Analysis of Specular Reflectivity of Thin Films Using Machine Learning

Dissertation

der Mathematisch-Naturwissenschaftlichen Fakultät
der Eberhard Karls Universität Tübingen
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

vorgelegt von
Alessandro Greco
aus Tübingen

Tübingen
2022

Gedruckt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Eberhard Karls Universität Tübingen.

Tag der mündlichen Qualifikation:	29.07.2022
Dekan:	Prof. Dr. Thilo Stehle
1. Berichterstatter:	Prof. Dr. Frank Schreiber
2. Berichterstatter:	Prof. Dr. Martin Oettel
3. Berichterstatter:	Prof. Dr. Yuri Gerelli

Abstract

X-ray and neutron scattering encompass a large variety of complementary and non-invasive measurement techniques that are used to study a large range of materials. The continued improvements of X-ray and neutron sources, as well as advancements in detector technologies have enabled the development of sophisticated measurement setups with the ability to gather large amounts of information. These modern techniques tend to produce a lot of data, which is typically analyzed using theoretical models. Increasingly, however, the rate at which data is produced is outpacing the rate at which it can be analyzed. To fully exploit the potential of these techniques and avoid bottlenecks in scientific productivity, equally advanced methods of data analysis must be developed. In recent years, machine learning, specifically deep neural networks (NNs), have emerged as a promising solution for this, since their data-driven heuristic models can often process data many times faster than conventional methods.

The research in this work focuses on the first published application of NNs for the analysis of specular reflectometry data and demonstrates further improvements of the method. X-ray and neutron reflectometry are commonly used to study various important systems, such as surfaces, interfaces, liquid and solid thin films, layered structures and magnetic materials. As shown in this work, NNs can extract sample properties from reflectometry data within a fraction of a second, which is on par with the high-end speed of modern measurements. This is demonstrated with, but not limited to, organic molecular thin films on silicon substrates. Furthermore, this work discusses the NN performance on different challenging cases and shows methods of successfully dealing with systematic and statistical artifacts in the data. This thesis culminated in the development of *mlreflect*, a Python-based analysis package that implements the achievements of this work and that is available both online and on the Maxwell cluster of the Deutsches Elektronen-Synchrotron. Thus, this work constitutes a significant step towards the goal of fully-automatized reflectivity data analysis and may even serve as a guide for the analysis of other types of scattering data.

Deutsche Zusammenfassung

Röntgen- und Neutronenstreuung umfassen eine Vielzahl von komplementären und nicht-invasiven Messverfahren zur Untersuchung vieler Materialien. Die kontinuierliche Verbesserung von Röntgen- und Neutronenquellen sowie neue Detektortechnologien haben zur stetigen Entwicklung fortschrittlicher Messmethoden geführt, die in der Lage sind große Informationsmengen zu erfassen. Diese Daten werden üblicherweise mit Hilfe theoretischer Modelle analysiert, jedoch übersteigt zunehmend die Geschwindigkeit der Datenerzeugung die der Datenverarbeitung. Um das Potenzial neuer Techniken voll ausschöpfen zu können und Engpässe in der wissenschaftlichen Produktivität zu vermeiden, müssen ebenso fortschrittliche Methoden der Datenanalyse entwickelt werden. In den letzten Jahren hat sich das maschinelle Lernen, insbesondere tiefe neuronale Netze (NNs), als vielversprechende Lösung für diese Aufgabe erwiesen, da ihre datenbasierten, heuristischen Modelle Messdaten oft um ein Vielfaches schneller verarbeiten können als konventionelle Methoden.

Diese Dissertation demonstriert die erste veröffentlichte Anwendung von NNs zur Analyse von spekulären Röntgen- und Neutronenreflektometriemessungen. Diese werden häufig zur Untersuchung wichtiger Systeme wie Oberflächen, Grenzflächen, flüssiger und fester Schichtstrukturen und magnetischer Materialien eingesetzt. Mit Hilfe von NNs können in Sekundenbruchteilen Probeneigenschaften aus Reflektometriedaten extrahiert werden, was der Geschwindigkeit moderner Messmethoden gleichkommt. Dies wird anhand von Dünnschichten organischer Moleküle auf Siliziumsubstraten gezeigt, ist aber nicht auf diese beschränkt. Überdies werden besonders schwierig zu analysierende Daten sowie Methoden zum Umgang mit systematischen und statistischen Fehlern untersucht. Die Ergebnisse dieser Forschung wurden schließlich in einem Python-basierten Analysepaket namens *mlreflect* umgesetzt. Dieses ist sowohl online als auch auf dem Maxwell-Cluster des Deutschen Elektronen-Synchrotrons verfügbar. Somit stellt diese Arbeit einen bedeutenden Schritt in Richtung einer vollautomatischen Reflektivitätsdatenanalyse dar und kann sogar als Leitfaden für die Analyse anderer Streudaten dienen.

Acknowledgments

I would like to thank all of the many people, both within and outside our research group, who helped and supported me during the process of creating this work. Of course first and foremost, I want to express my sincerest gratitude to my Ph.D. supervisor Prof. Frank Schreiber for granting me the incredibly unique and fruitful opportunity to work on the cutting edge frontier of applying machine learning to scattering data analysis. On many occasions, Prof. Schreiber gave me the guidance and encouragement necessary to go beyond my limits and grow both professionally and personally. His foresight about the increasing relevance of this new research field enabled me to make several important and timely scientific contributions, which I will always be very grateful for.

In this context, I also want to thank Prof. Stefan Kowarik and his former master student Christos Karapanagiotis for the inspiring discussions that provided the basis for my research and which gave me an important head start in this new and competitive field. In addition, I would like to also thank my second Ph.D. supervisor Prof. Martin Oettel for offering me valuable advice and support throughout my work. I also want to thank the "Machine Learning for Science" Cluster of Excellence for their scientific feedback specifically on machine learning.

I am also incredibly grateful to Dr. Alexander Hinderhofer who always made room in his schedule for countless technical discussions ranging from broad ideas down to the nitty-gritty. Without his scientific advice, large parts of this work would not have been possible. I also want to thank Dr. Alexander Gerlach for his efforts in providing me with the necessary computational resources and for always providing a fresh scientific perspective on my work.

Furthermore, I owe my thanks and appreciation to Vladimir Starostin with whom I worked very closely for most of my work. Vladimir helped me wrap my head around many difficult concepts in mathematics and programming and I am certain that without him, my work would not have reached its current level of maturity. I

also want to thank Valentin Munteanu, who many times provided me with inspiring new ideas and pointers to important literature.

Moreover, I am grateful to Dr. Maximilian Skoda with whom I had many pleasant and insightful discussions regarding neutron reflectometry and neutron sources in general.

Also, I want to express my gratitude to Evelyn Edel who, through her detailed and thorough work, helped me a lot in understanding and using the fast Fourier transform algorithm. It was a pleasure working with her and I wish her all the best for the future.

Importantly, I also thank Nadine Rußegger for letting me use her thin film samples in my annealing experiments. This generous donation saved me a lot of time and effort. I also want to thank Ingrid Dax for providing me her valuable X-ray reflectivity data to test my analysis package.

My thanks also go to everybody who helped me on our many beamtimes at P08, DESY. In particular I want to thank Ekaterina Kneschaurek, Niels Scheffczyk, Dr. Ivan Zaluzhnyy and Dr. Linus Pithan who put in many hours and night shifts for my project. Also, I want to thank the DESY staff, especially Dr. Florian Bertram, Dr. Chen Shen, Dr. André Rothkirch and Dr. Frank Schlünzen, for their experimental support as well as their help in setting up the *mlreflect* package on the Maxwell cluster.

Of course, I also owe my gratitude to everybody who graciously took the time to proof-read this thesis, in particular Anita Girelli, Dr. Linus Pithan, Dominik Zietlow and Annika Timmins.

Lastly, I want to sincerely thank everyone who, despite my sometimes rather reserved nature, broke the ice with me and made me feel welcome and at home in our research group.

Contents

Abstract **v**

Deutsche Zusammenfassung **vii**

Acknowledgments **ix**

I. Introduction and Theory 1

1. Introduction **3**

1.1. Scientific background and motivation 3
1.2. Structure of this thesis 9

2. Experimental methods and materials **11**

2.1. Basic scattering theory 11
2.1.1. X-rays and neutrons as propagating waves 11
2.1.2. Definition of the atomic scattering length 12
2.1.3. Scattering from continuous media 14
2.2. Specular reflectivity measurements of thin films 15
2.2.1. Purpose and concept of reflectometry measurements 15
2.2.2. Fresnel reflectivity 17
2.2.3. The kinematical approximation 22
2.2.4. The Parratt and matrix algorithms for multi-layer reflectivity 25
2.2.5. Rough interfaces and the Névot-Croce factor 30
2.2.6. Data formats and vector notation of measured reflectivity . . . 33

2.3. Materials studied and sample preparation 34
2.3.1. Organic semiconductor thin films 34
2.3.2. Molecular beam deposition in ultra-high vacuum 34

3. Machine learning fundamentals **37**
3.1. Definition of the task from a machine learning perspective 37
3.2. Feedforward neural networks 39
3.2.1. Basic terms and linear regression 39
3.2.2. The multi-layer perceptron 42
3.3. Training neural networks 46
3.3.1. Stochastic gradient descent 46
3.3.2. Gradient calculation through back-propagation 47
3.4. Input and output standardization 50
3.5. Adaptive optimizers 53
3.5.1. SGD with momentum 53
3.5.2. RMSprop 54
3.5.3. ADAM 55
3.5.4. Learning rate schedules 56
3.6. Validation and generalization 56

II. Results and Discussion **61**

4. Analysis of reflectivity data using neural networks **63**
4.1. Introduction 63
4.2. Neural network architecture and training 64
4.3. Performance comparison to conventional LMS fitting 69
4.4. Conclusions from this chapter 73

5. Neural network performance in the light of challenging cases **75**
5.1. Introduction 75

5.2. Training data simulation 76
5.2.1. General simulation parameters 76
5.2.2. Training data modifications 78
5.3. Neural network design and training 82
5.4. Results and discussion 86
5.4.1. Definition of the prediction accuracy 86
5.4.2. Comparison of the prediction accuracy with the curve mean squared error 87
5.4.3. Influence of different SLD combinations on the prediction accuracy 92
5.4.4. Influence of noise and background on the prediction accuracy . 94
5.4.5. Other challenging cases and prospects 97
5.5. Conclusions from this chapter 97

6. The analysis pipeline of the *mlreflect* package 101
6.1. Introduction 101
6.2. Description of the analysis pipeline 102
6.2.1. Overview 102
6.2.2. Preprocessing (step I.) 102
6.2.3. Neural network predictions (step II.) 106
6.2.4. Postprocessing (step III.) 107
6.3. Performance test on thin films 108
6.4. Differences between simulated and experimental data 111
6.5. Influence of systematic measurement errors 114
6.6. Fourier transforms as a method for feature engineering 119
6.7. Implementation of *mlreflect* at P08/DESY 120
6.8. Conclusions from this chapter 121

7. Remarks on current limitations and future research 123
7.1. Fully-connected vs. convolutional architectures 123
7.2. Fixed input size and q dependence 124

7.3. Non-unique solutions 125
7.4. Including *a priori* knowledge at inference time 127
7.5. Experimental test and training sets 128
7.6. Training optimization 129

8. Conclusion and outlook 131
8.1. Summary and conclusion 131
8.2. Outlook 134

III. Appendix 137

A. Additional figures for Chapter 4 139

B. Additional figures for Chapter 5 145

C. Additional figures for Chapter 6 151

D. Availability of the *mlreflect* package 161
D.1. Download from PyPI 161
D.2. Web documentation on Read the Docs 161
D.3. Open source on GitHub 162

E. Example of ambiguous box models 163

F. Datasets used in this work 169

Bibliography 171

List of publications 191

List of acronyms 193

Part I.

Introduction and Theory

1. Introduction

1.1. Scientific background and motivation

Scattering with X-rays and neutrons offers a large variety of complementary and non-invasive measurement techniques for the investigation of different materials and physical systems [1–3]. The increasing brilliance of both X-ray and neutron sources and the development of better detector technologies have enabled new types of faster measurements [4]. These include, among others, the tracking of dynamics and *in situ* processes, such as crystal growth, exchange mechanisms, phase transitions and morphological changes [5–11]. Due to the high pixel resolution and fast read-out times of the required detectors, advanced experimental setups often produce a large amount of data. This data is usually complex and requires a significant amount of expertise and mathematical modeling to be analyzed, which is usually done manually by experts. Thus, extracting physical information from data is starting to become the main bottleneck for scientific productivity (Fig. 1.1). Gaining equity between the speed of data acquisition and data analysis could enable on-the-fly data analysis. Aside from enabling faster decision-making by researchers during experiments, this would allow the development of feedback systems for even more advanced experimental setups. It is clear that, as the tools for data acquisition become more powerful, methods of analyzing the data have to become equally as advanced to be able to fully exploit their benefits.

This abundance of information is not exclusive to science, since the trend of collecting and storing increasing amounts of data about ourselves and our environment can be observed in almost all parts of our lives [12]. The term "Big Data"

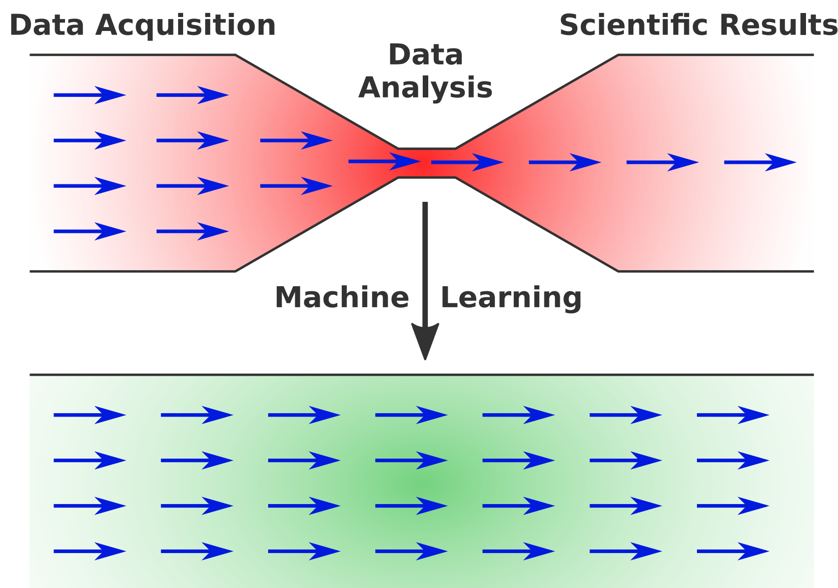

Figure 1.1.: Data analysis as a potential bottleneck for scientific productivity. ML-based analysis methods, such as the ones discussed in this work, may help prevent this problem.

describes the phenomenon of producing and storing more data than can feasibly be processed by conventional means. In the last two decades, there has been a resurgence of machine learning (ML) methods based on neural networks (NNs) which have shown promising results in handling large amounts of data in an automatized fashion. Prominent examples of this are the prediction of protein folding from their amino acid sequence with unprecedented accuracies of up to 90 % [13] as well as vast improvements in the field of natural language processing [14]. While the idea of artificial NNs already experienced two previous waves of popularity in the 1950s and 80s [15–17], major breakthroughs have only been achieved within the last 10–15 years [18–20]. The reasons for this lie mainly within the availability of large datasets for training [21–25] as well as the necessary computing hardware capable of training much more complex NN models than before ("deep learning") [26]. In particular, the increased usage of graphics processing units (GPUs) [27–29] in combination with the development of a specialized software infrastructure [30–36] have led to wide-spread gains in performance for ML methods.

In recent years, ML-based methods, often deep NNs, have also been employed to tackle various tasks in the material and physical sciences [37–42]. While one of the first applications of NNs in scattering physics was demonstrated for ellipsometry measurements already decades ago [43], it has only become more wide-spread within the last 5 years [44]. Many of the published ML solutions are targeted towards crystal structure and space group determination of X-ray diffraction data [45–53]. To train ML models, these methods make use of large amounts of published diffraction datasets that have been systematically collected in various databases over many decades. Another recent application in scattering physics is the use of ML to determine the shape of particles from small-angle scattering measurements in transmission [54–57] or in grazing-incidence geometry [58–60]. Furthermore, in the case of grazing-incidence wide-angle scattering, deep NNs have also been employed for the tracking of diffraction features in real-time measurements [61]. Other related fields, such as diffuse scattering [62], Raman scattering [63], electron scattering [64] and research with X-ray free electron lasers (XFELs) [65, 66] have also seen some development in this direction. In many cases, the ML model is designed to extract physical information *via* the same pathway as a manual analysis, *i.e.* by assuming a pre-established theoretical model, albeit in a faster and more automized way. Other approaches are designed to optimize the experiment itself or post-process the data prior to manual data analysis, such as denoising of the data [56].

In this light, the work presented in this thesis focuses on the analysis of specular X-ray reflectivity (XRR) and neutron reflectivity (NR) and ways to quickly extract information from those measurements using ML. Reflectometry is a well-established and common surface scattering method that allows the investigation of the scattering length density (SLD) profile (*i.e.* atom or electron density) perpendicular to the surface of a sample [67–70]. The shape of the SLD profile gives information about the thickness, roughness and composition of layered structures and thin films, which are of importance for many industrial processes and fundamental research. XRR and NR have been used extensively in the study of a large variety of systems, such as liquid and solid thin films [71–77], layers of polymers [78, 79], lipids [80, 81],

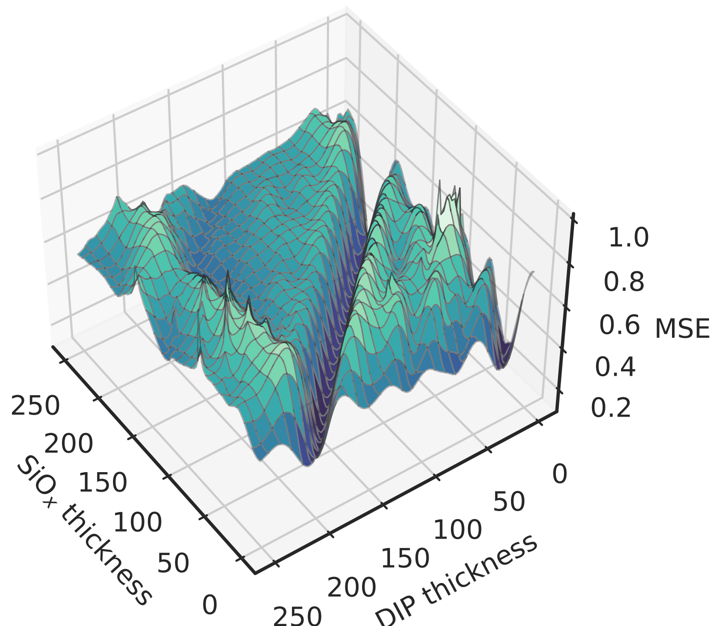

Figure 1.2.: A simplified example of the MSE surface of a reflectivity fit with two layers on a silicon substrate: a silicon oxide layer with a thickness of 10 Å and an organic thin film (DIP) with a thickness of 200 Å. The MSE surface shows many local minima across the entire parameter space. Because of the phase problem, there is also a deep "canyon" of similar solutions along the diagonal where the sum of both thicknesses is 210 Å. These multiple solutions can make conventional LMS fitting approaches slow. The problem is also further amplified if there are more fitting parameters, *e.g.* roughness and SLD, which is usually the case.

self-assembled mono-layers [82, 83] and organic semiconductors (OSCs) [84–86]. Polarized NR in particular can also be used to study magnetic materials [87]. Moreover, *in situ* measurements, such as real-time XRR at synchrotrons, have become increasingly important, since they enable the investigation of sample dynamics, *e.g.* during film growth or annealing [88–90]. Furthermore, advanced experimental setups for very fast data acquisition have been developed recently that achieve up to 10 XRR measurements per second [7, 8, 91].

While these methods give new physical insights, the rate at which the data is acquired has long surpassed the speed at which it can be analyzed. The typical way to analyze reflectometry data is *via* least mean squares (LMS) fitting algorithms. The working principle of these algorithms consists of iteratively simulating reflectiv-

ity curves following well-established theoretical models and minimizing the residuals between the simulation and the measured data. However, even for relatively simple systems, *i.e.* 1–2 layers, the mean squared error (MSE) surface can be non-trivial. Fig. 1.2 shows the MSE surface when fitting a two-layer system (organic thin film + silicon oxide on a silicon substrate) with a conventional algorithm. Despite there being only two open fitting parameters, the surface has many local minima and shows a deep "canyon" of similar solutions. This can make finding the global minimum with an LMS algorithm difficult, especially if there are more fitting parameters or experimental noise, which is typically the case. Furthermore, sometimes even multiple exact solutions exist due to the so-called "phase problem" [1, 92] and unless very good starting parameters are provided, a simple gradient descent algorithm will likely not converge to the correct minimum. The problem of complicated error surfaces is generally solved by using more sophisticated minimization algorithms which have been implemented in various software packages [93–101]. The most prominent among them are based on differential evolution [102], a stochastic minimization algorithm that has been shown to reliably find the global minimum even on very complicated error surfaces. However, due to the stochastic nature of this algorithm, the search of the solution space is usually quite time-consuming. As a possible solution, several recently published papers have demonstrated how ML methods, in particular NNs, can aid the analysis of reflectometry data. Most of the approaches demonstrate ways of directly extracting the sample parameters from the data [103–111], but NNs have also been applied to decrease the time needed for NR measurements by reducing noise in the data [112].

The research presented in this thesis mainly focuses on the application of fully-connected neural networks (FCNNs) for the fast and automated analysis of reflectometry data from one or two layer systems of OSC thin films on silicon substrates (see schematic in Fig. 1.3). The results were in large part already laid out in previous publications [103, 107, 111]. OSC thin films are subject to many studies due to their uses as solar cells and light-emitting diodes [113, 114]. However, the methods discussed herein are not inherently exclusive to these systems and can be adapted for

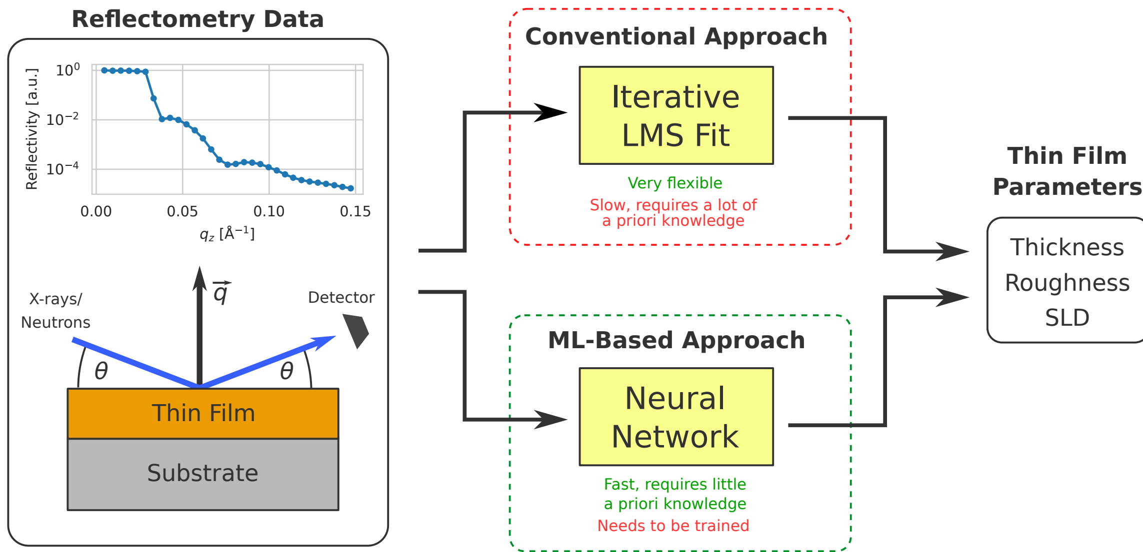

Figure 1.3.: Schematic comparison of conventional LMS fitting approaches and ML-based approaches for analyzing reflectometry data. Extracting sample parameters from the data with the conventional approach is very flexible, but generally slow and requires a large amount of *a priori* knowledge. In contrast, the ML-based approach is generally fast and requires less prior knowledge, but the NN must be properly trained before use.

other materials and substrates. The work presented here culminated in the development of the Python-based analysis package *mlreflect* which was developed as part of a project funded by the Bundesministerium für Bildung und Forschung (BMBF) and was made available both online and on the package repository of the Maxwell cluster at Deutsches Elektronen-Synchrotron (DESY). Thus, this research constitutes a large step towards fully-automated ML solutions for reflectivity data analysis.

1.2. Structure of this thesis

This thesis is divided into three parts: Part I: Introduction and Theory, Part II: Results and Discussion and Part III: Appendix.

Part I Chap. 2 discusses experimental techniques, in particular the basics of scattering theory as well as a mathematical description of specular reflectivity from a sample. The chapter also contains a short description of the used materials and the method of depositing organic molecules in ultra-high vacuum.

Chap. 3 introduces the concept of ML and discusses how ML techniques can be used to facilitate reflectivity data analysis. Furthermore, various important numerical methods are discussed, such as NNs and stochastic gradient descent.

Part II Chap. 4 discusses the first published approach of employing a NN for reflectivity data analysis [103]. Chap. 5 introduces improvements to this approach in terms of data preprocessing and discusses particularly challenging cases [107]. Lastly, Chap. 6 demonstrates a complete analysis pipeline that combines and refines the results from previous chapters. It also introduces a method of improving the NN performance by re-sampling the input data. Additionally, the chapter describes the open-source Python package *mlreflect* which implements the analysis pipeline [111]. Chap. 7 provides a discussion of the remaining limitations and possible improvements of the described methods. Chap. 8 summarizes all core results of this work and gives a brief outlook on future research opportunities.

Part III The Appendix contains additional figures for Chapters 4–6 that show supplementary results which are not critical to support the conclusions but might still be useful to the reader. Appendix D explains how to access and download the *mlreflect* package online. Appendix E provides additional information and calculations for the discussion about reflectivity measurements with ambiguous solutions. Appendix F provides a complete table of all of the reflectivity data sets used in this work, in-

cluding information about when and where the data was acquired. Also, a list of all acronyms used in this work can be found at the end of the Appendix.

The Appendix also includes the bibliography as well as a list of all publications by the author of this thesis.

2. Experimental methods and materials

2.1. Basic scattering theory

2.1.1. X-rays and neutrons as propagating waves

Both neutrons and X-rays are commonly used as probes in scattering experiments to investigate various material properties. X-rays are a form of electromagnetic radiation usually classified as waves with wavelength between $\lambda = 0.1$–$100\,\text{Å}$. In contrast, neutrons are uncharged subatomic particles that, together with protons, constitute the nuclei of atoms. According to the wave-particle dualism, a particle beam of neutrons can also be treated as a mass wave with a De Broglie wavelength $\lambda = h/(m_\mathrm{n}v) = h/\sqrt{2m_\mathrm{n}E}$ propagating through space. Here, h is Planck's constant and m_n, v and E are the mass, velocity and energy of a single neutron, respectively. In terms of scattering, X-rays mainly interact with electrons [1] while neutrons mainly interact with the nuclei of atoms *via* the strong interaction and the magnetic moments (both nuclear and electronic) [3]. Despite the fact that the elementary process of scattering from matter is different for neutrons and X-rays, a lot of the scattering formalism can be treated in a similar way. Thus, for the discussions in this thesis, we shall treat both probes as monochromatic waves that are elastically scattered. Furthermore, the discussion will be limited to non-magnetic and non-resonant scattering.

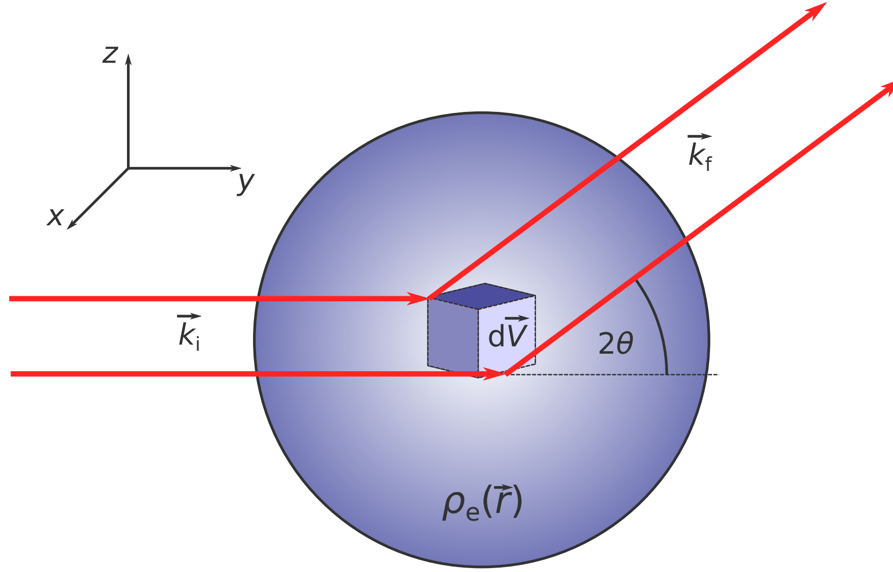

Figure 2.1.: An incoming wave that is scattered by the electron cloud of an atom. The distribution of electrons in space if given by $\rho_e(\vec{r})$. The total scattered wave is obtained by integrating over all possible phase differences $\exp(i\vec{q}\vec{r})$ weighted by $\rho_e(\vec{r})$.

2.1.2. Definition of the atomic scattering length

Let us first consider the scattering of X-rays and neutrons from single objects. For X-rays, the simplest case is the scattering from a single electron, which is described by Thompson scattering [115]. Here, an incident plane wave $\vec{E}_i$ is scattered as a spherical wave

$$\vec{E}_f = \vec{E}_i b_e \frac{e^{-ik_0 r}}{r}, \tag{2.1}$$

where k_0 is the absolute wave vector, b_e the scattering length of the electron, and r the distance from the electron. Here, b_e quantifies how strongly the wave is scattered and for a single electron is given by the classical electron radius[1] $b_e = r_e$.

Extending this concept to the scattering from individual atoms, to describe the total scattering amplitude we have to consider the entire electron cloud, as shown in Fig. 2.1.

[1] $r_e = e^2/(4\pi\epsilon_0 m_e c^2) \approx 2.818 \times 10^{-15}\,\mathrm{m} = 2.818 \times 10^{-5}\,\text{Å}$

We can obtain the atomic scattering length b by integrating over the scattering contribution b_e of all electrons weighted by their distribution $\rho_\mathrm{e}(\vec{r})$ and a phase factor depending on their relative distance

$$b(\vec{q}) = \int b_\mathrm{e}\rho_\mathrm{e}(\vec{r})e^{-i\vec{q}\vec{r}}dV(\vec{r}) \tag{2.2}$$

$$= r_\mathrm{e}\int \rho_\mathrm{e}(\vec{r})e^{-i\vec{q}\vec{r}}dV(\vec{r}) = r_\mathrm{e}f(\vec{q}), \tag{2.3}$$

where $\vec{q}$ is the moment transfer vector and $f(\vec{q})$ the atomic form factor [115]. In general, $\vec{q}$ is defined as the difference between the final and incoming wave vectors

$$\vec{q} = \vec{k}_\mathrm{f} - \vec{k}_\mathrm{i}, \tag{2.4}$$

If $q = |\vec{q}|$ is small, the integral over the electron density simply yields the total number of electrons Z. Hence, the scattering length of an atom with atom number Z is approximately given by $b = r_\mathrm{e}f(q \approx 0) \approx r_\mathrm{e}Z$.

For neutrons, the scattering length of a single atom is determined by the neutron-nucleus interaction, which can be approximated by the Fermi pseudo-potential $V_\mathrm{N}(r)$ [3, 115]. Using Fermi's Golden Rule, the interaction of a neutron wave ψ_i with the potential $V_\mathrm{N}(r)$ can be written as

$$b = \frac{m_\mathrm{n}}{2\pi\hbar^2}\langle\psi_\mathrm{f}|V_\mathrm{N}|\psi_\mathrm{i}\rangle = \frac{m_\mathrm{n}}{2\pi\hbar^2}\int V_\mathrm{N}e^{i\vec{q}\vec{r}}d\vec{r}. \tag{2.5}$$

Since the typical range of the nuclear potential V_N is $r < 1 \times 10^{-5}$ Å and for cold neutrons $q \approx 2\pi$ Å^{-1}, the phase factor becomes

$$e^{i\vec{q}\vec{r}} \approx 1, \tag{2.6}$$

which means that the atomic scattering length b is q-independent. The evaluation of Eq. 2.5 with the inclusion of the spin term yields

$$b^\pm = b_\mathrm{c} + b_\mathrm{s}^\pm \tag{2.7}$$

where b_c is the coherent (spin-independent) and $b_\mathrm{s}^{\pm}$ the spin-dependent part of the atomic scattering length. Whether the scattering length takes the value b^+ or b^- depends on the corresponding neutron spin state (spin up $+/\uparrow$ or spin down $-/\downarrow$) relative to the nucleus spin. For elastic scattering from non-magnetic materials, averaging over the spin-dependent scattering of all nuclei in a sample leads to an incoherent contribution that amounts to a constant background in the detected signal. How this background can influence reflectivity data analysis is discussed in more detail in Chap. 5. When scattering from magnetic materials, the spin-dependent part leads to a coherent contribution depending on the orientation of the nuclear spins as well as the polarization of the neutron beam. In this thesis, only elastic scattering from non-magnetic materials is discussed, therefore we shall only consider the coherent part of the scattering length in the following, *i.e.* $b = b_\mathrm{c}$.

In general, for both X-rays and neutrons, the scattering length is a complex expression

$$b = b' + ib''. \tag{2.8}$$

The imaginary part of b accounts for a decay in the scattering amplitude due to absorption.

Using the same expression of the scattering length b for both neutrons and X-rays allows us to write a lot of the following scattering formalism in a common form.

2.1.3. Scattering from continuous media

The scattering from continuous media can in principle be treated like the scattering from a distribution of scatterers similar to the atomic form factor in the previous section by introducing the quantity of SLD $\rho(\vec{r})$. For a medium with only one atom type, the SLD can be written as

$$\rho(\vec{r}) = b\rho_\mathrm{a}(\vec{r}), \tag{2.9}$$

where $\rho_{\mathrm{a}}(\vec{r})$ is the atom density inside the medium and b the complex atomic scattering length. Similarly, the SLD for X-rays can be expressed in terms of the electron density of the medium

$$\rho(\vec{r}) = r_e \rho_{\mathrm{e}}(\vec{r}). \tag{2.10}$$

Since the SLD is the product of a scattering length and a density, it has units of m^{-2}.

If we ignore multiple scattering within the medium, we can write the scattering amplitude $A(\vec{q})$ as a simple Fourier transform (FT) of the SLD similar to Eq. 2.2 [3]

$$A(\vec{q}) = \int \rho(\vec{r}) e^{-i\vec{q}\vec{r}} d\vec{r}. \tag{2.11}$$

Furthermore, it is possible to define a refractive index n for both X-rays and neutrons to characterize the scattering between different media [67, 115, 116]. If we define n in terms of the SLD, we can write

$$n = 1 - \frac{\lambda^2}{2\pi}\rho = 1 - \frac{2\pi}{k^2}\rho = 1 - \delta + i\beta, \tag{2.12}$$

where ρ is the average SLD of the medium. This definition of the refractive index will be used throughout this thesis, in particular during the discussion of specular reflectivity in the next section.

2.2. Specular reflectivity measurements of thin films

2.2.1. Purpose and concept of reflectometry measurements

Reflectometry measurements, both with neutrons and X-rays, are commonly used to investigate the structure and morphology of a large variety of thin films, surfaces and layered structures. The main idea is that through the careful choice of the scattering vector, *i.e.* the specular scattering geometry, the SLD profile $\rho(z)$ of the sample along the surface normal can be investigated. Fig. 2.2 shows a common experimental setup to measure specular reflectivity. The sample is placed on a diffractometer and

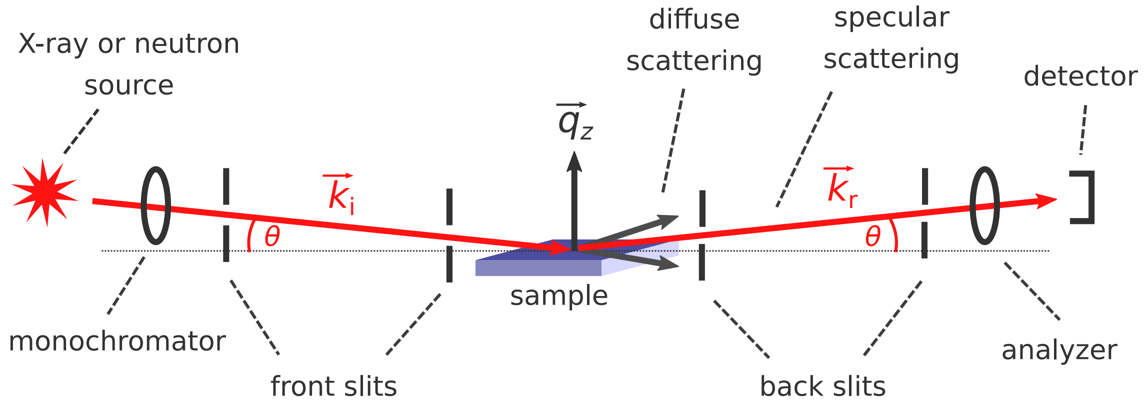

Figure 2.2.: Typical experimental setup for specular reflectivity measurements. An incident beam (neutrons or X-rays) impinges on the sample at an incident angle θ and the specular reflection (at the same outgoing angle θ). Both the incoming and outgoing beam are collimated *via* a pair of slits and passed through a monochromator and analyzer, respectively, to ensure the specular condition. This results in a moment transfer vector $\vec{q}_z$ perpendicular to the sample surface for the measured signal. Each measurement is typically performed over a range of q_z values.

monochromatic, collimated radiation impinges on the sample at an incident angle θ. The reflected beam is then measured with a detector at the same outgoing angle θ. The specular condition ensures that $\vec{q}_z$ is always perpendicular to the sample and each measurement is conducted over a range of angles and q_z values.

The incoming beam is reflected by both individual atoms and molecules, but also at the interfaces of different materials. This means in addition to Bragg reflections, one can also observe features that correspond to structures on the nanometer or even micrometer scale. Fig. 2.3 shows a typical example of an XRR measurement of an organic thin film on a silicon substrate. The most important features of the data with regards to this thesis are the Kiessig fringes, which relate to the film thickness and roughness, as well as the position of the total reflection edge (TRE), which relates to the SLD. This ultimately allows the investigation of the complete SLD profile along the surface normal, including film thickness and interface roughness.

The analysis of this type of data, although long established, is not trivial and requires careful mathematical modeling. In the following chapters these models will be derived.

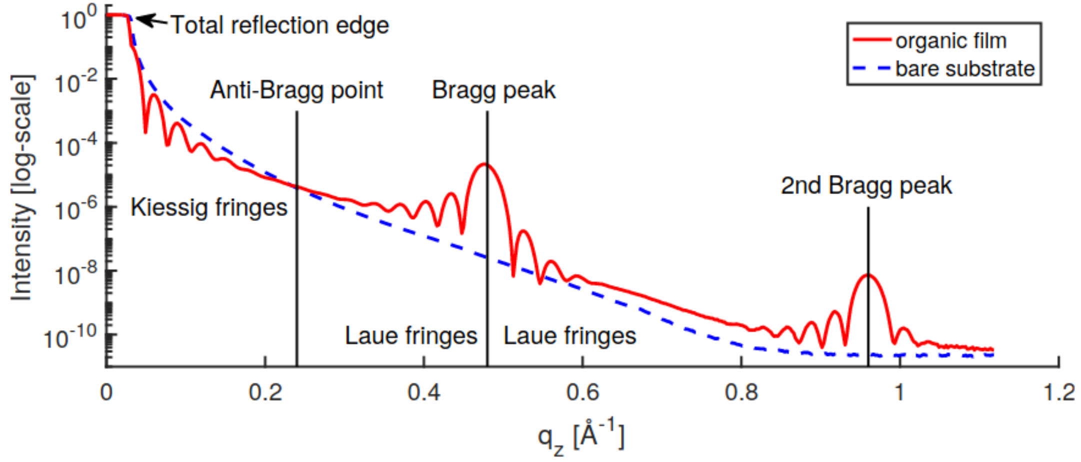

Figure 2.3.: Example of a typical reflectivity measurement of an organic thin film on a silicon substrate. The most important features with respect to this thesis are the Kiessig fringes and the total reflection edge (TRE). The periodicity and decay of the Kiessig fringes yield information about the thickness and roughness of the film, respectively. The position of the TRE is given by the critical angle and thus related to the SLD. Figure adopted from [117].

2.2.2. Fresnel reflectivity

Before considering more complicated mathematical models for describing the reflectivity from complex samples, it is useful to first derive the reflectivity from a flat surface of a semi-infinite, homogeneous slab as a function of the scattering vector q_z. For brevity, the mathematical treatment will assume X-rays as a probe, however, due to the similarities discussed in Sec. 2.1, an analogous treatment for neutrons can be found. Wherever possible, quantities will be expressed in such a way as to apply to both neutrons and X-rays.

For X-rays, if we assume that the plane wave $\vec{E}_\mathrm{i}$ impinges on a flat surface which forms the interface between medium 0 and medium 1 as depicted in Fig. 2.4, then the solutions of Helmholtz's equation for the propagation of of the electric field in a medium

$$\Delta\vec{E} + k_j^2\vec{E} = 0 \tag{2.13}$$

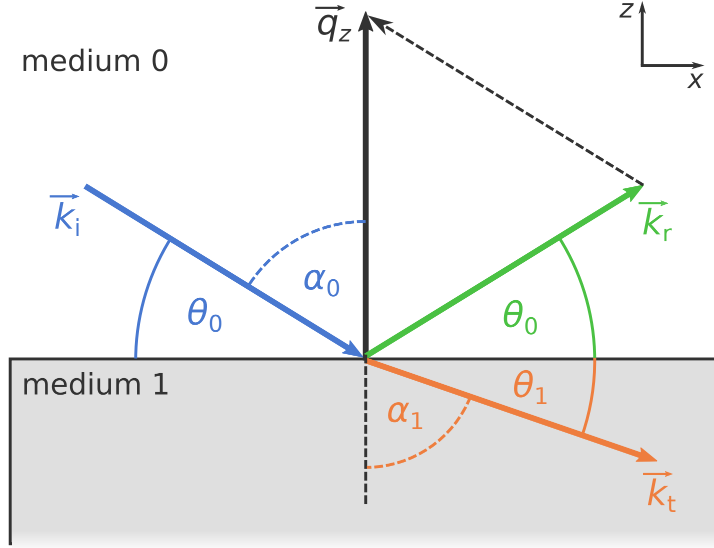

Figure 2.4.: A wave $\vec{E}_\mathrm{i}$ with wave vector $\vec{k}_\mathrm{i}$ in medium 0 (vacuum) impinges on the interface to medium 1 with the refractive index n. One part of the wave is reflected back into medium 0 with the wave vector $\vec{k}_\mathrm{r}$ while the other part is transmitted and refracted into medium 1 with the wave vector $\vec{k}_\mathrm{t}$.

for the incident, the reflected and the transmitted waves are

$$\vec{E}_\mathrm{i} = A_\mathrm{i} e^{i(\omega t - \vec{k}_\mathrm{i}\vec{r})}\hat{e}_y, \tag{2.14}$$

$$\vec{E}_\mathrm{r} = A_\mathrm{r} e^{i(\omega t - \vec{k}_\mathrm{r}\vec{r})}\hat{e}_y, \tag{2.15}$$

$$\vec{E}_\mathrm{t} = A_\mathrm{t} e^{i(\omega t - \vec{k}_\mathrm{t}\vec{r})}\hat{e}_y, \tag{2.16}$$

respectively. Here, k_j is the wave vector of the electromagnetic wave in medium j. Using the trigonometric identities, we can determine that the wave vectors in the different media are hence given by

$$\vec{k}_\mathrm{i} = k_0(\sin(\alpha_0)\hat{e}_x - \cos(\alpha_0)\hat{e}_z), \tag{2.17}$$

$$\vec{k}_\mathrm{r} = k_0(\sin(\alpha_0)\hat{e}_x + \cos(\alpha_0)\hat{e}_z), \tag{2.18}$$

$$\vec{k}_\mathrm{t} = k_0 n(\sin(\alpha_1)\hat{e}_x - \cos(\alpha_1)\hat{e}_z), \tag{2.19}$$

where

$$k_0 = |\vec{k}_\mathrm{i}| = |\vec{k}_\mathrm{r}| = \frac{|\vec{k}_\mathrm{t}|}{n} = \frac{2\pi}{\lambda} \tag{2.20}$$

and n is the refractive index of medium 1.

Since the tangential component of the wave must be continuous at the interface, the sum of the waves above the interface (incident and reflected) must be equal to the wave below the interface (transmitted), *i.e.*

$$A_\mathrm{i} e^{i(\omega t - k_0 \sin(\alpha_0) x)} + A_\mathrm{r} e^{i(\omega t - k_0 \sin(\alpha_0) x)} = A_\mathrm{t} e^{i(\omega t - k_0 n \sin(\alpha_1) x)}. \tag{2.21}$$

Using Snell-Descartes' second law

$$\sin(\alpha_0) = n \sin(\alpha_1), \tag{2.22}$$

Eq. 2.21 simplifies to

$$A_\mathrm{i} + A_\mathrm{r} = A_\mathrm{t}. \tag{2.23}$$

Using the Maxwell-Faraday equation, it can be shown that the tangential component of the magnetic component of the wave is given by

$$B = -\frac{1}{i\omega}\frac{\partial E_y}{\partial z}. \tag{2.24}$$

If we assume that the media are non-magnetic, *i.e.* B is also conserved, we can rewrite Eq. 2.21 and Eq. 2.23 as

$$(A_\mathrm{i} - A_\mathrm{r}) \cos(\alpha_0) = A_\mathrm{t} n \cos(\alpha_1). \tag{2.25}$$

In a similar fashion, the continuity of the parallel component can also be shown for neutrons [70]. By introducing $r = A_\mathrm{r}/A_\mathrm{i}$ and $t = A_\mathrm{t}/A_\mathrm{i}$, we can now write the well-known Fresnel equations [118]

$$r = \frac{\cos(\alpha_0) - n\cos(\alpha_1)}{\cos(\alpha_0) + n\cos(\alpha_1)}, \tag{2.26}$$

$$t = \frac{2\cos(\alpha_0)}{\cos(\alpha_0) + n\cos(\alpha_1)}. \tag{2.27}$$

We can also replace the angles α_j by the grazing incidence angle θ_j, *i.e.* the angle with respect to the surface plane, to write

$$r = \frac{\sin(\theta_0) - n\sin(\theta_1)}{\sin(\theta_0) + n\sin(\theta_1)} \approx \frac{\theta_0 - \theta_1}{\theta_0 + \theta_1}, \tag{2.28}$$

$$t = \frac{2\sin(\theta_0)}{\sin(\theta_0) + n\sin(\theta_1)} \approx \frac{2\theta_0}{\theta_0 + \theta_1}. \tag{2.29}$$

Here we also used that $n \approx 1$ for neutrons and X-rays. For the further discussions in this thesis, it will be useful to express r and t in terms of the momentum transfer vector

$$q_z = 2k_0 \sin\left(\frac{2\theta}{2}\right) = 2k_z. \tag{2.30}$$

With this we can write

$$r = \frac{q_{z,0} - nq_{z,1}}{q_{z,0} + nq_{z,1}} \approx \frac{q_{z,0} - q_{z,1}}{q_{z,0} + q_{z,1}}, \tag{2.31}$$

$$t = \frac{2q_{z,0}}{q_{z,0} + nq_{z,1}} \approx \frac{2q_{z,0}}{q_{z,0} + q_{z,1}}. \tag{2.32}$$

In this context, it is useful to consider how the critical angle θ_c is related to the SLD of a given medium. Using again a variant of Snell's law

$$\cos(\theta_0) = n\cos(\theta_1), \tag{2.33}$$

we can set $\theta_0 = \theta_\mathrm{c}$, $\theta_1 = 0$ and use the Taylor expansion of $\cos(\theta_\mathrm{c})$ to obtain

$$\theta_\mathrm{c} \approx \sqrt{2(1-n)}. \tag{2.34}$$

When using the definition of the refractive index from Eq. 2.12, the real part $(1-\delta)$ describes the refraction of the wave vector from medium 0 to medium 1 and the imaginary part $i\beta$ accounts for the attenuation of the amplitude due to absorption, *i.e.*

$$e^{ink_0z} = e^{i(1-\delta)k_0z}e^{-\beta k_0 z}. \tag{2.35}$$

The critical angle can then be found by combining Eq. 2.34 and Eq. 2.12 into

$$\theta_\mathrm{c} = \sqrt{2(\delta - i\beta)} = \sqrt{\frac{4\pi\rho}{k_0^2}}, \tag{2.36}$$

where ρ is the complex SLD of medium 1. Likewise, the critical moment transfer vector is

$$q_\mathrm{c} = 2k_0\sin(\theta_\mathrm{c}) \approx \sqrt{8k_0^2(\delta - i\beta)} = \sqrt{16\pi\rho}. \tag{2.37}$$

Using Eq. 2.34 and $\cos^2(x) + \sin^2(x) = 1$, we can approximate Snell's law for small angles and express it in terms of the critical angle so that

$$\begin{aligned} n\sin(\theta_1) &= n\sqrt{1-\cos^2(\theta_1)} \\ &= \sqrt{n^2 - (n\cos(\theta_1))^2} \\ &= \sqrt{n^2 - \cos^2(\theta_0)} \end{aligned} \tag{2.38}$$

$$n\theta_1 = \sqrt{\theta_0^2 - \theta_\mathrm{c}^2}. \tag{2.39}$$

Approximating $q/2k_0 = \sin(\theta) \approx \theta$, we can express Eq. 2.39 in terms of the wave vector transfer

$$nq_1 = \sqrt{q_0^2 - q_\mathrm{c}^2} \approx q_0 \tag{2.40}$$

and we can write the Fresnel equation as

$$r_\mathrm{F} \approx \frac{\theta_0 - \sqrt{\theta_0^2 - \theta_\mathrm{c}^2}}{\theta_0 + \sqrt{\theta_0^2 - \theta_\mathrm{c}^2}} = \frac{q_{z,0} - \sqrt{q_{z,0}^2 - q_\mathrm{c}^2}}{q_{z,0} + \sqrt{q_{z,0}^2 - q_\mathrm{c}^2}}. \tag{2.41}$$

The experimentally measured quantity, however, is the intensity, which we obtain by taking the modulus squared of r_F:

$$R_\mathrm{F}(q_z) = r_\mathrm{F} r_\mathrm{F}^* = |r_\mathrm{F}|^2 = \left| \frac{q_{z,0} - \sqrt{q_{z,0}^2 - q_\mathrm{c}^2}}{q_{z,0} + \sqrt{q_{z,0}^2 - q_\mathrm{c}^2}} \right|^2 \tag{2.42}$$

where R_F is commonly called the "Fresnel reflectivity". Since $n \approx 1$, we can assume

$$q_{z,1} \approx q_{z,0} = q_z, \tag{2.43}$$

that is, we ignore refraction effects and assume q_z does not change significantly between media. The Fresnel reflectivity can then be written as

$$R_\mathrm{F} \approx \frac{q_c^4}{16 q_z^4} = \frac{16\pi^2 \rho_\mathrm{s}^2}{q_z^4} \tag{2.44}$$

where ρ_s is the average SLD of the slab.

2.2.3. The kinematical approximation

After having introduced the concept of the Fresnel reflectivity, we shall now discuss ways to describe the reflected intensity from more complicated sample structures, such as multi-layer samples. While a fully analytical formula for calculating the reflected amplitude r is in general desirable and more precise, it is still useful to employ approximations to identify general trends and mathematical relationships between the measurement and the studied sample structure.

A commonly applied approximation is the so called kinematical approximation, where multiple scattering is generally not considered. This allows for the important simplification that the reflected amplitude r of the sample can be described as the

sum of the individual scattering contributions $r_{j,j+1}$ of each interface within the sample modified by the corresponding phase factor:

$$r = r_{0,1} + \sum_{j=1}^{N} r_{j,j+1} \exp\left[-i \sum_{k=1}^{j} q_{z,k} h_k\right], \tag{2.45}$$

where h_k is the thickness of the kth layer in the sample. Fig. 2.5 shows a schematic example of how an incoming wave is scattered multiple times at different interfaces. If we assume, as before, that q_z does not change significantly between media, *i.e.* Eq. 2.43 holds, we can write

$$r = r_{0,1} + \sum_{j=1}^{N} r_{j,j+1} \exp\left[-iq_z \sum_{k=1}^{j} h_k\right]. \tag{2.46}$$

Thus, combining Eq. 2.37, Eq. 2.40 and Eq. 2.43 yields

$$r_{j,j+1} = \frac{q_{z,j} - q_{z,j+1}}{q_{z,j} + q_{z,j+1}} = \frac{q_{z,j}^2 - q_{z,j+1}^2}{(q_{z,j} + q_{z,j+1})^2} = \frac{q_{c,j}^2 + q_{c,j+1}^2}{4q_z^2} = \frac{4\pi(\rho_{j,j+1} - \rho_j)}{q_z^2}. \tag{2.47}$$

If we substitute the Fresnel coefficient R_F from Eq. 2.44 and choose the origin of the z axis such that the interface between medium 0 and medium 1 is situated at $Z_1 = 0$ and the interfaces $j, j+1$ are situated at Z_{j+1}, then we can write the reflected intensity as

$$R(q_z) = rr^* = \left|\sum_{j=0}^{N} r_{j,j+1} e^{-iq_z Z_{j+1}}\right|^2 = \frac{R_\mathrm{F}(q_z)}{\rho_\mathrm{s}^2} \left|\sum_{j=0}^{N} (\rho_{j+1} - \rho_j) e^{-iq_z Z_{j+1}}\right|^2. \tag{2.48}$$

In the case of graded interfaces, these can be simply treated as an infinite series of slabs of infinitesimal thickness. Consequently, the sum converges to an integral over the sample:

$$R(q_z) = R_\mathrm{F}(q_z) \left|\frac{1}{\rho_s} \int_{-\infty}^{\infty} \frac{d\rho(z)}{dz} e^{iq_z z} dz\right|^2 \tag{2.49}$$

This form of the kinematical approximation is sometimes called the “Master formula” [1] and describes $R(q_z)$ as the Fresnel reflectivity $R_\mathrm{F} \propto q_z^{-4}$ multiplied with the

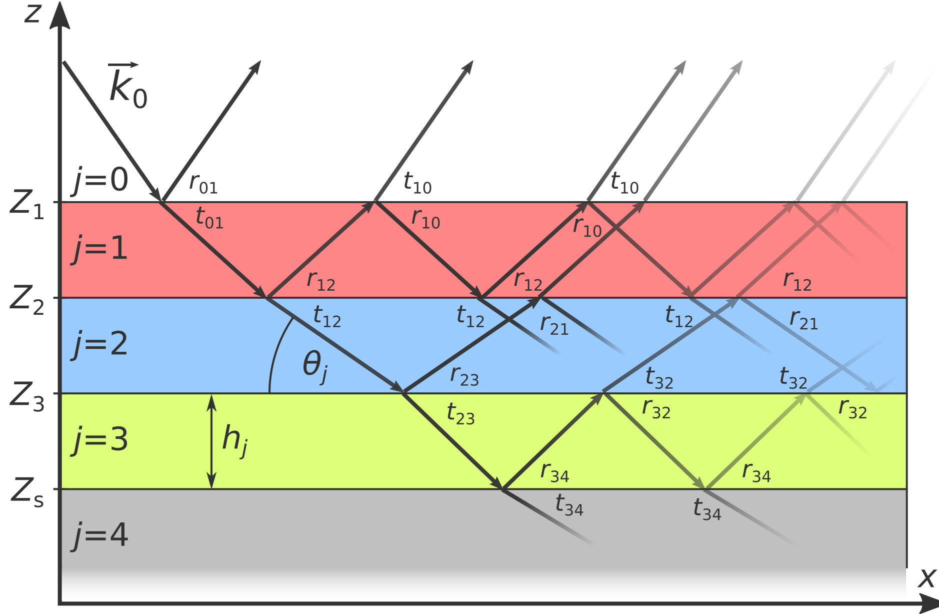

Figure 2.5.: Example of reflectivity from a multi-layer system with 5 different media with refractive indices n_j $(j = 1, 2, 3, \ldots, s)$, where $j = 0$ is the ambient medium and $j = s = 4$ the substrate. The incident beam is both reflected and refracted at the interfaces at positions Z_j depending on the reflection and transmission coefficients r and t, respectively. The beam can reflect or refract multiple times at different interfaces before exiting the sample at the top interface Z_1. The total reflected amplitude $r(q_z)$ is a summation of all different pathways through the sample with their respective phases.

modulus squared FT of the derivative of the SLD profile in z direction. However, because of the properties of th e FT, $R(q_z)$ can also be written as the FT of the SLD profile itself, *i.e.*

$$R(q_z) = R_{\mathrm{F}} \left| \frac{q_z}{\rho_s} \int_{-\infty}^{\infty} \rho(z) e^{iq_z z} dz \right|^2 = \left| \frac{4\pi}{q_z} \right|^2 \left| \int_{-\infty}^{\infty} \rho(z) e^{iq_z z} dz \right|^2 . \qquad (2.50)$$

These expressions show that the reflected intensity contains information about the sample in form of the absolute amplitudes of the Fourier components, however, the phase information is lost. This consequently means that the SLD profile $\rho(z)$ cannot simply be reconstructed from the measured intensity $R(q_z)$ *via* a simple reverse FT. This creates the problem of potentially ambiguous solutions for a given reflectivity

measurement. An example of this was illustrated by Sivia *et al.* [92], a mathematical treatment of which is shown in Appendix E.

In the special case of $\rho(z)$ being a box function of height ρ_0 and width d, this can be demonstrated more easily. In this case, the FT of the SLD profile is given by

$$\int_{-\infty}^{\infty} \rho(z) e^{iq_z z} dz = d\rho_0 \mathrm{sinc}\left(\frac{qd}{2}\right) \tag{2.51}$$

using $\mathrm{sinc}(x) = \sin(x)/x$. It is apparent that in this case, the FT is dominated by the product of the height and width of the box.

These commonly known problems have an impact on how information is extracted from measurements and thus, are also important for ML-based solutions, such as those discussed in Chap. 4–6. Importantly, the various approximations that were applied in the kinematical description lead to certain deviations from the data obtained from actual experiments. One of the largest deviations happens for smaller values of q_z due to multiple reflections being neglected. For values of $q_z \lesssim 3q_c$, the likelihood of a wave being reflected at an interface becomes high enough for multiple reflections to be significant. Also, total reflection, *i.e.* $R(q_z < q_c) = 1$ (ignoring absorption), is a behavior not described by the kinematical approximation.

2.2.4. The Parratt and matrix algorithms for multi-layer reflectivity

The ML techniques shown in this thesis all use supervised learning methods that mainly make use of simulated data. Since the training data should be as close to experimentally acquired data as possible, a more precise mathematical description for the reflected intensity from a sample is necessary. This chapter will introduce a theoretical calculation of multi-layer reflectivity based on matrix operations. The general concept of this method originally stems from the field of optics and was developed by Abelès [119]. The application to reflectometry was later implemented by Heavens [120]. Fig. 2.6 shows a comparison between the kinematical approximation derived in the previous chapter and the matrix method.

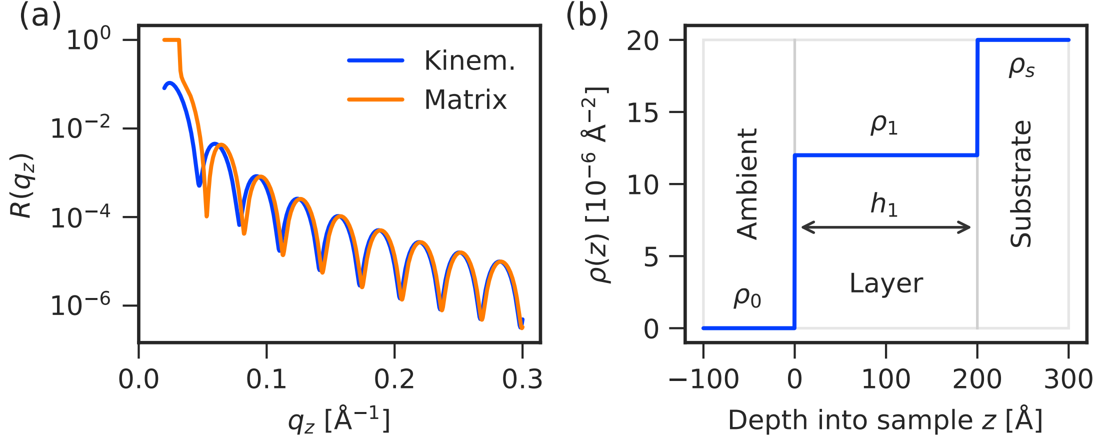

Figure 2.6.: (a) Comparison of specular reflectivity simulated with both the kinematical approximation and the matrix method (dynamical theory). The kinematical approximation significantly deviates from the exact solution for $q_z < 0.1\,\text{Å}^{-1}$ and fails to capture the TRE. (b) The underlying SLD profile of the one-layer system used for the simulation.

Another method for calculating multi-layer reflectivity is the more well-known Parratt formalism [71], which was developed around the same time. Parratt's method, however, is a recursive algorithm, which is arguably easier to understand intuitively, but generally much slower to compute on conventional hardware. This is especially relevant in the age of deep learning which heavily utilizes GPUs, since they are specialized in performing matrix operations. Due to these advantages, the matrix method is implemented in most analysis packages that rely on the simulation of reflectivity data, such as the *mlreflect* package introduced in Chap. 6.

Here, the description of the matrix method will largely follow the treatment from [121], which originally assumes X-rays as a probe. For non-magnetic samples, however, the treatment works similarly for neutrons. Let us consider an incoming wave $\vec{E}^-$ with the wave vector $\vec{k}_j$ that is traveling downwards in the direction of the sample surface, *i.e.*

$$\vec{E}^- = A^- e^{i(\omega t - k_{x,j}x - k_{z,j}z)}\hat{e}_y, \tag{2.52}$$

where

$$k_{x,j} = k_j \cos(\theta_j) \quad \text{and} \tag{2.53}$$

$$k_{z,j} = -k_j \sin(\theta_j) = -\sqrt{k_j^2 - k_{x,j}^2}. \tag{2.54}$$

As in previous sections, θ is the incident angle relative to the surface it impinges on and j denotes the index of the medium the wave is traveling in. At each interface, the upwards and downwards traveling waves are superimposed to

$$E_j(x,z) = \left(A_j^+ e^{ik_{z,j}z} + A_j^- e^{-ik_{z,j}z}\right) e^{i(\omega t - k_{x,j}x)} \tag{2.55}$$

and the magnitude of the upwards and downwards traveling waves in each stratum can thus be written as

$$U(\pm k_{z,j}, z) = A_j^{\pm} e^{\pm ik_{z,j}z}. \tag{2.56}$$

As described by Eq. 2.21 and Eq. 2.24 in Subsec. 2.2.2, the tangential components of the waves as well as their derivatives must be conserved at each interface. Hence, the sum of the waves above and below each surface must be equal such that

$$U(k_{z,j}, Z_{j+1}) + U(-k_{z,j}, Z_{j+1}) = U(k_{z,j+1}, Z_{j+1}) + U(-k_{z,j+1}, Z_{j+1}), \tag{2.57}$$

$$k_{z,j}\left(U(k_{z,j}, Z_{j+1}) - U(-k_{z,j}, Z_{j+1})\right) = k_{z,j+1}\left(U(k_{z,j+1}, Z_{j+1}) - U(-k_{z,j+1}, Z_{j+1})\right) \tag{2.58}$$

The above set of equations can be conveniently expressed in the more compact matrix form

$$\begin{bmatrix} U(k_{z,j}, Z_{j+1}) \\ U(-k_{z,j}, Z_{j+1}) \end{bmatrix} = \underbrace{\begin{bmatrix} p_{j,j+1} & m_{j,j+1} \\ m_{j,j+1} & p_{j,j+1} \end{bmatrix}}_{\mathcal{R}_{j,j+1}} \begin{bmatrix} U(k_{z,j+1}, Z_{j+1}) \\ U(-k_{z,j+1}, Z_{j+1}) \end{bmatrix}, \tag{2.59}$$

where the matrix that transforms the magnitudes of the wave from medium j to medium $j+1$ is called the refraction matrix $\mathcal{R}_{j,j+1}$ with the matrix elements

$$p_{j,j+1} = \frac{k_{z,j} + k_{z,j+1}}{2k_{z,j}}, \tag{2.60}$$

$$m_{j,j+1} = \frac{k_{z,j} - k_{z,j+1}}{2k_{z,j}}. \tag{2.61}$$

The advantage of this formulation is that the transformation of the waves between media can be performed in a single matrix multiplication, which is a computationally very cheap operation.

Similarly to the refraction matrix, a translation matrix $\mathcal{T}_j$ can be found, which relates the magnitude of a wave in medium j at height z with the magnitude at height $z + h$:

$$\begin{bmatrix} U(k_{z,j}, z) \\ U(-k_{z,j}, z) \end{bmatrix} = \underbrace{\begin{bmatrix} e^{-ik_{z,j}h_j} & 0 \\ 0 & e^{ik_{z,j}h_j} \end{bmatrix}}_{\mathcal{T}_j} \begin{bmatrix} U(k_{z,j+1}, z + h_j) \\ U(-k_{z,j+1}, z + h_j) \end{bmatrix} \tag{2.62}$$

The total transformation of the wave between all layers of the sample can simply be obtained by multiplying the $\mathcal{R}_{j,j+1}$ and $\mathcal{T}_j$ for each layer. Assuming a sample with N layers, this leads to

$$\begin{bmatrix} U(k_{z,0}, Z_1) \\ U(-k_{z,0}, Z_1) \end{bmatrix} = \underbrace{\mathcal{R}_{0,1} \prod_{j=1}^{N} (\mathcal{T}_j \mathcal{R}_{j,j+1})}_{\mathcal{M}} \begin{bmatrix} U(k_{z,s}, Z_s) \\ U(-k_{z,s}, Z_s) \end{bmatrix}, \tag{2.63}$$

where $j = 0$ is the index of the ambient medium and $j = s = N + 1$ is the index of the substrate layer. Since the multiplication of square matrices results in another square matrix, the whole calculation can be performed as a sequence of fast matrix

multiplications. The product of these matrices is called the transfer matrix $\mathcal{M}$. We can thus write

$$\begin{bmatrix} U(k_{z,0}, Z_1) \\ U(-k_{z,0}, Z_1) \end{bmatrix} = \mathcal{M} \begin{bmatrix} U(k_{z,s}, Z_s) \\ U(-k_{z,s}, Z_s) \end{bmatrix} = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \begin{bmatrix} U(k_{z,s}, Z_s) \\ U(-k_{z,s}, Z_s) \end{bmatrix}. \quad (2.64)$$

Since the reflection coefficient is given by the ratio of the reflected wave to the incident wave, we can write it as the ratio of the downwards and upwards traveling waves at the top most interface Z_1.

$$r = \frac{U(k_{z,0}, Z_1)}{U(-k_{z,0}, Z_1)} = \frac{M_{11}U(k_{z,s}, Z_s) + M_{12}U(-k_{z,s}, Z_s)}{M_{21}U(k_{z,s}, Z_s) + M_{22}U(-k_{z,s}, Z_s)} \quad (2.65)$$

If we now assume that the substrate is sufficiently thick such that it will contain no upwards traveling wave, *i.e.*

$$U(k_{z,s}, Z_s) = 0, \quad (2.66)$$

the reflection coefficient will reduce to

$$r = \frac{M_{12}}{M_{22}}. \quad (2.67)$$

For a simple system containing just one flat interface, we obtain

$$r = r_{0,1} = \frac{M_{12}}{M_{22}} = \frac{m_{0,1}}{p_{0,1}} = \frac{k_{z,0} - k_{z,1}}{k_{z,0} + k_{z,1}} = \frac{q_{z,0} - q_{z,1}}{q_{z,0} + q_{z,1}}, \quad (2.68)$$

which is exactly the same as the Fresnel reflectivity from Eq. 2.31 derived in Subsec. 2.2.2.

Another import special case are two layers on a substrate, *e.g.* an organic thin film and a silicon oxide layer on top of a silicon substrate, which constitutes the majority of systems discussed in this work. In this case the reflection coefficient is

$$r = \frac{M_{12}}{M_{22}} = \frac{r_{0,1} + r_{1,2}e^{-2ik_{z,1}h_1} + r_{2,s}e^{-2i(k_{z,1}h_1 + k_{z,2}h_2)} + r_{0,1}r_{1,2}r_{2,s}e^{-2ik_{z,2}h_2}}{1 + r_{0,1}r_{1,2}e^{-2ik_{z,1}h_1} + r_{1,2}r_{2,s}e^{-2ik_{z,2}h_2} + r_{0,1}r_{2,s}e^{-2i(k_{z,1}h_1 + k_{z,2}h_2)}} \quad (2.69)$$

where

$$r_{j,j+1} = \frac{m_{j,j+1}}{p_{j,j+1}} \tag{2.70}$$

was used for brevity. Using $q_z = 2k_z$, we can write

$$r(q_z) = \frac{r_{0,1} + r_{1,2}e^{-iq_{z,1}h_1} + r_{2,s}e^{-i(q_{z,1}h_1+q_{z,2}h_2)} + r_{0,1}r_{1,2}r_{2,s}e^{-iq_{z,2}h_2}}{1 + r_{0,1}r_{1,2}e^{-iq_{z,1}h_1} + r_{1,2}r_{2,s}e^{-iq_{z,2}h_2} + r_{0,1}r_{2,s}e^{-i(q_{z,1}h_1+q_{z,2}h_2)}} \tag{2.71}$$

2.2.5. Rough interfaces and the Névot-Croce factor

In reality, most surfaces and interfaces are not perfectly flat, but instead have a lateral structure. Often, different types of surface and interface morphologies are combined under the term roughness. In practical terms, roughness describes how much the height profile $z(x, y)$ of an interface deviates from a perfectly flat plane. Of course, many types of morphologies for interfaces exist which affect the interference of reflected waves differently. A common distinction is whether the height deviations are correlated or uncorrelated on the relevant length scale of our probe. Here, we shall only consider roughness in the form of uncorrelated height variations with a random distribution, since they are the most prominent in the types of samples discussed in Part II of this thesis. Further, we shall assume that the height profile $z(x, y)$ of a given interface follows the Gaussian distribution

$$p(z) = \frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{z^2}{2\sigma^2}} \tag{2.72}$$

where the standard deviation σ of the distribution denotes the roughness of the interface.

It has been shown by Névot & Croce [122] that this type of interface leads to an additional factor of

$$\frac{r_{\text{rough}}}{r_{\text{flat}}} = e^{-\frac{1}{2}q_{z,0}q_{z,1}\sigma^2} \tag{2.73}$$

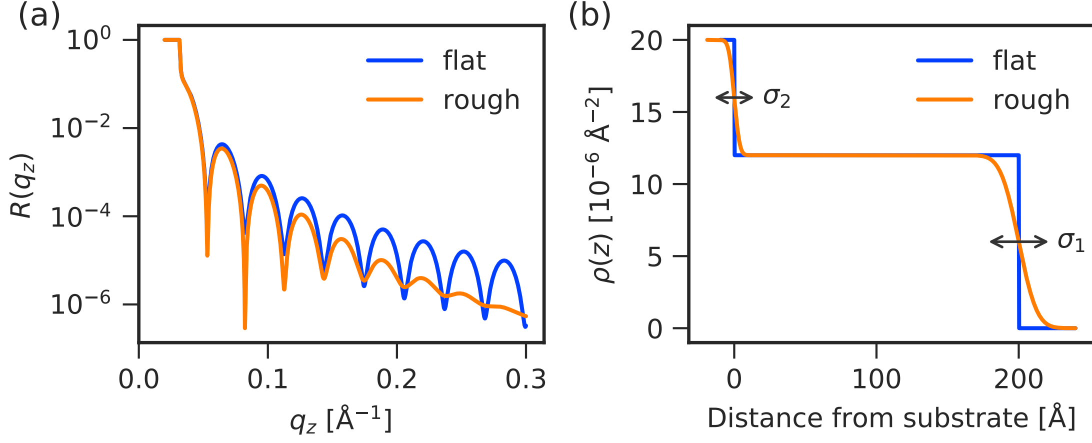

Figure 2.7.: (a) Comparison of specular reflectivity with and without roughness of a single-layer system. In the rough case, the top roughness $\sigma_1 = 10$ Å while the bottom roughness (substrate) $\sigma_2 = 3$ Å. The Kiessig fringes as well as the entire curve decay much faster compared to the case with flat interfaces. (b) The underlying SLD profile of the one-layer systems used for the simulation. The smeared-out step function indicates the roughness of the interfaces.

between the reflection coefficient of a rough interface and a flat interface. This additional factor is hence usually called the Névot-Croce factor. Another similar form of this factor is the Debye-Waller factor

$$\frac{r_{\text{rough}}}{r_{\text{flat}}} = e^{-\frac{1}{2}q_z^2\sigma^2}, \tag{2.74}$$

which can be obtained from Eq. 2.73 if refraction does not play a significant role and Eq. 2.43 holds. The Debye-Waller factor can also be derived in the kinematical approximation by using the Master formula from Eq. 2.49. Under the assumption that the height profile $z(x, y)$ of an interface follows the Gaussian distribution $p(z)$, the average SLD in the x,y plane can be described by the integral over this distribution, *i.e.*

$$\rho(z) = \langle \rho(x, y, z) \rangle_{x,y} = \int_{-\infty}^{z} p(z')(\rho_1 - \rho_0)dz' = \frac{(\rho_1 - \rho_0)}{2} \left(\text{erf}\left(\frac{z}{\sigma\sqrt{2}} \right) + 1 \right), \tag{2.75}$$

where erf(z) is the error function

$$\text{erf}(z) = \frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2} dt. \tag{2.76}$$

Consequently, the derivative of the of the SLD profile in z direction is given by

$$\frac{d\rho(z)}{dz} = p(z)\Delta\rho. \tag{2.77}$$

Thus, when performing the FT in the kinematical approximation introduced in Subsec. 2.2.3, we also obtain the Debye-Waller factor

$$\frac{r_{\text{rough}}}{r_{\text{F}}} = \frac{1}{\Delta\rho} \int_{-\infty}^{\infty} \frac{d\rho(z)}{dz} e^{iq_z z} dz = \int_{-\infty}^{\infty} p(z) e^{iq_z z} dz \propto e^{-\frac{1}{2} q_z^2 \sigma^2}. \tag{2.78}$$

It is evident that, under the influence of this type of roughness, the reflected intensity $R(q_z)$ from an interface is reduced by an exponentially decaying factor that depends on both the magnitude of the roughness σ as well as on q_z. Thus, for rough interfaces, $R(q_z)$ decays much faster. Fig. 2.7 shows a comparison of the reflectivity from flat interfaces with that from interfaces with a finite roughness, where not only the Kiessig oscillations are dampened, but also the overall intensity decays faster. While the above treatment of roughness was shown for a single interface, an analogous treatment for multi-layer systems can be applied, where the reflected amplitudes $r_{j,j+1}$ at each interface are reduced by the Névot-Croce factor depending on the respective roughness σ_{j+1}. To apply the effects of roughness to the matrix method described in Subsec. 2.2.4, it is sufficient to enforce the condition

$$\frac{r_{j,j+1}^{\text{rough}}}{r_{j,j+1}^{\text{flat}}} = e^{-\frac{1}{2} q_{z,j} q_{z,j+1} \sigma_{j+1}^2}, \tag{2.79}$$

which, using Eq. 2.68, ultimately leads to a reduction of the matrix elements $m_{j,j+1}$ and $p_{j,j+1}$ by the factors $e^{-\frac{1}{2}(k_{z,j+1}+k_{z,j})^2 \sigma_{j+1}^2}$ and $e^{-\frac{1}{2}(k_{z,j+1}-k_{z,j})^2 \sigma_{j+1}^2}$, respectively.

2.2.6. Data formats and vector notation of measured reflectivity

So far, we have introduced useful and important ways of modeling the specular reflectivity from a sample mathematically. However, in order to extract information from a given reflectometry measurement, the measured signal has to be recorded, stored and processed. For typical specular reflectometry using a point detector, the measured signal $R(q_z)$ is stored as a set of arrays containing discrete values for R and q_z. For the purpose of this work, these arrays will be referred to as the d-dimensional vectors $\boldsymbol{R} \in \mathbb{R}^d$ and $\boldsymbol{q} \in \mathbb{R}^d$, respectively, where d is the number of measured data points, $\boldsymbol{q}$ contains the discrete q_z positions where the reflectivity was measured and

$$\boldsymbol{R}_i = R(\boldsymbol{q}_i). \tag{2.80}$$

This notation is useful, because it conforms to the common tensor notation used for NNs that will be introduced in Chap. 3.

It is important to note that nowadays line detectors for lab sources and area detectors for facilities (*i.e.* synchrotrons and neutron sources) are quite common. In these cases, the reflected intensity is recorded for a larger range of outgoing angles beyond the specular condition. Usually, the multi-dimensional signal is converted to the 1-dimensional form of Eq. 2.80 before storing the data. However, increasingly, the entire detector image for each $\boldsymbol{q}_i$ is saved. This has the advantage of allowing researchers to reprocess the data at a later point, *e.g.* by integrating over a different region of interest (ROI) or changing the way the background is subtracted. So far only monochromatic measurements have been considered, but measurements can also be performed using polychromatic sources or by scanning through different wavelengths instead of different angles. While the raw data of these measurements might look different, they are also usually at some point transformed into the typical 1-dimensional form for easier data analysis.

Thus, despite the variety of raw data formats and measurement setups, for the purpose of this work all stored reflectivity data will be described in the form of Eq. 2.80 so that it is compatible with typical data analysis software [93, 95, 96].

2.3. Materials studied and sample preparation

2.3.1. Organic semiconductor thin films

This work discusses the analysis of reflectivity data stemming mainly from organic thin films on oxidized silicon substrates. These thin films consist of molecular organic semiconductors (OSCs), which are materials with conjugated π orbitals that have been intensively studied for decades due to their applications in solar cells, light-emitting diodes or transistors [123–125]. In particular, the materials used were

- Diindenoperylene (DIP) [126–128],
- Copper(II)-phthalocyanine (CuPc) [129, 130],
- α-sexithiophene (6T) [131, 132],
- Pentacene (PEN) [133, 134],
- Perylene diimide (PDI) derivatives, [135–137],
- Dinaphtho[2,3-b:2',3'-f]thieno[3,2-b]thiophene (DNTT) [138, 139],

as well as mixtures thereof. Since their specific differences in electronic properties and film growth behavior are not the primary focus of this work, the reader is referred to the respective references for a general overview.

2.3.2. Molecular beam deposition in ultra-high vacuum

All of the organic thin films studied in this work were grown on silicon (Si)/silicon oxide (SiO_x) substrates using molecular beam deposition within an ultra-high vacuum (UHV) chamber [140]. A large part of the reflectivity measurements were also conducted *in situ* during growth, as illustrated in Fig. 2.8. Other measurements were conducted post-growth, either *ex situ* or during separate annealing experiments.

It is important to note that this study was mainly concerned with the analysis of the data and not with the preparation of the samples. As such, the author of this work was not involved with the sample preparation and only partially involved in

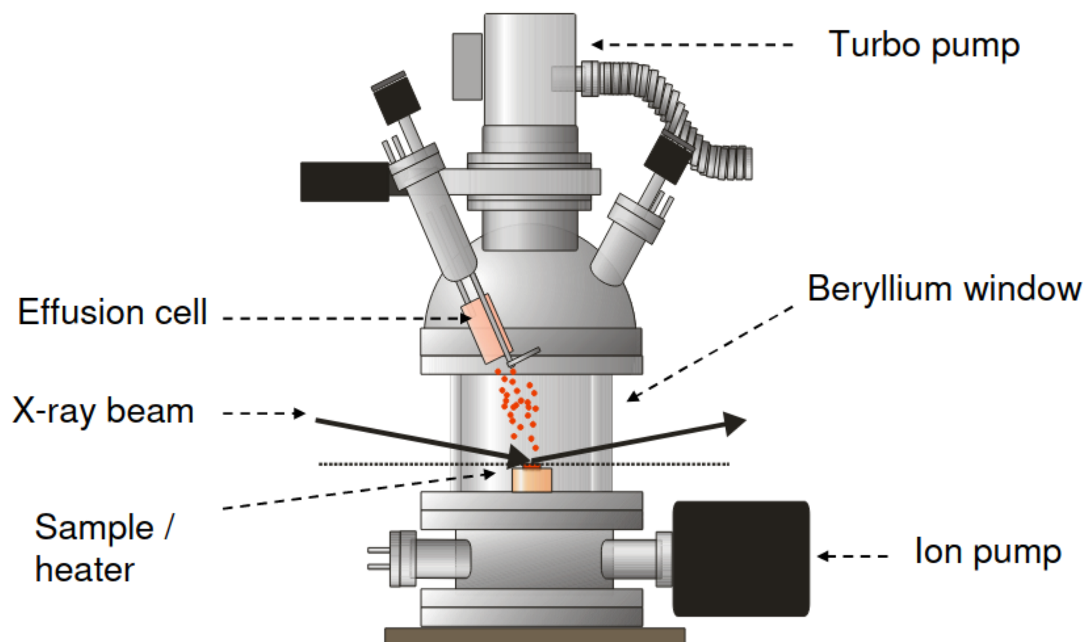

Figure 2.8.: Portable UHV chamber used for the *in situ* measurement of reflectivity during film growth. The organic molecules are deposited on the substrate at a chosen rate *via* a molecular beam by heating the effusion cell. For real-time measurements, the whole chamber is mounted on a diffractometer. Figure adopted from [89].

data acquisition. Tab. F.1 in Appendix F lists each dataset used in this work as well as its origin and credit goes to the respective people involved with sample preparation and data acquisition.

3. Machine learning fundamentals

3.1. Definition of the task from a machine learning perspective

The focus of the research presented in this thesis is the use of ML algorithms to analyze reflectivity data. The term "machine learning" describes a wide range of data-driven methods of finding mathematical models to solve a large variety of problems. According to Mitchell [141], a "learning" algorithm consists of three components: a task, an experience and a performance measure. A given algorithm is said to learn if its performance for a given task improves with increased experience. Here, experience typically refers to the exposure to data. ML algorithms are used to find a mapping between certain features $\boldsymbol{x}$ (also called "input") to a given output $\boldsymbol{y}$.

The possible tasks for ML algorithms encompass a large variety. Prominent examples are [26]

- Classification
- Machine translation
- Anomaly detection
- Synthesis
- Denoising
- Probability mass function estimation
- Regression

Out of those, regression is the most relevant for the scientific results discussed in this thesis (Part II). In this context, regression refers to the approximation of a

heuristic function with a ML model based on empirical data. Furthermore, we can distinguish between different types of learning strategies, such as supervised, unsupervised and reinforcement learning. In this thesis, we exclusively discuss supervised learning, where the ML model is designed to predict properties about the data based on past experiences.

In this work, the task to be solved is the extraction of physical information about a given sample from its reflected intensity $R(q_z)$. In Sec. 2.2 it was shown that $R(q_z)$ is closely related to the SLD profile $\rho(z)$ of the sample. Through the introduction of the matrix method in Subsec. 2.2.4, we have seen that $\rho(z)$ can be parameterized by the thickness h, roughness σ and average SLD ρ of each film layer. Thus, the matrix formalism can be considered to be a function

$$m(\boldsymbol{p}) = \boldsymbol{R} \tag{3.1}$$

that transforms a vector of sample parameters

$$\boldsymbol{p} = \begin{bmatrix} h \\ \sigma \\ \rho \\ \vdots \end{bmatrix} \tag{3.2}$$

into a vector of reflectivity values $\boldsymbol{R}$. To fulfill the task of extracting the parameters $\boldsymbol{p}$ from a given reflectivity curve $\boldsymbol{R}$, it is necessary to find the inverse function

$$M(m(\boldsymbol{p}); \boldsymbol{w}) = M(\boldsymbol{R}; \boldsymbol{w}) = \boldsymbol{p}. \tag{3.3}$$

Here, $\boldsymbol{w}$ represents a tensor of learnable parameters that can be adjusted through regression, that we shall specify later. Fig. 3.1 shows a simplified and abstracted plot of the mapping of individual reflectivity curves $\boldsymbol{R}$ to their corresponding parameter set. The data points represent the experience that is available to the ML algorithm to learn the underlying function $M(m(\boldsymbol{p}); \boldsymbol{w})$. In reality, this is a highly non-linear

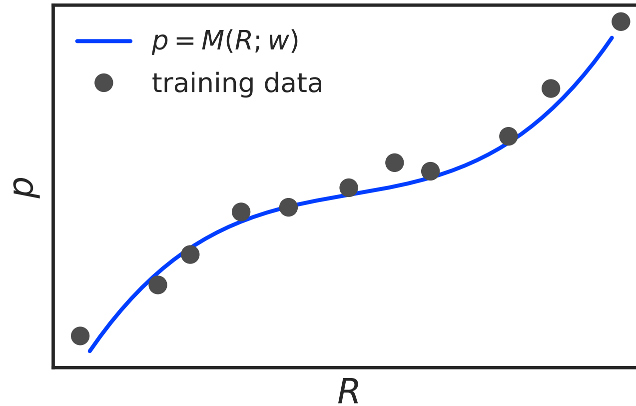

Figure 3.1.: A simplified and abstracted illustration of the mapping of individual reflectivity curves $\boldsymbol{R}$ to their corresponding sample parameters $\boldsymbol{p}$. The goal of a ML algorithm is to approximate the underlying mapping function $M(\boldsymbol{R};\boldsymbol{w})$ using non-linear regression. In reality, the mapping is highly non-linear, multidimensional and not always unique.

and multidimensional problem that requires non-linear regression to be solved. Also, due to the phase problem and experimental artifacts, $\boldsymbol{R} \rightarrow \boldsymbol{p}$ is not unique for all $\boldsymbol{R}$, *i.e.* it is not a single-valued function. However, in the following we shall assume that at least for a subset of $\boldsymbol{R}$, the mapping can be approximated by a function $M(\boldsymbol{R};\boldsymbol{w})$ within a finite error bar.

Furthermore, the question arises of how to generally model $M(\boldsymbol{R};\boldsymbol{w})$, *i.e.*, which and how many terms should be used and what form of non-linearity they should take on. Chapters 4–6 demonstrate the use of NNs to solve this regression problem, since they are very flexible models. The next chapters will describe the basic components of a NN and ways to use it to solve a regression task.

3.2. Feedforward neural networks

3.2.1. Basic terms and linear regression

Since NNs share many similarities with linear regression, we can use it as a starting point to introduce basic terms and concepts of ML. Let us first consider a function

$f(\boldsymbol{x};\boldsymbol{w},\boldsymbol{b})$ with the input vector $\boldsymbol{x} \in \mathbb{R}^n$ and the output vector $\boldsymbol{y} \in \mathbb{R}^m$. Furthermore, the function is parameterized by a tensor of weights $\boldsymbol{w} \in \mathbb{R}^{n\times m}$ and a tensor of biases $\boldsymbol{b} \in \mathbb{R}^m$, so that

$$\begin{aligned} f(\boldsymbol{x};\boldsymbol{w},\boldsymbol{b}) &= \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + \boldsymbol{b} = \boldsymbol{y}^* \quad (3.4) \\ &= \begin{bmatrix} w_{11} & \dots & w_{n1} \\ \vdots & \ddots & \vdots \\ w_{1m} & \dots & w_{nm} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix} = \begin{bmatrix} y_1^* \\ \vdots \\ y_m^* \end{bmatrix}, \quad (3.5) \end{aligned}$$

where $\boldsymbol{w}^{\mathrm{T}}$ is the transposed weight tensor and $\boldsymbol{y^*}$ the predicted output vector for non-optimized weights and biases. As we can see for the special case of $m = 1$, *i.e.*

$$f(\boldsymbol{x};\boldsymbol{w},b_1) = \begin{bmatrix} w_{11} & \dots & w_{n1} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} + b_1 = w_{11}x_1 + \ldots + w_{n1}x_n + b_1 = y_1^*, \quad (3.6)$$

the weights $\boldsymbol{w}$ determine how the input features $\boldsymbol{x}$ are mapped to the output y_j^* through the multiplication of each feature x_i with the weight w_{ij} with an additional offset by b_1. For convenience, we shall omit the explicit mention of the biases $\boldsymbol{b}$ from here on, since their treatment is analogous to that of the weights $\boldsymbol{w}$.[1]

Since the task of an ML algorithm is to solve the regression problem of finding the parameters $\boldsymbol{w}$ for which

$$f(\boldsymbol{x};\boldsymbol{w}) = \boldsymbol{y}^* = \boldsymbol{y}, \quad (3.7)$$

it is useful to introduce a performance measure that indicates how close we are to achieving this condition. This metric is usually called the "objective function", "cost function" or "loss function" and will in the following simply be called the "loss" L. If we assume that $\boldsymbol{X}$ is a dataset of N input vectors $\boldsymbol{x}_i$ and $\boldsymbol{Y}$ a dataset of N corresponding output vectors $\boldsymbol{y}_i$ that together constitute the targets of the

[1] The bias b can simply be treated as another weight w_{b} for an additional constant input $x_{\mathrm{b}} = 1$.

regression, a simple and common performance measure is the MSE between each instance of the expected output $\boldsymbol{y}_i$ and the output predicted by the model $\boldsymbol{y}_i^*$. Since the MSE is also often used as an objective function in curve fitting procedures, such as the conventional fitting of reflectivity curves, it might seem like an intuitive choice. Its origin, however, lies in statistics, where minimizing the MSE is equivalent to maximizing the log-likelihood of the model $f(\boldsymbol{x};\boldsymbol{w})$ if each output is assumed to follow a normal distribution [142].[1] Thus, using the MSE we can write the loss for a given pair of inputs and outputs $(\boldsymbol{x},\boldsymbol{y})$ as

$$\mathcal{L}(\boldsymbol{y}^*,\boldsymbol{y}) = (\boldsymbol{y}^* - \boldsymbol{y})^2 \tag{3.8}$$

$$= (f(\boldsymbol{x};\boldsymbol{w}) - \boldsymbol{y})^2. \tag{3.9}$$

The total loss for the entire dataset is then given by

$$L(\boldsymbol{X},\boldsymbol{Y};\boldsymbol{w}) = \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}(\boldsymbol{y}_i^*,\boldsymbol{y}_i) = \frac{1}{N}\sum_{i=1}^{N}\left(f(\boldsymbol{x}_i;\boldsymbol{w}) - \boldsymbol{y}_i\right)^2. \tag{3.10}$$

For this demonstrative example, L can be minimized solving for

$$\nabla_{\mathbf{w}} L(\boldsymbol{X},\boldsymbol{Y};\boldsymbol{w}) = 0, \tag{3.11}$$

which we shall omit here. However, for many regression tasks, such as the ML methods discussed in this work, it is not feasible to find the minimum of the loss analytically and thus, the task has to be solved numerically. In the field of ML, the process of numerically finding the minimum of L through the iterative optimization of $\boldsymbol{w}$ is called “training”. By far the most prominent and successful of these algorithms is stochastic gradient descent (SGD), which will be described in Subsec. 3.3.1.

[1] In general, each output can also have a different standard deviation. Then, maximizing the log-likelihood would be the same as minimizing χ^2, where the standard deviation of each output effectively acts as weighting factor for the terms in the MSE.

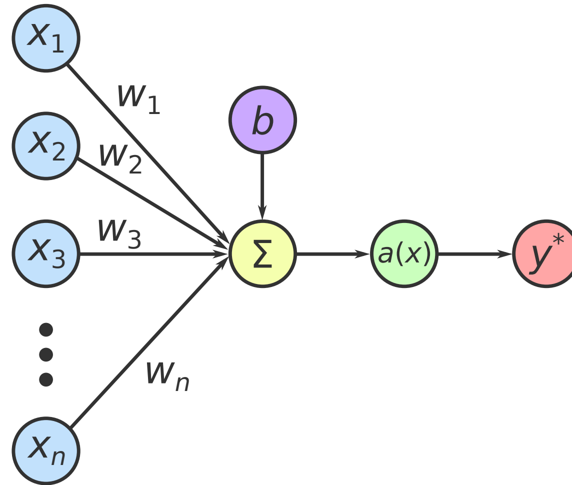

Figure 3.2.: Schematic graph-view representation of a perceptron. A set of n input features x_i (usually represented as the vector $\boldsymbol{x}$) are each multiplied with a corresponding weight w_i (usually represented as the tensor $\boldsymbol{w}$). Their sum plus the bias b is then passed through an activation function $a(x)$ to obtain the predicted output y^*.

3.2.2. The multi-layer perceptron

In the context of ML, the term neural network describes a class of functions that are nowadays used to solve a large variety of ML problems [142, 143]. While NNs as a mathematical concept have already been known for decades, only recently they have become viable and popular tools. The simplest form of a NN is called FCNN or multi-layer perceptron (MLP), which in turn is composed of building blocks called "perceptrons" [15]. A perceptron is simply the function used for linear regression from Eq. 3.6 wrapped by a non-linear activation function $a(x)$, *i.e.*

$$y^* = a\left(b + \sum_{i=1}^{N} w_i x_i\right) = a(b + \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}). \tag{3.12}$$

A typical graph representation of a perceptron is shown in Fig. 3.2. The purpose of the activation function is mainly to introduce nonlinearities into the NN, which is ultimately required to perform non-linear regression[1]. Activation functions sometimes

[1]Without activation functions, a NN is just a large sum of linearly weighted terms and can only model linear functions.

take the shape of sigmoid functions, such as the logistic function, the hyperbolic tangent or the error function. These loosely mimic the “on” and “off” states of biological neurons, since they provide a way to measure the relative degree of activation for the perceptron between the lower and upper bound of the function. An example of the logistic function and its derivative are shown in Fig. 3.3. Another popular family of activation functions are modified linear functions, the most prominent being the rectified linear unit (ReLU), which is used exclusively in this work. ReLU (shown in Fig. 3.3) is defined as

$$a(x) = \begin{cases} x \text{ for } x > 0 \\ 0 \text{ for } x \leq 0 \end{cases} \tag{3.13}$$

if the argument of a is a scalar. If the argument is a tensor, then a is applied to each scalar tensor element independently. Over the last decade, ReLU has become the default activation function for many NN models because its similarity to linear functions makes it is easy to handle numerically [144, 145]. An important example of this is that its derivative is fast to compute and, in contrast to sigmoid-based functions, does not approach 0 if x is large, which helps to reduce the problem of vanishing gradients during the training step. Many variations of ReLU (*e.g.* “leaky” ReLU) have been developed over the years to address some nuanced shortcomings of ReLU for certain applications [146, 147], however, they will not be discussed here.

A model with multiple outputs, that is, with an output vector $\boldsymbol{y}^*$, can be achieved by expanding the dimensions analogous to Eq. 3.4 so that

$$f(\boldsymbol{x}; \boldsymbol{w}) = a(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}) = \boldsymbol{y}^*. \tag{3.14}$$

This is equivalent to using several perceptrons in parallel. Again, as described in the previous section, the bias tensor $\boldsymbol{b}$ was dropped for convenience.

According to ML terminology, $\boldsymbol{x}$ and $\boldsymbol{y}^*$ are typically called the “input layer” and “output layer”, respectively. Also, the individual vector elements of each layer are often called “neurons”. It is apparent that despite the non-linearity, the range of functions that can be approximated using this function is still very limited. A

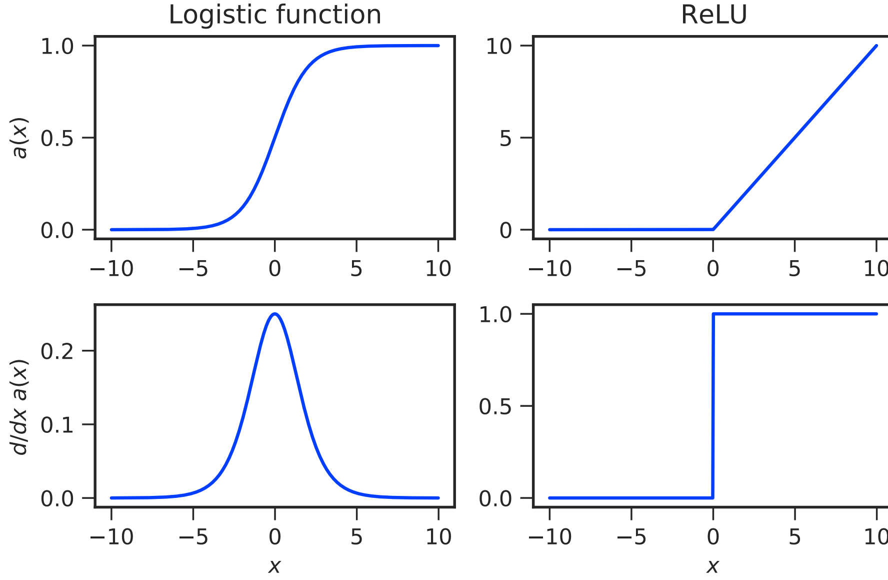

Figure 3.3.: An illustration of two common non-linear activation functions, the logistic function and the ReLU function, as well as their respective derivatives. Since the derivative of the logistic function approaches 0 for both high and low inputs, it can prevent efficient gradient descent. It is important to note that for the results in Part II of this thesis, ReLU was used exclusively since it generally performed better.

more flexible model is the so-called MLP, where several perceptrons are connected in series where the output of one perceptron serves as the input for the next. Thus, for a two-layer perceptron, we can define each perceptron as

$$f_{1/2}(\boldsymbol{x};\boldsymbol{w}_{1/2}) = \boldsymbol{y}^*_{1/2}, \tag{3.15}$$

respectively. The MLP is then given by

$$f(\boldsymbol{x};\boldsymbol{w}) = f_2(\boldsymbol{y}^*_1;\boldsymbol{w}_2) = f_2(f_1(\boldsymbol{x};\boldsymbol{w}_1);\boldsymbol{w}_2) = \boldsymbol{y}^*, \tag{3.16}$$

where $\boldsymbol{y}^*_2 = \boldsymbol{y}^*$. A graph representation of this MLP is shown in Fig. 3.4. Here, $\boldsymbol{y}^*_1$ is typically called “hidden layer” and serves as a purely intermediate output that does not directly correspond to the actual output $\boldsymbol{y}^*$. Thus, $\boldsymbol{y}^*_1$ can in principle take on

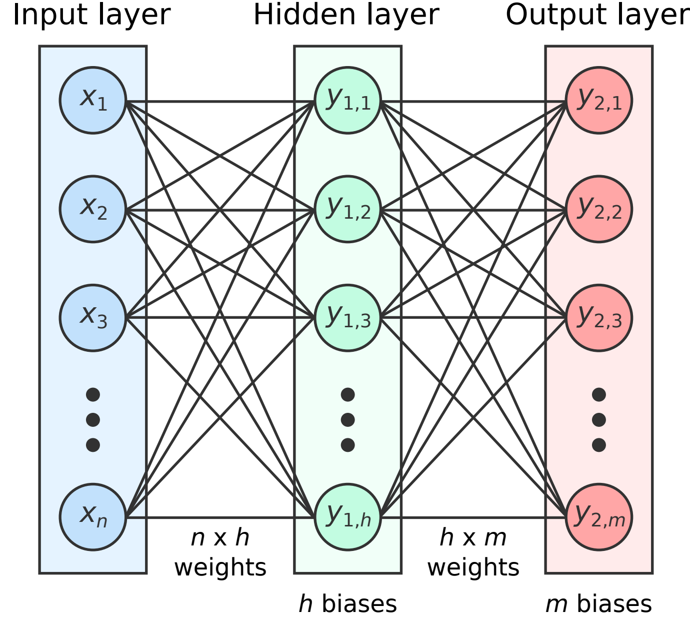

Figure 3.4.: Schematic graph-view representation of a FCNN with one hidden layer. The FCNN can be seen as two perceptrons with multiple outputs in series, where the hidden layer is both the output of the first perceptron and the input for the second. Each numerical value in this network (circles) is called a “neuron”.

any dimensionality, which, in turn, affects the dimensionality of the parameters $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$.

It has been shown that a FCNN with only one hidden layer can in principle be used to approximate a broad variety of interesting functions if the size of the hidden layer, and thus, the number of parameters, is sufficiently large [148, 149]. In practice, however, NNs often have more than one hidden layer, since nesting non-linear functions in this way makes it often easier to find a suitable minimum during training.

In terms of solving the task of analyzing reflectivity that was laid out in Sec. 3.1, the function describing the NN model, f, corresponds to the inverse mapping function M from Eq. 3.3. Thus, the input of the NN is the vector of reflectivity values $\boldsymbol{R}$ and the output is the vector of sample parameters $\boldsymbol{p}$, *i.e.*

$$f(\boldsymbol{x}; \boldsymbol{w}) = \boldsymbol{y}^* = M(\boldsymbol{R}; \boldsymbol{w}) = \boldsymbol{p}. \tag{3.17}$$

3.3. Training neural networks

3.3.1. Stochastic gradient descent

After having introduced the idea of regression in the context of ML, we shall now introduce a method to train the ML model $f(\boldsymbol{x}; \boldsymbol{w})$, *i.e.*, determine the concrete values for the parameters $\boldsymbol{w}$ such that the loss $L(\boldsymbol{X}, \boldsymbol{Y}; \boldsymbol{w})$ is minimized with regard to the regression targets $\boldsymbol{X}$ and $\boldsymbol{Y}$. In contrast to the example of linear regression shown in the previous section, it is not feasible to simply calculate a closed-form solution to find the minimum of the loss function and thus, it has to be numerically optimized. Most ML methods use optimization algorithms based on a well-established iterative method called stochastic gradient descent (SGD) [150]. The variations of SGD that are most commonly used (also within this work) are described later in Sec. 3.5.

Gradient descent is a method of minimizing a function with respect to a set of parameters in iterative steps along the negative gradient until a local minimum is reached and the algorithm converges. During each step, the gradient

$$\boldsymbol{g} = \nabla_{\boldsymbol{w}} L(\boldsymbol{w}) \tag{3.18}$$

is calculated for the current weights $\boldsymbol{w}$ by calculating the partial derivative for each parameter in the model given $\boldsymbol{X}$ and $\boldsymbol{Y}$. All parameters are then updated along the gradient according to the rule

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \alpha \boldsymbol{g}, \tag{3.19}$$

where α is a scalar called the "learning rate" that defines the size of each step. The question of what value α should take is difficult to answer in general, since it strongly depends on the shape of the loss surface. Fig. 3.5 shows an example of non-linear regression with only one parameter and a convex loss for different learning rates. Even in this simplified example, an α that is too large or too small can significantly hinder convergence. This is even more so the case with highly multidimensional loss

surfaces that are not always convex and have multiple local minima, where choosing an appropriate learning rate can be crucial for the algorithm to converge at all. For this reason, usually adaptive optimization algorithms are employed (see Sec. 3.5).

In the field of ML, the process of minimizing the loss is called "training", and the dataset of regression targets $(\boldsymbol{X}^{\text{train}}, \boldsymbol{Y}^{\text{train}})$ is called the "training set". For many ML problems, it is advantageous to have a training set that is as large as possible. However, this can make calculating the gradient for each update computationally slow. SGD solves this problem by only using a random subset of the training set to calculate the gradient of the loss for each update. These subsets are usually called "minibatches", and thus this method is sometimes referred to as minibatch SGD [151]. The number of instances within each minibatch are typically called the "batch size". By choosing a random minibatch from the training set, we can assume that the gradient and resulting update are, on average, representative of the entire test set. Furthermore, a given instance is only reused for training once all other instances within the training set have been used. The number of steps necessary for each minibatch to be used once is called an "epoch". For example, assuming the training set contains 100,000 instances and the batch size is 100, then after 1000 updates one epoch has passed, *i.e.* each instance has been used exactly once.

3.3.2. Gradient calculation through back-propagation

As shown in the previous section, conventional learning algorithms for ML methods require the calculation of the gradient of the loss function $\nabla_{\boldsymbol{w}} L(\boldsymbol{X}^{\text{mb}}, \boldsymbol{Y}^{\text{mb}}; \boldsymbol{w})$ for each minibatch in order to perform an update of the weights $\boldsymbol{w}$. Since NNs with multiple layers are nested functions, an algorithm called "back-propagation" [16] (sometimes simply called backprop) is used to efficiently calculate the gradient. At the heart of back-propagation is the chain rule of calculus which states that the derivative of a nested function

$$f(x) = f(g(x)) \tag{3.20}$$

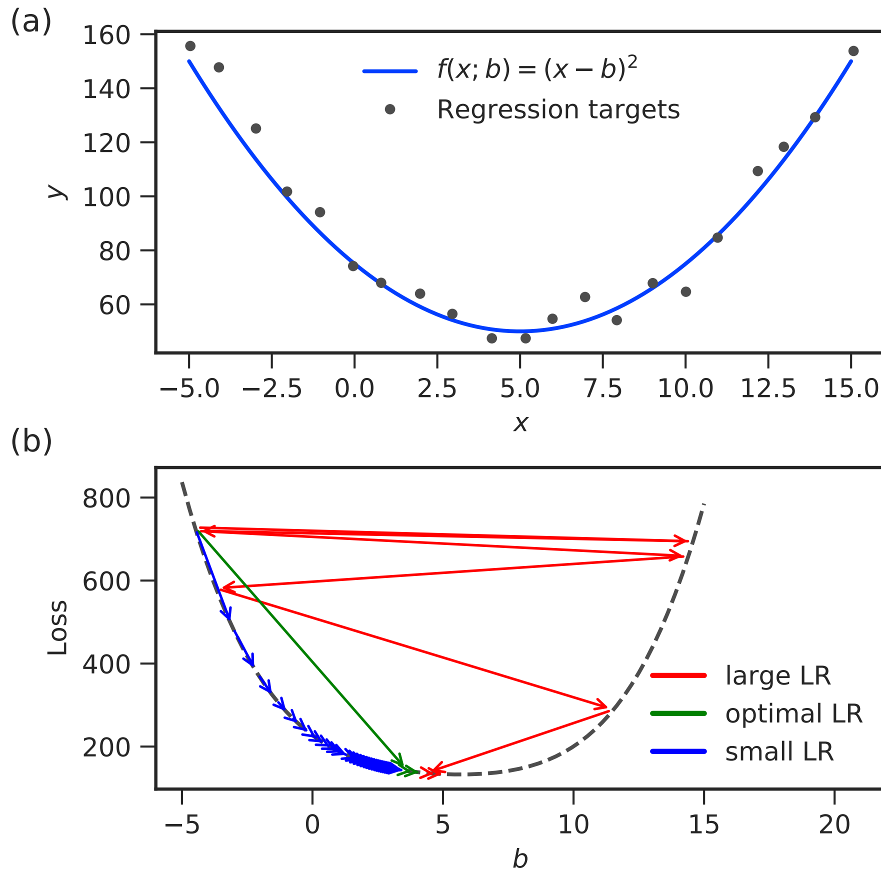

Figure 3.5.: Example of the gradient descent algorithm for the non-linear regression of a parabolic function $f(x; b) = (x - b)^2$ with one parameter. (a) Plot of the regression targets, *i.e.* training data, and the optimal function for $b = 5$. (b) Shows the discrete steps of the gradient descent algorithm on its way to the minimum for three different learning rates. If the learning rate is too high, the steps skip over the minimum multiple times or the algorithm might diverge. If the learning rate is too small, it can take very long for the algorithm to converge due to a small step size. With an adequate learning rate, the algorithm only needs a few steps to converge.

can be written as

$$\frac{df}{dx} = \frac{df}{dg}\frac{dg}{dx}. \tag{3.21}$$

We can easily demonstrate the usefulness of the chain rule using the NN with a single hidden layer given by Eq. 3.15. Here, the loss for a given set of inputs and outputs can be written as

$$\begin{aligned} \mathcal{L}(\boldsymbol{y}^*, \boldsymbol{y}) &= (\boldsymbol{y}^* - \boldsymbol{y})^2 & (3.22) \\ &= (f_2(f_1(\boldsymbol{x}; \boldsymbol{w}_1); \boldsymbol{w}_2) - \boldsymbol{y})^2. & (3.23) \end{aligned}$$

The loss for an entire minibatch is then given by

$$L(\boldsymbol{X}^{\text{mb}}, \boldsymbol{Y}^{\text{mb}}; \boldsymbol{w}) = \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}(\boldsymbol{y}^*, \boldsymbol{y}), \tag{3.24}$$

where i is the index in a given minibatch and N is the minibatch size.

It is instructive to first consider the case where the input, output and hidden layer are all scalars, *i.e.* $x, y, w_1, w_2 \in \mathbb{R}$ and $f_1, f_2 : \mathbb{R} \to \mathbb{R}$. The per-input loss is then given by

$$\mathcal{L}(y^*, y) = (f_2(f_1(x; w_1); w_2) - y)^2. \tag{3.25}$$

Using the chain rule, we can write the derivative of that loss as

$$\frac{d\mathcal{L}}{dw} = \frac{\partial \mathcal{L}}{\partial w_1} + \frac{\partial \mathcal{L}}{\partial w_2} = \frac{\partial \mathcal{L}}{\partial f_2}\frac{\partial f_2}{\partial f_1}\frac{\partial f_1}{\partial w_1} + \frac{\partial \mathcal{L}}{\partial f_2}\frac{\partial f_2}{\partial w_2}. \tag{3.26}$$

All of the involved partial derivatives are straightforward to calculate since each function is simply a weighted sum of parameters. For example, the derivative of the hidden layer with regard to its weights is given by

$$\frac{\partial f_1}{\partial w_1} = \frac{\partial}{\partial w_1} a(w_1 x + b) = \begin{cases} x \text{ for } w_1 x + b > 0 \\ 0 \text{ for } w_1 x + b \leq 0 \end{cases}, \tag{3.27}$$

where $a(x)$ is the ReLU function described earlier.

The same principle can be applied to the more general, multidimensional case where $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{y} \in \mathbb{R}^m$, $\boldsymbol{w}_1 \in \mathbb{R}^{n\times h}$, $\boldsymbol{w}_2 \in \mathbb{R}^{h\times m}$, $f_1 : \mathbb{R}^n \to \mathbb{R}^h$ and $f_2 : \mathbb{R}^h \to \mathbb{R}^m$, with the only difference being a larger number of cross terms. Thus, analogous to Eq. 3.26, we can write the gradient of the loss with respect to the weights as

$$\nabla_{\boldsymbol{w}}\mathcal{L} = \nabla_{\boldsymbol{w}_1}\mathcal{L} + \nabla_{\boldsymbol{w}_2}\mathcal{L} = \nabla_{\boldsymbol{f}_2}\mathcal{L}\boldsymbol{J}^{\boldsymbol{f}_2}_{\boldsymbol{f}_1}\boldsymbol{J}^{\boldsymbol{f}_1}_{\boldsymbol{w}_1} + \nabla_{\boldsymbol{f}_2}\mathcal{L}\boldsymbol{J}^{\boldsymbol{f}_2}_{\boldsymbol{w}_2}, \tag{3.28}$$

where

$$\boldsymbol{J}^{\boldsymbol{a}}_{\boldsymbol{b}} = \begin{bmatrix} \nabla^{\mathrm{T}}_{\boldsymbol{b}} a_1 \\ \vdots \\ \nabla^{\mathrm{T}}_{\boldsymbol{b}} a_n \end{bmatrix} = \begin{bmatrix} \frac{\partial a_1}{\partial b_1} & \cdots & \frac{\partial a_1}{\partial b_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial a_n}{\partial b_1} & \cdots & \frac{\partial a_n}{\partial b_m} \end{bmatrix} \tag{3.29}$$

denotes the $n \times m$ Jacobian matrix of $\boldsymbol{a} \in \mathbb{R}^n$ with regard to $\boldsymbol{b} \in \mathbb{R}^m$.

From this, it is evident that the chain rule can easily be applied recursively for a NN with an arbitrary number of layers and with an arbitrary number of neurons in each layer. Importantly, Eq. 3.26 and Eq. 3.28 also show that some terms appear multiple times when using this technique. Thus, to avoid unnecessarily re-calculating the same terms multiple times, implementations of the back-propagation algorithm in software packages, such as TensorFlow [30], include additional computational details beyond the simple chain rule. Since these details are beyond the scope of this work, they will not be discussed here.

3.4. Input and output standardization

When SGD or other gradient-based methods are used to solve a regression problem, it is important to consider the scale of both the input $\boldsymbol{x}$ and the corresponding output $\boldsymbol{y}$, since their scale implicitly affects the scale of all parameters $\boldsymbol{w}$. For instance, if different inputs or outputs differ by orders of magnitude, the optimal values for different parameters may also differ correspondingly. Combined with the fact that the parameters are usually initialized all on the same scale [152], this can lead to a very distorted loss surface where two optimal parameters w_1^{opt} and w_2^{opt} are

orders of magnitude apart. This also leads to a gradient with a different scale in some directions compared to others and raises the question of which learning rate to choose: a small learning rate might be optimal for w_1, but would lead to very small steps relative to w_2, effectively increasing the necessary training time. Conversely, choosing a larger learning rate that is optimal for w_2 can lead to steps that are too large for w_1, meaning it will often skip over minima, hindering convergence and also increasing training time. An illustration of this is given in Fig. 3.6.

This is particularly relevant when dealing with reflectivity data, since data points measured at different q_z values can easily vary by many orders of magnitude. For these reasons, it is common to normalize all inputs x_i to be on a similar scale before using any training algorithm. The most common form of this is called standardization, where the distribution of each input x_i in the training data is assumed to follow a normal distribution. Here, each input is transformed so that

$$\hat{x}_i = \frac{x_i - \mu_i}{\sigma_i}, \tag{3.30}$$

where μ_i and σ_i are the mean and standard deviation. In the context of training NNs with reflectometry data, this method was first introduced in [107] (see Chap. 5). In the case of input data that is obtained from measurements, such as reflectometry, it is possible to choose a form of input transformation that makes use of physical knowledge about the underlying distribution of data points rather than simply assuming a normal distribution. Using the kinematical approximation for reflectivity, for example, one can assume that $R(q_z) \propto q_z^{-4}$ and thus multiply each reflectivity data by its corresponding value of q_z^4 to normalize the data points. While this might seem preferable to standardization, it usually performs similar or worse than standardization due to the fact that the kinematic approximation does not hold near the TRE, which leads to divergences near the critical angle.

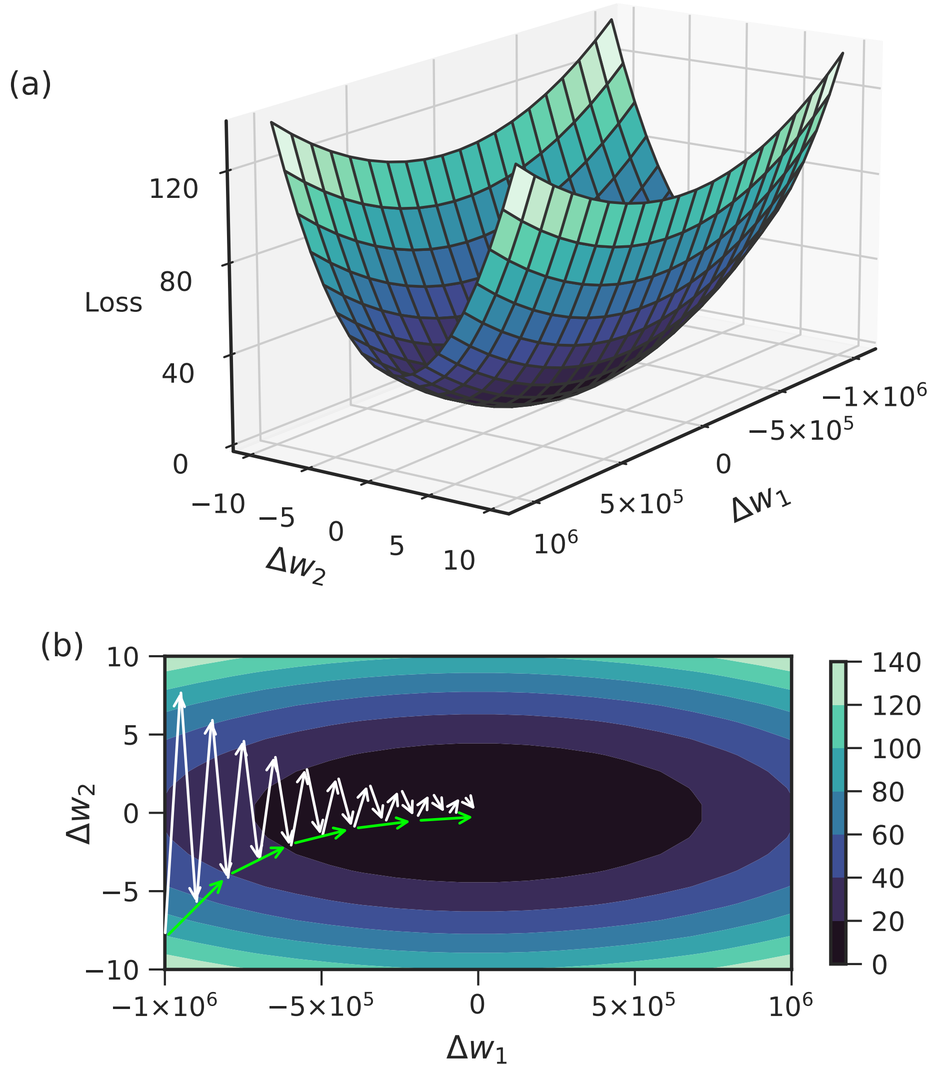

Figure 3.6.: (a) Example of a heavily-stretched 2-dimensional loss surface due to inputs on different orders of magnitude. The axes Δw_1 and Δw_2 represent the distance of the weights w_1 and w_2 from their optimal values. Since $x_1 \sim 1 \times 10^{-6}$ and $x_2 \sim 1 \times 10^{1}$, Δw_1 and Δw_2 are 6 orders of magnitudes apart. (b) Contour plot of the loss surface shown in (a). The white arrows indicate a gradient descent algorithm that skips over the minimum for Δw_2 due to the absolute value of the gradient $\nabla_{w_1,w_2} L$ being dominated by w_1. This leads to many more necessary steps to converge than the optimal case indicated with green arrows.

3.5. Adaptive optimizers

Since finding the minimum of the training loss using SGD can be both difficult and slow when the number of optimized parameters is large, several other optimization algorithms have been developed in recent years. One of the most important classes of optimizers is called "adaptive algorithms", since they include additional terms that take past updates into account. Here, we shall discuss three optimizers that are relevant for the work of this thesis: SGD with momentum, RMSprop and ADAM.

3.5.1. SGD with momentum

SGD with momentum is an adaptation of the SGD algorithm with the addition of exponentially decaying terms of previously calculated gradients [153]. Each update of the weights is hereby defined as

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \boldsymbol{v}, \tag{3.31}$$

where v represents an accumulating velocity term that is usually initialized as $\boldsymbol{v} = 0$. Before each weight update, the velocity term is updated based on the previous velocity and the current gradient, *i.e.*

$$\boldsymbol{v} \leftarrow \beta_1 \boldsymbol{v} - \alpha \boldsymbol{g}. \tag{3.32}$$

This means that the value $\boldsymbol{v}$ by which the weights $\boldsymbol{w}$ are updated accumulates over past gradients, where the parameter $\beta_1 \in [0, 1)$ represents an exponentially decaying weighting factor of the past gradients. This has the effect that, if the gradient points into a similar direction for several updates in a row, the steps in that direction become larger with each update. On the other hand, if the gradient changes erratically in one or more dimensions, the velocity terms in those dimensions "average out" over time.

SGD with momentum can also be described using a physical analogy in the form of discretized Newtonian dynamics, where the loss $L(\boldsymbol{w})$ acts as an energy. If we

assume a particle of unity mass that rests at position $\boldsymbol{w}_t$ at time step t, then we can say the particle experiences a force

$$\boldsymbol{F}(t) = \boldsymbol{a}(t) = \frac{\partial^2}{\partial t^2}\boldsymbol{w}(t) = \frac{\partial}{\partial t}\boldsymbol{v}(t) = -\gamma \boldsymbol{v} - \boldsymbol{g}. \tag{3.33}$$

In a discretized form we can write

$$\boldsymbol{a}_t = -\gamma \boldsymbol{v}_t - \boldsymbol{g}_t \tag{3.34}$$

and thus

$$\boldsymbol{v}_{t+1} = \boldsymbol{v}_t + \boldsymbol{a}_t \Delta t = \boldsymbol{v}_t + (-\gamma \boldsymbol{v}_t - \boldsymbol{g}_t)\Delta t = \beta_1 \boldsymbol{v}_t - \alpha \boldsymbol{g}_t, \tag{3.35}$$

which is analogous to Eq. 3.32 if $\beta_1 = (1 - \gamma \Delta t)$ and $\Delta t = \alpha$. The two forces acting on the particle are the gradient $\boldsymbol{g}$ as well as the term $\gamma \boldsymbol{v}$, which can be seen as viscous drag on the particle. The former term causes the particle to pick up speed along steep gradients, which allows it to ignore small deviations in the gradient and efficiently descend to the minimum. The latter term ensures that the particle will slow down over time and come to rest once it has found a minimum.

3.5.2. RMSprop

RMSprop is itself an adaptation of the ADAgrad (short for "adaptive gradient") algorithm, which is designed to slow down each consecutive update by dividing each step by the root mean squares of all past gradients. However, in order to avoid an overly fast decay of the step size, RMSprop additionally introduces an exponential decay to each squared gradient term. The update rule is then defined as

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \alpha \frac{\boldsymbol{g}}{\sqrt{\boldsymbol{s}} + \delta}, \tag{3.36}$$

where $\boldsymbol{s}$ is the weighted sum of squared gradients

$$\boldsymbol{s} \leftarrow \beta_2 \boldsymbol{s} + (1 - \beta_2)\boldsymbol{g}^2, \tag{3.37}$$

usually initialized as $\boldsymbol{s} = 0$.[1] Here, $\beta_2 \in [0, 1)$ is a weighting parameter for past squared gradients similar to β_1, however, without a simple physical analogy. The additional parameter $\delta \sim 10^{-8}$ is simply a small number added for numerical stability to avoid division by 0.

The advantage of RMSprop comes into play when descending along the gradient of a convex structure on the loss surface. Here, the algorithm helps avoid skipping over the minimum by slowing down the learning if the recent gradient was very steep, but avoids unnecessarily slowing down the learning process near the minimum, *i.e.* when the past gradients were relatively small. Both of these features allows the algorithm to safely converge, however, sometimes at the cost of speed, especially if the position on the loss surface is far from a minimum. Nonetheless, RMSprop has been a very popular and successful optimizer for many problems in the past.

3.5.3. ADAM

The ADAM algorithm (short for “adaptive moments”) is arguably one of the most widely-used and successful optimizers [154]. It combines RMSprop with a variation of SGD with momentum into a single algorithm, which gives it a great amount of flexibility. The update rule of the ADAM optimizer is given by

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \alpha \frac{\boldsymbol{v}'}{\sqrt{\boldsymbol{s}} + \delta}, \tag{3.38}$$

where

$$\boldsymbol{v}' \leftarrow \beta_1 \boldsymbol{v}' + (1 - \beta_1)\boldsymbol{g} \tag{3.39}$$

is a slightly modified version of the momentum term from Eq. 3.32. Since ADAM combines properties from RMSprop as well as a momentum term, it converges quickly even if the initialized parameters are far from a minimum.

[1] Here, all division and multiplication operations are applied element-wise.

Out of the three described algorithms, ADAM has been found to be the most successful optimizer for the ML models discussed in Part II of this work. Thus, it will be considered the default optimizers unless specified otherwise.

3.5.4. Learning rate schedules

Aside from choosing different optimizers, another common way of adjusting the step size of each update is *via* learning rate schedules. These change the learning rate parameter α over the course of the training process. Since the learning rate is such an important parameter for training, this has been shown to help the training to converge into better minima even when used in conjunction with good optimizers. The most common applications of this are exponential or polynomial decay of the learning rate as a function of the number of steps. They have a similar effect on the training as the optimizers discussed previously, although independent of the actual loss surface $L(\boldsymbol{w})$. Another recently developed approach is using cyclical learning rate schedules, where α is decreased and then increased periodically. Increasing α again can act as a shock to the system, allowing the algorithm to dislodge from local minima and ultimately explore a larger space on the loss surface.

The method that is used in Chap. 5 and Chap. 6 is a conditional learning rate reduction, where α is reduced whenever a certain condition is met during training. Here, the learning rate is halved whenever the training loss stagnates for more than 10 epochs in a row, which often leads to a significant drop in the loss.

3.6. Validation and generalization

So far, we have discussed how a ML model can be trained using a finite number of data points called the training set. Assuming successful training, we obtain a function that can map each training input to each training output. The question remains whether the model performs similarly on data points outside of the training set. In other words, the approximated function might perfectly pass through each data point without capturing the general trend of the data. This is a common problem in

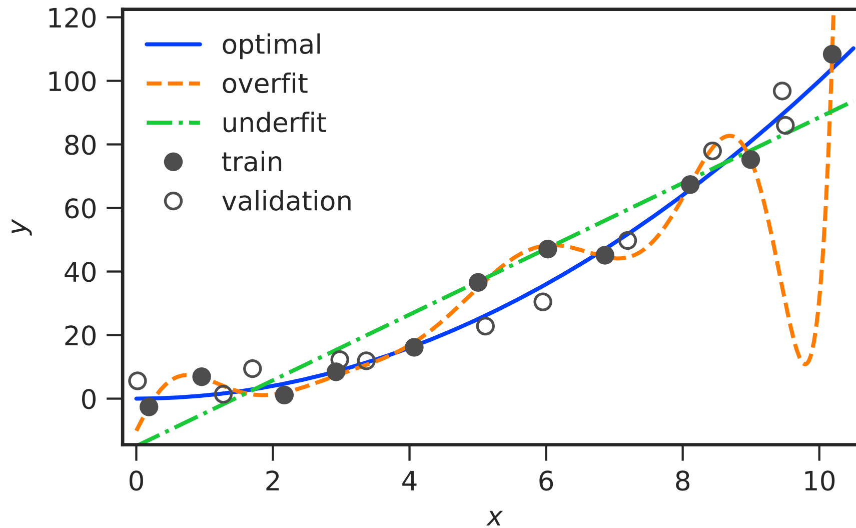

Figure 3.7.: Example of non-linear regression from a training set. The blue curve shows an optimal fit while the yellow and green curves show an over- and underfitted function. The problem with overfitting becomes especially apparent when using a second dataset for validation. This shows the need for a validation dataset to judge the level of overfitting when training ML models.

regression called "overfitting" which is also important when fitting experimental data to a physical model. The researcher often has to make a decision about the number of open parameters to include in a model in comparison to the complexity of the data. In ML, it is also common to introduce regularization mechanisms that penalize overfitting in the training loss [26, 155]. The ability of the model to approximate complex non-linear functions is sometimes called the "capacity" of the model. If the capacity is too low, the model is too simple to describe the data adequately ("underfitting"). In contrast, if the capacity is too high, the model is flexible enough to accommodate every data point at the expense of describing the underlying physical trend. An illustration of this can be seen in Fig. 3.7.

In machine learning, the ability to interpolate output beyond the training set is usually called "generalization". A common way to measure the degree of generalization is by periodically validating the model during training with data that was not used for the training itself, meaning it was not used to generate the gradient for updating the weights. This data set is usually called "validation set" and the loss

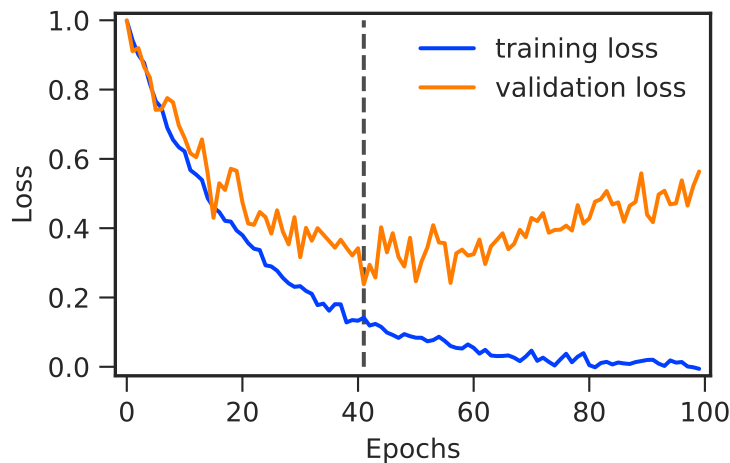

Figure 3.8.: An example of the training and validation loss during the training of a NN. The validation loss is generally higher, indicating that there is some degree of overfitting. However, up to epoch 40, the validation loss still decreases, which means that the overall performance improves. After that, however, the validation loss starts to increase, indicating that further training would be detrimental to the general performance of the model and training should be stopped.

calculated with it is called the "validation loss" (in contrast to the training set and the training loss). Typically, the validation loss is calculated after every epoch and compared to the training loss at that time. Thus, it is crucial to have a validation set that is representative of the underlying distribution of data points the model is meant to approximate. By decreasing the capacity of the model, overfitting can be decreased, however, this usually also increases the training loss. Thus, it is important to find suitable capacity that avoids both over- and underfitting.

In addition to being a function of the capacity, overfitting can also be a function of the number of training steps. In many cases, such as the ones discussed in this work, both the training and validation loss decrease initially, while, at some point, the validation loss begins to stagnate or even increase again, as shown in Fig. 3.8. This is because the loss surface $L(\boldsymbol{w})$ might have several local minima with some being more general while others are more specific to the training set. Thus, in this work, the so-called "early stopping" method is employed, where both validation and training loss are monitored simultaneously and training is stopped if the validation loss starts

increasing. Early stopping is a commonly used criterion for finding the optimal number of training epochs that has been shown to function as a regularizer [156, 157].

Part II.

Results and Discussion

4. Analysis of reflectivity data using neural networks‡

4.1. Introduction

As described in previous chapters, NR and XRR are powerful tools to determine the physical properties of surfaces, thin films and layered structures. The analysis of reflectometry data is typically done *via* iterative LMS fitting algorithms that tend to be slow and require expert knowledge. Considering the increasing speed at which reflectometry measurements can be performed, new and fast methods for data analysis have to be developed. This chapter introduces and discusses the first published application of a fully-connected neural network (FCNN) to analyze reflectivity data [103] and demonstrates that NNs can not only be used to reduce the necessary user input and the time needed to extract thin film properties from XRR data, but also promise to alleviate the requirement of *a priori* knowledge about the studied system. This makes NNs ideal for autonomous applications in real-time measurements. The discussion in this chapter is based on the performance of a FCNN with six hidden layers trained with simulated XRR data, and tested on five real-time XRR datasets of growing organic thin films on silicon (Si) substrates with a silicon oxide (SiO_x) layer.

‡This chapter is largely based on the publication Greco *et al.* 2019 [103]. The Python code for the neural network was co-developed by A. Greco with the help of V. Starostin, C. Karapanagiotis, L. Pithan, S. Liehr and S. Kowarik. The neural network design and data analysis were done by A. Greco. The samples used for testing were prepared and measured by A. Hinderhofer, C. Lorch, T. Hosokai and S. Kowarik, as described in Tab. F.1. A. Gerlach provided computational support and F. Schreiber supervised the project.

4.2. Neural network architecture and training

For this study, an FCNN using supervised learning with simulated training and validation data was used[1]. All of the code was written in Python 3.7 with the help of the TensorFlow 2.1 framework. As described in Sec. 3.2, the information is processed by NNs from a set of input neurons to a set of output neurons through multiple hidden layers of neurons. The term "layer" is used here to describe sets of neurons and should not be confused with the same term used to describe the layered structure of thin films. The input layer of the NN represents the measured X-ray intensity values $\boldsymbol{R} \in \mathbb{R}^{52}$ at different momentum transfer values $\boldsymbol{q}$ (for details about the notation see Subsec. 2.2.6). The output layer $\boldsymbol{y} \in \mathbb{R}^{4}$ corresponds to the different thin film parameters, *i.e.* film and oxide thickness, roughness, and density. A schematic of the architecture used in this study is shown in Fig. 4.1. In the case of fully-connected models, such as the one described herein, the value of each neuron after the input layer is calculated by a weighted sum of all neurons in the previous layer. This way, for any given reflectivity curve a corresponding output can be calculated. In all cases, ReLU was used as an activation function, which is a common default setting that performs well on many tasks.

The NN model employed in this chapter (Fig. 4.1) consists of six fully-connected hidden layers with 400, 800, 400, 300, 200 and 100 neurons. For the results discussed here, the output of three independently trained NNs with the same hyperparameters and training data, but random initialization, was averaged to compensate for outliers in the prediction. Both the simulated and the experimental data were normalized to 1 and passed through a log function before being used as input. This was done to reduce the number of orders of magnitude over which the input data are distributed. A wide distribution of input values is a common problem that can inhibit training, as described Sec. 3.4. In Chap. 5, a more sophisticated normalization method is discussed. Furthermore, each output parameter of the model was normalized to

[1]The code used in this chapter was published on Zenodo at https://doi.org/10.5281/zenodo.3478344.

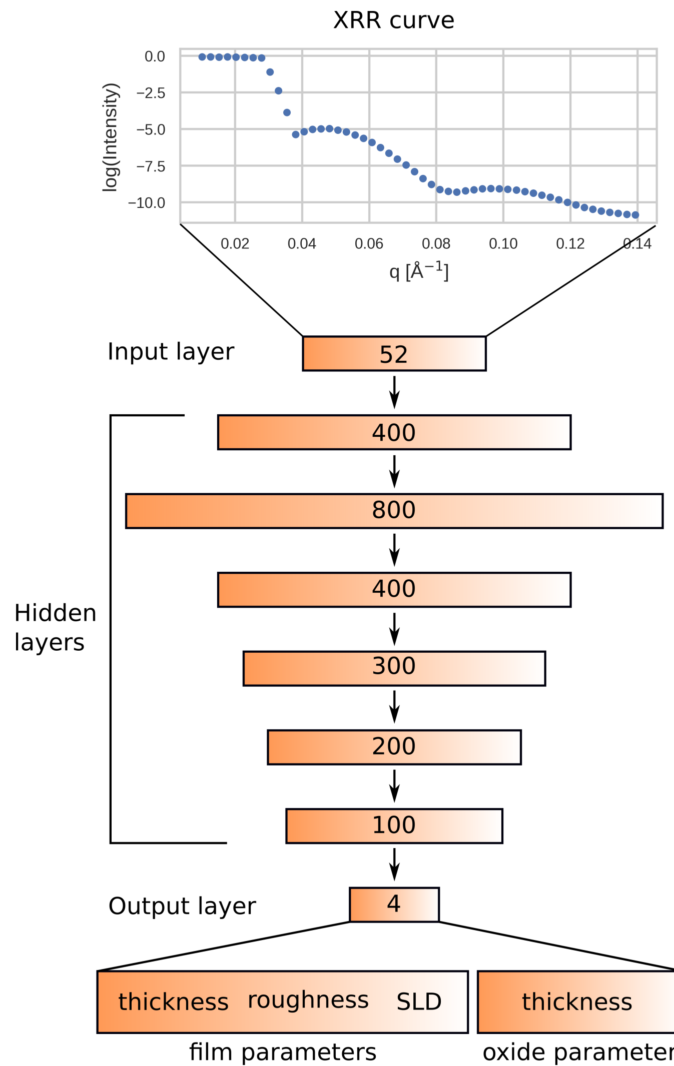

Figure 4.1.: Schematic of the NN architecture used in this work. The input layer consists of 52 reflectivity values at discrete q_z positions. The output layer consists of 4 sample parameters: 3 film parameters (thickness, roughness and SLD) and one substrate parameter (thickness of the native silicon oxide). All layers are fully connected with the next by weights that are randomly initialized and then optimized. Figure adopted from [103].

the minimum and maximum values of the training data so that the loss function is unitless MSE for all thin film parameters.

To keep track of the performance of the model during training and to judge its ability to generalize and yield good results on data that is not included in the training set, its accuracy was evaluated with independently generated validation data. After every epoch, the trained model computes the output of the validation data and the validation error is calculated using the same error function as for the training set. In general, a validation error that is much higher than the training error signifies that the network is overfitting on the training data. On the other hand, if the validation and training errors are very similar, the capacity of the model might be too low to capture important features in the data. The training and validation errors shown in Fig. 4.2 are representative of a typical training session of the NN described above. Even though the training and validation loss could be further reduced by an order of magnitude through longer training, lower accuracies on experimental data were observed when the model was trained for more than 60 epochs. The reason for this is that the performance on simulated data is not a perfect estimator for the performance on experimental data. With increasing training time, the NN becomes more optimized for features that are only present in the simulation, to the detriment of common features shared by both simulation and experiment. This can to some degree be remedied by adding noise to the training data, which is explored in more detail in Chap. 6. Here, the model with the lowest validation loss within 60 epochs was used to achieve a trade-off between a low training loss and overfitting. This method is called early stopping and is a common regularization technique, as described in Sec. 3.6. While overfitting is a general issue of many ML problems, the number of epochs after which it occurs can vary strongly for different types of data and NN architectures. Thus, the optimal number of epochs has to be determined empirically for a given problem and is likely to depend also on the quantity and quality of training data and its similarity to the experimental data.

One of the most important factors that influence the performance of a given NN architecture is the quality and choice of the training data. It is crucial to have a

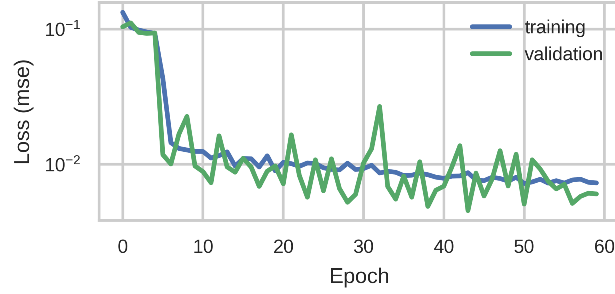

Figure 4.2.: Characteristic training and validation error during training of the NN demonstrated in this study. Since the validation error is very close to the training error, there is likely no overfitting with respect to the validation data. Figure adopted from [103].

sufficiently large and varied data set for the network to find a generalized solution over the entire parameter space. Ideally, a large training dataset of experimental data with precisely labeled thin film parameters should be available for training, validation and testing of the NN model. However, since it is unfeasible to measure and fit the number of reflectivity curves necessary for training, simulated training and validation data was used. In total, 200,000 XRR curves with a 4:1 training/validation split were simulated using an adaptation of the optical matrix method [119, 120], which is a computationally more efficient alternative to the recursive Parratt formalism [71] (see Subsec. 2.2.4 for more details). For this purpose, parts of the Refl1D source code (Copyright 2006—2011, University of Maryland) were used [93]. The sample structure was assumed to consist of three layers: Si, SiO_x and the deposited thin film. The ambient medium was assumed to be vacuum. The interface roughness was assumed to follow the Névot-Croce model as described in Subsec. 2.2.5. The roughness of Si/SiO_x substrates is known to be very low, and thus, a constant roughness for the SiO_x and Si layers of 1 and 2.5 Å, respectively, were assumed. Furthermore, the SLDs of those layers were assumed to be constant with values of 17.8×10^{-6} Å^{-2} and 20.1×10^{-6} Å^{-2}, respectively. The parameters of thickness, SLD and roughness were uniformly distributed within the generated training data. For the deposited film, the ranges of thickness and SLD were 20–300 Å and $1–14\times10^{-6}$ Å^{-2}, respectively. Training data with a film thickness below 20 Å were excluded since, owing

to their ambiguity given the used q_z range, they were the most difficult for the NN, and by removing them the accuracy on the rest of the data could be improved. The range of the film roughness was from 0 Å to half the film thickness, but limited to 60 Å. The thickness of the native oxide layer was assumed to be within the range of 3–30 Å. The reflectivity curves were simulated in a q_z range of 0.01–0.14 Å^{-1} at 52 equally spaced points, which is comparable to the resolution of our experimental data. The small q_z range was chosen to avoid conflicts with Bragg reflections and corresponding Laue oscillations, which are not part of the box model.

For the performance evaluation of the NN, experimentally measured XRR curves of real-time growth of diindenoperylene (DIP), copper(II)-phthalocyanine (CuPc) and α-sexithiophene (6T) on silicon substrates with a native oxide layer were used. Appropriate footprint corrections and normalization were applied to the data before use. The output of the model was judged against a conventional LMS fit that was performed manually on 20 % of the curves. The rest of the film parameters were linearly interpolated within one measurement. The fit was performed with six open parameters: the thickness, roughness and SLD of the deposited film, the thickness and roughness of the oxide layer, and the roughness of the silicon substrate. For CuPc and 6T, a thin void layer with a thickness of 3 Å and a roughness of 1 Å between the substrate and the film was included. This was done because, for some organic thin films, the electron density (and thus SLD) at the interface with the substrate is lower than in the bulk, and including a void layer with a finite roughness improves the fit quality. In these cases, the NN model is intentionally simpler than the manual fit, but since the void layer is thin compared with the deposited film, it is possible to directly compare the film thicknesses obtained from both the NN and the LMS fit. The densities of the silicon and its oxide layer were assumed to be constant across all experiments as described above. In order to make all XRR curves compatible with the same fixed size of the input layer, the reflectivity curves for all experiments were interpolated on a logarithmic scale to the same 52 q_z values without significant change in curve shape.

4.3. Performance comparison to conventional LMS fitting

To evaluate the accuracy of the NN model, its performance was tested on two datasets. The first consisted of 20,000 independently simulated curves with the same parameter range as the training data. The second consisted of the five experimental real-time XRR datasets described before. In the case of the simulated data, the mean average percentage errors of the film thickness, roughness and SLD were 8, 16 and 6 %, respectively. Although already quite good, these metrics reveal that for this NN model, there is still a significant portion of curves with a large error. Furthermore, within the given q_z range, it seems to be intrinsically more difficult to correctly determine the roughness than the other two parameters. Since the synthetic test data was generated using the same process as the training data, we cannot expect better performance on data that was generated using a different process, such as experimental data. Furthermore, since the parameter distribution of the simulated test data is different from that of the experimental test data, the mean errors of the two are not directly comparable. In Chap. 5 it will be shown that the simulated data contains a lot of featureless curves which might inflate the average error somewhat. While a further reduction of the training and validation loss could be achieved in principle, *e.g.* by training for more epochs, it was observed that this generally leads to a decrease of the performance on experimental data. This means that the training loss cannot necessarily be used to estimate how the NN will perform on experimental data, and the training process is ultimately limited by the fact that the simulation does not perfectly describe the experiment.

For the performance evaluation with the experimental data, the film properties determined by the model were compared with a manual LMS fit using a genetic algorithm (GenX [96]). From hereon, this manual fit is considered to be the ground truth. The studied systems were two DIP films, one CuPc film and one 6T film, each grown at 303 K, as well a third DIP film, grown at 403 K. Three out of five of these data sets have already been analyzed and published (DIP 303 K [9], DIP

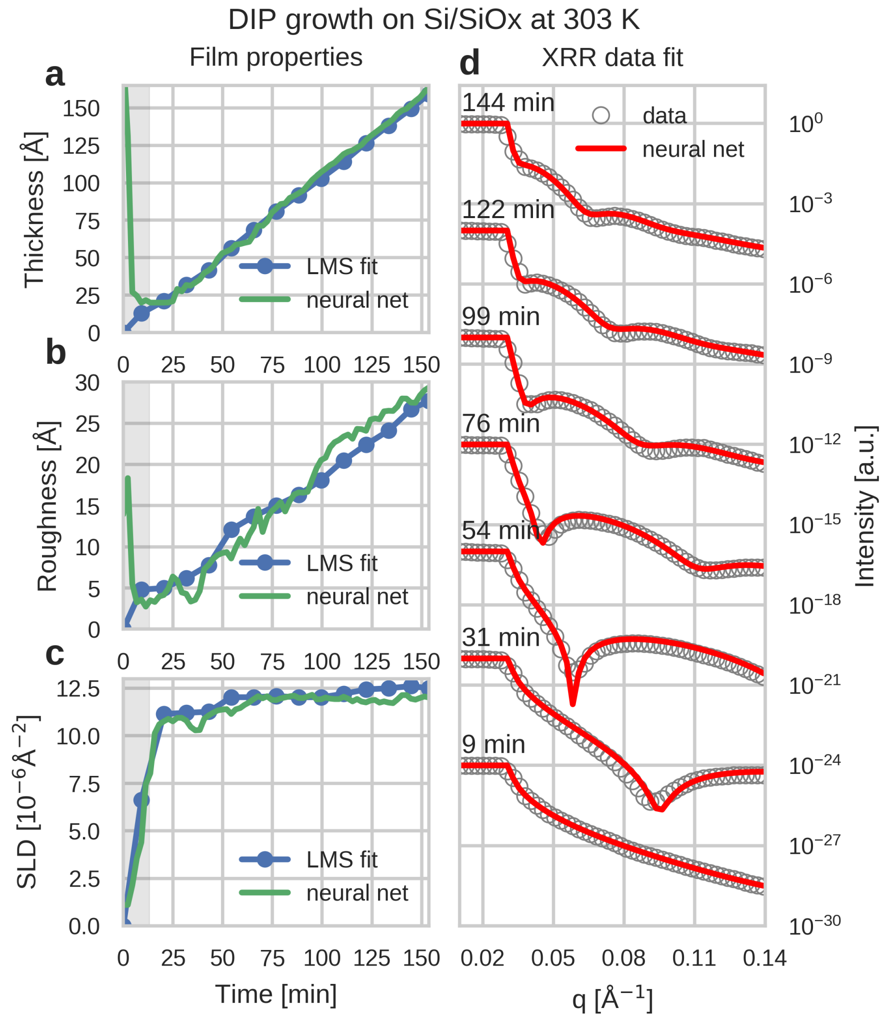

Figure 4.3.: Fitting performance of the NN model on a DIP film grown at 303 K with a deposition rate of 1 Å min^{-1}. (a–c) Comparison of the film parameters predicted by the NN with results from least mean square fitting with human supervision at different times during growth. The shaded area marks films with thicknesses below 20 Å where the network has not been trained and consistently predicts thick films with high roughness. (d) Overlay of the experimental XRR data with data simulated using the parameters predicted by the NN during different times during growth. Figure adopted from [103].

Table 4.1.: Mean absolute percentage error and standard deviation of the predictions on experimental XRR curves with respect to the values obtained *via* a conventional LMS fit with manually set bounds and starting points. Predictions of films with a thickness below the training range of the NN (<20 Å) and high roughness (>30 Å) were excluded. DIP 303 K (1) is shown in Fig. 4.3, all others are shown Appendix A. Table adopted from [103].

	DIP 403 K	DIP 303 K (1)	DIP 303 K (2)	CuPc 303 K	6T 303 K	all
thickness	(17±20) %	(4±4) %	(6±9) %	(16±13) %	(4±3) %	(11±10) %
roughness	(20±14) %	(12±11) %	(15±11) %	(26±18) %	(16±11) %	(18±13) %
SLD	(11±9) %	(3±2) %	(9±8) %	(6±5) %	(10±6) %	(8±6) %

403 K [88] and 6T [86]). For more details on the origins of the data, see Appendix F. A performance comparison for one DIP film grown at 303 K is shown in Fig. 4.3a–c (similar plots for the other 4 datasets can be found in Appendix A). It is immediately apparent that, for most of the series, the determined values are close to the ones obtained *via* the manual LMS fit. It is important to note that this is already a remarkable achievement, since the NN has no information about any temporal correlation between the XRR curves, which is the kind of knowledge a researcher would use when selecting bounds and starting points for an LMS fit. Furthermore, the output for each XRR curve was obtained on average within 77 ms when using just a single curve as input, and within 0.03 ms/curve when using 20,000 curves at once. Compared with a manual fit, this is orders of magnitude faster and can compete with the speed at which modern 2D detectors operate. Also, after training, no additional input was necessary. This makes it in principle possible to determine film properties during measurements in real time without the need for human supervision.

Fig. 4.3d shows an overlay of experimental reflectivity data with simulated curves using the determined film parameters at different times during growth. In general, the curves show a good agreement, which indicates that the determined parameters are close to the real ones. Among all the tested data, the NN performed worst on films with low thickness and thick films with high roughness. This is a general problem that affects all fitting methods, since the corresponding XRR curves do not have pronounced features, such as Kiessig oscillations, and are thus difficult to distinguish

from each other (see Chap. 5). In a conventional LMS fit, this situation can sometimes be remedied by imposing strict boundaries which limit the fit parameters to what is experimentally expected. This method, however, is only indirectly available to the type of FCNNs used here by tuning the range and distribution of the training data. Though it is ultimately desirable to also reliably fit these curves using our NN approach, it is clear that any fitting result based on data with a higher amount of ambiguity will also have a higher level of uncertainty.

Tab. 4.1 shows the mean average percentage error of the NN output when compared with the values determined *via* the manual fit, excluding films with thicknesses below 20 Å. Similar to the results for simulated data, the error is highest for the film roughness and lowest for the SLD. However, on average, the accuracy on experimental data is 2–3 percentage points lower in all three categories. There are likely several reasons for this: firstly, there is already an error attached to the ground truth parameters that were extracted *via* a manual fit. Therefore, the errors of both the LMS and NN fit contribute to the total error. This is not the case for the simulated test data, where the underlying simulation parameters are known. Secondly, and most importantly, the simulated training data differs from the experimentally measured data with regard to finite experimental resolution, noise and other experimental artifacts. Furthermore, it is reasonable to assume that the chosen theoretical model does not perfectly describe the physical reality. As a result, the training data may contain features that the NN relies on for its heuristic prediction but are not present in the experimental data.

Apart from relying on these metrics, the general physical validity of the determined parameters can be confirmed by considering knowledge about the measured system and the experiment. Out of the three parameters, the thickness is the easiest to verify, since in all experiments the films were grown at a constant rate. This expected linear behavior is obtained for all experiments and coincides perfectly with the LMS fit. The obtained thickness values can also be verified to a high degree of certainty by considering the periodicity of the Kiessig fringes. In addition, the obtained SLD shows the qualitatively expected behavior of a continuous increase

during the beginning of the thin-film growth with saturation at a value that is somewhat lower than the SLD of the solid-state crystal. This indicates the transition from a bare substrate to an organic thin film with a constant in-plane-averaged electron density. Among the three determined properties for each experiment, the roughness evolution is arguably the most difficult to judge, since it strongly depends on the specific molecular system and on several important experimental parameters, such as the growth rate and the substrate temperature [5, 74]. In the studied systems, however, the roughness is expected to increase overall for higher film thicknesses, which is a behavior observed for the predictions from all five datasets.

4.4. Conclusions from this chapter

This chapter demonstrated how a straightforward NN model with fully-connected layers can be used to extract the film thickness, roughness and density parameters from real-time reflectivity data of thin films. The NN model was trained on simulated data and tested on simulated and experimental data. Although the accuracy was lower on the experimental data, it still achieved high accuracies with a mean absolute percentage error of 8–18 % with respect to the result determined *via* a manual fit. Importantly, among the three parameters, the film roughness was the most difficult to determine for the model in both the synthetic and the experimental data. While the accuracy on synthetic data could in theory be increased by training the model for longer, this did not translate to improved accuracy for the experimental predictions. Thus, Chap. 5 and Chap. 6 focus in part on generating better training data that more accurately represents the experiment and allows training without overfitting to features that are only present in the simulation. Nevertheless, it is important to understand and improve the results on simulated data, since they essentially represent the upper limit of what can be expected in terms of accuracy.

While the NN model shown in this chapter only represents a first proof of concept, the demonstrated performance would already be sufficient for a preliminary screening of reflectivity data before further analysis. However, the NN predictions

sometimes contained significant outliers due to the sensitivity to experimental artifacts. Successful strategies to remedy this are discussed in Chap. 6. Furthermore, the extremely fast computation times of 0.03–77 ms per curve and the fact that, after training, no further user input is needed mean that this approach is perfectly suited for *in situ* applications, such as monitoring film parameters during real-time measurements, as is demonstrated in Chap. 6. In addition, this approach is easily transferable to neutron reflectivity data, however, some important differences, such as the different cross sections of neutrons (coherent as well as incoherent) need to be taken into account, which is discussed in Chap. 5.

5. Neural network performance in the light of challenging cases[‡]

5.1. Introduction

The previous chapter introduced the concept of using FCNNs for the fast analysis of XRR data. The goals of this chapter are to extend the discussed concepts to NR data by addressing its differences compared to XRR data and to identify pathological experimental conditions that are detrimental to the application of ML techniques. These conditions are discussed in the light of three types of challenges with regard to the measured data that are especially (but not exclusively) relevant in the context of NR: 1) Reflectivity curves without strong features that have a low information content, 2) curves without a TRE, and 3) data with significant noise or background.

The results of this chapter show that applying different types of random noise and background intensities to the training data results in a NN model that is more robust towards these perturbations when determining thin film properties, but still struggles with certain particularly difficult edge cases. The previous chapter assumed a sample with 4 undetermined parameters, *i.e.* a completely undetermined thin film on top of a fully-characterized Si substrate with an SiO_x layer of undetermined thickness. Furthermore, all SLDs in the system were confined to positive numbers. As mentioned in Sec. 2.1, this is a reasonable assumption for X-rays, however, scattering

[‡]This chapter is largely based on the publication Greco *et al.* 2021 [107]. The design of the neural network and the training data were developed by A. Greco. M. Skoda helped develop the training data modifications from a neutron scattering perspective. All data analysis was performed by A. Greco. A. Gerlach provided computational support and F. Schreiber supervised the project.

Table 5.1.: Parameter ranges for simulated training, validation and test data. Table adopted from [107].

	thickness [Å]	roughness [Å]	SLD [10^{-6}Å^{-2}]
ambient	—	—	0
layer	20–300	0–60	-8–16
substrate	—	0–10	-8–16

with neutrons can also produce negative SLDs. Thus, this chapter will investigate the performance of the NN assuming an entirely undetermined film and substrate (in total 5 open sample parameters), including negative SLDs.

5.2. Training data simulation

5.2.1. General simulation parameters

The training and validation data used in this chapter were simulated using a model of a single-layer on a substrate with a total of 5 open parameters: substrate roughness, substrate SLD, layer thickness, layer roughness and layer SLD. The ambient SLD was assumed to be 0 (*e.g.* air/vaccum). For the training and validation of the NN, $3{\times}10^6$ and $2{\times}10^4$ parameter sets were generated, respectively. The values of each set were generated within the ranges given in Tab. 5.1 with a "bolstered" sampling density towards the limits of each parameter range, as shown in Fig. 5.1. This was done to make the local density of sampled values near the limits more similar to the density towards the center of the distribution. The number of generated parameter sets was chosen to cover as much of the parameter space as possible while still maintaining technical feasibility in terms of training time and occupied memory. The range of possible SLD values for the substrate and layer was specifically designed to encompass a large spectrum of negative and positive SLDs of the most common elements [158]. This was done to investigate how different combinations of negative and positive SLDs affect the prediction performance of the NN.

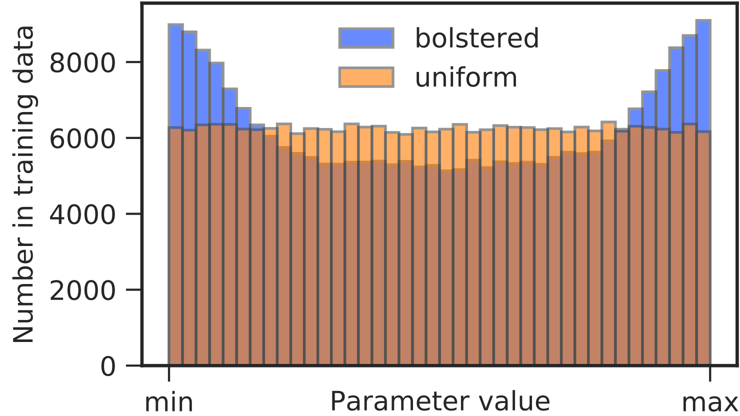

Figure 5.1.: Comparison of a uniform distribution with a "bolstered" uniform distribution. In the latter, 15 % of the total number of samples are sampled from Gaussian distributions at the limits of the parameter range. This was done to reduce the effect that data points towards the center of a distribution are often better fitted than the ones at the limits.

From the generated parameter sets, reflectivity curves were simulated using the same implementation of the Matrix method as in Chap. 4. The simulated q_z range was restricted to a range of 0.01–0.3 Å^{-1} in order to avoid q_z ranges where Bragg reflections and Laue oscillations might appear in real measurements, since they are not described by the slab model. Thus, we can be sure that the NN predictions are only based on Kiessig fringes and other features related to the layer structure which would be present in an experimentally measured curve. Within this q_z range, the reflected intensity values $\boldsymbol{R} \in \mathbb{R}^{100}$ were simulated at 100 equally-spaced discrete points $\boldsymbol{q} \in \mathbb{R}^{100}$. This number was chosen to be comparable with common point densities of experiments, and was increased from the previous 52 points to avoid the need to downsample data that was measured with higher point density, as is the case for the real-time XRR measurements discussed in Chap. 6.

In contrast to the training procedure in the last chapter, different types of noise and background intensities were added to each curve when a minibatch was drawn from the training set. This means that every time the NN encounters one of the training curves, the curve is modified with different noise and background. This step is crucial to avoid overfitting and to prevent the network from learning heuristics

that do not work on imperfect data by forcing it to learn how to denoise the input data. The perturbations added to the data were Poisson noise, statistical noise, curve scaling and constant background. Each type of curve modification is described in detail in the following section.

5.2.2. Training data modifications

5.2.2.1. Poisson noise from counting statistics

Statistical noise in scattering data results from the counting statistics of scattered particles arriving at the detector and it is dependent on the expected counting rate I, *i.e.* the recorded intensity. Since this noise generally follows a Poisson distribution, the noise of a simulated reflectivity curve can be calculated by replacing each intensity value R_i with a random value picked from the distribution

$$p_s(x) = \frac{p_\mathrm{p}(x; sR_i)}{s} \tag{5.1}$$

where $s = I_0$ is the theoretical maximum number of counts at total reflection (for a monochromatic experiment) and p_p is the Poisson distribution

$$p_\mathrm{p}(x; I) = \frac{N^x e^I}{x!}. \tag{5.2}$$

Since the simulated intensities $\boldsymbol{R}$ only range from 0 to 1, they must be scaled to values which could occur in an experiment $\boldsymbol{I} = s\boldsymbol{R}$ before calculating the noise. In this study, for every curve, a scaling factor (corresponding to the flux) was randomly chosen on a logarithmic scale within $s = [10^6, 10^8]$ to represent different experimental conditions. For the upper limit of $s = 10^8$, there is no noticeable noise in the chosen q_z range anymore. An example of a curve with a noise level of $s = 10^6$ is shown in Fig. 5.2a.

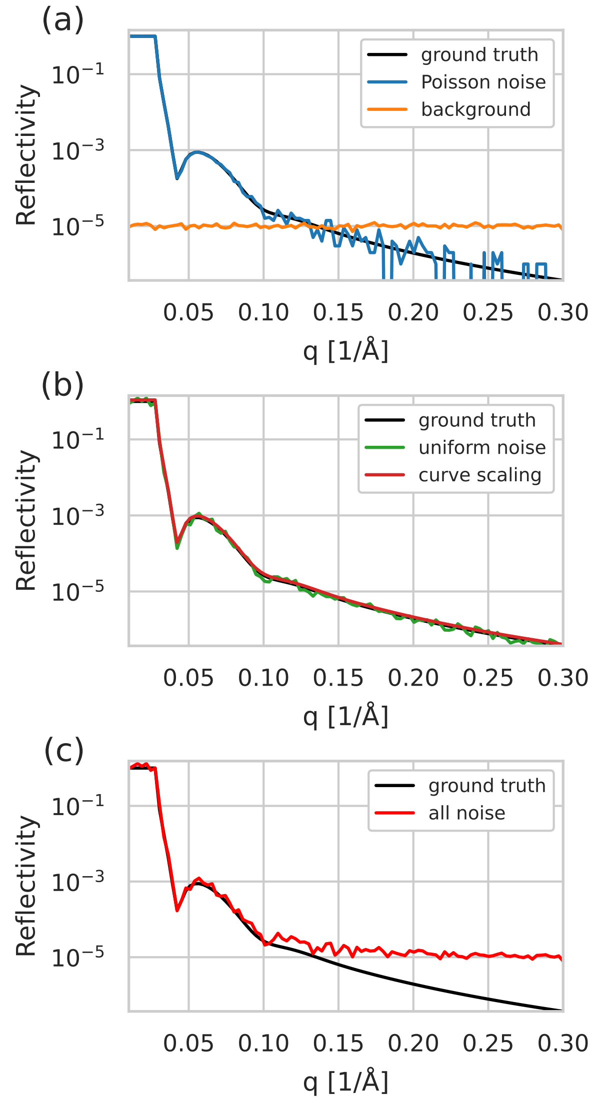

Figure 5.2.: Example of a simulated reflectivity curve with different noise and background applied to it. a) The ground truth curve and the same curve with Poisson noise with a level $s = 10^6$ as well as a background of $b = 10^{-5}$. b) The same curve with curve scaling and uniform noise applied to it, respectively. c) Comparison of the ground truth with a curve with all 4 modifications applied to it. Figure adopted from [107].

5.2.2.2. Uniform noise

Since the statistical noise mainly affects low-intensity regions in a reflectivity curve, the first half of the curve, *i.e.* low-q_z features and in particular the TRE, remains mostly unaffected by it. Despite this, experimental data may contain noise or other small deviations in this region. For example, in time-of-flight (TOF) neutron scattering experiments, the intensity given by the normalized reflectivity curve is not necessarily proportional to the counts on the detector, since the incoming beam generally has an energy spectrum with a non-uniform distribution (*e.g.* Maxwell-Boltzmann). The final intensity of the curve is then obtained by normalizing the number of counts in each channel (*i.e.* q_z value) with the corresponding incoming intensity of that energy. This can effectively lead to worse counting statistics in regions with seemingly higher intensity, such as near the TRE, compared to lower intensity regions. Furthermore, the beam shape and random errors in the measurement angle typically lead to non-negligible deviations of the intensity. This is particularly pronounced near the TRE, where slight errors in the angle might translate to large errors in intensity.

To make the NN robust against these types of errors, a random scaling factor α_i is multiplied to each intensity value R_i of each input curve $\boldsymbol{R}$, so that the new intensity is given by

$$R_i^* = \alpha_i R_i \tag{5.3}$$

where each element of $\boldsymbol{\alpha} \in \mathbb{R}^{100}$ is uniformly sampled from the range $[0.7, 1.3]$. An example of a curve with uniform noise applied to it is shown in Fig. 5.2b.

5.2.2.3. Curve scaling

In order to analyze reflectivity data, the measured intensity is typically normalized to the intensity at total reflection. For monochromatic experiments, this step is preceded by an angular dependent footprint correction. For polychromatic experiments (*e.g.* TOF), the normalization must additionally take into account the above-mentioned energy spectrum, usually obtained *via* measuring the direct beam.

Both of these corrections produce an error on the normalization procedure (which itself has a finite accuracy) and may result in distortions of the data. This effect is further exacerbated if there is no TRE, since the intensity at total reflection is not directly available, *i.e.* the naturally given absolute scale of the TRE is missing.

To make the NN robust against these slight distortions, during training, every input curve $\boldsymbol{R}$ is multiplied by a random scaling factor β, so that the new curve $\boldsymbol{R}^*$ is given by

$$\boldsymbol{R}^* = \beta \boldsymbol{R} \tag{5.4}$$

where β is uniformly sampled from the range $[0.9, 1.1]$. An example of a curve with random scaling applied to it is shown in Fig. 5.2b.

5.2.2.4. Residual background

Both X-ray and neutron scattering experiments contain background intensity stemming from various sources, such as background radiation or detector noise. Within the q_z range discussed in this study (max. $0.3\,\text{Å}^{-1}$), for XRR these effects are usually negligible compared to the measured intensity of the reflected beam.

In addition, neutron reflectivity data usually contains background resulting from incoherent scattering [3]. In practice, most of this background is already removed during the data reduction step, *e.g. via* calibration with a pure transmission measurement. During data analysis, the residual background is then routinely approximated by a constant value, although more complex models exist [159].

To account for this, the residual background in the data was approximated by a q_z-independent array of values $\boldsymbol{b} \in \mathbb{R}^{100}$ with normally distributed fluctuations with mean b and standard deviation $\sigma_b = 0.1b$. The fluctuations were added to account for random deviations in the background. Thus, for a given curve $\boldsymbol{R}$, the background b_i added to each intensity value R_i was randomly picked from the normal distribution

$$p_b(x; b, \sigma_b) = \frac{1}{\sqrt{2\pi\sigma_b^2}} \exp\left(-\frac{(x-b)^2}{2\sigma_b^2}\right). \tag{5.5}$$

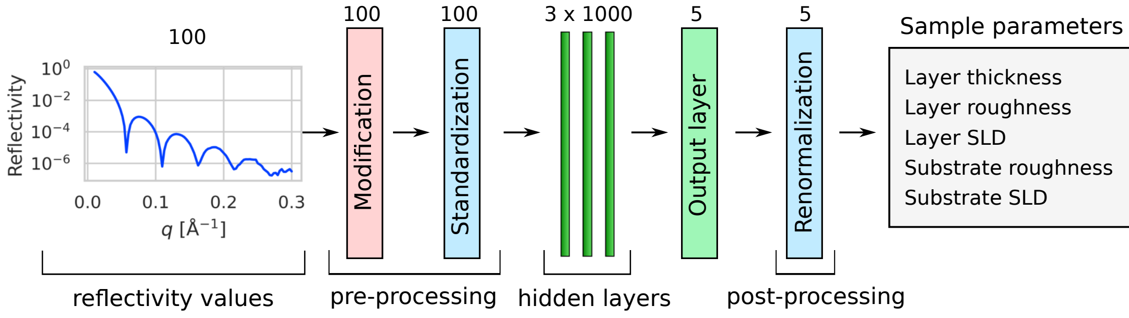

Figure 5.3.: Architecture of the fully-connected NN used in this chapter. The addition of noise and background to the input data during the modification step was only performed during training. Figure adopted from [107].

In this work, the background level of each curve was randomly chosen on a logarithmic scale within $b = [10^{-7}, 10^{-4}]$. An example of an added background with $b = 10^{-5}$ is shown in Fig. 5.2a.

5.3. Neural network design and training

The NN used in this chapter is a fully-connected model with 100 input neurons, 3 hidden layers with 1000 neurons each and 5 output neurons as shown in Fig. 5.3. Compared to Chap. 4, the number of layers is reduced and their width is increased to simplify the NN architecture. The number of trainable parameters is roughly twice as large to account for the more complex task. The input corresponds to the reflectivity values $\boldsymbol{R}$ at 100 discrete points in q_z space, as described in the previous section. The output corresponds to the 5 open thin film parameters $\boldsymbol{y} \in \mathbb{R}^5$ as shown in Tab. 5.1. As an activation function, ReLU was chosen for all hidden layers. During training, whenever a mini batch of 512 curves is drawn from the training set, curve modifications are applied as described in Sec. 5.2. Then, each input R_i is independently standardized by subtracting the mean $\bar{R}_i$ and dividing by the standard deviation $\tilde{R}_i$ across the entire randomly modified training set (determined before the training). The standardized input is thus given by

$$\hat{R}_i = \frac{R_i - \bar{R}_i}{\tilde{R}_i} \tag{5.6}$$

with the mean

$$\bar{R}_i = \frac{1}{N} \sum_{n=1}^{N} R_{n,i} \tag{5.7}$$

and the standard deviation

$$\tilde{R}_i = \sqrt{\frac{1}{N} \sum_{n=1}^{N} (R_{n,i} - \bar{R}_i)^2} \tag{5.8}$$

where $\boldsymbol{R}_n$ is a curve from the training set of size $N = 3 \times 10^6$.

The output values y_j were normalized by the greater absolute value of either the minimum or maximum of their respective ranges given in Tab. 5.1. This effectively confined all output values to a range from -1 to 1.

The ADAM algorithm [154] was used as an optimizer with the recommended default parameters and a starting learning rate of 10^{-3}. Furthermore, the learning rate was reduced by half each time the validation loss stagnated for 10 consecutive epochs in order to avoid skipping over narrow minima in the loss function space. The MSE of the normalized outputs was used as the loss function. The NN was trained on a GeForce RTX 2080 Ti GPU and an Intel® Core™ i5-9600K CPU for 175 epochs with a training time of about 6.5 min per epoch, amounting to a total training time of about 19 h.

To test whether the exclusion of featureless curves during training could boost the overall performance of the model, two identical NNs as shown in Fig. 5.3 were trained and compared. Model 2 was trained with the entire training set as described in Sec. 5.2, while model 1 was trained with the exclusion of two subsets. The first excluded subset contained all curves with a thickness of less than 20 Å. As was shown in Chap. 4, films with low thicknesses tend to perform poorly due to the chosen q_z range, since the minimum resolvable film thickness is essentially limited to $2\pi/q_{\max} = 20$ Å. The second subset contained all curves where the SLD contrasts between the layer and the substrate or the layer and the ambient SLD was less than $1{\times}10^{-6}$ Å^{-2}. In these cases, the refractive index of the film layer is almost the same as at least one of its adjacent media, leading to very few features. Fig. 5.4

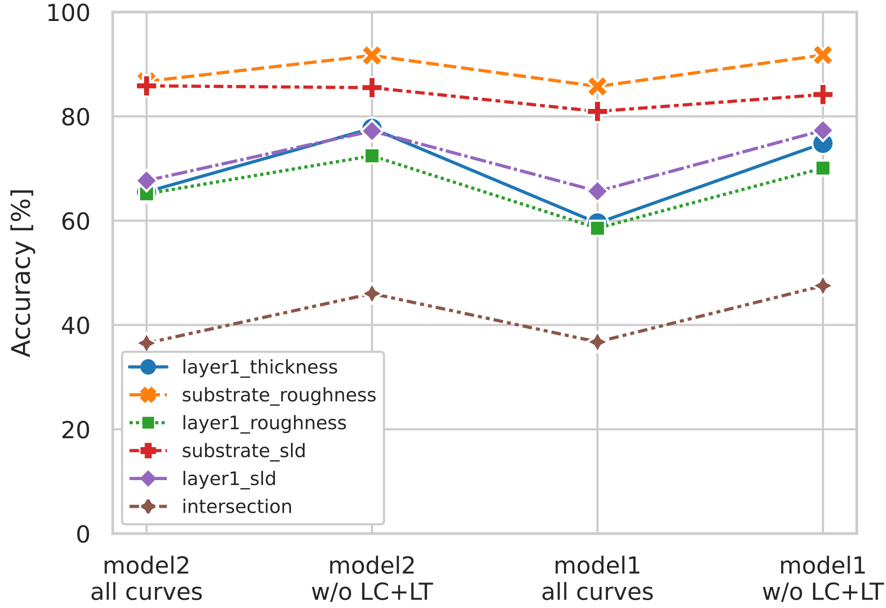

Figure 5.4.: Testing accuracy of two models trained with (model 2) and without (model 1) low-contrast (LC) and low-thickness (LT) cases. Model 1 was used to produce the results discussed in this chapter. Figure adopted from [107].

shows the prediction accuracy as defined in Subsec. 5.4.1 for both models and for each parameter. Both models performed better when low-contrast (LC) and low-thickness (LT) cases were also removed from the test set, however, model 1 showed a higher accuracy on the entire test set. Fig. 5.5 also shows that the distribution of the absolute error for model 1 is narrower for all parameters, *i.e.* closer to 0, especially considering the two roughness parameters. Thus, model 1 was chosen for all further analysis in this chapter.

The training and validation loss curves of model 1 are shown in Fig. 5.6. Overall, the training and validation loss are almost identical to each other, with the validation loss only being slightly higher after 70 epochs. The reason why this difference is so small is that the training data is randomly modified with noise and background during each epoch. This means that the network sees “fresh” curves every time, enabling it to generalize better. Since epoch 147 showed the lowest validation loss, the network parameters saved at this point were chosen for all further testing.

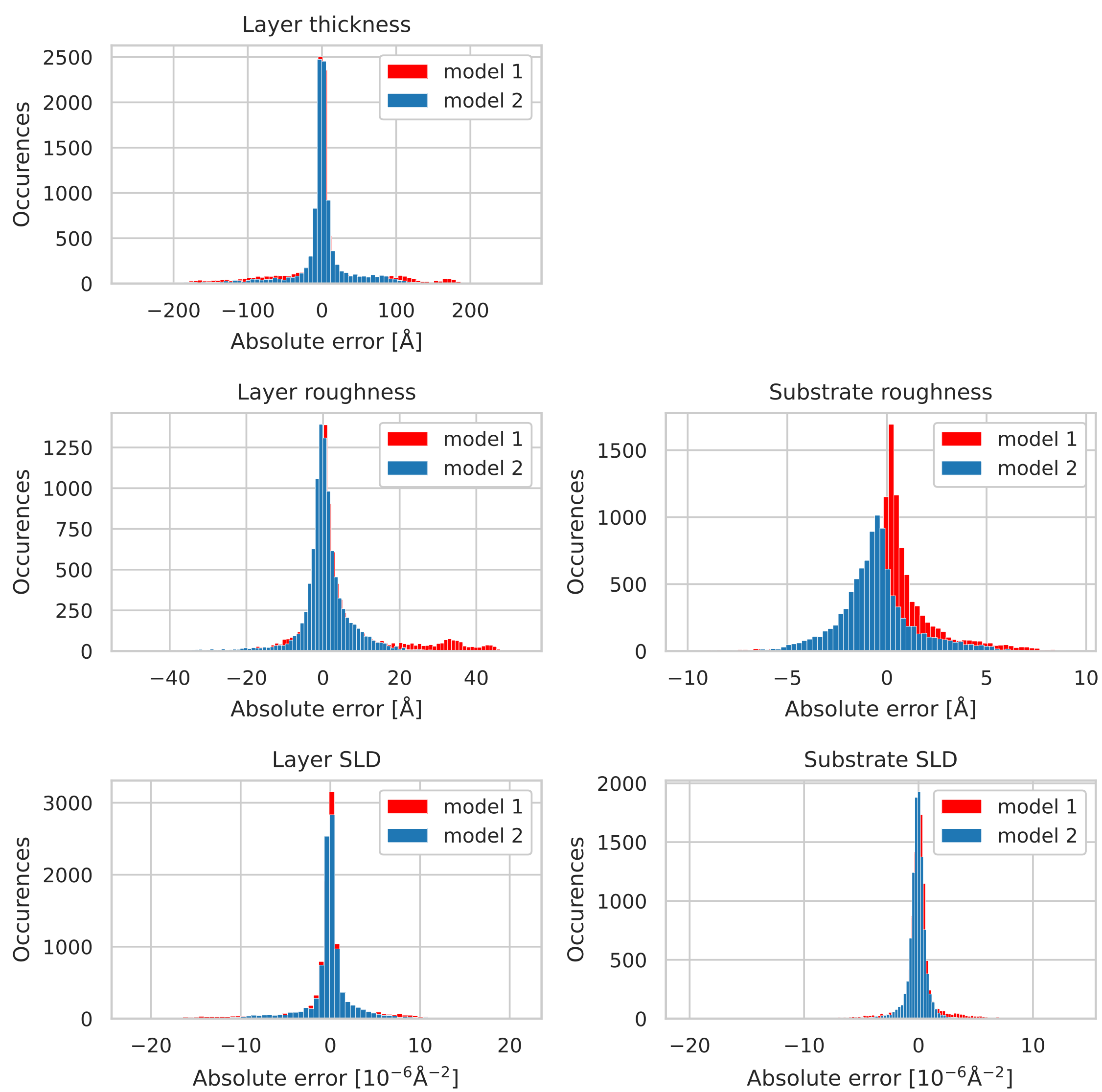

Figure 5.5.: Absolute testing error of two models trained with (model 2) and without (model 1) low-contrast (LC) and low-thickness (LT) cases. Model 1 was used to produce the results discussed in this chapter. Figure adopted from [107].

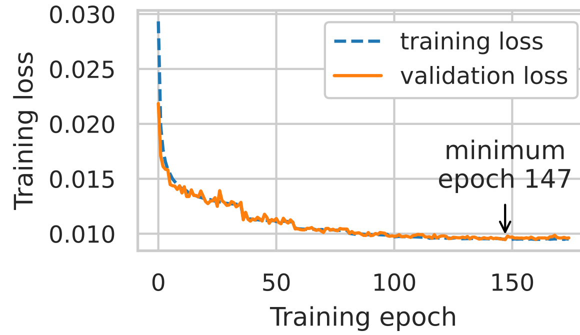

Figure 5.6.: Training and validation loss during training. The discrete downward steps of the loss coincide with a reduction of the learning rate by a factor of 0.5. The training was stopped after 175 epochs because the validation error did not decrease further. The lowest validation error was observed at epoch 147. Figure adopted from [107].

5.4. Results and discussion

5.4.1. Definition of the prediction accuracy

The performance of the trained NN was tested using 10,000 simulated reflectivity curves that were generated within the same ranges as the training data, excluding cases with low SLD contrast. The prediction error as a function of the ground truth (GT) for each of the 5 thin film parameters is shown in Fig. 5.7. In order to better quantify the performance of the NN, all predictions were separated into two classes: those close to the GT were classified as "correct", whereas all others were classified as "incorrect". The condition for which a prediction is considered "correct" was defined for each parameter separately. For the thickness and roughnesses, all errors smaller than 10 % of the GT or smaller than 3 Å were considered "correct". The absolute condition was added to avoid divergence for small GT values. For the SLDs, all errors with an absolute value smaller than 1×10^{-6} Å^{-2} are classified as "correct". In Fig. 5.7, all "correct" predictions are colored green, whereas all "incorrect" predictions are colored red. In the following analysis, the percentage of correctly classified predictions out of all predictions will be used to discuss the

performance of the NN model under different circumstances. From here on, we will call this metric the prediction accuracy.

The prediction accuracy for each parameter of the 10,000 simulated curves is shown in Fig. 5.8. The corresponding distribution of absolute errors can be seen in Fig. 5.5 (model 1). For the entire test set, *i.e.* the full parameter space, the accuracy of the individual parameters lies between 71–92 %, whereas the accuracy of all 5 parameters being correct at the same time is 48 %. The latter is arguably the most important metric for the NN performance, since it represents the average likelihood that the NN predicts all 5 parameters of an unknown curve correctly. Thus, in the following, this metric will be the focus of the discussion and it will be compared for different subsets of reflectivity curves within the test set.

5.4.2. Comparison of the prediction accuracy with the curve mean squared error

For conventional curve fitting tools, often the MSE (or an equivalent metric, *e.g.* χ^2) between the data and the fitted curve is used to judge the goodness of the fit. This is then used to find a solution to the inverse problem of inferring the correct thin film parameters. However, in some cases, due to a very flat MSE surface with respect to the fitted parameters, there exist many, equally well-fitting curves with different (and potentially wrong) fit parameters. In these cases, reliably extracting the correct thin film parameters is difficult or even impossible without any prior physical knowledge.

To identify especially difficult cases for MSE fitting heuristically, we can calculate the logarithmic MSE between the GT curve $\boldsymbol{R}^{\mathrm{GT}}$ and the curve simulated from the NN prediction $\boldsymbol{R}^{*}$ as

$$E_{\mathrm{MSE}} = \frac{1}{d} \sum_{i}^{d} \left(\log \left(R_i^{\mathrm{GT}} \right) - \log \left(R_i^{*} \right) \right)^2 , \tag{5.9}$$

where d is the number of measured reflectivity values. In this case, an MSE of 0.1 or lower can be considered an adequate fit, whereas an MSE of 0.01 or lower can

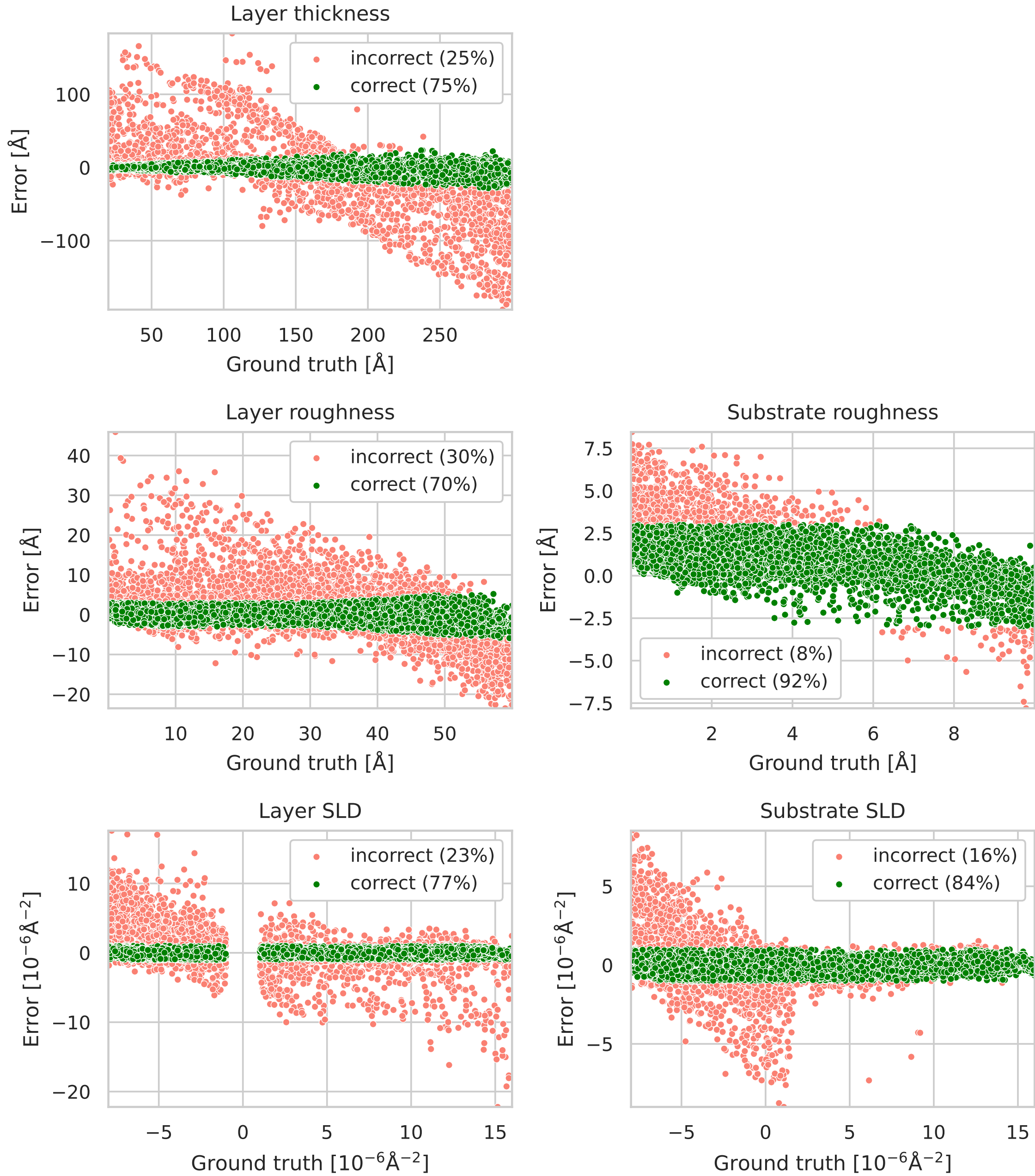

Figure 5.7.: Prediction error of 10,000 simulated reflectivity curves as a function of the GT for each of the 5 thin film parameters. For thicknesses and roughnesses, errors below 3 Å and 10 % are considered correct. For SLDs, absolute errors below $1{\times}10^{-6}$ Å^{-2} are considered correct. Correct and incorrect predictions are colored green and red, respectively. The gap around 0 for the layer SLD results from the exclusion of low contrasts between the layer and the ambient medium as described in Sec. 5.3. Figure adopted from [107].

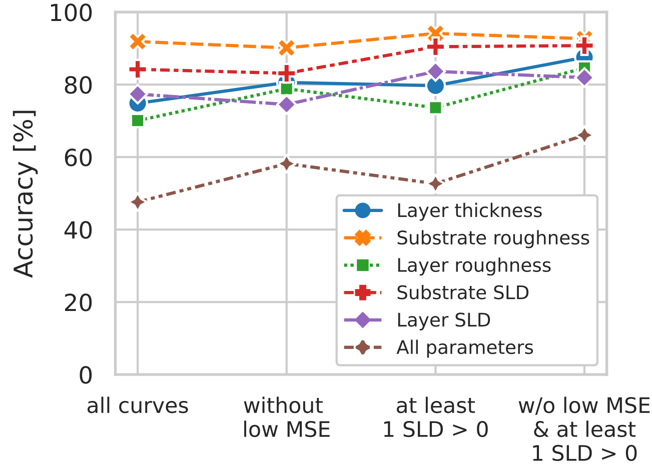

Figure 5.8.: Prediction accuracy of the NN for the entire test set as well as for 3 smaller subsets where particularly difficult edge cases have been removed: 1) Removed curves that are incorrect, but still have a very low MSE, 2) Removed curves where both layer and substrate have a negative SLD (*i.e.* no TRE), 3) Removed both cases of 1) and 2). By removing difficult edge cases, the accuracy of all parameters being correct is increased from 48 % to 66 %. Figure adopted from [107].

be considered a near perfect fit. In terms of the test set, 73 % of the predicted curves have an MSE of 0.1 or lower and 46 % of 0.01 or lower. When comparing these predictions to the GT, only 60 % of the curves with an MSE < 0.01 were actually predicted correctly according to our aforementioned criteria. Thus, about a fourth of the predictions consist of wrong, but extremely well-fitting NN predictions that look convincing to a visual inspection. Interestingly, most of these curves are also completely monotonic, which means that these are curves without strong Kiessig oscillations. Strong Kiessig oscillations would ensure that the MSE surface is not flat, therefore reducing the solution space drastically. Prominent reasons for a lack of strong features are a low SLD contrast, very low film thickness compared to the q_z range or a high roughness compared to the thickness. Other studies have shown that even in simple systems, certain combinations of SLDs and thicknesses can mathematically lead to the same reflectivity curve due to the phase problem [92]. However, there also exist other conditions that are harder to formalize and

others have attempted to quantify the information content in reflectivity data using Bayesian methods and information theory [160–162]. Moreover, when taking into account experimental boundary conditions, such as a limited q_z range and experimental errors, even mathematically different but similar solutions can be ambiguous.

Fig. 5.9a shows a curve from the test set with the corresponding prediction from the NN with the SLD profiles shown in Fig. 5.9b. Although the MSE between the two curves has a very low value of 0.004, the SLD profiles deviate significantly from each other. While the substrate parameters are predicted almost perfectly, the film thickness is off by 125 Å (42 %). The reason why both SLD profiles are reasonable solutions is that the product of the film thickness and film SLD is the same for both the GT and the prediction, *i.e.* $175\,\text{Å} \times 3 \times 10^{-6}\,\text{Å}^{-2} = 300\,\text{Å} \times 1.75 \times 10^{-6}\,\text{Å}^{-2}$. This means that both films have effectively the same optical density, leading to the same phase difference of the wave traveling through the film. This can also be confirmed by looking at the description of the reflected intensity discussed in Sec. 2.1. The example of a simplified box model in Eq. 2.51 shows that the reflected intensity strongly depends on the product of film thickness and film SLD. Since there are no visible Kiessig oscillations despite the high thickness of the film, it is likely that many parameter combinations could produce a good fit.

Consequently, all curves that have a low MSE but are classified as incorrect most likely fall into one of two categories: 1) The fit is actually close to the solution, but falls just outside our accuracy margin. In these cases, the solution is likely already near the MSE minimum and can be reached quickly *via* a simple gradient descent refinement using the prediction as starting values. This approach will be introduced in Chap. 6. 2) The MSE surface is very flat with regard to one or more parameters, leading to multiple solutions with a similar MSE even for large deviations from the GT. In these cases, the problem of fitting the data stems from the ambiguity of the data itself, and hence it cannot reasonably be expected that the NN (or another algorithm) can reliably find the true solution. Therefore, we chose to omit curves from both categories for all following accuracy calculations because they are either probably close enough for refinement or not expected to be feasibly solvable without

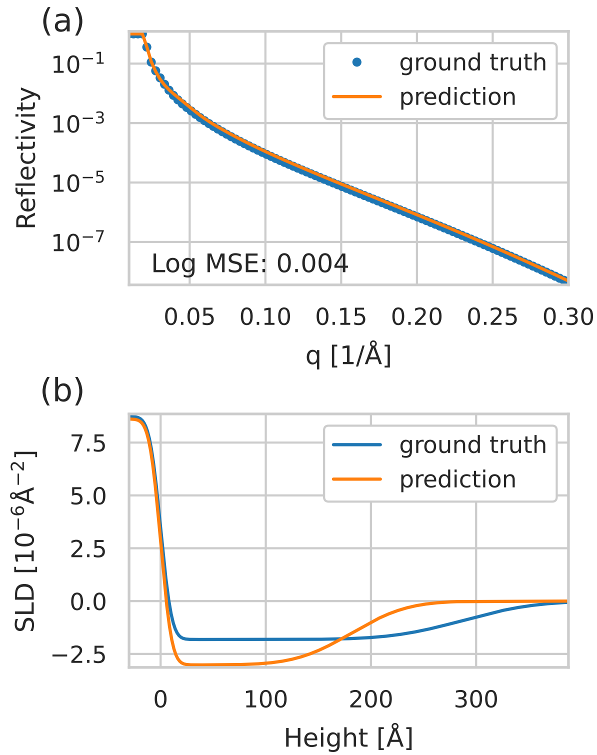

Figure 5.9.: a) Example of a reflectivity curve and its corresponding NN prediction where the MSE is very low although it is classified as incorrect. b) SLD profiles of the GT and the prediction. Although the reflectivity curves fit perfectly, the SLD profiles differ significantly in terms of the film thickness. The reason for this is that the product of film SLD and film thickness is the same for the GT and the prediction, *i.e.* $175\,\text{Å} \times 3 \times 10^{-6}\,\text{Å}^{-2} = 300\,\text{Å} \times 1.75 \times 10^{-6}\,\text{Å}^{-2}$. Figure adopted from [107].

prior knowledge. After removing those curves, the total accuracy on the test set (all 5 parameters correct) rose from 44 % to 58 %.

5.4.3. Influence of different SLD combinations on the prediction accuracy

An essential part of reflectivity data is the presence of a TRE or a lack thereof. The TRE is located at the critical angle which is related to the square root of the SLD contrast $\Delta\rho$ as stated by Eq. 2.37. For thicker layers, q_c is given by the SLD contrast between the ambient medium and the layer. For thinner layers, q_c is mainly affected by the contrast between the ambient medium and the substrate. Thus, the TRE contains direct information about the absolute SLDs in the system. Furthermore, it gives a clear way of calibrating measured data, since it is expected that below the TRE almost 100 % of intensity is reflected (not accounting for absorption).

To understand the effect of the TRE on the NN performance, the testing set was separated into four different categories: 1) Both the substrate and layer SLDs are positive, 2) only the substrate SLD is negative, 3) only the layer SLD is negative, and 4) both SLDs are negative. In cases 1) and 3), a TRE is expected to be present in the data since q_c is positive for at least one SLD. Conversely, in the last case there can be no TRE since q_c is always negative. In case 2), the TRE depends on the layer SLD, but typically a sharp TRE only forms for higher thicknesses.

The prediction accuracy on the test set for each of these four cases is shown in Fig. 5.10. It is apparent that the accuracy drops drastically when both the layer and the substrate SLDs are below 0, while for the other three cases it stays in a similar range. When both SLDs are negative, the percentage of predictions where all parameters are correct drops as low as 3 %. However, in contrast, for the rest of the cases the accuracy is between 62 % and 74 %, which is above the average of 58 % described in the last section. This shows that the NN seems to struggle specifically with cases where there is guaranteed to be no TRE in the data. This hypothesis is supported by the fact that in Fig. 5.7, the error of the substrate SLD drastically

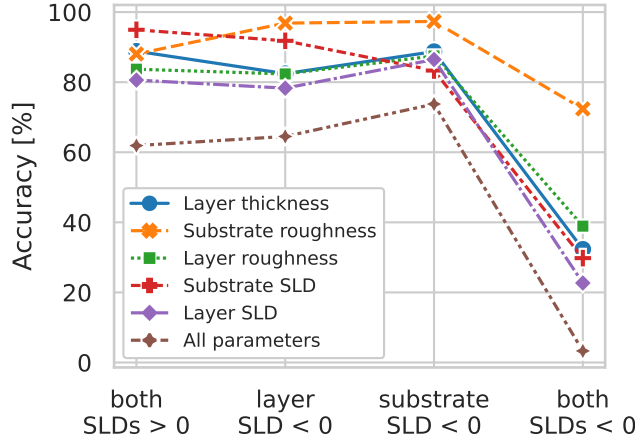

Figure 5.10.: Prediction accuracy for each of the 5 parameters and all 5 parameters together for each of the 4 investigated SLD combinations. The accuracy of all parameters drops significantly when both the layer and the substrate SLD are below 0. Figure adopted from [107].

increases for values smaller than $2\times10^{-6}\,\text{Å}^{-2}$, which corresponds exactly to a TRE at $q_c = 0.01\,\text{Å}^{-1}$ which in turn is the lowest q_z value used in this study. Thus, the performance seems to be negatively impacted as soon as the position of the TRE moves below our detectable q_z range.

From this, we can conclude that during the training process, the NN model learned to extract crucial information from the TRE, so the prediction performance is adversely affected if this information is not available. This is not necessarily unexpected, since the TRE is also an important feature for conventional data analysis. Moreover, it is important to note that not only the prediction accuracy of the SLD itself is adversely affected, but also all other parameters as well.

Of course, it would be desirable to increase the prediction accuracy also for curves without a TRE. While ML models, such as NNs, can potentially extract information and make inferences from measured data more efficiently than conventional analysis methods, it is important to bear in mind that these methods are not able to restore missing information. Thus, data with less information encoded in it will always result in lower prediction accuracies. In order to extract the maximum information possible from difficult measurements, in might be useful to train models that are

specialized towards specific, difficult edge cases. For the purpose of discussing the influence of different noise sources on the prediction accuracy in the following section, the low-performing curves where both SLDs < 0 were removed from the test set.

5.4.4. Influence of noise and background on the prediction accuracy

Every reflectivity measurement contains a number of imperfections such as statistical noise, background, the angular and energy resolution, the beam profile, the slit settings, the beam divergence and the beam footprint. Thus, training ML models on simulated data without any of these imperfections is likely going to lead to overfitting to features that might be obfuscated in real data. Out of these imperfections, this section focuses on the four different sources of noise and background described in Sec. 5.2 and discusses their effect on the prediction accuracy. The data modifications contained 5 different background levels $b = \{0, 10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}\}$, 5 different Poisson noise levels (equivalent to incident intensity) $s = \{\text{off}, 10^6, 5{\times}10^6, 10^7, 5{\times}10^7\}$, 2 uniform noise levels (on/off) and 2 scaling levels (on/off), resulting in a total of 100 different combinations. Each type of modification is described in Sec. 5.2.

For each of the combinations, modified variants of the original 10,000 test curves were created and the subsets of difficult edge cases described in the previous sections were removed. For the unmodified cases, the percentage of correctly predicted curves reached 66 %, which represents the maximum accuracy achieved in this chapter. The dependence of the prediction accuracy on each of the modifications is summarized in detail in Fig. 5.11 (similar plots for the individual parameters can be found in Appendix B). In the top plot of the figure, each point refers to the prediction accuracy of a subset of test curves with a specific noise and background combination. The four different colors/symbols distinguish between the four combinations of uniform noise and curve scaling being turned on or off, respectively. The horizontal axis distinguishes between the 25 combinations of Poisson noise and background levels. The combination of each noise level and background level can be read from the bottom

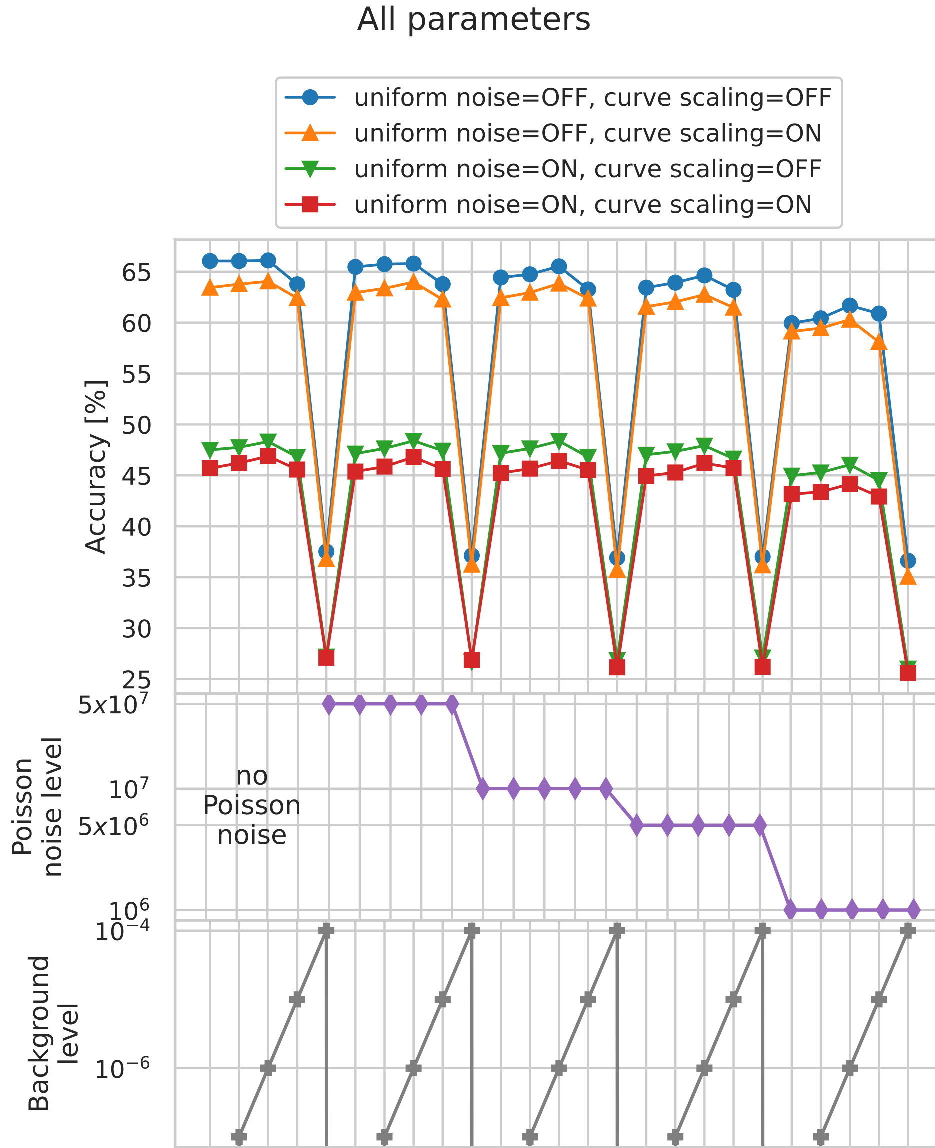

Figure 5.11.: Summary of the prediction accuracy of the test curves for 100 different noise and background combinations. Each point in the top panel refers to the prediction accuracy of test curves with a specific noise and background combination. The four different colors/symbols distinguish between the four binary combinations of uniform noise and curve scaling being turned on or off, respectively. The horizontal axis distinguishes between the 25 combinations of Poisson noise and background levels. The combination of each noise level and background level can be read from the bottom two plots. The gray line (crosses) indicates the added background b for a given point along the horizontal axis, with the background periodically increasing from left to right. Similarly, the purple line (diamonds) indicates the incident intensity s that was used to calculate the Poisson noise. Since lower intensities mean higher relative noise, the noise added to the curves increases from left to right. Figure adopted from [107].

two plots. The gray line (crosses) indicates the added background b for a given point along the horizontal axis, with the background periodically increasing from left to right. Similarly, the purple line (diamonds) indicates the incident intensity s that was used to calculate the Poisson noise. Since lower intensities mean higher relative noise, the noise added to the curves increases from left to right.

It is apparent that uniform noise and added background have the strongest impact on the prediction accuracy, whereas curve scaling and Poisson noise only have a minor influence. When curve scaling is turned on, the accuracy is decreased by 1–2 percentage points independent of any other modification. In contrast, Poisson noise seems to only play a role for incident intensities of 10^7 or lower, with its strongest effect at 10^6 where the accuracy is reduced by about 5 percentage points compared to the case without Poisson noise. When adding background to the curves, the performance remains almost unaffected for background levels up to 10^{-6}. However, for levels of 10^{-5}, we observe a small decrease in accuracy and for 10^{-4}, the accuracy is reduced by 30 percentage points compared to the unmodified curves. This strongly suggests that there is a critical value above which an additive constant background will obfuscate too much information and therefore make a lot of the curves unsolvable for the NN.

It is important to note that the accuracy drops by about 20 percentage points when uniform noise is turned on. The difference between Poisson noise and uniform noise is that the latter also affects regions of high intensity, such as the TRE. Thus, the reason why the detrimental effect of uniform noise is relatively strong might be related to information that is encoded in the TRE as described in the last section. By modifying the TRE, some of the information might be lost. The same is likely true for the curve scaling, since it also affects the TRE, but the scaling factor might not be large enough to produce a strong effect. Interestingly, the decrease in accuracy caused by curve scaling and uniform noise compounds with the effect of a high background. This suggests that these effects are independent of each other, since a high background mainly affects high-q_z features while scaling and uniform noise affect mainly low-q_z features (*e.g.* the TRE).

5.4.5. Other challenging cases and prospects

In addition to the issues discussed before, further challenges may arise from more complex situations given by the systems under study. First of all, we have assumed a box-like SLD. If the SLD exhibits a profile incompatible with the approximation by a box, *e.g.* a sloping or graded profile, further and more elaborate training is most likely needed. Second, samples that consist of multiple layers (*i.e.* beyond one layer on a substrate) are not included in this study. The incorporation is in principle straightforward, but of course the solution space increases with the number of parameters, and difficulties are expected when different combinations or orders of layers result in similar XRR or NR curves. In these cases it might be necessary to constrain the solution space of the inverse problem by providing any available *a priori* knowledge about the studied system to the NN. This is discussed further in Chap. 7.

For some applications, it is common to combine multiple data sets during analysis. This is certainly possible for ML approaches, but it requires a broader training strategy. One application may be reflectivity time series during growth, annealing or oxidation experiments, where it is beneficial to fit all XRR or NR curves of a series together. By applying boundary conditions, such as demanding a monotonically increasing thickness, the ambiguities in the analysis can potentially be reduced. Another example concerns NR from *magnetic* structures where several data sets with different polarization (↑↑, ↑↓, etc., as well as spin flip and non-spin flip) need to be fitted simultaneously [87], possibly with prior knowledge from XRR measurements to determine the chemical structure. An example of this approach has also been demonstrated by others [108]. Thus, there is no fundamental reason why ML could not be employed for other methods as long as the approach is sufficiently tailored to the data.

5.5. Conclusions from this chapter

This chapter provides insights into the behavior of NNs when predicting thin film parameters from reflectivity data in the light of certain challenging cases. These in-

sights are necessary to understand which types of reflectivity curves can be processed easily by the NN and which are more difficult. This understanding will help further improve and adapt the design of machine learning models to the specific needs of scattering data for which a simple inversion is not possible. The results show that 3 subsets of the reflectivity data seem to be particularly difficult for the NN: 1) Curves with ambiguous solutions where the MSE surface between the curve and the fit as a function of the parameters is flat, 2) curves where the SLD of both the layer and the substrate are negative (*i.e.* no TRE), and 3) curves with noise on low-q_z features or particularly high background.

When tested on noise-free data, the trained NN was able to correctly predict all 5 thin film parameters 48 % of the time (individual accuracies ranged from 71–92 %). It was identified that subsets 1) and 2) mainly consist of curves that lack information-rich features such as oscillations or a TRE. Thus, by removing these cases from the test set the accuracy increased to 66 % (82–93 % for individual parameters). By further studying the influence of different noise and background sources, we showed that, if applied to the training set, most curve modifications do not significantly impact the prediction accuracy. However, if a critical threshold of 10^{-4} for the background was crossed, the accuracy dropped significantly. Fortunately, this value is rarely exceeded in experimental data, since high backgrounds are usually already subtracted at the data reduction step, leaving only a smaller residual background. Thus, background is not likely to play an important role when using the NN on real data (within the studied q_z range). Furthermore, when moderate uniform noise was applied to the data, the performance of the NN was noticeably affected due to its effect on the TRE. This further shows the importance of the TRE as a source of information and, together with the SLD dependence of the performance, indicates that the NN model has learned to extract critical information from the TRE and is picking up on real physical features of the data.

To further improve the NN performance in the future, there are two clear strategies, both of which rely on a narrowing of the task. Firstly, one might choose to narrow the task of the NN to cases with well-defined solutions and remove classes

of difficult edge cases from the training set. This approach might be favorable when the experimental data is expected to have clear features where training with data without clear features would only serve to make the task harder without any benefit. This is also the approach taken in Chap. 6. A second approach might be to select a subset of difficult edge cases that are most likely to appear in the experimental data (such as curves without a TRE) and create a training set that focuses on these cases. The present work may also give an indication about the information content of different reflectivity curves. Under the premise that NN is able to extract close to the maximum amount of information from a reflectivity curve, the difficult curves identified herein may also be curves with a low amount of physical information. Therefore, simulations and predictions using the shown NN may help with experimental design through identifying ambiguous and difficult measurement results and then avoiding these parameter combinations or complementing them with additional information.

Lastly, since the results of this chapter are based on simulated data, future efforts should also be focused on translating these achievements to experimental data. To this end, it is necessary to investigate other data imperfections such as a finite angular and energy resolution and the influence of the beam shape and their effect on the prediction accuracy. Furthermore, Chap. 6 will also show that experimental data is necessary to evaluate how much noise should be added to the training data.

6. The analysis pipeline of the *mlreflect* package[‡]

6.1. Introduction

The principle of applying NNs to the task of reflectivity data analysis has been discussed in Chap. 4 while Chap. 5 discussed potential challenges as well as different ways of preprocessing the data. This chapter focuses on the differences between simulated and experimental data and how this knowledge can be used to further optimize the obtained results. Furthermore, the lessons learned from this and previous chapters are combined in a Python-based reflectivity data analysis package called *mlreflect* that can reliably predict the thickness, roughness and SLD of a thin film layer. The performance of this pipeline was tested on a large experimental dataset of 242 XRR curves from different organic thin films on Si/SiO_x substrates by comparing the result of the pipeline with manually supervised LMS fits that include physical knowledge and carefully chosen boundary conditions. This is a quantitative and qualitative difference compared to other studies [103–109, 112], where most or all of the performance analysis is done with simulated data. In this context, the effect of experimental deviations from the theory on the training and prediction quality of the NN will be discussed. Using an example curve, it will be shown how the

[‡]This chapter is largely based on the publication Greco *et al.* 2022 [111]. The *mlreflect* package was developed by A. Greco. E. Edel helped implement the FFT described in Sec. 6.6. All data analysis was performed by A. Greco. The samples used for testing were prepared and measured by A. Hinderhofer, C. Lorch, S. Kowarik, I. Dax, N. Rußegger and A. Greco, as described in Tab. F.1. F. Bertram and C. Shen helped implement the *mlreflect* package at P08/DESY. A. Gerlach provided computational support and F. Schreiber supervised the project.

extremely fast prediction speed of the NN can also be leveraged to compensate for small experimental errors.

Moreover, *mlreflect* was developed as part of a project funded by the BMBF and was made available on the package repository of the Maxwell cluster at DESY. It was successfully employed at beamline P08 for the *in situ* analysis during the annealing of organic thin films. The package and source code are also fully available online on GitHub and the Python Package Index (PyPI) as detailed in Appendix D. Everything described in this chapter refers to version 0.21.0 of *mlreflect* which is archived on `https://doi.org/10.5281/zenodo.6467048`.

6.2. Description of the analysis pipeline

6.2.1. Overview

The NN is implemented using TensorFlow [30] and the reflectivity data is simulated using the the matrix formalism implemented in the Refl1d package [93], as described in previous chapters. After the experimental data is read, the workflow of the *ml-reflect* package can conceptually be separated into three steps: I. preprocessing, II. prediction and III. postprocessing, as depicted in Fig. 6.1. Each of these steps is described in the following.

6.2.2. Preprocessing (step I.)

6.2.2.1. Reading of the raw data

Before any data can be processed, it has to be read from its raw format, *i.e.* how it was saved by the measurement software. The current version supports the reading of raw data in the SPEC and FIO formats through the `SpecLoader` and `FioLoader` classes of the `xrrloader` subpackage.

SPEC[1] is a well-established software package for instrument control and data acquisition that has been in use at many large-scale facilities and university labo-

[1] `https://certif.com/`

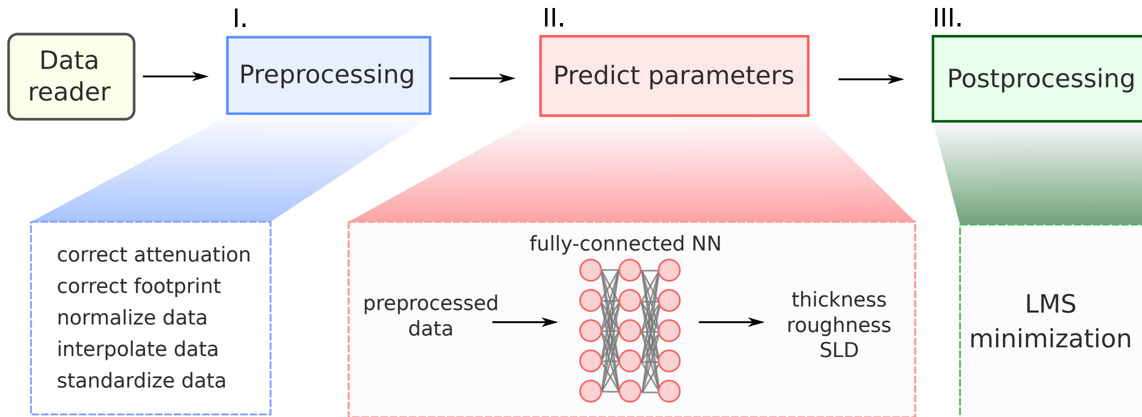

Figure 6.1.: A schematic description of the analysis pipeline. The pipeline consists of three main steps: I. preprocessing, II. parameter prediction *via* the NN and III. postprocessing. Step I. includes geometrical and other experiment-specific corrections. The data is also normalized, transformed into q_z space, interpolated and standardized. In step II., the preprocessed data is fed into a trained, fully-connected NN that yields an initial guess for the thin film parameters. During step III., this initial guess is used as starting parameters for a fast Levenberg-Marquardt fit that finds the nearest LMS minimum. Figure adopted from [111].

ratories for decades. Although slowly being replaced by newer software, it is still in common use and many facilities continue to offer legacy support or backwards compatibility. In terms of reflectivity, the file format of SPEC saves the individual intensity values $\boldsymbol{R}$ alongside the corresponding scattering angle $\boldsymbol{\theta}$ (and many other parameters) in a simple table. For 2D detectors, the intensities are usually already integrated over a certain region of interest (ROI), *i.e.* the part of the image that contains the reflected beam. The column-based format of SPEC files makes them easy to read for both humans and machines, but it sometimes lacks the flexibility to save and reprocess more complex data. In *mlreflect*, since the ROI is fixed for SPEC files, the user only has to provide the names of the columns that contain the intensities and angles (which are usually specific to a given setup) to the `SpecFitter` class of the `mlreflect.curve_fitter` subpackage.

In contrast, FIO is a proprietary DESY format where the 2D detector images for each data point are saved together with a metadata file that contains the angles, but also specifies the ROI. This allows adjusting the ROI even after the measurement,

which can be used to improve the resolution. This also means that for *mlreflect*, an explicit ROI has to be provided in addition to the names of the angles so that the expected intensity values $\boldsymbol{R}$ are obtained.

In principle, the package could be extended with more classes that accommodate different data formats without loss of any functionality as long as the data can be eventually converted into two vectors $\boldsymbol{R}$ and $\boldsymbol{q}$ that contain the reflected intensity and its corresponding q_z values, respectively. A prominent candidate would be polychromatic measurements which are more common for NR, where the different wavelengths in the spectrum would have to be converted into the correct q_z values.

6.2.2.2. Data rescaling

Often, experimentally acquired data has to be rescaled to account for technical differences in the incoming intensity. A common reason is, for instance, that different levels of attenuation are applied to the beam for different q_z positions. This is done to avoid damage to the detector since the reflected intensity can change by several orders of magnitude between different q_z positions. As a result, the intensity of the measured reflected beam has to be normalized by the attenuation factor.

Another typical correction would be the rescaling of the reflected intensities in polychromatic experiment. As already mentioned in Chap. 5, the intensities in the spectrum of a polychromatic beam are usually not uniformly distributed and thus, the reflected intensity at a given q_z has to be normalized by the relative incoming intensity of its corresponding wavelength.

Many other reasons for data rescaling exist and could be implemented in future versions this package. In the current version of *mlreflect*, only attenuation correction is applied by default, which is implemented in the `ReflectivityLoader` class of the subpackage `mlreflect.xrrloader.dataloader`.

6.2.2.3. Footprint correction

The *mlreflect* package provides footprint correction to account for the changing beam footprint on the sample at different angles, which amounts to a multiplication of

the data with a geometric factor [163]. These are implemented in the subpackage `mlreflect.xrrloader.footprint`. Here we assume a flat sample and a beam with a Gaussian profile but, in principle, corrections for other sample or beam shapes can be implemented at this stage. The data is then normalized by dividing by the highest intensity value and is transformed from angular space into q_z space.

6.2.2.4. Data interpolation

Since the NN model is trained on a fixed number of fixed q_z points and in general, the experimental q_z values may differ from that, the corresponding measured intensities have to be re-interpolated. Thus, all reflectivity curves are automatically interpolated on a logarithmic scale to the 109 equally-spaced q_z expected by the NN (see Subsec. 6.2.3). The interpolation is implemented in the `mlreflect.data_generation` subpackage.

While in principle any experimental q_z values are possible, it is recommended to use a similar q_z range and a similar point density to what is expected by the NN to avoid strong interpolation artefacts that may negatively impact the performance. If a significantly different range or point density are required, the model should be re-trained.

6.2.2.5. Input standardization

To ensure that all intensity values of a given input curve are on a similar scale, the data is standardized in the same way as was described in [107]. The effect on the general shape of the curves is comparable to multiplying the data with the inverse of the Fresnel reflectivity $R_\mathrm{F}(q_z) \propto q_z^{-4}$, but, importantly, avoids the divergence for small values of q_z, *i.e.* close to and below the total reflection edge, where the kinematical approximation does not hold [1]. The input standardization is implemented in the `InputPreprocessor` class of the `mlreflect.training` subpackage.

6.2.3. Neural network predictions (step II.)

The initial parameter prediction (step II. in Fig. 6.1) is obtained by feeding the preprocessed input vector into the trained NN model. As described in previous chapters, the NN employed here is a fully-connected model that takes an input of discrete intensity points at equally-spaced q_z values ranging from 0.02 to 0.15 Å^{-1}. To better match the point density of the annealing experiments described in Sec. 6.7 the input size was increased slightly to 109 points. The output of the NN consists of 3 thin film parameters: the film thickness, the Névot-Croce film roughness [122] and the real part of the scattering length density of the film. Compared to Chap. 5, the size of the NN was reduced to three hidden layers with 512 neurons each due to the smaller number of predicted sample parameters. The training loss was calculated as the MSE between the normalized predicted and ground truth parameters. The model was trained on 250,000 simulated reflectivity curves with a batch size of 512. For every batch, uniform noise and curve scaling were applied to each curve and the inputs were standardized as described in Chap. 5. The optimal noise level during training was identified to be 0.3, which will be discussed in more detail in Sec. 6.4. The noise and curve scaling are implemented in the `noise` module of the `mlreflect.data_generation` subpackage. The training data was generated assuming a sample structure of an organic thin film on top of an oxide-capped silicon substrate with air as an ambient medium and with X-rays as the probe. The thin film parameters in the training data spanned a large range of 20–1000 Å for the thickness, 0–100 Å for the roughness and 1–14$\times$10^{-6} Å^{-2} for the SLD, which is significantly larger than what was described in Chap. 4. Each parameter was sampled from a bolstered uniform distribution as described in Fig. 5.1. Also, as described in Chap. 4, the maximum roughness was restricted to no more than half the thickness of the film. The sampling of the parameters is implemented in the `ReflectivityGenerator` class of the `mlreflect.data_generation` subpackage.

Moreover, a similar approach could easily be employed for neutrons or other sample structures by retraining the neural network with different training data. This

approach is also expected to work for a larger number of layers as long as the trained parameter space does not create too many ambiguous solutions, *i.e.*, the number and range of fitting parameters should remain similar. For a larger parameter space, an extended q_z range might be necessary to reduce ambiguity in the data. For the purposes of this work, the q_z range was limited to avoid conflicts with the Bragg peaks of organic molecules around 0.3 Å^{-1} which are not described by the slab model.

6.2.4. Postprocessing (step III.)

6.2.4.1. Optimization of the *q* shift

After the initial parameter predictions have been obtained, *mlreflect* provides the option to perform an optimization for a possible q_z shift in the data. Using this method, the parameter prediction as described above is repeated a certain number of times (by default 1000) while shifting the q_z values of the measurement by a small, randomly chosen value before the data is interpolated. Then, from all predictions, the best one is chosen. This is meant to correct small misalignments that can occur during sampling the measurement. Furthermore, it introduces small perturbations into the prediction, allowing for sampling from many similar solutions, thereby improving the results. This step is implemented in the `CurveFitter` class of the `mlreflect.data_generation` subpackage and a detailed explanation and justification of the q_z shifting method is given in Sec. 6.5.

6.2.4.2. LMS refinement of the results

The results from the previous step are further optimized by a conventional LMS minimizer that searches for the parameters that produce the best fitting reflectivity curve for the measured data. Since the initial predictions can be used as starting parameters and are already very close to the ground truth, a simple and fast minimizer, like the Levenberg-Marquardt algorithm [164], is used over more powerful, but slower methods, such as stochastic algorithms [96, 97, 102]. The LMS fit is implemented

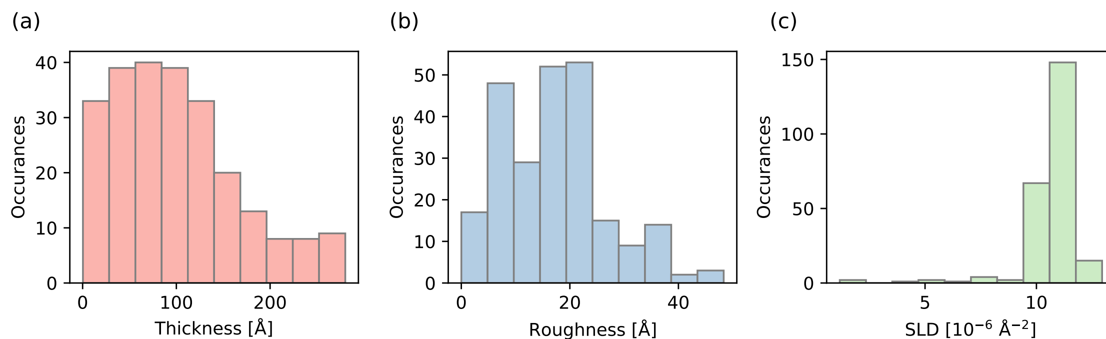

Figure 6.2.: Ground truth distribution of the three sample parameters thickness (a), roughness (b) and SLD (c) within the experimental test set of 242 XRR curves. The parameters were obtained by a conventional LMS fit. Figure adopted from [111].

in the `minimizer` module of the `mlreflect.curve_fitter` subpackage and uses the `scipy.optimize.curve_fit` method.

6.3. Performance test on thin films

The performance of the analysis pipeline was tested on 242 experimental XRR curves from *in situ* and *ex situ* experiments of 9 organic thin films on Si/SiO$_x$ (1–79 curves per sample at different thicknesses). A more detailed description of the datasets can be found in Appendix F. The distributions of thickness, roughness and SLD of the film within this test set is shown in Fig. 6.2. The measurements were conducted using three different synchrotron radiation sources: the European Synchrotron Radiation Facility (ESRF) [165], DESY [166] and the Swiss Light Source (SLS) [167], as well as using our own laboratory source. Similar to Chap. 4, to obtain a benchmark, each reflectivity curve was first fitted on a logarithmic scale with an LMS fit based on the commonly used differential evolution algorithm [102] using manually chosen initial values and bounds for each parameter. The thin film model used for the fit was the same as what was used for training the NN.

In the following analysis, these manually fitted parameters represent the ground truth, and thus the performance of the pipeline is measured as the absolute error with

respect to this ground truth. The ground truth was compared with the prediction results of the NN (step II.) as well as the results of an automized subsequent LMS fit using the predicted parameters (step III.). Across all 242 curves, the NN predictions have a median absolute error (median percentage error) of 6.0 Å (7.1 %) for the film thickness, 2.0 Å (12.4 %) for the interface roughness and 0.72×10^{-6} Å^{-2} (6.8 %) for the SLD. Both on an absolute scale as well as on a relative scale, this is a significant improvement to the first published model [103] demonstrated in Chap. 4, since the possible ranges for the thickness and roughness parameter have been greatly expanded and the network is generalized over a larger parameter space. Importantly, since all of the data stems from organic thin films, the SLDs in the test set are mainly clustered around 10–13$\times 10^{-6}$ Å^{-2}. Since the NN was trained equally with SLDs ranging from 1–14$\times 10^{-6}$ Å^{-2}, the results are not assumed to be specific to the test data as long as absorption does not play a major role. Moreover, it is important to highlight that the dataset also contains curves with a high roughness-to-thickness ratio where the Kiessig oscillations are strongly damped. In Chap. 5 it was shown that these "flat" curves are usually more challenging to predict and among the emerging solutions offered in this field, discussions about the performance on curves with little to no features are mostly absent. This is of course due to the challenge of extracting information from data that inherently contains less information. Yet, the network presented here also performs well on experimental data with high relative roughness.

The next step in the pipeline is to further refine these results *via* an LMS fit using the predictions from the NN as starting parameters. Since the predictions are robust and already quite close to the ground truth there is no need for powerful but slow minimization algorithms such as genetic or differential evolution algorithms, which are normally employed to find the global minimum. Thus, finding the minimum takes only a fraction of a second per curve and can be fully automized. After this refinement procedure, the median absolute error (median percentage error) was even closer to the ground truth with 2.3 Å (2.3 %) for the thickness, 1.0 Å (5.8 %) for the roughness and 0.47×10^{-6} Å^{-2} (4.3 %) for the SLD. A comparison of the error

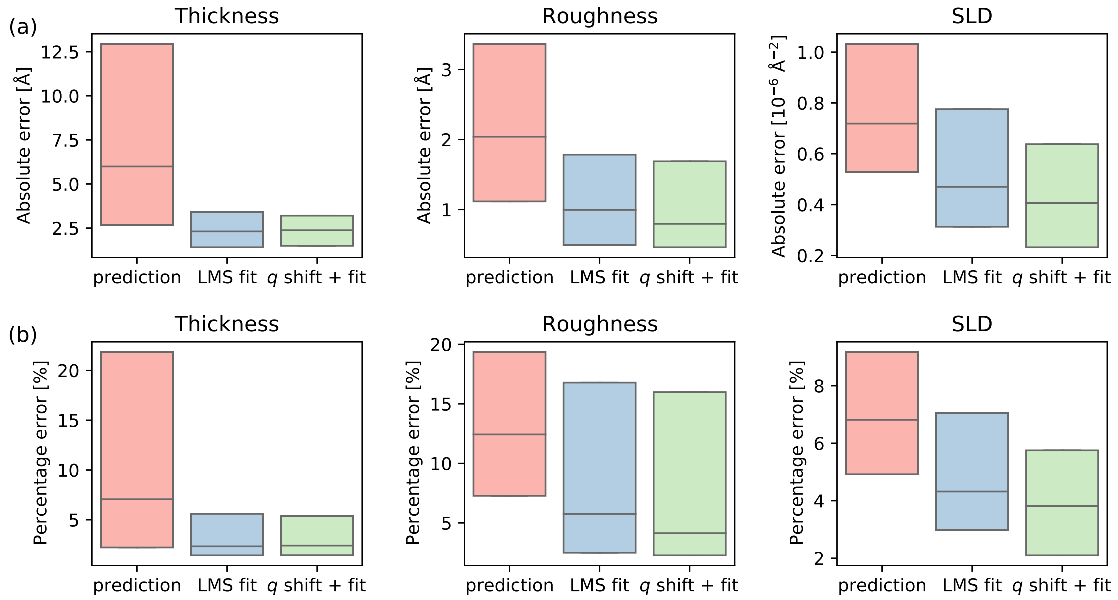

Figure 6.3.: a) Box plot of the absolute errors for 242 measured reflectivity curves for each of the three predicted parameters. The upper and lower edge of the boxes represent the 1st and 3rd quartile with the horizontal line inside the boxes denoting the median. The red boxes represent the error compared to the pure NN predictions. The blue boxes represent the error after applying a simple LMS minimization using the NN predictions as starting parameters. The green boxes show the error when a q_z shift optimization was performed before the LMS fit. b) The same box plots of the median error but as a percentage of the ground truth. All results were obtained for a training noise level of $n = 0.3$. Figure adopted from [111].

distributions before and after refinement is shown in Fig. 6.3. A detailed breakdown of the prediction error with respect to each parameter can be found in Appendix C.

The residual error can be easily attributed to the fact that every fit has a finite accuracy and hence the ground truth itself contains a certain error which is likely comparable to the reported error. In conclusion, these results show that the analysis pipeline as described above performs similarly to a human researcher in most circumstances. Furthermore, it is important to note that the results were obtained much faster than *via* a human-guided fit. Excluding the time it took to train the NN (about 20 min for a given sample structure), the initial parameter predictions of the 242 curves were obtained after only 1 s with about 2 additional minutes for the further refinement steps, resulting in a total fitting time of about 0.4 s per curve. In contrast, producing the ground truth fits took about 6 h because of the need to carefully select fitting boundaries to prevent the fit from running into non-physical minima.

6.4. Differences between simulated and experimental data

A well-known property of artificial NNs is that they require large amounts of representative training data to learn a generalized model and not overfit the training set. In the context of the work presented here, *i.e.* supervised learning using scattering data, this would arguably mean acquiring thousands of scattering patterns from a large variety of different samples and analyzing them manually to create the training set. Since this is a quite time-consuming and challenging task, NN models in the field of scattering physics are typically trained with simulated data based on well-established theoretical models. In most cases, the simulation is additionally modified with certain artifacts, such as noise, to better mimic experimental conditions. However, to what degree this is necessary is difficult to estimate since the only available

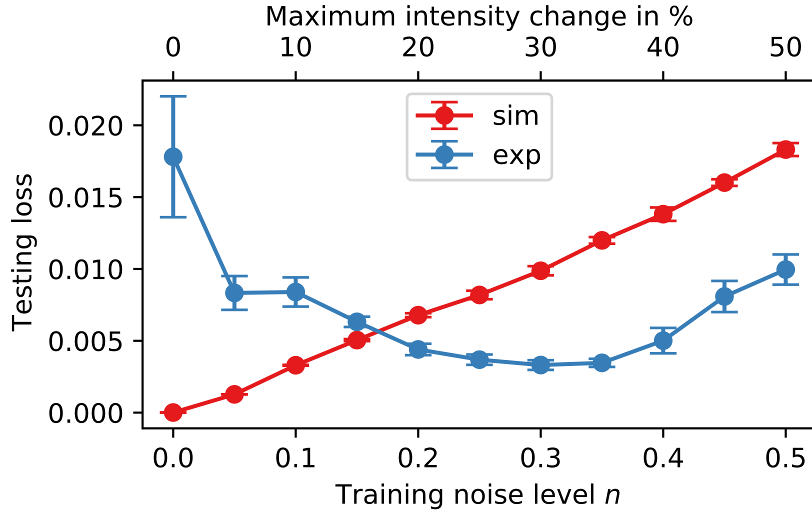

Figure 6.4.: Comparison of the testing loss calculated from a simulated test set (100000 curves) and an experimental test set (242 curves) for different levels of uniform noise n that were applied to the training data. For each noise level a separate model was trained. With increasing noise level, the loss from the simulated data increases linearly while the loss from the experimental data shows a clear minimum at noise levels 0.3–0.35. The error bars represent the standard deviation from 5 independent training repetitions. Figure adopted from [111].

metric is typically the performance on other simulated data (validation loss), which is expected to become worse with increasing perturbations.

In this section, we investigate how applying uniform noise to the training data affects the NN performance on the previously mentioned experimental dataset of 242 curves. For this purpose, 11 copies of the same NN were trained as described above using data with different levels of uniform noise, *i.e.*, for a given noise level n, each data point R_i^* in the noisy curve was sampled uniformly between the values $R_i(1-n)$ and $R_i(1+n)$.[1] Thus, n denotes the maximum relative change of a given data point R_i of a given simulated curve. The n for each trained model ranged from 0 to 0.5 in 0.05 increments. It is important to note that the applied uniform noise is not meant to model a specific physical noise type, such as Poisson noise for counting statistics. Rather, uniform noise was chosen as a q_z-independent catch-all noise that affects the

[1] The type of uniform noise described here is the same as in Subsec. 5.2.2.2 if $n = 0.3$.

whole curve equally and thus, makes the NN robust against errors across the entire q_z range. Fig. 6.4 shows a comparison of the losses calculated with a simulated test set as well as with the experimental test set for each model. Since the loss is calculated as the mean squared error of all three (normalized) sample parameters, it is a unitless measure for the overall accuracy of the model. For $n = 0$, the simulated test set shows a loss close to zero ($\sim 10^{-7}$), whereas the loss based on the experimental data is about 5 orders of magnitude higher. This shows that without any noise, the NN significantly overfits the simulation and thus performs suboptimally on real data. As expected, the loss of the simulated data increases monotonically with increasing noise, since the information content of the data is reduced. In contrast, on the experimental dataset, performance improves significantly with increasing noise up to a noise level of 0.3–0.35. Beyond this, even higher noise levels seem to again worsen the performance. This very clearly demonstrates that there exists an optimal noise level for which the added noise acts as an effective regularization technique that prevents overfitting. If the noise level is too high, however, the consequent lack of information in the training data is likely detrimental to the training process. Thus, $n = 0.3$ was chosen as the ideal noise level for data similar to the experimental test set, which notably contains data from different X-ray sources.

An interesting question arises about how this value is related to the amount of noise in the experimental data. To investigate this, the test data was separated into four groups with varying amounts of noise. While the noise in the data is not uniformly distributed, an equivalent noise level (ENL) can be calculated by subtracting the ground truth fit from the data and taking the absolute mean of all data points. Fig. 6.5a shows the distribution of the ENL across the entire dataset and how the distribution was split into the four subsets with a different ENL. Fig. 6.5b shows the optimal training noise (for which the loss had a minimum) for each of the four categories, as well as the entire dataset. The error bars represent the standard deviation of five independent training repetitions. Evidently, the ENL of the data does not seem to have a strong influence on the optimal noise level except for the 0.4–0.5 category, where it is slightly lower. This is due to the main error in the data

not being statistical (*e.g.* Poisson noise), but rather systematic in nature. Since the role of the uniform noise on the training data is not to mimic the noise in the data, but to account for these systematic deviations, the entire dataset benefits from a similar training noise level. Hence, this noise level has been implemented as the default value in the *mlreflect* analysis pipeline. Data that differs significantly from the test set in terms of experimental artifacts might of course produce slightly different results, although the general trend is expected to be the same. This highlights the importance of having a large experimental test set with representative experimental artifacts, since metrics based only on simulated data are clearly not sufficient to evaluate the training.

6.5. Influence of systematic measurement errors

All reflectometry measurements are performed with a finite accuracy due to a number of different error sources. Since these errors are detrimental to the experiment they can potentially impede the extraction of information from the data and therefore should be avoided or minimized as much as possible. However, a finite error inevitably remains for every measurement and the precise nature of experimental errors strongly depends on the chosen scattering geometry. Among the typical statistical errors are the signal-to-noise ratio, the angular resolution of the diffractometer and the spectral resolution of the source [168]. Among the systematic errors are, for example, the convolution of the data with the slit functions [169], the accuracy of the sample alignment and the accuracy of the footprint correction, *i.e.* how accurate the beam and sample shape can practically be determined. Having imperfect data obviously impacts the analysis, since the data deviates from the ideal physical model it is compared with. Since the NN model presented here is trained to solve a very particular task that assumes well-defined data, these errors can negatively impact the prediction quality. In general, it is easier to make the NN robust against statistical errors by introducing them during training, as described before. However, sometimes systematic errors, such as a small misalignment can also seriously misguide the ML

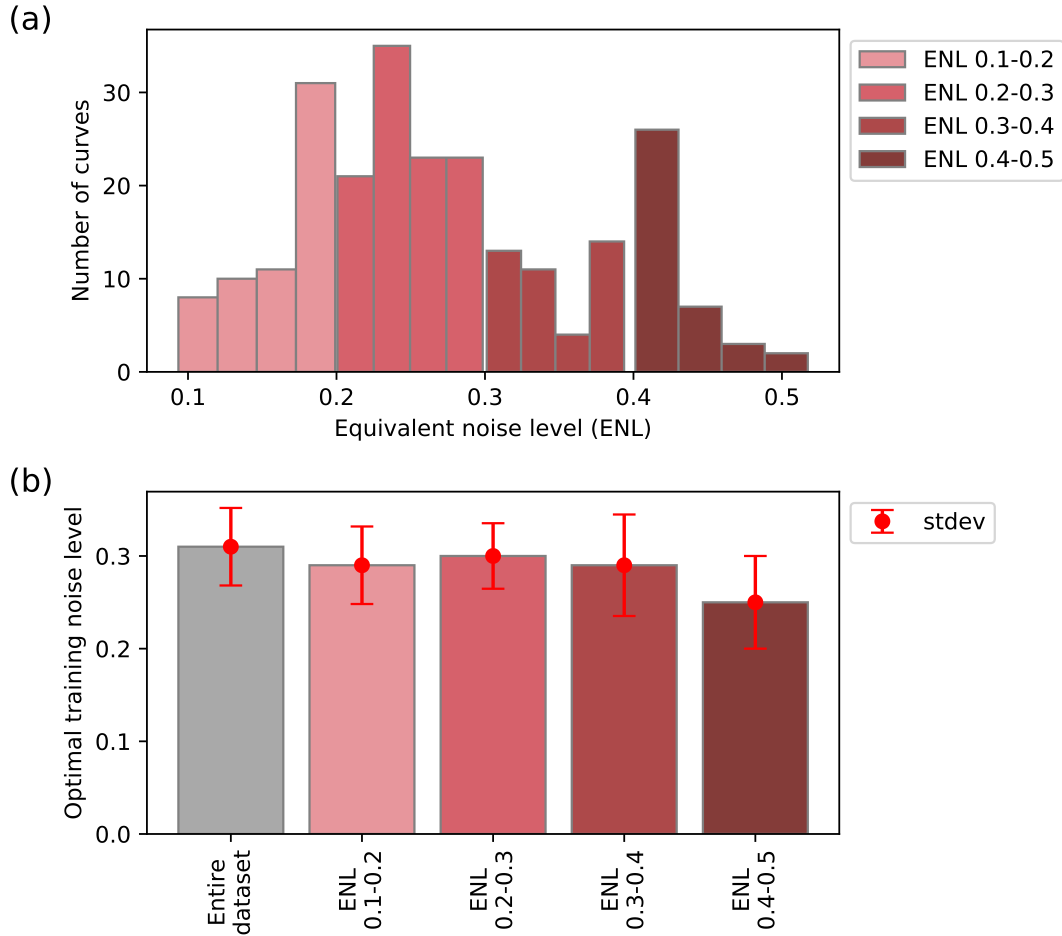

Figure 6.5.: (a) Distribution of the equivalent noise level (ENL) in the experimental testing dataset of 242 XRR curves. The dataset was split into four categories with varying ENLs to test each separately. (b) Optimal training noise for different ENLs in the training data. The optimal level for the entire dataset corresponds to the minimum shown in Fig. 6.4 of the main manuscript. The error bars represent the standard deviation of five independent training repetitions. Figure adopted from [111].

prediction, as shown in Fig. 6.6. Therefore, it would be useful to correct or compensate some of these errors during inference time after the data has been acquired. A possible solution could be an automated method for sampling through slight variations of the data, exploiting both the sensitivity and speed of the trained NN model. Since the NN assumes data that conforms to an idealized physical model, it might fail if the data contains anomalies with respect to that model. Since predictions with the NN are very fast, it is possible to scan through thousands of modified reflectivity curves within less than a second. For each of these variants, the log MSE between the data and the predicted curve as defined by Eq. 5.9 can be calculated and only the one with the lowest error is subsequently selected. An implementation of this method that identifies small systematic alignment errors and automatically applies an appropriate shift to the data is demonstrated in the following.

Fig. 6.6a shows an XRR measurement of a 690 Å thick N,N'-Dioctyl-3,4,9,10-perylenedicarboximide (PDI-C8) film on Si/SiO_x which was measured and tested in addition to the 242 test curves. Here, the normal pipeline as described above did not converge to the correct minimum since the initial guess of the NN was not close enough. The reason for this is the much higher thickness of the film, which leads to denser Kiessig oscillations in the data. This, in turn, creates many narrow minima on the MSE surface for the LMS algorithm to get trapped in. As a result, the NN prediction needs to be even closer to the ground truth for the subsequent fit to converge. Tab. 6.1 shows the predicted thin film parameters in comparison to the ground truth. A possible reason for the suboptimal NN prediction might be small imperfections in the data due to finite measurement errors, such as a small variation in sample alignment. In regions of high derivatives, even a small shift of the data along the q_z axis can lead to strong differences in the observed intensities at a given q_z value, even on a logarithmic scale. Of course, if the data has dense oscillations, this effect becomes more pronounced. For models trained on simulated data, this can be critical, since normally a substantial change of certain input neurons, especially near the TRE, corresponds to important information and will be interpreted by the network accordingly. To check whether this can be remedied, the q_z values were

Table 6.1.: Predicted and fitted thin film parameters based on the reflectivity data of a PDI-C8 film on Si/SiO_x (shown in Fig. 6.6). The ground truth labels were obtained *via* a manually supervised LMS fit. After applying the described q_z variation, the prediction results improved significantly. A subsequent LMS refinement only led to comparatively small improvements. Table adopted from [111].

	Thickness [Å]	Roughness [Å]	SLD [10^{-6}Å^{-2}]
ground truth	688.3	27.1	10.5
prediction	536.7	30.3	11.2
shift + prediction	690.8	31.0	11.0
shift + prediction + fit	690.5	27.5	10.8

shifted by a small value Δq_z during the interpolation step before passing the data to the NN. This procedure was repeated 1000 times with randomly sampled Δq_z ranging from -1×10^{-3} to 1×10^{-3} Å^{-1}. Then, for each prediction, the quality of the prediction was evaluated by calculating the log MSE between the corresponding simulation and the measured curve.

When plotting the log MSE between the prediction and the input against Δq_z (Fig. 6.6), we can observe a value $\Delta q_{\mathrm{min}} = 5.2\times10^{-4}$ Å^{-1} for which the log MSE shows a clear minimum. From Fig. 6.6a it is apparent that the predicted curve based on the shifted data shows much better agreement with the data than the normal prediction. The corresponding predicted parameters for Δq_{min} are shown in Tab. 6.1 and are much closer to the ground truth values (comparable to values shown in the previous section). This indicates that there exists a certain shift Δq_{min} that can (at least partially) compensate for the experimental error. This is especially valuable since it allows the pipeline to continue with the LMS refinement step, which ultimately leads to a near-perfect fit. It is interesting to note that Δq_{min} is very small, corresponding to a change of the angle of incidence of only about 4×10^{-3} degrees for a wavelength of 1.54 Å. It seems intuitive that such a small shift in the data could be caused by a variety of the above-mentioned error sources. However, although Δq_{min} is seemingly small, due to the high derivatives close to the TRE and the Kiessig fringes, shifting the data by Δq_{min} still has a noticeable effect on each data point. For conventional LMS fitting, this might not seem critical at first, since the MSE surface likely has

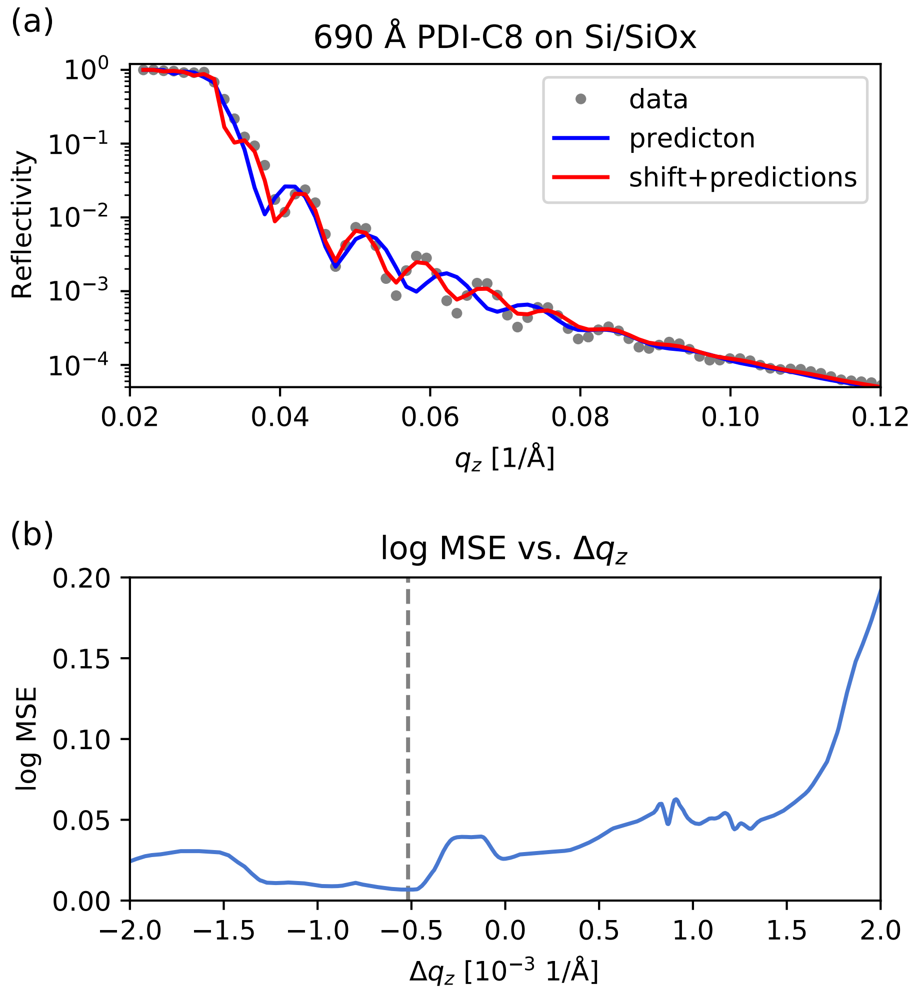

Figure 6.6.: a) Comparison of the NN predictions from reflectivity data from a 690 Å thick PDI-C8 film on Si/SiO_x. The blue curve shows the native prediction, whereas the red curve shows the prediction after the data was shifted by $\Delta q_{\mathrm{min}} = 5.2 \times 10^{-4}\,\text{Å}^{-1}$ before the interpolation step. It is apparent that the latter is in much better agreement with the data. b) Shows the log MSE between the predicted curve and the data for different Δq_z. The minimum MSE at Δq_{min} is indicated by the dashed line. Figure adopted from [111].

a minimum close to the real one in terms of the film thickness. However, for the roughness and density parameters this might not be the case and thus, most fitting programs allow the user to manually shift the data if necessary.

While in principle any type of modification like this could be conceivably applied to the data to scan for the lowest MSE, this method has shown significantly better results than varying the data randomly, such as adding Gaussian noise. This is because a translation of the curve preserves most of the information in the data while still varying every data point, in contrast to Gaussian sampling which is q_z-independent and inevitably destroys information. To test the stability of this method, the Δq_z sampling procedure was applied to all 242 curves discussed in the previous section (where the pipeline already succeed) and compared the results with the original mean absolute error. When looking at Fig. 6.3, it becomes clear that scanning for Δq_{min} did not harm the mean absolute error, but instead on average even improved the results slightly for all three parameters. While the log MSE of the predictions is already very close to the minimum, most of the data likely still has a finite alignment error which, however, was apparently not sufficient to affect the prediction. This can still be compensated by applying a small shift, ultimately leading to an even better fit. Because this screening for Δq_z yielded significant improvements on some data and was relatively fast (still less than a second per curve), this method is by default enabled in the analysis pipeline of the *mlreflect* package.

6.6. Fourier transforms as a method for feature engineering

As mentioned in Sec. 2.2, the specular reflectivity $R(q_z)$ of a single layer on a substrate well above the critical angle can approximately be described as the product of the Fresnel reflectivity $R_{\mathrm{F}}(q_z)$ from a flat surface and the squared FT of the SLD contrast of the sample along the surface normal (see Eq. 2.49). Although the phase of the FT is lost by taking its absolute square, the inverse FT of $R(q_z)/R_{\mathrm{F}}(q_z)$ still carries some

important information, such as the frequency of the Kiessig oscillations (and thus the film thickness). As a result, performing an inverse FT on the reflectivity data presents itself as an obvious way to create additional input features that may facilitate the NN training. To test this hypothesis, a NN model was trained with an additional preprocessing step before the first layer that performs a fast Fourier transform (FFT) on the standardized input and adds the real and imaginary Fourier components to it, leading to a input layer size of 219 neurons. All other model parameters and training ranges were kept the same as described above. When testing the trained model on the 242 experimental curves, it performed similarly to the model without the added FT. The median absolute error (median percentage error) was 6.2 Å (8.9 %) for the film thickness, 2.3 Å (13.3 %) for the interface roughness and 0.76×10^{-6} Å^{-2} (7.2 %) for the SLD, which is 4 %, 19 % and 6 % higher, respectively, compared to the base model. From this, we can conclude that the base model (*i.e.* without the added FT) had likely already learned to implicitly extract all available frequency information from the data, and adding the Fourier components explicitly does not lead to a better training result. Furthermore, the fact that the results are slightly worse when the FT is added can be attributed to the increased number of trainable parameters due to the larger number of neurons in the model. This means more parameters need to be optimized to achieve the best training result, which is generally a more difficult task. For these reasons and the added computational requirements during both training and inference time, the FT preprocessing was not included the default NN the analysis pipeline of the *mlreflect* package. Nevertheless, this does not rule out that for certain scattering geometries, a suitable implementation of the FT could still be beneficial.

6.7. Implementation of *mlreflect* at P08/DESY

The work presented in this thesis was part of a project funded by the BMBF in close collaboration with DESY. One of the main goals of this project was the development of a publicly available software package and its eventual implementation

at DESY and specifically at the high-resolution diffraction beamline P08 at PETRA III [166]. To this end, the analysis package *mlreflect* described in this chapter has been installed on the DESY's Maxwell cluster with the help of André Rothkirch and Frank Schlünzen. The Maxwell cluster is a computational resource offered by DESY for photon science data analysis, GPU-accelerated computations (*e.g.* for artificial intelligence applications), high performance computing and scientific computing in general that can be used by any DESY user both on-site and off-site.

The *mlreflect* package is available as part of the the module repository of the Maxwell cluster and can be used easily with Jupyter notebooks as a graphical interface. Jupyter notebooks serve as an interactive, web-based interface for Python programs and are in wide-spread use within many fields of science. Since the Maxwell cluster has access to all data measured at DESY, *mlreflect* can be used to analyze data during the experiment directly at the beamline.

6.8. Conclusions from this chapter

This chapter discussed the optimized analysis pipeline *mlreflect*, which was developed for the automated analysis of reflectivity data using ML. The pipeline was tested on a large dataset of 242 XRR curves, containing *in situ* and *ex situ* measurements of organic thin films on Si/SiO_x substrates, where it showed a performance comparable to a manually supervised LMS fit for most of the data. Therefore, *mlreflect* is a useful tool for the automated pre-screening or even on-the-fly analysis of reflectivity data.

It was discussed that for the effective evaluation of trained ML models an experimental dataset of significant size is necessary. Most studies so far mainly focus on the performance of the model with regard to simulated data and include only few, if any, experimental test data. However, this may be misleading, since the results presented here clearly show that the performance on simulated data cannot easily be generalized to experimental conditions. This indicates the need for openly available

data repositories that provide large and varied training and testing data for the entire research community.

Furthermore, this chapter showed the influence of possible systematic errors (such as misalignment) that can be present in the data on the performance of the NN. The prediction speed of the NN model can be exploited to improve the overall performance by transforming the data slightly. These results highlight the necessity to account for these differences between simulated theoretical models and real data in order to obtain stable results. Again, the larger the variety of experimental data available for testing, the easier it is to train a NN that is robust towards these systematic errors.

Although the results shown here are demonstrated with systems of one layer on a Si/SiO_x substrate, the demonstrated NN model could easily be retrained to determine any single layer of any sample structure. While determining multiple layers at once is in principle possible and has been demonstrated before [104, 105], this type of NN architecture might not be ideal to tackle this type of inverse problem with multiple solutions since they map exactly one solution to a given input. Therefore, as discussed in Chap. 7, architectures that yield probabilities as an output might be more suitable for multi-layer problems.

7. Remarks on current limitations and future research

The idea of applying NNs to reflectivity data analysis has only been introduced three years ago. Yet, despite the remarkable achievements in this area, there remains room for improvement. As a guideline for future research, this chapter outlines current limitations of the method used in this work and offers possible paths forward.

7.1. Fully-connected vs. convolutional architectures

All NN models described in this work are fully-connected, meaning that each neuron in a given layer is connected to each neuron in the previous layer where the value of each connected neuron enters into a weighted sum, as described in Sec. 3.2. This sum treats the value of each neuron independently, *i.e.* the sequence of neurons in the previous layer is not explicitly considered. The input layer consists of a vector of reflectivity values $\boldsymbol{R}$ that are typically ordered by increasing q_z. This means that two neighboring values in $\boldsymbol{R}$ have an implicit physical relationship that is not considered in the NN architectures used in this work. While this relationship can be learned through training, other architectures might be more efficient. An alternative architecture could be convolutional neural networks (CNNs), which function similarly to convolutional filters in image processing and are specifically designed to extract spatial features from data [26]. Typically, CNNs deal with images, *i.e.* 2D data and they have been successfully used in the analysis of 2D scattering data [58, 60, 61]. However, there is no reason why this could not be applied to 1D reflectometry data as

well. In the case of reflectometry, a single curve can be treated as a 1D image and the convolutional kernels could be trained to extract certain features maps (gradients, edges, *etc.*) from the data, which can then be used to predict thin film parameters potentially more efficiently. Thus, it might be useful to explore this type NN in the future and compare its performance to current approaches.

7.2. Fixed input size and q dependence

As already mentioned in Chap. 4, an important limitation of the NN architectures used in this work is the fixed length of the input vector $\boldsymbol{R}$. This is coupled with the fact that the respective q_z values for each input are chosen at training time and must remain the same at inference time. In many experiments, however, the q_z range and the measured number of data points can vary strongly depending on experimental requirements. Furthermore, even if the number of data points fits the input size of the NN, since the q_z value for each input is implicitly determined during training, the NN cannot accept reflectivity values measured at different q_z. Thus, the input reflectivity has to be interpolated to the specific q_z values chosen during training. This method works if the data points are measured close to the trained q_z values compared to the length scale of the features in the data, such as the periodicity of the Kiessig fringes or the TRE drop-off. This, however, creates an experimental limitation on how many data points can be measured. A researcher may want to adjust the number of measured data points for various reasons, for instance, if a higher q_z resolution is necessary to resolve specific features (more data points) or if a higher time resolution is necessary to resolve *in situ* dynamics (less data points). Apart from the number of data points, the maximum q_z range is also fixed after training, which again limits flexibility during experiments.

A potentially straightforward way of solving this problem could be to include a second input vector $\boldsymbol{q}$ for the NN that contains the respective q_z for each reflectivity value. The network could then be trained with a larger variation of q_z values in the data, potentially leading to a more flexible model. Another solution was demon-

strated, in which a CNN was used to interpret images of reflectivity plots that were produced in a standardized fashion [104]. Each pixel of the image corresponds to an input for the NN, which means the input size is only determined by the chosen image size and not the number of measured data points *per se*. While it has been employed successfully in some cases, this method, seems subobtimal, since the majority of input values do not contain any information and are just filler points to create the 2D image. However, the idea behind this method could be adapted by reducing the dimensionality of the data, so that the input vector essentially represents a 1D image with one axis being q_z and the other axis being the reflected intensity, similar to the method proposed in this work.

7.3. Non-unique solutions

A very prominent and fundamental problem in reflectivity data analysis is that, due to the phase problem, the back-transformation from the scattering pattern to the sample structure does not always have a unique solution. In this work, it was assumed that the mapping of the reflectivity $\boldsymbol{R}$ to the parameters $\boldsymbol{p}$ can be approximated by the regression of a function $M(\boldsymbol{p}; \boldsymbol{w})$, as described in Sec. 3.1. In reality, however, this is not the case for the entire parameter space, and it is difficult to formally define the number of possible solutions of an arbitrary reflectivity curve. Appendix E shows an example of two simple box models that are very different, but produce mathematically identical scattering patterns (within kinematical approximation) due to the phase problem. However, even in cases where there are theoretically distinct solutions, they can be practically impossible to distinguish if there are a) no defining features, such as Kiessig fringes or the TRE or b) if the q_z range is too small. The latter case occurs if the curves only become significantly different for larger q_z, *e.g.* if the film thickness is small (wide oscillations) or the roughness is high (weak oscillations). An illustration of the ambiguity in the mapping $\boldsymbol{R} \rightarrow \boldsymbol{p}$ is shown in Fig. 7.1b. Although $\boldsymbol{p} \rightarrow \boldsymbol{R}$ is always unique and can be described as a function $m(\boldsymbol{p}) = \boldsymbol{R}$, the mapping $\boldsymbol{R} \rightarrow \boldsymbol{p}$ is only unique for certain parts of the parame-

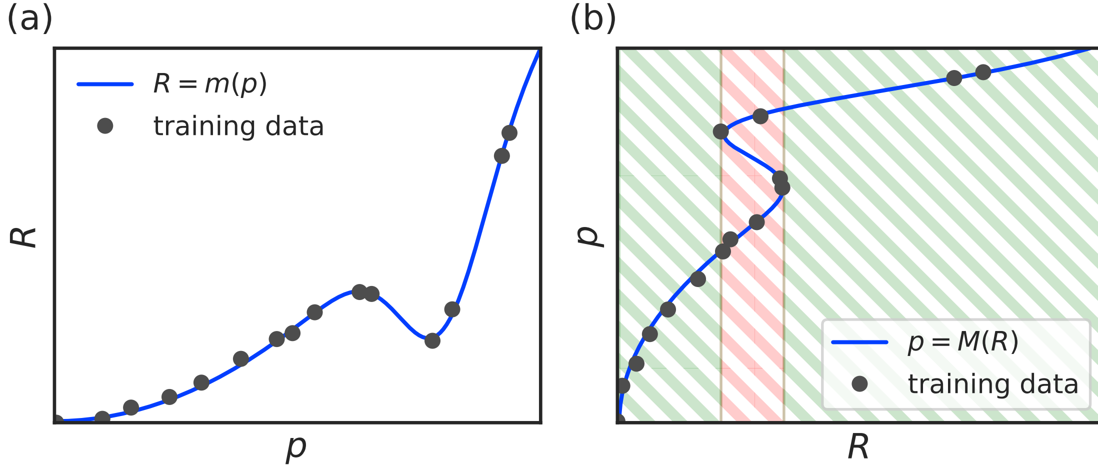

Figure 7.1.: Schematic demonstration of the non-reversibility of reflectivity data. (a) The transformation from the sample parameters $\boldsymbol{p}$ to the respective reflectivity curve $\boldsymbol{R}$ follows a function $m(\boldsymbol{p}) = \boldsymbol{R}$, *i.e.* each parameter set leads to only one reflectivity curve. The gray dots represent data points that are used during training to approximate the underlying function. (b) The inverse transformation from $\boldsymbol{R}$ to $\boldsymbol{p}$ is not unique. The green-shaded area represents segments that can be described as a function, where the NN is expected to work well. The red-shaded area represents a segment with multiple solutions, where the behavior of the NN is not well defined.

ter space. Thus, the implicit assumption within this work was that, by restricting the solution space (fixed substrate parameters and only 3 thin film parameters), the back-transformation becomes unique. Restricting the possible solutions is a way of adding *a priori* knowledge about the studied systems during training. This knowledge can be external information, such as what substrate was used or the number of deposited layers, but it can also come in the form of complementary measurements (see next section). Keeping too many parameters fixed during training, however, may lead to a NN model that is specialized for certain types of samples and it might be preferable to rather add prior information at inference time.

Moreover, if only little prior knowledge is available during training, it might not be possible to restrict the solution space to avoid multiple solutions. This can be problematic for a model that expects exactly one solution for a given input, since at inference time the model must pick one of the solutions and ignore the others. How-

ever, already during training, multiple solutions may lead to unstable gradients, since similar inputs can lead to drastically different outputs. This ultimately slows down training and may even prevent the NN from converging at all. As a result, the model might completely fail in cases of ambiguity, *i.e.* choose neither of the possible solutions. This might also explain the poor performance of the NN on certain challenging cases in Chap. 5. Thus, to tackle a larger variety of sample structures with potentially more ambiguity, *e.g.* multi-layers, different NN architectures should be considered. A potential solution for this could be NN architectures that do not predict a single parameter set $\boldsymbol{p}^*$, but instead a probability distribution of potential solutions. A well-known example of this are mixture density networks (MDNs), which predict a probability distribution constructed from superimposed m-dimensional normal distributions, where $\boldsymbol{p} \in \mathbb{R}^m$ [170, 171]. A similar implementation has already been demonstrated by others [109, 172] and may be a possible path forward.

7.4. Including *a priori* knowledge at inference time

Using the approach described in this work, the sample architecture and the possible solution space for each parameter is chosen during training and cannot be changed afterwards unless the model is retrained. This means training is, so far, the only opportunity at which *a priori* knowledge can be added to the system. However, additional information may be available at inference time, either through previous predictions of the NN or through information from complementary measurements. For example, when considering a NN trained to predict the thickness of two layers simultaneously, if the thickness of one of the layers becomes known at inference time, *i.e.* after training, it would be inefficient to predict both of them naively and risk a higher error. A possible solution would be to use the MDN architecture described in the previous section. If an MDN could be successfully trained to predict a probability distribution of solutions for a given input curve, additional *a priori* knowledge could be used to filter out improbable solutions in a post-processing step.

Another example of using complementary information at inference time are NR measurements of the same sample but at different levels of deuteration [161]. This can be used to obtain multiple datasets with different SLD contrasts, which increases the overall available information content. Other works have shown that multiple datasets can already be included during training, *e.g.* in the case for polarized NR [108]. A more flexible solution could be to construct a NN that takes two types of inputs: the reflectivity data of a single curve (as before) and additionally a prior probability of the sample parameters. Reflectivity data from complementary measurements could then be fed to the NN consecutively, each time using the previously estimated parameters as prior probabilities for the next prediction. This would allow the combination of an arbitrary number of complementary reflectivity curves, as well as allow the injection of other external information.

7.5. Experimental test and training sets

So far, virtually all of the ML applications in scattering physics have heavily relied on simulations to train the NN models. Using simulations has several advantages: 1) It makes use of the mathematical descriptions which have been derived through many decades of rigorous research, 2) an almost infinite amount of training data can be generated and 3) the ground truth is already known with infinite accuracy and no manual "labeling" of the data is necessary. However, a major drawback of simulations is that they are based on idealized models that do not perfectly capture every detail of experimental data. The added noise and other methods presented in this work try to remedy this to some degree, however, it is quite challenging, if not impossible, to account for all deviations of the data from the theory in a formalized way.

One of the major reasons for the success of deep learning in recent decades was the large amount of available data for training. This refers to large datasets of "real" data, such as images of objects that were carefully labeled manually. Importantly, this real data also contains real noise and artifacts, which allow the network to

converge to a more generalized model than through idealized simulations. Thus, in the long-term it might be beneficial for the scattering community to build a large pool of labeled, *i.e.* manually analyzed, experimental data that can be used not only for testing, but also for training. This is clearly a challenging task, since, in contrast to labeling images, the analysis requires specialized labor from decentralized research groups, which need to share their results in a standardization fashion.

Even if the use of experimental data for training might be infeasible in the short-term, its use for testing the NN performance is still crucial. Thus, a database of reflectivity data open to researchers would be advantageous for standardized testing and comparisons between different published models. Fortunately, there has been a recent increase of national and international efforts aimed at improving the data infrastructure from which future ML endeavors may profit, such as the DAPHNE4NFDI consortium in Germany[1]. In this context, databases to store analyzed scattering data for training and testing (including reflectivity data) are being developed [173]. For example, the test data used in Chap. 6 was uploaded to an openly-accesible online repository at `https://doi.org/10.5281/zenodo.6497438`.

7.6. Training optimization

In general, solving the regression task of finding the correct minimum of the loss function as described in Chap. 3 is difficult and the optimal training parameters are usually determined heuristically. These are called hyperparameters, since they are themselves not adjusted during training and include the size and number of hidden layers, the learning rate, the optimizer, the loss function, the training set size, the minibatch size, the applied noise level and the application of other regularization techniques. Since it is difficult to estimate *a priori* how to adjust these parameters, it is common in the field of ML to perform a rigorous and systematic hyperparameter optimization. For the research in this work, this approach was not deemed efficient, since the general architecture still had to be determined and performing a hyperpa-

[1] `https://www.daphne4nfdi.de`

rameter grid search for each possible architecture was beyond the scope of this thesis. Thus, the number and size of the hidden layers, the learning rate and the effectiveness of regularization techniques were determined mainly through manual trial and error. However, with the foundation of a working model now established, further effort could be spent in systematically tuning certain hyperparameters to achieve the best possible performance. One such attempt was already shown in Chap. 6, where different models were trained with varying degrees of training noise to find the optimal noise level. A similar approach could be taken for other hyperparameters, ideally with the use of large-scale computational clusters. It is important to note, however, that any optimization should always be performed with regards to the performance on experimental data, since the optimal hyperparameters for simulated data may differ considerably.

8. Conclusion and outlook

8.1. Summary and conclusion

The research in this work shows how FCNNs can reliably be employed to analyze specular reflectivity data of OSC thin films on Si/SiO$_x$ substrates. However, the method is not limited to these materials or sample structures, as has been demonstrated before [99, 174]. It was shown in Chap. 4 that, even on moderate CPUs, the analysis speed of the NN itself for a single reflectivity curve can be as fast as 0.03 ms. When embedded into the *mlreflect* package shown in Chap. 6, the time per curve for the entire analysis pipeline, *i.e.* including post-processing steps, was 100 ms. This is much faster than conventional LMS fits using differential evolution, which often take several minutes and up to hours in addition to requiring focused human attention. This means that the *mlreflect* pipeline is on par with current high-end measurement speeds, which can also be as fast as 100 ms per curve [7, 8]. Importantly, the post-processing steps of the pipeline, such as sampling different q_z shifts, have not yet been optimized in terms of computation time and are generally much slower than the NN itself. Thus, it is likely that by optimizing operations in these steps, the overall analysis time of the pipeline could be further reduced to the range of milliseconds per curve.

Reducing the analysis time is important, since as more advanced and data-rich measurement techniques become available, analysis methods have to become equally as fast to avoid bottlenecks and a build-up of data that remains not analyzed. Obtaining results from their measurements more quickly, even preliminary ones, helps researchers save time and resources. This is especially important for experiments

at large-scale facilities, such as synchrotron or neutron sources, as the measurement time there is very valuable and usually limited. NNs can be trained in advance based on the sample architecture of planned measurements and can then be used to inform the researchers' decision-making on-site, thereby leaving more time for experiments. Additionally, the NN prediction pipeline could be integrated into the measurement software to enable on-line feedback mechanisms based on parameters derived from the data, such as adjusting the heating of the effusion cell during *in situ* film growth to control the growth rate.

Chap. 5 showed that, within the trained parameter space, there are several challenging cases where the demonstrated NN performed poorly and these can even occur in reflectivity data of simple sample architectures, *e.g.* 1–2 layers on a substrate. This is especially the case for "smooth" curves without pronounced features, such as Kiessig fringes or the TRE. The information content in the data also depends on the measured q_z range, which may be limited for various reasons. One reason can be Bragg reflections and Laue oscillations which for the molecular thin films used in this work occur at relatively low q_z ($\approx 0.3\,\text{Å}^{-1}$). However, the maximum q_z may also be limited by the available dynamic range of the measurement. Ambiguity in the data during training can lead to unstable gradients that hinder convergence to a better minimum. In Chap. 5, removing ambiguous cases from the training set has shown to help the performance of the NN also on "easier" reflectivity curves with a higher information content.

Chap. 6 discussed the necessity of adding noise to the simulated training data to avoid overfitting and improve performance on experimental data. While it is not surprising that noise on the training data can act as a regularizer, it is difficult to quantify how much noise should be added. This work demonstrated a method of systematically testing different noise levels during training to find the optimum. The exact type and level of noise may depend on specifics of the data one wants to analyze. For example, in NR measurements of surfactant molecules, the dominant type of noise can be the Poisson noise of the counting statistics [81]. While the general methodology of finding the optimal amount of training noise is expected to

work also for other noise types, one needs a sufficiently large set of experimental test data as a reference.

In addition to statistical errors, Chap. 6 introduced a post-processing method to deal with certain systematic errors, specifically small q_z shifts. The solution offered in this work utilizes the fast prediction times of the NN to re-sample the input data with small systematic perturbations and then picks the best result among the whole ensemble of predictions. This method works because the MSE between the measured curve and the predicted curve can be used to estimate the loss, *i.e.* the distance of the predicted parameters from the ground truth parameters. Thus, this method acts very similarly to a conventional LMS fitting except that the sampling of the parameter space is not done *via* an iterative algorithm but through the NN predictions, which is much more efficient.

Importantly, all of the above-mentioned advances in applying ML to reflectivity data analysis have been implemented in the open source Python package *mlreflect*. The succesful development and delivery of *mlreflect* was also part of a project funded by the BMBF in collaboration with DESY. The package is available both online (see Appendix D) as well as on the Maxwell cluster of DESY[1], which is accessible to most DESY users. In addition to the source code, there are also a full API documentation and brief online tutorials on how to use *mlreflect* to determine sample parameters or train new NN models for other sample architectures. As a result, *mlreflect* is not only a tool that can be used for the fast analysis of specular reflectometry data, but it might also serve as an example for others in the X-ray and neutron scattering communities that wish to develop other ML-based tools for scattering data analysis.

In conclusion, the ML methods described in this work, including the *mlreflect* package, constitute a significant step towards the ultimate goal of fully-automatized reflectivity data analysis.

[1] https://confluence.desy.de/display/MXW/

8.2. Outlook

The following gives a brief overview of possible further ML applications for reflectometry data analysis and other scattering methods. Within the field of reflectometry, applications beyond the research presented in this work might entail, among others:

- Estimating posterior distributions of individual sample parameters or entire SLD profiles [106], *e.g.* through MDNs [109] or neural posterior estimation [175] (especially relevant for more complex samples, such as multi-layers)
- Reducing ambiguity in data by co-refining multiple reflectivity curves, *e.g.* obtained from tunable reference layers [83], polarized NR [108] or time-correlated real-time data sets with specific growth models [176]
- Removing the requirement of providing reflectivity data measured at fixed q_z values, *e.g.* by providing the q_z values for each reflectivity input explicitly during training
- Employing CNN architectures that may be able to extract information from the data more efficiently, if optimized correctly
- Denoising of reflectometry data to decrease exposure time, *e.g.* through variational autoencoders [112]
- Integrating ML solutions like *mlreflect* into existing analysis software for reflectometry, such as *GenX* [96]

For other scattering methods, ML can in principle also be used to extract sample parameters that are otherwise obtained *via* conventional analysis methods. However, there are also other potential applications that can help process large data volumes or improve the quality of measurements, such as:

- Predicting other material properties based on structural information, *e.g.* charge transfer in organic thin films [39]

- Real-time tracking of diffraction features, *e.g.* Bragg peak position and width in grazing-incidence wide-angle measurements [61]
- Classification/pre-screening of the data, *e.g.* by particle shape in the case of small-angle scattering [54, 60]
- Detecting anomalies in the data, *e.g.* unknown phases in crystallographic data
- Data reduction/compression of large datasets, *e.g.* using autoencoders
- Enhanced stabilization of source properties, *e.g.* at synchrotrons [177] or XFELs [65], where an increased beam coherence is beneficial for methods such as X-ray photon correlation spectroscopy [178] or ptychography [179]

The above-mentioned methods may help alleviate current and future bottlenecks in scattering data analysis or may enable entirely new types of experiments. However, it is important to note that most ML-related research for scattering physics has mainly been conducted within the last 8 years and, given the complexity of the task, the field is arguably still in its infancy. Thus, as the field matures within the coming years and decades, even more sophisticated and varied ML solutions are likely to emerge.

Part III.

Appendix

A. Additional figures for Chapter 4

This chapter contains additional figures of datasets that were used to calculate the predictions accuracy values in Chap. 4, but were not explicitly shown there. Fig. A.1–A.4 demonstrate the prediction performance of the NN on four *in situ* datasets: two DIP films, a CuPc film and a 6T film. In general, the NN performs worse on XRR curves that have less pronounced features, such as low thickness or high roughnesses. These curves are marked with shaded areas and were not included in the accuracy statistics.

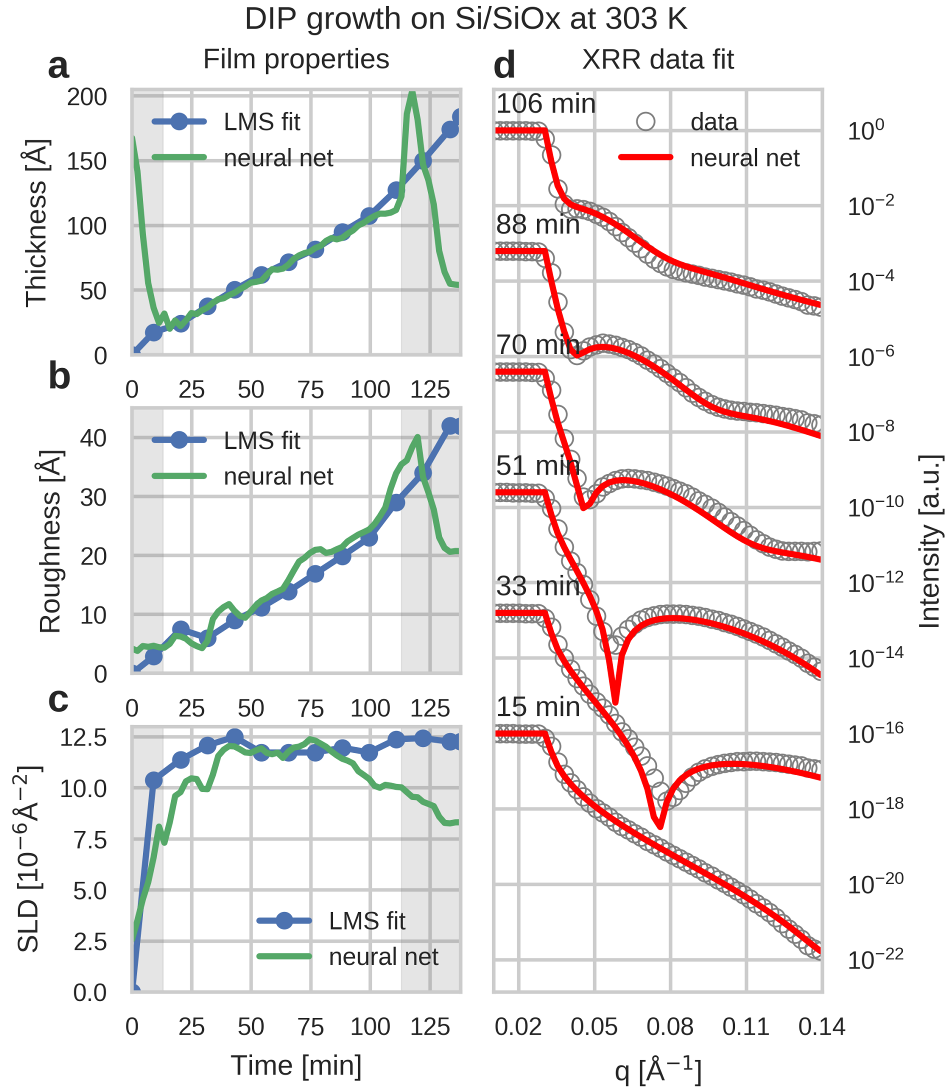

Figure A.1.: Fitting performance of the NN model on a DIP film grown at 303 K with a deposition rate of 1.3 Å min^{-1}. (a-c) Comparison of the film parameters predicted by the NN with results from least mean square fitting with human supervision at different times during growth. The shaded area marks films with low thickness (below 20 Å) or high roughness (above 30 Å) where data is difficult to fit for the NN. (d) Overlay of the experimental data with data simulated using the parameters predicted by the NN during different times during growth. Figure adopted from [103].

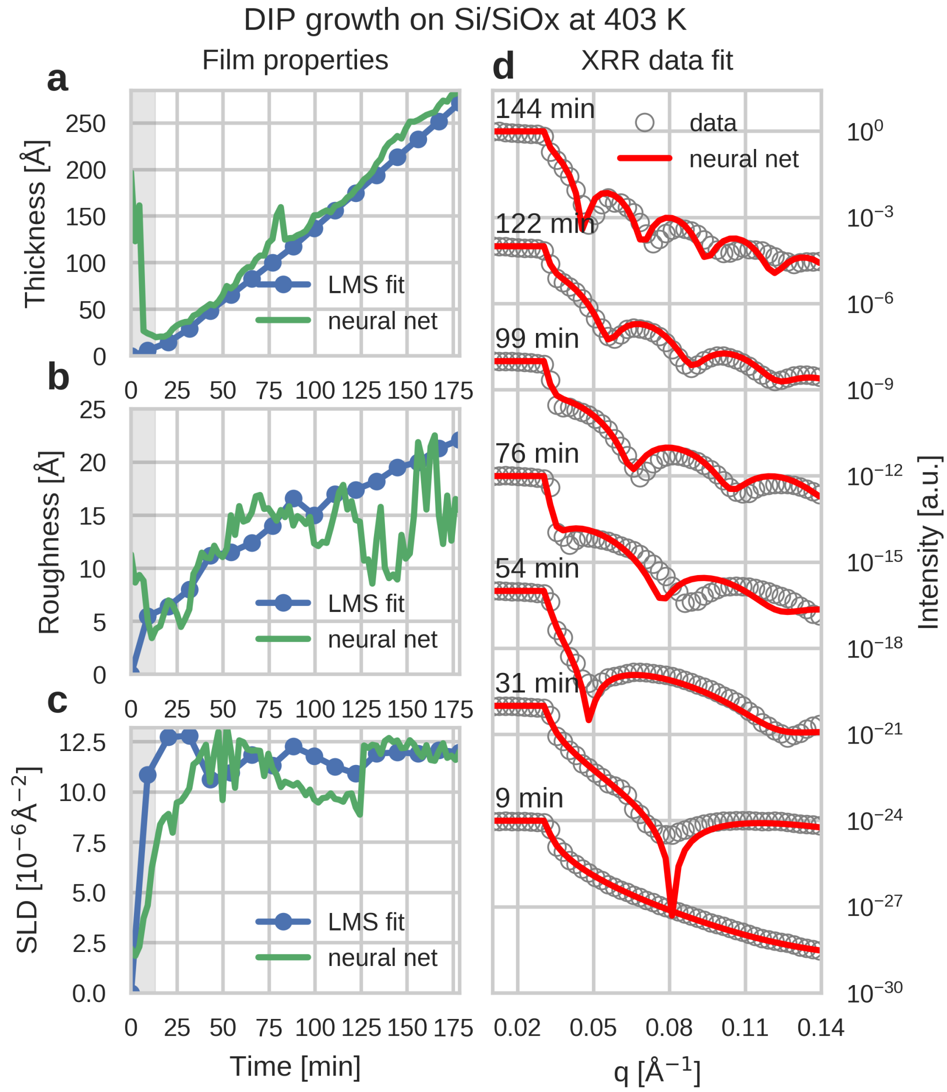

Figure A.2.: Fitting performance of the NN model on a DIP film grown at 403 K with a deposition rate of 1 Å min^{-1}. (a-c) Comparison of the film parameters predicted by the NN with results from least mean square fitting with human supervision at different times during growth. The shaded area marks films with low thickness (below 20 Å) where the data is difficult to fit for the NN. (d) Overlay of the experimental data with data simulated using the parameters predicted by the NN during different times during growth. Figure adopted from [103].

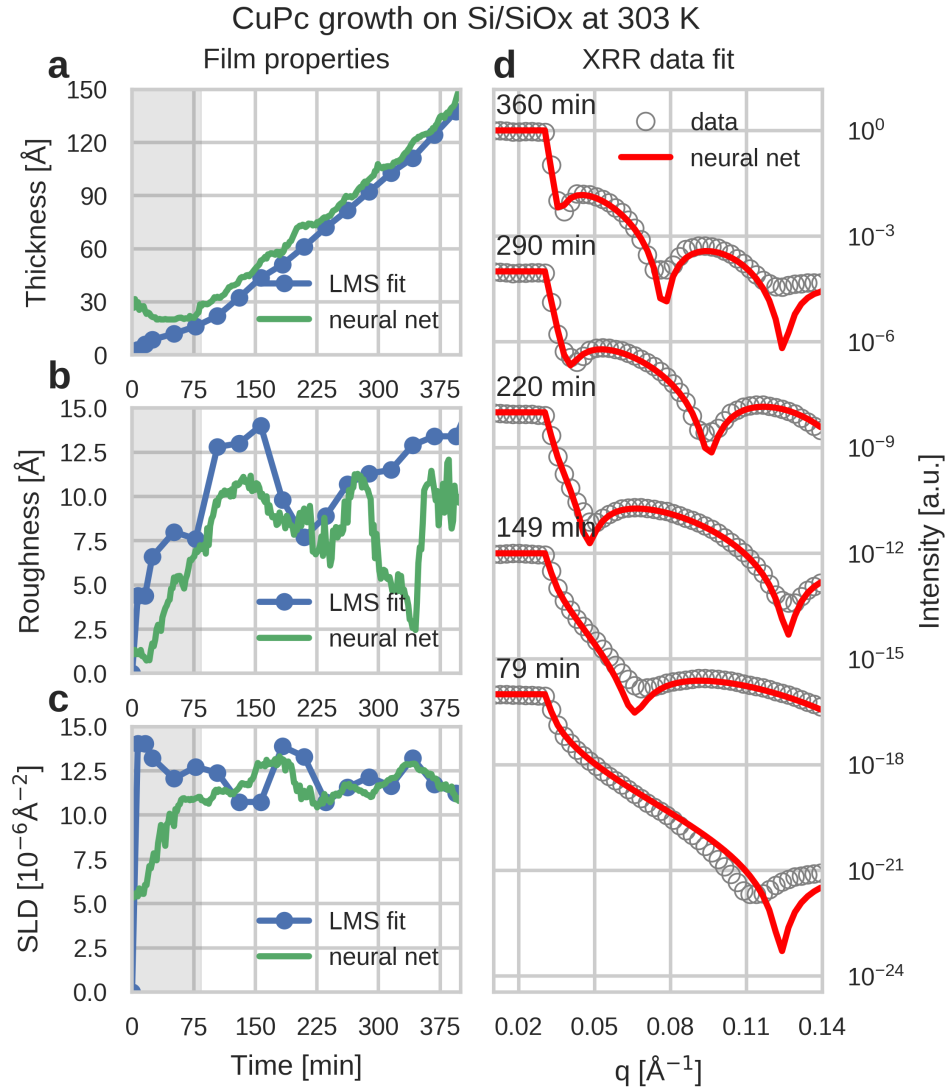

Figure A.3.: Fitting performance of the NN model on a CuPc film grown at 303 K with a deposition rate of 0.4 Å min^{-1}. (a-c) Comparison of the film parameters predicted by the NN with results from least mean square fitting with human supervision at different times during growth. The shaded area marks films with low thickness (below 20 Å) where the data is difficult to fit for the NN. (d) Overlay of the experimental data with data simulated using the parameters predicted by the NN during different times during growth. Figure adopted from [103].

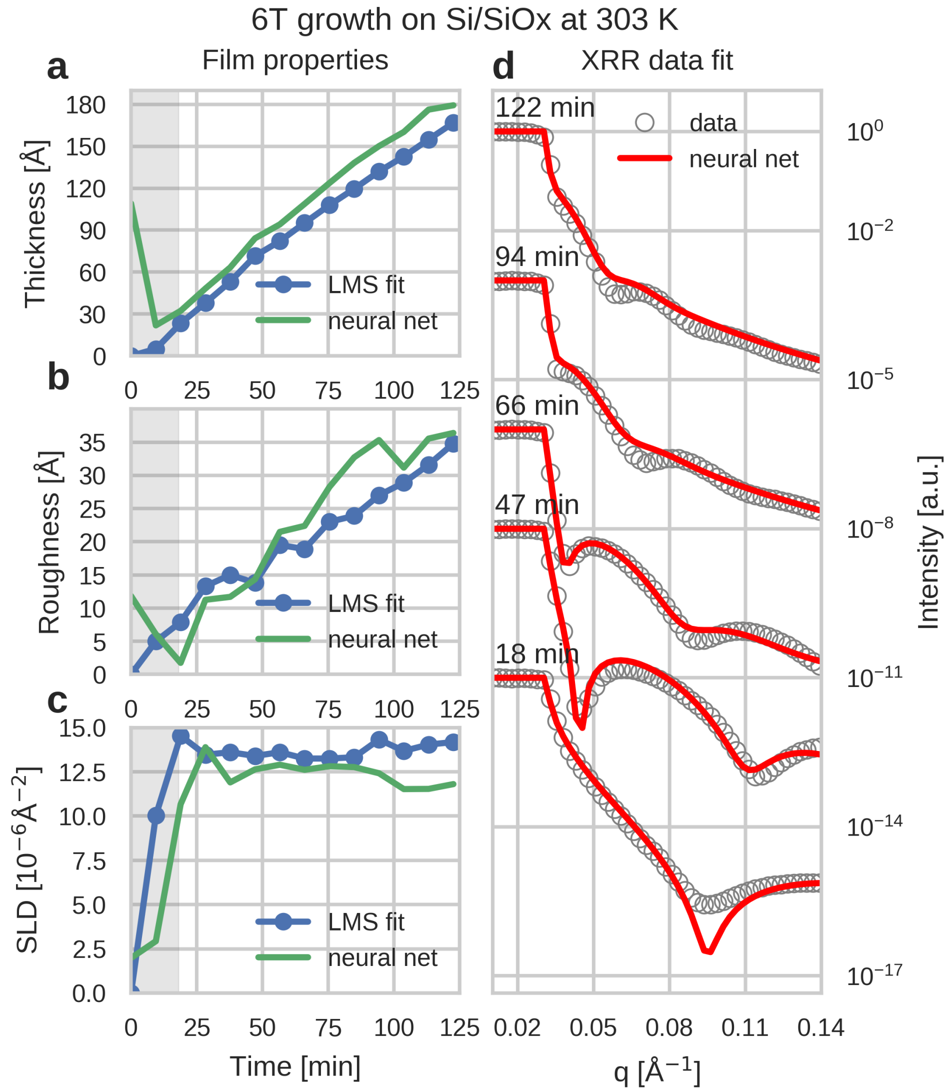

Figure A.4.: Fitting performance of the NN model on a 6T film grown at 303 K with a deposition rate of 1.3 Å min^{-1}. (a-c) Comparison of the film parameters predicted by the NN with results from least mean square fitting with human supervision at different times during growth. The shaded area marks films with low thickness (below 20 Å) where the data is difficult to fit for the NN. (d) Overlay of the experimental data with data simulated using the parameters predicted by the NN during different times during growth. Figure adopted from [103].

B. Additional figures for Chapter 5

This chapter contains additional figures regarding the performance evaluation discussed in Chap. 5. Fig. B.1–B.5 show the accuracy of each thin film parameter with regard to each of the four discussed curve modifications: curve scaling, uniform noise, Poisson noise and background.

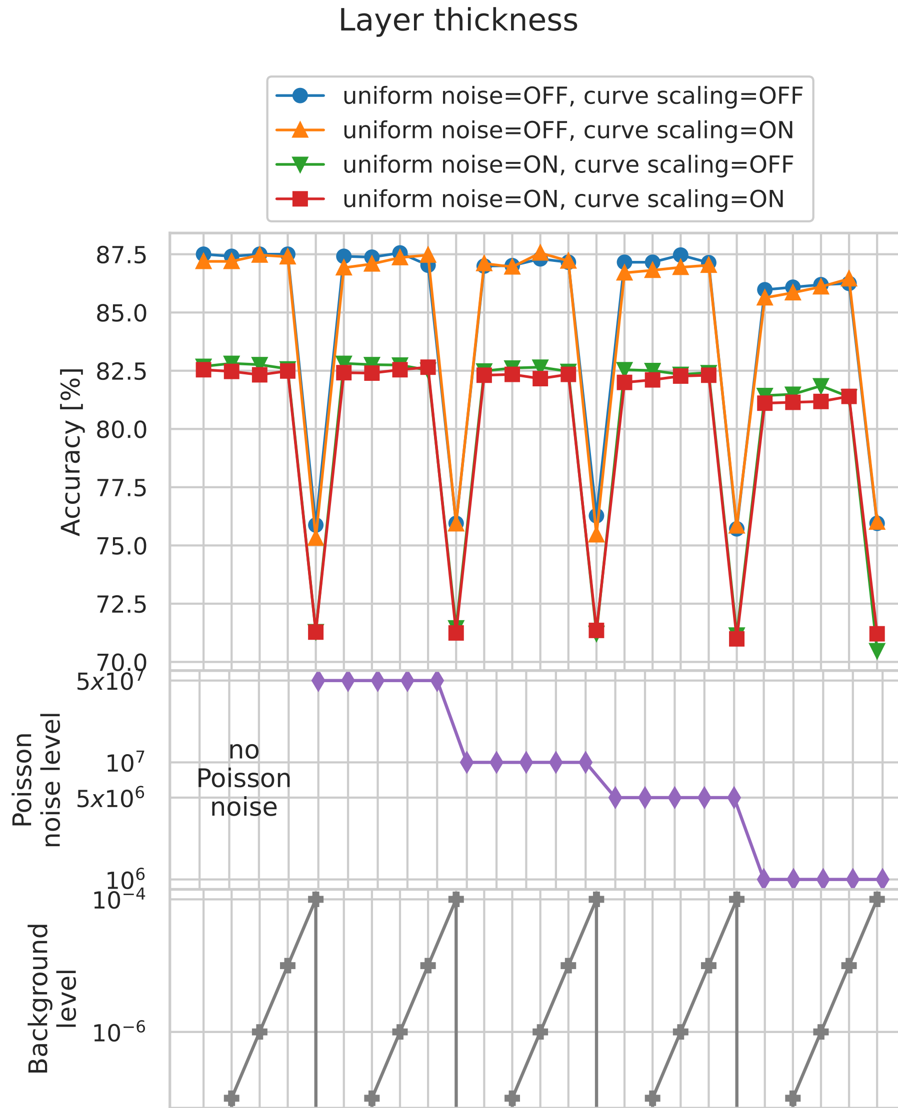

Figure B.1.: Prediction accuracy of the layer thickness for 100 different noise and background combinations. Each point in the top panel refers to the prediction accuracy of test curves with a specific noise and background combination. The four different colors/symbols distinguish between the four binary combinations of uniform noise and curve scaling being turned on or off, respectively. The horizontal axis distinguishes between the 25 combinations of Poisson noise and background levels. The combination of each noise level and background level can be read from the bottom two plots. The gray line (crosses) indicates the added background b for a given point along the horizontal axis, with the background periodically increasing from left to right. Similarly, the purple line (diamonds) indicates the incident intensity s that was used to calculate the Poisson noise. Since lower intensities mean higher relative noise, the noise added to the curves increases from left to right. Figure adopted from [107].

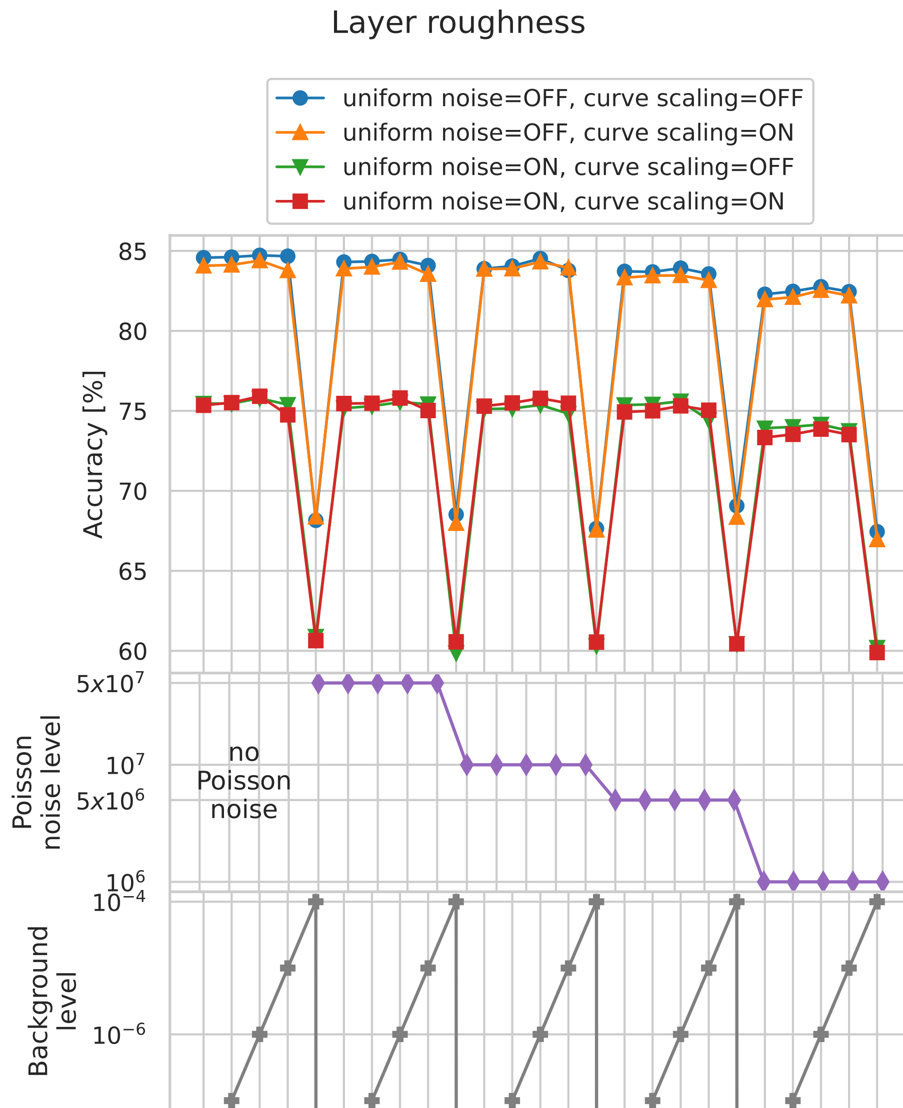

Figure B.2.: Prediction accuracy of the layer roughness for 100 different noise and background combinations. For further description of the plot see Fig. B.1. Figure adopted from [107].

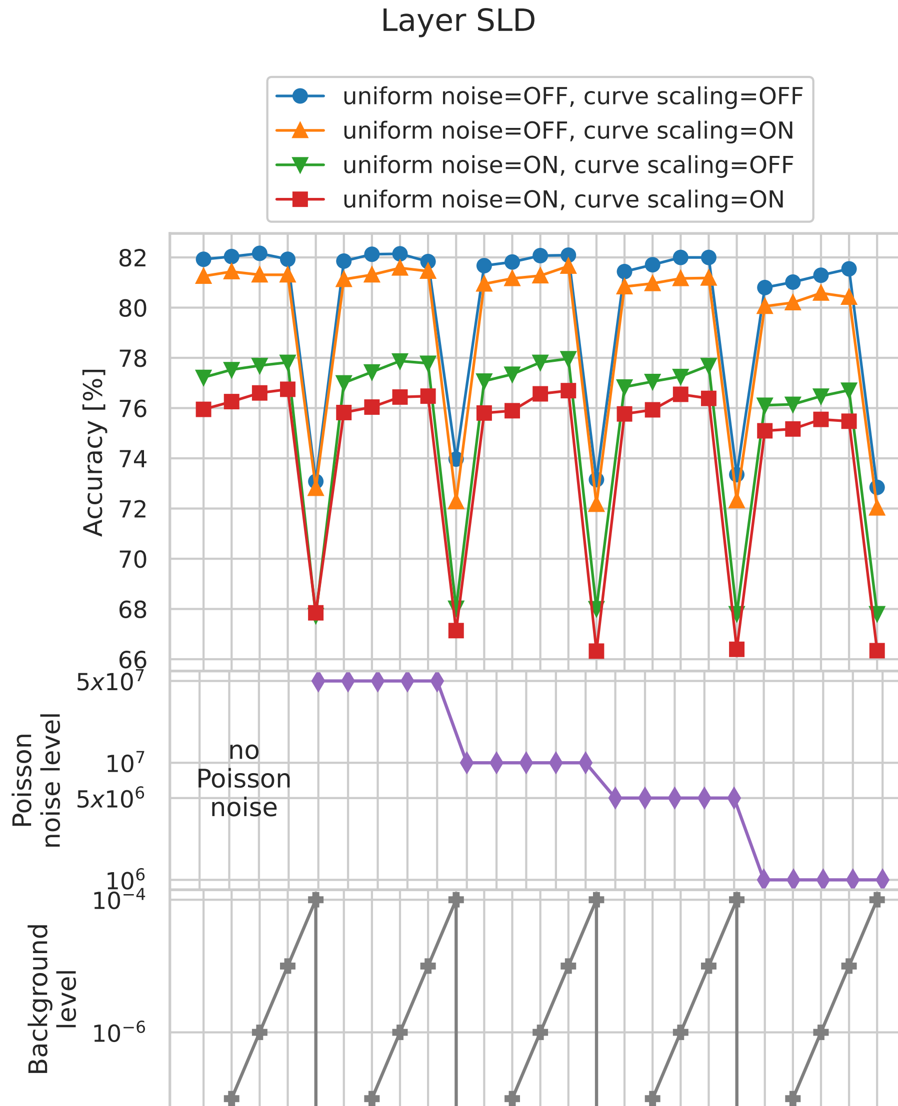

Figure B.3.: Prediction accuracy of the layer SLD for 100 different noise and background combinations. For further description of the plot see Fig. B.1. Figure adopted from [107].

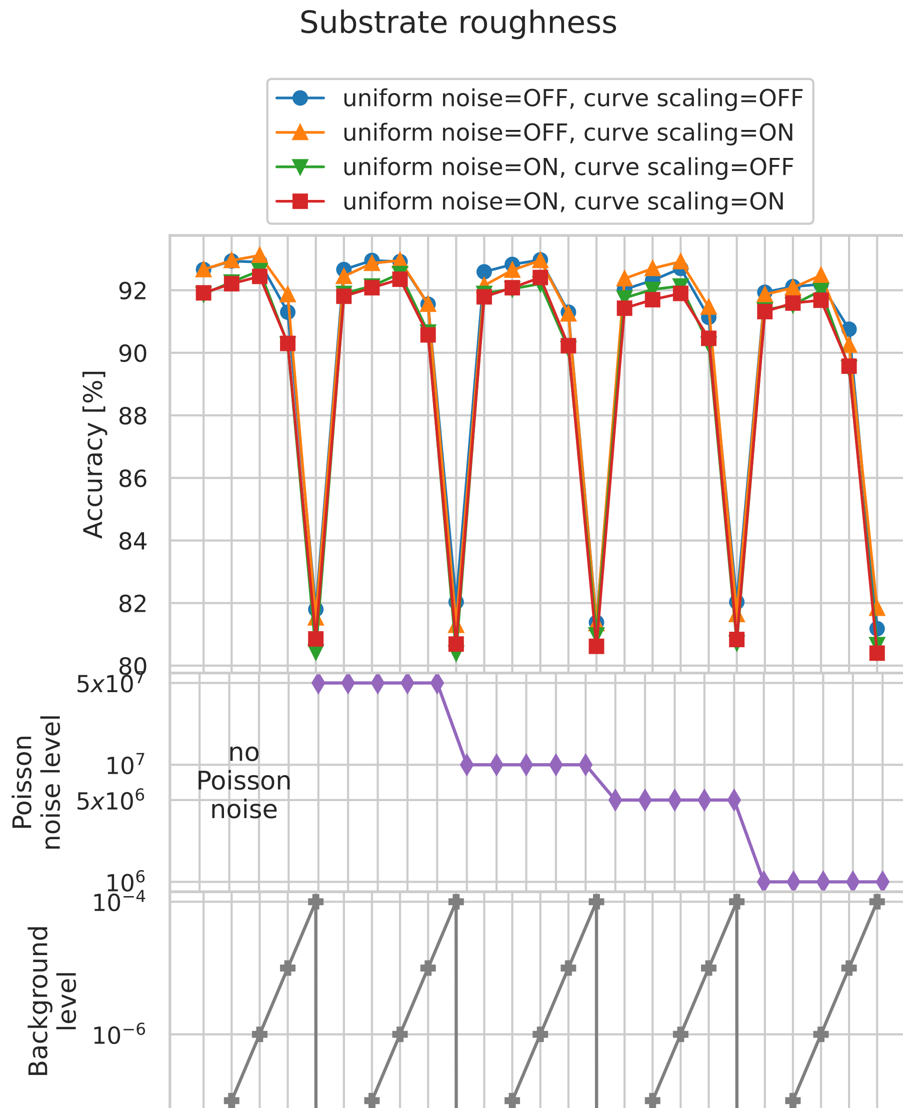

Figure B.4.: Prediction accuracy of the substrate roughness for 100 different noise and background combinations. For further description of the plot see Fig. B.1. Figure adopted from [107].

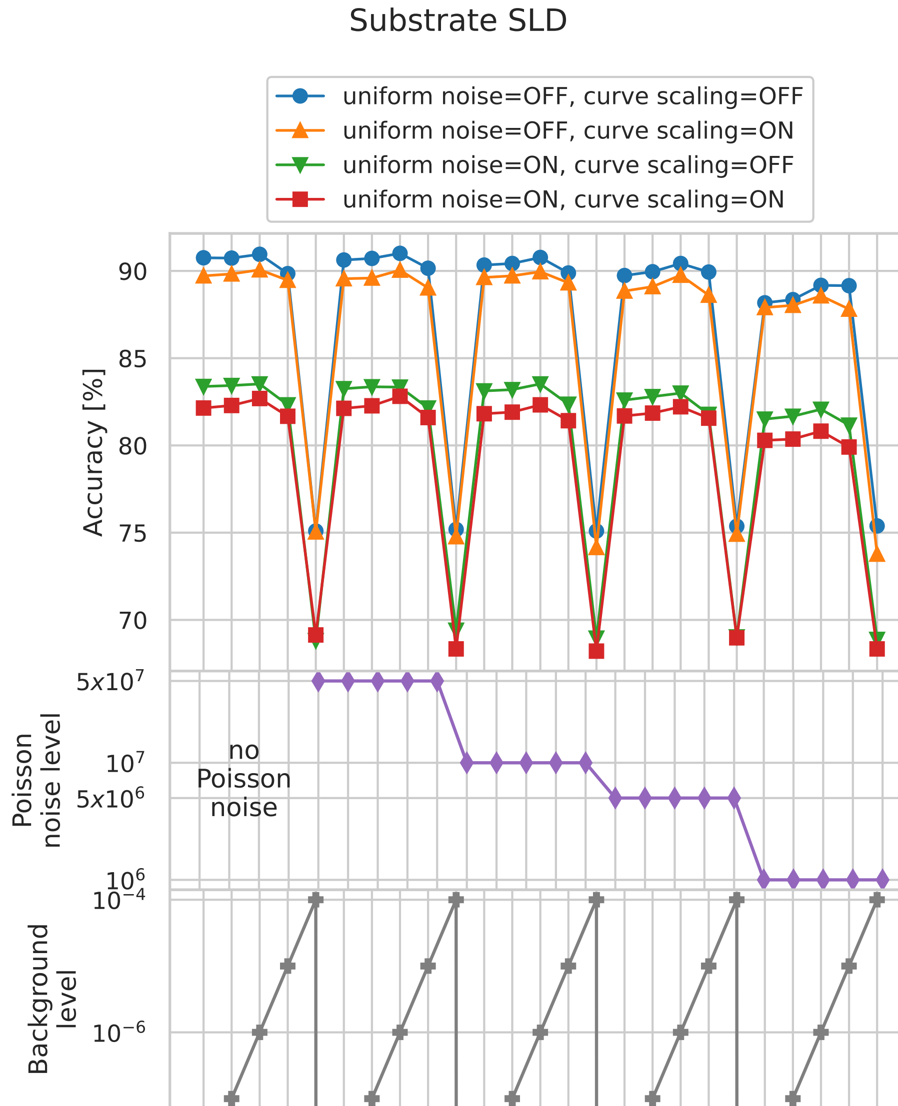

Figure B.5.: Prediction accuracy of the substrate SLD for 100 different noise and background combinations. For further description of the plot see Fig. B.1. Figure adopted from [107].

C. Additional figures for Chapter 6

This chapter contains detailed histograms of the absolute error distribution from the *mlreflect* analysis pipeline described in Chap. 6 for each parameter. The histograms expand on the condensed form shown in Fig. 6.3 and show the errors for the full pipeline (neural network prediction + q_z shift + LMS fit) with respect to each GT parameter. Fig. C.1–C.3 show the errors with respect to the GT thickness, Fig. C.4–C.6 with respect to the GT roughness and Fig. C.7–C.9 with respect to the GT SLD.

The majority of outliers are due to ambiguous fits (*e.g.* from “featureless” curves) where multiple parameter combinations lead to a good fit. A common case are very thin films where there are no oscillations visible in the chosen q_z range.

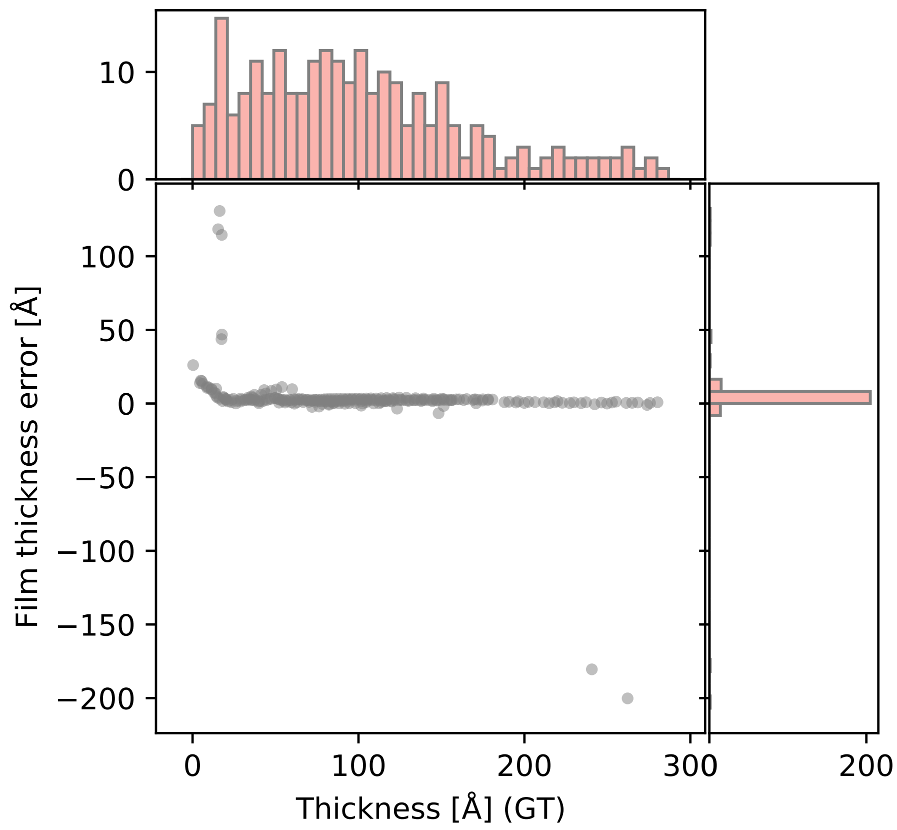

Figure C.1.: Distribution of the absolute thickness error from the full pipeline fit with respect to the ground truth (GT) thickness. Each dot represents a single curve in the testing dataset. Figure adopted from [111].

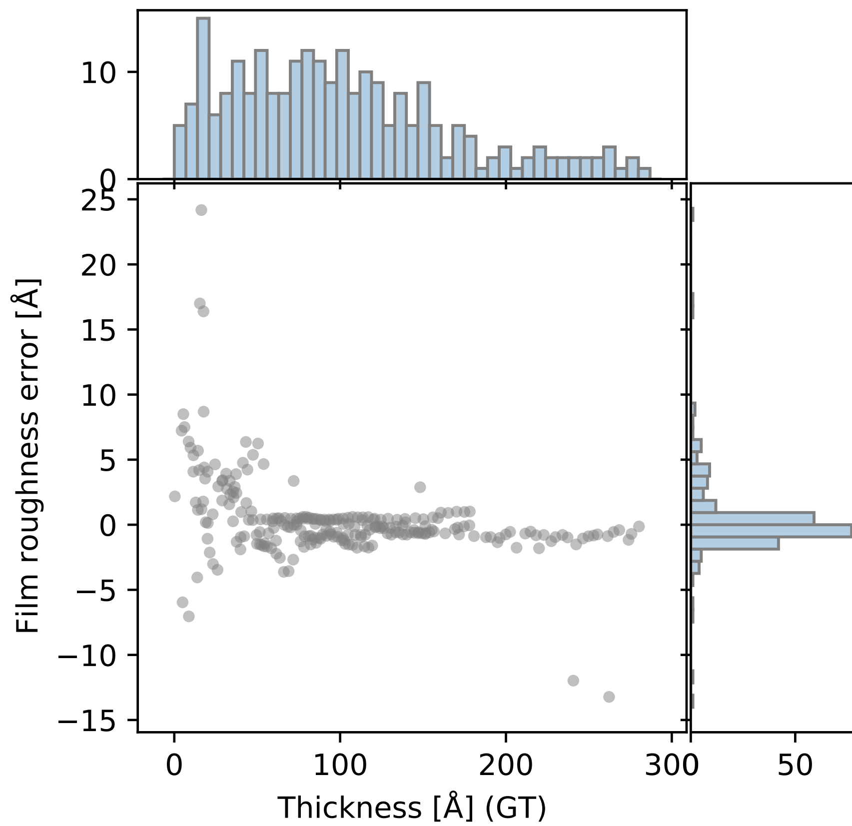

Figure C.2.: Distribution of the absolute roughness error from the full pipeline fit with respect to the ground truth (GT) thickness. Each dot represents a single curve in the testing dataset. Figure adopted from [111].

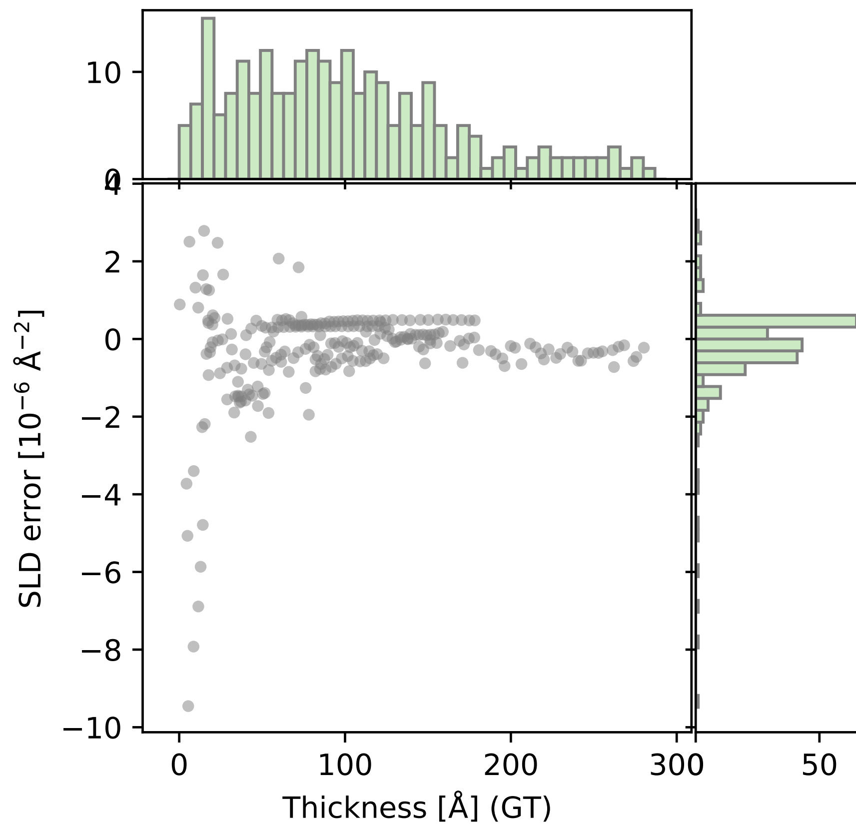

Figure C.3.: Distribution of the absolute SLD error from the full pipeline fit with respect to the ground truth (GT) thickness. Each dot represents a single curve in the testing dataset. Figure adopted from [111].

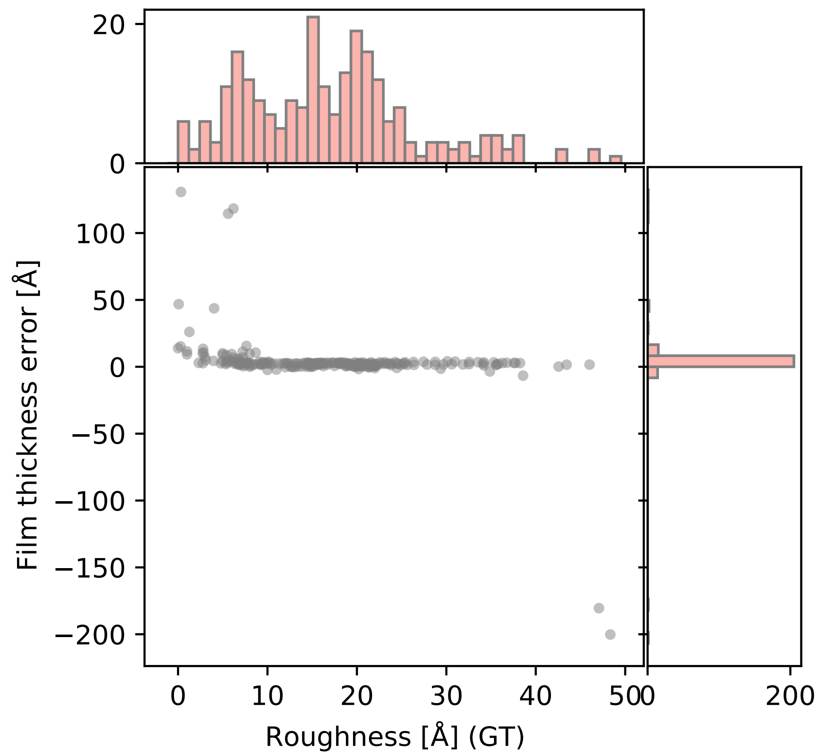

Figure C.4.: Distribution of the absolute thickness error from the full pipeline fit with respect to the ground truth (GT) roughness. Each dot represents a single curve in the testing dataset. Figure adopted from [111].

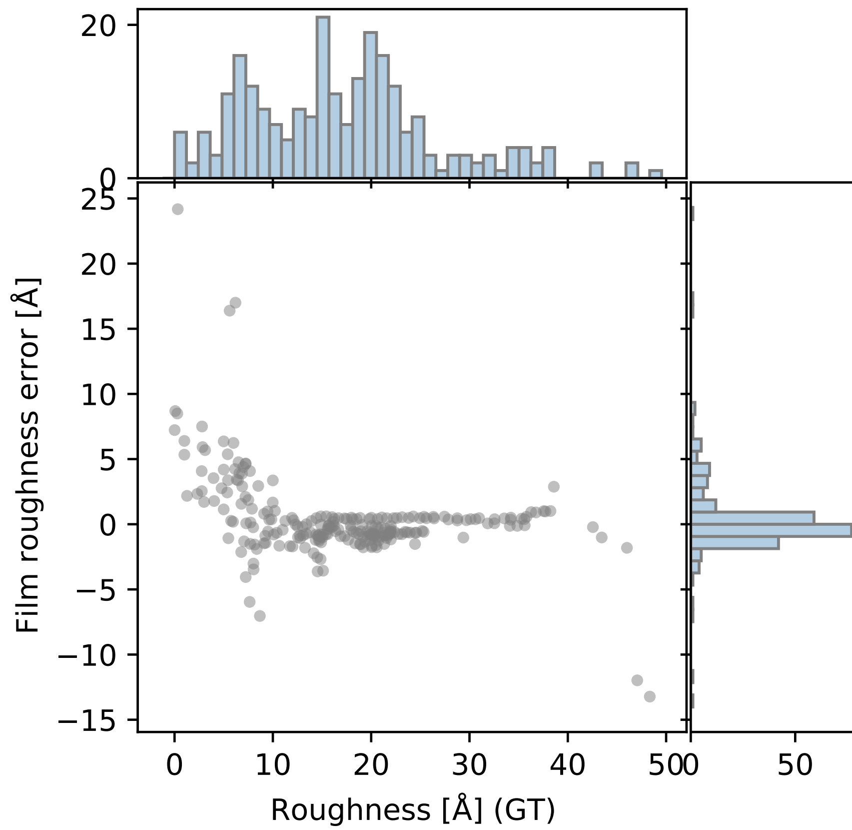

Figure C.5.: Distribution of the absolute roughness error from the full pipeline fit with respect to the ground truth (GT) roughness. Each dot represents a single curve in the testing dataset. Figure adopted from [111].

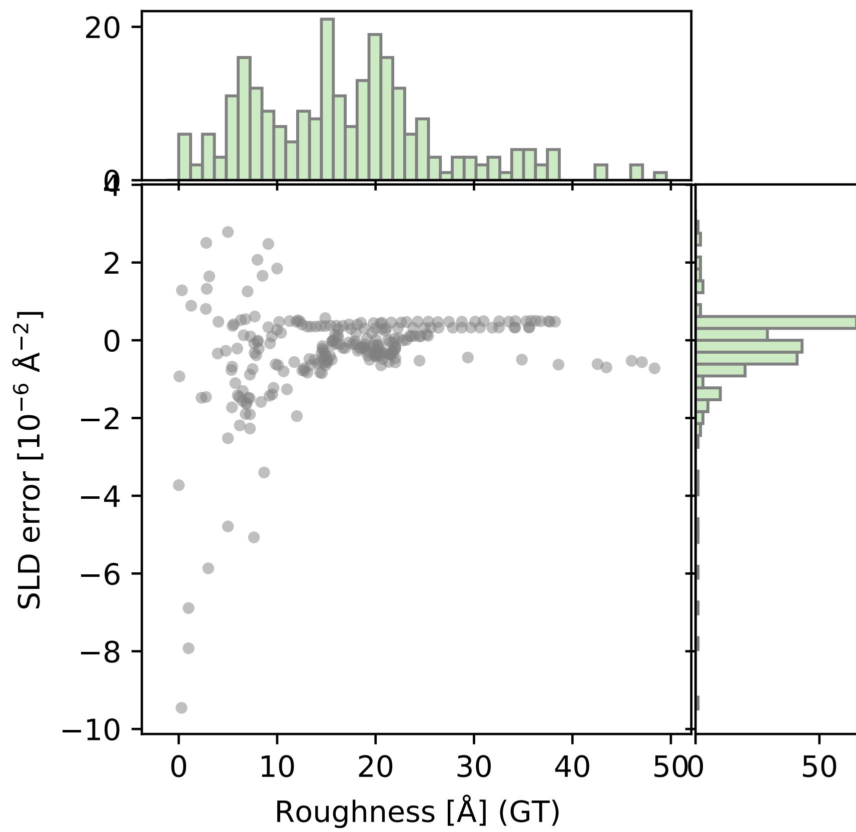

Figure C.6.: Distribution of the absolute SLD error from the full pipeline fit with respect to the ground truth (GT) roughness. Each dot represents a single curve in the testing dataset. Figure adopted from [111].

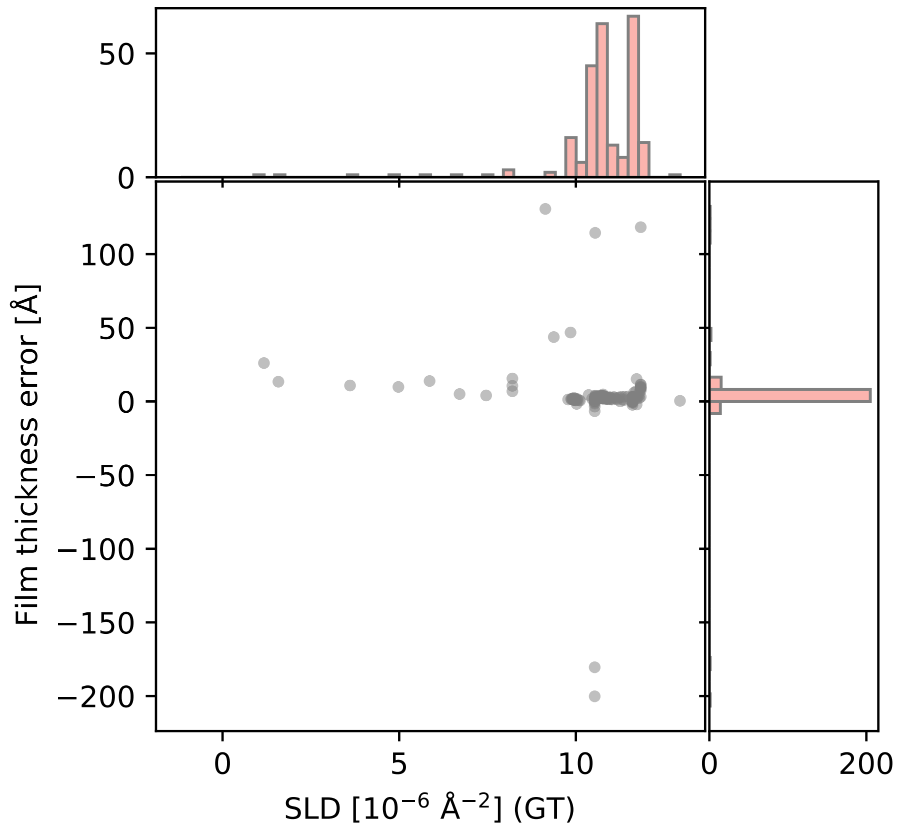

Figure C.7.: Distribution of the absolute thickness error from the full pipeline fit with respect to the ground truth (GT) SLD. Each dot represents a single curve in the testing dataset. Figure adopted from [111].

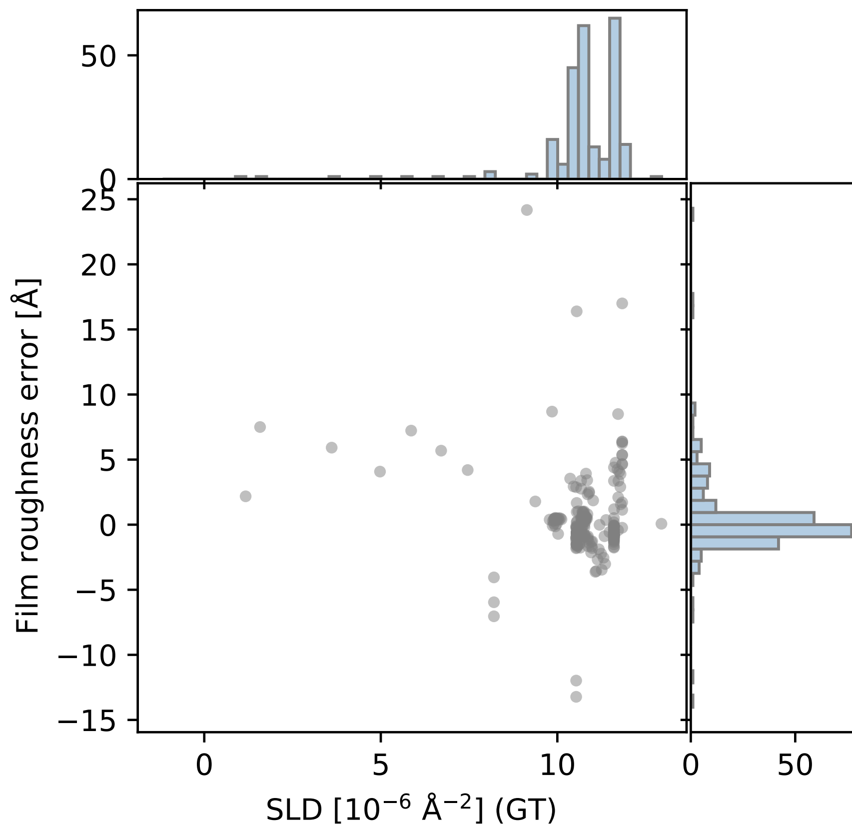

Figure C.8.: Distribution of the absolute roughness error from the full pipeline fit with respect to the ground truth (GT) SLD. Each dot represents a single curve in the testing dataset. Figure adopted from [111].

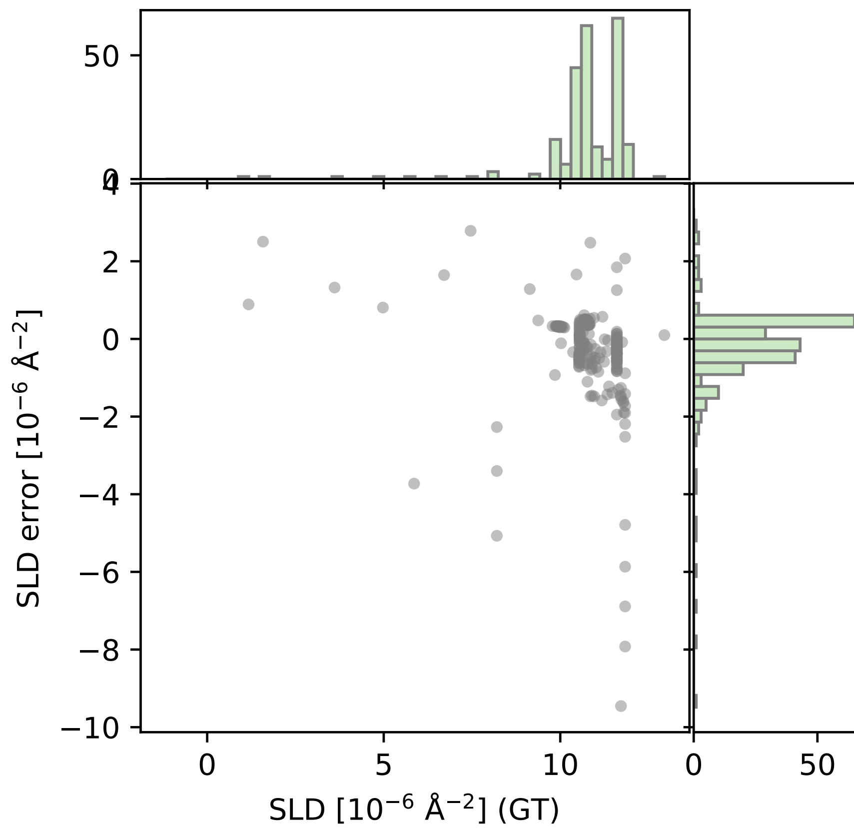

Figure C.9.: Distribution of the absolute SLD error from the full pipeline fit with respect to the ground truth (GT) SLD. Each dot represents a single curve in the testing dataset. Figure adopted from [111].

D. Availability of the *mlreflect* package

D.1. Download from PyPI

The *mlreflect* package can be found on the PyPI under https://pypi.org/project/mlreflect. PyPI is the most used repository of Python packages and allows the download and installation of packages from any machine with access to the internet free of charge. *mlreflect* can be downloaded from PyPI from any command line interface (*e.g.* Linux bash or Windows Powershell) with the command:

```
python -m pip install mlreflect
```

This is the recommended way to install *mlreflect* for users who just want to use the package for their data analysis and are not primarily interested in the source code.

D.2. Web documentation on Read the Docs

A complete web documentation of *mlreflect* is hosted publicly on the Read the Docs website under https://mlreflect.readthedocs.io/en/latest/index.html. This includes examples on how to use the default NN model described in Chap. 6 as well as an example on how to train a new model, *e.g.* for different substrates. It also includes a full application programming interface (API) documentation based on the Python docstrings.

D.3. Open source on GitHub

The full source code of the *mlreflect* package can be reviewed and downloaded on GitHub under `https://github.com/schreiber-lab/mlreflect`. GitHub is a widely-used platform that allows the public sharing of code. As per the requirements of the underlying BMBF project, *mlreflect* is open source and licensed with the commonly used MIT license. As such, anybody is allowed to read the source code and make modified copies of it. This is meant to provide the scattering community with the tools and knowledge to not only use the software, but also build and expand on the concepts that were developed in this work.

E. Example of ambiguous box models

This chapter shows an example based on [92], where two different SLD profiles $\rho^{\mathrm{A}}(z)$ and $\rho^{\mathrm{B}}(z)$, *i.e.* two different sample structures, lead to mathematically identical reflectivity curves due to the phase loss. This example is provided as an illustration of possible ambiguity in reflectometry and is not meant to be a complete description of all possible solutions.

This calculation uses the kinematical description of reflectivity given by Eq. 2.49, where modulus squared of the FT can be written as

$$F(q_z) = \left| \int_{-\infty}^{\infty} \frac{d\rho(z)}{dz} e^{iq_z z} dz \right|^2 . \tag{E.1}$$

If the SLD profile $\rho(z)$ follows a box model, *i.e.* it is comprised solely of N step functions at positions z_k, its derivative becomes a sum of delta functions

$$\frac{d\rho(z)}{dz} = \sum_{k=1}^{N} (\rho_k - \rho_{k-1}) \delta(z - z_k). \tag{E.2}$$

This means the FT reduces to a sum of exponentials with an amplitude based on the SLD contrast, *i.e.*

$$F(q_z) = \left| \sum_{k=1}^{N} (\rho_{k+1} - \rho_k) e^{iq_z z_k} \right|^2 . \tag{E.3}$$

Assuming a box model A with the 4 discrete SLDs ρ_1^{A}, ρ_2^{A}, ρ_3^{A} and ρ_4^{A} with steps at z_1, z_2 and z_3, then

$$F^A(q_z) = \left|(\rho_2^{\mathrm{A}} - \rho_1^{\mathrm{A}})e^{iq_z z_1} + (\rho_3^{\mathrm{A}} - \rho_2^{\mathrm{A}})e^{iq_z z_2} + (\rho_4^{\mathrm{A}} - \rho_3^{\mathrm{A}})e^{iq_z z_3}\right|^2 \tag{E.4}$$

If $z_1 = 0$ and z_3 is a multiple of z_2, *e.g.* $z_3 = 2z_2$, the absolute square of the FT can be written as

$$F^A(q_z) = \left|(\rho_2^{\mathrm{A}} - \rho_1^{\mathrm{A}}) + (\rho_3^{\mathrm{A}} - \rho_2^{\mathrm{A}})e^{i\alpha} + (\rho_4^{\mathrm{A}} - \rho_3^{\mathrm{A}})e^{2i\alpha}\right|^2, \tag{E.5}$$

where $\alpha = q_z z_2$. It is interesting to now compare this with another box model B with 3 discrete SLDs ρ_1^{B}, ρ_2^{B} and ρ_1^{B} with steps at z_1 and z_3, *i.e.*

$$F^B(q_z) = \left|(\rho_2^{\mathrm{B}} - \rho_1^{\mathrm{B}})e^{iq_z z_1} + (\rho_3^{\mathrm{B}} - \rho_2^{\mathrm{B}})e^{iq_z z_3}\right|^2 = \left|(\rho_2^{\mathrm{B}} - \rho_1^{\mathrm{B}}) + (\rho_3^{\mathrm{B}} - \rho_2^{\mathrm{B}})e^{2i\alpha}\right|^2. \tag{E.6}$$

To find all cases for which the two models produce the same scattering pattern, one must solve for

$$F^A(q_z) = F^B(q_z). \tag{E.7}$$

For simplicity, only the specific example shown in Fig. E.1 [92] is considered, where $\rho_1^{\mathrm{A}} = \rho_1^{\mathrm{B}} = 0$ and $\rho_2^{\mathrm{A}} = \rho_4^{\mathrm{A}} = \rho_3^{\mathrm{B}}$ and thus,

$$\begin{aligned}
F^A(q_z) &= \left|\rho_2^{\mathrm{A}} + (\rho_3^{\mathrm{A}} - \rho_2^{\mathrm{A}})e^{i\alpha} + (\rho_2^{\mathrm{A}} - \rho_3^{\mathrm{A}})e^{2i\alpha}\right|^2 && \text{(E.8)}\\
&= \left|\rho_2^{\mathrm{A}} + (\rho_3^{\mathrm{A}} - \rho_2^{\mathrm{A}})(e^{i\alpha} - e^{2i\alpha})\right|^2 && \text{(E.9)}\\
&= \left|\rho_2^{\mathrm{A}} + (\rho_3^{\mathrm{A}} - \rho_2^{\mathrm{A}})(\cos(\alpha) - \cos(2\alpha)) + i(\rho_3^{\mathrm{A}} - \rho_2^{\mathrm{A}})(\sin(\alpha) - \sin(2\alpha))\right|^2. && \text{(E.10)}
\end{aligned}$$

Using the rule for the modulus squared of complex numbers $|a + ib|^2 = a^2 + b^2$,

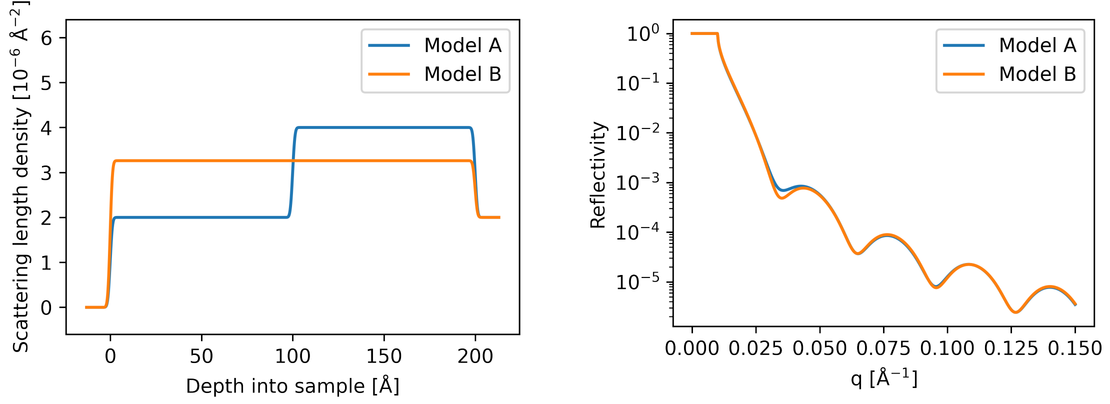

Figure E.1.: Left: Two SLD profiles $\rho^{\mathrm{A}}(z)$ and $\rho^{\mathrm{B}}(z)$ that produce the same reflectivity curve $R(q_z)$ within the kinematical approximation of Eq. 2.49. Right: Corresponding reflectivity profiles simulated using the matrix method from Subsec. 2.2.4. The small difference between the reflectivity curves is due to multiple scattering, which is not included in the kinematical approximation.

$F^A(q_z)$ can be written as

$$F^A(q_z) = (\rho_2^{\mathrm{A}} + (\rho_3^{\mathrm{A}} - \rho_2^{\mathrm{A}})(\cos(\alpha) - \cos(2\alpha)))^2 + (\rho_3^{\mathrm{A}} - \rho_2^{\mathrm{A}})^2(\sin(\alpha) - \sin(2\alpha))^2 \tag{E.11}$$

$$\begin{aligned} &= (\rho_2^{\mathrm{A}})^2 + 2\rho_2^{\mathrm{A}}(\rho_3^{\mathrm{A}} - \rho_2^{\mathrm{A}})(\cos(\alpha) - \cos(2\alpha)) \\ &\quad + (\rho_3^{\mathrm{A}} - \rho_2^{\mathrm{A}})^2(\cos(\alpha) - \cos(2\alpha))^2 \\ &\quad + (\rho_3^{\mathrm{A}} - \rho_2^{\mathrm{A}})^2(\sin(\alpha) - \sin(2\alpha))^2 \end{aligned} \tag{E.12}$$

$$\begin{aligned} &= (\rho_2^{\mathrm{A}})^2 + 2\rho_2^{\mathrm{A}}(\rho_3^{\mathrm{A}} - \rho_2^{\mathrm{A}})(\cos(\alpha) - \cos(2\alpha)) \\ &\quad + (\rho_3^{\mathrm{A}} - \rho_2^{\mathrm{A}})^2(\cos^2(\alpha) - 2\cos(\alpha)\cos(2\alpha) + \cos^2(2\alpha) \\ &\quad + \sin^2(\alpha) - 2\sin(\alpha)\sin(2\alpha) + \sin^2(2\alpha)) \end{aligned} \tag{E.13}$$

$$\begin{aligned} &= (\rho_2^{\mathrm{A}})^2 + 2\rho_2^{\mathrm{A}}(\rho_3^{\mathrm{A}} - \rho_2^{\mathrm{A}})(\cos(\alpha) - \cos(2\alpha)) \\ &\quad + (\rho_3^{\mathrm{A}} - \rho_2^{\mathrm{A}})^2(2 - 2\cos(\alpha)\cos(2\alpha) - 2\sin(\alpha)\sin(2\alpha)) \end{aligned} \tag{E.14}$$

$$= (\rho_2^{\mathrm{A}})^2 + 2\rho_2^{\mathrm{A}}(\rho_3^{\mathrm{A}} - \rho_2^{\mathrm{A}})(\cos(\alpha) - \cos(2\alpha)) + 2(\rho_3^{\mathrm{A}} - \rho_2^{\mathrm{A}})^2(1 - \cos(\alpha)) \tag{E.15}$$

$$\begin{aligned} &= (\rho_2^{\mathrm{A}})^2 + 2\rho_2^{\mathrm{A}}(\rho_3^{\mathrm{A}} - \rho_2^{\mathrm{A}})\cos(\alpha) - 2\rho_2^{\mathrm{A}}(\rho_3^{\mathrm{A}} - \rho_2^{\mathrm{A}})\cos(2\alpha) \\ &\quad + 2(\rho_3^{\mathrm{A}} - \rho_2^{\mathrm{A}})^2 - 2(\rho_3^{\mathrm{A}} - \rho_2^{\mathrm{A}})^2\cos(\alpha). \end{aligned} \tag{E.16}$$

Furthermore, in the special case of $\rho_3^{\mathrm{A}} - \rho_2^{\mathrm{A}} = \rho_2^{\mathrm{A}}$, Eq. E.16 simplifies into

$$F^A(q_z) = 3(\rho_2^{\mathrm{A}})^2 - 2(\rho_2^{\mathrm{A}})^2 \cos(2\alpha) = (\rho_2^{\mathrm{A}})^2(3 - 2\cos(2\alpha)). \tag{E.17}$$

A similar result can be derived for $F^B(q_z)$, where

$$F^B(q_z) = \left|\rho_2^{\mathrm{B}} + (\rho_3^{\mathrm{B}} - \rho_2^{\mathrm{B}})e^{2i\alpha}\right|^2 \tag{E.18}$$

$$= (\rho_2^{\mathrm{B}})^2 + 2\rho_2^{\mathrm{B}}(\rho_3^{\mathrm{B}} - \rho_2^{\mathrm{B}})\cos(2\alpha) + (\rho_3^{\mathrm{B}} - \rho_2^{\mathrm{B}})^2(\sin^2(2\alpha) + \cos^2(2\alpha)) \tag{E.19}$$

$$= (\rho_2^{\mathrm{B}})^2 + 2\rho_2^{\mathrm{B}}(\rho_3^{\mathrm{B}} - \rho_2^{\mathrm{B}})\cos(2\alpha) + (\rho_3^{\mathrm{B}} - \rho_2^{\mathrm{B}})^2 \tag{E.20}$$

$$= (\rho_2^{\mathrm{B}})^2 + 2\rho_2^{\mathrm{B}}(\rho_2^{\mathrm{A}} - \rho_2^{\mathrm{B}})\cos(2\alpha) + (\rho_2^{\mathrm{A}} - \rho_2^{\mathrm{B}})^2. \tag{E.21}$$

Despite the numerous conditions, Eq. E.7 is still underdetermined and by assuming $\rho_2^{\mathrm{A}} = 2$, the equation can be written as

$$12 - 8\cos(2\alpha) = (\rho_2^{\mathrm{B}})^2 + 2\rho_2^{\mathrm{B}}(2 - \rho_2^{\mathrm{B}})\cos(2\alpha) + (2 - \rho_2^{\mathrm{B}})^2 \tag{E.22}$$

$$= 2\rho_2^{\mathrm{B}}(2 - \rho_2^{\mathrm{B}})\cos(2\alpha) + 4 - 4\rho_2^{\mathrm{B}} + 2(\rho_2^{\mathrm{B}})^2 \tag{E.23}$$

$$= 4 + (4\rho_2^{\mathrm{B}} - 2(\rho_2^{\mathrm{B}})^2) - (4\rho_2^{\mathrm{B}} - 2(\rho_2^{\mathrm{B}})^2)\cos(2\alpha) \tag{E.24}$$

and it becomes apparent that this equation has a solution for

$$4\rho_2^{\mathrm{B}} - 2(\rho_2^{\mathrm{B}})^2 = 8 \tag{E.25}$$

or

$$4\rho_2^{\mathrm{B}} - 2(\rho_2^{\mathrm{B}})^2 - 8 = 0. \tag{E.26}$$

The two roots of this quadratic equation can be found *via*

$$(\rho_2^{\mathrm{B}})_{1/2} = \frac{4 \pm \sqrt{16 + 64}}{4} = \frac{4 \pm \sqrt{80}}{4} = 1 \pm \sqrt{5} \approx \begin{cases} +3.24 \\ -1.24 \end{cases}. \tag{E.27}$$

This shows that, within the kinematical approximation, box models A and B as shown in Fig. E.1 produce exactly the same reflectivity curve. The reflectivity curves shown in Fig. E.1 were simulated using the more precise matrix method shown in Subsec. 2.2.4. Despite this, the curves are almost identical and it is reasonable to assume that moderate experimental noise might obfuscate this difference.

F. Datasets used in this work

Tab. F.1 shows all datasets used in this work for NN testing as well as additional information, such as who authored them and in which chapter they were used. More information can be found in the caption of the table.

Table F.1.: A summary of all datasets used in this work for testing NN performance. Credit for preparing the samples and acquiring the data goes to the respective people. All data analysis presented in this work was performed by A. Greco. "#Curves" refers to the number of curves in each dataset. "Materials" refers to the types of molecules of the deposited thin film. All films were grown on Si/SiO_x substrates. "T_{growth}" refers to the substrate temperature during film growth. "Year" refers to the date the data was acquired. "Source" refers to the source used to conduct the reflectometry measurements, *i.e.* the respective beamline and facility acronym in the case of synchrotron sources and a commercial GE XRD3003 diffractometer from the research group of Prof. Schreiber (Institut für Angew. Physik, Universität Tübingen) in the case of the $CuK_{\alpha1}$ source. "Chapters" refers to the chapters in this work where the dataset was used to produce the results.

Dataset name	#Curves	Materials	T_{growth}	Sample preparation	Data acquisition	Year	Source	Chapters
DIP RT 1	69	DIP	303 K	A. Hinderhofer	A. Hinderhofer	2010	MSX04/SLS	4, 6
DIP RT 2	61	DIP	303 K	C. Lorch	C. Lorch	2012	MSX04/SLS	4, 6
DIP HT	79	DIP	403 K	S. Kowarik	S. Kowarik	2005	ID10b/ESRF	4, 6
CuPc RT	228	CuPc	303 K	T. Hosokai	T. Hosokai	2010	ID10b/ESRF	4
6T RT	15	6T	303 K	C. Lorch	C. Lorch	2012	MSX04/SLS	4
PEN RT	20	PEN	303 K	I. Dax	I. Dax	2021	$CuK_{\alpha1}$	6
PEN LT	11	PEN	77 K	I. Dax	I. Dax	2021	$CuK_{\alpha1}$	6
PDI-C8 RT	1	PDI-C8	303 K	N. Rußegger	A. Greco	2021	P08/DESY	6
DNNT:PDIF (1:2) HT	1	DNTT/PDIF	403 K	N. Rußegger	A. Greco	2021	P08/DESY	6
PDI-C8:DIP (1:1) RT	1	PDI-C8	303 K	N. Rußegger	A. Greco	2021	P08/DESY	6

Bibliography

1. Als-Nielsen, J. & McMorrow, D. *Elements of Modern X-ray Physics* 2nd edition (John Wiley & Sons, Ltd, Chichester, 2011).
2. Lovesey, S. W. *Theory of neutron scattering from condensed matter* (Clarendon Press, 1984).
3. Sivia, D. S. *Elementary Scattering Theory for X-ray and Neutron Users* (Oxford University Press, 2011).
4. Yabashi, M. & Tanaka, H. The next ten years of X-ray science. *Nat. Photonics* **11,** 12 (2017).
5. Kowarik, S. Thin film growth studies using time-resolved x-ray scattering. *J. Phys.: Condens. Matter* **29,** 043003 (2017).
6. Greco, A., Hinderhofer, A., Dar, M. I., Arora, N., Hagenlocher, J., Chumakov, A., Grätzel, M. & Schreiber, F. Kinetics of Ion-Exchange Reactions in Hybrid Organic–Inorganic Perovskite Thin Films Studied by In Situ Real-Time X-ray Scattering. *J. Phys. Chem. Lett.* **9,** 6750 (2018).
7. Joress, H., Brock, J. D. & Woll, A. R. Quick X-ray reflectivity using monochromatic synchrotron radiation for time-resolved applications. *J. Synchrotron Rad.* **25,** 706 (2018).
8. Lippmann, M., Buffet, A., Pflaum, K., Ehnes, A., Ciobanu, A. & Seeck, O. H. A new setup for high resolution fast X-ray reflectivity data acquisition. *Rev. Sci. Instrum.* **87,** 113904 (2016).

9. Hinderhofer, A., Gerlach, A., Kowarik, S., Zontone, F., Krug, J. & Schreiber, F. Smoothing and coherent structure formation in organic-organic heterostructure growth. *Europhys. Lett.* **91,** 56002 (2010).

10. Miyadera, T., Shibata, Y., Koganezawa, T., Murakami, T. N., Sugita, T., Tanigaki, N. & Chikamatsu, M. Crystallization Dynamics of Organolead Halide Perovskite by Real-Time X-ray Diffraction. *Nano Lett.* **15,** 5630 (2015).

11. Gisriel, C., Coe, J., Letrun, R., Yefanov, O. M., Luna-Chavez, C., Stander, N. E., Lisova, S., Mariani, V., Kuhn, M., Aplin, S., Grant, T. D., Dörner, K., Sato, T., Echelmeier, A., Villarreal, J. C., Hunter, M. S., Wiedorn, M. O., Knoska, J., Mazalova, V., Roy-Chowdhury, S., Yang, J.-H., Jones, A., Bean, R., Bielecki, J., Kim, Y., Mills, G., Weinhausen, B., Meza, J. D., Al-Qudami, N., Bajt, S., Brehm, G., Botha, S., Boukhelef, D., Brockhauser, S., Bruce, B. D., Coleman, M. A., Danilevski, C., Discianno, E., Dobson, Z., Fangohr, H., Martin-Garcia, J. M., Gevorkov, Y., Hauf, S., Hosseinizadeh, A., Januschek, F., Ketawala, G. K., Kupitz, C., Maia, L., Manetti, M., Messerschmidt, M., Michelat, T., Mondal, J., Ourmazd, A., Previtali, G., Sarrou, I., Schön, S., Schwander, P., Shelby, M. L., Silenzi, A., Sztuk-Dambietz, J., Szuba, J., Turcato, M., White, T. A., Wrona, K., Xu, C., Abdellatif, M. H., Zook, J. D., Spence, J. C. H., Chapman, H. N., Barty, A., Kirian, R. A., Frank, M., Ros, A., Schmidt, M., Fromme, R., Mancuso, A. P., Fromme, P. & Zatsepin, N. A. Membrane protein megahertz crystallography at the European XFEL. *Nat. Commun.* **10,** 5021 (2019).

12. Hilbert, M. & López, P. The World's Technological Capacity to Store, Communicate, and Compute Information. *Science* **332,** 60 (2011).

13. Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žıdek, A., Potapenko, A., Bridgland, A., Meyer, C., Kohl, S. A. A., Ballard, A. J., Cowie, A., Romera-Paredes, B., Nikolov, S., Jain, R., Adler, J., Back, T., Petersen, S., Reiman, D., Clancy, E., Zielinski, M., Steinegger, M., Pacholska, M., Berghammer, T., Bodenstein, S.,

Silver, D., Vinyals, O., Senior, A. W., Kavukcuoglu, K., Kohli, P. & Hassabis, D. Highly accurate protein structure prediction with AlphaFold. *Nature* **596,** 583 (2021).

14. Young, T., Hazarika, D., Poria, S. & Cambria, E. Recent Trends in Deep Learning Based Natural Language Processing. *IEEE Comput. Intell. Mag.* **13,** 55 (2018).
15. Rosenblatt, F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychol. Rev.* **65,** 386 (1958).
16. Rumelhart, D. E., Hinton, G. E. & Williams, R. J. Learning representations by back-propagating errors. *Nature* **323,** 533 (1986).
17. McClelland, J. L., McNaughton, B. L. & O'Reilly, R. C. Why there are complementary learning systems in the hippocampus and neocortex: Insights from the successes and failures of connectionist models of learning and memory. *Psychol. Rev.* **102,** 419 (1995).
18. Delalleau, O. & Bengio, Y. *Shallow vs. deep sum-product networks* in *NeurIPS* (2011).
19. Montúfar, G. Universal approximation depth and errors of narrow belief networkswith discrete units. *Neural Comput.* **26,** 1386 (2014).
20. Pascanu, R., Gülçehre, Ç., Cho, K. & Bengio, Y. *How to construct deep recurrent neural networks* in *ICLR* (2014).
21. Krizhevsky, A. & Hinton, G. *Learning multiple layers of features from tiny images* tech. rep. (University of Toronto, 2009).
22. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B. & Ng, A. Y. *Reading digits in natural images with unsupervised feature learning* in *NeurIPS* (2011).
23. Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K. & Fei-Fei, L. *ImageNet: A Large-Scale Hierarchical Image Database* in *CVPR* (2009).
24. Deng, J., Berg, A. C., Li, K. & Fei-Fei, L. *What does classifying more than 10,000 image categories tell us?* in *ECCV* (2010).

25. Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R. & Fei-Fei, L. *Large-scale video classification with convolutional neural networks* in *CVPR* (2014).
26. Goodfellow, I., Bengio, Y. & Courville, A. *Deep Learning* http://www.deeplearningbook.org (MIT Press, 2016).
27. Raina, R., Madhavan, A. & Ng, A. Y. *Large-scale deep unsupervised learning using graphics processors* in *ICML* (eds Bottou, L. & Littman, M.) (2009), 873.
28. Chellapilla, K., Puri, S. & Simard, P. *High Performance Convolutional Neural Networks for Document Processing* in *IWFHR* (ed Lorette, G.) http://www.suvisoft.com (2006).
29. Ciresan, D. C., Meier, U., Gambardella, L. M. & Schmidhuber, J. Deep big simple neural nets for handwritten digit recognition. *Neural Comput.* **22,** 1 (2010).
30. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mane, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viegas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y. & Zheng, X. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems* 2016. arXiv: 1603.04467 [cs.DC].
31. Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D. & Bengio, Y. *Theano: a CPU and GPU math expression compiler* in *Proc. Scipy* (2010).
32. Goodfellow, I. J., Warde-Farley, D., Lamblin, P., Dumoulin, V., Mirza, M., Pascanu, R., Bergstra, J., Bastien, F. & Bengio, Y. *Pylearn2: a machine learning research library* 2013. arXiv: 1308.4214 [stat.ML].

33. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J. & Chintala, S. *PyTorch: An Imperative Style, High-Performance Deep Learning Library* in *NeurIPS* (2019).

34. Dean, J., Corrado, G. S., Monga, R., Chen, K., Devin, M., Le, Q. V., Mao, M. Z., Ranzato, M., Senior, A., Tucker, P., Yang, K. & Ng, A. Y. *Large scale distributed deep networks* in *NeurIPS* (2012).

35. Jia, Y. *Caffe: An open source convolutional architecture for fast feature embedding.* `http://caffe.berkeleyvision.org/`. 2013.

36. Chen, T., Li, M., Li, Y., Lin, M., Wang, N., Wang, M., Xiao, T., Xu, B., Zhang, C. & Zhang, Z. *MXNet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems* 2015. arXiv: `1512.01274 [cs.DC]`.

37. Erdmann, M., Glombitza, J., Kasieczka, G. & Klemradt, U. *Deep Learning for Physics Research* (WORLD SCIENTIFIC, 2021).

38. Ludwig, A. Discovery of new materials using combinatorial synthesis and high-throughput characterization of thin-film materials libraries combined with computational methods. *npj Comput. Mater.* **5,** 70 (2019).

39. Egger, A. T., Hörmann, L., Jeindl, A., Scherbela, M., Obersteiner, V., Todorović, M., Rinke, P. & Hofmann, O. T. Charge Transfer into Organic Thin Films: A Deeper Insight through Machine-Learning-Assisted Structure Search. *Adv. Sci.* **7,** 2000992 (2020).

40. Kusne, A. G., Gao, T., Mehta, A., Ke, L., Nguyen, M. C., Ho, K.-M., Antropov, V., Wang, C.-Z., Kramer, M. J., Long, C. & Takeuchi, I. On-the-fly machine-learning for high-throughput experiments: search for rare-earth-free permanent magnets. *Sci. Rep.* **4,** 6367 (2014).

41. Martynec, T., Karapanagiotis, C., Klapp, S. H. L. & Kowarik, S. Machine learning predictions of surface migration barriers in nucleation and non-equilibrium growth. *Comms. Mats.* **2,** 90 (2021).

42. Kern, S., Liehr, S., Wander, L., Bornemann-Pfeiffer, M., Müller, S., Maiwald, M. & Kowarik, S. Artificial neural networks for quantitative online NMR spectroscopy. *Anal. Bioanal. Chem.* **412,** 4447 (2020).

43. Urban III, F., Barton, D. & Boudani, N. Extremely fast ellipsometry solutions using cascaded neural networks alone. *Thin Solid Films* **332,** 50 (1998).

44. Chen, Z., Andrejevic, N., Drucker, N. C., Nguyen, T., Xian, R. P., Smidt, T., Wang, Y., Ernstorfer, R., Tennant, D. A., Chan, M. & Li, M. Machine learning on neutron and x-ray scattering and spectroscopies. *Chem. Phys. Rev.* **2,** 031301 (2021).

45. Park, W. B., Chung, J., Jung, J., Sohn, K., Singh, S. P., Pyo, M., Shin, N. & Sohn, K.-S. Classification of crystal structure using a convolutional neural network. *IUCrJ* **4,** 486 (2017).

46. Oviedo, F., Ren, Z., Sun, S., Settens, C., Liu, Z., Hartono, N. T. P., Ramasamy, S., DeCost, B. L., Tian, S. I., Romano, G., Kusne, A. G. & Buonassisi, T. Fast and interpretable classification of small X-ray diffraction datasets using data augmentation and deep neural networks. *npj Comput. Mater.* **5,** 60 (2019).

47. Aguiar, J., Gong, M. L., Unocic, R., Tasdizen, T. & Miller, B. Decoding crystallography from high-resolution electron imaging and diffraction datasets with deep learning. *Sci. Adv.* **5,** eaaw1949 (2019).

48. Souza, A., Oliveira, L. B., Hollatz, S., Feldman, M., Olukotun, K., Holton, J. M., Cohen, A. E. & Nardi, L. *DeepFreak: Learning Crystallography Diffraction Patterns with Automated Machine Learning* 2019. arXiv: 1904.11834 [cs.LG].

49. Sullivan, B., Archibald, R., Azadmanesh, J., Vandavasi, V. G., Langan, P. S., Coates, L., Lynch, V. & Langan, P. BraggNet: integrating Bragg peaks using neural networks. *J. Appl. Crystallogr.* **52,** 854 (2019).

50. Vecsei, P. M., Choo, K., Chang, J. & Neupert, T. Neural network based classification of crystal symmetries from x-ray diffraction patterns. *Phys. Rev. B* **99,** 245120 (2019).

51. Lee, J.-W., Park, W. B., Lee, J. H., Singh, S. P. & Sohn, K.-S. A deep-learning technique for phase identification in multiphase inorganic compounds using synthetic XRD powder patterns. *Nat. Commun.* **11,** 86 (2020).

52. Dong, H., Butler, K. T., Matras, D., Price, S. W., Odarchenko, Y., Khatry, R., Thompson, A., Middelkoop, V., Jacques, S. D., Beale, A. M. & Vamvakeros, A. A deep convolutional neural network for real-time full profile analysis of big powder diffraction data. *npj Comput. Mater.* **7,** 74 (2021).

53. Liu, Z., Sharma, H., Park, J.-S., Kenesei, P., Miceli, A., Almer, J., Kettimuthu, R. & Foster, I. *BraggNN: Fast X-ray Bragg Peak Analysis Using Deep Learning* 2021. arXiv: 2008.08198 [eess.IV].

54. Franke, D., Jeffries, C. M. & Svergun, D. I. Machine Learning Methods for X-Ray Scattering Data Analysis from Biomacromolecular Solutions. *Biophys. J.* **114,** 2485 (2018).

55. Archibald, R. K., Doucet, M., Johnston, T., Young, S. R., Yang, E. & Heller, W. T. Classifying and analyzing small-angle scattering data using weighted k nearest neighbors machine learning techniques. *J. Appl. Crystallogr.* **53,** 326 (2020).

56. Chang, M.-C., Wei, Y., Chen, W.-R. & Do, C. Deep learning-based super-resolution for small-angle neutron scattering data: attempt to accelerate experimental workflow. *MRS Commun.* **10,** 11 (2020).

57. Song, G., Porcar, L., Boehm, M., Cecillon, F., Dewhurst, C., Goc, Y. L., Locatelli, J., Mutti, P. & Weber, T. Deep Learning Methods On Neutron Scattering Data. *EPJ Web Conf.* **225** (eds Lyoussi, A., Giot, M., Carette, M., Jenčič, I., Reynard-Carette, C., Vermeeren, L., Snoj, L. & Dû, P. L.) 01004 (2020).
58. Liu, S., Melton, C. N., Venkatakrishnan, S., Pandolfi, R. J., Freychet, G., Kumar, D., Tang, H., Hexemer, A. & Ushizima, D. M. Convolutional neural networks for grazing incidence x-ray scattering patterns: thin film structure identification. *MRS Commun.* **9,** 586 (2019).
59. Ikemoto, H., Yamamoto, K., Touyama, H., Yamashita, D., Nakamura, M. & Okuda, H. Classification of grazing-incidence small-angle X-ray scattering patterns by convolutional neural network. *J. Synchrotron Rad.* **27,** 1069 (2020).
60. Van Herck, W., Fisher, J. & Ganeva, M. Deep learning for x-ray or neutron scattering under grazing-incidence: extraction of distributions. *Mater. Res. Express* **8,** 045015 (2021).
61. Starostin, V., Munteanu, V., Greco, A., Kneschaurek, E., Pleli, A., Bertram, F., Gerlach, A., Hinderhofer, A. & Schreiber, F. Tracking perovskite crystallization via deep learning-based feature detection on 2D X-ray scattering data. *npj Comput. Mater.* **8,** 101 (2022).
62. Samarakoon, A. M., Barros, K., Li, Y. W., Eisenbach, M., Zhang, Q., Ye, F., Sharma, V., Dun, Z., Zhou, H., Grigera, S. A., Batista, C. D. & Tennant, D. A. Machine-learning-assisted insight into spin ice Dy2Ti2O7. *Nat. Commun.* **11,** 892 (2020).
63. Lussier, F., Thibault, V., Charron, B., Wallace, G. Q. & Masson, J.-F. Deep learning and artificial intelligence methods for Raman and surface-enhanced Raman scattering. *Trends. Analyt. Chem.* **124,** 115796 (2020).
64. Kaufmann, K., Zhu, C., Rosengarten, A. S., Maryanovsky, D., Harrington, T. J., Marin, E. & Vecchio, K. S. Crystal symmetry determination in electron diffraction using machine learning. *Science* **367,** 564 (2020).

65. Sanchez-Gonzalez, A., Micaelli, P., Olivier, C., Barillot, T., Ilchen, M., Lutman, A., Marinelli, A., Maxwell, T., Achner, A., Agåker, M., Berrah, N., Bostedt, C., Bozek, J., Buck, J., Buckbaum, P., Carron Montero, S., Cooper, B., Cryan, J., Dong, M., Feifel, R., Frasinski, L., Fukuzawa, H., Galler, A., Hartmann, G., Hartmann, N., W, H., Johnson, A., Knie, A., Lindahl, A., Liu, J., Motomura, K., Mucke, M., O'Grady, C., Rubensson, J.-E., Simpson, E., RJ, S., Såthe, C., Ueda, K., Vacher, M., Walke, D., Zhaunerchyk, V., Coffee, R. & Marangos, J. Accurate prediction of X-ray pulse properties from a free-electron laser using machine learning. *Nat. Commun.* **8,** 15461 (2017).

66. Ke, T.-W., Brewster, A. S., Yu, S. X., Ushizima, D., Yang, C. & Sauter, N. K. A convolutional neural network-based screening tool for X-ray serial crystallography. *J. Synchrotron Radiat.* **25,** 655 (2018).

67. Tolan, M. *X-ray Scattering from Soft-Matter Thin Films: Materials Science and Basic Research* (Springer, Berlin, 1999).

68. Holý, V., Pietsch, U. & Baumbach, T. *High-Resolution X-Ray Scattering from Thin Films and Multilayers* (Springer Berlin, 1999).

69. Daillant, J. & Gibaud, A. *X-Ray and Neutron Reflectivity* (Springer, 2009).

70. Zhou, X.-L. & Chen, S.-H. Theoretical foundation of X-ray and neutron reflectometry. *Phys. Rep.* **257,** 223 (1995).

71. Parratt, L. G. Surface studies of solids by total reflection of X-rays. *Phys. Rev.* **95,** 359 (1954).

72. Braslau, A., Pershan, P. S., Swislow, G., Ocko, B. M. & Als-Nielsen, J. Capillary waves on the surface of simple liquids measured by x-ray reflectivity. *Phys. Rev. A* **38,** 2457 (1988).

73. Metzger, T., Luidl, C., Pietsch, U. & Vierl, U. Novel versatile X-ray reflectometer for angle and energy dispersive characterization of liquid and solid surfaces and interfaces. *Nucl. Instrum. Methods Phys. Res* **350,** 398 (1994).

74. Michely, T. & Krug, J. *Islands, Mounds, and Atoms. Patterns and Processes in Crystal Growth Far from Equilibrium* (Springer, Berlin, 2004).

75. Lehmkühler, F., Paulus, M., Sternemann, C., Lietz, D., Venturini, F., Gutt, C. & Tolan, M. The Carbon Dioxide-Water Interface at Conditions of Gas Hydrate Formation. *J. Am. Chem. Soc.* **131,** 585 (2008).

76. Seeck, O. H., Kim, H., Lee, D. R., Shu, D., Kaendler, I. D., Basu, J. K. & Sinha, S. K. Observation of thickness quantization in liquid films confined to molecular dimension. *EPL* **60,** 376 (2002).

77. Fragneto-Cusani, G. Neutron reflectivity at the solid/liquid interface: examples of applications in biophysics. *J. Phys.: Condens. Matter* **13,** 4973 (2001).

78. Ankner, J., Majkrzak, C. & Satija, S. Neutron reflectivity and grazing angle diffraction. *J. Res. Natl. Inst. Stand. Technol.* **98,** 47 (1993).

79. Mukherjee, M., Bhattacharya, M., Sanyal, M. K., Geue, T., Grenzer, J. & Pietsch, U. Reversible negative thermal expansion of polymer films. *Phys. Rev. E* **66,** 061801 (2002).

80. Neville, F., Cahuzac, M., Konovalov, O., Ishitsuka, Y., Lee, K. Y. C., Kuzmenko, I., Kale, G. M. & Gidalevitz, D. Lipid Headgroup Discrimination by Antimicrobial Peptide LL-37: Insight into Mechanism of Action. *Biophys. J.* **90,** 1275 (2006).

81. Skoda, M. W., Thomas, B., Hagreen, M., Sebastiani, F. & Pfrang, C. Simultaneous neutron reflectometry and infrared reflection absorption spectroscopy (IRRAS) study of mixed monolayer reactions at the air–water interface. *RSC Adv.* **7,** 34208 (2017).

82. Wasserman, S. R., Whitesides, G. M., Tidswell, I. M., Ocko, B. M., Pershan, P. S. & Axe, J. D. The structure of self-assembled monolayers of alkylsiloxanes on silicon: a comparison of results from ellipsometry and low-angle x-ray reflectivity. *J. Am. Chem. Soc.* **111,** 5852 (1989).

83. Skoda, M. W., Conzelmann, N. F., Fries, M. R., Reichart, L. F., Jacobs, R. M., Zhang, F. & Schreiber, F. Switchable β-lactoglobulin (BLG) adsorption on protein resistant oligo (ethylene glycol) (OEG) self-assembled monolayers (SAMs). *J. Colloid Interface Sci.* **606,** 1673 (2022).
84. Tidswell, I. M., Ocko, B. M., Pershan, P. S., Wasserman, S. R., Whitesides, G. M. & Axe, J. D. X-ray specular reflection studies of silicon coated by organic monolayers (alkylsiloxanes). *Phys. Rev. B.* **41,** 1111 (1990).
85. Fenter, P., Schreiber, F., Bulović, V. & Forrest, S. Thermally induced failure mechanisms of organic light emitting device structures probed by X-ray specular reflectivity. *Chem. Phys. Lett.* **277,** 521 (1997).
86. Lorch, C., Banerjee, R., Frank, C., Dieterle, J., Hinderhofer, A., Gerlach, A. & Schreiber, F. Growth of competing crystal phases of α-sexithiophene studied by real-time X-ray scattering. *J. Phys. Chem. C* **119,** 819 (2015).
87. Majkrzak, C. Polarized neutron reflectometry. *Phys. B Condens. Matter* **173,** 75 (1991).
88. Kowarik, S., Gerlach, A., Sellner, S., Schreiber, F., Cavalcanti, L. & Konovalov, O. Real-time observation of structural and orientational transitions during growth of organic thin films. *Phys. Rev. Lett.* **96,** 125504 (2006).
89. Kowarik, S. *Real-time studies of thin film growth of organic semiconductors* PhD thesis (Wadham College, Oxford, 2006).
90. Woll, A. R., Desai, T. V. & Engstrom, J. R. Quantitative modeling of in situ X-ray reflectivity during organic molecule thin film growth. *Phys. Rev. B* **84,** 075479 (2011).
91. Mocuta, C., Stanescu, S., Gallard, M., Barbier, A., Dawiec, A., Kedjar, B., Leclercq, N. & Thiaudiere, D. Fast X-ray reflectivity measurements using an X-ray pixel area detector at the DiffAbs beamline, Synchrotron SOLEIL. *J. Synchrotron Rad.* **25,** 204 (2018).

92. Sivia, D. S., Hamilton, W. A., Smith, G. S., Rieker, T. P. & Pynn, R. A novel experimental procedure for removing ambiguity from the interpretation of neutron and x-ray reflectivity measurements: "Speckle holography". *J. Appl. Phys.* **70,** 732 (1991).

93. Kienzle, P., Krycka, J., Patel, N. & Sahin, I. *Refl1D* `https://www.nist.gov/ncnr/data-reduction-analysis/reflectometry-software`. 2011.

94. Nelson, A. Co-refinement of multiple-contrast neutron/X-ray reflectivity data using *MOTOFIT*. *J. Appl. Crystallogr.* **39,** 273 (2006).

95. Nelson, A. R. J. & Prescott, S. W. *refnx*: neutron and X-ray reflectometry analysis in Python. *J. Appl. Crystallogr.* **52,** 193 (2019).

96. Björck, M. & Andersson, G. GenX: An extensible X-ray reflectivity refinement program utilizing differential evolution. *J. Appl. Crystallogr.* **40,** 1174 (2007).

97. Danauskas, S. M., Li, D., Meron, M., Lin, B. & Lee, K. Y. C. Stochastic fitting of specular X-ray reflectivity data using. *J. Appl. Crystallogr.* **41,** 1187 (2008).

98. Maranville, B., Ratcliff II, W. & Kienzle, P. *reductus*: a stateless Python data reduction service with a browser front end. *J. Appl. Crystallogr.* **51,** 1500 (2018).

99. Doucet, M., Ferraz Leal, R. & Hobson, T. Web interface for reflectivity fitting. *SoftwareX* **7,** 287 (2018).

100. Gerelli, Y. *Aurore*: new software for neutron reflectivity data analysis. *J. Appl. Crystallogr.* **49,** 330 (2016).

101. Pospelov, G., Van Herck, W., Burle, J., Carmona Loaiza, J. M., Durniak, C., Fisher, J. M., Ganeva, M., Yurov, D. & Wuttke, J. *BornAgain*: software for simulating and fitting grazing-incidence small-angle scattering. *J. Appl. Crystallogr.* **53,** 262 (2020).

102. Storn, R. & Price, K. Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *J. Global. Optim.* **11,** 341 (1997).

103. Greco, A., Starostin, V., Karapanagiotis, C., Hinderhofer, A., Gerlach, A., Pithan, L., Liehr, S., Schreiber, F. & Kowarik, S. Fast fitting of reflectivity data of growing thin films using neural networks. *J. Appl. Crystallogr.* **52,** 1342 (2019).

104. Mironov, D., Durant, J. H., Mackenzie, R. & Cooper, J. F. K. Towards automated analysis for neutron reflectivity. *Mach. Learn.: Sci. Technol.* **2,** 035006 (2021).

105. Doucet, M., Archibald, R. K. & Heller, W. T. Machine learning for neutron reflectometry data analysis of two-layer thin films. *Mach. Learn.: Sci. Technol.* **2,** 035001 (2021).

106. Loaiza, J. M. C. & Raza, Z. Towards reflectivity profile inversion through artificial neural networks. *Mach. Learn.: Sci. Technol.* **2,** 025034 (2021).

107. Greco, A., Starostin, V., Hinderhofer, A., Gerlach, A., Skoda, M. W. A., Kowarik, S. & Schreiber, F. Neural network analysis of neutron and x-ray reflectivity data: pathological cases, performance and perspectives. *Mach. Learn.: Sci. Technol.* **2,** 045003 (2021).

108. Andrejevic, N., Chen, Z., Nguyen, T., Fan, L., Heiberger, H., Zhou, L.-J., Zhao, Y.-F., Chang, C.-Z., Grutter, A. & Li, M. Elucidating proximity magnetism through polarized neutron reflectometry and machine learning. *Appl. Phys. Rev.* **9,** 011421 (2022).

109. Kim, K. T. & Lee, D. R. Probabilistic parameter estimation using a Gaussian mixture density network: application to X-ray reflectivity data curve fitting. *J. Appl. Crystallogr.* **54,** 1572 (2021).

110. Lee, B., Yu, K., Jeon, J. & Choi, E. J. Machine learning analysis of broadband optical reflectivity of semiconductor thin film. *J. Korean Phys. Soc.* **80,** 374 (2022).

111. Greco, A., Starostin, V., Edel, E., Munteanu, V., Russegger, N., Dax, I., Shen, C., Bertram, F., Hinderhofer, A., Gerlach, A. & Schreiber, F. Neural network analysis of neutron and X-ray reflectivity data: automated analysis using mlreflect, experimental errors and feature engineering. *J. Appl. Crystallogr.* **55,** 362 (2022).

112. Aoki, H., Liu, Y. & Yamashita, T. Deep learning approach for an interface structure analysis with a large statistical noise in neutron reflectometry. *Sci. Rep.* **11** (2021).

113. Hinderhofer, A. & Schreiber, F. Organic-Organic Heterostructures: Concepts and Applications. *ChemPhysChem* **13,** 628 (2012).

114. Reineke, S., Thomschke, M., Lüssem, B. & Leo, K. White organic light-emitting diodes: Status and perspective. *Rev. Mod. Phys.* **85,** 1245 (2013).

115. de Bergevin, F. in *Lect. Notes Phys.* (eds Daillant, J. & Gibaud, A.) 1 (Springer, Berlin, Heidelberg., 2009).

116. Fermon, C., Ott, F. & Menelle, A. in *Lect. Notes Phys.* (eds Daillant, J. & Gibaud, A.) 183 (Springer, Berlin, Heidelberg., 2009).

117. Reisz, B. *Structure Formation during Organic Molecular Beam Deposition* PhD thesis (Eberhard Karls Universität Tübingen, 2021).

118. Fresnel, A. Mémoire sur la loi réflexion imprime a la lumiere polarisée. *Mém. de l'Acad.* **11,** 393 (1823).

119. Abelès, F. La théorie générale des couches minces. *J. Phys. Radium* **11,** 307 (1950).

120. Heavens, O. S. *Optical properties of thin solid films* (London: Butterworths Scientific Publications, 1955).

121. Gibaud, A. & Vignaud, G. in *Lect. Notes Phys.* (eds Daillant, J. & Gibaud, A.) 85 (2009).

122. Névot, L. & Croce, P. Characterisation of surfaces by grazing x-ray reflection. *Revue de Physique Appliquée* **15,** 761 (1980).

123. Schwoerer, M. & Wolf, H. C. *Organic Molecular Solids* (Wiley, 2006).

124. Brütting, W., Berleb, S. & Mückl, A. G. Device physics of organic light-emitting diodes based on molecular materials. *Org. Electron.* **2,** 1 (2001).

125. Dimitrakopoulos, C. & Malenfant, P. Organic Thin Film Transistors for Large Area Electronics. *Adv. Mater.* **14,** 99 (2002).

126. Dürr, A. C., Schreiber, F., Münch, M., Karl, N., Krause, B., Kruppa, V. & Dosch, H. High structural order in thin films of the organic semiconductor diindenoperylene. *Appl. Phys. Lett.* **81,** 2276 (2002).

127. Dürr, A. C., Schreiber, F., Ritley, K. A., Kruppa, V., Krug, J., Dosch, H. & Struth, B. Rapid Roughening in Thin Film Growth of an Organic Semiconductor (Diindenoperylene). *Phys. Rev. Lett.* **90,** 016104 (2003).

128. Heilig, M., Domhan, M. & Port, H. Optical properties and morphology of thin diindenoperylene films. *J. Lumin.* **110,** 290 (2004).

129. Hill, I. G. & Kahn, A. Combined photoemission/*in vacuo* transport study of the indium tin oxide/copper phthalocyanine/N,N'-diphenyl-N,N'-bis(l-naphthyl)-1,1'biphenyl-4,4"diamine molecular organic semiconductor system. *J. Appl. Phys.* **86,** 2116 (1999).

130. Bao, Z., Lovinger, A. J. & Dodabalapur, A. Organic field-effect transistors with high mobility based on copper phthalocyanine. *Appl. Phys. Lett.* **69,** 3066 (1996).

131. Dinelli, F., Murgia, M., Levy, P., Cavallini, M., Biscarini, F. & de Leeuw, D. M. Spatially Correlated Charge Transport in Organic Thin Film Transistors. *Phys. Rev. Lett.* **92,** 116802 (2004).

132. Hörmann, U., Wagner, J., Gruber, M., Opitz, A. & Brütting, W. Approaching the ultimate open circuit voltage in thiophene based single junction solar cells by applying diindenoperylene as acceptor. *Phys. Status Solidi RRL* **5,** 241 (2011).

133. Ruiz, R., Choudhary, D., Nickel, B., Toccoli, T., Chang, K.-C., Mayer, A. C., Clancy, P., Blakely, J. M., Headrick, R. L., Iannotta, S. & Malliaras, G. G. Pentacene Thin Film Growth. *Chem. Mater.* **16,** 4497 (2004).

134. Faltermeier, D., Gompf, B., Dressel, M., Tripathi, A. K. & Pflaum, J. Optical properties of pentacene thin films and single crystals. *Phys. Rev. B* **74,** 125416 (2006).

135. Piliego, C., Cordella, F., Jarzab, D., Lu, S., Chen, Z., Facchetti, A. & Loi, M. A. Functionalized perylenes: origin of the enhanced electrical performances. *Appl. Phys. A* **95,** 303 (2009).

136. Ringk, A., Roelofs, W. C., Smits, E. C., van der Marel, C., Salzmann, I., Neuhold, A., Gelinck, G. H., Resel, R., de Leeuw, D. M. & Strohriegl, P. n-Type self-assembled monolayer field-effect transistors for flexible organic electronics. *Org. Electron.* **14,** 1297 (2013).

137. Belova, V., Wagner, B., Reisz, B., Zeiser, C., Duva, G., Rozbořil, J., Novák, J., Gerlach, A., Hinderhofer, A. & Schreiber, F. Real-Time Structural and Optical Study of Growth and Packing Behavior of Perylene Diimide Derivative Thin Films: Influence of Side-Chain Modification. *J. Phys. Chem. C.* **122,** 8589 (2018).

138. Zschieschang, U., Ante, F., Kälblein, D., Yamamoto, T., Takimiya, K., Kuwabara, H., Ikeda, M., Sekitani, T., Someya, T., Nimoth, J. B. & Klauk, H. Dinaphtho[2,3-b:2′,3′-f]thieno[3,2-b]thiophene (DNTT) thin-film transistors with improved performance and stability. *Org. Electron.* **12,** 1370 (2011).

139. Kraft, U., Takimiya, K., Kang, M. J., Rödel, R., Letzkus, F., Burghartz, J. N., Weber, E. & Klauk, H. Detailed analysis and contact properties of low-voltage organic thin-film transistors based on dinaphtho[2,3-b:2′,3′-f]thieno[3,2-b]thiophene (DNTT) and its didecyl and diphenyl derivatives. *Org. Electron.* **35,** 33 (2016).

140. Ritley, K. A., Krause, B., Schreiber, F. & Dosch, H. A portable ultrahigh vacuum organic molecular beam deposition system for in situ x-ray diffraction measurements. *Rev. Sci. Instrum.* **72,** 1453 (2001).

141. Mitchell, T. M. *Machine Learning* (McGraw-Hill, New York, 1997).

142. Friedman, J., Hastie, T. & Tibshirani, R. *The Elements of Statistical Learning* (Springer series in statistics, New York, 2001).

143. Hecht-Nielsen, R. *Neural Networks for Perception* (ed Wechsler, H.) 65 (Academic Press, 1992).

144. Nair, V. & Hinton, G. E. *Rectified Linear Units Improve Restricted Boltzmann Machines* in *ICML* (2010), 807.

145. Glorot, X., Bordes, A. & Bengio, Y. *Deep Sparse Rectifier Neural Networks* in *AISTATS* (eds Gordon, G., Dunson, D. & Dudík, M.) (2011), 315.

146. Maas, A. L., Hannun, A. Y. & Ng, A. Y. *Rectifier Nonlinearities Improve Neural Network Acoustic Models* in *ICML* **30** (2013), 3.

147. He, K., Zhang, X., Ren, S. & Sun, J. *Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification* in *ICCV* (2015), 1026.

148. Cybenko, G. Approximation by superpositions of a sigmoidal function. *Math. Control Signals Syst.* **2,** 303 (1989).

149. Hornik, K. Approximation capabilities of multilayer feedforward networks. *Neural Netw.* **4,** 251 (1991).

150. Bottou, L. in *Online Learning and Neural Networks* (ed Saad, D.) revised, oct 2012 (Cambridge University Press, Cambridge, UK, 1998).

151. Wilson, D. R. & Martinez, T. R. The general inefficiency of batch training for gradient descent learning. *Neural Networks* **16,** 1429 (2003).

152. Glorot, X. & Bengio, Y. *Understanding the difficulty of training deep feedforward neural networks* in *AISTATS* (2010), 249.

153. Polyak, B. & Juditsky, A. Some methods of speeding up the convergence of iteration methods. *USSR Comput. Math. Math. Phys.* **4,** 1 (1964).

154. Kingma, D. P. & Ba, J. *Adam: A Method for Stochastic Optimization* 2017. arXiv: 1412.6980 [cs.LG].

155. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **15,** 1929 (2014).

156. Bishop, C. M. *Regularization and complexity control in feed-forward networks* in *ICANN* **1** (1995), 141.

157. Sjöberg, J. & l. Ljung. Overtraining, regularization and searching for a minimum, with application to neural networks. *Int. J. Control* **62,** 1391 (1995).

158. Sears, V. F. Neutron scattering lengths and cross sections. *Neutron News* **3,** 26 (1992).

159. Hoogerheide, D. P., Heinrich, F., Maranville, B. B. & Majkrzak, C. F. Accurate background correction in neutron reflectometry studies of soft condensed matter films in contact with fluid reservoirs. *J. Appl. Crystallogr.* **53,** 15 (2020).

160. Sivia, D. & Webster, J. The Bayesian approach to reflectivity data. *Phys. B: Condens. Matter* **248,** 327 (1998).

161. Treece, B. W., Kienzle, P. A., Hoogerheide, D. P., Majkrzak, C. F., Lösche, M. & Heinrich, F. Optimization of reflectometry experiments using information theory. *J. Appl. Crystallogr.* **52,** 47 (2019).

162. Durant, J. H., Wilkins, L., Butler, K. & Cooper, J. F. K. Determining the maximum information gain and optimizing experimental design in neutron reflectometry using the Fisher information. *J. Appl. Crystallogr.* **54,** 1100 (2021).

163. Gibaud, A., Vignaud, G. & Sinha, S. K. The correction of geometrical factors in the analysis of X-ray reflectivity. *Acta Cryst. A* **49,** 642 (1993).

164. Moré, J. J. in *Numerical Analysis* (ed Watson, G. A.) 105 (Springer, New York, 1977).

165. Smilgies, D.-M., Boudet, N., Struth, B. & Konovalov, O. Troika II: a versatile beamline for the study of liquid and solid interfaces. *J. Synchrotron Rad.* **12,** 329 (2005).

166. Seeck, O. H., Deiter, C., Pflaum, K., Bertam, F., Beerlink, A., Franz, H., Horbach, J., Schulte-Schrepping, H., Murphy, B. M., Greve, M. & Magnussen, O. The high-resolution diffraction beamline P08 at PETRA III. *J. Synchrotron Rad.* **19,** 30 (2011).

167. Patterson, B., Abela, R., Auderset, H., Chen, Q., Fauth, F., Gozzo, F., Ingold, G., Kühne, H., Lange, M., Maden, D., Meister, D., Pattison, P., Schmidt, T., Schmitt, B., Schulze-Briese, C., Shi, M., Stampanoni, M. & Willmott, P. The materials science beamline at the Swiss Light Source: design and realization. *Nucl. Instrum. Methods Phys. Res. A* **540,** 42 (2005).

168. Sentenac, D., Shalaginov, A. N., Fera, A. & de Jeu, W. H. On the instrumental resolution in X-ray reflectivity experiments. *J. Appl. Crystallogr.* **33,** 130 (2000).

169. Gibbs, D., Ocko, B. M., Zehner, D. M. & Mochrie, S. G. J. Absolute x-ray reflectivity study of the Au(100) surface. *Phys. Rev. B* **38,** 7303 (1988).

170. Bishop, C. M. *Mixture density networks* tech. rep. (Aston University, Birmingham, 1994).

171. Kruse, J. *Technical report: Training Mixture Density Networks with full covariance matrices* 2020. arXiv: 2003.05739 [cs.LG].

172. Unni, R., Yao, K. & Zheng, Y. Deep Convolutional Mixture Density Network for Inverse Design of Layered Photonic Structures. *ACS Photonics* **7,** 2703 (2020).

173. Hinderhofer, A., Greco, A., Starostin, V., Munteanu, V., Pithan, L., Gerlach, A. & Schreiber, F. Machine Learning for Scattering Data: Strategies, Perspectives, and Applications to Surface Scattering. in preparation (2022).

174. Lee, J. & Jin, J. A novel method to design and evaluate artificial neural network for thin film thickness measurement traceable to the length standard. *Sci. Rep.* **12,** 2212 (2022).

175. Dax, M., Green, S. R., Gair, J., Macke, J. H., Buonanno, A. & Schölkopf, B. Real-Time Gravitational Wave Science with Neural Posterior Estimation. *Phys. Rev. Lett.* **127,** 241103 (2021).

176. Mareček, D., Oberreiter, J. & Kowarik, S. Artificial intelligence analysis enables fast in situ X-ray reflectivity measurements through sparse sampling. *J. Appl. Crystallogr.* submitted (2022).

177. Leemann, S. C., Liu, S., Hexemer, A., Marcus, M. A., Melton, C. N., Nishimura, H. & Sun, C. Demonstration of Machine Learning-Based Model-Independent Stabilization of Source Properties in Synchrotron Light Sources. *Phys. Rev. Lett.* **123,** 194801 (2019).

178. Chen, S.-W., Guo, H., Seu, K. A., Dumesnil, K., Roy, S. & Sinha, S. K. Jamming Behavior of Domains in a Spiral Antiferromagnetic System. *Phys. Rev. Lett.* **110,** 217201 (2013).

179. Shapiro, D. A., Yu, Y.-S., Tyliszczak, T., Cabana, J., Celestre, R., Chao, W., Kaznatcheev, K., Kilcoyne, A. L. D., Maia, F., Marchesini, S., Meng, Y. S., Warwick, T., Yang, L. L. & Padmore, H. A. Chemical composition mapping with nanometre resolution by soft X-ray microscopy. *Nat. Photonics* **8,** 765 (2014).

List of publications

1. Hinderhofer, A., Greco, A., Starostin, V., Munteanu, V., Pithan, L., Gerlach, A. & Schreiber, F. Machine Learning for Scattering Data: Strategies, Perspectives, and Applications to Surface Scattering. in preparation (2022).
2. Wang, C., Unger, F., Hiller, J., Hausch, J., Körbel, S., Hagara, J., Kneschaurek, E., Greco, A., Bertram, F., Takimiya, K., Hinderhofer, A., Gerlach, A., Meixner, A., Schreiber, F. & Broch, K. Packing-Structure-Induced Dramatic Transition of the Excited State Dynamics in the Solid State of Anthradithiophene and its Derivatives. submitted (2022).
3. Starostin, V., Munteanu, V., Greco, A., Kneschaurek, E., Pleli, A., Bertram, F., Gerlach, A., Hinderhofer, A. & Schreiber, F. Tracking perovskite crystallization via deep learning-based feature detection on 2D X-ray scattering data. *npj Comput. Mater.* **8,** 101 (2022).
4. Arora, N., Greco, A., Meloni, S., Hinderhofer, A., Mattoni, A., Rothilsberger, U., Hagenlocher, J., Caddeo, C., Zakeeruddin, S. M., Schreiber, F., Graetzel, M., Friend, R. H. & Dar, M. I. Kinetics of metal halide perovskite conversion reactions at the nanoscale. *Comms. Mats.* **3,** 22 (2022).
5. Greco, A., Starostin, V., Edel, E., Munteanu, V., Russegger, N., Dax, I., Shen, C., Bertram, F., Hinderhofer, A., Gerlach, A. & Schreiber, F. Neural network analysis of neutron and X-ray reflectivity data: automated analysis using mlreflect, experimental errors and feature engineering. *J. Appl. Crystallogr.* **55,** 362 (2022).

6. Greco, A., Starostin, V., Hinderhofer, A., Gerlach, A., Skoda, M. W. A., Kowarik, S. & Schreiber, F. Neural network analysis of neutron and x-ray reflectivity data: pathological cases, performance and perspectives. *Mach. Learn.: Sci. Technol.* **2,** 045003 (2021).

7. Baumeler, T., Arora, N., Hinderhofer, A., Akin, S., Greco, A., Abdi-Jalebi, M., Shivanna, R., Uchida, R., Liu, Y., Schreiber, F., Zakeeruddin, S. M., Friend, R. H., Graetzel, M. & Dar, M. I. Minimizing the Trade-Off between Photocurrent and Photovoltage in Triple-Cation Mixed-Halide Perovskite Solar Cells. *J. Phys. Chem. Lett.* **11,** 10188 (2020).

8. Greco, A., Starostin, V., Karapanagiotis, C., Hinderhofer, A., Gerlach, A., Pithan, L., Liehr, S., Schreiber, F. & Kowarik, S. Fast fitting of reflectivity data of growing thin films using neural networks. *J. Appl. Crystallogr.* **52,** 1342 (2019).

9. Greco, A., Hinderhofer, A., Dar, M. I., Arora, N., Hagenlocher, J., Chumakov, A., Grätzel, M. & Schreiber, F. Kinetics of Ion-Exchange Reactions in Hybrid Organic–Inorganic Perovskite Thin Films Studied by In Situ Real-Time X-ray Scattering. *J. Phys. Chem. Lett.* **9,** 6750 (2018).

10. Da Vela, S., Braun, M. K., Dörr, A., Greco, A., Möller, J., Fu, Z., Zhang, F. & Schreiber, F. Kinetics of liquid–liquid phase separation in protein solutions exhibiting LCST phase behavior studied by time-resolved USAXS and VSANS. *Soft Matter* **12,** 9334 (2016).

List of acronyms

6T α-sexithiophene 34, 68, 69, 71, 139, 143, 170

API application programming interface 133, 161

BMBF Bundesministerium für Bildung und Forschung 8, 102, 120, 133, 162

CNN convolutional neural network 123, 125, 134

CuPc copper(II)-phthalocyanine 34, 68, 69, 71, 139, 142, 170

DESY Deutsches Elektronen-Synchrotron 8, 101–103, 108, 120, 121, 133, 170

DIP diindenoperylene 6, 34, 68–71, 139–141, 170

DNTT dinaphtho[2,3-b:2',3'-f]thieno[3,2-b]thiophene 34, 170

ENL equivalent noise level 113, 115

ESRF European Synchrotron Radiation Facility 108, 170

FCNN fully-connected neural network 7, 42, 45, 63, 64, 72, 75, 131

FFT fast Fourier transform 101, 120

FT Fourier transform 15, 24, 25, 32, 119, 120, 163, 164

GPU graphics processing unit 4, 26, 83, 121

GT ground truth 86–91, 151–160

LC low-contrast 84, 85

LMS least mean squares 6–8, 63, 68, 69, 71, 72, 101, 103, 107–110, 116, 117, 121, 131, 133, 151

LT low-thickness 84, 85

MDN mixture density network 127, 134

ML machine learning v, 4, 5, 7–9, 25, 37–43, 46, 47, 56, 57, 66, 75, 93, 94, 97, 114, 121, 128, 129, 133–135

MLP multi-layer perceptron 42, 44

MSE mean squared error 6, 7, 41, 66, 83, 87, 89–91, 98, 106, 116–119, 133

NN neural network v, 4, 5, 7–9, 33, 39, 42, 43, 45, 47, 49–51, 58, 63–73, 75–77, 80–83, 86, 87, 89–93, 96–99, 101–103, 105, 106, 108–114, 116–118, 120, 122–129, 131–133, 139–143, 161, 169, 170

NR neutron reflectivity 5–7, 63, 75, 97, 104, 128, 132, 134

OSC organic semiconductor 6, 7, 34, 131

PDI perylene diimide 34

PDI-C8 N,N'-Dioctyl-3,4,9,10-perylenedicarboximide 116–118, 170

PDIF N,N'-1H,1H-perfluorobutyldicyanoperylene-carboxydiimide 170

PEN pentacene 34, 170

PyPI Python Package Index 102, 161

ReLU rectified linear unit 43, 44, 49, 64, 82

ROI region of interest 33, 103, 104

SAS small-angle scattering 5, 135

SGD stochastic gradient descent 9, 41, 46, 47, 50, 53, 55

Si silicon 6, 7, 34, 63, 67, 75, 101, 108, 116–118, 121, 122, 131, 170

SiO$_x$ silicon oxide 6, 7, 34, 63, 67, 75, 101, 108, 116–118, 121, 122, 131, 170

SLD scattering length density 5, 6, 14–17, 20–22, 24–26, 31, 32, 38, 65, 67–69, 71–73, 75, 76, 83, 86, 88–94, 97, 98, 101, 106, 108, 109, 117, 119, 120, 128, 134, 151, 154, 157–160, 163–165

SLS Swiss Light Source 108, 170

TOF time-of-flight 80

TRE total reflection edge 16, 17, 26, 51, 75, 80, 81, 89, 92, 93, 96, 98, 99, 105, 116, 117, 124, 125, 132

UHV ultra-high vacuum 9, 34, 35

XFEL X-ray free electron laser 5, 135

XRR X-ray reflectivity 5, 6, 16, 63, 67–71, 75, 77, 81, 97, 101, 108, 116, 121, 139