Bachelor's Thesis

Fast Scattering XRR Data Analysis Using Convolutional Neural Networks (CNNs)

Submitted by
Maximilian Hilbert

at the
Faculty of Science
Institute of Applied Physics
Eberhard-Karls Universität Tübingen

under supervision of
Prof. Dr. Frank Schreiber

in
September 2022

Eigenständigkeitserklärung

Hiermit erkläre ich,

– dass ich diese Arbeit selbst verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

– dass die eingereichte Arbeit weder vollständig noch in wesentlichen Teilen Gegenstand eines anderen Prüfungsverfahrens war.

Maximilian Hilbert

Tübingen, September 2022

Abstract

In recent years, the established methods of X-ray reflectometry have been systematically complemented by automated data analysis, which requires little to no prior knowledge about the nature of film parameters. Their ability to analyze data in real-time makes machine learning methods an interesting alternative to classical, recursive and thus slow procedures.

In this study, a comparison was made between the existing Multi-layer Perceptron (MLP) model [1] and an alternative Convolutional Neural Network (CNN) approach. For this purpose, new network architecture and adapted data preprocessing were established and model-specific parameters were optimized. The network was trained on X-ray reflectometry data, simulated with the *mlreflect* package and evaluated on simulated and experimentally recorded data.

In a 'proof of concept', the two models were first compared with respect to each other, using a (pseudo) test data set, enriched by statistical errors. Three film-specific parameters film thickness, surface roughness and Scattering Length Density (SLD) have been determined.

The results show, that the novel CNN approach is capable of predicting film-parameter out of synthetical XRR data. The comparison of both models on experimental (*in situ*) data shows that the parameters of film thickness and SLD can be predicted by the new approach with comparable uncertainties in the median and standard deviation. However, with respect to surface roughness, the formerly established MLP approach still shows better results in comparison to the presented approach in this thesis, both in the median and the corresponding standard deviation.

Deutsche Kurzzusammenfassung

In den vergangenen Jahren wurden die etablierten Methoden der Röntgenreflektometrie durch eine automatisierte Datenanalyse, welche wenig bis keine Vorkenntnisse über die Beschaffenheit der Film Parameter voraussetzt, systematisch ergänzt. Ihre Fähigkeit, Daten in Echtzeit auszuwerten, macht die Methoden des maschinellen Lernens klassischen, rekursiven und damit langsamen Verfahren gegenüber zu einer interessanten Alternative.

In der vorliegenden Arbeit wurde ein Vergleich zwischen dem existierenden Multilayer Perceptron (MLP) Modell [1] und einem alternativen Convolutional Neural Network (CNN)-Ansatz durchgeführt. Hierfür wurde eine neue Netzwerkarchitektur und eine angepasste Datenverarbeitung etabliert und modellspezifische Parameter optimiert. Das Netzwerk wurde auf mit dem *mlreflect* Paket simulierten Röntgenreflektometriedaten trainiert und auf ebenfalls simulierten und experimentell aufgenommenen Daten evaluiert.

In einem „Proof of Concept“ wurden zunächst die beiden Modelle hinsichtlich eines mit statistischen Fehlern angereicherten (pseudo) Test Datensatzes miteinander verglichen. Hierbei sind die drei filmspezifischen Parameter Filmdicke, Oberflächenrauigkeit und die Streuliniendichte bestimmt worden. Die Ergebnisse dieses Tests ergeben, dass das CNN Modell in der Lage ist, die genannten Parameter aus synthetischen Röntgenreflektometriedaten vorherzusagen.

Der Vergleich beider Modelle auf experimentellen (*in situ*) Daten ergibt, dass Parameter wie Filmdicke und Streuliniendichte im Median und der Standardabweichung vom neuen Ansatz mit vergleichbaren Unsicherheiten vorhergesagt werden können. Hinsichtlich der Oberflächenrauigkeit zeigt allerdings der bekannte MLP Ansatz sowohl im Median als auch der zugehörigen Standardabweichung bessere Ergebnisse im Vergleich zum hier vorgestellten CNN Ansatz.

Acknowledgements

I would like to thank all people that eagerly supported me throughout this thesis. Namely, I would like to thank all members of the Schreiber Group for sharing their knowledge and for the many meaningful discussions about details in programming and neural network design.

Especially thankful I am for the help of Linus Pithan and Alessandro Greco. Without their help, the development would not have been feasible.

Furthermore, I would like to thank the Schreiber Group for supporting my work, by providing the access to computational resources and infrastructure. I also thank DESY (Hamburg, Germany), a Member of the Helmholtz Association HGF, for providing access to the Maxwell computational resources.

Last but not least I want to thank Jasmin and my family for their continuing support throughout all challenges and successes in life and science.

Contents

Abstract **v**

Deutsche Kurzzusammenfassung **vii**

Acknowledgements **ix**

1 Introduction **1**

1.1 Neural networks vs. iterative approaches 1

1.2 Structure of this thesis 2

2 Scientific background **3**

2.1 X-ray scattering 3

2.1.1 Specular reflectivity 3

2.1.2 Scattering Length Density (SLD) 5

2.1.3 Rough surfaces 6

2.2 Conclusions from this chapter 8

3 Neural Networks for regression tasks **9**

3.1 Multi Layer Perceptron (MLP) 9

3.2 Convolutional Neural Network (CNN) 11

3.2.1 Convolution layer 11

3.2.2 Pooling layer 14

3.3 Loss functions 15

3.4 Backpropagation in neural networks 16
3.4.1 Gradient descent 16
3.5 Conclusions from this chapter 17

4 State of the art - Regression on XRR data 19
4.1 Parratt's exact recursive algorithm 19
4.2 Machine learning on XRR data using MLP models 20
4.3 Goal of this thesis 22

5 CNN approach on XRR Data 23
5.1 Data preprocessing and regularization 23
5.1.1 Noise application 23
5.1.2 Early stopping 25
5.1.3 Data standardization 25
5.2 High Performance Computing (HPC) 26
5.2.1 Hyperparameter search and model selection 26
5.2.2 Finding the optimal architecture 28
5.2.3 Conclusions from this chapter 30

6 Results and Discussion 31
6.1 Back-simulation and application of the Log-error 31
6.2 Proof of Concept: Performance on synthetic data 32
6.3 Model performance on experimental data 37
6.4 Conclusion and Outlook 41

A List of Acronyms 43

B Bibliography 45

Chapter 1

Introduction

1.1 Neural networks vs. iterative approaches

In the last years, machine learning emerged as an important field of research not only in every days life but also in the natural sciences. Particularly in the area of X-Ray Reflectivity (XRR) data analysis, machine learning is a tempting alternative to standard analysis approaches [1], usually pursued by using fitting algorithms based on the paratt or optical matrix formalism. [2]

This is the case, because it is possible to shift the computational cost from evaluation time towards training and therefore development, meaning that *in situ* applications of trained Neural Network (NN) models become significantly faster in analyzing real-time data than classical iterative methods.

Currently, data generation in synchrotron-beamlines is significantly faster than data analysis, the latter commonly using algorithmic approaches, leading to the issue of an emerging bottleneck in real-time data analysis. Regarding the previous work done in the field of analyzing XRR data using NN, it can be said, that CNNs promise to be an equally strong approach in comparison to the currently known MLPs.

This is assumed, due to the fact that CNNs have the possibility of detecting patterns in large amounts of data and taking the spatial distribution of values into account. Furthermore, the noise-resistance of CNNs are tempting in a physics context, because of statistical and systematical errors that are always present in experiments.

Although Machine Learning (ML) in general and Deep Learning (DL) in particular, are known since the 1950s, they were not capable of producing significant breakthroughs due to the lack of parallel computation power. In recent years, the possibilities have changed and there is access to High Performance Computation

(HPC) Clusters, suitable to process large amounts of data and compute the necessary mathematical operations in order to find minima of convex loss functions.

One should particularly mention the high degree of parallel processing power, which comes along when using Graphics Processing Units (GPUs).

In the following chapters, it becomes clear, that even for a bachelor's thesis on the topic of DL, it is necessary to have access to these computation resources, carrying the hyperparameter search or testing different network architectures.

1.2 Structure of this thesis

After the introduction in chapter one, the scientific background information about scattering and reflectivity patterns is given in chapter two. Close attention will be paid to the influence of the so-called thin film parameters on the shape of the reflectivity curve, due to the fact, that these parameters become vital information in the following regression problem.

This theoretical introduction is then followed by a technical chapter about the properties of NNs for regression problems in particular. MLP and CNN architectures and their individual components will be discussed in further detail, necessary to understand the architectural decisions that have been made later on.

In chapter 4 there will be a brief discussion about already published approaches to reflectivity data, most importantly the work.

In chapter 5, detailed explanations of the decision-making in the process of developing an alternative to recent publications will be given.

In the last chapter, the CNN's performance on simulated, as well as on experimentally recorded XRR data will be shown in comparison to the MLP model. Furthermore, the remaining optimization possibilities are outlined.

Note that the complete code base that has been established in this thesis is available under GitHub.[1]

[1] `https://github.com/MaximilianHilbert/Bachelor-s-Thesis-CNNs`

Chapter 2

Scientific background

Although X-ray scattering is a well-known phenomenon in soft matter physics, the implications on experimental setups are still of importance, especially when analyzing XRR curves. Due to the fact, that the shape of the curve appears to have a high impact on the performance of automated data analysis approaches the influence of different material parameters will be given in this chapter.

2.1 X-ray scattering

2.1.1 Specular reflectivity

As a first intuitive explanation of X-Ray Reflectivity (XRR), it is necessary to introduce the model of finite-thickness slabs, where the X-rays can either transmit or reflect from the surface or the interfaces, in the case of a multi-layer system. A finite-thickness sample typically consists of one or more layers and an underlying substrate.

The classical approach in order to explain the phenomenon of XRR is the scattering of an incident X-ray beam on electrons, where the far-field approximation holds. In such a case, the absolute value of the scattering vector is given by

$$|\mathbf{Q}| = 2k \sin\theta = \left(\frac{4\pi}{\lambda}\right) \sin\theta, \tag{2.1}$$

where k is the absolute value of the wave vector, λ the wavelength of the X-ray and θ the glancing angle.

For the scope of this thesis, this is the main definition and case of application. In an experimental setup, one can therefore transform an angle of incidence θ, for a

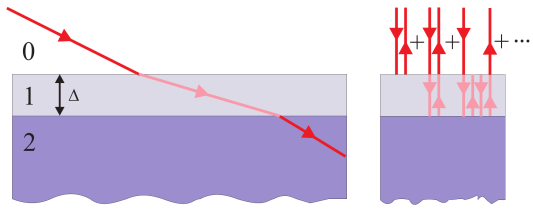


Figure 2.1: Model of a finite thickness slab, where the different layers are shown in the first half, transmission and reflection beams in the second half of the picture. Figure adopted from [3].

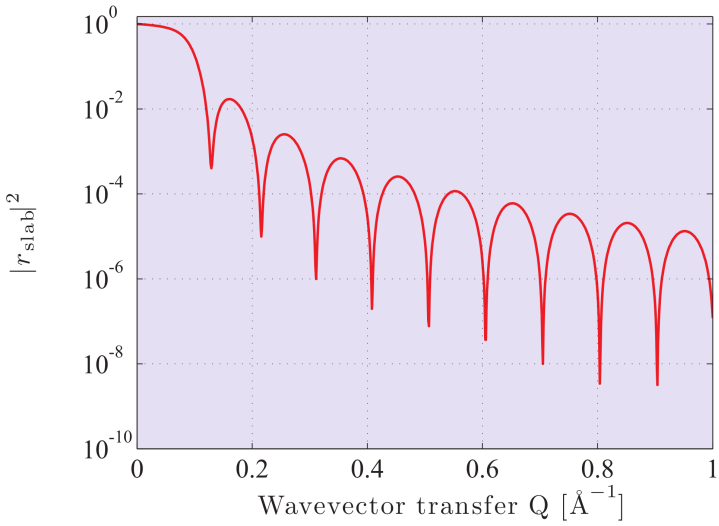


Figure 2.2: Clearly visible kiessig fringes in the reflectivity curve, measured against the wavevector transfer. Figure adopted from [3].

given beam of sufficient brilliance, to an absolute value of the scattering vector and vice versa. The partial reflectance of incoming beams on each interface leads to interference. This phenomenon can easily be seen in fig. 2.1.

As a result, the measured sum of reflections shows oscillations, commonly known as *kiessig fringes*, depicted in fig. 2.2. The defining condition for *specular* reflection that has to hold, is, that the angle between the incoming beam and the surface has to be equal to the angle between the outgoing beam and the surface. This is the case when no surface roughness is present. The layer thickness influences the XRR curve in a way, that the angular distance between two peaks is inversely proportional to the layer thickness. [3, 4]

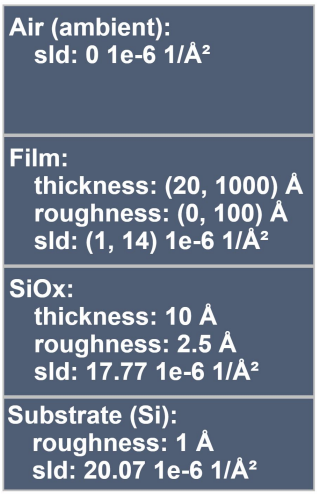


Figure 2.3: Box model used in this thesis. The ranges of the film layer are depicted by the brackets and are equal to the training ranges, used for the machine learning part of this thesis.

2.1.2 Scattering Length Density (SLD)

In the following work, the Scattering Length Density (SLD) is an important parameter. In general, one can imagine the SLD as the scattering power of a given material. In order to define SLD one typically starts with

$$n = 1 - \delta + \mathrm{i}\beta, \tag{2.2}$$

where n denotes the complex refraction index and δ, β are the real and imaginary parts, respectively. For the scope of this thesis, only the real part is of further interest.

Note that

$$\delta = \frac{2\pi\rho r_0}{k^2}, \tag{2.3}$$

where r_0 is the electron radius and $k = \frac{2\pi}{\lambda}$ the absolute value of the wave vector. For the case of XRR, the quantity ρr_0 is then defined as Scattering Length Density (SLD). [3]

Although the definition of SLD on neutron reflectivity is analogous to the given description above, one can derive from

$$n = 1 - \delta = 1 - \lambda^2 \frac{N_b}{2\pi}, \tag{2.4}$$

where

$$N_b = \frac{\sum_{i=1}^{N} b_i}{V_m}, \tag{2.5}$$

with b_i being the scattering contributions of the nuclei in the unit cell and V_m the corresponding volume. [5]

Note that the imaginary part in eq. (2.4) has already been omitted. From comparing eq. (2.4) by eq. (2.2), one can define Scattering Length Density (SLD) for neutron reflectivity finally as N_b, described in eq. (2.5).

One can state, that in the case of X-rays an increase in electron density of the material and therefore increase in SLD, the critical angle q_c, for which the total reflection occurs, shifts towards a higher value. From a heuristical standpoint, this behavior is clear; for higher electron densities X-rays have simply a higher possibility to reflect than they have to transmit the material.

2.1.3 Rough surfaces

Interfaces of real existing multilayer structures are not perfectly plane but rough to a certain extent. Surface roughness is another important parameter in the further analysis of XRR curves. In general, one can state, that the influence of surface roughness on the intensity curve is, that the envelope slope of the curve increases with increasing roughness. From a heuristical standpoint, this phenomenon is clear; due to the stronger unevenness of the material, the reflected beams become diffuse as shown in fig. 2.4. This leads to a stronger decrease in intensity, depicted in fig. 2.5. [3, 4]

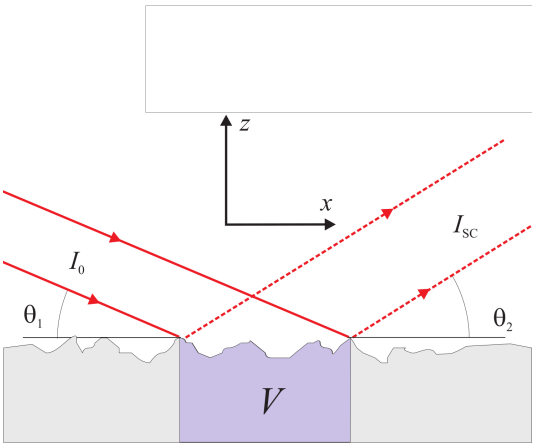


Figure 2.4: Incident beams scattered on a rough surface. Figure adopted from [3].

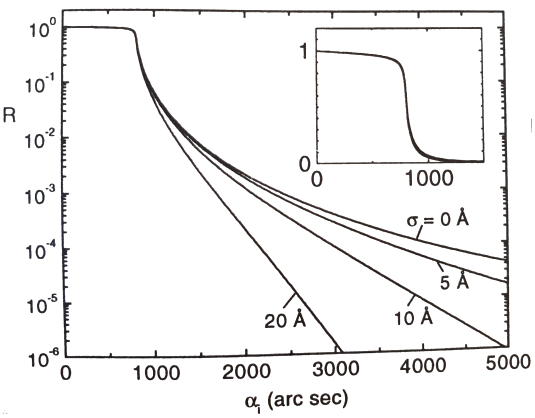


Figure 2.5: Shown influence of surface roughness on XRR curves. A higher amount of roughness leads to a fast descending intensity curve. Figure adopted from [4].

2.2 Conclusions from this chapter

In this chapter the film parameters thickness, roughness and SLD have been introduced, which are relevant for the scope of this thesis. The characteristic shapes XRR curves get by altering these parameters, is depicted in fig. 2.6. The main task of the CNN will be to detect these characteristic patterns in XRR data and infer film thickness, roughness and SLD.

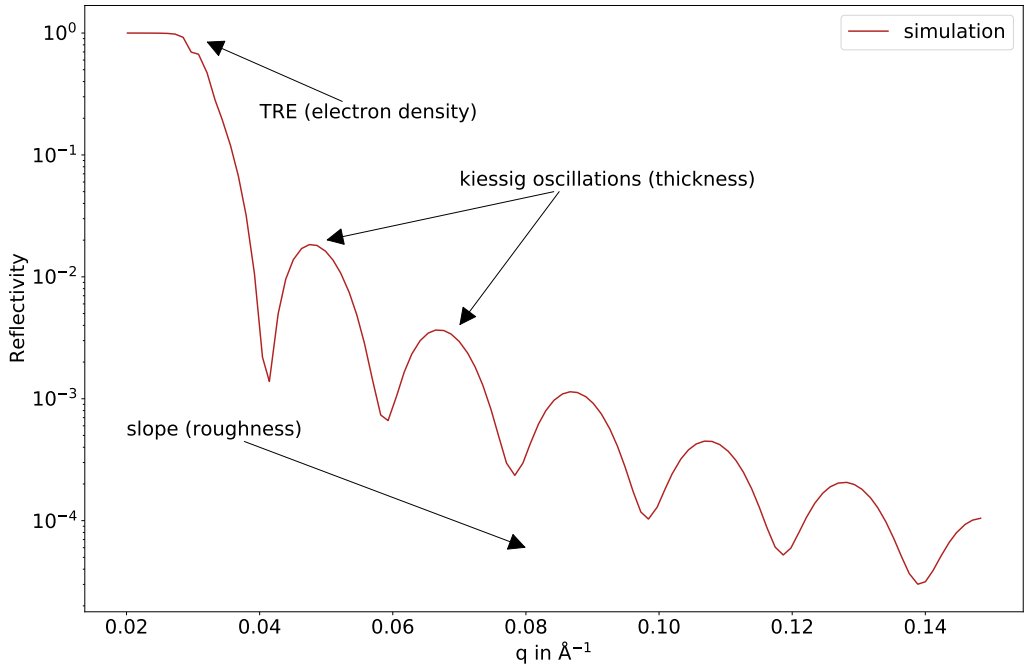


Figure 2.6: Curve features of a simulated XRR curve. The position of the Total Reflection Edge (TRE) is given by the electron density of the material, the angular distance of the kiessig oscillations is given by the inverse thickness of the material and the overall slope $\propto q^{-4}$, is given by the roughness.

Chapter 3

Neural Networks for regression tasks

In the last years, NNs varying in their architectures have been on the rise. Their extraordinary strength in detecting patterns in complex data, classifying objects and predicting with low computation time, have been proven in various data sets and problems. Especially CNNs became popular due to their capability of processing image data efficiently, leading to a significant acceleration of autonomous driving, where subsystems for object detection come into play in addition to other NNs on seriously power-limited computers onboard of the vehicle. Although CNNs are known for classification problems, they are also capable of solving regression problems in the field of soft matter physics and other sciences, where pattern recognition becomes a vital part. This chapter contributes to the theoretical foundations necessary to introduce MLPs and CNNs, respectively.

3.1 Multi Layer Perceptron (MLP)

In general, a MLP consists of one or more neurons per layer, that are able to process vector-shaped data. This architecture can be concatenated over the course of several layers, then denoted as *hidden layers* and followed by an output layer, shaped like the *targets* one wants to train the network on. In a *supervised learning* approach the training data is directly tied to the targets beforehand. The network's purpose is then to find a closed 'formula', typically references as a 'model', restoring the mapping between data and targets so that new data can be mapped to the desired output.

More formally, a MLP can be defined as a finite sum of activations a_j, obeying

$$a_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}, \tag{3.1}$$

where $x_1, \ldots, x_D$ are the input data and $j = 1, \ldots, M$. Note that M represents the number of succeeding hidden units the first layer has interactions with (see fig. 3.1). The superscript (1) denotes the first layer, $w_{j0}^{(1)}$ the bias parameter and $w_{ji}^{(1)}$ the weights, that get trained later on. See section 3.4.1 for the details on the training process. These activations will now be the input of an *activation function*. [6]

$$z_j = h(a_j). \tag{3.2}$$

In the case of a regression task, this activation function $h(\cdot)$ is typically chosen as $h(x) = \max(0, x)$, commonly known as ReLU (Rectified Linear Unit). This is a good choice for a variety of reasons, for instance, the faster training and good generalization capabilities in comparison to other functions.

Furthermore, ReLU has technical advantages, such as consistent gradients, as well as vanishing second-order gradients, that could potentially conflict with training. Further activation functions are not subject to discussion in this thesis but can become crucial, for instance when dealing with classification tasks. [6, 7]

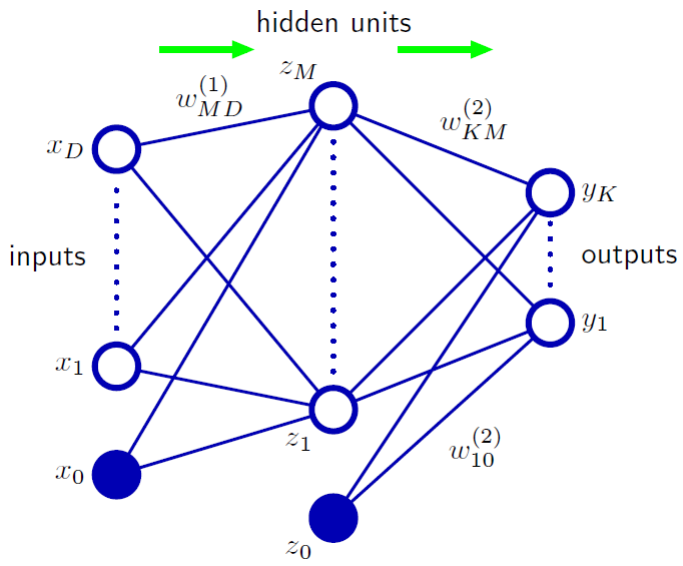


Figure 3.1: Graphical illustration of a feed-forward net that processes vector-like inputs $\mathbf{x}$ using a weight vector $\mathbf{w_M}$ to compute hidden units $\mathbf{z_M}$. Typically a concatenation of these operations with different weights $\mathbf{w_K}$ takes place until reaching an output. Note that for each layer an additional *bias parameter* is introduced, denoted as x_0 and z_0, respectively. Note furthermore that $y_1, \ldots y_K$ are continuous in the case of regression. Figure adopted from [6].

3.2 Convolutional Neural Network (CNN)

A Convolutional Neural Network (CNN) is a special architecture of DL models, usually known for its capabilities to classify images, heavily used in autonomous driving. This architecture has been mainly developed by Yann LeCun in the 1990s and is inspired by human brain cells, which seem to be ordered in a way that problems get chopped into several, easier-to-solve chunks, that get processed separately. [8]

In fact, solving a given regression problem is possible using a combination of CNNs and MLPs. For a detailed description of this architecture and how it is used in this thesis to solve the XRR regression problem, see chapter 5.

3.2.1 Convolution layer

In contrast to a MLP, the CNN architecture is not using matrix multiplications between all weights within layers and the input data but applies a mathematical function known as *convolution* to the input data, or the output of previous layers, in a multi-layer case. While from a purely mathematical standpoint the convolution

Figure 3.2: Classified obstacles recorded by eight cameras of the tesla autopilot vehicle and visualized by colored boxes. Screenshot adopted from [9].

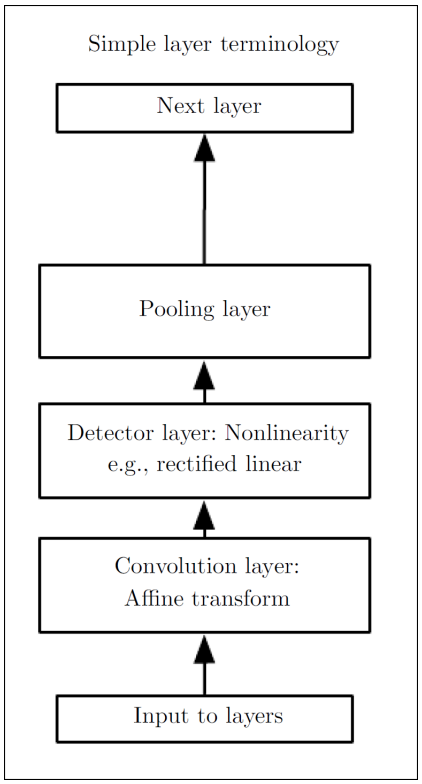


Figure 3.3: The terminology of a common CNN architecture used in this thesis. The individual stages will be discussed in the following. Note that the detector layer has already been outlined in section 3.1, which can be seen as similar to a MLP approach. Figure adopted from [10].

operation is defined as

$$s(t) \equiv (x * w)(t) = \int x(a)w(t-a)\mathrm{d}a, \tag{3.3}$$

where x, w are arbitrary functions so that the integral is properly defined. The definition in ML usually differs slightly, due to the fact that it is typically applied to discrete input data, like time-series, with discretized time steps in a 1D case, or images in 2D. Although the following definition is in principle extendable to higher dimensions, for the scope of this thesis, the 1D definition is sufficient.

In that regard, convolution gets defined in a discrete form via,

$$s(t) \equiv (x * w)(t) = \sum_{j=1}^{n} x(j)w(t-j), \tag{3.4}$$

where n denotes the total number of samples used as inputs to the convolutional function and j depicts the current value on the lattice.

In a CNN, convolutional operations in the sense of eq. (3.4) are performed by a concept known as *kernel*. In the 1D case, one can imagine a kernel as a sliding window, running across the vector-shaped input data. Each step of the sliding process then invokes a convolutional operation between the window and the data.

Due to the fact, that the kernel is always smaller than the input data, it, therefore reduces the spatial complexity of the input data significantly. The output of a kernel is commonly referred to as a *feature map*. As aforementioned, the feature map can now serve as input subject to the next convolutional layer or to a sampling method described in section 3.2.2.

It is important to underline the fact, that convolution performed by a kernel is also referred to as the process of *feature-extraction*. One can imagine the convolutional operation as computing the extent of 'overlap' between data and kernel, leading to the detection of patterns like edges, curvatures, etc.; given that the kernel has been trained sufficiently on data containing these structures. Note furthermore, that an important advantage of CNNs is, that the kernel gets trained, requiring fewer weights that have to be adapted, whereas single neuron weights have to be trained in the MLP case. A detailed explanation of the training process in regards to a CNN is given in section 3.4. [10]

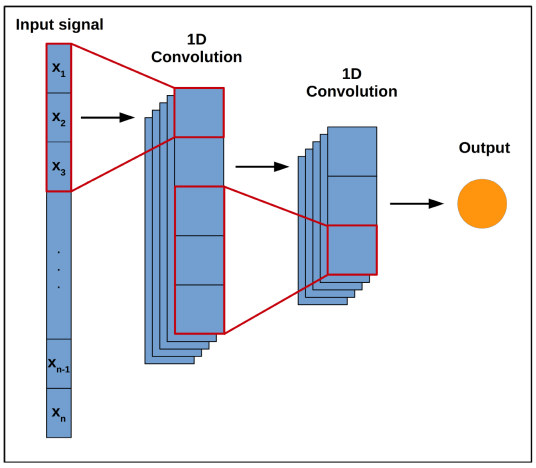


Figure 3.4: Graphical illustration of a convolutional operation performed between a kernel and the input data in 1D. Note that the degree of multiplicity the convolutions are performed in each layer, is known as the number of *filters* applied onto the data. Figure adopted from [11].

In general, a parallelized application of independent convolutions within one convolutional layer can be done, which is then referred to as multiple *filters* that have been applied to the data. A visual explanation of this process can be seen in fig. 3.4.

3.2.2 Pooling layer

Sampling methods are placed behind the convolution and detector stages and serve as a further reduction of content within the network. It essentially replaces data with a surrounding statistic that has been formed either as computing the maximum (*max pooling*) of neighboring data points or the average (*average pooling*).

The appealing concept behind any pooling operation is, that due to its nature, the input gets invariant to small distortions or shifts as well as missing data points. For a graphical example, see fig. 3.5. In general, one can state, that pooling operations lead to higher robustness against noise, due to an increase in the signal-to-noise ratio. This is the case as long as one can state, that the noise added leads to high values remaining high and low values remaining low, which is typically the case in physics. Although there exist more pooling functions, this lies outside of the scope of this thesis to discuss. [10, 12]

Despite the fact, that pooling is a key concept behind CNN models, one should always bear in mind, that pooling 'destroys' information in the sense that it filters by the strongest signals. Therefore, 'over-pooling' can cause a weak prediction capability of the model.

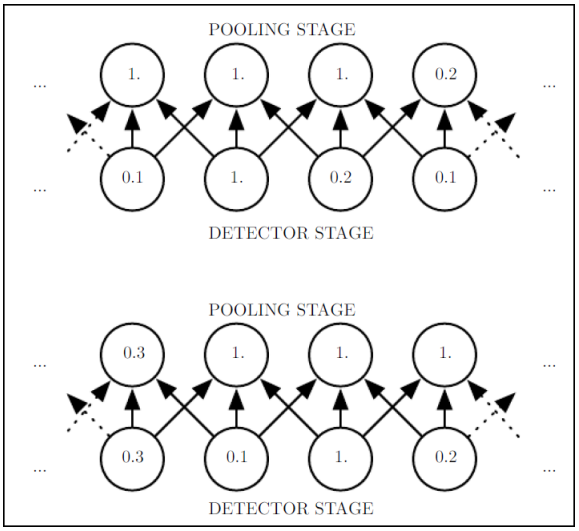


Figure 3.5: Top: Lower part shows the output of the nonlinearity, while the upper part shows the output of max pooling with a window size of three and a padding of one. Bottom: Outputs of the non-linearity got shifted, due to a shift in input data. The output of max pooling has only changed in half of its values, although the input has changed in all of its values. One can therefore see a CNN as invariant to small translations of the input data. Figure adopted from [10].

3.3 Loss functions

In order to introduce metrics that are suitable to measure the prediction capabilities of any NN and to find a suitable weight-adaption (see section 3.4), it is necessary to introduce the *loss function*. In a regression problem traditionally the Mean Squared Error (MSE) is a useful choice. It is commonly known as

$$\mathrm{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - y_i^p)^2 \,, \tag{3.5}$$

where y_i are the target values, y_i^p the target predictions and n denotes the number of data points. In general, the MSE is also applicable to multidimensional target values. The concept behind DL is to find a global minimum of the defined loss function by using the *gradient descent* algorithm, described in section 3.4.1.

As a further metric in the scope of this thesis, it becomes necessary to introduce the so-called Log error, which is typically applied in physics when calculating differences in logarithmic curves. It gets used in the application *GenX*[1], typically used for fitting XRR curves and available as open source. The Log error is defined

[1] `https://sourceforge.net/p/genx/git/ci/master/tree/genx/genx/fom_funcs.py`

as

$$\text{Log error} = \frac{1}{n}\sum_{i=1}^{N} |\log_{10} y_i - \log_{10} y_i^p|, \tag{3.6}$$

where similarily to above y_i are the target values, y_i^p the target predictions and n denotes the number of data points.

3.4 Backpropagation in neural networks

The principles described in section 3.1 and section 3.2 are commonly referred to as the *forward pass*, or the *prediction step* for obvious reasons. In the following the respective *backward pass* will be introduced, which leads to the training progress of the network and the adaption of weights to the given data set.

This iterative process will be continued over the course of multiple passes, called *epochs*. The threshold of maximum epochs is either set explicitly or dynamically, e.g. via *early stopping*, described in section 5.1.2.

In order to find the optimal combination of weights w_{ji} one wants to minimize the loss function, suitable to the given problem, introduced in section 3.3.

3.4.1 Gradient descent

The most common approach to minimizing loss functions is the iteration-based *gradient descent* method, which computes gradients for the whole training data set to achieve a weight update, as follows

$$\omega_{i+1} = \omega_i - \eta \nabla_{\omega_i} \mathcal{L}\left(\omega_i; \mathbf{x}, \mathbf{y}\right), \tag{3.7}$$

where ω_i is the weight at iteration step i, $\mathcal{L}(\cdot)$ the loss function, η the so-called *learning rate* and $\mathbf{x}, \mathbf{y}$ denote the entire data and targets, respectively.

Due to the fact that deep learning is mostly performed on big data sets, a single weight update after passing the entire data set is slow and inefficient. One can therefore expand the approach towards a noisier but faster and easier to parallelize version of gradient descent.

This method performs a weight update once a *batch* has been passed through the network. A batch is referring to a chunk of data, consisting of several data points, that the entire data set has been chopped into. In fact, the noisiness of the gradient

is even desired, due to the inherited regularization and the probability to skip local minima of the loss function.

One would then define *(mini)-batch gradient descent* as the following iteration

$$\omega_{i+1} = \omega_i - \eta \nabla_{\omega_i} \mathcal{L}\left(\omega_i; \mathbf{x}^{(j:j+m)}, \mathbf{y}^{(j:j+m)}\right), \tag{3.8}$$

where $\mathbf{x}^{(j:j+m)}$ represents a batch of m data points, $\mathbf{y}^{(j:j+m)}$ the respective target batch and η the learning rate.

As shown in the past, the learning rate should not be kept constant, on the contrary, it should be consequently lowered over the course of training. This can be easily achieved by using *ADAM*, a dynamic optimizer, whose mathematical details lie outside of this thesis.

3.5 Conclusions from this chapter

In the case of a regression problem on 1D data, one can state, that there are up- and downsides coming along with the different approaches of NN architectures. Although MLP models have the significant downside of being prone to the curse of dimensionality, this is not considered relevant in this context, because XRR data are always 1D and of a low enough footprint, so that the problem of exploding number neurons will not become a significant problem.

Model	Upsides	Downsides
MLP	proven functionality	high number of weights
CNN	noise-resistant utilizes spatial interactions lower number of weights	high number of hyperparameters still needs a MLP in last layer

Table 3.1: Overview of the up-and downsides of the different architectural approaches discussed in this thesis.

Chapter 4

State of the art - Regression on XRR data

Although regression analysis of XRR data is not a novel topic, it has been discussed since the 1950s and is still a subject of research. As [13] showed, it is possible to find an exact method in order to calculate reflectivity out of film parameters. The success of ML in this regard is mostly due to the faster execution time than the common, recursive approach.

4.1 Parratt's exact recursive algorithm

When the analytical approaches on calculating reflectivity out of thin film parameters get extended, it is possible to find an exact recursive method on the XRR problem. The general idea behind *Parratt's method* is, that on each interface of a multilayer structure the incoming beam splits into a transmitted and a reflected component.[3]

This means, that in the case of the N-th layer, the reflected beam transmits to the $N-1$ layer to a certain ratio and propagates further through the layer structure, from the bottom up. Hence, a method that includes all interface reflection and transmission events can be seen as an exact solution to the XRR problem. [3]

Furthermore, this recursive method can also be used in the case of XRR fitting, where the thin film parameters, which influence the XRR curves shape can be optimized as far as necessary to achieve a high enough accuracy between the fitted curve and the real data points.

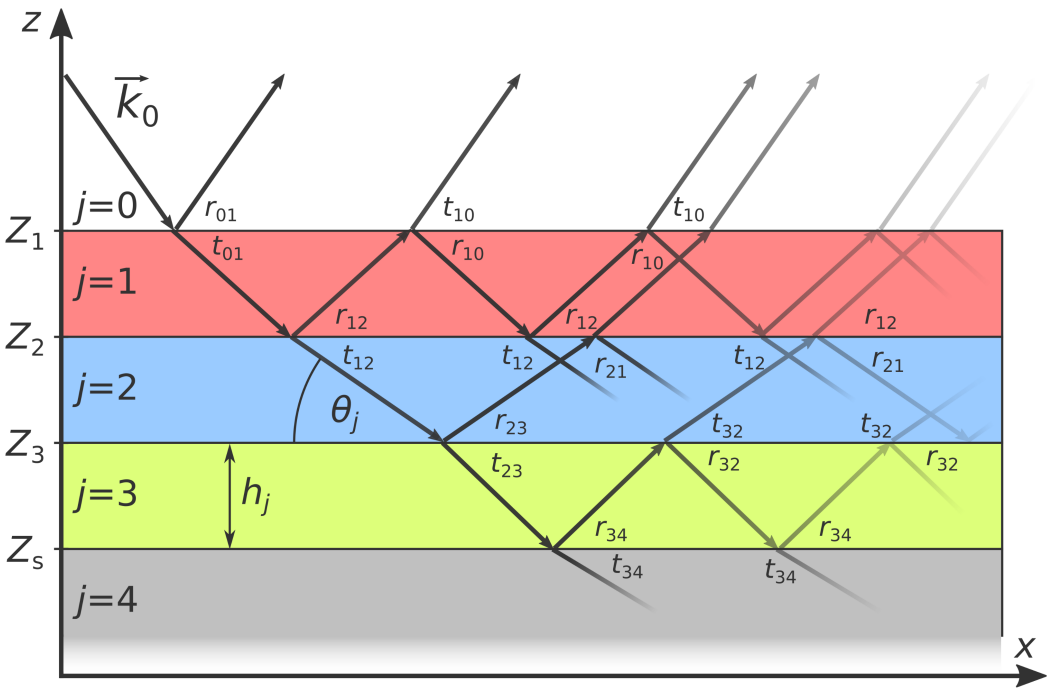


Figure 4.1: Multilayer system consisting of 3 identical thin film layers, with thicknesses h_j and an ambient layer ($j = 0$) as well as a substrate ($j = 4$). The r_{ji}, t_{ji} depict the reflective and transmissive components of the beams. Parratt's exact method calculated the sum of all possible pathways across the multilayer structure until the beams exits towards the detector. Figure adopted from [14].

Although the method is exact in a way that it takes into account every possible reflection/transmission event within the layer structure, it is computationally expensive. In fact, the computational load emerging from the calculation is $\propto N^2$.

This serious limitation can be eliminated by making use of modern ML methods, that are only applying a pre-trained function (model) onto the input XRR data, which maps directly to the thin film parameters of interest. Although ML models can include a high number of parameters (weights), it is still possible to reach execution times, that are necessary for *in situ* application. [14]

4.2 Machine learning on XRR data using MLP models

The current state of the Art on XRR data analysis, published in 2019, followed a fully-connected MLP approach as described in section 3.1, proven to reach a median test error of 7.1% for film thickness, 12.4% for film roughness and 6.8% for film SLD, measured on 242 experimental test curves. [15, 16]

This performance has been achieved by a variation of the model itself over the years, starting in 2019 with 6 fully-connected layers of different sizes, 3 layers with 1 000 units each and finally a 3-layer network with 500 neurons per layer in 2022. The latest model is considered the default model in this thesis and all comparisons are made with respect to this and the aforementioned experimental data set.

The network has been trained on simulated XRR data in the sample ranges of $(20, 1000)$Å, film thickness, $(0, 100)$Å film roughness and $(1, 14) \times 10^{-6}$Å^{-2} film SLD. The simulation of the training data was carried by the *mlreflect* package available on PiPy[1] in the q-range of $(0.02, 0.15)$ Å^{-1}.

Succeeding the simulation, uniform noise has been applied batch-wise to achieve a reasonable amount of regularization. In order to find a suitable noise level, the performance of the network with respect to the experimental test set has been computed for several noise levels. A minimum of the testing loss has been found for a noise level of 30%. [1, 16, 17]

In [1] it has also been shown, that the MLP approach is capable of performing real-time fits on *in situ* measurements, recorded on a beamline, leading to the vanishing of data generation and evaluation bottleneck, discussed in chapter 1.

In order to be able to compare the default MLP model with a new approach it is necessary to perform the same data simulation, data preprocessing and testing on the simulated as well as experimentally gained data set.

[1] `https://pypi.org/project/mlreflect/`

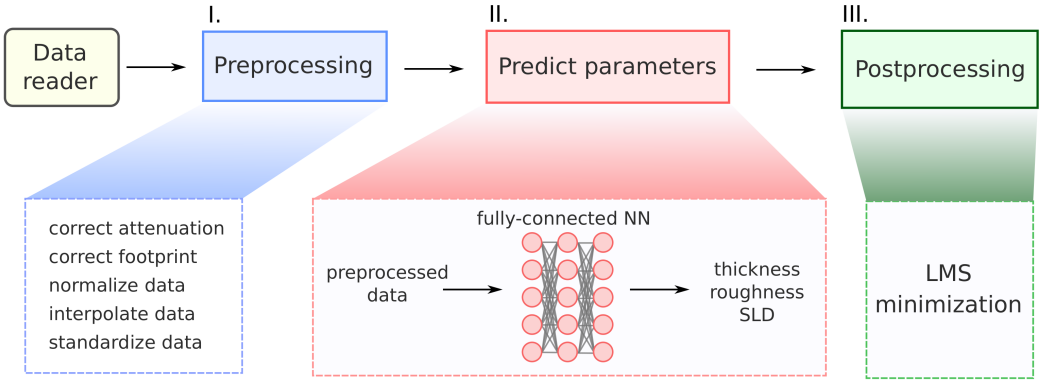


Figure 4.2: Analysis pipeline of the fully-connected default model. In step (I) the correction of the footprint of the experimental data, standardization and interpolation procedures take place. In step (II) the MLP model predicts the output parameters of film thickness, film roughness and film SLD. In step (III) the following LMS fit gets started, using the predicted parameters as initial values. Figure adopted from [1].

4.3 Goal of this thesis

The goal of this thesis is to diversify the number of possible architectures, used for fast scattering data analysis further. In an optimal scenario, the application of using CNNs on XRR data comes to the conclusion that a higher performance on a broad experimental data set can be achieved, possibly due to the higher noise robustness of the CNN compared to the default MLP model.

Chapter 5

CNN approach on XRR Data

In this study, a CNN approach on XRR data is being performed. The following chapter describes in detail the necessary steps in order to establish a CNN on simulated training data, containing the steps of preprocessing, training and testing on synthetic and experimental data, respectively.

5.1 Data preprocessing and regularization

The issue of preventing any NN from overfitting, is one of the most defying tasks in machine learning, especially when using synthetic training data, or data that does not depict the real-world distribution in a sufficient way.

Therefore, *preprocessing* is a commonly used technique to handle this challenge in multiple possible ways. Using purely synthetic (simulated) data for training lacks statistical and systematical errors that are always included in real-world data, whereas (experimental) test data contain these deviations. In general, one could call a NN, purely trained on synthetic data as overfitted, due to its lack of generalization capability on the test set.

5.1.1 Noise application

One possibility to prevent the NN from overfitting on synthetic data is introducing a preprocessing layer, which applies statistical errors to the training data. The distribution to choose statistical errors from has been broadly discussed in [17].

For the scope of this thesis, the application of uniform noise is of special relevance, due to the fact that it pushes the training data towards containing high statistical errors, hence important edge cases are taken into account. This is in contrast to

gaussian noise, where small statistical errors are more probable than high ones. The application of noise has been established following

$$R_i^{noise} = R_i \cdot U_{[1-p,1+p]}, \tag{5.1}$$

where R_i are the individual data points of the reflectivity curve and $U_{[1-p,1+p]}$ is the uniform distribution, characterized through the noise level p.

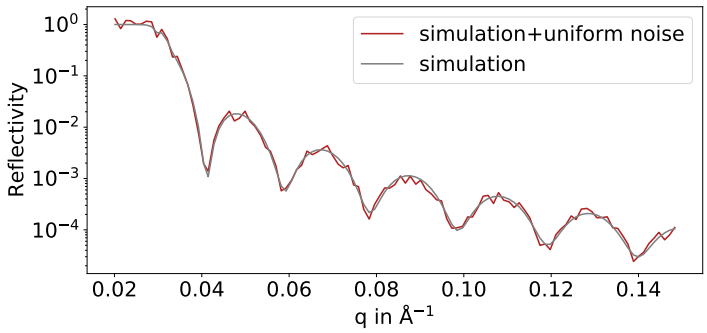


Figure 5.1: Uniform noise applied on a single training curve, which has been drawn from the training set. As a simulation toolkit *mlreflect* has been used. Note that the training ranges for the sample parameters, as well as the q-range, have been adopted from section 4.2.

5.1.2 Early stopping

In order to prevent any kind of NN from skipping a possible global minimum of the loss function and therefore leading to a high validation or testing error, one traditionally aborts the learning process after the validation loss per epoch stops descending over the course of several epochs. This procedure is referred to as *early stopping* and is commonly used across all DL techniques.

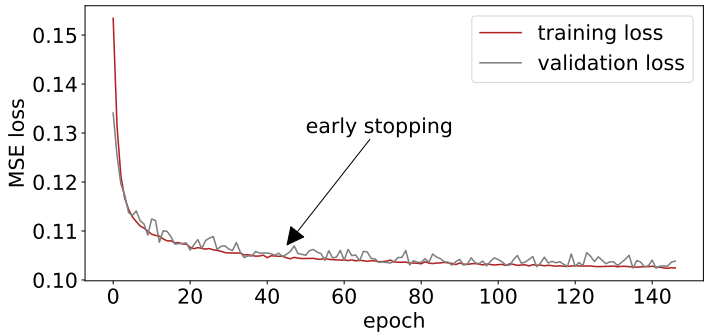


Figure 5.2: Demonstration of early stopping aborting the training of the CNN, due to a stopped decrease of validation loss for three consecutive epochs. Training for more epochs generally leads to an increase in validation error due to an overfit to the data. Note that in this specific case, early stopping detects a peak in the validation error, leading to the problem, that the validation error would still be possible to decrease. Tuning the 'patience' parameter, which describes the number of epochs the validation error has to rise until the training is ended, could be adapted to solve this problem and find an even lower validation error.

5.1.3 Data standardization

In many cases where deep-learning-based regression is applied, the underlying data set used, occupies a largely spread interval, sometimes several orders of magnitude in relative difference. As shown in previous chapters, the underlying XRR data set is of such shape, as well as the target parameters film thickness, roughness and SLD.

Therefore, the input data is typically standardized before training. For this study, the solely relevant standardization is the commonly known *Z-Score*, where the input interval is mapped to a region with zero mean and unit standard deviation. This procedure leads to more stable and faster training performance. For further information on the details behind data standardization see [14, 17].

The Z-Score used in this thesis on input and output data of the network is given by

$$\hat{\mathbf{x}} = \frac{\mathbf{x} - \mu}{\sigma}, \tag{5.2}$$

where $\mathbf{x}$ denotes the data vector, $\hat{\mathbf{x}}$ the standardized data vector, μ the mean and σ the standard deviation of the original distribution.

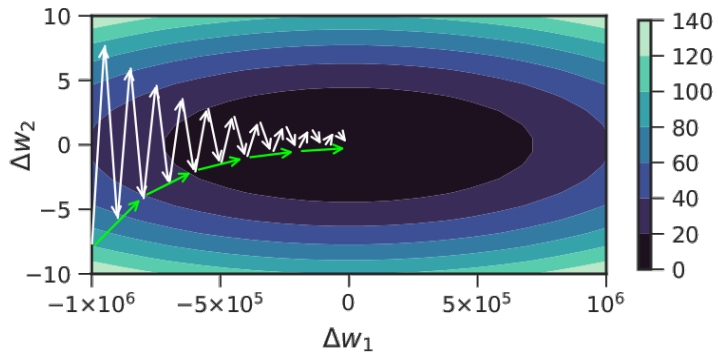


Figure 5.3: Contour plot of optimization steps taken by the optimizer on data spread across several orders of magnitudes. Due to the higher absolute values of Δw_2 the optimizer steps over the optimal solution several times, prior to reaching the minimum, leading to slow convergence. The optimal path is depicted in green. This visualization shows the importance of standardizing the data set. Figure adopted from [14].

5.2 High Performance Computing (HPC)

5.2.1 Hyperparameter search and model selection

In order to build a CNN model, that is suitable for solving the task of mapping XRR curves to the parameters of interest, it is important to find a sufficient set of *hyperparameters* for which the testing loss is in a minimum. In this study, the hyperparameters of interest are the number of filters in each convolutional layer, the size of those and the batch size. A detailed description of the influence of those has been given in section 3.2. The tested hyperparameter combinations are shown in table 5.1. To finally find the optimal combination, copies of the same model had to be trained, preferably in parallel.

In this study, the computational power of the *maxwell*[1] High Performance Computation (HPC) Cluster at DESY, Hamburg has been used to achieve a sufficiently fast training. To reach compatibility of *mlreflect* and the custom training pipeline, a *containerization* solution has been implemented, containing a Linux operating system (OS), *mlreflect* (incl. TensorFlow), the custom code and architecture, subject to this thesis.

[1] `https://confluence.desy.de/display/MXW/Maxwell+Cluster`

Hyperparameter	Sets/Ranges tested
Number of filters 1, 2, 3	$\{5-70\}^{(5)}$, $\{10-70\}^{(5)}$, $\{15-70\}^{(5)}$
Filter sizes 1, 2, 3	$\{4-10\}^{(1)}$, $\{3-9\}^{(1)}$, $\{2-8\}^{(1)}$
Batch size	$\{1024, 512, 256, 128\}$

Table 5.1: Hyperparameter sets tested in this study. Note that the superscripts (1), (5) indicate the step size. The optimal combination of hyperparameters has been found by applying the model finding process described in fig. 5.4. The total number of tested hyperparameters is 1 482 624.

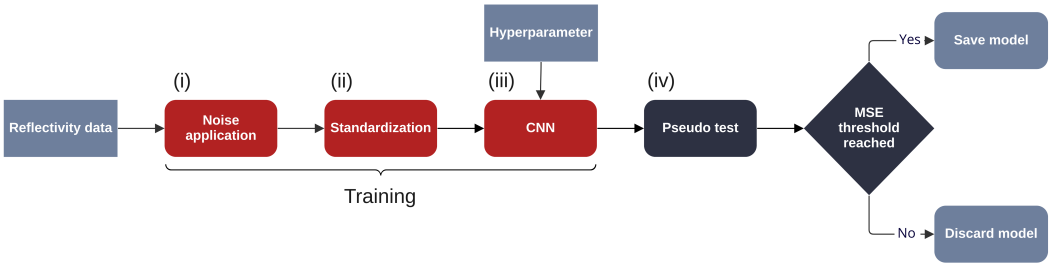


Figure 5.4: Flow chart of the training and evaluation pipeline established in this study. In step (i) a batch-wise application of uniform noise, described in section 5.1.1, is performed. In step (ii) standardization is executed, according to the rules described in section 5.1.3. In step (iii) the CNN processes the data, using one permutation of the hyperparameters given in table 5.1. With respect to the predictions on the pseudo test set, the MSE loss is then calculated in step (iv). Whether the MSE loss is below a certain threshold, the model gets saved. In a succeeding step the minimum MSE across all saved models gets selected, in order to find the optimal model.

The procedure then follows the main goal of parallelization, meaning, that the training data, as well as a pseudo test set, is generated once and saved to disk. In the case of training, there is no noise applied at this stage. This is the case because the noise gets applied batch-wise. In the case of the pseudo-test set, noise is applied directly to the data.

In the following, multiple computation jobs get submitted to the batch management system, each of which consists of a single hyperparameter permutation. After training, the script executes a pseudo test on the saved (noisy), but also simulated, data and saves the performance metric to a log file. In the following step, the minimum pseudo-test loss gets selected, leading to a model with optimal hyperparameters for the given problem. This pipeline is shown in fig. 5.4.

5.2.2 Finding the optimal architecture

As a starting point for developing a functional model that solves the XRR problem, the parameter ranges for training, described in section 4.2 were significantly narrowed down. Due to the fact that the pseudo test set has also been adapted to these parameters, the model is expected to perform generally better. From a heuristic standpoint, this behavior is obvious. In order for the model to adapt to XRR curves that have similar shapes, the necessary amount of weights to solve this problem is significantly reduced.

This procedure has been established, because of the fact, that it is easily possible to overfit a model by accident, due to the usage of too many layers, in turn introducing an unnecessary amount of complexity to the problem. With the presented iterative approach, graphically described in fig. 5.5, this issue can not arise. Furthermore, for every point in time, it is secured that the training pipeline described in fig. 5.4 is functional and free from programming errors or a failing standardization.

Although this procedure is reasonable for following a minimum-complexity approach and is therefore also computationally efficient, it is important to keep in mind, that performance on a synthetic test set is still significantly different from an experimental performance.

After refining the model in a sense that, the prediction capability on a simulated test set with training ranges as described in section 4.2 met the expectations, the final model gets tested on the experimental data set, whose performance metrics are discussed in section 6.3.

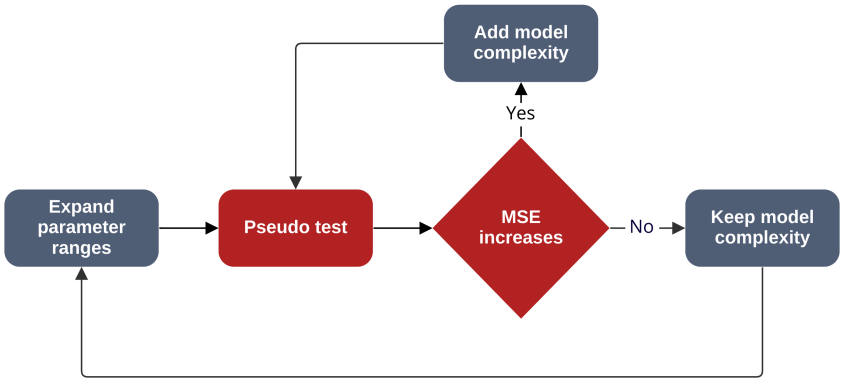


Figure 5.5: Determining the model complexity by an iterative process of expanding the training ranges, testing the model and checking if the MSE performance decreases, the model gets either more complex by adding layers or adapting the hyperparameter search sets (see table 5.1), or the model remains in its current complexity level. This process is repeated until the parameter ranges are identical to the published values, described in section 4.2. Note that in this iterative process no experimental data set is involved and the model 'sees' the final test set only on evaluation.

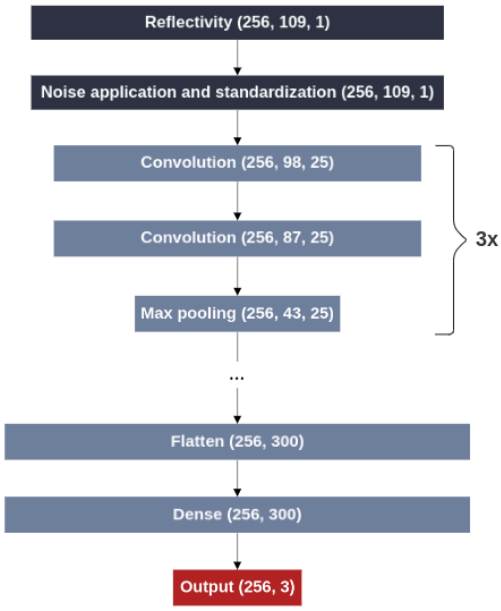


Figure 5.6: Architecture of the final 6-layer model, using a combination of convolutional layers, max pooling and a single dense layer before the final output, carrying the three parameters film thickness, roughness, and SLD. Note that the corresponding input shapes are three-dimensional tensors, where the first dimension corresponds to the batch size, the second to the length of the curve and the third to the filter size. The flatten layer's purpose is to reduce the dimensionality of the convolutional output to a 1D vector. Note that the lengths of the beams are not to scale and the third dimension of the input layer is existing only for technical reasons. Note furthermore, that the total number of trainable parameters ends up being about 167 000.

5.2.3 Conclusions from this chapter

In this chapter, the training pipeline, the model selection process and the hyper-parameter training have been shown. Furthermore, the importance of HPC-power has been demonstrated as well as the importance of trial-and-error when finding the optimal model, suitable for a given regression problem. Although one could argue that this procedure is inefficient, however, it is the only known way to find models that work and can be evaluated, which is shown in the following chapter. This chapter showed also how important the details in a training pipeline can be, such as input data standardization or regularization.

Chapter 6

Results and Discussion

As a proof of concept, the refined model, described in fig. 5.6 gets tested on a pseudo test set that has been simulated and has uniform noise applied. This performance proves the basic functionality of the model during development and shows, that with a certain amount of regularization, synthetic XRR curves can be subject to a functional prediction of film parameters. As a final step, the prediction capability of the established model gets evaluated on the experimental data set of 242 XRR curves, described in section 4.2.

6.1 Back-simulation and application of the Log-error

In general, there are alternative loss functions for measuring the performance of a regression problem that have not yet been covered in this thesis so far. A possible second performance metric would be applied on the back-simulated XRR curves that emerge out of the predicted film parameters and the sample structure. The advantage of a metric that gets directly applied to curves is that there is a clear and visible comparison between prediction and ground truth.

Furthermore, the nature of a 'fit' as typically used in physics becomes obvious. In order to achieve such an additional performance metric, one has to establish an additional pipeline that processes the predictions of the CNN and utilizes *mlreflect* to back-simulate the corresponding XRR curves. Once this is done, one can measure the prediction performance also with the Log error, introduced in section 3.3.

Although trying to find a visible comparison seems tempting, it is not obvious that the Log error on back-simulated curves is a valuable performance metric. This is mostly the case, because of an ambiguity in the curves emerging out of similar film

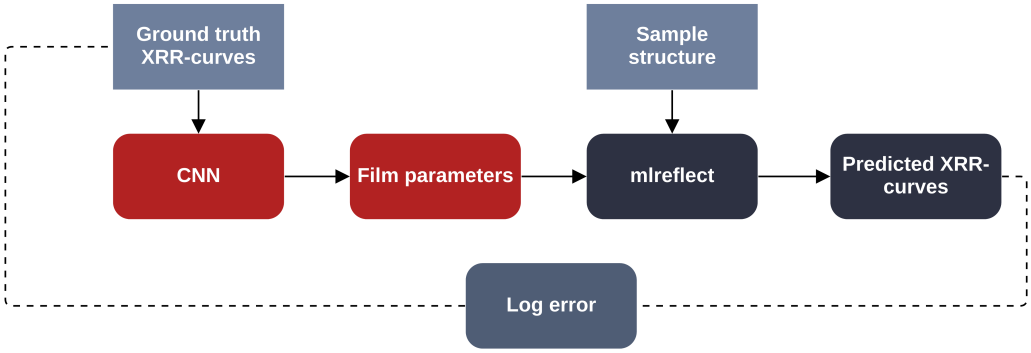


Figure 6.1: Back-simulation pipeline established in this thesis to achieve a direct comparison between ground truth XRR curves and prediction of film parameters. As a first step (not shown in the chart) *mlreflect* is used to simulate XRR curves and noise gets applied. Then the CNN model (see section 5.2.2) predicts the film parameters. In a succeeding step, *mlreflect* is utilized to simulate back the XRR curves, making use of the sample structure (see section 2.1.1) and the predicted film parameters. Note that there are cut-off errors and simulation-specific uncertainties introduced in this step, although these are considered marginal. In a last step the Log error (see section 3.3) is calculated.

parameters, thus a different set of parameters can lead to similar back-simulated curves. This is true for edge cases, but still relevant for statistical reasoning. Hence, a comparison between those curves as a performance metric can not hold alone.

In general, one could argue, that the problem of extracting film parameters out of XRR curves without prior knowledge, is highly dependent on the goal that has been defined. More precisely, one has to define, whether one is interested in a correct set of film parameters, or exactly aligned back-simulated curves. Because of the dependency on human preference, both performance metrics will be shown in the following chapter for the pseudo test set, as well as for the experimental test set.

6.2 Proof of Concept: Performance on synthetic data

As the first step in order to determine the functionality of the CNN model, a performance test on synthetic (noisy) data has been carried out. Based on the metrics this 'pseudo' test delivers, one can estimate the degree of overfitting with respect to the training data. In case of no regularisation, the synthetic data are predicted with extremely low errors, however, generalization might be on a poor level. With a uniform noise level of 30%, one can achieve comparable performance with the default MLP model, with respect to the single absolute errors over the three parameters of interest, see fig. 6.3, as well as on the performance metrics, calculated over all three parameters, see fig. 6.2a and the back-simulated curves, see fig. 6.2b.

Model	Thickness (in Å; %)	Roughness (in Å; %)	SLD (in 10^{-6}Å^{-2}; %)
MLP	(27.02; 8.29)	(2.62; 11.10)	(0.44; 6.05)
CNN	(101.82; 22.77)	(10.22; 24.51)	(0.28; 3.72)

Table 6.1: Results of the pseudo test for the MLP and CNN on the same data, in absolute and relative errors.

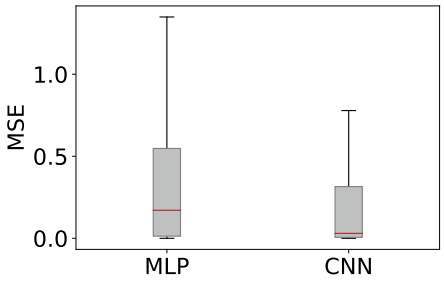
Model	MSE	Log error
MLP	0.171	0.028
CNN	0.030	0.034

Table 6.2: Peformance metrics for the MLP and CNN pseudo test, once on the predicted parameters (MSE) and on the back-simulated XRR curves (Log error), both introduced in section 3.3.

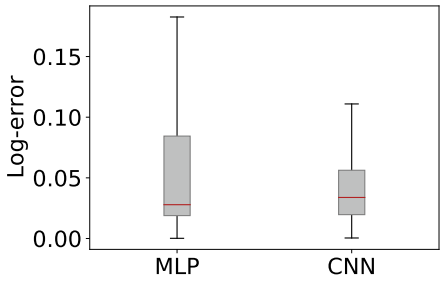
Interestingly, in a direct comparison between the two performance metrics (MSE over the parameters in fig. 6.2a and Log error on the back-simulated curves in fig. 6.2b) one can see a significant difference in interpreting the performance.

The Log error seems to be in favor of the MLP model, while the MSE is significantly lower for the CNN model. Although the standard deviation of both metrics appears to be smaller for the CNN. The different outcomes of the performance metrics could be the case, because of the ambiguity problem, discussed in section 6.1.

In general, one can state, that the CNN model is capable of predicting film parameters out of synthetic XRR data with comparable errors to the default MLP model. In parts, this performance can even be interpreted as higher than the MLP model, especially when focussing on a narrow error distribution. The detailed performances are shown in table 6.1 and table 6.2, respectively.



(a) Comparison of MSE performance for all three parameters calculated on 10 240 synthetic reflectivity curves between the default MLP and the CNN model.



(b) Log-error on 10 240 synthetic test XRR curves whose film parameters got predicted by the MLP and CNN model and were back-simulated by *mlreflect*.

Figure 6.2: Performance metrics of both models on simulated test data.

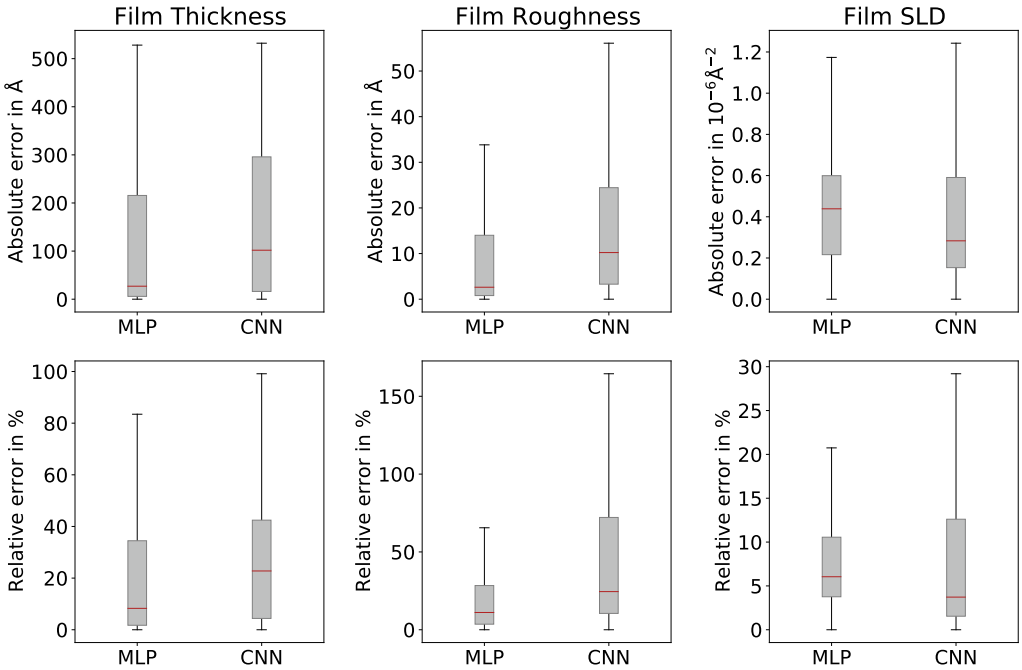


Figure 6.3: Comparison of absolute and relative absolute error between the predictions of the CNN and the MLP model on the pseudo test set.

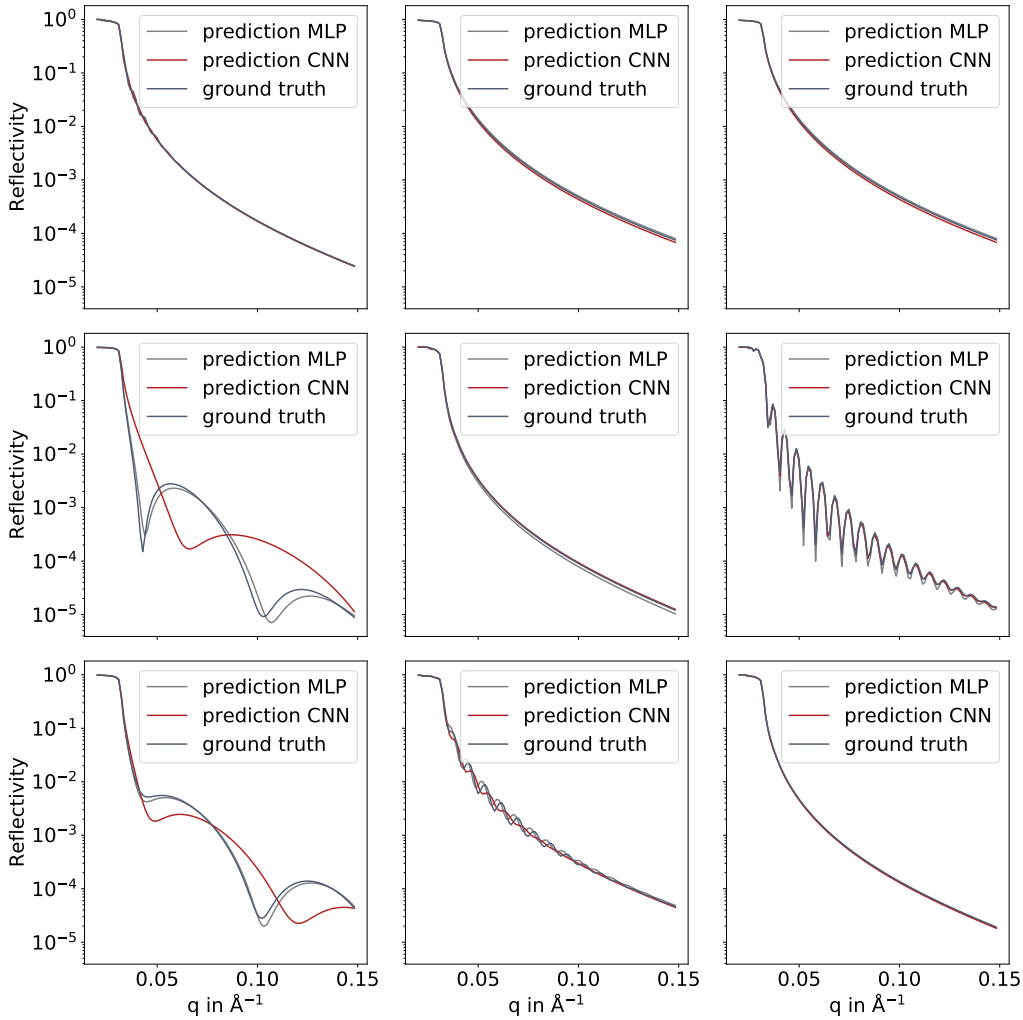


Figure 6.4: Comparison of the back-simulated XRR curves out of the predictions from the MLP and CNN models, respectively, against the ground truth. Note furthermore that the fit of both models is extraordinarily precise on almost all sample curves. Note that the pseudo test set does not contain noise, but the CNN has been regularized using a 30% uniform noise level.

6.3 Model performance on experimental data

The final test, to achieve a resilient comparison between the MLP and CNN model, is performed on 242 experimental XRR curves, identical to the data set introduced in section 4.2. To be compatible with the trained model, which has an input of 109 data points per curve, one has to make use of an interpolation pipeline, that corrects the footprint of differently gained experimental test curves. At this point, again, the *mlreflect* package has been used to achieve this correction.

As depicted in fig. 6.5a, one can state that the CNN model is capable of achieving slightly lower median MSE values than the MLP model, although with a slightly higher standard deviation. On the Log error, depicted in fig. 6.5b, the CNN's median lies higher than the MLP's, although here the standard deviation of the CNN is significantly smaller.

Model	Thickness (in Å; %)	Roughness (in Å; %)	SLD (in 10^{-6}Å^{-2}; %)
MLP	(6.42; 8.64)	(1.73; 9.68)	(0.79; 7.53)
CNN	(7.83; 8.44)	(3.23; 18.76)	(0.69; 6.48)

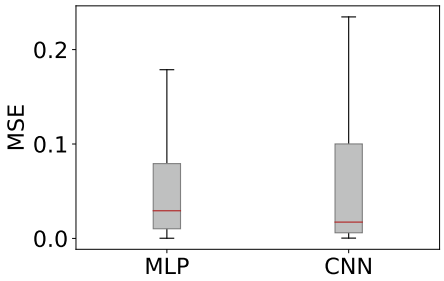
Table 6.3: Results of the final test on 242 experimental XRR curves for the default MLP and the CNN model in absolute and relative errors.

Model	MSE	Log error
MLP	0.029	0.149
CNN	0.017	0.175

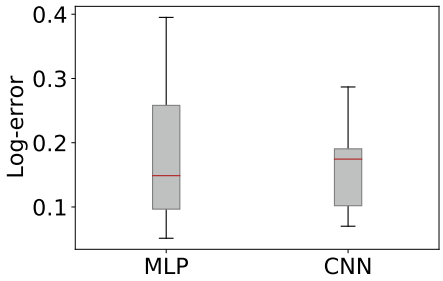
Table 6.4: Performance metrics of the final test on 242 experimental XRR curves for the default MLP and the CNN model, on the predicted parameters (MSE) and on the back-simulated XRR curves (Log error), both introduced in section 3.3.

Splitting up the general statistical statements into the singular absolute errors, with regard to the three film parameters, one can find the distributions shown in fig. 6.6. One can state, that the CNN is capable of solving the XRR problem for film thickness and SLD. This is due to the fact, that the error distributions overlap and the median values are comparable to the default model.

On the other hand, one has to state, that the parameter of film roughness is only predicted with a high median error, as well as a higher standard deviation than the default approach appears to have.



(a) Comparison of MSE calculated over all three parameters, measured among the predictions emerged of 242 experimental XRR curves between the MLP and CNN model respectively.



(b) Log-error on 242 experimental XRR curves whose film parameters were predicted by the MLP and CNN model, respectively and back-simulated by *mlreflect*.

Figure 6.5: Performance metrics of both models on experimental data.

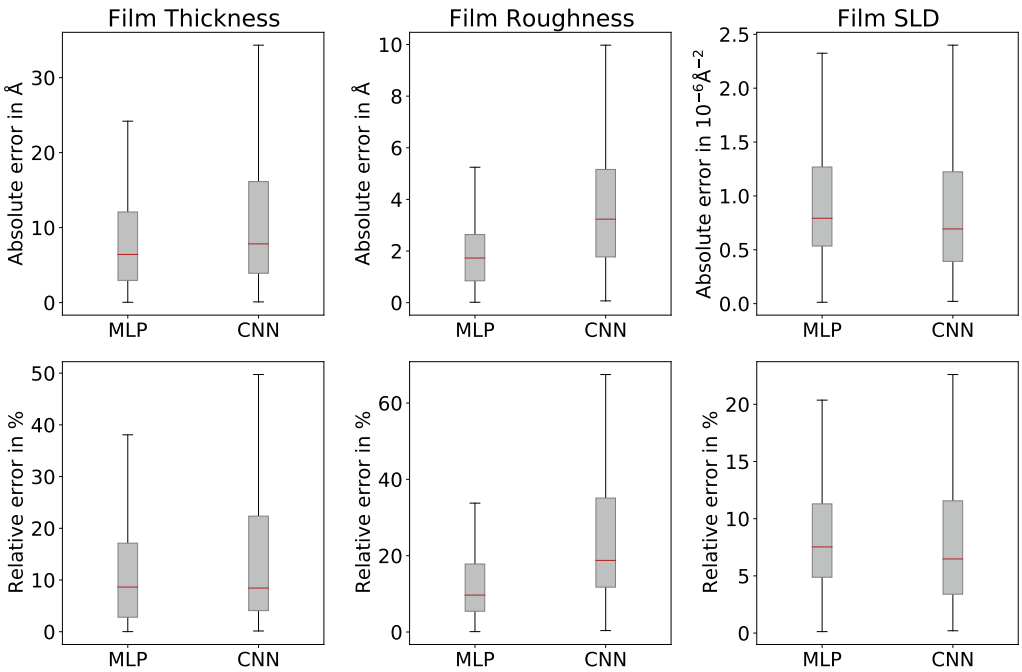


Figure 6.6: Comparison of absolute errors between the predictions of the MLP and CNN on the experimental data set, containing 242 XRR curves. Note that the CNN has been trained on a noise level of 30%.

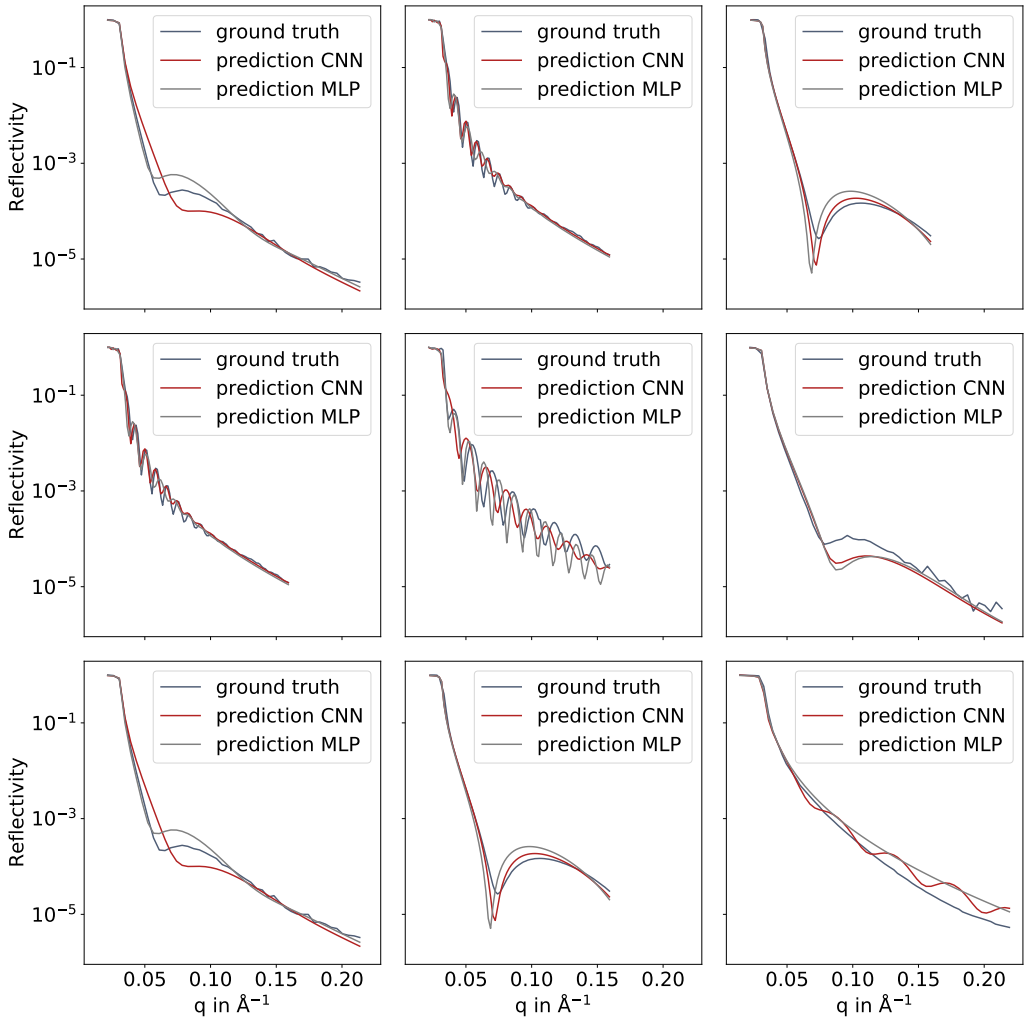


Figure 6.7: Comparison between the ground and back-simulated curves, that emerge from the predicted parameters.

6.4 Conclusion and Outlook

In general, one can state, that the approach of using CNNs on solving the XRR problem is a viable alternative to the MLP problem. On the parameters of film thickness and SLD, comparable results to the default MLP model have been achieved. Although two out of three parameters reach similar performances, the CNN model is not capable of finding a well enough fit for the film roughness. One can interpret this behavior in a sense, that the MLP model is using approx. $500\,000$ and the CNN model only approx. $170\,000$ parameters, leading to a major difference in model capacity.

Although this phenomenon is known for comparisons between these architectures, it is a probable cause for the lower prediction capability the CNN reaches. This leads to the explanation, that the MLP model already learned all content being present in the data. Furthermore, it is possible that the used pooling functions applied in the CNN approach, are destroying information to a higher extent as intended.

Although there has been much effort invested in the simulation of synthetical XRR curves, there are still significant shortcomings associated with that approach. For instance, it is problematic to train on manually set training ranges, because of the emerging lack of generalization beyond these ranges. An ideal case would be to train on thousands of training curves, each recorded in a real-world environment, that span the whole region of parameters, one could encounter in a real-world application. Of course, due to the expensive environment, where XRR curves are typically recorded, a massive amount of curves is not easy to gather. Furthermore, experiments of that kind typically run for long time frames. [14]

One possible solution to store the currently recorded values across multiple instruments/facilities at different research groups would be the project of building a XRR database. [15]

In addition to that, studies on noise application such as [1] showed, that it is a non-trivial problem to find a noise function that is capable of simulating real-world noise to an extent that the model does not overfit to the training data, but does remain flexible enough for prediction on an experimental data set.

A possible alternative solution would be to use experimental data for training. Furthermore, this can be elevated by a procedure called *data augmentation*, where one tries to alter the existing (experimental) data set to an extent, that training on this

data becomes possible. The application of uniform noise can be already seen as a first step leading in this direction.

In fact, there are already carried out approaches on this topic, where a prior Fourier analysis of the XRR data has been computed, although the approach did not improve the performance of the default MLP model. [18]

In a conclusion, it can be said that the currently driven research on ML based XRR data analysis will probably emerge as the leading fitting method in this field and further research on this topic should be done. The fact that ML based data analysis can be applied *in situ*, brings interesting possibilities like the feedback-based adaption of instruments and should be investigated further.

Appendix A

List of Acronyms

CNN Convolutional Neural Network. i, v, vii, 1, 2, 8, 9, 11–15, 17, 22, 23, 25–27, 31–35, 37, 38, 41

DL Deep Learning. 1, 2, 11, 15, 25

HPC High Performance Computation. 1, 2, 26, 30

ML Machine Learning. 1, 13, 19, 20, 42

MLP Multi-layer Perceptron. v, vii, 1, 2, 9, 11–13, 17, 21, 22, 32–35, 37, 38, 41, 42

MSE Mean Squared Error. 15, 29, 33, 34, 37, 38

NN Neural Network. 1, 2, 9, 15, 17, 23, 25

SLD Scattering Length Density. v, 5, 6, 37, 41

XRR X-Ray Reflectivity. i, v, 1–8, 11, 15, 17, 19–23, 25, 26, 28, 31–35, 37, 38, 41, 42

Appendix B

Bibliography

[1] Alessandro Greco, Vladimir Starostin, Evelyn Edel, Valentin Munteanu, Nadine Rußegger, Ingrid Dax, Chen Shen, Florian Bertram, Alexander Hinderhofer, Alexander Gerlach, and Frank Schreiber. Neural network analysis of neutron and x-ray reflectivity data: automated analysis using, experimental errors and feature engineering. *J Appl Crystallogr*, 55(2):362–369, 2022.

[2] Artur Glavic and Matts Björck. Genx 3: the latest generation of an established tool. *J Appl Crystallogr*, 55:1063–1071, 2022.

[3] Jens Als-Nielsen and Des McMorrow. *Elements of Modern X-ray Physics*. John Wiley & Sons, Ltd, Chichester, United Kingdom, 2011.

[4] V Holy, T Baumbach, and U Pietsch. *High-resolution x-ray scattering from thin films and multilayers*. Springer, Berlin, 1999.

[5] L. Koester, H. Rauch, and E. Seymann. Neutron scattering lengths: A survey of experimental data and methods. *Atomic Data and Nuclear Data Tables*, 49 (1):65–120, 1991.

[6] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer Science and Business Media, LLC., New York, USA, 2006.

[7] Chigozie Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall. Activation functions: Comparison of trends in practice and research for deep learning. *CoRR*, abs/1811.03378, 2018.

[8] Matthieu Deru and Alassane Ndiaye. *Deep Learning - mit Tensorflow, Keras und Tensorflow.js*. Rheinwerk Computing, Bonn, Germany, 2020.

[9] Tesla Inc. Autopilot full self-driving hardware (neighborhood short), 2017. URL `https://www.tesla.com/de_DE/autopilot`. [Online; accessed May

13, 2022].

[10] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, Cambridge, USA, 2016.

[11] Alex Shenfield and Martin Howarth. A novel deep learning model for the detection and identification of rolling element-bearing faults. *Sensors*, 20(18): 5112, 2020.

[12] Y-Lan Boureau, Jean Ponce, and Yann LeCun. A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, page 111–118, Madison, USA, 2010. Omnipress.

[13] L. G. Parratt. Surface studies of solids by total reflection of x-rays. *Phys. Rev.*, 95:359–369, 1954.

[14] Alessandro Greco. *Analysis of Specular Reflectivity of Thin Films Using Machine Learning*. PhD thesis, University of Tuebingen, 2022.

[15] Linus Pithan, Alessandro Greco, Alexander Hinderhofer, Alexander Gerlach, Stefan Kowarik, Nadine Rußegger, Ingrid Dax, and Frank Schreiber. Reflectometry curves (xrr and nr) and corresponding fits for machine learning, 2022. URL `https://zenodo.org/record/6497438`.

[16] Alessandro Greco, Vladimir Starostin, Christos Karapanagiotis, Alexander Hinderhofer, Alexander Gerlach, Linus Pithan, Sascha Liehr, Frank Schreiber, and Stefan Kowarik. Fast fitting of reflectivity data of growing thin films using neural networks. *J Appl Crystallogr*, 52(6):1342–1347, 2019.

[17] Alessandro Greco, Vladimir Starostin, Alexander Hinderhofer, Alexander Gerlach, Maximilian W A Skoda, Stefan Kowarik, and Frank Schreiber. Neural network analysis of neutron and x-ray reflectivity data: pathological cases, performance and perspectives. *Machine Learning: Science and Technology*, 2 (4):045003, 2021.

[18] Evelyn Edel. *Analysis of X-Ray Reflectivity Data using Machine Learning and Fourier Transforms (unpublished)*. Bachelor's thesis, University of Tuebingen, 2021.