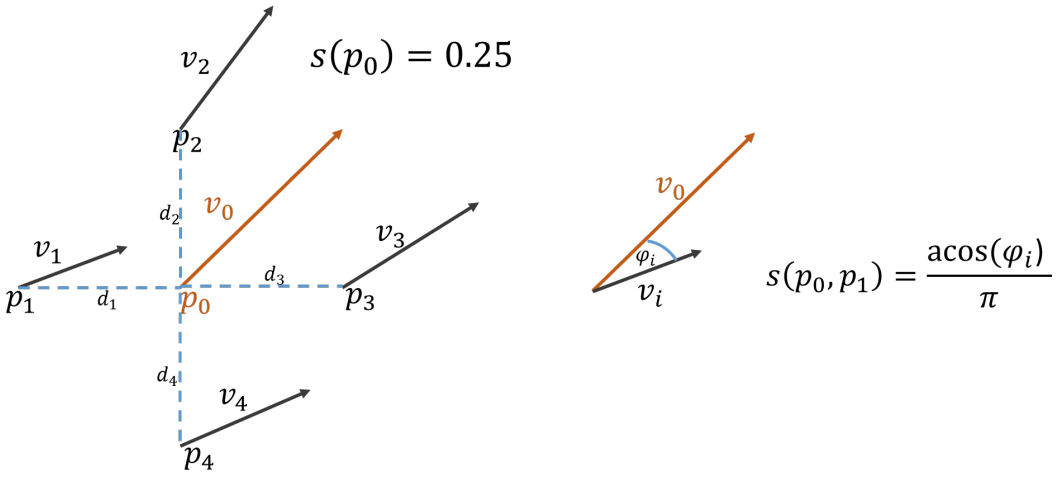
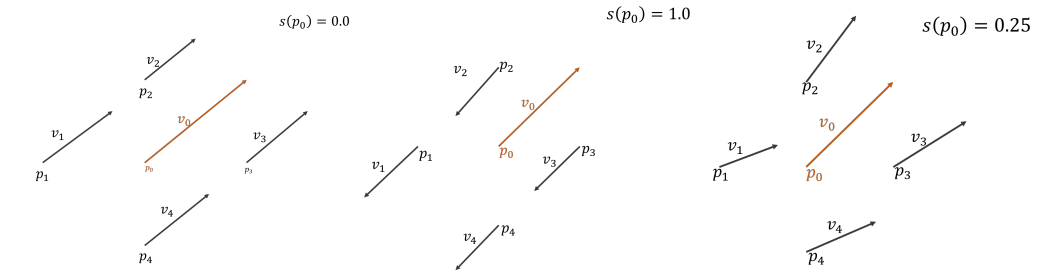



FIGURE 5.9: The idea underlying our turbulence measure is comparing each point and its respective vector quantity with its neighborhood. As a similarity measure the angle between vectors is used. Normalizing them to the interval [0,1] is done to manipulate the color map. In the case of data defined on unstructured grids, distances between points are used for weighting similarity.

### 5.2.5 Angular Direction Changing Rate

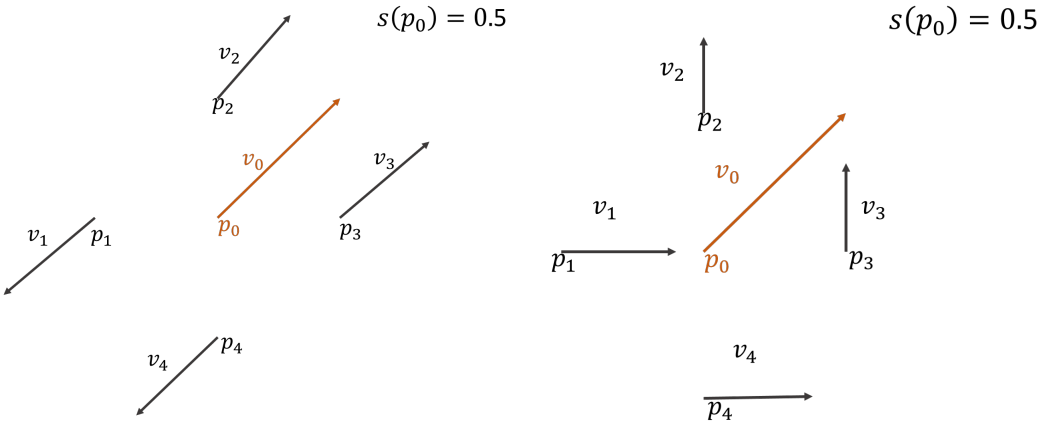
#### Fundamental Idea

Our early visualization attempts of the geodynamo's inner core A.2 behavior suffered from the fact that we struggled with a meaningful and effective way to define and visualize "structures of interest" to the scientist, especially structures indicative of turbulent behavior. Concerning the magnetic field data of primary interest to study the earth's geodynamo, typical features seen in liquid or gas flow fields are not present in the magnetic field data of concern here. Thus, standard vector field visualization methods cannot be applied directly to geodynamo data since the physical meaning is different from liquid or gas flow fields. Therefore, we decided to devise a new approach to analyze geodynamo data. Instead of extracting turbulent regions in the data set or finding single encapsulated drivers for global behavior, we analyze turbulence and its inherent structure itself. A more specific definition of the term turbulence is needed for our data. To highlight regions in the data that exhibit turbulent behavior, we devised a method that perceptually emphasizes such regions in volume visualizations. We found out that the magnetic field vectors' directions need to diverge above a certain amount compared to others in a local neighborhood to indicate locally turbulent behavior. We define turbulence as follows:




(A) No turbulence, maximum similarity (B) Maximum turbulence, minimal similarity (C) Low turbulence, high similarity

FIGURE 5.10: Similarity-based turbulence shown in a plane with four equidistant neighbors. In (a) all vectors point in the same direction, implying that the similarity in direction is maximal and turbulence is minimal. Image (b) shows the opposite case that can only happen in boundary regions or when singularities are encountered. Image (c) depicts the average case. Turbulence needs to be interpreted differently for each data set: In the case of the geodynamo this would be classified as a no-turbulence case.




(A) Border case (B) Normal behavior

FIGURE 5.11: Special case, where the calculated similarity value is the same, but the interpretation of turbulence is different. Image (a) depicts a sharp edge or border in the vector field. Image (b) shows a situation that can be classified as either normal-turbulence or eddy behavior. In the case of magnetic fields, situation (a) cannot happen unless a singularity in the data set is hit. The situation shown in (b) could be improved for interpretation by using a larger neighborhood. Our experiments have shown that one should use at least 10 neighbors in the 3D spatial domain.

**Definition.** The turbulence of a vector field $V$ is described by the divergence of a specific vector quantity $\overrightarrow{v}(p_0)$ at a point $p_0$ in its spatial domain relative to the vector quantities in a local neighborhood defined by a set of points $P_{nb}$ around $p_0$.

Following this definition, a proper metric is needed to measure divergence. Our first simple approach used the difference in the vectors' angles in a spatial neighborhood and applied weights to them based on distances. Figure 5.9 and Figure 5.10 show how this definition can describe divergence of vectors in a local region. Due to its ambiguity, non-uniqueness, there exist multiple cases that can produce the same or nearly the same numerical value, as shown in Figure 5.11. Despite the ambiguity of this measure, it still produces valuable results, and the special cases tend to occur only occasionally or "exist only in theory." A more in-depth investigation is necessary to understand this behavior in detail.

### Mathematical Description

We now introduce the essential and needed formulas underlying our approach. They are:

$$t\left(p_0\right) = \sum_{i=1}^{n} \frac{d_i}{d_{sum}} \cdot \frac{\arccos\left(\varphi_i\right)}{\pi} \tag{5.18}$$

$$\cos\left(\varphi_i\right) = \frac{\langle \overrightarrow{v_0}, \overrightarrow{v_i} \rangle}{||\overrightarrow{v_0}||\,||\overrightarrow{v_i}||} \tag{5.19}$$

$$R_n\left(x\right) = \frac{e^{\lambda \cdot x} - 1}{e^{\lambda} - 1} \tag{5.20}$$

Here, $p_0$ is the position with vector value $\overrightarrow{v_0}$, for which the turbulence measure is calculated. The variable $d_i$ is the Euclidean distance between the points $p_0$ and $p_i$ with vector value $\overrightarrow{v_i}$. Depending on the specific data distribution and/or grid type, other distance metrics should be considered as well. The value of $d_{sum}$ is the accumulated sum of all distances to the points in the local neighborhood. The notation $\langle \cdot, \cdot \rangle$ refers to the standard scalar product of two vectors. Equation 5.20 can be used to normalize the turbulence measure such that low-angle differences are weighted less than linearly and high-angle differences more than linearly, to emphasize more drastically the turbulent regions of actual interest. We used this normalization step to guide transfer function design for volume rendering.

**Run-time Behavior, Scalability and Complexity**

The theoretical run-time of the proposed method is given as follows:

$$T(n) = n \cdot [T_{search}(m, n) + T_{turb}(m)] \tag{5.21}$$

$$T(n) = n \cdot [O(m \cdot \log(n)) + O(m)], \tag{5.22}$$

where $n$ is the number of vertices in the data set and $m$ the number of vertices in a local neighborhood. The method highly depends on an efficient data structure that supports fast access to local neighborhood vertices. We then decided to use a KD- tree as a data structure to access the required neighboring vertices efficiently. Our method is implemented as an in situ method. By fixing the number of available neighboring vertices and storing them together with each vertex (or at least pointers), one must only calculate the turbulence measure. The geodynamo data set is defined as a topologically structured-hexahedral grid. The set of local neighborhood vertices is therefore implicitly defined by the index of a vertex. This fact reduces computation time substantially for our method and thus only adds little overhead computation time to the simulation itself's computations. Suppose an in situ approach was not feasible, e.g., due to a much more complicated underlying grid topology. In that case, one could split the data domain into sub-domains roughly equal to the number of available computer nodes and perform parallel processing. Following such a strategy, a sub-domain should be larger than the used local neighborhood; otherwise, the communication overhead resulting when performing computations at sub-domain borders would eliminate the computational gain.

## 5.3 Topology-based Segmentation

So far, we used the sensitivity analysis (Section 5.1) and extended it with gradient and topology-based metrics (Section 5.2). The definition of areas of interest was then purely visually, based on the resulting fields. In this section, we want to investigate a more automated approach by directly using topology-based segmentation methods. These methods directly aim to segment the domain into subsets. As such, we can directly use the result to define a segmentation. With the increasing complexity of a part or simulation domain, the model complexity needed to learn a prediction model, like the one introduced in Chapter 2, is also increasing. As a consequence, we

introduce more model parameters, which complicate the interpretability of the learned model. For complex parts, a segmentation before training a prediction model helps us achieve good results while maintaining the model's interpretability. We will investigate four approaches to segment a displacement field ensemble based on its variance. The variance field is the result of a sensitivity analysis of the whole training data.

### 5.3.1 Morse Theory

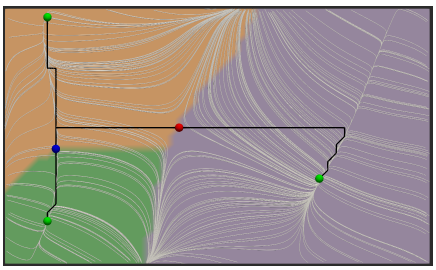
The Morse Theory [111] investigates the topology of a surface employing its critical points. The Morse-Complex partitions the manifold based on the behavior of the gradient. The gradient $\nabla$ of a function $f$ defines a smooth vector field on the manifold, where zeros define the critical points. On this vector field, we can now define integral lines.

$$\frac{\delta}{\delta t} l(t) = \nabla f(l(t)) \tag{5.23}$$

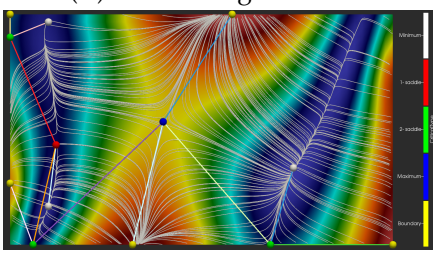
In such a gradient vector field, the integral lines are monotonic, which means that an integral line's start and the endpoint is never the same. This allows us to define an ascending/descending manifold as follows:

- Let $p$ be a critical point of the Manifold $M$. Then the **ascending** manifold of $p$ is the set of points, which means that integral line origin is $p$.
- the **descending** manifold of $p$ is the set of points, which integral lines destination is $p$.

We can now use these definitions to define a segmentation of our displacement field. In Figure 5.12a we visualized the critical points together with the resulting descending manifold and the integral lines of the gradient field. In Figure 5.12b we used the input scalar field as the background visualization to reconnect the results to the initial domain. If we compare the input field with the resulting three segmentations, we are missing some information. The ascending manifold results in only one segment, as the maximas are on the boundary of our domain.



(A) Descending Manifold



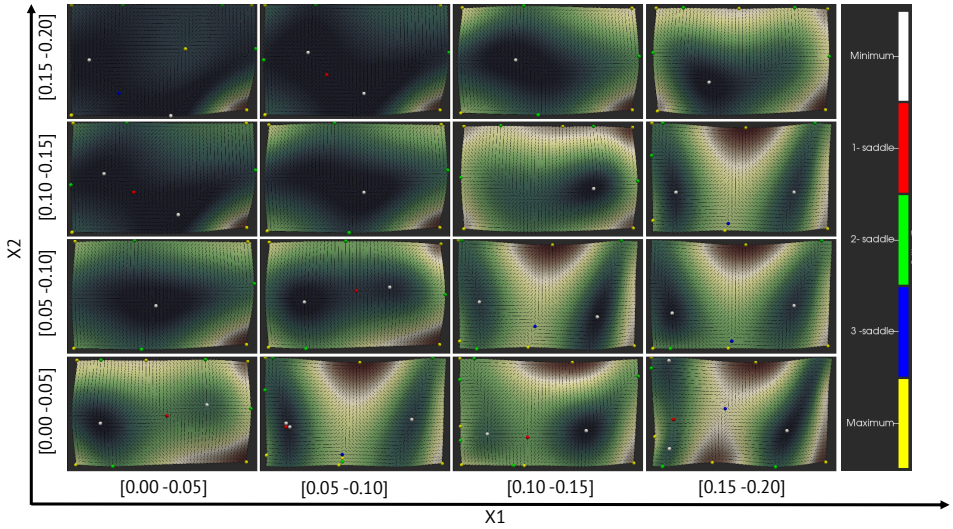
(B) Reeb graph on Scalar Field

FIGURE 5.12: Results from applying the Morse-Theory to a Scalar Field.

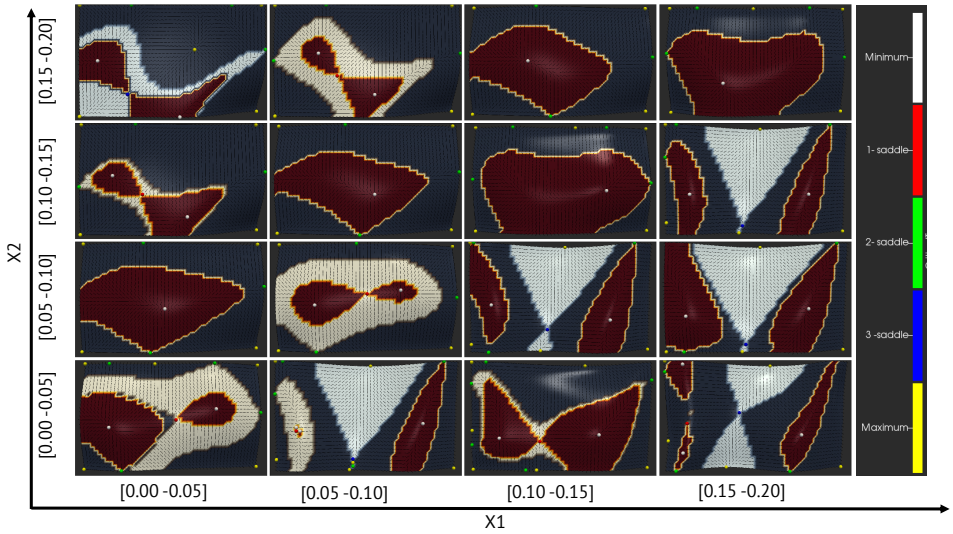
### 5.3.2 Topological Analysis of the Sheet Metal Ensemble

We can now use these analysis steps to enhance our initial sensitivity analysis further. With the concept of the matrix layout, we can now directly compare the field's variance concerning different parameter combinations and the topological properties of it.

In Figure 5.13 we showed the comparison of the variance of the field (top) and its topological segmentation (bottom) using Morse-Smale complexes. Chapter 4 we already found out that the parameters X1 and X2 have a strong contradicting behavior. We are now varying the parameter ranges of both and analyze the evolution of the variance. The results are then arranged in a matrix view, similar to the scatter plot matrix concept. We can observe that with increasing X2, the variance is decreasing (top rows), and with increasing X1, the variance is increasing (right columns). If we now focus only on the most right column where X1 $> 0.15$, we observe that with increasing X2, the overall variance is decreasing. In the topological analysis, we can further see that the number of regions decreases from six to two regions. If we now assume a direct linear correlation, the pattern should reoccur on the matrix's diagonal as well. This is only the case for the two higher value ranges where X1 and X2 are greater than $0.1$. However, in general, we can still observe similar patterns in the resulting topological regions on the matrix's diagonal axis, which confirms our initial result that X1 and X2 are strongly correlated. Now we can further specify this correlation. For $X1\,, X2 > 0.1$ we have a




(A) Variance Field Correlations




(B) Morse-Smale Complex on the Variance Field

FIGURE 5.13: Combined Sensitivity Analysis and Topological Segmentation. We can directly compare the input scalar field correlations with the resulting topological segmentation with the matrix view. The segmentation patterns reveal a strong positive correlation between X1 and X2.

near-linear correlation in the whole domain, while for $0.0 < X1\,, X2 < 0.1$ the influence of X2 on X1 is decreasing.
The other result from the machine learning model study in Chapter 3 was that 10 - 20 neurons in the layer are enough to find a linear model, which maps the deformation field onto the six input parameters. With our topological analysis, we found no more than seven topological relevant regions that describe our ensemble's variance in the eight parameter ranges (four in the positive direction and four in the negative direction). Thus we can conclude that the 10 - 20 neurons are also covering the topological properties of the ensemble's variance.

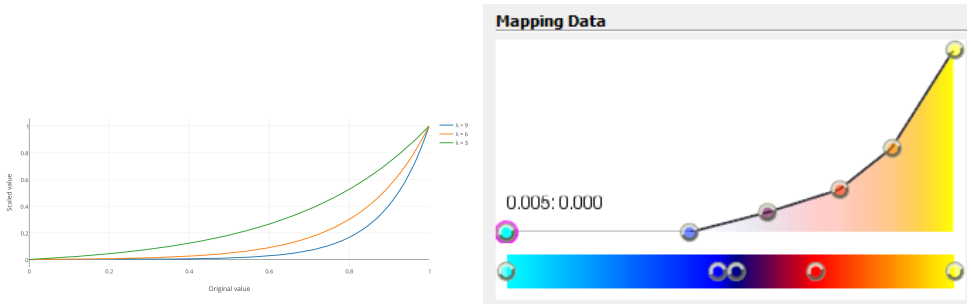
## 5.4 Application - Measuring Turbulence in a Magnetic Field via Local Vector Field Similarity

### 5.4.1 Visualization

In order to visualize a turbulence field, typical scalar field visualization methods can be used. For smaller and structured data sets, volume rendering can be used in all analysis and visualization tasks. In most cases, this gives the best perception of the underlying structure of the turbulence field. The remaining visualization parameter, therefore, is the transfer function. For many application scientists, this is usually hard to achieve without proper guidance. In this way, the following transfer function was developed with the application scientists:

$$f\left(x\right) = \frac{e^{\lambda \times} - 1}{e^{\lambda} - 1}, x \in [0, 1] \tag{5.24}$$

Where $x$ is the result from the proposed turbulence metric normalized to a range from $0$ to $1$ and $\lambda$ is a parameter to steer at which point the higher turbulence value should affect the opacity more than linear. For our results we were using $\lambda = 6.5$. However, this function allows only the first guidance. To achieve better results, one needs to manually adapt the points opacity and use the proposed function for interpolation.
For large and unstructured data sets, like the one dealt with here, the use of volume rendering through all steps is not affordable. Even with decent graphics computation capacities, it is impossible to achieve nearly real-time results for the volume rendering. The most challenging factor here is the




(A) Exemplary behavior of the transfer function for different $\lambda$ values.

(B) Example of a transfer function for color and opacity used for geodynamo visualization.

FIGURE 5.14: The chosen transfer function for different parameter values. A manual adjustment is needed to achieve the best results in terms of supporting a scientist's specific data exploration purposes. For the hot-cold color map, the values need to be shifted accordingly.

unstructured grid of the underlying data, which slows down the computation massively. Therefore multi-layered iso-surfaces or cells only showing up when a certain threshold is reached are more suitable for the first analysis tasks. Nevertheless, with the near future task of visualizing unsteady data, there is a need to deal with this problem efficiently. One wants to create the visual results for each time step with minimal user interaction. The next big challenge for the volume rendering in our case would be to either speed up the computation for unstructured grids in general or find a suitable method to transfer the unstructured into a structured grid with a low error.
One future goal is the visualization of Poincare maps. The driving challenge here is to find a proper seeding strategy to find field lines with more than one periodicity.

**Theorem.** Given a plane $E_0$ and an integral curve in the vector field starting at point $p_0$, the curve has to cross the plane $E_0$ after a limited number of integration steps in order to be considered for a Poincare map.

This theorem leads to three major types of magnetic field lines:

1. The magnetic field line starts at a given point $p_0$, and there is **no** plane $E$ which the line is crossing more than ones.

2. The magnetic field line starts at a given point $p_0$, and there is at least one plane $E$ which the line is crossing more than ones.

3. The magnetic field line is starting at a given point $p_0$ and under a given number of maximal integration steps, there is **no** plane $E$ the line is crossing more the ones.

For Poincare maps, usually, a slice through the whole domain is used. To have a complete map with a given resolution of $n$ points, one needs to integrate $n$ field lines. Considering our example data set, one slice with the same resolution as the data set would need about 1000 integration lines. To capture the maximum amount of periodicity, one needs a significant large integration time, which leads to high computational costs. Thus knowing in advance which areas need a higher integration time than others is useful to speed up the computation time. For example, it is of major interest for geophysics to find those integration lines leaving towards the poles and the domain after a certain number of orbits. Therefore it is sufficient to split the domain into areas for leaving and staying field lines.

### 5.4.2 Derived Seeding Strategy for Streamlines and Streamsurfaces

1. Seeding field lines outside of the turbulent structure will reveal mostly horizontal, polewards oriented ones (See Figure 5.15). These lines are also leaving the domain and have no periodicity that can be observed.

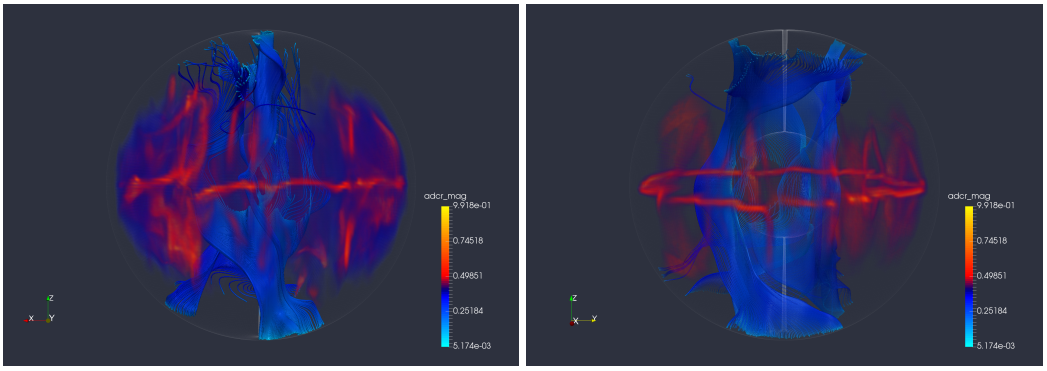
2. Seeding field lines inside the turbulent structure will **align** with the turbulence field pattern.

3. Seeding field lines in high turbulent areas as depicted in Figure 5.17 one can find crucial details in the underlying vector field like eddies or something similar to saddle points.

### 5.4.3 Results and Evaluation

To show the results, we mainly used two data sets from the geodynamo simulation. In the present study, the spatial resolution is $(N_r, N_\theta, N_\phi) = (97, 144, 288)$. Both represent a snapshot when simulations reach a quasi-steady state. We fix the $E = 5 \times 10^{-5}$, $Pr = 1.0$, and $Pm = 0.5$. the only parameter changed is the Rayleigh number to be $Ra = 1.2 \times 10^7$ and $Ra = 2.8 \times 10^7$, which is the ratio of buoyancy to diffusivity. Consequently, convection is more turbulent with increasing the Rayleigh number. The average Reynolds number $Re = UL/\nu$ is 173.4 and 91.1, respectively. In Fig 5.15, a structure tends to be more chaotic, and it is harder to perceive the pattern induced by the spherical harmonics. In Figure 5.15b the structure is much more smooth, and a pattern similar to the expected spherical harmonics is

shown. Interestingly, in the lower turbulent case, the high peaks are clustered on an equatorial ring. Wherein the turbulent case, the high peaks are a lot more scattered. This method's primary goal is to provide a good seeding strategy to visualize magnetic field lines. Based on the resulting scalar field, we are now able to apply the strategies for seeding streamlines mentioned earlier in Sec. 5.4.2. The alignment of the seeded streamlines with the referring turbulence field's structure is a general property of the metric. If this is the case, a wide variety of applications, not only for visualization but also for steering, come in handy. The second remarkable property is that high encapsulated turbulence values (areas) are correlated with what is typically denoted as critical points. We have not investigated yet if it is possible to find all critical points using this approach. However, we were able to find critical regions of interest where the mathematical definition's strict application would have failed. It is especially the case in Figure 5.17b, where behavior is found that reminds on a saddle point, but due to the magnetic field properties, there cannot be a saddle point in the proper sense. This way, one can reveal exciting features to the application scientists that were not recognized otherwise.




(A) $Ra = 2.8 \times 10^7$. Magnetic field lines seeded outside the turbulent area.View on the equator.

(B) $Ra = 1.2 \times 10^7$. Magnetic field lines seeded outside the turbulent area. View on the equator.

FIGURE 5.15: Streamlines of the magnetic field are seeded outside the turbulent areas marked by the volume rendered scalar field of turbulence. The field lines seeded outside this area are nearly straight going from one pole to the other. One can clearly see the difference in the overall turbulence of the field based on different Rayleigh numbers chosen. It is assumed that the field lines outside the marked area are responsible for building the dipole structure, as these are majorly the ones leaving the outer core domain.

### 5.4.4 Domain Expert Evaluation

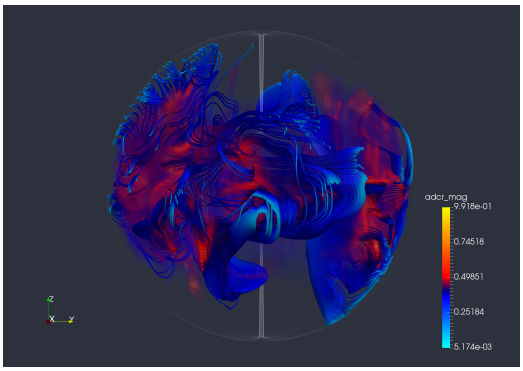
The present results show that large directional changes of the magnetic field locate along with the equator and extend along with the rotation axis. The equatorial plane's intense changes are due to the change in the direction of the zonal (longitudinal) magnetic fields generated by the zonal flow ($\Omega$-effect). Simultaneously, the magnetic field line is twisted towards the $z$-direction by the convection columns' helical flow (so-called $\alpha$-effect). However, the extracted area does not match the area where the magnetic energy is effectively generated because the present extraction is not considered the magnetic energy amplitude. Consequently, the present methods are useful for finding the area where the magnetic field is twisted by the turbulence and finding if such turbulence contributes to generating a large-scale magnetic field. We also expect that the present scheme potentially represents the flow's turbulence to apply the velocity field.

### 5.4.5 Conclusions

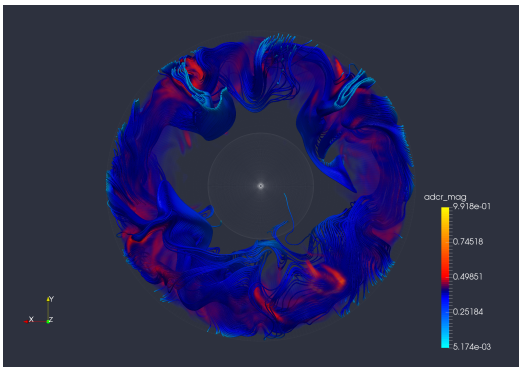
It has been shown that Magnetic Fields are a special kind of vector field in terms of visualization, which is mainly related to the fact that magnetic fields are, unlike flow fields, Hamiltonian systems. These systems allow for a more restricted behavior regarding critical points but induce a more turbulent behavior, leading to more topological and structural motivated visualization techniques. With a proper definition of turbulence on vector fields, we could visualize the underlying topological structure of the magnetic field (see Figure 5.18. It allowed us to study the structure and the field lines emerging either within or outside the structure. This way, we were able to reveal the two main types of field lines in the geodynamo. Those who are staying inside the outer core and maintaining the field's strength and those are leaving the outer core at the pole regions building the typical dipole shape of the magnetic field observed at the earth's surface and above. The upcoming challenges are leading in two directions. First of, as the magnetic field is a Hamiltonian system, the use of Poincaré maps to visualize and analyze the maintaining field lines rotating around the core seems very promising. On the other hand, the challenge remains in analyzing and visualizing the unsteady behavior of the geodynamo. Due to its large size, it is a visual challenge regarding the features or structures to be tracked and visualized over time and a computational to allow a seamless comparison between multiple time steps.

## 5.5 Discussion

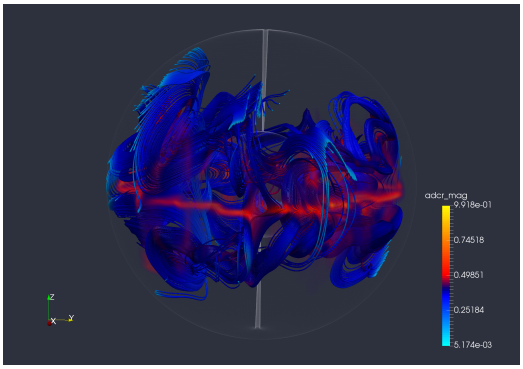
Mapping the impact of the parameters into the domain space is not a trivial task. Therefore we first developed a concept to view the results from a sensitivity analysis interactively. This was achieved by exploiting the functionalities of the *ttkCinema* concept. Here used the general concept of SQL queries on the database to achieve interactivity. With a proper query formulation, we then showed how to map each parameter's impact into the domain space. Further, we extended this newly created view of the scatter plot matrix to allow the evaluation of parameter cross-correlations in domain space. The generalization of this visualization concept could then display various other metrics defined in the domain. As such, we were finally able to observe different segmentation results after applying topological analysis techniques. This allows not only to evaluate one possible segmentation and incorporate the relationship of different parameter cross-correlations in the segmentation step. In an ongoing study, we now observe the effects of different topological segmentation approaches under different parameter pairs for using a minimalist machine learning model. Such a model's final goal would be to train the model only with a small number of representatives for a derived segmentation. Consequently, we could drastically reduce the number of parameters of our model and simultaneously increase the interpretability of the model.



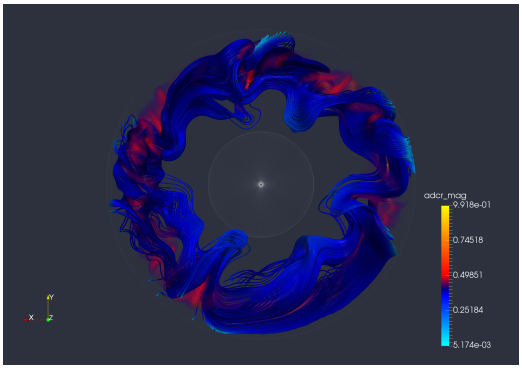

(A) $Ra = 2.8 \times 10^7$. Magnetic field lines seeded inside the turbulent area. View on the equator.




(B) $Ra = 2.8 \times 10^7$. Magnetic field lines seeded inside the turbulent area. Top view on one of the poles.

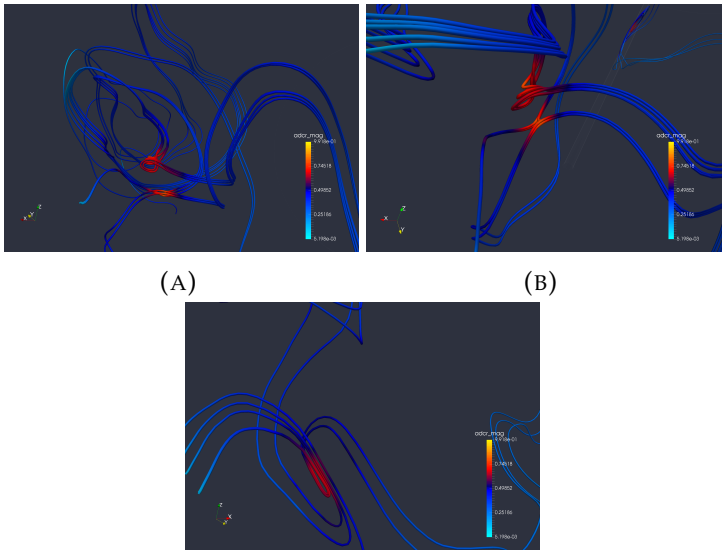



(C) $Ra = 1.2 \times 10^7$. Magnetic field lines seeded inside the turbulent area. View on the equator.




(D) $Ra = 1.2 \times 10^7$. Magnetic field lines seeded inside the turbulent area. Top view on one of the poles.

FIGURE 5.16: Streamlines of the magnetic field are seeded inside the turbulent structure depicted with the volume-rendered scalar field. Lines seeded in this area tend to stay inside the outer core even for the more turbulent case (a) and (b). In the more stable version of the simulation (c),(d) nearly none of the field lines are leaving the outer core domain. The ones that are leaving the domain here are of special interest and lead to further investigation. So far, no method precisely find these field lines.



(A) (B)

(C)

FIGURE 5.17: High turbulence Examples.From the mathematical description it can be shown, that the measurement only bare produce results with a value $t\,(p_0) \geq 0.75$. So it makes sense to investigate those areas in more detail. There is no single interpretation of what a high turbulence value could be, so here are some examples we investigated. In (a) we found a small eddy at the given point. In (b) close to the eddy we found something that looks like a saddle point, although we know they cannot exist in theory. In (c) again we found an eddy, but with a strong elliptical shape.

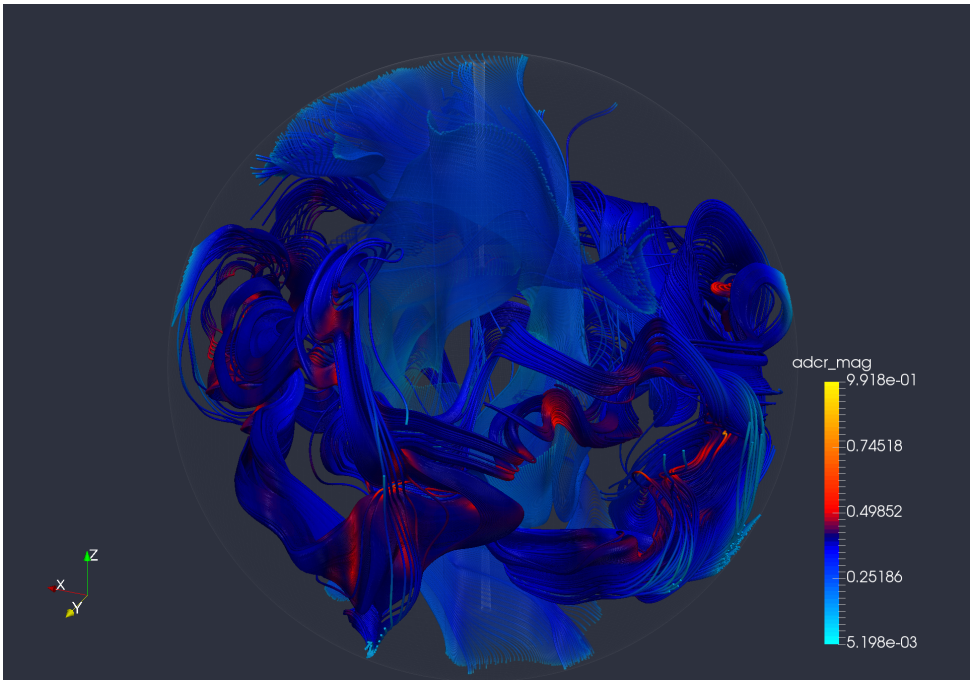



FIGURE 5.18: Dipole Visualization.With our proposed method, it is now possible to efficiently separate the two kinds of field lines. On the one hand, the field lines are building up the dipole and can be found by seeding streamlines outside the turbulent domain. While the driver magnetic field lines keep the geodynamo stable and stay inside the outer core. Those can be found by seeding streamlines inside the turbulent structure.

# Chapter 6

# Organizing Ensembles for Visual Analytics

Today's supercomputer architectures allow computational scientists to perform research with increasingly complex models producing high-resolution, high-fidelity data in increasingly shorter times. It is often no longer possible to store data set at full resolution. It is also extremely inefficient and expensive to transfer such large data sets from tertiary storage to memory analysis. Consequently, *in situ* visualization – i.e., the generation and storage of visualizations/images done as part of an ongoing computer simulation – has been developed[112]. Nevertheless, while an in situ approach is advantageous in many ways, it limits the possibilities of "complete data exploration" to the set of pre-generated visualizations. Ahrens et al. [113], [114] recognized that generating a large set of images covering the underlying parameter value space well still allows a scientist to meaningfully analyze and discover important model behaviors while accelerating the exploratory process significantly as all visualizations already have been generated.
The CinemaDB was first introduced by Ahrens et al. [101]. The basic idea is to organize the images created from in-situ approaches. Thereby it incorporates three main concepts that scientific analysts demands, which are: Interactive Database Exploration, Metadata Search, and Post-processing & Recombination. The interactive exploration database enables a set of interactions of pre-generated visualization and analysis results. The idea is to allow the user interactive queries on the database based on predefined parameters. These parameters can either be visualization or application-driven. Visualization driven parameters are for example: *view port* or *camera angle*, visualization method-specific parameters, e.g. *iso-values* for iso-surface visualization. Application-driven parameters, on the other hand, specify a set of variables from the application scenario. The most prominent example here is *time*, but this could also be any parameter from the application.

The Metadata Search allows to store and retrieve even more complex information alongside the images in the database. The idea is to formulate complex queries on the database entries to allow better analysis of the results. For example, we can develop a query that retrieves every time step, where the maximum value of field $X$ is more significant than a threshold of $th$. We already made extensive use of this feature in the chapters before.
The post-processing and recombination concept extends the initial focus of mainly storing images in the database by storing arbitrary visualization or data products. It allows us to link original raw data to our database entries and intermediate results like depth-images or triangulations. As such, it enables us to start our post-processing pipeline at any position.
In the following study of video compression techniques, we choose the depth images as our starting point for the compression. The use of depth images is a superior choice. It has already reduced the data to the visible range but still allows for compositing multiple results and custom rendering and coloring approaches in the post-processing phase.

**Related Work** There is a rich body of work focusing on data reduction for computational model output. Among the typical approaches are multiresolution techniques, adaptive refinement, compression; a survey and overview of techniques typically utilized in situ are given by Li et al. [115]. Common to all these techniques is that only reduced data is available for post hoc analysis and visualization. Often, explicit error bounds are not available or very difficult to estimate except in specific circumstances (e.g. [116]). Thus, the analysis's accuracy is in direct competition with data reduction, where it is essential to hit the sweet spot between the reduction rate and data quality [117]. Taking an alternate approach, in situ visualization [112], [118] generates visualization imagery on full-fidelity data. Through clever implementation, such as storing scalar value images and depth images instead of RGB images, compositing and color mapping can keep the image database size small. Lukasczyk et al. showed that under certain conditions, it is possible to reconstruct parameter sets not stored in the database, such as, e.g., camera positions, further enabling free exploration [119]. However, for vast parameter spaces that arise when combining many different visualizations in the interest of exploration, the visualization image database can become very large, making its storage and use difficult. For specific scenarios, optimized image formats can be defined, e.g., in the case of volume rendering [120] or contour tree analysis [121]. However, this only marginally reduces the size

of corresponding image databases. Image database storage also typically utilizes image compression techniques such as, e.g., wavelet compression [122] or commodity image compression codecs (e.g., JPEG) in both lossy and lossless modes. However, compressing each image on his own cannot leverage the high degree of similarity between images corresponding to closely neighboring parameter settings.

Based on the proof-of-concept work of Berres et al. [123], we assume that video compression is a suitable way of reducing the size of large visualization image databases. However, it is unclear which factors affect compression rate (i.e., data reduction), image quality, and retrieval performance, which are the vital pertinent aspects to consider when using compression for visualization image databases. In this paper, we investigate these aspects in a more comprehensive quantitative experiment.

## 6.1 Video Compression for Visualization Ensembles

### 6.1.1 Motivation

For complex visualizations generated, for example, by sampling large or high-dimensional visualization parameter spaces, the databases needed for storage can still be of substantial size. We adapt image and video compression techniques to reduce data size and enable more efficient management and visualization databases. The goal is to leverage inter-image similarity typically encountered in video image frame sequences and used to achieve substantial compression. By “linearizing" a visualization image database, it is possible to adapt video compression to support much more efficient data analysis. This approach was first investigated by Berres et al. [123], where it was shown that the approach is feasible and beneficial. However, their linearization approach and choice of compression parameter values, e.g., video codec and image encoding, were ad hoc. Our work is motivated by gaining more insight into these parameters’ effects concerning video-compressed image databases’ overall usefulness.

We present the results of a broader investigation of the different aspects of applying video compression to visualization image databases.
We quantitatively evaluate video compression for five different data sets with significantly different image characteristics expected to affect compression.

We employ three general-purpose and easy-to-use video codecs (H.264 [124], H.265 [125], VP9 [126]). We consider several quality metrics and compression/decompression efficiency concerning the compression ratio, serving as the main parameter for all three codecs. We can now formulate the following hypothesis for our study:

**Hypothesis 1** *The compression performance from video encoding techniques is significantly higher than general data compression or pure image compression.*

**Hypothesis 1.1** *The order axis of the collection has an impact on the compression result.*

**Hypothesis 1.2** *The test set (application domain) of the collection has an impact on the compression result.*

**Hypothesis 1.3** *The visualization technique of the collection has an impact on the compression result.*

**Hypothesis 1.4** *There is a significant difference in compression performance for the different video compression techniques.*

**Hypothesis 2** *The retrieval performance of video encoded image databases is significantly higher than with standard databases*

**Hypothesis 2.1** *The retrieval performance depends on the compression technique.*

**Hypothesis 2.2** *The retrieval performance depends on the axes ordering for the encoding.*

### 6.1.2 Experimental Setup

In the following, we describe the pipeline we employ to carry out our study. A conceptual overview is given in Figure 6.1.

#### Data Sets and Visualizations

For the generation of the visualization image database, we use ParaView [127] in combination with the TTK framework [100]. Prototype visualization pipelines are created in ParaView and stored as state files, which we then parameterize. The filters `ttkCinemaImaging` and `ttkCinemaWriter` are then used to render the images for a Cartesian product of parameter samples; including simulation time, camera positions, and iso-value (where appropriate).

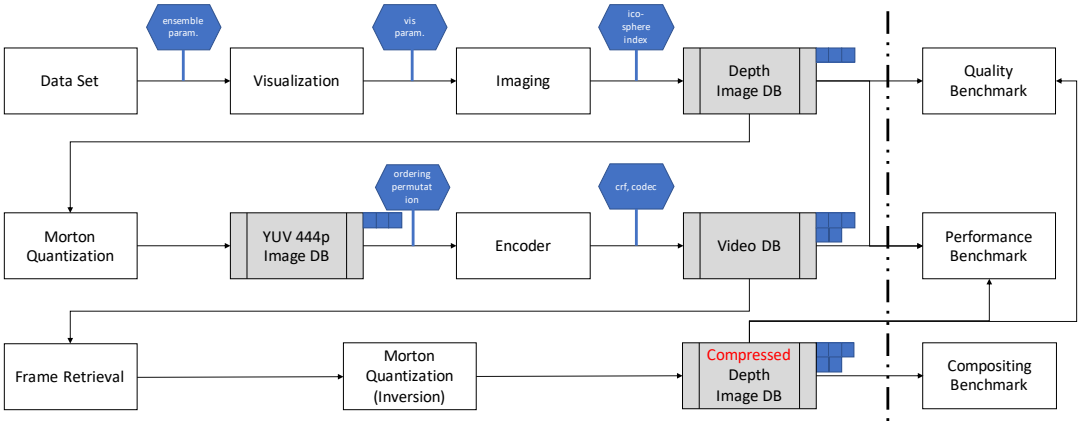



FIGURE 6.1: Overview of the image database generation, compression, and decompression pipeline underlying our study.

Camera positions are located on spherical grid vertices, and the cameras are aimed towards the data sets' centers. Following the reasoning of Ahrens et al. [101], we focus on a modern implementation of visualization image databases based mainly on storing depth images instead of color images, which allows recombination of different visualizations. As an intermediate product, we obtain a depth image visualization database, which associates a depth image with each parameter set and represents the ground truth for error measurements. The image resolution is chosen as 512x512 throughout the entire study.

### Image Database Linearization

We linearize the depth image database by enumerating the Cartesian product parameter space using an arbitrary ordering of axes. We hypothesize (and empirically confirm, cf. Section 6.1.5) that ordering axes in a manner that the fastest axis (along which subsequent images lie in the linearization) should be chosen such that the inter-image similarity is large to benefit compression efficiency.

Most video codecs assume that the changes from one frame to the other happen smoothly and in a predictable way over a more extended period (see the preset parameters [128]). Therefore we assume that the best compression performance results will occur for a smooth ordering of the images. We are investigating the impact of camera path, data time step, ensemble parameter, and visualization parameter as different approaches for the dominating ordering axis (see Figure6.2).

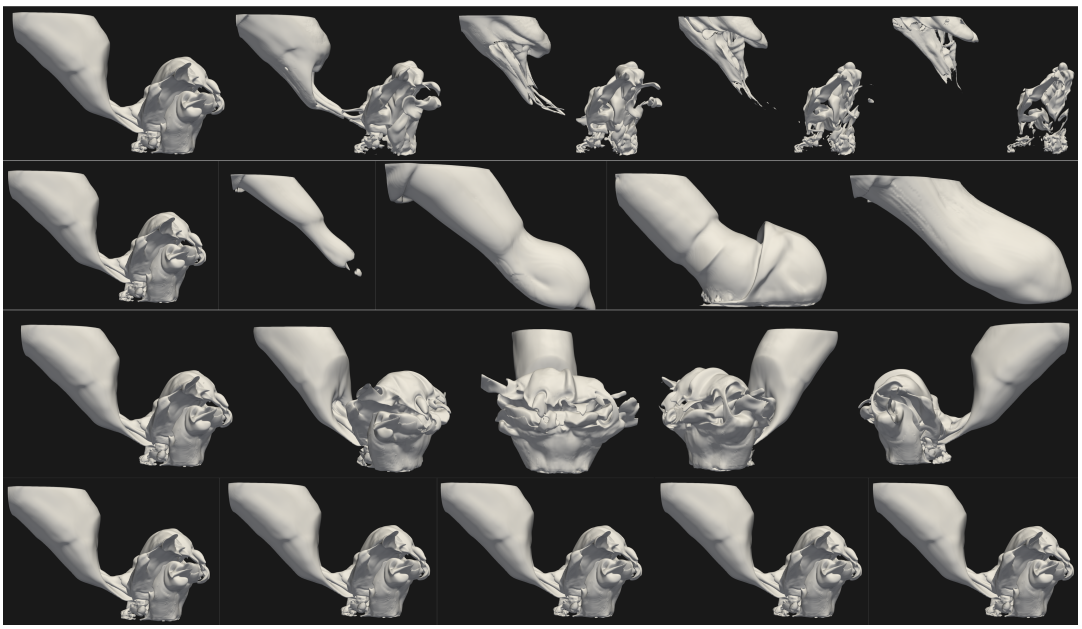


FIGURE 6.2: Showing different ordering strategies for one data set. Depending on which axis is chosen to be the dominant one, the similarity between successive images could vary dramatically. From top to bottom: Iso-value, Ensemble Parameter, Camera rotation, Simulation time step.

**Camera Location** The camera parameters are usually the first parameters that are used while creating an image database. A collection of images ordered by camera angles is similar to tracking shots in video sequences, which target video compression techniques. Based on the number of viewpoints needed to capture a visualized data set; however, it is not necessarily the axis with the smoothest transition between successive images. Especially if storing a minimum amount of images needed to visually adequately reconstruct geometry [119], camera location index as the fastest axis is not necessarily the right choice.

**Data Time** For unsteady data sets, time provides a natural parametrization. In practice, memory or I/O constraints determine the step size between successive discrete time points. Furthermore, considering ensemble data, step sizes may vary per ensemble member. The chosen visualization techniques also play a crucial role here. In contrast, some techniques will result in more significant visual changes with small changes in the time, and others may behave in the opposite direction. Therefore, we do not rely on smooth transitions between images for our setup when choosing time as the dominant axis.

**Ensemble parameter space** In ensemble data sets, we are particularly interested in the differences in the input parameter space changes, which leads to similar effects for setting the time step size mentioned above. Therefore, for our investigation, we expect that it behaves similarly to the time ordering for unsteady data sets if they have a matching resolution of the step sizes.

**Isovalue** Concerning algorithmic parameters that change the visualization result, we consider iso-value in the cases described above as a proxy for more general settings. Most notably, it is one of the most often changed parameters when exploring a data set. In general, variation between images should be small under a fine-grained sampling, as the surface varies slowly and smoothly with a change in iso-value. Thus, we hypothesize that this parameter is the right choice for the fastest axis in the linearization if the sampling is not too coarse.

### Depth Image Encoding

Before applying video compression to depth images, these must be transferred into a suitable format for ingestion by video codecs. While depth images encode a single scalar in the range $[0, 1]$ per pixel, the lack of robust support for single-channel image formats in video encoders (e.g., the gray-scale format *12greyle*) makes it difficult to pass the depth values directly. Furthermore, a lack of format with high-bit depth – at most 12 bits – would induce inacceptable quantization to the depth images and make them practically unusable for compositing.

Hence, we encode the scalar field (depth image) into an image format with three channels and 8 bits per channel using Morton coding [129], mapping a 24-bit depth value into 3x8 bits. Here, Morton coding is much preferable to the simple mapping of the high, middle, and low bytes to three channels since the Z-order curve underlying this coding guarantees that close-by depth values will be mapped to close-by tuples. The effect of this mapping, interpreting the three channels as red, green, and blue colors, is illustrated in Figure 6.4. We pass the result of this mapping to the video codec in the *yuv444p* format. YUV is the natural color space in which all considered codecs operate, and errors induced through in-codec color space conversion can be avoided. Note that video codecs typically accept various formats that represent color information at a reduced resolution compared to luminance information, such as, e.g., the often-used *yuv420p* format. However, we do

not see an indication at this time how such sub-sampling would benefit the compression of depth images and thus only consider *yuv444p*. In general, this process applies to any scalar field (not only depth) input on the images.

#### Video Compression

Following linearization and encoding, we pass the *yuv444p* image sequence directly to the video codec. As a general interface to different codecs, we employ the `ffmpeg` [128] tool, and encode using three major and broadly available video codecs:

- H.264 [124] The H.264 uses motion estimation to minimize temporal and spatial redundancies. It is classified as a block-oriented motion compensating compression technique.
- H.265 [125] The H.265 is based on the same principle as the H.264. Its main difference is the increased coding tree unit from 16×16 to 64×64, though leading in general to higher compression rates.
- VP9 [126] The VP9 codec is also a block-based format. Its main application area is for web streaming, and therefore it is designed to ensure a certain bit-rate rather than a consistent quality like H.264 and H.265.

**Codec Parameters.** In general, a vast number of parameters affect each encoder and allow tweaking it to different types of input. To keep our study reasonable and achieve comparable results, we opt to focus on the *constant rate factor* (**CRF**) as the main parameter that affects the amount compression for each codec. We consider the values 0 (lossless), 10, 20, and 30 to represent different compression levels. Typical CRF choices for natural images are in the range from 18 to 24.

Furthermore, all codecs support variable-rate encoding, which adapts bandwidth based on heuristics, in overall better image quality. However, as the assumptions underlying these heuristics are geared towards natural images and are not well documented, we choose not to examine this mode due to a large amount of unpredictability it induces on results. Note that the VP9 codec is primarily intended for streaming applications. Thus, constant rate encoding is not its optimal mode of operation; however, we still consider it in this study to its ubiquitous use and generally good compression/quality performance.

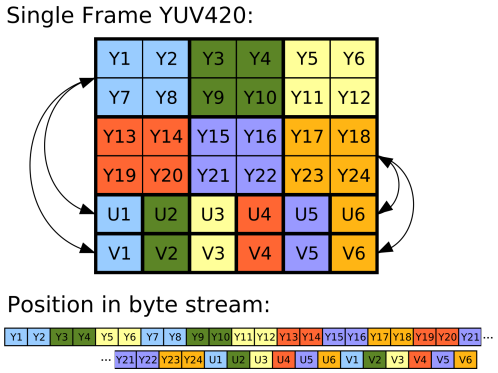
The H.264 and H.265 codecs are further able to trade off compression speed against image quality through a preset choice. We here use the *veryfast* preset to achieve the best speed. While image quality would improve with slower presets, we obtain a lower bound on image quality at the expense of much longer compression times, which we consider to best reflect real-world considerations. A summary of the parameters steered by these presets can be found in [128]

As the compression process's output, we obtain a single video file representing the entire visualization depth image database.

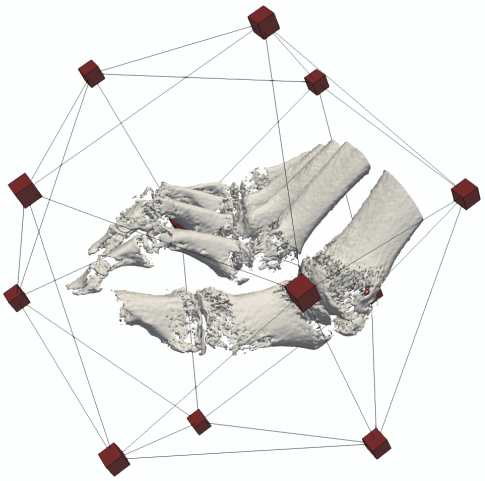
### Image Retrieval

To retrieve images, we again employ the `ffmpeg` tool to retrieve a single image from the video file in *yuv444p* format and decode it using the Morton order.

Retrieved images are compared against the (uncompressed) ground truth images, using the metrics described in the following section.




(A) Exemplary depiction of an unequally distributed pixel format. Picture from https://en.wikipedia.org/wiki/YUV ©Public Domain



(B) Depiction of an ico-sphere. Showing the view ports used for rendering 3d-visualization products. The exact positioning and definition of the near and far- plane allow for efficient geometry reconstruction [119].

FIGURE 6.3: Pixel format and Ico-sphere

The depth and scalar images are generated based on the approach of Lukasczyk et al. [119] that is implemented in the "CinemaImaging" module of the Topology ToolKit (TTK) [100]. This module utilizes the Visualization

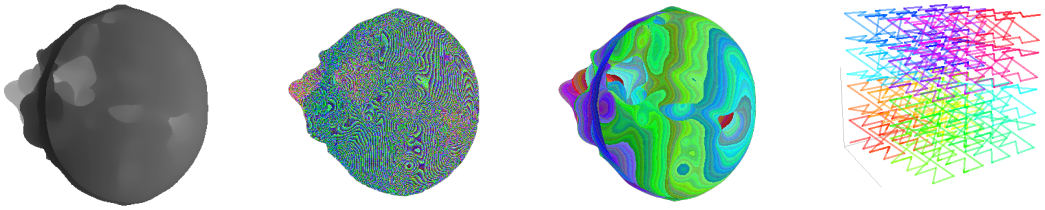
ToolKit (VTK) to render opaque geometry for a Cartesian product of parameter samples, including time, camera positions, and iso-values. Using VTK as a rendering backend makes it possible to deploy our simulation runtime approach via Catalyst [127]. Our experiments' datasets are depicted from cameras located on spherical grid vertices, and the cameras aim towards the dataset centers. (Figure 6.3b). The resulting images are stored individually in a Cinema database together with their corresponding parameters.

### 6.1.3 From Scalars to pixel formats - Morton Quantization

The lack of a good support for scalar like image formats in video encoders (like *12greyle*) forces us to transfer our scalar field (depth image) into an image-like 3 channel format. Most video encoders accept the YUV pixel format. The format itself comes in different variations, were the three channels Y,U,V are differently distributed per pixel. Resulting in the 3 most common variations of the YUV format, namely: yuv444p, yuv422p, yuv420p. While yuv444p uses 3 bytes per pixel, yuv422p uses 4 bytes per 2 pixel and yuv420p uses 6 bytes per 4 pixels. The p is indicating that the array is flattened out in one dimension. For our experiment we choose to stick with the yuv444p format as it maintains the 24bit precision of our depths values. The goal therefor is to find a transformation that is maintaining this precision and is invertible. The easiest approach, give every channel the respective float value as an unsigned 8bit integer, therefor is not suitable as we loose 16bit of precision. As the pixel format only accept integers, we first transfer the scalar value into a 24 bit unsigned integer. Because our depths values are normalized between 0.0 and 1.0 this can be achieved via:

$$s_{int} = s * (2^{24} - 1) \tag{6.1}$$

Next, we have to split the 24bit integer into 3 x 8-bit parts. The naive approach would be to use the modulo operator for this, resulting in $Y = s[0, 7]$, $U = s[8, 15]$, $V = s[16 - 23]$, with $s[i]$ referring to the i-th bit of $s$. This would lead to images like the one in Figure 6.4 (b). The resulting image is boisterous, and the underlying structures and features are not visible anymore. They are resulting in an encoded video nearly the same size as the original image database. As such, we need a conversion that leads to less noisy images to allow the video compressor to find similar compressible parts in an image sequence. Therefore, Morton encoding, proposed by Morton [130],



(A) Ground Truth (B) Modulo (C) Morton (D) Z-order curve

FIGURE 6.4: Encoding of a 24-bit depth image (a) to 3x8 bit channels using reinterpretation as three bytes (b) or Morton coding (c). The three resulting channels are interpreted as RGB for illustration purposes. (d) shows the Z-order curve underlying the Morton coding.

often referred to as z-order curve, is used. The idea is to fit a smooth parameterized function into a 3-dimensional space, such that the transitions from one axis to the other are smooth(see Figure6.4).

For the z-order curve we choose the following implementation:

LISTING 6.1: Example Code for the used Morton Encoding on the depth image values.

```
def float_to_morton(n):
    return unshift(n), unshift(n >> 1),
        unshift(n >> 2)

def unshift(n):
    n &= 0x09249249
    n = (n ^ (n >> 2)) & 0x030c30c3
    n = (n ^ (n >> 4)) & 0x0300f00f
    n = (n ^ (n >> 8)) & 0xff0000ff
    return n
```

### 6.1.4 Evaluation Metrics

For our evaluation, we distinguish three significant factors: Quality of the reconstructed visualizations images, compression efficiency in the sense of file size and time to retrieve an image in the original domain, and, especially for surfaces, how much the errors introduced by compression affect the compositing of two or more visualizations post hoc.

**Performance Metrics**

For the performance measure, we use the compression and retrieval rate. For the compression rate, we use the highest z-lib compressed depth images as our base value. All values are then relative increases or decreases in percent. We randomly draw 300 frames from each video for the retrieval rate and apply the Morton inversion to them to end up with depth images. We measure the time from the start of the retrieval (the video already loaded) to the conversion end. For comparison, we measure the time for 500 database calls using Paraview and the ttkCinema filter. The measures are then normalized to time to retrieve one image. We do not take speed ups from parallel computing into account, as such practical values can differ.

### 6.1.5 Results

The complete results can be found in the published article of the study [131]. As such, we only review the results focusing on the hypothesis formulated in Section 6.1.1. Figure 6.5 shows that the video compression techniques always achieve a higher compression rate than the *zlib* approach in the highest setting, which confirms the first hypothesis. Additionally, we can observe that the codecs have varying performance based on the chosen constant rate factor, which aligns with our hypothesis 1.3. The axis order, on the other hand, only has a minor influence on the compression rate. As such, we cannot confirm hypothesis 1.1. In Figure 6.6, we can confirm hypothesis 1.2 and 1.3 that there is a significant impact on the compression rate based on the data set. Essentially streamline visualizations are more challenging to compress than iso-surface approaches. Regarding hypothesis 2, we can confirm that the retrieval times from video compressed databases are significantly higher than for uncompressed ones (see Figure 6.7. The retrieval time is in the order of 2.5 to 3.5 seconds, respectively. In contrast to our hypotheses 2.1 and 2.2, we did not find any effect of the ordering or compression technique. For the investigates constant rate factors, we observe that the visual quality is only visibly decreasing with $crf > 20$ (see Figure 6.8).

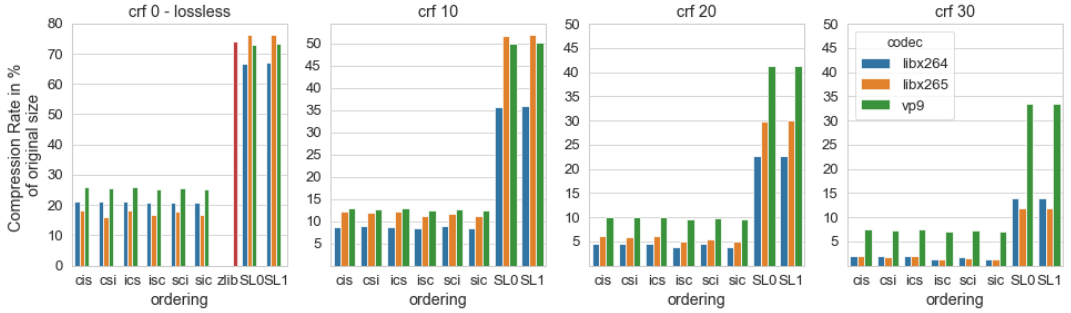



FIGURE 6.5: Results for the compression performance. Compression rate in % of the original uncompressed image data base. grouped by image ordering for iso-surfaces(s: camera **s**phere, c:**c**ycle time step, i:**i**so-value), image ordering for streamlines (S: camera **s**phere, C:**c**ycle time step) and constant rate factors, colored by the used codec.

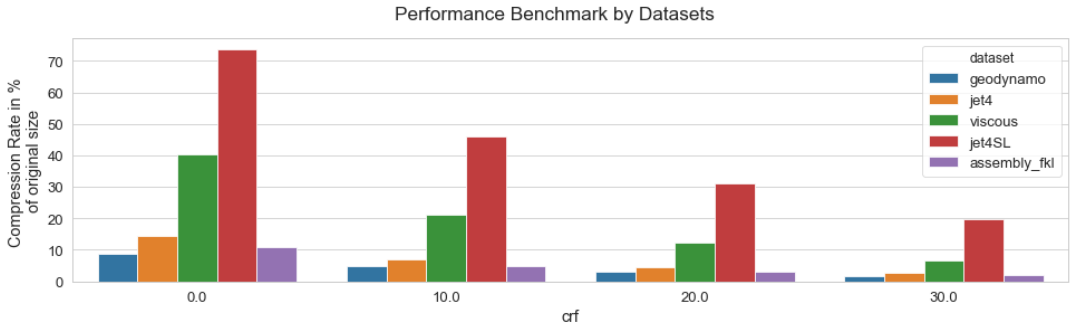



FIGURE 6.6: Results for the compression performance and constant rate factors, colored by the used data set.

## 6.2 Discussion

We have provided a study concerning the applicability of lossy video compression to visualization image databases. Our findings confirm observations made by Berres et al. [123], thereby strengthening further the argument that video compression is a viable and beneficial approach leading to excellent compression rates compared to a general-purpose lossless compressor. Further, we observed only minor image quality loss and, in some cases, no loss at all. The implementation complexity is manageable but needs further improvement to retrieve single frames, which should incorporate suitable meta-data. The use of lossy general-purpose compression techniques – i.e., techniques not primarily aimed at video compression – such as the ZFP [132] compressor, should be investigated too. For real-world applications, e.g., in situ visualizations, the ability to perform compression and encoding in parallel is essential. There is hardware-enabled acceleration for most codecs to

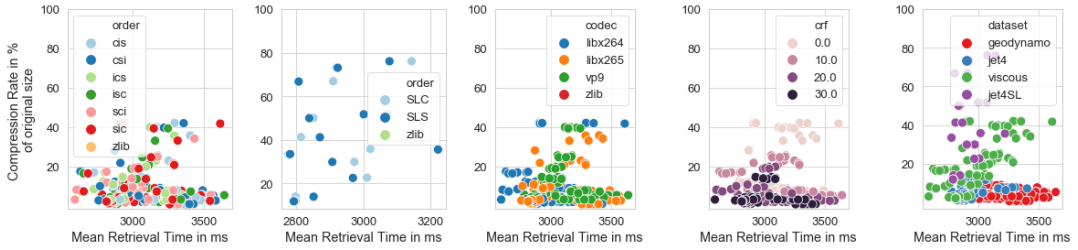



FIGURE 6.7: Comparison of compression rate and retrieval time clustered by codec, image ordering for iso-surfaces(s: camera **s**phere, c:**c**ycle time step, i:**i**so-value), image ordering for streamlines (S: camera **s**phere, C:**c**ycle time step), constant rate factor and data set.

run the GPU encoding, which drastically improves the encoding speed. Additional independent encoding of subsequences could be a valid alternative. As we found out, the image order is only of minor importance. Future research would be of interest to investigate the optimal ratio between retrieval time, compression, and image batch size.

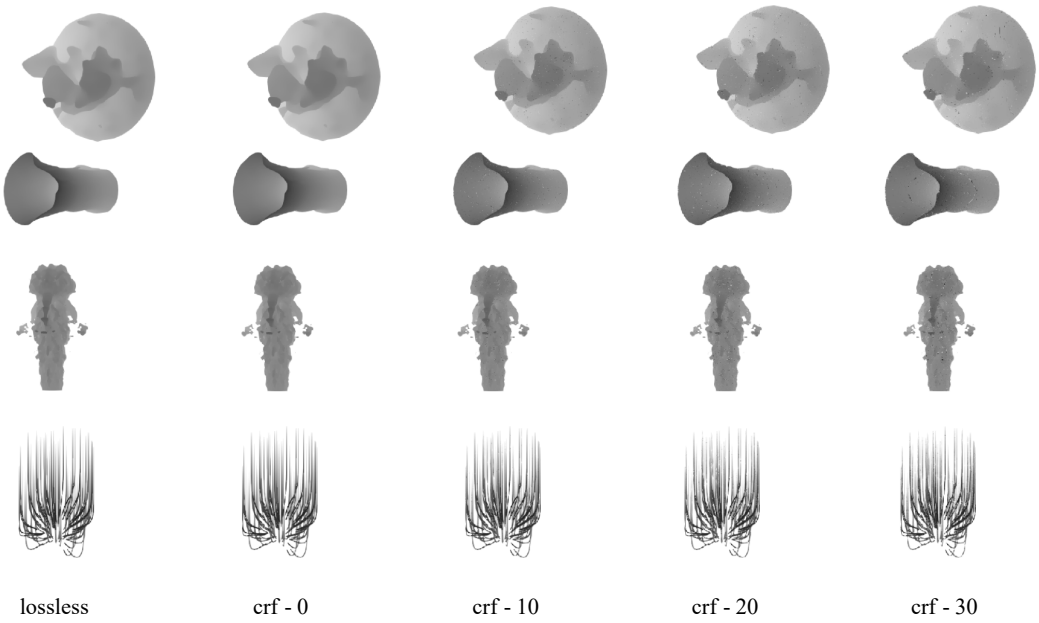



FIGURE 6.8: Depth Images before and after compression with the H.264 codec. The visually detectable differences are very small. Starting at crf= 10 the noise in the images is increasing with each step. Still most of the topological structures remain intact.

# Chapter 7

# Conclusion and Future Directions

Enabling the application engineer to develop their machine learning methods is a challenging task. It demands an understandable and interpretable model deduction approach. In the presented thesis, we elaborated on the challenges to meet those requirements for the application in reverse engineering for linear elastic metal sheets. We showed that the concept of minimalist machine learning could achieve superior results in the assembly's well-structured application. However, it can be easily adapted to many other applications in manufacturing and production.

**Dealing with Uncertainty** The challenge of uncertainty in both the machine learning model and the associated digital twin can be tackled with the combined use of the classical Ishikawa's method [47] and our advanced concept of an Object-oriented Virtual Reference Scheme (OOVRS). The tremendous advancements in simulation, physical modeling, and machine learning pave the way for smart factories' deployment in the next decades. Building trust persists to be the main goal in the future for simulations and machine learning models. We showed that uncertainty in both models and methods is a crucial criterion for trustworthiness. However, the analysis and testing effort for achieving reliable uncertainty quantizations is an ongoing challenge. On the other hand, with the OOVRS, we propose a solution for the integration problem of process chain elements while increasing transparency with a human-readable concept in compliance with the SOLID principles [50]. Sticking to the SOLID principles is a necessity to allow for robust software system architecture in smart factory applications. Jointly this opens up a plethora of service applications with such an integrative scheme throughout the whole production process. Thinking about the increasing demand for transparency and traceability of production chains in industry 4.0, we introduced a future proof concept for dealing with uncertainties in smart factory eco-systems.

**Minmalistic Model and Distance Metrics** The choice of a proper distance metric as a loss function in the machine learning model is critical in the first phase of model generation. In general, we could show that comparing the two deformation fields is a non-trivial task. With the use of visual analytics and a profound study design, we were able to determine the differences between different metrics and finally show that the proposed custom metric, which combines both the change in the length of the vector in each direction and the rotation of the vector is the best choice. Specifically, it is superior to the other when we continuously reduce the model's complexity, one of our main goals for an interpretable model and one step further to a minimalist machine learning model. The thoughtful deployment of visual analytics tools and model parameter study design enables increased interpretability of the model. Further, we could show that a simple model with one hidden layer can solve the prediction task. Compared to the effort put in simulation models for linear elastic behavior, this is a stunning result. It shows that the linearity in the underlying physical principles persists throughout the scales. Further, it shows that increasing the geometries discretization resolution is not continuously required to achieve more precise results in the studied application for reverse parameter prediction. A study design aiming for a minimalistic model turned out to be the right approach for such predictive engineering tasks, like the one investigated here.

**Integrative Visual Classifier Perfomance Benchmark** To further increase the proposed model's understandability, we analyzed the prediction parameters' influence on the resulting difference between a target deformation and the training deformation fields. Here we showed the deduction of a powerful visual analytics tool providing user- and task-adaptable guided representations that enable full situation awareness while supporting detailed actions. We support multiple levels of data and information abstraction. The derived tool is then used to benchmark the expected performance increase of a segmentation method. We could confirm the Accuracy Paradox [19] implications in the studied application for reverse parameter prediction in assembly. Instead of solely relying on the classifier's performance metrics, deploying an integrative visual analytics approach on the training data showed a considerable impact on the model results' interpretability. It allows a preliminary hypothesis formulation of, e.g., parameter correlations, for which the

classifiers could be tested. Moreover, the interpretability of the model's performance increases this way significantly. It further allows us to compare different classification methods on a high level using ROC curves and detailed scope using decision boundaries. The detailed scope contributes to a direct comparison of machine learning behavior under different training data features (variance, topology), which ultimately results in a benchmarking tool for monitoring the general high-level machine learning parameters: input data (fractions, segmentation), and methodology (e.g., decision tree, neural net).

**Impact Maps - Incorporating the Domain Space in the Visual Analytics Pipeline** To reduce the number of trainable parameters, we strove for an excellent and explainable segmentation of our domain. The goal is to find a minimal number of segments such that our regression model's performance is still as high as for the more elaborate setting. With the use of a user- and task-guided sensitivity and topological analysis, we were finally able to evaluate different segmentation results. Further, we enabled the incorporation of parameter cross-correlations in the segmentation step. The sensitivity analysis concept showed excellent results for evaluating the input data's parameter distribution in domain space. Combining this concept with an interactive query and post-processing (e.g., variation, topology, or feature computation) contributes to the high interpretability of the input data without the sufferings resulting from summarizing values or determining good representatives. We achieved a smooth link for visual correlation analysis tasks throughout different scopes by introducing a matrix layout with queries based on the SPLOM concept for 1-dimensional values. Further, we enabled the incorporation of post hoc analysis steps into the correlation analysis pipeline. The screen size is mainly limiting the number of parameters examined at once. When using a set of representative or summarizing single values, the application of dimension reduction steps such as the Principal Component Analysis showed excellent results while retaining the essential interpretability using bi-plots. Transferring these to the domain space remains an ongoing challenge.

**Applying Video Compression for Reducing File Sizes in Large Ensembles** The post-processing and recombination concept of the Cinema approach extends the initial focus of mainly storing images in the database by storing arbitrary visualization or data products. This allows for continually growing

collections of data with multiple types.
Using video compression to reduce the data size of image-like series further showed promising results. The video compressors' loss-less setting is of vital interest for many applications and already resulted in three times smaller data sizes than the commonly used *ZLIB* compression in the highest setting. The impact on the series's ordering is smaller than expected and decreases with input's complexity (e.g., the ordering's impact nearly vanishes for streamline-visualizations, while for the relatively smooth geometry in the assembly use-case, the view-port is the best choice as dominating ordering axis). Nonetheless, the native video compressors suffer from the exact frame retrieval bottleneck. For a direct replacement of the raw data with the compressed one, the retrieval and reconstruction time overhead has to be minimal. The native application for the video compressors, retrieving an image series above a set frame rate, is:

1. maximum demanding for roughly 100-120 frames per second
2. Accepting a delay when jumping to a timestamp
3. Not demanding for an exact timestamp to initial frame position mapping (only surjective, but no bijective)

As a consequence, the retrieval rates are too slow for most applications. For general applicability, the compressed data structure requires adaption for post hoc processing and visualization demands.

In summary, we could state that our model's transparency is increased with our visual analytics guided model deduction approach. We provide a set of tools to cover the challenges of causality and bias in our models while retaining fairness and safety in our predictions. This way, we achieve an explainable and reliable predictive model with consequent visual analytics application throughout all stages. In future research, we will focus on the relationship between topology and learned features and their impact on explainable machine learning models in engineering applications.

# Appendix A

# Application Scenarios

## A.1 Assembly Simulation for metal sheet parts

### A.1.1 Generic Metal Sheet Deformation

We present an example based on the deformation of a simple sheet metal car body part to show the performance of the proposed method. The chosen simulation model shows a similar deformation per length ratio like real exterior car components during assembly. Thus, this example is representative for real-world use cases. Figure A.1 shows the geometry and used boundaries. The shown part has dimensions of 50 mm by 30 mm and has a thickness of 0.5mm. The material model is a linear-elastic model with the properties of steel (E-Modulus of 210GPa and Poisson ratio of 0.3). The boundaries are shown in the picture as well, two boundaries located opposite each other on the edges in X-direction, four boundaries in Z-direction on the part surface, and two boundaries in Y-direction on the same edge. The Y-boundaries are set to zero to hold the part in place. The other directions are modeled as displacement load and varied within the tolerance interval of +/- 0.1 mm, which leads to different shapes the part can take on.

### A.1.2 Automotive Engine Hood

For a more application-driven view, a real-world example from the automotive industry was chosen. The geometry is an engine hood. This part has two hinges, two locks and two buffers as mechanical boundaries attaching the hood to the chassis, see Figure A.2. A finite element (FE) simulation predicts deflections during the assembly process of the part. However, the final shape

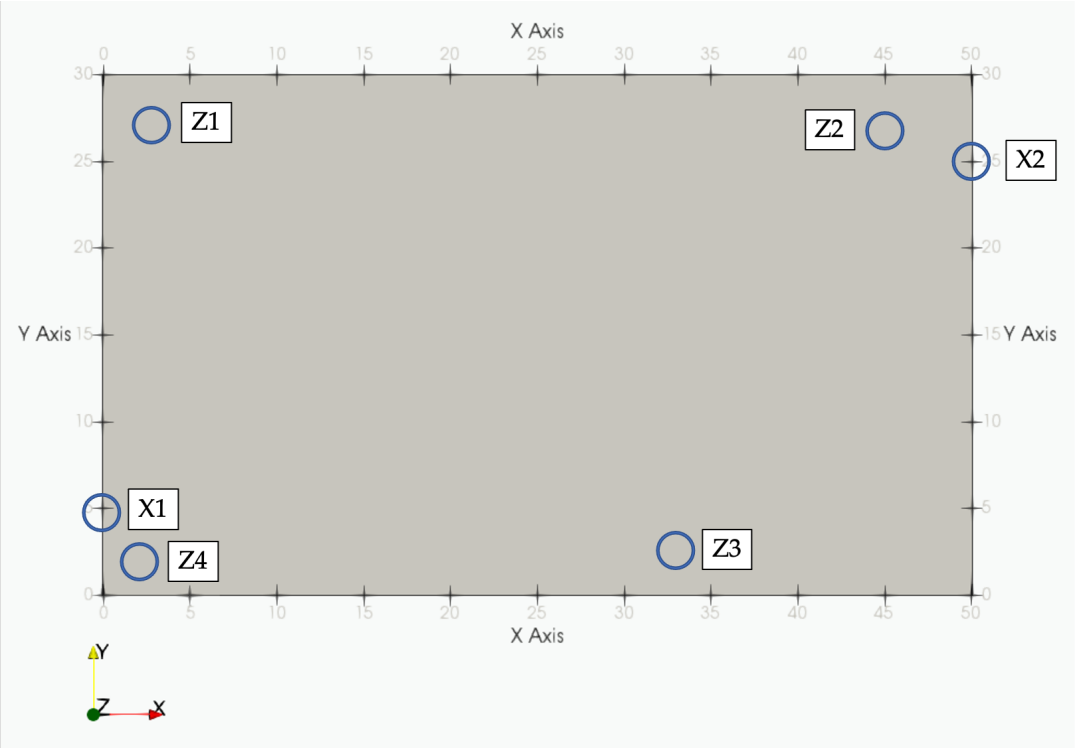



FIGURE A.1: Generic sheet metal example. The boundary conditions for the simulations used in this paper are depicted and labeled. The directions (X, Y and Z) for this part are indicated.

of a material part can vary due to production uncertainties. To deal with uncertainties and tolerances during the production, the engine hood's boundaries, i.e., hinges, lock and buffers, are adjustable. Adjusting these boundaries properly to obtain an acceptable gap and flushness is a challenging task [133]. A post-assembly measurement induces necessary corrections. The goal of the proposed method is to find the best set of changes from measured deviations, which forms the optimal set of adjustments. The method uses as input an assemble of statistical distributed simulations that cover the solution space spanned by the available adjustment possibility of each boundary. The used car hood is an assembly containing seven individual sheet metal parts, connected by spot welds and different types of adhesives. Based on the CAD files, a simulation model was created by meshing the geometry with 3D-shell elements and connecting the assembly considering spot weld, adhesive positions, and thicknesses of components. The material model is linear-elastic with an e-modulus of 210Gpa and a Poisson rate of 0.3. Two fixed, external loads, modeling the gas springs near the hinges with the magnitude of 580N each, complete the model.

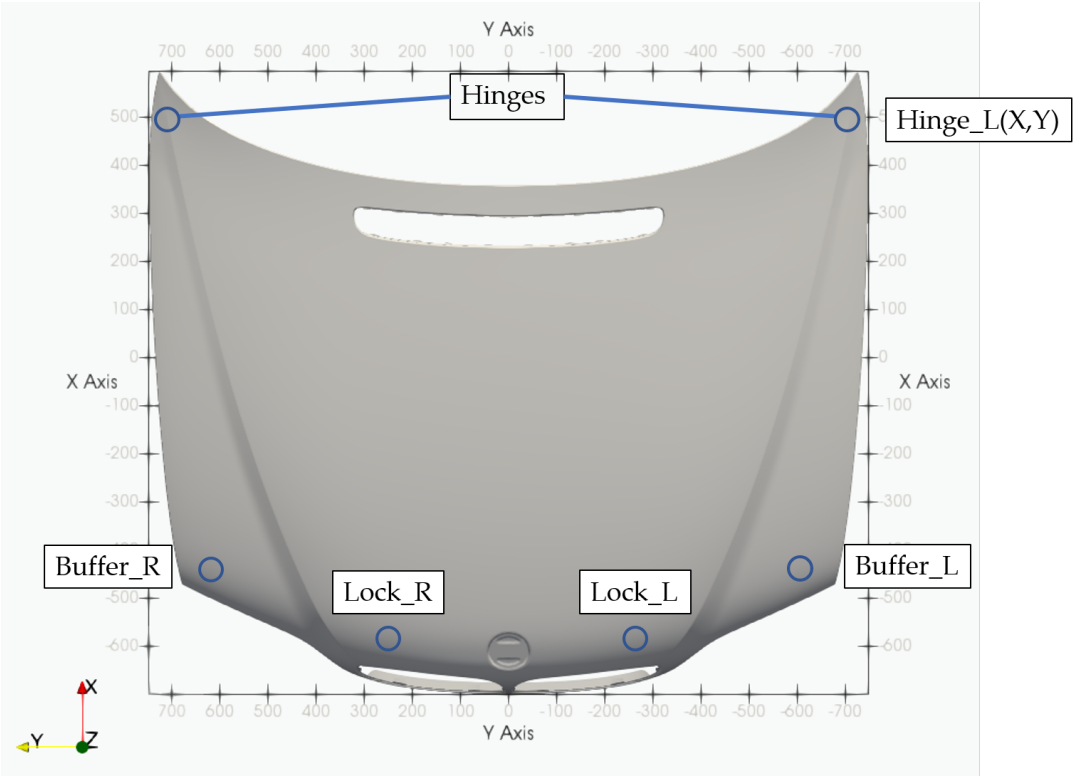



FIGURE A.2: Sheet metal car body part used in automotive industry. The used boundaries for the simulation are highlighted, with external forces used via gas springs. The coordinate system for this part is shown.

## A.2 Geodynamo - Simulating the emergence of earth's magnetic field

This data set is the result from a numerical simulation of the earth's liquid outer core based on the effect of the geodynamo. The resulting domain is covering the whole outer core and is stored in a spherical unstructured grid. The resulting fields like pressure, temperature, velocity and magnetic field are characterized by high turbulence and large structures. The main challenge in this data set is its spherical domain and structures which are mostly hard to deal with standard approaches for cubic domains. In addition the time steps are in the order of thousands of years.

## A.3 Viscous Fingering

The data set is the result from a simulation of salt dissolving in water. The domain consists of a cylindrical flow, at the top of the cylinder a solid body of salt is placed that is dissolved by the water. The resulting fields are the velocity of the flow and concentration of salt in the water. The data set is time varying and consists of multiple parameter settings.

## A.4 Jet Flow

The jet flow is an artificial unsteady flow simulation resembling the outlet of a jet engine. It is a well studied example data set for flow visualization and analysis and has very characteristic features. The resulting fields are the velocity and the temperature.

# Appendix B

# List of Publications

## Published

- Ruediger-Flore P., Glatt M., Aurich J.C.: Herausforderungen für die Nutzbarkeit von Maschinellem Lernen bei hohem Variantenreichtum und kleinen Serien - Implikationen am Beispiel der Nutzfahrzeugproduktion *Zeitschrift für wissenschaftlichen Fabrikbetrieb*, Vol.116(7-8), pp.538-543, (2021)

- Ruediger P., Garth C., Hagen H., Leitte H.: Is smaller always better? - Evaluating Video Compression Techniques for Simulation Ensembles. *2nd International Conference of of the DFG International Research Training Group 2057 – Physical Modeling for Virtual Manufacturing (iPMVM 2020)*, Dagstuhl OASIcs series, Vol.89, pp.10:1-10:18, (2021)

- Ruediger P., Claus F., Hamann B., Hagen H., Leitte H.: Combining Visual Analytics and Machine Learning for Reverse Engineering in Assembly Quality Control. *Journal of Imaging Science and Technology*, pp. 60405-1-60405-13, (2020)

- Ruediger P., Spilski J., Nûjîn K., Gsuck S., Beese, N.O., Schlittmeier S., Lachmann T., Ebert A.: Cognitive Indicators for Acoustic Source Localization and Presence in a vivid 3D Scene. *Proceedings of the 23rd International Conference on Acoustics*, pp. 8234-8241, (2019).

- Ruediger P., Hagen H.: Dealing with Uncertainties in Manufacturing Process Simulations. *Applied Mechanics and Materials*, Vol. 869, pp. 226–233, (2017).

- Ruediger P., Weber C., Matsui H., Eric H., Hamann B., Hagen H., Kellogg L.: Pre-filtering Turbulent Vector Fields in the Geodynamo. *IEEE Visualization Conference*, Application Track, (2016)

- Scherrer A., Ruediger P., Dinges A., Küfer K-H., Schwidde I., Kümmel S.: Breast cancer therapy planning–a novel support concept for a sequential decision making problem. *Health Care Management science*, Bd. 18(3), pp. 389 -405 (2015)

- Scherrer A., Ruediger P., Dinges A., Küfer K-H., Schwidde I., Kümmel S.: Software assisted decision making in breast cancer therapy planning. *Operational research applied to health services (ORAHS) 2013* ,Conference Proceedings, Istanbul (Turkey), pp. 99 -102 (2013)

- Scherrer A., Ruediger P., Dinges A., Küfer K-H., Schwidde I., Kümmel S.: Software assisted decision making in breast cancer therapy planning. *Operations Research Proceedings*, pp. 405 -411 (2013)

## Accepted and in publication

- Ruediger-Flore P., Leonhardt V., Claus F., Hagen H., Aurich J.C., Garth C.: PREVIS - A Combined Machine Learning and Visual Interpolation Approach for Interactive Reverse Engineering in Assembly Quality Control *to appear in Procedia CIRP, 15th CIRP Conference on Intelligent Computation in Manufacturing Engineering*,(2021)

# Bibliography

[1] Finale Doshi-Velez and Been Kim. "Towards A Rigorous Science of Interpretable Machine Learning". In: *arXiv:1702.08608 [cs, stat]* (Mar. 2017). arXiv: 1702.08608. URL: http://arxiv.org/abs/1702.08608.

[2] David Silver, Aja Huang, Chris J. Maddison, et al. "Mastering the game of Go with deep neural networks and tree search". en. In: *Nature* 529.7587 (Jan. 2016), pp. 484–489. ISSN: 0028-0836, 1476-4687. DOI: 10.1038/nature16961. URL: http://www.nature.com/articles/nature16961.

[3] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, et al. "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis". In: *arXiv:2003.08934 [cs]* (Aug. 2020). arXiv: 2003.08934. URL: http://arxiv.org/abs/2003.08934.

[4] Ricardo Martin-Brualla, Noha Radwan, Mehdi S. M. Sajjadi, et al. "NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections". In: *arXiv:2008.02268 [cs]* (Aug. 2020). arXiv: 2008.02268. URL: http://arxiv.org/abs/2008.02268.

[5] Qian Yu, Yongxin Yang, Feng Liu, et al. "Sketch-a-Net: A Deep Neural Network that Beats Humans". en. In: *International Journal of Computer Vision* 122.3 (May 2017), pp. 411–425. ISSN: 0920-5691, 1573-1405. DOI: 10.1007/s11263-016-0932-3. URL: http://link.springer.com/10.1007/s11263-016-0932-3.

[6] Davide Castelvecchi. "Can we open the black box of AI?" en. In: *Nature* 538.7623 (Oct. 2016), pp. 20–23. ISSN: 0028-0836, 1476-4687. DOI: 10.1038/538020a. URL: http://www.nature.com/articles/538020a.

[7] Cynthia Rudin. "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead". en.

In: *Nature Machine Intelligence* 1.5 (May 2019), pp. 206–215. ISSN: 2522-5839. DOI: 10.1038/s42256-019-0048-x. URL: http://www.nature.com/articles/s42256-019-0048-x.

[8] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, et al. "Towards Deep Learning Models Resistant to Adversarial Attacks". In: *arXiv:1706.06083 [cs, stat]* (Sept. 2019). arXiv: 1706.06083. URL: http://arxiv.org/abs/1706.06083.

[9] Kasey Panetta. *5 Trends Drive the Gartner Hype Cycle for Emerging Technologies, 2020*. Englisch. Aug. 2020. URL: https://www.gartner.com/smarterwithgartner/5-trends-drive-the-gartner-hype-cycle-for-emerging-technologies-2020/.

[10] Carlos A Escobar and Ruben Morales-Menendez. "Machine learning techniques for quality control in high conformance manufacturing environment". en. In: *Advances in Mechanical Engineering* 10.2 (Feb. 2018), p. 168781401875551. ISSN: 1687-8140, 1687-8140.

[11] Michael Sharp, Ronay Ak, and Thomas Hedberg. "A survey of the advancing use and development of machine learning in smart manufacturing". en. In: *Journal of Manufacturing Systems* 48 (July 2018), pp. 170–179. ISSN: 02786125. DOI: 10.1016/j.jmsy.2018.02.004. URL: https://linkinghub.elsevier.com/retrieve/pii/S0278612518300153.

[12] Mohammad Saeid Mahdavinejad, Mohammadreza Rezvan, Mohammadamin Barekatain, et al. "Machine learning for internet of things data analysis: a survey". en. In: *Digital Communications and Networks* 4.3 (Aug. 2018), pp. 161–175. ISSN: 23528648. DOI: 10.1016/j.dcan.2017.10.002. URL: https://linkinghub.elsevier.com/retrieve/pii/S235286481730247X.

[13] Xiang Wan and Nadia R. Sanders. "The negative impact of product variety: Forecast bias, inventory levels, and the role of vertical integration". en. In: *International Journal of Production Economics* 186 (Apr. 2017), pp. 123–131. ISSN: 09255273. DOI: 10.1016/j.ijpe.2017.02.002. URL: https://linkinghub.elsevier.com/retrieve/pii/S0925527317300312.

[14] European Union - General Secretary. *European Commission Report on the Impact of Demographic Change*. English. Tech. rep. 1. Bruxxel: European Union, June 2020, p. 30. URL: https://ec.europa.eu/info/sites/info/files/demography_report_2020_n.pdf.

[15] Cornelio Yanez-Marquez. "Toward the Bleaching of the Black Boxes: Minimalist Machine Learning". In: *IT Professional* 22.4 (July 2020), pp. 51–56. ISSN: 1520-9202, 1941-045X. DOI: 10.1109/MITP.2020.2994188. URL: https://ieeexplore.ieee.org/document/9143261/.

[16] Tom M Mitchell and others. "Machine learning. 1997". In: *Burr Ridge, IL: McGraw Hill* 45.37 (1997), pp. 870–877.

[17] E. H. Simpson. "The Interpretation of Interaction in Contingency Tables". en. In: *Journal of the Royal Statistical Society: Series B (Methodological)* 13.2 (July 1951), pp. 238–241. ISSN: 00359246. DOI: 10.1111/j.2517-6161.1951.tb00088.x. URL: http://doi.wiley.com/10.1111/j.2517-6161.1951.tb00088.x.

[18] Dietrich Braess, Anna Nagurney, and Tina Wakolbinger. "On a Paradox of Traffic Planning". en. In: *Transportation Science* 39.4 (Nov. 2005), pp. 446–450. ISSN: 0041-1655, 1526-5447. DOI: 10.1287/trsc.1050.0127. URL: http://pubsonline.informs.org/doi/abs/10.1287/trsc.1050.0127.

[19] Francisco J. Valverde-Albacete and Carmen Peláez-Moreno. "100% Classification Accuracy Considered Harmful: The Normalized Information Transfer Factor Explains the Accuracy Paradox". en. In: *PLoS ONE* 9.1 (Jan. 2014). Ed. by Matteo G. A. Paris, e84217. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0084217. URL: https://dx.plos.org/10.1371/journal.pone.0084217.

[20] Tom Dietterich. "Overfitting and undercomputing in machine learning". In: *ACM computing surveys (CSUR)* 27.3 (1995). Publisher: ACM New York, NY, USA, pp. 326–327.

[21] J.J. Thomas and K.A. Cook. "A visual analytics agenda". In: *IEEE Computer Graphics and Applications* 26.1 (Jan. 2006), pp. 10–13. ISSN: 0272-1716. DOI: 10.1109/MCG.2006.5. URL: http://ieeexplore.ieee.org/document/1573625/.

[22] Zhuwei Qin, Fuxun Yu, Chenchen Liu, et al. "How convolutional neural network see the world - A survey of convolutional neural network visualization methods". In: *arXiv:1804.11191 [cs]* (May 2018). arXiv: 1804.11191. URL: http://arxiv.org/abs/1804.11191.

[23] Fred Hohman, Minsuk Kahng, Robert Pienta, et al. "Visual Analytics in Deep Learning: An Interrogative Survey for the Next Frontiers". In: *IEEE Transactions on Visualization and Computer Graphics* 25.8 (Aug. 2019), pp. 2674–2693. ISSN: 1077-2626, 1941-0506, 2160-9306. DOI: 10.1109/TVCG.2018.2843369. URL: https://ieeexplore.ieee.org/document/8371286/.

[24] Hani Hagras. "Toward Human-Understandable, Explainable AI". In: *Computer* 51.9 (Sept. 2018), pp. 28–36. ISSN: 0018-9162, 1558-0814. DOI: 10.1109/MC.2018.3620965. URL: https://ieeexplore.ieee.org/document/8481251/.

[25] David Gunning. "Explainable artificial intelligence (xai)". In: *Defense Advanced Research Projects Agency (DARPA), nd Web* 2.2 (2017).

[26] John Robert Taylor. *An Introduction to Error Analysis: The Study of Uncertainties in Physical Measurements*. en. Google-Books-ID: giFQcZub80oC. University Science Books, Jan. 1997. ISBN: 978-0-935702-75-0.

[27] Miguel A. Martínez, Eva M. Valero, and Javier Hernández-Andrés. "Adaptive exposure estimation for high dynamic range imaging applied to natural scenes and daylight skies". en. In: *Applied Optics* 54.4 (Feb. 2015), B241. ISSN: 1559-128X, 2155-3165. DOI: 10.1364/AO.54.00B241. URL: https://www.osapublishing.org/abstract.cfm?URI=ao-54-4-B241.

[28] Gracious Ngaile and Taylan Altan. "Simulations of manufacturing processes: Past, present, and future". In: *Advanced technology of plasticity 2002—Proceedings of the 7th ICTP* 1 (2002), pp. 271–282. URL: https://ercnsm.osu.edu/sites/ercnsm.osu.edu/files/uploads/F_genereal/505.pdf.

[29] Don E. Bray and Roderick K. Stanley. *Nondestructive evaluation: a tool in design, manufacturing and service*. CRC press, 1996. URL: https://books.google.de/books?hl=de&lr=&id=MKrpYoR6sHIC&oi=fnd&pg=PA1&dq=evaluation+manufacturing&ots=bztTfJBpct&sig=n-9lZ_nY2Yq8L9UTiCQYci6Nmik.

[30] Norbert FM Roozenburg and Johannes Eekels. *Product design: fundamentals and methods*. Vol. 2. Wiley Chichester, 1995. URL: http://tocs.ulb.tu-darmstadt.de/34922482.pdf.

[31] Faker Zouaoui and James R. Wilson. "Accounting for input model and parameter uncertainty in simulation". In: *Proceedings of the 33nd conference on Winter simulation*. IEEE Computer Society, 2001, pp. 290–299. URL: http://dl.acm.org/citation.cfm?id=564165.

[32] Faker Zouaoui and James R. Wilson. "Accounting for parameter uncertainty in simulation input modeling". In: *Iie Transactions* 35.9 (2003), pp. 781–792. URL: http://www.tandfonline.com/doi/abs/10.1080/07408170304413.

[33] Russell R. Barton, Barry L. Nelson, and Wei Xie. "Quantifying input uncertainty via simulation confidence intervals". In: *INFORMS journal on computing* 26.1 (2013), pp. 74–87. URL: http://pubsonline.informs.org/doi/abs/10.1287/ijoc.2013.0548.

[34] Tiefang Zou, Ming Cai, Ronghua Du, et al. "Analyzing the uncertainty of simulation results in accident reconstruction with Response Surface Methodology". In: *Forensic science international* 216.1 (2012), pp. 49–60. URL: http://www.sciencedirect.com/science/article/pii/S0379073811004099.

[35] Michael D. McKay, John D. Morrison, and Stephen C. Upton. "Evaluating prediction uncertainty in simulation models". In: *Computer Physics Communications* 117.1-2 (1999), pp. 44–51.

[36] Xiaoping Du and Wei Chen. "Methodology for managing the effect of uncertainty in simulation-based design". In: *AIAA journal* 38.8 (2000), pp. 1471–1478. URL: http://arc.aiaa.org/doi/abs/10.2514/2.1125.

[37] Richard G. Hills. "Model validation: model parameter and measurement uncertainty". In: *Journal of Heat Transfer* 128.4 (2006), pp. 339–351. URL: http://proceedings.asmedigitalcollection.asme.org/article.aspx?articleid=1448314.

[38] Shun-Feng Su and CS George Lee. "Manipulation and propagation of uncertainty and verification of applicability of actions in assembly tasks". In: *IEEE Transactions on Systems, Man, and Cybernetics* 22.6 (1992), pp. 1376–1389. URL: http://ieeexplore.ieee.org/abstract/document/199463/.

[39] Kristin Potter, Paul Rosen, and Chris R. Johnson. "From quantification to visualization: A taxonomy of uncertainty visualization approaches". In: *Uncertainty Quantification in Scientific Computing*. Springer, 2012, pp. 226–249. URL: http://link.springer.com/chapter/10.1007/978-3-642-32677-6_15.

[40] Fei Tao, Jiangfeng Cheng, Qinglin Qi, et al. "Digital twin-driven product design, manufacturing and service with big data". en. In: *The International Journal of Advanced Manufacturing Technology* 94.9-12 (Feb. 2018), pp. 3563–3576. ISSN: 0268-3768, 1433-3015. DOI: 10.1007/s00170-017-0233-1. URL: http://link.springer.com/10.1007/s00170-017-0233-1.

[41] Pak Ki Kwok, Mian Yan, Ting Qu, et al. "User acceptance of virtual reality technology for practicing digital twin-based crisis management". en. In: *International Journal of Computer Integrated Manufacturing* (Aug. 2020), pp. 1–14. ISSN: 0951-192X, 1362-3052. DOI: 10.1080/0951192X.2020.1803502. URL: https://www.tandfonline.com/doi/full/10.1080/0951192X.2020.1803502.

[42] Luca Liliana. "A new model of Ishikawa diagram for quality assessment". In: *IOP Conference Series: Materials Science and Engineering* 161 (Nov. 2016), p. 012099. ISSN: 1757-8981, 1757-899X. DOI: 10.1088/1757-899X/161/1/012099. URL: https://iopscience.iop.org/article/10.1088/1757-899X/161/1/012099.

[43] Manfred Roza, Jeroen Voogd, and Derek Sebalj. "The Generic Methodology for Verification and Validation to support acceptance of models, simulations and data". en. In: *The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology* 10.4 (Oct. 2013), pp. 347–365. ISSN: 1548-5129, 1557-380X. DOI: 10.1177/1548512912459688.

[44] Patrick Ruediger and Hans Hagen. "Dealing with Uncertainties in Manufacturing Process Simulations". In: *Applied Mechanics and Materials*. Vol. 869. Trans Tech Publ, 2017, pp. 226–233.

[45] Paul De Bièvre. "The 2007 International Vocabulary of Metrology (VIM), JCGM 200:2008 [ISO/IEC Guide 99]: Meeting the need for intercontinentally understood concepts and their associated intercontinentally agreed terms". en. In: *Clinical Biochemistry* 42.4-5 (Mar. 2009), pp. 246–248. ISSN: 00099120. DOI: 10.1016/j.clinbiochem.2008.09.007. URL: https://linkinghub.elsevier.com/retrieve/pii/S0009912008003949.

[46] Cunbo Zhuang, Jianhua Liu, and Hui Xiong. "Digital twin-based smart production management and control framework for the complex product assembly shop-floor". en. In: *The International Journal of Advanced Manufacturing Technology* 96.1-4 (Apr. 2018), pp. 1149–1163. ISSN: 0268-3768, 1433-3015. DOI: 10.1007/s00170-018-1617-6. URL: http://link.springer.com/10.1007/s00170-018-1617-6.

[47] Kaoru Ishikawa. *Guide to quality control*. TS156. I3713 1994. 1982.

[48] VDI-Fachbereich Fabrikplanung und betrieb. *VDI 2870 - Lean production systems - List of methods*. Vol. Blatt 2. VDI 2870. VDI-Gesellschaft Produktion und Logistik, 2013.

[49] Tom Drozda, Charles Wick, John T. Benedict, et al., eds. *Tool and manufacturing engineers handbook: a reference book for manufacturing engineers, managers, and technicians*. 4th ed. Dearborn, Mich: Society of Manufacturing Engineers, 1983. ISBN: 978-0-87263-085-7.

[50] Robert C. Martin. *Agile software development: principles, patterns, and practices*. Alan Apt series. Upper Saddle River, N.J: Prentice Hall, 2003. ISBN: 978-0-13-597444-5.

[51] W. Duch. "Coloring black boxes: visualization of neural network decisions". In: *Proceedings of the International Joint Conference on Neural Networks, 2003.* Vol. 3. Portland, OR, USA: IEEE, 2003, pp. 1735–1740. ISBN: 978-0-7803-7898-8. DOI: 10.1109/IJCNN.2003.1223669. URL: http://ieeexplore.ieee.org/document/1223669/.

[52] Fan-Yin Tzeng and Kwan-Liu Ma. "Opening the Black Box - Data Driven Visualization of Neural Networks". In: *VIS 05. IEEE Visualization, 2005.* Minneapolis, MN, USA: IEEE, 2005, pp. 383–390. ISBN: 978-0-7803-9462-9. DOI: 10.1109/VISUAL.2005.1532820. URL: http://ieeexplore.ieee.org/document/1532820/.

[53] Wojciech Samek, Alexander Binder, Gregoire Montavon, et al. "Evaluating the Visualization of What a Deep Neural Network Has Learned". In: *IEEE Transactions on Neural Networks and Learning Systems* 28.11 (Nov. 2017), pp. 2660–2673. ISSN: 2162-237X, 2162-2388. DOI: 10.1109/TNNLS.2016.2599820. URL: http://ieeexplore.ieee.org/document/7552539/.

[54] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps". In: *arXiv:1312.6034 [cs]* (Apr. 2014). arXiv: 1312.6034. URL: http://arxiv.org/abs/1312.6034.

[55] Matthew D. Zeiler and Rob Fergus. "Visualizing and Understanding Convolutional Networks". en. In: *Computer Vision – ECCV 2014*. Ed. by David Fleet, Tomas Pajdla, Bernt Schiele, et al. Vol. 8689. Series Title: Lecture Notes in Computer Science. Cham: Springer International Publishing, 2014, pp. 818–833. ISBN: 978-3-319-10589-5 978-3-319-10590-1. DOI: 10.1007/978-3-319-10590-1_53. URL: http://link.springer.com/10.1007/978-3-319-10590-1_53.

[56] Sebastian Bach, Alexander Binder, Grégoire Montavon, et al. "On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation". en. In: *PLOS ONE* 10.7 (July 2015). Ed. by Oscar Deniz Suarez, e0130140. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0130140. URL: https://dx.plos.org/10.1371/journal.pone.0130140.

[57] Daniel Smilkov, Shan Carter, D. Sculley, et al. "Direct-Manipulation Visualization of Deep Networks". In: *arXiv:1708.03788 [cs, stat]* (Aug. 2017). arXiv: 1708.03788. URL: http://arxiv.org/abs/1708.03788.

[58] Luisa M. Zintgraf, Taco S. Cohen, Tameem Adel, et al. "Visualizing Deep Neural Network Decisions: Prediction Difference Analysis". In: *arXiv:1702.04595 [cs]* (Feb. 2017). arXiv: 1702.04595. URL: http://arxiv.org/abs/1702.04595.

[59] Angelos Chatzimparmpas, Rafael M. Martins, Kostiantyn Kucher, et al. "StackGenVis: Alignment of Data, Algorithms, and Models for Stacking Ensemble Learning Using Performance Metrics". In: *arXiv:2005.01575 [cs, stat]* (Sept. 2020). arXiv: 2005.01575. URL: http://arxiv.org/abs/2005.01575.

[60] Alon Brutzkus, Amir Globerson, Eran Malach, et al. "SGD Learns Over-parameterized Networks that Provably Generalize on Linearly Separable Data". In: *arXiv:1710.10174 [cs]* (Oct. 2017). arXiv: 1710.10174. URL: http://arxiv.org/abs/1710.10174.

[61] Alexandr Andoni, Rina Panigrahy, Gregory Valiant, et al. "Learning Polynomials with Neural Networks". In: *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*. ICML'14. event-place: Beijing, China. JMLR.org, 2014, pp. II–1908–II–1916.

[62] Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. "In Search of the Real Inductive Bias: On the Role of Implicit Regularization in Deep Learning". In: *arXiv:1412.6614 [cs, stat]* (Apr. 2015). arXiv: 1412.6614. URL: http://arxiv.org/abs/1412.6614.

[63] Chiyuan Zhang, Samy Bengio, Moritz Hardt, et al. "Understanding deep learning requires rethinking generalization". In: *arXiv:1611.03530 [cs]* (Feb. 2017). arXiv: 1611.03530. URL: http://arxiv.org/abs/1611.03530.

[64] Fangfang Zhou, Xiaoru Lin, Chang Liu, et al. "A survey of visualization for smart manufacturing". en. In: *Journal of Visualization* 22.2 (Apr. 2019), pp. 419–435. ISSN: 1343-8875, 1875-8975. DOI: 10.1007/s12650-018-0530-2. URL: http://link.springer.com/10.1007/s12650-018-0530-2.

[65] Panpan Xu, Honghui Mei, Liu Ren, et al. "ViDX: Visual Diagnostics of Assembly Line Performance in Smart Factories". In: *IEEE Transactions on Visualization and Computer Graphics* 23.1 (Jan. 2017), pp. 291–300. ISSN: 1077-2626. DOI: 10.1109/TVCG.2016.2598664. URL: http://ieeexplore.ieee.org/document/7536610/.

[66] Devarajan Ramanujan and William Z. Bernstein. "VESPER: Visual Exploration of Similarity and Performance Metrics for Computer-Aided Design Repositories". In: *Volume 3: Manufacturing Equipment and Systems*. College Station, Texas, USA: American Society of Mechanical Engineers, June 2018. ISBN: 978-0-7918-5137-1. DOI: 10.1115/MSEC2018-6527. URL: https://asmedigitalcollection.asme.org/MSEC/proceedings/MSEC2018/51371/College%20Station,%20Texas,%20USA/274092.

[67] Wenchao Wu, Yixian Zheng, Kaiyuan Chen, et al. "A Visual Analytics Approach for Equipment Condition Monitoring in Smart Factories of Process Industry". In: *2018 IEEE Pacific Visualization Symposium (PacificVis)*. Kobe: IEEE, Apr. 2018, pp. 140–149. ISBN: 978-1-5386-1424-2. DOI: 10.1109/PacificVis.2018.00026. URL: https://ieeexplore.ieee.org/document/8365986/.

[68] Weixiang Sun, Jin Chen, and Jiaqing Li. "Decision tree and PCA-based fault diagnosis of rotating machinery". en. In: *Mechanical Systems and Signal Processing* 21.3 (Apr. 2007), pp. 1300–1317. ISSN: 08883270. DOI: 10.1016/j.ymssp.2006.06.010. URL: https://linkinghub.elsevier.com/retrieve/pii/S0888327006001336.

[69] Liu Jiang, Shixia Liu, and Changjian Chen. "Recent research advances on interactive machine learning". en. In: *Journal of Visualization* 22.2 (Apr. 2019), pp. 401–417. ISSN: 1343-8875, 1875-8975. DOI: 10.1007/s12650-018-0531-1. URL: http://link.springer.com/10.1007/s12650-018-0531-1.

[70] Mihaela Jarema, Ismail Demir, Johannes Kehrer, et al. "Comparative visual analysis of vector field ensembles". In: *2015 IEEE Conference on Visual Analytics Science and Technology (VAST)*. Chicago, IL, USA: IEEE, Oct. 2015, pp. 81–88. ISBN: 978-1-4673-9783-4. DOI: 10.1109/VAST.2015.7347634. URL: http://ieeexplore.ieee.org/document/7347634/.

[71] B. Rieck and H. Leitte. "Exploring and Comparing Clusterings of Multivariate Data Sets Using Persistent Homology". en. In: *Computer Graphics Forum* 35.3 (June 2016), pp. 81–90. ISSN: 01677055. DOI: 10.1111/cgf.12884. URL: http://doi.wiley.com/10.1111/cgf.12884.

[72] Alexander Kumpf, Bianca Tost, Marlene Baumgart, et al. "Visualizing Confidence in Cluster-Based Ensemble Weather Forecast Analyses". In: *IEEE Transactions on Visualization and Computer Graphics* 24.1 (Jan. 2018), pp. 109–119. ISSN: 1077-2626. DOI: 10.1109/TVCG.2017.2745178. URL: http://ieeexplore.ieee.org/document/8019883/.

[73] F. Pedregosa, G. Varoquaux, A. Gramfort, et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[74] Jacob Goldberger, Geoffrey E Hinton, Sam T Roweis, et al. "Neighbourhood components analysis". In: *Advances in neural information processing systems*. 2005, pp. 513–520.

[75] Piotr Indyk and Rajeev Motwani. "Approximate nearest neighbors: towards removing the curse of dimensionality". en. In: *Proceedings of the thirtieth annual ACM symposium on Theory of computing - STOC*

*'98*. Dallas, Texas, United States: ACM Press, 1998, pp. 604–613. ISBN: 978-0-89791-962-3. DOI: 10.1145/276698.276876. URL: http://portal.acm.org/citation.cfm?doid=276698.276876.

[76] Kenneth L. Clarkson. "An algorithm for approximate closest-point queries". en. In: *Proceedings of the tenth annual symposium on Computational geometry - SCG '94*. Stony Brook, New York, United States: ACM Press, 1994, pp. 160–164. ISBN: 978-0-89791-648-6. DOI: 10.1145/177424.177609. URL: http://portal.acm.org/citation.cfm?doid=177424.177609.

[77] Alex J. Smola and Bernhard Schölkopf. *A tutorial on support vector regression*. 2004.

[78] Carl Edward Rasmussen. "Gaussian Processes in Machine Learning". In: *Advanced Lectures on Machine Learning*. Ed. by Olivier Bousquet, Ulrike von Luxburg, and Gunnar Rätsch. Vol. 3176. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 63–71. ISBN: 978-3-540-23122-6 978-3-540-28650-9. DOI: 10.1007/978-3-540-28650-9_4. URL: http://link.springer.com/10.1007/978-3-540-28650-9_4.

[79] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. "Additive Models, Trees, and Related Methods". In: *The Elements of Statistical Learning*. New York, NY: Springer New York, 2009, pp. 295–336. ISBN: 978-0-387-84857-0 978-0-387-84858-7. DOI: 10.1007/978-0-387-84858-7_9. URL: http://link.springer.com/10.1007/978-0-387-84858-7_9.

[80] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. "Neural Networks". In: *The Elements of Statistical Learning*. New York, NY: Springer New York, 2009, pp. 389–416. ISBN: 978-0-387-84857-0 978-0-387-84858-7. DOI: 10.1007/978-0-387-84858-7_11. URL: http://link.springer.com/10.1007/978-0-387-84858-7_11.

[81] John C. Platt. "Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods". In: *ADVANCES IN LARGE MARGIN CLASSIFIERS*. MIT Press, 1999, pp. 61–74.

[82] Jerry L Hintze and Ray D Nelson. "Violin plots: a box plot-density trace synergism". In: *The American Statistician* 52.2 (1998), pp. 181–184.

[83] Zhiyuan Zhang, Kevin T. McDonnell, Erez Zadok, et al. "Visual Correlation Analysis of Numerical and Categorical Data on the Correlation Map". In: *IEEE Transactions on Visualization and Computer Graphics* 21.2 (Feb. 2015), pp. 289–303. ISSN: 1077-2626, 1941-0506, 2160-9306. DOI: 10.1109/TVCG.2014.2350494. URL: https://ieeexplore.ieee.org/document/6881685/.

[84] L. Shao, A. Mahajan, T. Schreck, et al. "Interactive Regression Lens for Exploring Scatter Plots". en. In: *Computer Graphics Forum* 36.3 (June 2017), pp. 157–166. ISSN: 0167-7055, 1467-8659. DOI: 10.1111/cgf.13176. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.13176.

[85] Jun Wang and Klaus Mueller. "The Visual Causality Analyst: An Interactive Interface for Causal Reasoning". In: *IEEE Transactions on Visualization and Computer Graphics* 22.1 (Jan. 2016), pp. 230–239. ISSN: 1077-2626. DOI: 10.1109/TVCG.2015.2467931. URL: http://ieeexplore.ieee.org/document/7192729/.

[86] Chris Bryan, Xue Wu, Susan Mniszewski, et al. "Integrating predictive analytics into a spatiotemporal epidemic simulation". In: *2015 IEEE Conference on Visual Analytics Science and Technology (VAST)*. Chicago, IL, USA: IEEE, Oct. 2015, pp. 17–24. ISBN: 978-1-4673-9783-4. DOI: 10.1109/VAST.2015.7347626. URL: http://ieeexplore.ieee.org/document/7347626/.

[87] T. Bui-Thanh, M. Damodaran, and K. Willcox. "Aerodynamic Data Reconstruction and Inverse Design Using Proper Orthogonal Decomposition". en. In: *AIAA Journal* 42.8 (Aug. 2004), pp. 1505–1516. ISSN: 0001-1452, 1533-385X. DOI: 10.2514/1.2159. URL: https://arc.aiaa.org/doi/10.2514/1.2159.

[88] Stefano Gaggero, Giuliano Vernengo, Diego Villa, et al. "A reduced order approach for optimal design of efficient marine propellers". en. In: *Ships and Offshore Structures* 15.2 (Feb. 2020), pp. 200–214. ISSN: 1744-5302, 1754-212X. DOI: 10.1080/17445302.2019.1606877. URL: https://www.tandfonline.com/doi/full/10.1080/17445302.2019.1606877.

[89] Renee Swischuk, Laura Mainini, Benjamin Peherstorfer, et al. "Projection-based model reduction: Formulations for physics-based machine learning". en. In: *Computers & Fluids* 179 (Jan. 2019), pp. 704–717. ISSN: 00457930. DOI: 10.1016/j.compfluid.2018.07.021.

[90] Kuan Lu, Yulin Jin, Yushu Chen, et al. "Review for order reduction based on proper orthogonal decomposition and outlooks of applications in mechanical systems". en. In: *Mechanical Systems and Signal Processing* 123 (May 2019), pp. 264–297. ISSN: 08883270. DOI: 10.1016/j.ymssp.2019.01.018. URL: https://linkinghub.elsevier.com/retrieve/pii/S0888327019300184.

[91] S. Freitag, B.T. Cao, J. Ninić, et al. "Recurrent neural networks and proper orthogonal decomposition with interval data for real-time predictions of mechanised tunnelling processes". en. In: *Computers & Structures* 207 (Sept. 2018), pp. 258–273. ISSN: 00457949. DOI: 10.1016/j.compstruc.2017.03.020. URL: https://linkinghub.elsevier.com/retrieve/pii/S0045794917302195.

[92] Dong Hyun Jeong, Caroline Ziemkiewicz, Brian Fisher, et al. "iPCA: An Interactive System for PCA-based Visual Analytics". en. In: *Computer Graphics Forum* 28.3 (June 2009), pp. 767–774. ISSN: 01677055, 14678659. DOI: 10.1111/j.1467-8659.2009.01475.x. URL: http://doi.wiley.com/10.1111/j.1467-8659.2009.01475.x.

[93] Stephen V. Stehman. "Selecting and interpreting measures of thematic classification accuracy". en. In: *Remote Sensing of Environment* 62.1 (Oct. 1997), pp. 77–89. ISSN: 00344257. DOI: 10.1016/S0034-4257(97)00083-7. URL: https://linkinghub.elsevier.com/retrieve/pii/S0034425797000837.

[94] David Martin Powers. "Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation". In: (2011).

[95] Tom Fawcett. "An introduction to ROC analysis". en. In: *Pattern Recognition Letters* 27.8 (June 2006), pp. 861–874. ISSN: 01678655. DOI: 10.1016/j.patrec.2005.10.010. URL: https://linkinghub.elsevier.com/retrieve/pii/S016786550500303X.

[96] Patrick Ruediger, Felix Claus, Bernd Hamann, et al. "Combining Visual Analytics and Machine Learning for Reverse Engineering in Assembly Quality Control". en. In: *Journal of Imaging Science and Technology* 64.6 (Nov. 2020), pp. 60405–1–60405–13. ISSN: 1062-3701. DOI: 10.2352/J.ImagingSci.Technol.2020.64.6.060405. URL: https://www.ingentaconnect.com/content/10.2352/J.ImagingSci.Technol.2020.64.6.060405 (visited on 01/28/2021).

[97] David H Wolpert, William G Macready, and others. *No free lunch theorems for search*. Tech. rep. SFI-TR-95-02-010. Santa Fe Institute, 1995.

[98] Shawkat Ali and Kate A. Smith. "On learning algorithm selection for classification". en. In: *Applied Soft Computing* 6.2 (Jan. 2006), pp. 119–138. ISSN: 15684946. DOI: 10.1016/j.asoc.2004.12.002.

[99] Wei Chen, Ruichen Jin, and Agus Sudjianto. "Analytical Variance-Based Global Sensitivity Analysis in Simulation-Based Design Under Uncertainty". en. In: *Journal of Mechanical Design* 127.5 (2005), p. 875. ISSN: 10500472. DOI: 10.1115/1.1904642.

[100] Julien Tierny, Guillaume Favelier, Joshua A. Levine, et al. "The Topology ToolKit". In: *IEEE transactions on visualization and computer graphics* 24.1 (2018), pp. 832–842.

[101] James Ahrens, Sébastien Jourdain, Patrick O'Leary, et al. "An Image-based Approach to Extreme Scale in Situ Visualization and Analysis". In: *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. SC '14. event-place: New Orleans, Louisana. Piscataway, NJ, USA: IEEE Press, 2014, pp. 424–434. ISBN: 978-1-4799-5500-8. DOI: 10.1109/SC.2014.40. URL: https://doi.org/10.1109/SC.2014.40.

[102] S. Lloyd. "Least squares quantization in PCM". en. In: *IEEE Transactions on Information Theory* 28.2 (Mar. 1982), pp. 129–137. ISSN: 0018-9448. DOI: 10.1109/TIT.1982.1056489. URL: http://ieeexplore.ieee.org/document/1056489/.

[103] Dan Pelleg and Andrew Moore. "Accelerating exact *k* -means algorithms with geometric reasoning". en. In: *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '99*. San Diego, California, United States: ACM Press, 1999, pp. 277–281. ISBN: 978-1-58113-143-7. DOI: 10.1145/312129.312248. URL: http://portal.acm.org/citation.cfm?doid=312129.312248.

[104] Akira Okubo. "Horizontal dispersion of floatable particles in the vicinity of velocity singularities such as convergences". en. In: *Deep Sea Research and Oceanographic Abstracts* 17.3 (June 1970), pp. 445–454. ISSN: 00117471. DOI: 10.1016/0011-7471(70)90059-8.

[105] John Weiss. "The dynamics of enstrophy transfer in two-dimensional hydrodynamics". en. In: *Physica D: Nonlinear Phenomena* 48.2-3 (Mar. 1991), pp. 273–294. ISSN: 01672789. DOI: 10.1016/0167-2789(91)90088-Q. URL: https://linkinghub.elsevier.com/retrieve/pii/016727899190088Q.

[106] Peng Xiu, Fei Chai, Lei Shi, et al. "A census of eddy activities in the South China Sea during 1993–2007". en. In: *Journal of Geophysical Research* 115.C3 (Mar. 2010). ISSN: 0148-0227. DOI: 10.1029/2009JC005657. URL: http://doi.wiley.com/10.1029/2009JC005657.

[107] Jordi Isern-Fontanet, Jordi Font, Emilio García-Ladona, et al. "Spatial structure of anticyclonic eddies in the Algerian basin (Mediterranean Sea) analyzed using the Okubo–Weiss parameter". en. In: *Deep Sea Research Part II: Topical Studies in Oceanography* 51.25-26 (Dec. 2004), pp. 3009–3028. ISSN: 09670645. DOI: 10.1016/j.dsr2.2004.09.013. URL: https://linkinghub.elsevier.com/retrieve/pii/S0967064504002085.

[108] Alexis Chaigneau, Arnaud Gizolme, and Carmen Grados. "Mesoscale eddies off Peru in altimeter records: Identification algorithms and eddy spatio-temporal patterns". en. In: *Progress in Oceanography* 79.2-4 (Oct. 2008), pp. 106–119. ISSN: 00796611. DOI: 10.1016/j.pocean.2008.10.013. URL: https://linkinghub.elsevier.com/retrieve/pii/S0079661108001663.

[109] I. Frenger, N. Gruber, R. Knutti, et al. "Imprint of Southern Ocean eddies on winds, clouds and rainfall". en. In: *Nature Geoscience* 6.8 (Aug. 2013), pp. 608–612. ISSN: 1752-0894, 1752-0908. DOI: 10.1038/ngeo1863. URL: http://www.nature.com/articles/ngeo1863.

[110] Jan Sahner, Tino Weinkauf, Nathalie Teuber, et al. "Vortex and Strain Skeletons in Eulerian and Lagrangian Frames". In: *IEEE Transactions on Visualization and Computer Graphics* 13.5 (Sept. 2007), pp. 980–990. ISSN: 1077-2626. DOI: 10.1109/TVCG.2007.1053. URL: http://ieeexplore.ieee.org/document/4276078/.

[111] Marston Morse. *The calculus of variations in the large*. Vol. 18. American Mathematical Soc., 1934.

[112] Janine C. Bennett, Hank Childs, Christoph Garth, et al. "In Situ Visualization for Computational Science (Dagstuhl Seminar 18271)". In: *Dagstuhl Reports* 8.7 (2019). Ed. by Janine C. Bennett, Hank Childs,

Christoph Garth, et al. Place: Dagstuhl, Germany Publisher: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, pp. 1–43. ISSN: 2192-5283. DOI: 10.4230/DagRep.8.7.1. URL: http://drops.dagstuhl.de/opus/volltexte/2019/10171.

[113] James Ahrens. "Implications of numerical and data intensive technology trends on scientific visualization and analysis". In: *The 2015 SIAM Computational Science and Engineering (CSE)*. 2015.

[114] Patrick O'Leary, James Ahrens, Sébastien Jourdain, et al. "Cinema image-based in situ analysis and visualization of MPAS-ocean simulations". In: *Parallel Computing* 55 (2016), pp. 43 –48. ISSN: 0167-8191. DOI: https://doi.org/10.1016/j.parco.2015.10.005. URL: http://www.sciencedirect.com/science/article/pii/S0167819115001349.

[115] S. Li, N. Marsaglia, C. Garth, et al. "Data Reduction Techniques for Simulation, Visualization and Data Analysis". In: *Computer Graphics Forum* 37.6 (2018), pp. 422–447. DOI: 10.1111/cgf.13336.

[116] M. Hummel, R. Bujack, K. I. Joy, et al. "Error Estimates for Lagrangian Flow Field Representations". In: *Proceedings of the Eurographics / IEEE VGTC Conference on Visualization: Short Papers*. EuroVis '16. event-place: Groningen, The Netherlands. Goslar Germany, Germany: Eurographics Association, 2016, pp. 7–11. DOI: 10.2312/eurovisshort.20161153. URL: https://doi.org/10.2312/eurovisshort.20161153.

[117] Allison H. Baker, Dorit M. Hammerling, and Terece L. Turton. "Evaluating Image Quality Measures to Assess the Impact of Lossy Data Compression Applied to Climate Simulation Data". en. In: (2019). ISSN: 1467-8659. DOI: 10.1111/cgf.13707. URL: https://diglib.eg.org:443/xmlui/handle/10.1111/cgf13707.

[118] K. Ma. "In Situ Visualization at Extreme Scale: Challenges and Opportunities". In: *IEEE Computer Graphics and Applications* 29.6 (Nov. 2009), pp. 14–19. ISSN: 0272-1716. DOI: 10.1109/MCG.2009.120.

[119] Jonas Lukasczyk, Eric Kinner, James Ahrens, et al. "VOIDGA: A View-Approximation Oriented Image Database Generation Approach". In: *2018 IEEE 8th Symposium on Large Data Analysis and Visualization (LDAV)*. Berlin, Germany: IEEE, Oct. 2018, pp. 12–22. ISBN: 978-1-5386-6873-3.

DOI: 10.1109/LDAV.2018.8739204. URL: https://ieeexplore.ieee.org/document/8739204/.

[120] Anna Tikhonova, Carlos D. Correa, and Kwan-Liu Ma. "Explorable images for visualizing volume data". In: *IEEE Pacific Visualization Symposium PacificVis 2010, Taipei, Taiwan, March 2-5, 2010*. 2010, pp. 177–184. DOI: 10.1109/PACIFICVIS.2010.5429595. URL: https://doi.org/10.1109/PACIFICVIS.2010.5429595.

[121] Tim Biedert and Christoph Garth. "Contour Tree Depth Images For Large Data Visualization". In: *Eurographics Symposium on Parallel Graphics and Visualization, Cagliari, Sardinia, Italy, May 25 - 26, 2015.* 2015, pp. 77–86. DOI: 10.2312/pgv.20151158. URL: https://doi.org/10.2312/pgv.20151158.

[122] J. Woodring, S. Mniszewski, C. Brislawn, et al. "Revisiting wavelet compression for large-scale climate data using JPEG 2000 and ensuring data precision". In: *2011 IEEE Symposium on Large Data Analysis and Visualization*. Oct. 2011, pp. 31–38. DOI: 10.1109/LDAV.2011.6092314.

[123] Anne S. Berres, Terece L. Turton, Mark Petersen, et al. *Video Compression for Ocean Simulation Image Databases*. en. The Eurographics Association, 2017. DOI: 10.2312/envirvis.20171104.

[124] Gary J. Sullivan, Pankaj Topiwala, and Ajay Luthra. "The H.264/AVC Advanced Video Coding standard: overview and introduction to the fidelity range extensions". In: *SPIE Optics + Photonics*. 2004.

[125] Benjamin Bross, Mauricio Alvarez-Mesa, Valeri George, et al. "HEVC real-time decoding". In: *Optics & Photonics - Optical Engineering + Applications*. 2013.

[126] Adrian Grange, Peter de Rivaz, and Jonathan Hunt. "VP9 Bitstream & Decoding Process Specification v0.6". In: (2016), p. 171.

[127] Utkarsh Ayachit, Andrew Bauer, Berk Geveci, et al. "Paraview catalyst: Enabling in situ data analysis and visualization". In: *Proceedings of the First Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization*. ACM, 2015, pp. 25–29.

[128] Joey Blake. *FFmpeg preset files. Contribute to joeyblake/FFmpeg-Presets development by creating an account on GitHub*. May 2019. URL: https://github.com/joeyblake/FFmpeg-Presets.

[129] Volker Gaede and Oliver Günther. "Multidimensional Access Methods". In: *ACM Comput. Surv.* 30.2 (June 1998). Place: New York, NY, USA Publisher: ACM, pp. 170–231. ISSN: 0360-0300. DOI: 10.1145/280277.280279. URL: http://doi.acm.org/10.1145/280277.280279.

[130] Guy M. Morton. "A computer oriented geodetic data base and a new technique in file sequencing". In: (1966).

[131] Patrick Ruediger, Christoph Garth, Hans Hagen, et al. "Is Smaller Always Better? - Evaluating Video Compression Techniques for Simulation Ensembles". In: Open Access Series in Informatics (OASIcs) 89 (2021). Ed. by Christoph Garth, Jan C. Aurich, Barbara Linke, et al. ISSN: 2190-6807, 10:1–10:18. DOI: 10.4230/OASIcs.iPMVM.2020.10. URL: https://drops.dagstuhl.de/opus/volltexte/2021/13759.

[132] Peter Lindstrom. "Fixed-rate compressed floating-point arrays". In: *IEEE transactions on visualization and computer graphics* 20.12 (2014). Publisher: IEEE, pp. 2674–2683.

[133] H. Hagen F. Claus F.-A. Rupprecht. "Online Simulation Considering Production Uncertainties to Improve Assembly Quality". In: NAFEMS, 2019. ISBN: 978-1-910643-52-5. URL: https://www.nafems.org/publications/resource_center/nwc_19_356/.

**PATRICK RUEDIGER-FLORE**

ruediger@cs.uni-kl.de

ORCID:
https://orcid.org/0000-0003-3344-1331

## Professional Experience

| | |
|---|---|
| **07/2020 – today** | **Research Scientist**<br>**FBK - Institute for Manufacturing Technology and Production Systems** |
| **12/2015 – 07/2020** | **Research Scientist**<br>**AG Computer graphics & HCI, TU Kaiserslautern** |
| **10/2013 – 11/2015** | **Research Assistant**<br>**AG Computer graphics & HCI, TU Kaiserslautern** |
| **01/2014 – 07/2014** | **Exchange Scientist**<br>**UC Davis, Dep. of Computer Science, USA** |
| **10/2012 – 10/2014** | **Research Assistant**<br>**Fraunhofer ITWM, AG Optimization** |

## Education

| | |
|---|---|
| **01/2016 – 07/2021** | **TU Kaiserslautern**<br>**Promotion Computer Science** – IRTG 2057<br>Title: Visual Analytics for Machine Learning Performance Evaluation |
| **04/2014 – 12/2015** | **TU Kaiserslautern**<br>**Master of Science – Applied Computer Science**: Production Technologies<br>Title: Image Orientation Estimation of 3D images<br>for digital property computations |
| **10/2011 – 03/2014** | **TU Kaiserslautern**<br>**Bachelor of Science – Applied Computer Science** : Production Technologies<br>Title: Effiziente Therapieplanung bei Brustkrebs,<br>Algorithmik, Datenmodell und Visualisierung |