TECHNISCHE UNIVERSITÄT WIEN
Vienna | Austria

DIPLOMARBEIT

Inverse Scattering in One-dimensional Random Media Using Deep Learning

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Technische Physik

eingereicht von

Lukas Michael Rachbauer
Matrikelnummer 01255141

ausgeführt am Institut für Theoretische Physik
der Fakultät für Physik der Technischen Universität Wien

Betreuung
Betreuer: Univ.Prof. Dipl.-Ing. Dr.techn. Stefan Rotter

Wien, 20.05.2019

(Unterschrift Verfasser) (Unterschrift Betreuer)

Contents

Abstract

The inverse scattering problem is in general ill-posed and highly nonlinear. The aim of this thesis is to develop a fast algorithm that provides solutions to such inverse scattering problems in compactly supported one-dimensional random media. A promising candidate for this nonlinear task is Deep Learning, which showed great success in the recent past. The methodology of this approach is to train an Artificial Neural Network for a stochastic class of samples on numerically generated data. Inverse scattering is then performed by means of a simple forward-pass through the Artificial Neural Network.

It is shown that in cases where the inverse scattering problem has a unique solution and where the scattering is not too strong, an Artificial Neural Network is able to solve the inverse scattering problem more efficiently than preexisting methods.

1 Introduction

The forward scattering problem (FSP) consists in calculating the scattering behaviour of a system based on the structure of the system. The inverse scattering problem (ISP) involves the opposite task, i.e., to find the structure of a system based on the scattering data as encapsulated, e.g., in the so-called scattering matrix. There are many applications for the ISP like in medical imaging [1], non-destructive material testing [2], seismology [3,4] or nuclear physics [5], to name a few. However, conventional techniques for inverse scattering, like the filtered backprojection – an inversion of the Radon transform appearing in tomography [6] – do not exploit the effect of multiple scattering. But these multiply scattered waves also contain very valuable information about the scattering system.

In this thesis we propose an inverse scattering technique, which does not per se exclude the information from such multiple scattering events, using methods from Deep Learning (DL). DL is well suited for this task for a number of reasons: DL methods are able to learn nonlinear mappings from a sufficiently large set of training samples. Since the FSP is easy and fast to solve (for one-dimensional systems, which we are going to consider), we are able to produce in a short amount of time a considerable amount of training data, from which the DL algorithm can learn. A closely related advantage of DL is that the main computational work is done before the actual application of the method. This implies that DL algorithms are able to perform their tasks substantially faster than other methods. Also, the DL model can be "prepared" based on numerically generated data, and later be applied to real-world measurements. Recent advances confirm the capabilities of DL methods in inverse scattering scenarios [7–15].

We concentrate on one-dimensional Hermitian quantum systems. In this case the (analytical) theory of the ISP has already been established in the 1950s and 1960s [16–18]. However, the central integral equation in this theory can only be solved in special cases [16, 19]. Furthermore, an iterative procedure based on this integral equation [20] does not always yield satisfactory results. As we will demonstrate explicitly in this thesis, the proposed DL method is able to overcome those problems. The one-dimensional case we study is not only an interesting fundamental problem on its own, but also serves as a precursor for more relevant applications in more than one dimension.

This thesis is structured as follows: In Sec. 2 we give a brief overview over the one-dimensional FSP. In Sec. 3 we introduce the ISP, the central integral equation of the

ISP and an iterative way to solve it numerically. We demonstrate that this iterative approximation does not always converge. After giving a basic introduction to DL in Sec. 4, we treat the ISP using methods from DL in Sec. 5.
In the Appendix one can find the specifications of the used machine (Sec. A), how the random potentials are generated (Sec. B) and supplements regarding forward scattering (Sec. C) as well as inverse scattering (Sec. D).

2 One-dimensional forward scattering

Before dealing with the inverse scattering problem, we provide here a short overview over the forward scattering problem and its solution for one-dimensional Hermitian quantum systems. The scattering properties of such a system are fully captured in a quantity called the scattering matrix, which is introduced and discussed in the following subsection.

2.1 Scattering matrix

We consider a nonrelativistic quantum particle of mass $m > 0$ in a one-dimensional potential $V : \mathbb{R} \to \mathbb{R}$ with the support $\operatorname{supp}(V) = \overline{\{x \in \mathbb{R} : V(x) \neq 0\}} \subseteq [-a, a]$ for some appropriate $a > 0$. In the asymptotic regions $x < -a$ and $x > a$, where the potential vanishes, the momentum of the particle $k > 0$ fully determines its energy

$$E(k) = \frac{\hbar^2 k^2}{2m} > 0. \tag{1}$$

The wavefunction $\psi(k; x)$ of the quantum particle satisfies the stationary Schrödinger equation[1]

$$\frac{\partial^2 \psi}{\partial x^2}(k; x) = \frac{2m}{\hbar^2} V(x) \psi(k; x) - k^2 \psi(k; x). \tag{2}$$

The asymptotic behaviour of the wavefunction is given by

$$\begin{aligned} \forall x < -a : \psi(k; x) &= \psi_{\text{in}}^{\rightarrow} \mathrm{e}^{\mathrm{i}kx} + \psi_{\text{out}}^{\leftarrow} \mathrm{e}^{-\mathrm{i}kx}, \\ \forall x > a : \psi(k; x) &= \psi_{\text{out}}^{\rightarrow} \mathrm{e}^{\mathrm{i}kx} + \psi_{\text{in}}^{\leftarrow} \mathrm{e}^{-\mathrm{i}kx}, \end{aligned} \tag{3}$$

i.e. the wave can be decomposed into two counter-propagating plane waves in each of the two asymptotic regions. This is schematically shown in Fig. 1. The mapping from the incoming waves to the outgoing waves is linear and depends only on the momentum k, as specified by the scattering matrix[1]

$$S(k) = \begin{pmatrix} r(k) & t'(k) \\ t(k) & r'(k) \end{pmatrix}, \tag{4}$$

[1]Derivatives are always written as $\frac{d}{dx}$ or $\frac{\partial}{\partial x}$. A prime $'$ never denotes a derivative, but always belongs to the name of a function.

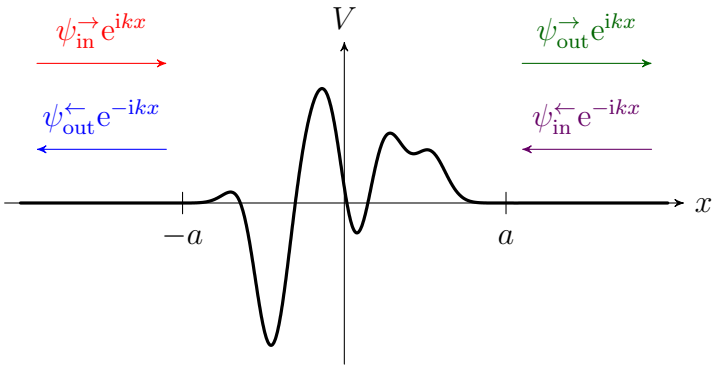

Figure 1: Setup of a one-dimensional scattering event. The potential V has a support inside of the interval $[-a, a]$. Outside this interval the wavefunction can be decomposed into inward propagating plane waves ($\psi_{\text{in}}^{\rightarrow} e^{ikx}$ and $\psi_{\text{in}}^{\leftarrow} e^{-ikx}$) and outward propagating plane waves ($\psi_{\text{out}}^{\leftarrow} e^{-ikx}$ and $\psi_{\text{out}}^{\rightarrow} e^{ikx}$).

which relates the incoming wave amplitudes to the outgoing ones:

$$\begin{pmatrix} \psi_{\text{out}}^{\leftarrow} \\ \psi_{\text{out}}^{\rightarrow} \end{pmatrix} =: S(k) \begin{pmatrix} \psi_{\text{in}}^{\rightarrow} \\ \psi_{\text{in}}^{\leftarrow} \end{pmatrix}. \tag{5}$$

The entries of the scattering matrix can be interpreted as follows:

$r(k)$ ... reflection amplitude w.r.t. incidence from the left,

$t(k)$ ... transmission amplitude w.r.t. incidence from the left,

$r'(k)$ ... reflection amplitude w.r.t. incidence from the right,

$t'(k)$ ... transmission amplitude w.r.t. incidence from the right.

Due to conservation of the probability current the scattering matrix has to be unitary, i.e.

$$S^{\dagger}(k) S(k) = \mathbf{1}. \tag{6}$$

This equation is equivalent to the following set of equations:

$$\begin{pmatrix} |r(k)|^2 + |t(k)|^2 & r^*(k) t'(k) + t^*(k) r'(k) \\ r'^*(k) t(k) + t'^*(k) r(k) & |r'(k)|^2 + |t'(k)|^2 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \tag{7}$$

where the asterisk denotes complex conjugation. The unitarity condition in Eq. (6) can

be reformulated to

$$S(k)\,S^{\dagger}(k)=\begin{pmatrix}|r(k)|^2+|t'(k)|^2 & r'^{*}(k)\,t'(k)+t^{*}(k)\,r(k)\\ r^{*}(k)\,t(k)+t'^{*}(k)\,r'(k) & |r'(k)|^2+|t(k)|^2\end{pmatrix}=\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}=\mathbf{1}. \tag{8}$$

Therefore it always holds that

$$\begin{aligned}|r'(k)|^2=|r(k)|^2=:R(k),\\ |t'(k)|^2=|t(k)|^2=:T(k).\end{aligned} \tag{9}$$

The functions $R(k)$ and $T(k)$ are called the reflection coefficient and the transmission coefficient, respectively.
A consequence of the unitarity of $S(k)$ is

$$|\det(S(k))|=|r'(k)\,r(k)-t'(k)\,t(k)|=1. \tag{10}$$

Due to the time-reversal symmetry it holds that

$$S^{\top}(k)=S(k)\iff t'(k)=t(k). \tag{11}$$

If the potential is symmetric under a parity transformation, i.e. $V(-x)=V(x)$, then additionally $r'(k)=r(k)$.

In the following we distinguish between two scenarios: incidence from the left (i.f.t.l.) or incidence from the right (i.f.t.r.).
In the first case (i.f.t.l.) the wavefunction has the asymptotic form

$$\begin{aligned}\forall x<-a:\psi(k;x)&=\frac{1}{t(k)}\mathrm{e}^{\mathrm{i}kx}+\frac{r(k)}{t(k)}\mathrm{e}^{-\mathrm{i}kx},\\ \forall x>a:\psi(k;x)&=\mathrm{e}^{\mathrm{i}kx}.\end{aligned} \tag{12}$$

This normalization allows for unique starting values in numerical calculations, where one typically propagates over the entire potential (i.e. from a to $-a$). In the second case (i.f.t.r.) it holds that

$$\begin{aligned}\forall x<-a:\psi(k;x)&=\mathrm{e}^{-\mathrm{i}kx},\\ \forall x>a:\psi(k;x)&=\frac{r'(k)}{t'(k)}\mathrm{e}^{\mathrm{i}kx}+\frac{1}{t'(k)}\mathrm{e}^{-\mathrm{i}kx}.\end{aligned} \tag{13}$$

2.2 Numerical forward scattering: The Numerov algorithm

The forward scattering problem (FSP) can be solved analytically only in very few special cases. One such case are potentials with a finite number of delta-peaks (cf. Subsec. C.1). We aim to apply our Deep Learning (DL) inverse scattering method not only to such potentials, but also more generally to smooth potentials with a compact support, for which the FSP can not be analytically solved. There are many numerical methods available, out of which we choose the Numerov algorithm, introduced in this subsection. In Subsec. C.2 we present two alternative algorithms, compare them against the Numerov algorithm and show that the latter surpasses the other ones regarding computation time as well as accuracy.

The Numerov algorithm is designed to solve ordinary differential equations of second order with no first order derivatives. Take as an example the one-dimensional stationary Schrödinger equation

$$\frac{\partial^2 \psi}{\partial x^2}(k;x) = \left(\frac{2m}{\hbar^2} V(x) - k^2\right) \psi(k;x) =: v(k;x)\,\psi(k;x), \tag{14}$$

which is precisely such a differential equation (for each k independently). In its most basic form the Numerov algorithm is a finite difference method and requires an equidistant grid in position space. Let the grid spacing $\Delta x > 0$ be sufficiently small, i.e.

$$k\Delta x = \frac{\sqrt{2mE}}{\hbar}\Delta x \ll 1. \tag{15}$$

We define the spatial discretization

$$\forall i \in \mathbb{Z} : x_i := i\Delta x, f_i := f(x_i) \tag{16}$$

for any function $f(x)$. The potential V has a support with $\mathrm{supp}(V) \subseteq [-a, a]$. The quantity corresponding to the boundary a in the discretized space is

$$I := \left\lceil \frac{a}{\Delta x} \right\rceil \implies a = (I - \varepsilon)\Delta x \implies (I-1)\Delta x < a \leq I\Delta x \tag{17}$$

with $\varepsilon \in [0, 1)$. This means that $V_i = 0$ for $i < -I$ and for $i > I$.

For convenience we define an auxiliary quantity $Q_i(k)$, which has no special physical

meaning, but abbreviates the equation describing the Numerov algorithm.

$$\forall i \in \mathbb{Z} : Q_i(k) := \left(1 - \frac{(\Delta x)^2}{12} v_i(k)\right) \psi_i(k). \tag{18}$$

$v_i(k) \equiv v(k; x_i)$ is defined in Eq. (14). Since the potential V_i and thus $v_i(x)$ is known, one can always switch between $Q_i(k)$ and $\psi_i(k)$. The Numerov algorithm reads

$$Q_{i+1}(k) + Q_{i-1}(k) = 12\psi_i(k) - 10Q_i(k) + \mathcal{O}\left((\Delta x)^6\right). \tag{19}$$

The global error is of the order $\mathcal{O}\left((\Delta x)^4\right)$, the same as with the classical Runge-Kutta method.

In the case "incidence from the left" we have the asymptotic wavefunction $\psi(k; x > a) = \mathrm{e}^{\mathrm{i}kx}$, i.e. $\psi_i(k) = \mathrm{e}^{\mathrm{i}kx_i}$ for $i > I$. Using the Numerov algorithm we can propagate through the potential from $i = I+2$ and $i = I+1$ (two initial values are needed) down to $i = -I-1$ and $i = -I-2$. From the asymptotic form (cf. Eq. (12)) we know that

$$\begin{aligned}
\psi_{-I-1}(k) &= \frac{1}{t(k)} \mathrm{e}^{\mathrm{i}k(-I-1)\Delta x} + \frac{r(k)}{t(k)} \mathrm{e}^{-\mathrm{i}k(-I-1)\Delta x}, \\
\psi_{-I-2}(k) &= \frac{1}{t(k)} \mathrm{e}^{\mathrm{i}k(-I-2)\Delta x} + \frac{r(k)}{t(k)} \mathrm{e}^{-\mathrm{i}k(-I-2)\Delta x}.
\end{aligned} \tag{20}$$

These equations can be solved for the reflection and the transmission amplitude:

$$\begin{aligned}
r(k) &= -\mathrm{e}^{-\mathrm{i}k(2I+3)\Delta x} \frac{\psi_{-I-1}(k) - \mathrm{e}^{\mathrm{i}k\Delta x}\psi_{-I-2}(k)}{\mathrm{e}^{\mathrm{i}k\Delta x}\psi_{-I-1}(k) - \psi_{-I-2}(k)}, \\
t(k) &= \mathrm{e}^{-\mathrm{i}k(I+2)\Delta x} \frac{\mathrm{e}^{2\mathrm{i}k\Delta x} - 1}{\mathrm{e}^{\mathrm{i}k\Delta x}\psi_{-I-1}(k) - \psi_{-I-2}(k)}.
\end{aligned} \tag{21}$$

The case "incidence from the right" is treated analogously: One starts with $\psi_i(k) = \mathrm{e}^{-\mathrm{i}kx_i}$ for $i < -I$, propagates through the potential from left to right, calculates $\psi_{I+1}(k)$ as well as $\psi_{I+2}(k)$ and from these:

$$\begin{aligned}
r'(k) &= -\mathrm{e}^{-\mathrm{i}k(2I+3)\Delta x} \frac{\psi_{I+1}(k) - \mathrm{e}^{\mathrm{i}k\Delta x}\psi_{I+2}(k)}{\mathrm{e}^{\mathrm{i}k\Delta x}\psi_{I+1}(k) - \psi_{I+2}(k)}, \\
t'(k) &= \mathrm{e}^{-\mathrm{i}k(I+2)\Delta x} \frac{\mathrm{e}^{2\mathrm{i}k\Delta x} - 1}{\mathrm{e}^{\mathrm{i}k\Delta x}\psi_{I+1}(k) - \psi_{I+2}(k)}.
\end{aligned} \tag{22}$$

2.3 Localization length

In one-dimensional disordered potentials that do not have any symmetric or periodic shape, waves typically localize. This means that the transmission across such a potential is exponentially decreasing for increasing potential length. In order to quantify the scattering strength of such a disordered (random) scattering potential V, one introduces the localization length ξ. Strictly speaking, one does not define a localization length for a specific potential, but rather for a whole stochastic class $\mathcal{C}$ of random potentials. Each such class can be characterized by statistical quantities, like the mean squared value of the potentials per length or the autocorrelation length.

Let L denote the length of a random potential $V \in \mathcal{C}$, i.e. the size of its support. We denote the transmission coefficient of V for a wave with momentum k by $T(k, L)$. Then the localization length $\xi_{\mathcal{C}}(k)$ is defined as

$$\frac{1}{\xi_{\mathcal{C}}(k)} := -\lim_{L\to\infty} \frac{1}{L} \langle \ln(T(k, L)) \rangle , \tag{23}$$

where the angle brackets denote an ensemble average over the appropriate subset $\{V \in \mathcal{C} : |\mathrm{supp}(V)| = L\}$.

Waves with a larger incident momentum k penetrate deeper into the potential, therefore the localization length $\xi_{\mathcal{C}}$ increases with k (cf. Fig. 5).

A good overview over the theory of localization is given in Ref. [21]. Localization originates from multiple interference of partially reflected waves. Bear in mind that the real-valued potential V does not cause any loss or absorption. In one dimension the localization length ξ is twice the mean free path ℓ of the quantum particle.

Let L be the length of a specific random potential $V \in \mathcal{C}$. Depending on the relation of L to the mean free path ℓ and to the localization length ξ, one distinguishes three scattering regimes:

- $L < \ell$: ballistic regime,
- $\ell < L < \xi$: diffusive transport,
- $L > \xi$: strong localization.

In one-dimensional systems diffusive transport practically does not occur since $\xi = 2\ell$ and the transition between the regimes is not sharp.

In the following we set $\hbar = m = 1$. We want to discuss the impact of different parameters on the localization length ξ. We found that the class $\mathcal{C} = \mathcal{S}^{0.01,20,0.01}_{-10,10,10/512}(\sigma_{\mathrm{pot}}, \sigma_{\mathrm{ker}}, 0)$

(cf. Sec. B.2) works well for demonstrational purposes. The numbers have the following meaning: $k_{\min} = 0.01$, $k_{\max} = 20$ and $\Delta k = 0.01$ determine the discretization of momentum space; $x_{\min} = -10$, $x_{\max} = 10$ and $\Delta x = \frac{10}{512} \approx 0.02$ determine the discretization of real space; σ_{pot} describes the amplitude of the potentials, σ_{ker} the autocorrelation length of the potentials and 0 indicates that the potentials have zero mean, i.e. $\int_{-\infty}^{\infty} V(x)\,dx = 0$.

At first we keep $\sigma_{\text{pot}} = 100$ and $\sigma_{\text{ker}} = 0.05$ fixed. An example is shown in Fig. 2. The reflection and transmission amplitudes as well as the transmission coefficient are calculated with the Numerov algorithm and shown in Fig. 3.

For $k \gtrsim 12.5$ the free-space energy E exceeds the maximum potential value $V_{\max} \approx 78$. This means that for $k \gtrsim 12.5$ the wavefunction is curved towards the x-axis everywhere (meaning oscillatory and no exponential behaviour). Still, the transmission is mostly close to zero up to $k \lesssim 15$ (cf. Fig. 3). This is an indication for localization by interference.

By averaging over 10^4 such random potentials, we can approximate the ensemble average $\langle \ln(T(k,L)) \rangle$. For each potential $V(x)$, the transmission coefficient is calculated for $V(x)\,\Theta(L-a-x)$ having support of length L, thus yielding $T(\cdot, L)$. Fig. 4 shows $\langle \ln(T(k=10,L)) \rangle$ as a function of L. Linear regression gives the localization length $\xi(k=10) \approx 1.02$ as the negative inverse slope of the curve plotted in Fig. 4. (Only the region $L \in [4\sigma_{\text{ker}}, 2a - 4\sigma_{\text{ker}}]$ is used for linear regression.) This linear behaviour holds for the entire inspected k-range, i.e. from $k = 0$ up to $k = 20$.

The localization length ξ as a function of momentum k is shown in Fig. 5. One can clearly see that there is strong localization for $k \lesssim 15$, since in this case the localization length ξ is much smaller than the length of the potential $2a$.

Other reliable indicators of localization are the relatively large fluctuations in the transmission coefficient $T(k)$, as can be seen in Fig. 3 for $k \in [10, 15]$. These fluctuations do not appear (as opposed to the exponential decay of $\mathrm{e}^{\langle \ln(T) \rangle}$) when the potential is absorbing and not random.

We now keep $k = 10$ fixed and investigate how the localization length ξ changes with σ_{pot} and σ_{ker}. Increasing σ_{pot} means that the amplitude of the potential is larger, leading to a shorter penetration depth, hence ξ decreases (Fig. 6). With increasing σ_{ker}, on the other hand, the potentials get more autocorrelated, i.e. less random, which reduces the scattering of the wave and thus leads to larger ξ (Fig. 7).

Now that we have investigated the forward scattering problem and related topics, we turn our focus to the inverse scattering problem in the subsequent section.

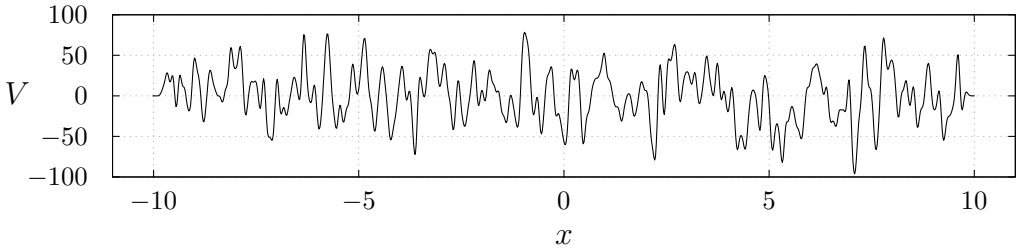

Figure 2: Instance of a random potential with $\sigma_{\text{pot}} = 100$ (determining the amplitude of the potential) and $\sigma_{\text{ker}} = 0.05$ (determining the autocorrelation length of the potential).

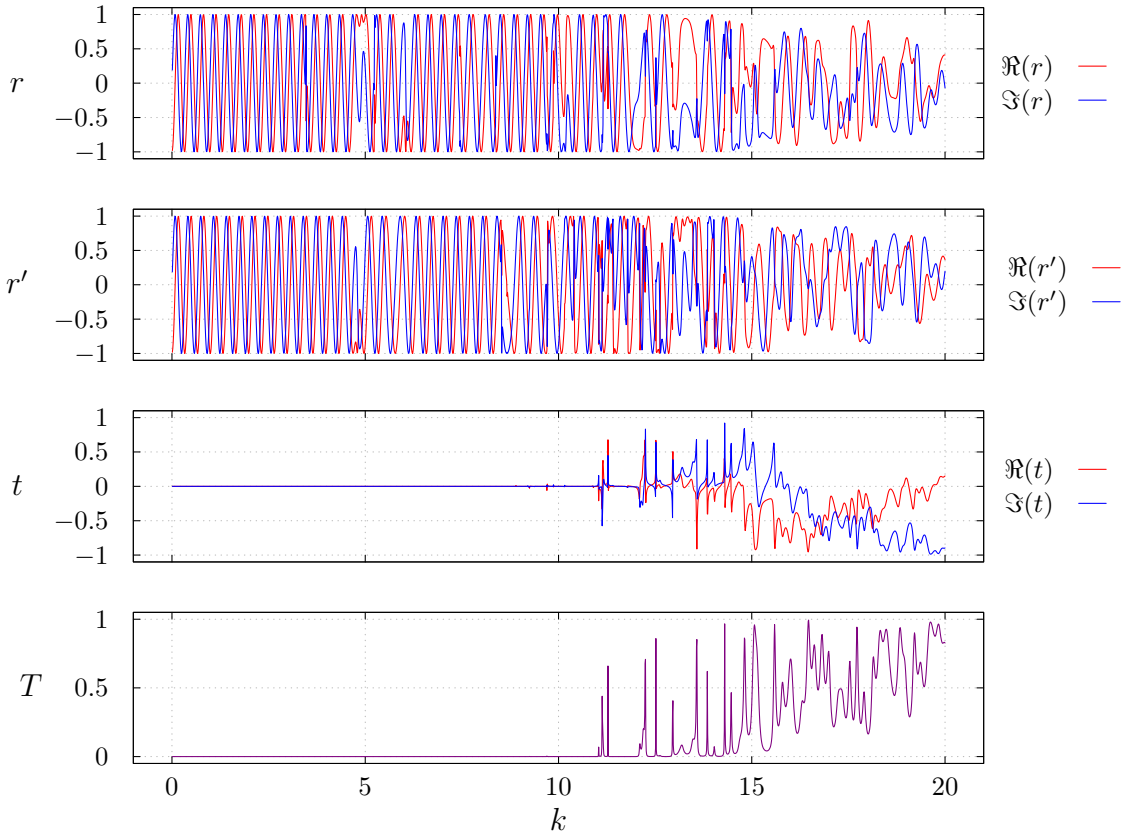

Figure 3: Left reflection amplitude $r(k)$, right reflection amplitude $r'(k)$, transmission amplitude $t(k)$ and transmission coefficient $T(k) = |t(k)|^2$ of the potential in Fig. 2.

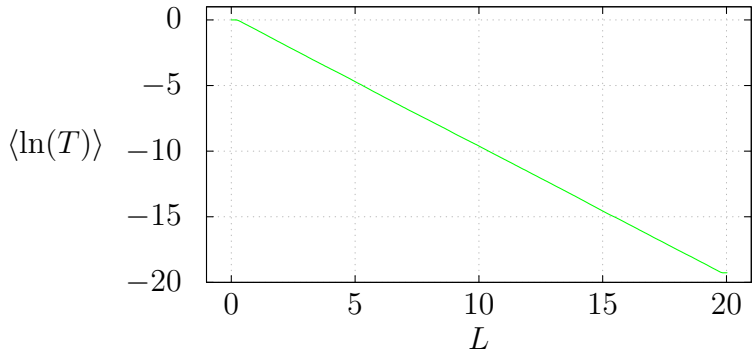

Figure 4: Ensemble average of the logarithm of the transmission coefficient for $k = 10$. From this linear relation the localization length can be read off as the negative inverse slope.

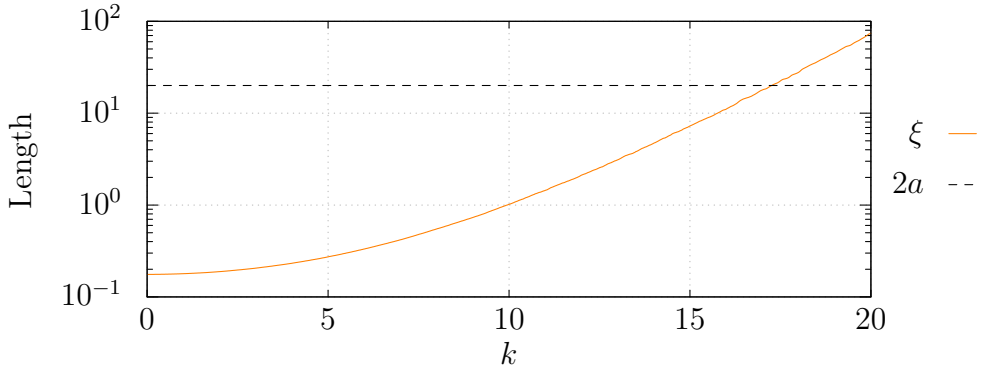

Figure 5: Localization length ξ as a function of momentum k for fixed $\sigma_{\text{pot}} = 100$ and $\sigma_{\text{ker}} = 0.05$, compared to the length of the potential, $2a$.

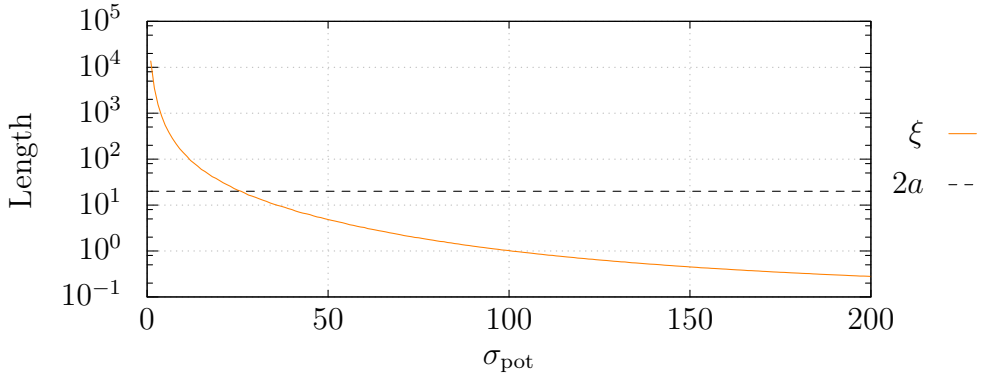

Figure 6: Localization length ξ as a function of the standard deviation σ_{pot} (determining the amplitude of the potential) for fixed $k = 10$ and $\sigma_{\text{ker}} = 0.05$, compared to the length of the potential, $2a$.

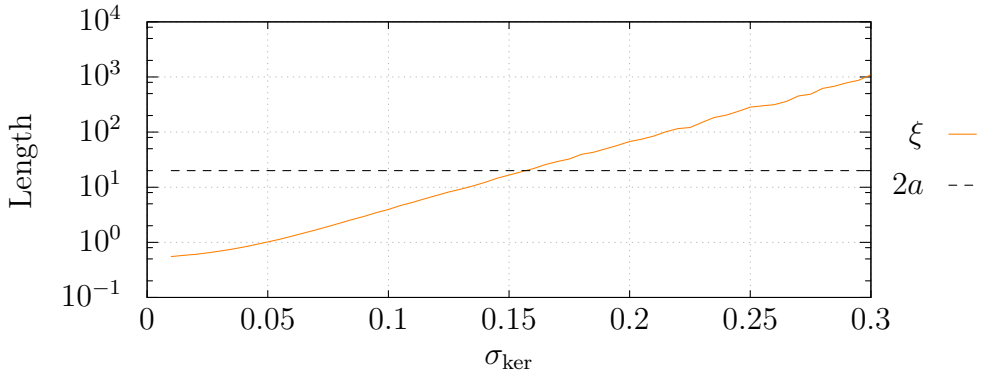

Figure 7: Localization length ξ as a function of the standard deviation σ_{ker} (determining the autocorrelation length, i.e. the "smoothness" of the potential – smaller values of σ_{ker} mean more strongly fluctuating potentials) for fixed $k = 10$ and $\sigma_{\mathrm{pot}} = 100$, compared to the length of the potential, $2a$.

3 One-dimensional inverse scattering

3.1 The inverse scattering problem[2]

While in the forward scattering problem one wishes to calculate the scattering matrix $S(k)$ corresponding to a certain potential $V(x)$, the aim of the inverse scattering problem (ISP) is to find a potential $V(x)$ which reproduces a given (physically meaningful) scattering matrix $S(k)$. The ISP formulated in this way is ill-defined, i.e. the mapping $V(x) \mapsto S(k)$ is not injective.
From now on we only consider potentials $V : \mathbb{R} \to \mathbb{R}$ belonging to the set

$$L_2^1 := \left\{ V : \mathbb{R} \to \mathbb{R} \,\middle|\, \int_{-\infty}^{\infty} |V(x)| \left(1 + x^2\right) dx < \infty \right\}, \tag{24}$$

since for such potentials a solution to the ISP is guaranteed to exist, if the potential has no bound states or if one provides additional data about the bound states (see below). In Subsec. D.1 we show that all potentials considered in this thesis lie in L_2^1.
Let $S(k)$ be a scattering matrix and $N \in \mathbb{N}_0$ a natural number. Then there exists a $2N$-parameter family of different potentials, all giving rise to the given $S(k)$ and possessing N bound states. This implies that the solution of the ISP is unique if the underlying potential does not support any bound states ($N = 0$).
Providing the energies of the N bound states, i.e. the discrete part of the spectrum of the Hamiltonian, if there is any, does not lift the degeneracy, meaning that there is still an N-parameter family of potentials, all having the same spectrum and the same scattering matrix $S(k)$. This remaining degeneracy can be lifted by some knowledge about the bound-state wavefunctions. For this purpose one defines two wavefunctions for each bound state, ψ_l and ψ_r, differing only by a factor, i.e. they are normalized differently. The corresponding normalization constants c_n and c_n' are precisely the additional information about the bound states we need to know in order to render the ISP uniquely solvable. This is established below in more detail.

The scattering matrix $S(k)$ of a given potential $V \in L_2^1$ can be analytically continued from $k \in \mathbb{R}^+$ to $k \in \mathbb{C}$. From Eq. (3) one can read off that

$$S(k < 0) = S^*(-k). \tag{25}$$

[2]The following discussion of the theory of the inverse scattering problem is largely based on Refs. [16, 17].

The scattering matrix $S(k)$ is unitary and continuous for $k \in \mathbb{R}$. The reflection amplitude $r(k)$ is meromorphic (i.e. holomorphic up to isolated poles) in the upper half of the complex plane $\mathbb{C}^+$, possessing only a finite number of simple poles on the positive imaginary axis. We denote the locations of these poles by $\mathrm{i}\kappa_1, \ldots, \mathrm{i}\kappa_N$. The number of poles N is exactly the number of bound states of $V(x)$ and the eigenenergies are given by

$$E_n = -\frac{\hbar^2 \kappa_n^2}{2m}. \tag{26}$$

The corresponding bound-state wavefunctions $\psi(\mathrm{i}\kappa_n; x)$ are real-valued and square integrable in x. Instead of normalizing them to unity we require that

$$\begin{aligned} \lim_{x \to -\infty} \psi_\mathrm{l}(\mathrm{i}\kappa_n; x)\, \mathrm{e}^{-\kappa_n x} &= 1, \\ \lim_{x \to \infty} \psi_\mathrm{r}(\mathrm{i}\kappa_n; x)\, \mathrm{e}^{\kappa_n x} &= 1. \end{aligned} \tag{27}$$

These asymptotic behaviours define two wavefunctions ψ_l and ψ_r for each bound state, differing only by a factor. The left / right normalization constants of the bound states are defined as

$$\begin{aligned} c_n &:= \left(\int_{-\infty}^{\infty} \psi_\mathrm{l}^2(\mathrm{i}\kappa_n; x)\, dx \right)^{-1}, \\ c_n' &:= \left(\int_{-\infty}^{\infty} \psi_\mathrm{r}^2(\mathrm{i}\kappa_n; x)\, dx \right)^{-1}, \end{aligned} \tag{28}$$

respectively. We highlight some general properties of the normalization constants in Subsec. D.2. It is these normalization constants (either all c_n or all c_n') one further has to provide (additionally to $S(k)|_{k \in \mathbb{R}}$ and the bound state energies E_n) in order to get a unique solution for the ISP. In fact, one does not need to know the entire scattering matrix $S(k)|_{k \in \mathbb{R}}$ but only the reflection amplitude from one side, i.e. $r(k)$ or $r'(k)$. In the following, we select the reflection amplitude from the left $r(k)$.

3.2 Marčenko integral equation

As we have seen in the previous subsection, we need the following data in order to obtain a unique solution of the inverse scattering problem (ISP): the reflection amplitude from one side (we choose here w.l.o.g. the left side) $r(k)$, the eigenenergies of all bound states $E_n = -\frac{\hbar^2}{2m}\kappa_n^2$ and the left normalization constants of all bound states c_n, defined in Eq. (28). The first central quantity in the theory of inverse scattering contains all

this information, is denoted by $\mathcal{R}$ and defined by

$$\mathcal{R}(x) := \frac{1}{2\pi}\int_{-\infty}^{\infty} r(k)\,\mathrm{e}^{-\mathrm{i}kx}\,dk + \sum_{n=1}^{N} c_n \mathrm{e}^{\kappa_n x}. \tag{29}$$

Since $r(k<0) = r^*(-k)$ (cf. Eq. (25)), we can rewrite

$$\mathcal{R}(x) = \frac{1}{\pi}\int_{0}^{\infty} \Re\left(r(k)\,\mathrm{e}^{-\mathrm{i}kx}\right) dk + \sum_{n=1}^{N} c_n \mathrm{e}^{\kappa_n x}. \tag{30}$$

From this we can see that $\mathcal{R}$ is a real-valued function. With this $\mathcal{R}$ one can solve the ISP by solving an integral equation, called the Marčenko integral equation. From the solution of this integral equation – the second central quantity in the theory of inverse scattering – denoted by $B(y,x)$, one can retrieve the potential $V(x)$.
The Marčenko integral equation has a simple form if the potential $V(x)$ vanishes for $x<0$, which we want to assume in the following. In Subsec. D.3 we show how to treat potentials, which vanish on different half-lines. The Marčenko integral equation reads

$$\mathcal{R}(2x) + B(y,x) + 2\int_{0}^{x} B(y, y-(x-x'))\,\mathcal{R}(2x')\,dx' = 0, \tag{31}$$

where $x \geq 0$, $y \geq 0$ and $x \leq y$. From the solution $B(y,x)$ one can calculate the potential according to

$$V(x) = \frac{\hbar^2}{m}\frac{dB(x,x)}{dx}. \tag{32}$$

Suppose that the potential V has a finite support $\mathrm{supp}(V) \subseteq [0,a]$, then one has to calculate $\mathcal{R}$ on $[0,2a]$ and B on $\left\{(y,x) \in [0,a]^2 : x \leq y\right\}$ only. This is a great advantage of the Marčenko integral equation formulated in Eq. (31), as opposed to other formulations, which are found more often in the literature. Two such alternative common formulations and the transformations between them are discussed in Subsec. D.4.
The stability of this inversion procedure (by solving the Marčenko integral equation) is discussed in Ref. [22]. In this paper it is shown that the instabilities in the ISP come from small errors in the poles of $r(k)$ for $k \in \mathbb{C}$ close to $k=0$. Furthermore the sensitivity to these errors grows with increasing potential values $V(x)$. The physical explanation behind this analysis is that the energy E of the wave is too small for it to penetrate deep enough into the potential V.

3.3 Iterative Marčenko approximation

The Marčenko integral equation (cf. Eq. (31)) can be analytically solved only for special cases [16, 19]. Ref. [20] proposes an iterative scheme to approximate the solution. The initial step of this iteration is obtained by neglecting the whole integral in Eq. (31):

$$B^{(0)}(y,x) = -\mathcal{R}(2x). \tag{33}$$

The successive steps are obtained by replacing B in the integrand of Eq. (31) by the one found in the previous step:

$$\forall \nu \geq 1 : B^{(\nu)}(y,x) = -\mathcal{R}(2x) - 2\int_0^x B^{(\nu-1)}(y, y-(x-x'))\,\mathcal{R}(2x')\,dx'. \tag{34}$$

The approximations of the potential are defined by

$$\forall \nu \geq 0 : V_{\text{Marč}}^{(\nu)}(x) := \frac{\hbar^2}{m}\frac{dB^{(\nu)}(x,x)}{dx}. \tag{35}$$

We now show that this iteration does not converge if there are bound states present. To this end we calculate the first two orders $V_{\text{Marč}}^{(0)}$ and $V_{\text{Marč}}^{(1)}$ explicitly:

$$\begin{aligned} V_{\text{Marč}}^{(0)}(x) &= \frac{\hbar^2}{m}\frac{dB^{(0)}(x,x)}{dx} = -\frac{\hbar^2}{m}\frac{d}{dx}\left(\mathcal{R}(2x)\right) = -\frac{2\hbar^2}{m}\frac{d\mathcal{R}}{dx}(2x) \\ &= \frac{\hbar^2}{m}\left(\frac{\mathrm{i}}{\pi}\int_{-\infty}^{\infty} r(k)\,k\mathrm{e}^{-2\mathrm{i}kx}\,dk - 2\sum_{n=1}^{N} c_n\kappa_n \mathrm{e}^{2\kappa_n x}\right). \end{aligned} \tag{36}$$

For $V_{\text{Marč}}^{(1)}$ we calculate

$$\begin{aligned} B^{(1)}(y,x) &= -\mathcal{R}(2x) + 2\int_0^x \mathcal{R}(2y-2x+2x')\,\mathcal{R}(2x')\,dx' \\ B^{(1)}(x,x) &= -\mathcal{R}(2x) + 2\int_0^x \mathcal{R}^2(2x')\,dx' \end{aligned}$$

$$V_{\text{Marč}}^{(1)}(x) = \frac{\hbar^2}{m}\frac{dB^{(1)}(x,x)}{dx} = \frac{2\hbar^2}{m}\left(-\frac{d\mathcal{R}}{dx}(2x) + \mathcal{R}^2(2x)\right). \tag{37}$$

The exponential terms $\propto \mathrm{e}^{2\kappa_n x}$ from the bound states do not drop out in the iteration procedure. Hence we can conclude that in the presence of bound states:

$$\forall \nu \in \mathbb{N}_0 : V_{\text{Marč}}^{(\nu)} \notin L_2^1. \tag{38}$$

Thus the sequence $\left(V_{\text{Marč}}^{(\nu)}\right)_{\nu\in\mathbb{N}_0}$ cannot converge to $V \in L_2^1$. This means that the absence of bound states is necessary for the validity (i.e. the convergence) of the iteration procedure. In Subsec. 3.4 we further investigate the convergence properties of the Marčenko iteration.

If we neglect the contributions from the bound states, then the first two orders $V_{\text{Marč}}^{(0)}$ and $V_{\text{Marč}}^{(1)}$ have the following property:

$$\int_{-\infty}^{\infty} V_{\text{Marč}}^{(0)}(x)\,dx \propto \int_{-\infty}^{\infty} r(k)\,k\delta(k)\,dk = 0, \tag{39}$$

$$\int_{-\infty}^{\infty} V_{\text{Marč}}^{(1)}(x)\,dx = \frac{\hbar^2}{\pi m}\int_{0}^{\infty} |r(k)|^2\,dk \geq 0. \tag{40}$$

Eqs. (36) and (37) are easily translated into terms of the reflection from the right $r'(k)$ (cf. Subsubsec. D.3):

$$\mathcal{R}'(x) := \frac{1}{2\pi}\int_{-\infty}^{\infty} r'(k)\,\mathrm{e}^{\mathrm{i}kx}\,dk\left\{+\sum_{n=1}^{N} c'_n \mathrm{e}^{-\kappa_n x}\right\} \tag{41}$$

$$\begin{aligned} \Longrightarrow\ V'^{(0)}_{\text{Marč}}(x) &= \frac{2\hbar^2}{m}\frac{d\mathcal{R}'}{dx}(2x)\,, \\ V'^{(1)}_{\text{Marč}}(x) &= \frac{2\hbar^2}{m}\left(\frac{d\mathcal{R}'}{dx}(2x) + \mathcal{R}'^2(2x)\right). \end{aligned} \tag{42}$$

We call $V_{\text{Marč}}^{(\nu)}(x)$ the left ν^{th}-order and $V'^{(\nu)}_{\text{Marč}}(x)$ the right ν^{th}-order Marčenko approximation. In Subsec. D.6 we show that the left and right Marčenko approximations reconstruct the left and right part of the potential best, respectively. We can combine the left and right Marčenko approximations into a weighted mean, defined by

$$\forall\nu\in\mathbb{N}_0 : W_{\text{Marč}}^{(\nu)}(x) := \frac{1}{2}\left(1-\frac{x}{a}\right)V_{\text{Marč}}^{(\nu)}(x) + \frac{1}{2}\left(1+\frac{x}{a}\right)V'^{(\nu)}_{\text{Marč}}(x)\,. \tag{43}$$

The weights for the left and right Marčenko approximations linearly shrink from 1 to 0 and grow from 0 to 1, respectively, when going from $x = -a$ to $x = a$.

In Subsec. D.5 we introduce another approximate solution of the inverse scattering problem, originating in the first-order Born approximation. This method is compared to the iterative Marčenko approach in Subsec. D.6.

3.4 Limitations of the iterative Marčenko approximation

In the previous subsection we discussed an iterative scheme for inverse scattering based on the Marčenko integral equation. We now want to analyse its quality regarding convergence.

We assume that the potential V has a finite support $\operatorname{supp}(V) \subseteq [0, a]$. In Ref. [23] it is proven that

$$x < x_{\mathrm{cr}} := \frac{\pi}{4 \max(|\mathcal{R}|)} \tag{44}$$

is a sufficient condition for the convergence of the Marčenko iteration in Eq. (34), i.e. this is the least convergence distance. Nevertheless, we do not know under which circumstances the iteration converges beyond this limit. Additionally, as shown in the previous subsection 3.3, the iteration does not converge if the bound-states-contribution in $\mathcal{R}$ is taken into account, i.e. the iteration is not applicable to potentials with bound states.

We want to focus on two kinds of potentials, namely nonnegative and zero-mean ones. Nonnegative potentials, i.e.

$$\forall x \in \mathbb{R} : V(x) \geq 0, \tag{45}$$

have no bound states, which means that the inverse scattering problem has a unique solution based on the reflection amplitudes (cf. Subsec. 3.1). Zero-mean potentials, i.e.

$$\int_{-\infty}^{\infty} V(x)\, dx = 0, \tag{46}$$

on the other hand have at least one bound state. In Ref. [24] there is a simple proof of this fact. So strictly speaking the Marčenko iteration cannot be used for zero-mean potentials. Still, we hope that at least the lowest orders constitute a good approximation, because the zeroth-order Marčenko approximation has the same property (cf. Eq. (39)). We intend to treat both nonnegative and zero-mean potentials with Deep Learning methods in Subsubsecs. 5.2.1 and 5.2.2, respectively.

We first study the convergence characteristics for the class $\mathcal{S}^{0.02,20,0.02}_{0,10,10/511}(10, 0.1, +)$ (cf. Subsec. B.2) based on a single randomly chosen sample, which is shown in Fig. 8. The left reflection amplitude $r(k)$ and its Fourier transform $\mathcal{R}(x)$ (cf. Eq. (29)) are plotted in Fig. 9. With Eq. (44) we can calculate the least convergence distance: $x_{\mathrm{cr}} \approx 0.686$. The Marčenko iteration is done up to $V^{(500)}_{\mathrm{Marč}}$. In Fig. 10 we compare the original potential to some stages of the iteration process. We see that for $x < x_{\mathrm{cr}}$ (indicated by a gray

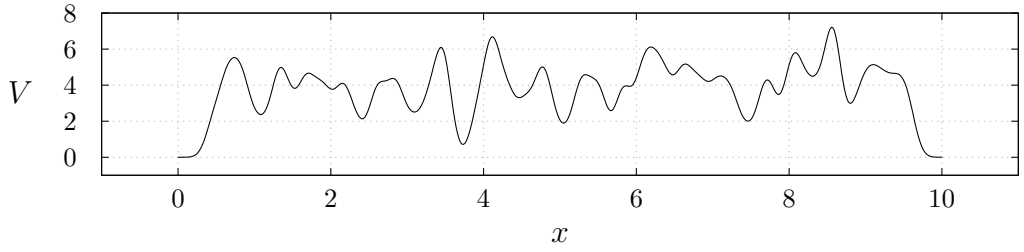

Figure 8: Instance of a random potential of the class $\mathcal{S}^{0.02,20,0.02}_{0,10,10/511}(10, 0.1, +)$, on which the Marčenko iteration is tested.

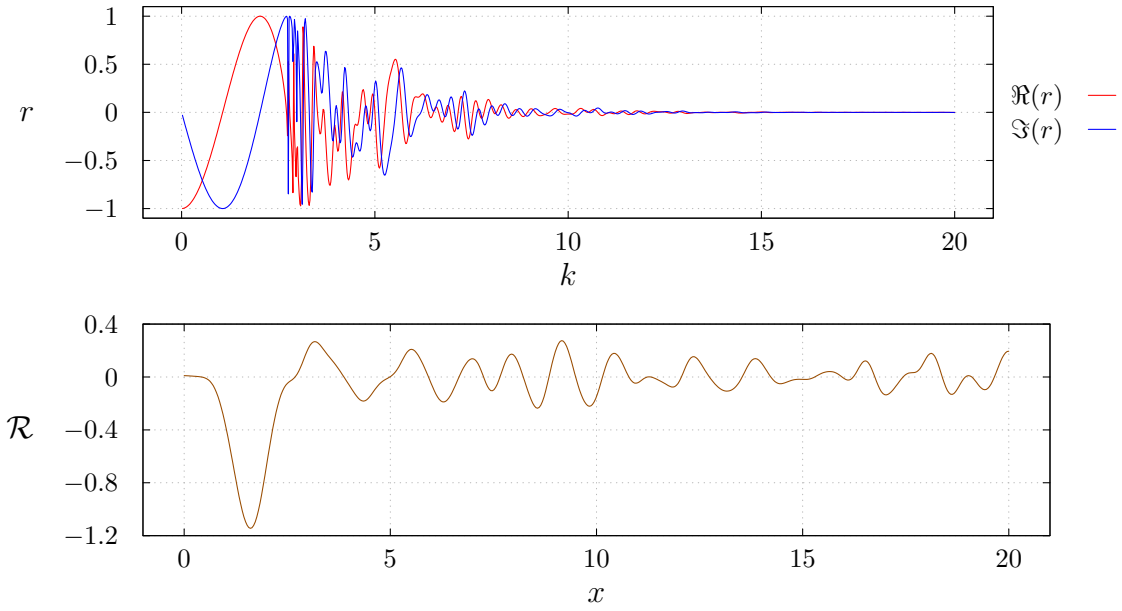

Figure 9: On the top: Momentum-dependent left reflection amplitude $r(k)$ of the potential in Fig. 8. On the bottom: Fourier transform $\mathcal{R}(x)$ of the left reflection amplitude $r(k)$.

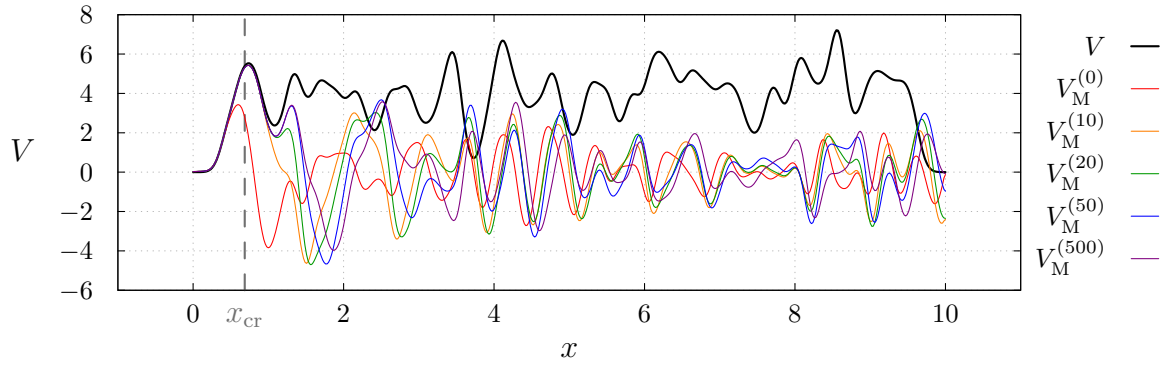

Figure 10: Comparison of the ground truth potential $V(x)$ from Fig. 8 to the left ν^{th}-order Marčenko approximation for $\nu \in \{0, 10, 20, 50, 500\}$. The least convergence distance is indicated by a dashed line marked with x_{cr}.

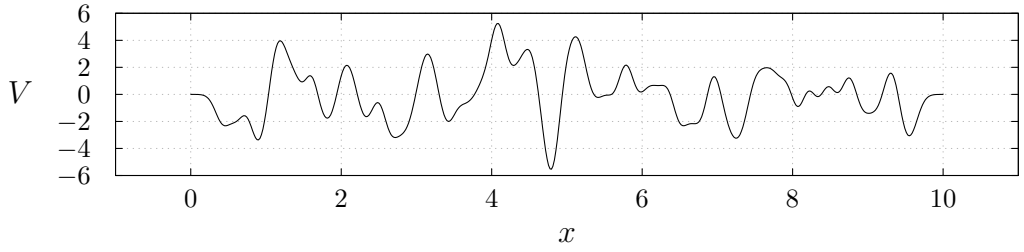

Figure 11: Instance of a random potential of the class $\mathcal{S}^{0.02,20,0.02}_{0,10,10/511}(10, 0.1, 0)$, on which the Marčenko iteration is tested.

dashed line) the iteration converges within less than 10 steps. For $x > x_{\mathrm{cr}}$ there is still a small region where the iteration converges satisfactorily fast. But for all other x-values beyond this region, the Marčenko iteration either does not converge at all or too slowly (one can see that the height of the second peak hardly changes from the 50^{th} iteration step to the 500^{th}). The 500 iterations already took approximately 20 seconds on the used machine (cf. Sec. A), so the Marčenko iteration alone is not a good candidate for a fast inversion procedure for random media.

We now take a look at the class $\mathcal{S}^{0.02,20,0.02}_{0,10,10/511}(10, 0.1, 0)$. Figs. 11, 12 and 13 show a potential from this class, its reflection amplitude $r(k)$ with its Fourier transform $\mathcal{R}(x)$ and the left ν^{th}-order Marčenko approximation for $\nu \in \{0, 10, 20, 30\}$, respectively. Although $\mathcal{R}(x)$ suggests a least convergence distance of $x_{\mathrm{cr}} \approx 1.2$, one clearly sees that the iteration converges essentially nowhere. In fact, the zeroth-order approximation has the least L^2-distance from the original potential. On the other hand, we have to admit that this is an extreme example and there are instances where convergence is far better

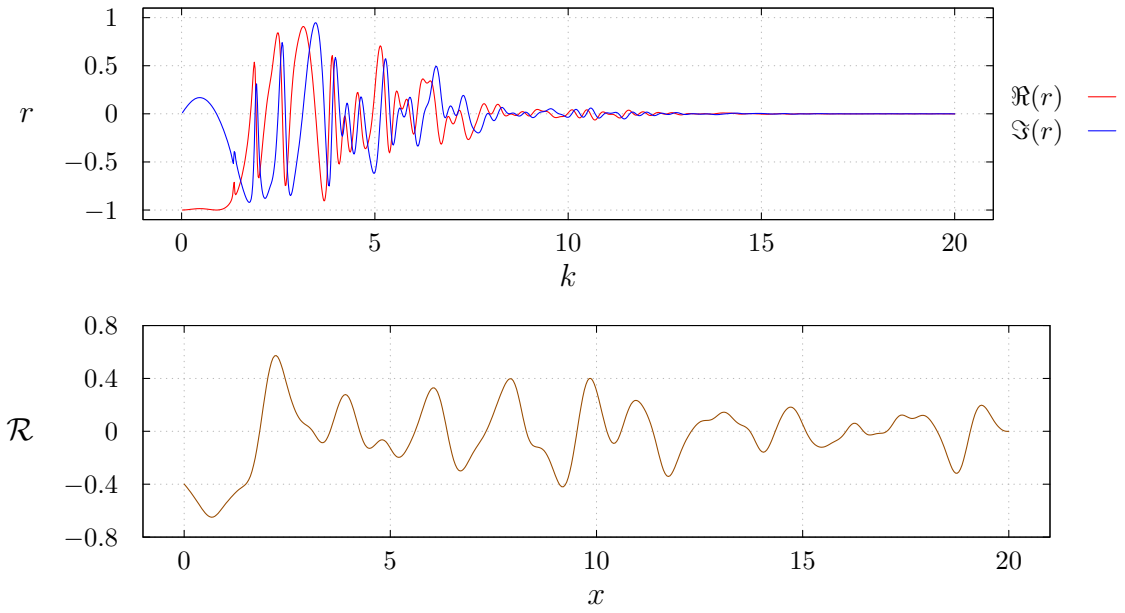

Figure 12: On the top: Momentum-dependent left reflection amplitude $r\,(k)$ of the potential in Fig. 11. On the bottom: Fourier transform $\mathcal{R}\,(x)$ of the left reflection amplitude $r\,(k)$.

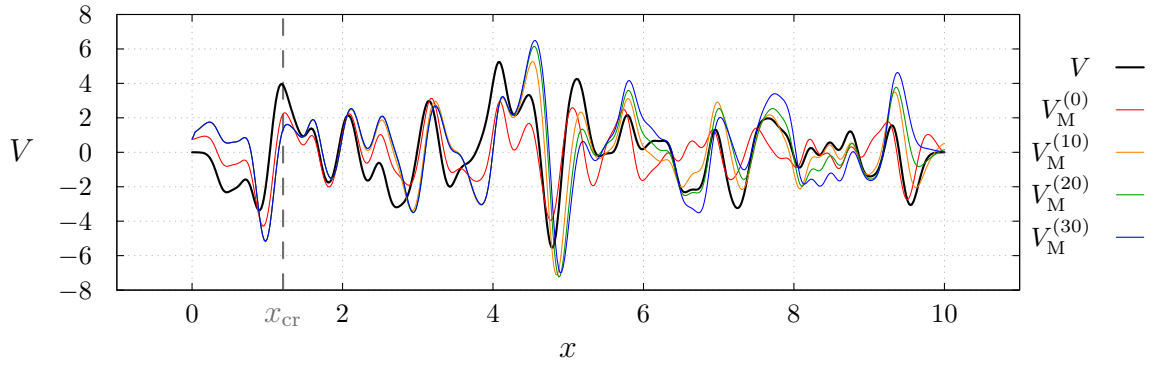

Figure 13: Comparison of the ground truth potential $V\,(x)$ from Fig. 11 to the left ν^{th}-order Marčenko approximation for $\nu \in \{0, 10, 20, 30\}$. The suggested least convergence distance (by Eq. (29)) is indicated by a dashed line marked with x_{cr}.

(only in certain regions and up to a certain iteration step). Still, we can conclude that the iteration is not suited for all zero-mean potentials.
We see that the Marčenko iteration alone does not yield satisfactory results for non-negative as well as for zero-mean potentials. In Subsec. 5.2 we address this problem with methods from Deep Learning. Since this is a new topic in my research group, I want to give a comprehensive introduction to the fundamentals of Deep Learning in the subsequent section.

4 Basics of Deep Learning

At first we want to clarify the concept of "Deep Learning" (DL) in the context of "Artificial Intelligence" (AI) and "Machine Learning" (ML).
AI can be defined as *"the effort to automate intellectual tasks normally performed by humans"* [25]. Thus AI also encompasses algorithms that are hard-coded and do not include any kind of "learning".
ML is a subdiscipline of AI. The task a ML algorithm is supposed to perform is not implemented hard-coded by hand, but rather learned by the algorithm itself by looking at data. During training the ML algorithm identifies statistical correlations between the presented input and output data, which are then (after learning) available as "rules" to process new data. This is a completely different paradigm compared to classical programming, where an algorithm is fed with rules and input data and produces some output data. Another way of describing a ML algorithm is that it searches inside a predefined hypothesis space a more useful representation of the input data in order to fulfil the task at hand.
DL is again a subfield of ML, where the just mentioned representation is realized as a sequence of simpler transformations in a layer-wise manner, each layer constituting a successively better representation. The term "deep" comes from the fact that DL algorithms usually consist of several layers.
For historical backgrounds or a more comprehensive introduction to DL we refer to the literature, especially to Ref. [25].

4.1 Artificial Neural Networks

The structure which underlies the method of DL is an Artificial Neural Network (ANN), also called a **model**. Mathematically speaking a model is nothing else than a family of continuous maps $f_w : \mathbb{R}^{n_0} \to \mathbb{R}^{n_L}$ (spanning the **hypothesis space**) parameterized by "trainable" parameters $w \in \mathbb{R}^d$.
A model is defined by a specific architecture. Here, we only describe the simplest architecture, a so-called fully connected feedforward neural network, in detail. All the other more sophisticated architectures are special forms, extensions or modifications thereof. Some of the most important ones are briefly highlighted in Subsec. 4.4.
A fully connected feedforward neural network (cf. Fig. 14) consists of a fixed number ($L+1$ with $L \in \mathbb{N}$) of **layers**. The layer with index 0 is called the input layer and the

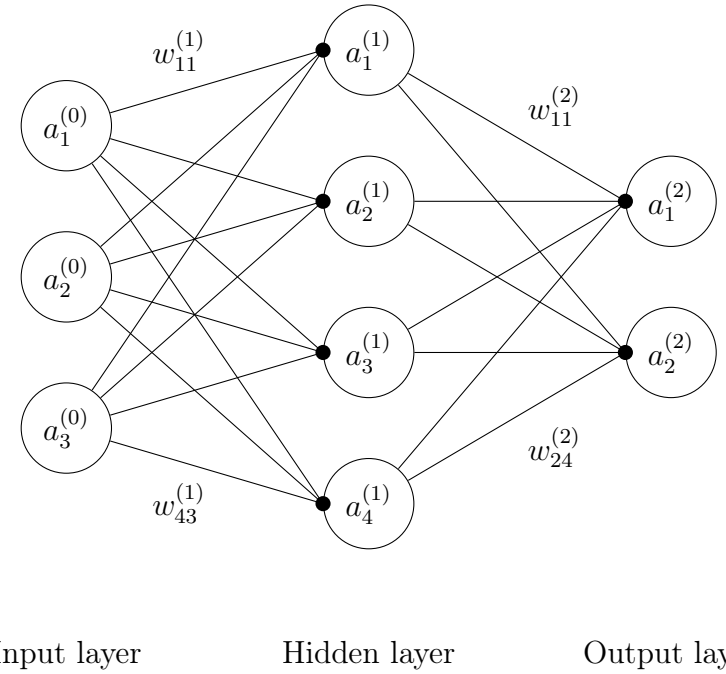

Figure 14: Schematic architecture of a fully connected feedforward neural network with $L + 1 = 3$ layers. The different layers contain $n_0 = 3$, $n_1 = 4$ and $n_2 = 2$ nodes, respectively. The nodes are indicated by circles, the connections (weights) by lines and the application of the activation function by a dot. The biases are not indicated and not all weights are labelled. This model has $d = 26$ trainable parameters (including the biases).

layer with index L is called the output layer, all the other layers inbetween are called hidden layers. Each layer contains a fixed number of so-called **artificial neurons** (or **nodes**) each holding a numerical value, called its **activation**. We denote by n_l the number of artificial neurons in layer $l \in \{0, \ldots, L\}$. Let $a_i^{(l)}$ be the activation of the i^{th} artificial neuron ($i \in \{1, \ldots, n_l\}$) in the l^{th} layer. The activations $a_i^{(0)}$ of the artificial neurons in the input layer are simply the values of the input vector. The activation $a_i^{(l)}$ for $l \geq 1$ is calculated from the activations of all the artificial neurons in the preceding layer as

$$a_i^{(l)} = \phi^{(l)} \left(b_i^{(l)} + \sum_{j=1}^{n_{l-1}} w_{ij}^{(l)} a_j^{(l-1)} \right) =: \phi^{(l)} \left(z_i^{(l)} \right). \tag{47}$$

The numbers $b_i^{(l)} \in \mathbb{R}$ and $w_{ij}^{(l)} \in \mathbb{R}$ are called the **biases** and **weights** of the l^{th} layer, respectively. They constitute the trainable parameters $w \in \mathbb{R}^d$ of the model. The number of trainable parameters amounts to

$$d = \sum_{l=1}^{L} \left(n_l + n_l n_{l-1} \right) = \sum_{l=1}^{L} n_l \left(1 + n_{l-1} \right). \tag{48}$$

The map $\phi^{(l)} : \mathbb{R} \to \mathbb{R}$ is called the **activation function** of the l^{th} layer. Since the whole model should learn nonlinear mappings, not all the activation functions are chosen linear. It is common to use monotonically increasing activation functions. Of course the range of the activation function in the output layer must match the possible range of desired output vectors. Some common activation functions are listed below:

- Sigmoid (mostly outdated): $\phi(x) = \dfrac{1}{1 + \mathrm{e}^{-x}}$.
- Hyperbolic tangent: $\phi(x) = \tanh(x)$.
- Rectified linear unit (ReLU): $\phi(x) = x\,\Theta(x)$.
- Leaky ReLU: $\phi(x) = \begin{cases} 0.01x & x < 0 \\ x & x \geq 0 \end{cases}$.
- Parametric ReLU (PReLU): $\phi(x) = \begin{cases} \alpha x & x < 0 \\ x & x \geq 0 \end{cases}$ with trainable α.
- Exponential linear unit (ELU): $\phi(x) = \begin{cases} \alpha\left(\mathrm{e}^x - 1\right) & x < 0 \\ x & x \geq 0 \end{cases}$ with fixed $\alpha \geq 0$.

- Scaled ELU (SELU): $\phi = 1.0507\, \phi_{\mathrm{ELU}}|_{\alpha=1.67326}$ [26].

If the activations of the nodes of a layer are calculated by Eq. (47), then this layer is called **dense**. Fig. 14 schematically shows the architecture of a simple fully connected feedforward neural network.
We now address the versatility of ANNs, on which we can learn something through the universal approximation theorem, stated in the following subsection.

4.2 Universal approximation theorem

George Cybenko [27] was one of the first to prove the so-called universal approximation theorem: Let $\phi \in C(\mathbb{R})$ be a continuous function with $\lim_{x\to-\infty} \phi(x) = 0$ and $\lim_{x\to+\infty} \phi(x) = 1$. Let $f \in C([0,1]^n)$ be a continuous function on the n-dimensional unit hypercube and $\varepsilon > 0$, then there exists an integer $N \in \mathbb{N}$, N scalars $\alpha_i, \theta_i \in \mathbb{R}$ and N vectors $y_i \in \mathbb{R}^n$ such that the function $g_{\alpha,\theta,y} \in C([0,1]^n)$ defined by

$$g_{\alpha,\theta,y}(x) := \sum_{i=1}^{N} \alpha_i \phi\left(y_i^\top x + \theta_i\right) \tag{49}$$

fulfils

$$\forall x \in [0,1]^n : |g_{\alpha,\theta,y}(x) - f(x)| < \varepsilon. \tag{50}$$

This means that a fully connected feedforward neural network with a single hidden dense layer is able to approximate any compactly supported continuous function with arbitrary accuracy, provided that it possesses sufficiently many nodes. In practice however one chooses deeper architectures with more hidden layers and less nodes per layer.
In order to get a model to perform the task we want it to perform, it has to "learn" from a sufficiently large set of examples. This procedure is called **learning**, **fitting** or **training**. How this is done is explained in the next subsection.

4.3 Fitting of a model

4.3.1 Cost function

In order to be able to train a model, one needs a quantitative measure for how good the performance of the model is. Depending on the task the model is supposed to perform, one has to choose a **cost function** (or loss function) $C : \mathbb{R}^{n_L} \times \mathbb{R}^{n_L} \to \mathbb{R}$. Given an

input-output pair $(x, y) \in \Gamma$, where $\Gamma \subseteq \mathbb{R}^{n_0} \times \mathbb{R}^{n_L}$ is the whole data space (the set of all possible samples), the cost function $C\left(f_w\left(x\right), y\right)$ should be a measure of how far off the model prediction $f_w\left(x\right)$ is from the desired output vector (the so-called ground truth) y.
The aim of fitting a model is to find the best model in the hypothesis space $\{f_w\}$, i.e. minimize the average

$$\mathscr{C}\left(w\right) := \left\langle C\left(f_w\left(x\right), y\right)\right\rangle_{(x,y)\in\Gamma}. \tag{51}$$

Before discussing how to do this minimization in the next subsubsection, we want to list some common DL tasks (T) with examples (Ex) and (partly empirically, partly theoretically) verified good choices of activation functions (AF) in the output layer and cost functions (CF):

- T: Regression: Output vectors have arbitrary range.
  Ex: Estimate the price of a house from attributes like number of rooms, accessibility to highways and local crime rate (this is referring to the Boston Housing Price dataset, cf. [28]).
  AF: Linear: $\phi^{(L)}\left(x\right) = x$.
  CF: Mean squared error (MSE): $C\left(y', y\right) = \frac{1}{n_L}\left\|y' - y\right\|_2^2$, mean absolute error (MAE): $C\left(y', y\right) = \frac{1}{n_L}\left\|y' - y\right\|_1$. n_L is the dimension of the vectors y' and y.

- T: Binary classification: Output vectors are real numbers $y \in [0, 1]$ representing the probability of one class (out of two classes).
  Ex: Determine if a review text is positive or negative.
  AF: Sigmoid: $\phi^{(L)}\left(x\right) = \left(1 + \mathrm{e}^{-x}\right)^{-1}$.
  CF: Binary crossentropy: $C\left(y', y\right) = -y\ln\left(y'\right) - \left(1 - y\right)\ln\left(1 - y'\right)$.

- T: Multi-label multiclass classification: Output vectors have components $y_i \in [0, 1]$ representing the probabilities for each class.
  Ex: Determine which kinds of animals are present in a picture.
  AF: Sigmoid: $\phi^{(L)}\left(x\right) = \left(1 + \mathrm{e}^{-x}\right)^{-1}$.
  CF: Sum of binary crossentropies: $C\left(y', y\right) = -\sum_i \left(y_i \ln\left(y_i'\right) + \left(1 - y_i\right)\ln\left(1 - y_i'\right)\right)$.

- T: Single-label multiclass classification: Output vectors have components $y_i \in [0, 1]$ with $\sum_i y_i = 1$ representing a probability distribution over the mutually exclusive classes.
  Ex: Recognize a handwritten digit.

AF: Softmax: $\phi_i^{(L)}(x) = \mathrm{e}^{x_i} / \sum_j \mathrm{e}^{x_j}$.
CF: Categorical crossentropy: $C(y', y) = -\sum_i y_i \ln(y'_i)$.

4.3.2 Backpropagation algorithm

In Eq. (51) we formulated the learning process as the minimization problem of an average cost function $\mathscr{C} : \mathbb{R}^d \to \mathbb{R}_0^+$ with respect to all trainable parameters w. The hypothesis space (spanned by the w's) usually has a very high dimensionality d. It is empirically validated that already simple minimization procedures lead to relatively good results. The most basic iterative algorithm is gradient descent:

$$w_{(k+1)} = w_{(k)} - \eta \nabla_w \mathscr{C}\left(w_{(k)}\right), \tag{52}$$

where $\eta > 0$ is the so-called **learning rate**.
A **hyperparameter** of a model is a parameter which has to be set by hand, e.g. number of layers and nodes, choice of activation functions and cost function, learning rate and other parameters related to the learning algorithm.
One problem of Eq. (52) is that we do not know the function $\mathscr{C}$ since one does not know the "shape" of the data space Γ (otherwise the DL problem would be solved already). Thus one approximates $\mathscr{C}$ by a mean value over a finite subset of Γ, usually called a **batch** $\{(x_\beta, y_\beta) : 1 \leq \beta \leq B\}$, where $B \in \mathbb{N}$ is the batch size and $\forall \beta : (x_\beta, y_\beta) \in \Gamma$.

$$\mathscr{C}(w) \approx \frac{1}{B} \sum_{\beta=1}^{B} C\left(f_w(x_\beta), y_\beta\right). \tag{53}$$

Let $N \in \mathbb{N}$ be the total number of available samples. If the gradient $\nabla_w \mathscr{C}$ is calculated using Eq. (53) with a batch size of $B = 1$ / $1 < B < N$ / $B = N$, one speaks of stochastic / mini-batch / batch gradient descent. One usually uses a certain fraction of all the N available samples for validation purposes, i.e. the model is not trained on these samples.
Since all the derivatives of all the functions involved in $C\left(f_w(x_\beta), y_\beta\right)$, i.e. the cost function and all the activation functions, are known analytically, one can calculate its gradient w.r.t. w quite easily using the chain rule. The resulting equations can be reformulated in terms of matrix multiplications. This linear-algebra-formulation of the calculation of $\nabla_w C\left(f_w(x_\beta), y_\beta\right)$ is called the **backpropagation algorithm**. (Due to the chain rule one has to "propagate" through all layers starting at the output layer

back to the input layer.) The backpropagation algorithm permits a certain degree of parallelization which can be exploited by graphics processing units (GPUs), which are typically very well suited for linear algebra computations.
Besides gradient descent there are many other optimization algorithms (cf. Ref. [29]), some also make use of the Hessian matrix of the cost function.
If one subdivides the $N' \in \mathbb{N}$ training samples into $\left\lfloor \frac{N'}{B} \right\rfloor$ batches of size B (and one batch of size $N' \mathrm{mod} B$), then all the N' training samples can be used to perform $\left\lceil \frac{N'}{B} \right\rceil$ iterative optimization steps. One such sweep of all training samples is called an **epoch**. Training is usually performed in multiple epochs since the model only learns "a bit" from a batch when taking an optimization step on it. After each epoch the training data is shuffled in order to alleviate overfitting, an undesirable phenomenon omnipresent in the training process. This phenomenon and possible counterstrategies are presented in the next subsubsection.

4.3.3 Overfitting and Regularization

A central problem in DL is the interrelation between **optimization** and **generalization**. The former means to train a model to perform better and better on the training data, whereas the latter is the capability of the model to also perform well on data it has not encountered before. The main aim is to get good generalization, but one can only influence the optimization process.
At the beginning of training the model has not yet learned all the important features in the data (**underfitting**). Training and validation loss are decreasing (more or less) proportionally. At some later point however, the model will start to learn patterns that are specific to the training data but irrelevant for unseen data (**overfitting**). Hence the generalizability of the model will get poorer, i.e. the validation loss will stagnate and eventually even start to rise.
Overfitting is naturally best prevented by using more training data. If this is not possible, then one has to put restrictions on the model such that it cannot afford to learn too specific features, leading to a better generalization. This procedure is called **regularization**.
A simple regularization method is to reduce the capacity of the model (determined by the number of trainable parameters) by using fewer layers, fewer nodes in the layers and so forth. However, the capacity should be large enough, such that the model does not underfit.

Another common regularization technique is **weight regularization**: in the spirit of Occam's razor [30] one should favour "simpler" models, i.e. models where the entropy of the distribution of the trainable parameters is low. This is achieved by adding to the average cost function $\mathscr{C}$ a penalty term (usually $\propto \|w\|_1$ or $\propto \|w\|_2^2$) which forces the weights of the model to take smaller values. The prefactor (i.e. the importance) of this penalty term is a hyperparameter which has to be chosen by hand.
Yet another simple way to mitigate overfitting is **dropout** [31]: During training a certain percentage p of nodes in the model is randomly set to zero ("dropped out"). When the model is evaluated (on test or validation data) every node is multiplied by the factor p to compensate for the higher number of active nodes. It is not yet fully (mathematically) understood why this often works remarkably well. A heuristic explanation might be that the noise introduced by dropout makes the model less prone to accidental meaningless correlations. The so-called dropout rate p is a hyperparameter of the model.
One must keep in mind that when doing the hyperparameter search by trial and error (or more sophisticated optimization algorithms), the hyperparameters of the model eventually overfit to the validation data. Therefore one should always prepare a separate test data set (or produce new training data for each new training process if possible).
There are two remaining issues concerning the training of a model, which we want to address in the next two subsubsections.

4.3.4 Batch Normalization

Consider a regression problem, where the value of an input-variable ranges over multiple orders of magnitude. It is (empirically) verified that models learn more easily if the distribution of the activations is standardized to zero mean and unit variance [32]. This is achieved by subtracting the mean value and then dividing by the standard deviation. **Batch normalization** [32] is a procedure built into the model as an individual layer, which standardizes the data on the fly: The layer holds an exponential moving average of the batch-wise mean and variance of the training data and is thus able to adaptively standardize data with changing mean and variance.

4.3.5 Initialization of the trainable parameters

The initialization of the trainable parameters should be random, i.e. not all parameters should be set to the same value. Another desired requirement is that the variance of the

activations and of the gradients (averaged over the training data and over the nodes in each layer separately) stay constant over the layers, because then the model can learn faster, i.e. the convergence of the training algorithm is better [33]. The most common initializations follow a truncated normal distribution with mean $\mu_{\text{init}} = 0$ and standard deviation σ_{init} or a uniform distribution over $\left[-\sqrt{3}\sigma_{\text{init}}, \sqrt{3}\sigma_{\text{init}}\right]$. There are different schemes suggesting different values for σ_{init} depending on the used activation function and on the number of nodes in the previous and current layer, n_{l-1} and n_l respectively:

- "LeCun" [34]: Good for $\phi = \tanh$, variance of the gradients not taken into account. Also good for $\phi = \phi_{\text{SELU}}$ [26]. $\sigma_{\text{init}} = \sqrt{\frac{1}{n_{l-1}}}$.
- "Glorot" (or "Xavier") [35]: Good for $\phi = \tanh$ and $\phi(x) = x$. $\sigma_{\text{init}} = \sqrt{\frac{2}{n_{l-1}+n_l}}$.
- "He" [36]: Good for $\phi = \text{ReLU}$. $\sigma_{\text{init}} = \sqrt{\frac{2}{n_{l-1}}}$.

From Eq. (47) we can see that a gradient propagating through the model does not pick up (multiplicatively) any biases. To be more specific, consider a layer λ somewhere before the layer l, i.e. $1 \leq \lambda < l$. Let $\mathsf{p}^{(\lambda)} \in \left\{b_m^{(\lambda)}, w_{mn}^{(\lambda)}\right\}$ be some trainable parameter of that layer. It then holds that

$$\frac{\partial a_i^{(l)}}{\partial \mathsf{p}^{(\lambda)}} = \frac{\partial \phi^{(l)}}{\partial z_i^{(l)}} \sum_{j=1}^{n_{l-1}} w_{ij}^{(l)} \frac{\partial a_j^{(l-1)}}{\partial \mathsf{p}^{(\lambda)}}. \tag{54}$$

Therefore the biases of the model can be initialized to zero without concern.

So far we have only discussed fully connected feedforward neural networks. However, there are other architectures as well, better suited for certain tasks. We want to present the two most important ones in the following subsection.

4.4 Advanced architectures

4.4.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are suitable for data with a certain "spatial" structure like images. Each convolutional layer consists of several so-called **channels**. For instance, if the input is an RGB-image, then the input layer may consist of 3 color channels. All channels of a layer are usually one-, two- or three-dimensional arrays of the same shape and size.

The connectivity of a convolutional layer to the next layer has two important characteristics: **locality** and **shared weights**. Locality implies a structure-conserving processing of the data and shared weights imply translational invariance.
Locality means that the activation of a certain node in a certain convolutional layer is calculated from the activations of only a specific subset of the nodes of the previous convolutional layer. The corresponding set of weights is called a **kernel** or a **filter**. Such a kernel is usually box-shaped.
Let $a_{c,\mathbf{i}}^{(l)}$ denote the activation of the node at position $\mathbf{i}$ in channel c in layer l. This activation value is calculated from the activations of the previous layer by

$$a_{c,\mathbf{i}}^{(l)} = \phi^{(l)} \left(b_{c,\mathbf{i}}^{(l)} + \sum_{c'} \sum_{\mathbf{j} \in U(\mathbf{i})} w_{c,c',\mathbf{i},\mathbf{j}}^{(l)} a_{c',\mathbf{j}}^{(l-1)} \right). \tag{55}$$

The c'-sum runs over all channels of the previous layer. The $\mathbf{j}$-sum is to be taken over a confined neighborhood $U(\mathbf{i})$ around the "pixel" $\mathbf{i}$. If $\mathbf{i}$ lies near or at the boundary of the data array, then an appropriate padding has to be invoked (or one is fine with the fact that the array shrinks by nearly the size of the kernel).
The second characteristic "shared weights" means that the filters $w_{c,c',\mathbf{i},\mathbf{j}}^{(l)}$ and biases $b_{c,\mathbf{i}}^{(l)}$ do not depend on $\mathbf{i}$, i.e. they are the same for every position:

$$a_{c,\mathbf{i}}^{(l)} = \phi^{(l)} \left(b_{c}^{(l)} + \sum_{c'} \sum_{\mathbf{j} \in U(\mathbf{i})} w_{c,c',\mathbf{j}}^{(l)} a_{c',\mathbf{j}}^{(l-1)} \right). \tag{56}$$

For a fixed channel c and pixel $\mathbf{i}$ this is visualized in Fig. 15.
Due to these two features (locality and shared weights) a CNN has considerably less trainable parameters than a fully connected feedforward neural network of the same size (i.e. number of layers and nodes per layer). An exemplary comparison is given later.
But why are these features beneficial (besides the reduced hypothesis space)? Take for example a CNN that should detect the presence of cats in images. Usually a certain pixel is more strongly correlated to nearby pixels than to pixels further away. This is why a kernel only has to be local. A CNN can nevertheless detect global structures in an image by processing the data through several convolutional layers, which has the effect of widening the field of view. Secondly, the CNN should be able to detect cats irrespective of their position in the image. This motivates and justifies the usage of

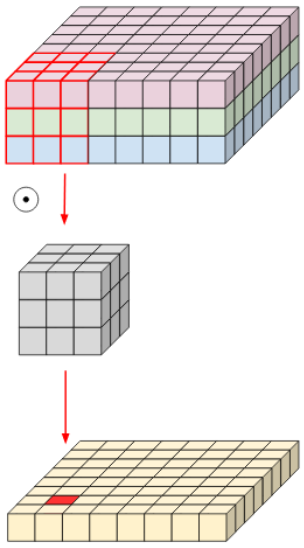

Figure 15: Schematic visualization of a convolutional layer for one output channel: On the top the input array with 3 channels (distinguished by their color) is depicted. The grey array symbolizes a kernel. The symbol $\odot$ denotes element-wise multiplication and subsequent summation. (The bias is omitted in this picture.) This operation is done for every kernel-sized region in the input array (indicated by red lines). The output channel is depicted at the bottom and the node, which holds the activation from the illustrated convolutional operation, is marked in red. (Taken from [37])

shared weights.
An alternative, even lighter, variant of a convolutional layer is the **depthwise separable convolution**. The idea is that the correlation across pixels in the same channel is more important than the correlation across different channels. A depthwise separable convolution consists of two parts (cf. Fig. 16):

1. Depthwise convolution: each channel is independently transformed (via convolution) into one or more representations.

2. Pointwise convolution: The transformed channels are stacked on top of another and convolved with kernels, which extend over only one pixel (but all channels).

This can be expressed in the following equation (assuming that in the first step each channel is transformed into only one representation):

$$a_{c,\mathbf{i}}^{(l)} = \phi^{(l)} \left(b_c^{(l)} + \sum_{c'} \tilde{w}_{c,c'}^{(l)} \sum_{\mathbf{j} \in U(\mathbf{i})} w_{c',\mathbf{j}}^{(l)} a_{c',\mathbf{j}}^{(l-1)} \right). \tag{57}$$

In order to quantify the significant reduction of the hypothesis space, we compare the number of trainable parameters d_1 between

- two dense layers, the first with $3 \cdot 224^2 = 150\,528$ nodes, the second with $64 \cdot 224^2 = 3\,211\,264$ nodes.
  $d_1 = 483\,388\,358\,656$.

- two convolutional layers, the first with 3 channels of shape 224×224, the second with 64 channels of shape 224×224, connected by kernels with a receptive field of size 3×3.
  $d_1 = 1\,792$.

- two depthwise separable convolutional layers with the same shapes as in the previous case. In the depthwise convolution each channel is transformed into one representation by convolution with a 3×3 kernel. The pointwise convolution is done with 64 kernels, i.e. the second layer has 64 channels.
  $d_1 = 283$.

A CNN consists of several convolutional layers stacked on top of another. The general working principle of a CNN is that the first layers identify simple features like edges in

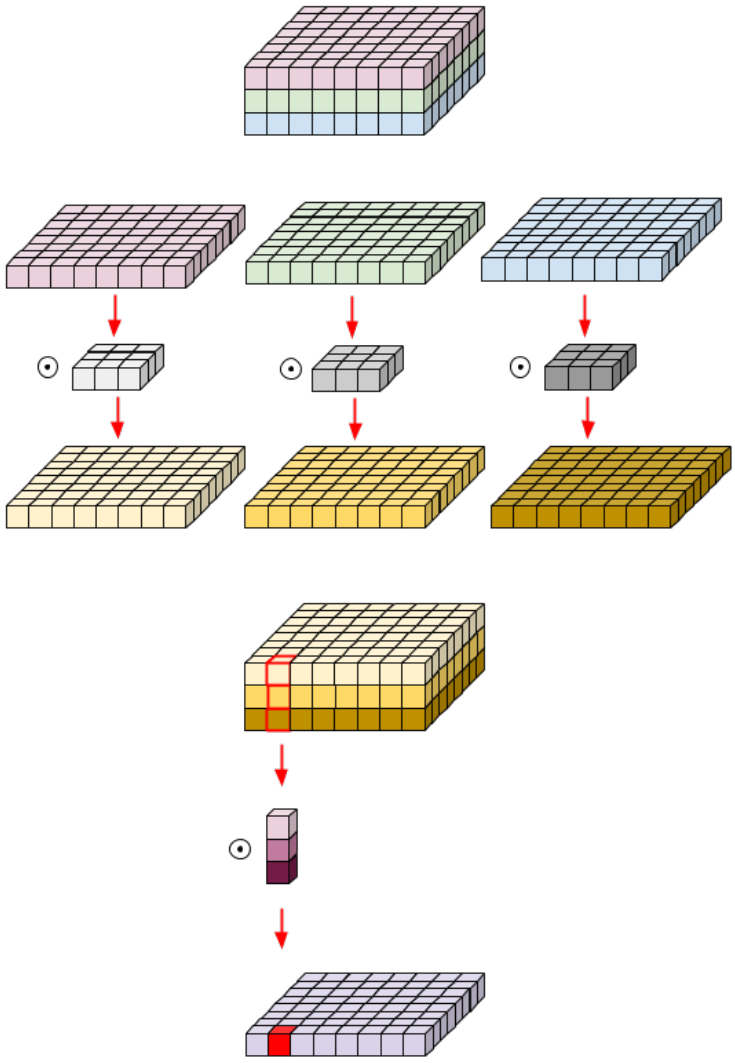

Figure 16: Schematic visualization of depthwise separable convolution for one output channel: The process is illustrated from top to bottom. At first, the 3 channels (distinguished by their color) are separated and each channel undergoes a convolution operation with an individual kernel. The channels transformed in this way (depicted as yellowish arrays) are stacked on top of each other. This stack is pointwise convolved; or viewed differently: the activation of a node in the output layer (marked in red) is a linear combination of the activations of the corresponding "pixels" in the transformed yellowish channels (indicated by red lines). (Taken from [37])

images. Deeper layers compose the information from the preceding layers into more and more sophisticated representations of the data, thereby features of higher and higher complexity can be detected [38].
The overall architecture of a CNN is often done in an **encoder-decoder-scheme**: In the encoder-part of the CNN the layers successively get compressed (in the pixel-direction) but deeper (in the channel-direction). Compression is (channelwise) achieved by **pooling** some neighboring pixels to a single pixel value, either taking the maximum or the mean value of the activations. In the subsequent decoder-part of the CNN the procedure is reversed. Such CNNs can be used for semantic image segmentation [39, and references therein], coloring greyscale-images [40, and references therein], denoising or sharpening images [41], to name a few examples.
If the decoder-part is replaced by some dense layers, a CNN can perform classification on the data (like recognizing handwritten digits [42]). More intricate architectures can perform more sophisticated tasks like artistic style transfer [43].

4.4.2 Recurrent Neural Networks

So far we have described ANNs with a fixed number of input nodes. However, there is a class called **Recurrent Neural Networks** (RNNs), which have a variable input and output size. They can internally store information and thus handle sequences of more or less arbitrary length. In order to describe sequences, we introduce a discrete time parameter $t \in \mathbb{N}$.
A node in an RNN is more complex than in fully connected feedforward neural networks, since it receives two input values: one is the output of the previous layer at the same time step, the other is the output of the same layer at the previous time step:

$$a_i^{(l)}(t) = f_{w^{(l)}}\left(a_j^{(l-1)}(t), a_k^{(l)}(t-1)\right). \tag{58}$$

We refer to the map $f_{w^{(l)}}$ of a particular layer l as a **cell** (also called a **unit**). The trainable parameters $w^{(l)}$ do not depend on the time t. Fig. 17 shows two common illustrations of a recurrent layer.
Care has to be taken when doing the backpropagation algorithm, i.e. when applying the chain rule, since (in the unwrapped picture) a certain trainable parameter appears multiple times in the network and there are several "paths" through the network to get to a specific trainable parameter in a specific node. This type of learning algorithm is called **backpropagation through time**.

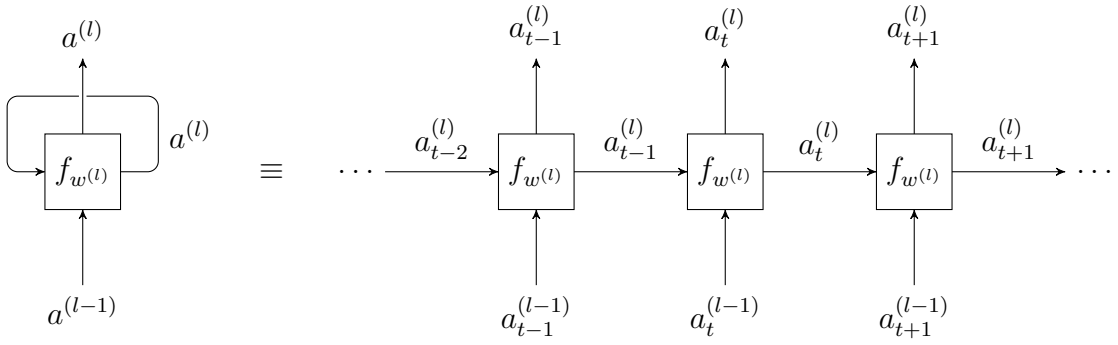

Figure 17: Schematic illustration of a recurrent layer. Due to lack of space $a^{(l)}(t)$ is written as $a_t^{(l)}$. On the left is the "wrapped" picture, on the right the "unwrapped" picture of the recurrent layer. The cell is indicated by a square and the data flow is depicted by arrows. At a given point in time the cell receives as inputs: the output from the previous layer at the same point in time, and the output from itself from one time step ago.

The original plain recurrent layer is defined by

$$a_i^{(l)}(t) = \tanh\left(b_i^{(l)} + \sum_{j=1}^{n_{l-1}} w_{ij}^{(l)} a_j^{(l-1)}(t) + \sum_{k=1}^{n_l} \tilde{w}_{ik}^{(l)} a_k^{(l)}(t-1)\right). \tag{59}$$

This simple cell however suffers from poor long-term memory [44, 45].

In 1997 Hochreiter and Schmidhuber presented an enhanced RNN cell called **Long Short-Term Memory (LSTM)** [46], which has an explicit internal memory state and different gates to control the flow of information inside the cell. Unlike in the plain RNN cell, the memory is not overwritten in every time step, but information can be held in the internal state over longer time periods. Ref. [47] gives a nice insight into the inner workings of LSTM cells. There exist several different variations of LSTM [48].

In 2014 Cho et al. introduced the **Gated Recurrent Unit (GRU)** [49], which is a simplified LSTM unit. It has less representational power than the LSTM, but it is easier to implement and faster to compute.

Both LSTM and GRU outperform the plain RNN cell. However, which of the two performs better on a given task depends on the task itself [47, 50, 51].

An RNN allows for different mappings of data, see Fig. 18. We want to give examples for each case:

- One to many: image captioning (i.e. image to text).

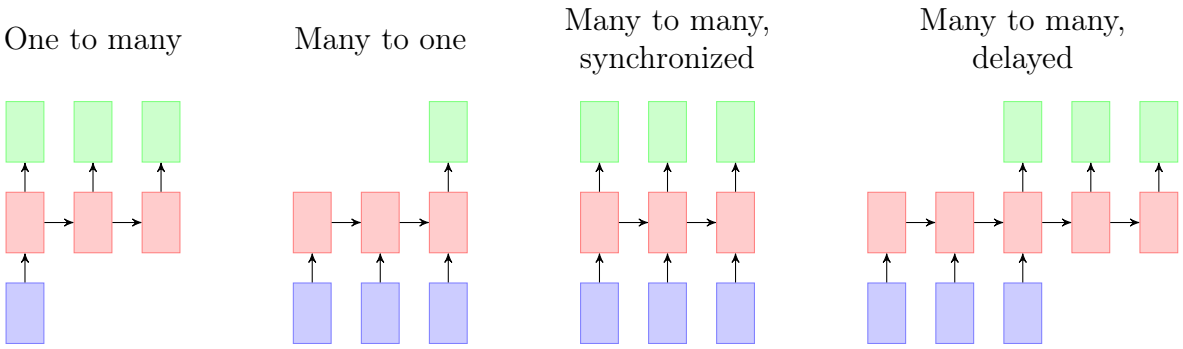

Figure 18: Depiction (in the unwrapped picture) of the diverse mappings an RNN can do. The blue boxes symbolize the input, the green boxes the output of the RNN. The red boxes symbolize the RNN itself (at different points in time).

- Many to one: text classification (i.e. text to label).
- Many to many, synchronized: frame-by-frame analysis of videos.
- Many to many, delayed: translation (i.e. text to text).

Ref. [52] gives an entertaining and insightful overview over what RNNs can and cannot do regarding text synthesis.
An RNN can also be operated bidirectionally. In this case there is in each layer a forward and (in parallel) a backward processing of the data.

4.4.3 Mixed and enhanced architectures

The different layers discussed in the previous subsubsections can be combined to form even more diverse models. A few examples:

- Convolutional layers + dense layers: image classification.
- Recurrent layers + dense layers: text classification.
- Recurrent convolutional layers for film analysis.

Another concept we want to introduce is **residual connections** [53]. Very deep CNNs are usually quite hard to train due to vanishing or exploding gradients. Another issue is that information can get lost in the downsampling (pooling) steps in an encoder-part of a CNN. In order to mitigate these effects one can introduce residual connections into the architecture of the model. This means that the input of a certain layer consists not

only of the output of the previous layer but also of the output of one or more layers earlier in the network. A typical example is the so-called U-Net [54].
Ref. [55] lists and compares some common CNN architectures.

In the subsequent subsection we briefly outline some other important DL concepts one should at least have heard of.

4.5 Further learning concepts

4.5.1 Transfer Learning

A model trained to perform a certain task can be reused to build a model, whose goal is to perform a more or less similar task as the original one. This is called **Transfer Learning**.
For instance: One has a pre-trained CNN able to distinguish cats from dogs and one wants a model being able to distinguish rabbits from hamsters. Then one can take the cats-dogs-CNN and train it on rabbits-hamsters-data while keeping the trainable parameters of the earlier layers fixed (e.g. train only the last dense classification layers). A heuristic explanation why this works is because the earlier layers usually recognize very basic features like edges, colors and rough shapes, which are present in both tasks. The more different the new task is (e.g. distinguish cars from trucks), the less layers can be kept fixed and more of the later layers have to be retrained.

4.5.2 Unsupervised Learning

All Deep Learning methods we have investigated so far require labelled data, i.e. one has to provide pairs of input data and target output data. Dealing with unlabelled data is referred to as **unsupervised Learning** (as opposed to **supervised Learning**).
An encoder-decoder-CNN is an example for an architecture capable of unsupervised Learning. In this context it is also called an **autoencoder**. By taking the input data as the desired output data, one forces the model to learn more compressed representations of the data. This (nonlinear) dimensionality reduction can be compared to principal component analysis (PCA) [56]. PCA is a linear statistical tool to extract the most relevant directions of a high-dimensional data set.

4.5.3 Generative Adverserial Networks

Generative Adverserial Networks (**GANs**) are an example for unsupervised Learning [57]. GANs are able to produce synthetical data similar to the training data, e.g. to come up with fake celebrity images [58].
A GAN consists of 2 neural networks: a **generator** network G and a **discriminator** network D. The aim of G is to produce data which looks like the training data. The aim of D is to identify if a sample is produced by G or if it is taken from the training data. The performance of D is measured by the fraction of correctly classified samples, whereas the performance of G is measured by the fraction of the mistakes that D makes. Due to this feedback loop both networks G and D simultaneously get better and better at their respective tasks during training.

4.5.4 Reinforcement Learning

The general setup of **Reinforcement Learning** is as follows: an **agent** is situated in an **environment** and can observe its **state** (the whole environment or only parts of it). Based on the observation the agent performs **actions**, which may change the state of the agent itself and/or the state of the environment. The agent rarely receives rewards, often delayed. The aim of Reinforcement Learning is to find an optimal strategy for choosing which actions to take in order to maximize the cumulative reward. This strategy can be implemented by an ANN.
Some applications of Reinforcement Learning are playing games (like AlphaGo [59]), autonomous driving and robotics.

With the first three sections we set up the theoretical framework for the main purpose of this thesis: the application of DL methods to one-dimensional inverse scattering, which is treated in the following section.

5 Application of Deep Learning to Inverse Scattering

In this section we present and evaluate our proposed approach to solve the one-dimensional inverse scattering problem using techniques from Deep Learning. From now on we use units where $m = \hbar = 1$.

5.1 Delta-potentials

As a warm-up, we investigate potentials with a finite (small) number of positive delta-peaks. Such potentials are easy to handle, they are fully characterized by a small amount of numbers, namely twice as much as the number of peaks: each peak has a position and a strength (height). Furthermore, the forward scattering problem can be solved analytically (cf. Subsec. C.1). Since the peaks are all chosen positive, the potentials do not possess any bound states, meaning that the inverse scattering problem has a unique solution. We have not tried to use a Recurrent Neural Network in this setup, i.e. the numbers of output nodes of the used Artificial Neural Networks (ANNs) are fixed, which is why we also only consider potentials with a fixed number of delta-peaks. We choose the setup $\mathcal{D}^{0.1,15,0.1}_{-7,7,0.5}(7;1,10)$ (cf. Subsec. B.1). Fig. 19 shows that the localization length $\xi(k)$ corresponding to this class is smaller nearly over the whole considered k-range than the average length of the potentials $\langle L\rangle := \langle x_7 - x_1\rangle \approx 11.2$ (x_1 is the position of the leftmost delta-peak and x_7 the position of the rightmost delta-peak), which means that localization is dominant in this setup.
We want to train an ANN to predict the values x_i and V_i of a potential

$$V(x) = \sum_{i=1}^{N} V_i\, \delta(x - x_i)\,, \tag{60}$$

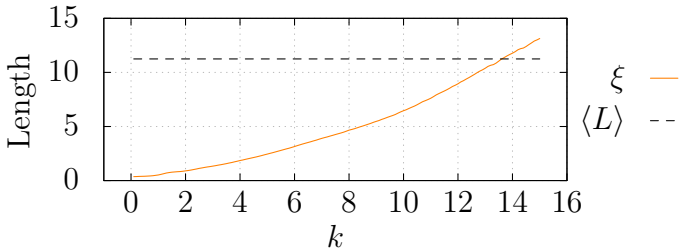

Figure 19: Momentum-dependence of the localization length ξ for the class $\mathcal{D}^{0.1,15,0.1}_{-7,7,0.5}(7;1,10)$. $\langle L\rangle \approx 11.2$ is the average length of the potentials. The localization length ξ as well as $\langle L\rangle$ are calculated using 10^5 samples.

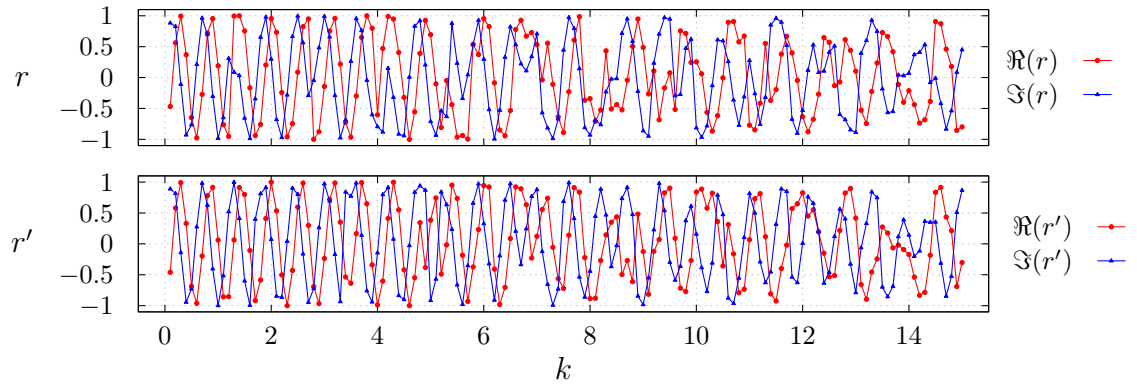

Figure 20: Example of an input vector the ANN receives, i.e. all the numerical values indicated by small dots and triangles.

with $N = 7$, $V_i \in [1, 10]$, $x_i \in [-7, 7]$ and $x_{i+1} - x_i > 0.5$, given its left and right reflection amplitudes $r(k)$ and $r'(k)$. This is a typical regression problem (cf. Subsubsec. 4.3.1). Since the values of the V_i are in the same order of magnitude as the positions x_i, we use the average of the mean squared errors as the cost function:

$$C(\mathbf{x}', \mathbf{V}'; \mathbf{x}, \mathbf{V}) = \frac{1}{2}\left(\frac{1}{7}\|\mathbf{x}' - \mathbf{x}\|_2^2 + \frac{1}{7}\|\mathbf{V}' - \mathbf{V}\|_2^2\right), \tag{61}$$

where $\mathbf{x}' = (x_1', \ldots, x_7')$ and $\mathbf{V}' = (V_1', \ldots, V_7')$ are the predictions of the ANN and the unprimed quantities are the respective ground truths. If the target output values lie in different orders of magnitude, one would have to standardize them (in the statistical sense) before showing them to the model (cf. Subsubsec. 4.3.4). The input values should be standardized as well.

An ANN only takes real values as input. Therefore we split the reflection amplitudes $r(k)$ and $r'(k)$ into their respective real and imaginary parts. This results in an input size (number of nodes in the input layer) $n_0 = 2 \cdot 2n_k$, where $n_k = 150$ is the number of k-values. Since $|r(k)| \leq 1$ and $|r'(k)| \leq 1$, all the input values lie in the interval $[-1, 1]$, which means that we do not have to standardize them. In Fig. 20, which shows a single exemplary input vector, we see that the data, which the ANN receives, is rather coarse-grained.

As an architecture we choose a fully connected feedforward neural network. Different hyperparameters were tested. During the hyperparameter search we always produced new training data before the fitting process of each newly initialized model. This is done so that the hyperparameters do not overfit to a single validation data set (cf.

Subsubsec. 4.3.3). The following architecture turns out to perform rather satisfactory (in a sense we quantify later):

- Number of layers $L + 1 = 10$, number of nodes in each layer: $n_0 = 4n_k = 600$, $n_1 = 600$, $n_2 = 520$, $n_3 = 440$, $n_4 = 360$, $n_5 = 280$, $n_6 = 200$, $n_7 = 120$, $n_8 = 40$, $n_9 = 2N = 14$.
- After each hidden layer there is a batch normalization layer. The number of trainable parameters is $d = 1\,253\,054$.
- The activation functions are chosen as ReLU, except for the last layer, where a linear activation function is used (because the output values can be negative as well). The initializers are "He" and "Glorot", respectively.
- The optimizer is Adam [60] with a learning rate of $5 \cdot 10^{-3}$.

Code Snippet 1 displays the Keras implementation of this ANN in order to eliminate all ambiguities.

Code Snippet 1 Keras implementation of the ANN architecture used for the inverse scattering of delta-potentials. `n_k` $= n_k = 150$ and `n_scatterers` $= N = 7$.

```
network = Sequential()
network.add(Dense(600, input_shape=(4*n_k,), activation='relu',
                  kernel_initializer='he_normal'))
network.add(BatchNormalization())
network.add(Dense(520, activation='relu', kernel_initializer='he_normal'))
network.add(BatchNormalization())
network.add(Dense(440, activation='relu', kernel_initializer='he_normal'))
network.add(BatchNormalization())
network.add(Dense(360, activation='relu', kernel_initializer='he_normal'))
network.add(BatchNormalization())
network.add(Dense(280, activation='relu', kernel_initializer='he_normal'))
network.add(BatchNormalization())
network.add(Dense(200, activation='relu', kernel_initializer='he_normal'))
network.add(BatchNormalization())
network.add(Dense(120, activation='relu', kernel_initializer='he_normal'))
network.add(BatchNormalization())
network.add(Dense(40, activation='relu', kernel_initializer='he_normal'))
network.add(BatchNormalization())
network.add(Dense(2*n_scatterers, activation='linear', kernel_initializer='glorot_normal'))
network.compile(loss='mean_squared_error', optimizer=Adam(lr=5e-3))
```

The data set consists of $5 \cdot 10^5$ samples, which is split into training and validation data at the ratio of $9 : 1$. Fitting is done with a batch size of 256 for 500 epochs, where the training data are shuffled after each epoch. Fig. 21 shows the training and the validation loss as a function of the epoch number.

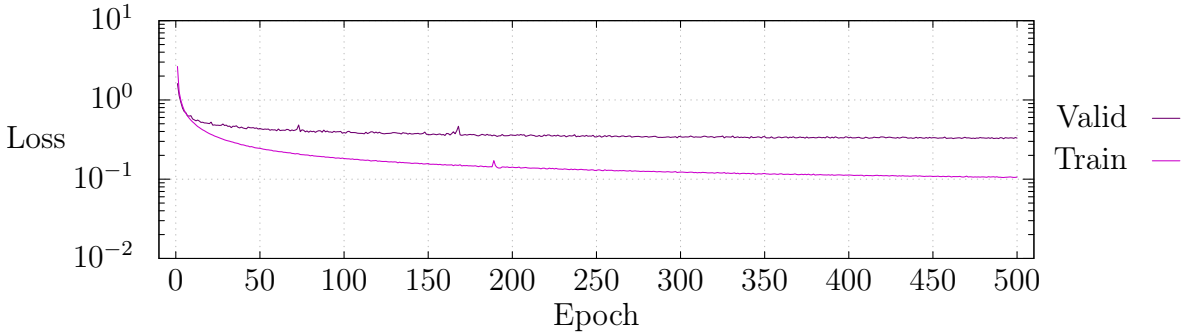

Figure 21: Learning curves of the model used for the inverse scattering of delta-potentials. "Train" is the training loss – the mean cost function value evaluated on the training data. "Valid" is the validation loss – the mean cost function value evaluated on the validation data, to which the model is not fitted.

In order to get a qualitative feeling about the performance of the model, we plot six test examples in Fig. 22. For a quantitative analysis, we compare the model against a benchmark, a search procedure and a nearest neighbour analysis. In each case we compute the average of the cost function

$$C\left(\mathbf{y},\mathbf{W};\mathbf{x},\mathbf{V}\right)=\frac{1}{2}\left(\frac{1}{7}\left\|\mathbf{y}-\mathbf{x}\right\|_2^2+\frac{1}{7}\left\|\mathbf{W}-\mathbf{V}\right\|_2^2\right), \tag{62}$$

over a test data set of size 10^4. $(\mathbf{x},\mathbf{V})$ denotes a test sample. Let $\mathbf{r}$ be the corresponding $4n_k$-dimensional real-valued vector containing the reflection amplitudes. What $(\mathbf{y},\mathbf{W})$ is, depends on the case:

- Model (ANN): $(\mathbf{y},\mathbf{W})=(\mathbf{x}',\mathbf{V}')$ is the prediction of the model, given $\mathbf{r}$ as an input. There is a subtle problem with this case, which is addressed below.
- Benchmark: $(\mathbf{y},\mathbf{W})$ is the average training datum $\langle(\mathbf{y},\mathbf{W})\rangle_{(\mathbf{y},\mathbf{W})\in\text{training data}}$.
- Search procedure: $(\mathbf{y},\mathbf{W})$ is *the* sample in the training data set (to which the model was fitted, to give a fair comparison), which minimizes $\left\|\mathbf{s}-\mathbf{r}\right\|_2^2$, where $\mathbf{s}$ are the reflection amplitudes of $(\mathbf{y},\mathbf{W})$.
- Nearest neighbour: $(\mathbf{y},\mathbf{W})$ is *the* sample in the training data set, which minimizes $C\left(\mathbf{y},\mathbf{W};\mathbf{x},\mathbf{V}\right)$.

The benchmark is the most basic guess one can make. It lies "in the middle" of the training data and has the least mean distance to any point in the data set.

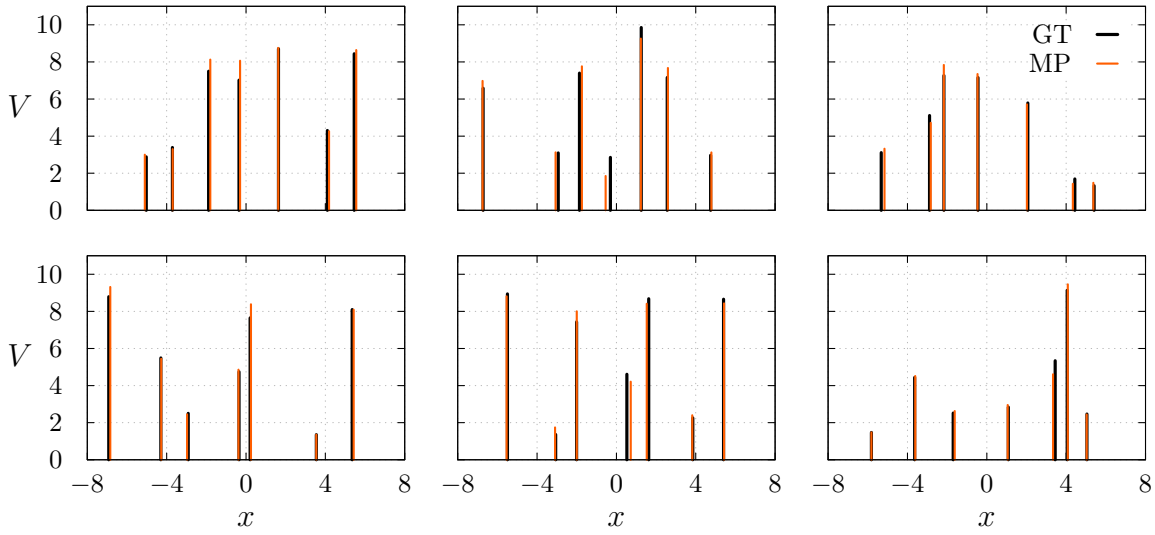

Figure 22: Six randomly generated test examples of the class $\mathcal{D}^{0.1,15,0.1}_{-7,7,0.5}(7;1,10)$. The height of each peak is the strength $V_i^{(\prime)}$ of the corresponding delta-peak. From the ground truths "GT" the reflection amplitudes are calculated and fed into the model, which returns the predictions "MP". The key, which is displayed only in the top right subplot, holds for every other subplot as well.

The search and the nearest neighbour procedures both look for the sample in the training data set, which lies closest to the given test sample. Both procedures however measure closeness in different ways.

The nearest neighbour approach is somehow unrealistic, because in a "real-life" scenario one is given $\mathbf{r}$ and the training data, but not $(\mathbf{x}, \mathbf{V})$. And with this information one can only do a benchmark guess, train a model or do a search by finding the sample in the training data set with the least "$\mathbf{r}$-distance" to the given $\mathbf{r}$. However, the nearest neighbour analysis gives the mean distance of a random sample to the nearest training data point, i.e. it is a measure of how dense the training data lie in the entire data space (consisting of *all* possible samples).

The performance of the model and of the search procedure both scale with the size of the training data set.

As mentioned above, there is a subtle problem when evaluating the performance of the model using the cost function in Eq. (62). To see this, we look at the sample in the test data set, which has the highest ANN-cost according to Eq. (62). Fig. 23 shows the ground truth of this sample compared to the model prediction, the benchmark, the search result and the nearest neighbour. The issue with the ANN is that it apparently confuses the small peaks and mixes up the order of the peaks. But the cost function in

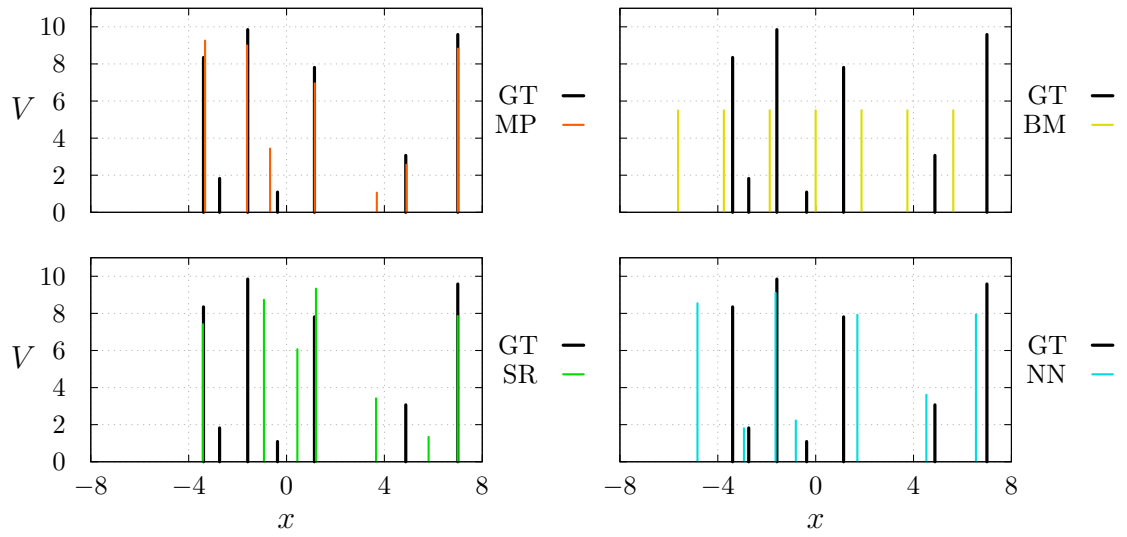

Figure 23: Sample from the test set, on which the model supposedly performs worst. "GT" is the ground truth, "MP" the model prediction, "BM" the benchmark, "SR" the search result and "NN" the nearest neighbour. The reason for the high ANN-cost is the fact that the model rearranges the peaks (e.g. the peak at $x \approx -1.6$ is the third in the ground truth, but the second in the model prediction).

Eq. (62) compares the respective first peaks, second peaks and so forth. This is however not the way we want to assess the model prediction in Fig. 23. We therefore introduce a modified cost function for the model evaluation: We construct a "distance matrix"

$$D_{ij} := \frac{1}{2}\left((x_i' - x_j)^2 + (V_i' - V_j)^2\right), \tag{63}$$

which is a measure for the distance between the j^{th} peak in the ground truth and the i^{th} peak in the model prediction. We search for the smallest element of this matrix, match the corresponding peaks and remove the associated row and column afterwards. This is iterated N times, such that there is a one-to-one matching between the peaks in the ground truth and the model prediction. The cost value is then calculated according to Eq. (62), but using the just described rearrangement. For the sample in Fig. 23 this yields an ANN-cost of 3.6 instead of 13.2 without the matching. This approach is however not used as the cost function in the first place, because it does not force the model to predict the peaks in the correct order and hence leaves an ambiguity, which makes it harder for the model to learn.

The computed average cost values are shown in Table 1. The ANN outperforms all the other methods. (Using the cost function in Eq. (62) – without the peak-matching

Table 1: Average performance (cost value, lower values are better) of the ANN, compared to the benchmark, the search procedure and the nearest neighbour analysis. The benchmark is the mean value of the training data set. The search / nearest neighbour procedure takes the sample from the training data set which lies nearest (w.r.t. the reflection amplitudes / potential parameters $(\mathbf{x}, \mathbf{V})$) to a given sample from the test data set. The training data set contains $5 \cdot 10^5$ samples and the test data consists of 10^4 samples. The best value is written in bold.

	ANN	Benchmark	Search	Nearest neighbour
Average cost value	**0.21**	3.9	5.2	0.35

outlined above – results in an average ANN-loss of 0.32.) It is surprising that the search procedure performs worst, even poorer than the benchmark.

Next, we want to see qualitatively how the model handles data it has not been trained for. We investigate four such cases: (1) the number of scatterers is reduced to $N = 6$, (2) the number of scatterers is increased to $N = 8$, (3) the minimal separation of scatterers is reduced to $\delta x = 0.1$, (4) the maximal delta-height is increased to $V_{\max} = 12$. The first two cases are interesting because the model is forced to return seven delta-peaks, but it receives data from a different number of delta-peaks. The last two cases test the generalizability of the model. In Fig. 24 we see three selected random examples for each of these cases, showing the typical behaviour of the model. The cases (1) and (2) reveal that the model "understands" something about the underlying physics. In the first two examples of (1) and in the first example of (2) we see that the model either fabricates or ignores small peaks in the middle of the potential, and weaker scatterers inbetween stronger ones indeed do not have as much impact on the scattering process as opposed to stronger scatterers. In the remaining examples of (1) and (2) we see that the model also tends to split a single scatterer into two smaller ones nearby or the other way around, which also makes sense physically. Case (3) demonstrates that the model cannot resolve scatterers, which are closer together than in the training data. Instead, it merges the two neighbouring peaks and "invents" a weak scatterer somewhere else. Case (4) shows that the model is hardly able to produce heights higher than the maximal value it was trained on (namely 10). Additionally, the model fails to reconstruct the scatterers near to the high peak(s).

Finally, we want to test the model's stability against noise. For this purpose we add Gaussian noise with a standard deviation of σ_{noise} to the input vectors of the test data set. In Fig. 25 we see that the performance of the model (measured by the enhanced cost function including the peak-matching) systematically gets worse with increasing

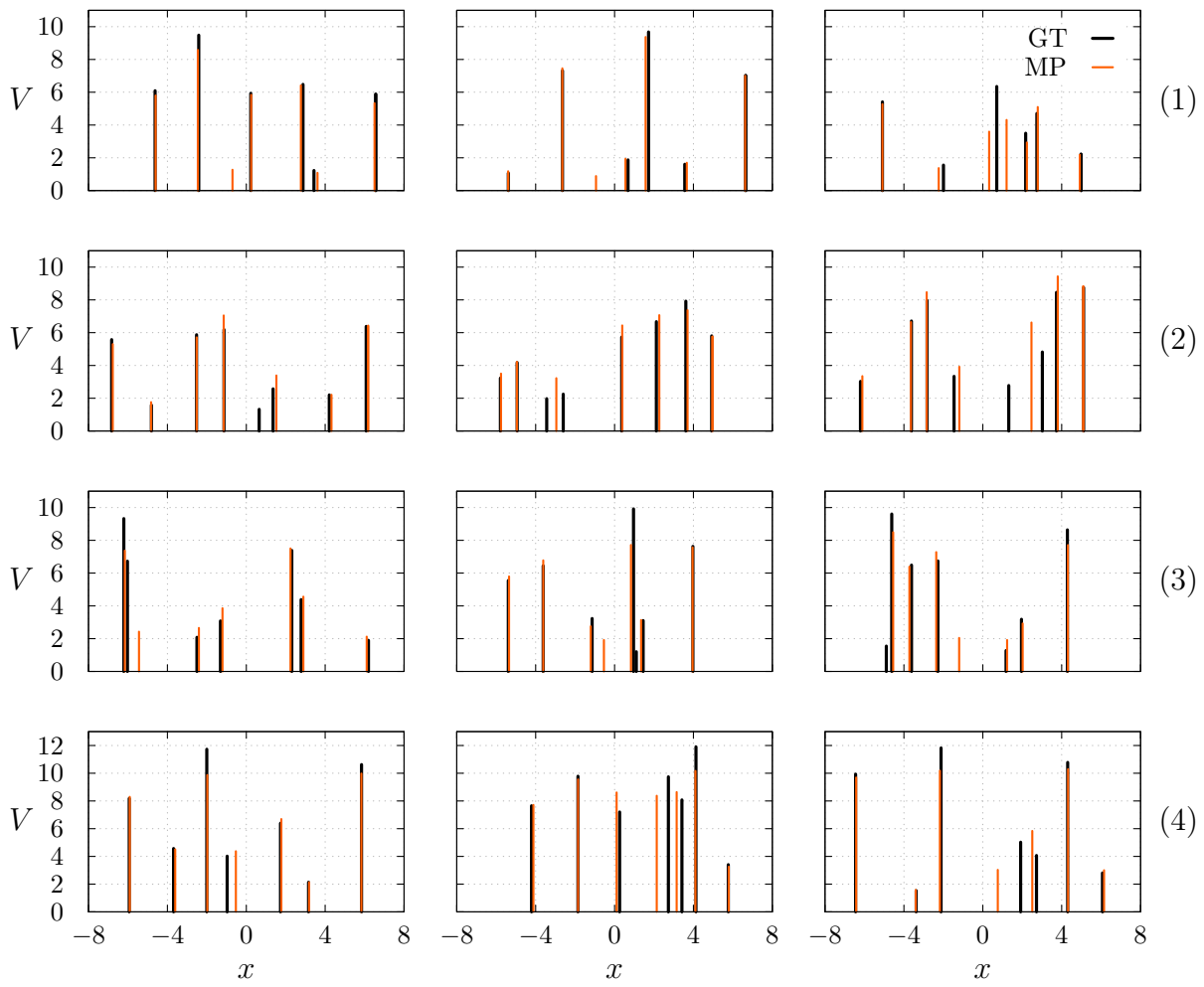

Figure 24: Ground truths (GT) and model predictions (MP) for configurations the model has not been trained for. The key in the top right plot holds for all other subplots as well. Each row shows three typical examples of a test case, indicated by the number on the right: (1) $N = 6$ scatterers, (2) $N = 8$ scatterers, (3) minimal separation $\delta x = 0.1$, (4) maximal height $V_{\max} = 12$.

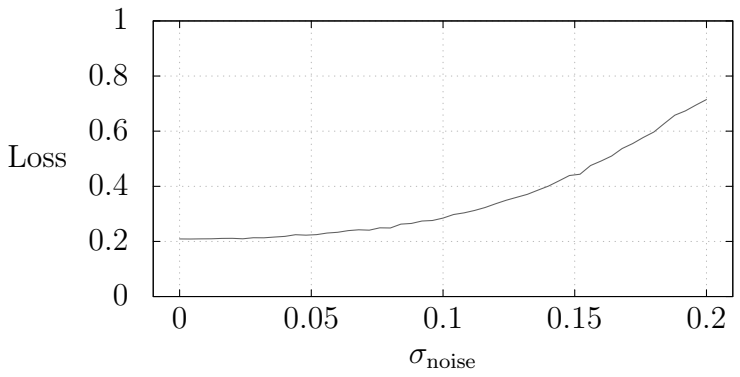

Figure 25: Performance of the model depending on the standard deviation σ_{noise} of the additive Gaussian noise applied to the input vectors. For each value of σ_{noise} the model was tested on 10^4 noisy samples.

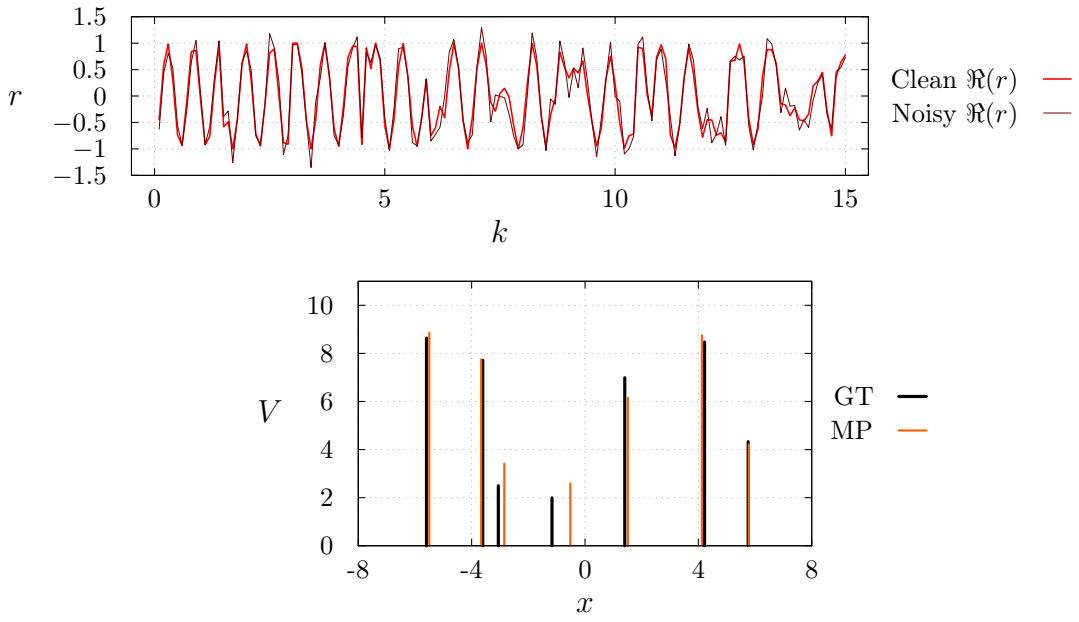

Figure 26: Test sample with $\sigma_{\text{noise}} = 0.2$. On the top a part of the clean input vector is compared to the noisy version. On the bottom the ground truth (GT) is compared to the model prediction (MP) based on the noisy signal.

σ_{noise}. Nevertheless, the quality of the model predictions stays relatively good, as can be seen for $\sigma_{\text{noise}} = 0.2$ in Fig. 26.
In summary, we observe that an ANN is capable of learning inverse scattering for a given class of delta-potentials. Whenever the model fails to reconstruct a scattering configuration, it does so in a physically plausible manner.

We now conclude the case of delta-potentials and proceed to compactly supported smooth potentials.

5.2 Compactly supported smooth potentials

A potential with a fixed number of delta-peaks is rather specific. We now consider the more general case of compactly supported smooth potentials. We treat the same two cases as in Subsec. 3.4, namely nonnegative,

$$\forall x \in \mathbb{R} : V(x) \geq 0, \tag{64}$$

and zero-mean potentials,

$$\int_{-\infty}^{\infty} V(x)\, dx = 0. \tag{65}$$

The former are convenient because they have no bound states and thus the solution of the corresponding inverse scattering problem (ISP) is unique (given only the reflection amplitudes, cf. Subsec. 3.1). The latter unfortunately do have at least one bound state [24] (cf. Subsec. 3.4). However, this shall not bother us, since the Artificial Neural Network (ANN) is fitted to a training data set containing samples from a prescribed class, denoted by the $\mathcal{S}$-symbol introduced in Subsec. B.2. This means that the ANN learns the statistics of $\mathcal{S}$ and is hopefully able to perform its task inside (and also – desirably – "a bit outside") of $\mathcal{S}$.
In Subsec. 3.4 we have seen that the Marčenko iteration does not yield satisfactory results for the investigated classes. We approach this problem with an ANN, which should learn the mapping from the scattering data (the reflection amplitudes) to the original potential. This time however, unlike in the case of delta-potentials in Subsec. 5.1, we do not take the raw scattering data as an input of the ANN. The first step we take is what we call a "backprojection", by which we mean a transformation of the scattering data, which lives in momentum space (meaning that the scattering matrix is a function of the momentum k), into real space. The result of this backprojection should

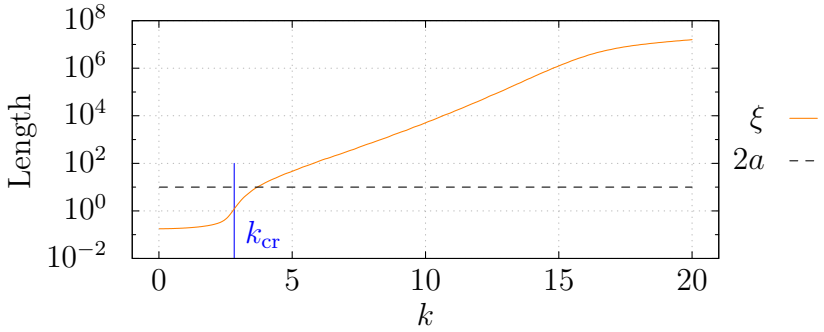

Figure 27: Momentum-dependent localization length $\xi(k)$ of the class $\mathcal{S}^{0.02,20,0.02}_{-256/51,5,5/255}(10, 0.1, +)$, based on 10^4 samples, compared to the length of the potentials $2a$. The blue line marks the critical momentum $k_{\mathrm{cr}} = \sqrt{2\langle V\rangle} = (2/\pi)^{1/4}\sqrt{\sigma_{\mathrm{pot}}} \approx 2.82$, below which ξ does not describe localization, but the exponential decay of the wavefunction inside the classically forbidden region (cf. Eq. (159)).

already resemble a more or less crude approximation of the scattering potential $V(x)$. We simply take the zeroth- and first-order Marčenko approximations (cf. Subsec. 3.3) as the backprojection method. The task of the ANN is to retrieve the original ground truth potential from the backprojection of the scattering data.

We can now state the main reason why we investigate zero-mean potentials, namely because the zeroth-order Marčenko approximation, which is now a part of the backprojection and thus a part of the input of the ANN, has the same property (cf. Eq. (39)) and we hope that this helps the ANN to learn the correct mapping.

Subsubsec. 5.2.1 is dedicated to the nonnegative potentials and Subsubsec. 5.2.2 to the zero-mean potentials.

5.2.1 Nonnegative potentials

We investigate the class $\mathcal{S}^{0.02,20,0.02}_{-256/51,5,5/255}(10, 0.1, +)$, which is essentially the same as the class $\mathcal{S}^{0.02,20,0.02}_{0,10,10/511}(10, 0.1, +)$ used in Subsec. 3.4, but with shifted support. In Subsec. 3.4 we saw that the Marčenko iteration does not yield satisfactory results for this class. We want to provide a solution to this problem with an ANN.

The localization length of the considered class is plotted in Fig. 27. We see that for $k \lesssim 2.5$ the wave gets exponentially suppressed due to penetration into the classically forbidden region. The localization regime is rather narrow, namely $3 \lesssim k \lesssim 3.5$. For $k \gtrsim 4$ the scattering is ballistic.

Regarding the ANN we choose a Convolutional Neural Network (CNN) architecture

with residual connections inspired by the so-called U-Net (cf. Subsubsecs. 4.4.1 and 4.4.3). The input of the network is not the raw scattering data like in the case of delta-potentials, but its backprojection into real space. An input tensor consists of four channels: the left and right zeroth- and first-order Marčenko approximations (cf. Eqs. (36), (37) and (42)),

$$V^{(0)}_{\text{Marč}}(x) = -\frac{2\hbar^2}{\pi m}\int_0^\infty k\,\Im\left(r(k)\,\mathrm{e}^{-2\mathrm{i}kx}\right)dk, \tag{66}$$

$$V'^{(0)}_{\text{Marč}}(x) = -\frac{2\hbar^2}{\pi m}\int_0^\infty k\,\Im\left(r'(k)\,\mathrm{e}^{2\mathrm{i}kx}\right)dk, \tag{67}$$

$$V^{(1)}_{\text{Marč}}(x) = V^{(0)}_{\text{Marč}}(x) + \frac{2\hbar^2}{\pi^2 m}\left(\int_0^\infty \Re\left(r(k)\,\mathrm{e}^{-2\mathrm{i}kx}\right)dk\right)^2, \tag{68}$$

$$V'^{(1)}_{\text{Marč}}(x) = V'^{(0)}_{\text{Marč}}(x) + \frac{2\hbar^2}{\pi^2 m}\left(\int_0^\infty \Re\left(r'(k)\,\mathrm{e}^{2\mathrm{i}kx}\right)dk\right)^2, \tag{69}$$

computed on the discrete space lattice. (The integrals are cut off at $k_{\min}$ and $k_{\max}$.) This way, and not by passing the weighted means $W^{(0)}_{\text{Marč}}$ and $W^{(1)}_{\text{Marč}}$ defined in Eq. (43) to the CNN, we let the CNN itself decide which information to take.

The desired output of the network is the original potential V on the discrete space lattice. Therefore the ANN maps from position space (as opposed to momentum space) to position space, which is the main reason why we use an encoder-decoder CNN architecture.

Both the input and output values do not require to be standardized.

Let $\mathbf{V} = (V(x_{\min}), \ldots, V(x_{\max}))$ be some potential (the ground truth, i.e. the desired output vector) and $\mathbf{V}' = (V'(x_{\min}), \ldots, V'(x_{\max}))$ the corresponding prediction of the model (based on the backprojection of the scattering data of $\mathbf{V}$). We define the mean squared error

$$C(\mathbf{V}', \mathbf{V}) = \frac{1}{n_x}\|\mathbf{V}' - \mathbf{V}\|_2^2 = \frac{1}{n_x}\sum_i \left(V'(x_i) - V(x_i)\right)^2 \tag{70}$$

as the cost function. $n_x = 2I = 512 = 2^9$ denotes the number of spatial points.

The number n_x is chosen such that it can be evenly divided in half up to nine times. This allows us to do pooling and upsampling (by a factor of 2) in the CNN without having to bother about "leftover-pixels".

The architecture of the used CNN is schematically displayed in Fig. 28 and described in

the associated caption. The number of trainable parameters amounts to $d = 15\,516\,525$. Due to memory issues the training data was produced with a Python generator function. This means that one does not have to provide the entire training data set before and during the fitting process; a generator function produces training samples on demand (before each mini-batch gradient step), which are deleted from memory afterwards. The fitting process is done in 200 epochs and each epoch consists of 156 mini-batch gradient steps, each based on a mini-batch with 64 samples. In total, the CNN is trained on nearly $2 \cdot 10^6$ samples, but it "sees" every sample only once. Validation is done on a fixed validation data set of size 10^3 at the end of each epoch. As an optimizer we choose Adam with a learning rate of 10^{-4}. The learning curves (the training and validation loss at the end of each epoch) of the model are shown in Fig. 29.

The performance of the trained model can be seen qualitatively with six test samples in Fig. 30. Though it is not perfect, when compared to Fig. 10, we see that the CNN certainly performs much better than the Marčenko approximation.

For a quantitative analysis, we compare the model against a benchmark and the 50^{th} order weighted mean Marčenko approximation $W^{(50)}_{\text{Marč}}$ (cf. Eq. (43)), for which we saw in 3.4 that it is a good compromise between the number of iterations and the accuracy around the region of convergence (cf. Fig. 10). A search procedure and a nearest neighbour analysis, like with the delta-potentials, is omitted, because the entire training data, to which the model was fitted, was not stored.

Given a ground truth potential $\mathbf{V}$ from a test data set, we calculate its reflection amplitudes. They are used for the backprojection, which is fed into the CNN, producing some prediction $\mathbf{V}'$, as well as for the Marčenko iteration, yielding after fifty iterations the weighted mean $\mathbf{W}^{(50)}_{\text{Marč}}$. The benchmark $\overline{\mathbf{V}}$ is simply the average of 10^4 potentials randomly drawn from the class $\mathcal{S}^{0.02,20,0.02}_{-256/51,5,5/255}\,(10, 0.1, +)$. The quality of $\overline{\mathbf{V}}$, $\mathbf{V}'$ and $\mathbf{W}^{(50)}_{\text{Marč}}$ w.r.t. the ground truth $\mathbf{V}$ is measured with the cost function in Eq. (70). We also record the computation time for the model prediction (including the backprojection) and the 50^{th} order weighted mean Marčenko approximation. Table 2 displays the cost function values averaged over a test data set of size 10^4 (processed in parallel) and the computation times averaged over 100 sequentially processed samples. The CNN significantly outperforms the other methods regarding the cost value. The Marčenko approximation is even far worse than the benchmark. Of course, the computation time heavily depends on the used machine. Still, the CNN is an order of magnitude faster than fifty Marčenko iterations. In Fig. 31 we plot the ground truth of a randomly chosen sample from the test set and compare it to the model prediction, the benchmark, and

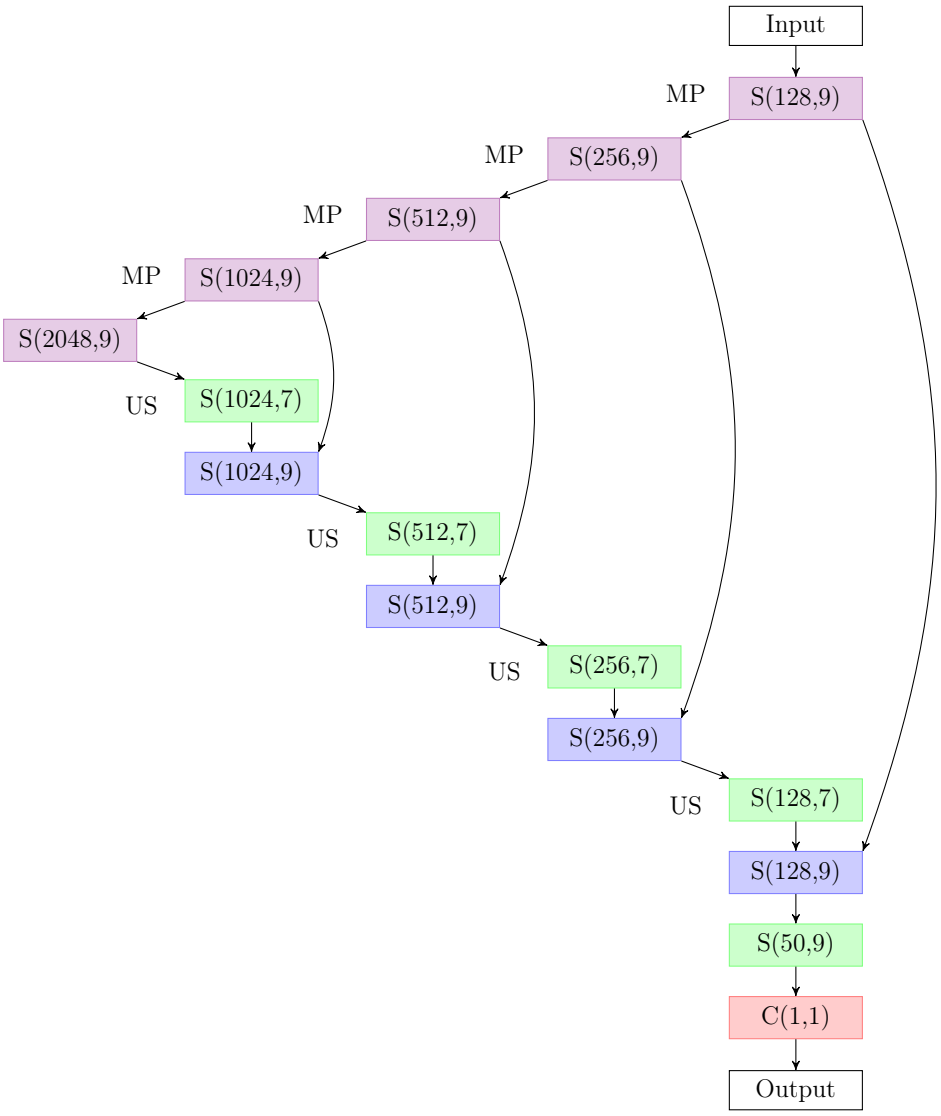

Figure 28: Schematic visualization of the used CNN architecture for nonnegative potentials: Violet or blue block = two 1D depthwise separable convolutional layers, green block = one 1D depthwise separable convolutional layer, red block = one 1D convolutional layer. The two numbers in each block give the number of output channels and the kernel size, respectively. The activation function is always ReLU, except for the red block, where a linear activation is used. The initializers are "He" and "Glorot", respectively. In each convolutional operation the input is padded in such a way that the output has the same size as the input. After each activation function there is a batch normalization layer, except for the red block. "MP" denotes a 1D max pooling layer, which halves the size of the array. "US" denotes a 1D upsampling layer, which doubles the size of the array. The input of each blue block is the concatenation of the outputs of the respectively indicated green and violet block (residual connections). The CNN has $d = 15\,516\,525$ trainable parameters.

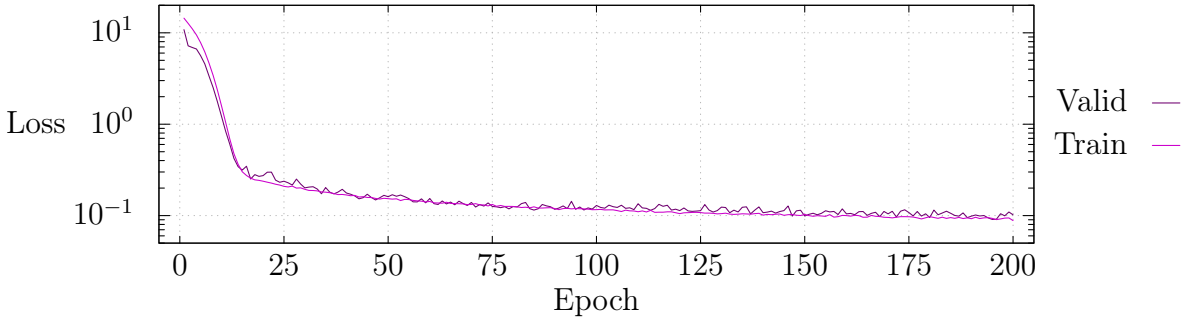

Figure 29: Learning curves of the model used for the inverse scattering of compactly supported smooth nonnegative potentials. "Train" is the training loss – the average cost function evaluated on the training data. "Valid" is the validation loss – the average cost function evaluated on the validation data, to which the model is not fitted.

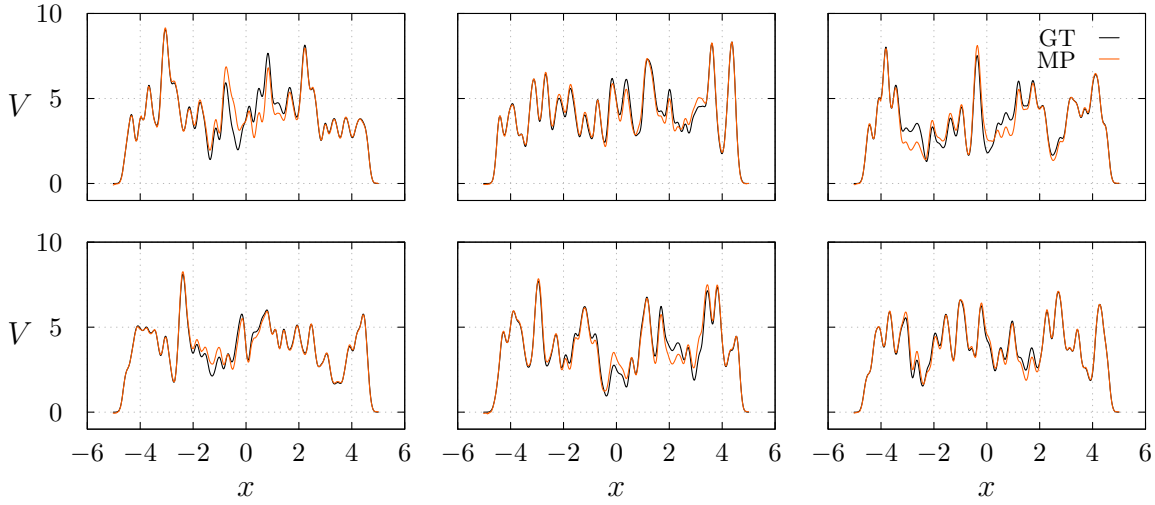

Figure 30: Six randomly generated test examples of the class $\mathcal{S}^{0.02,20,0.02}_{-256/51,5,5/255}(10, 0.1, +)$. From the ground truths "GT" the reflection amplitudes are calculated, backprojected according to Eqs. (66) – (69), and then fed into the model, which returns the predictions "MP". The key, which is displayed only in the top right subplot, holds for every other subplot as well.

Table 2: Quantitative comparison of the model (CNN) to the benchmark (mean potential in the considered class) and to the 50^{th} order weighted mean Marčenko approximation $W^{(50)}_{\text{Marč}}$. Averaging is done over a test data set consisting of 10^4 / 100 samples for the cost value / computation time. The best values are written in bold.

	CNN	Benchmark	Marčenko approx.
Average cost value	$\mathbf{9.3 \cdot 10^{-2}}$	1.7	16
Average computation time	$\mathbf{1.3 \cdot 10^{-1}\,s}$	—	1.6 s

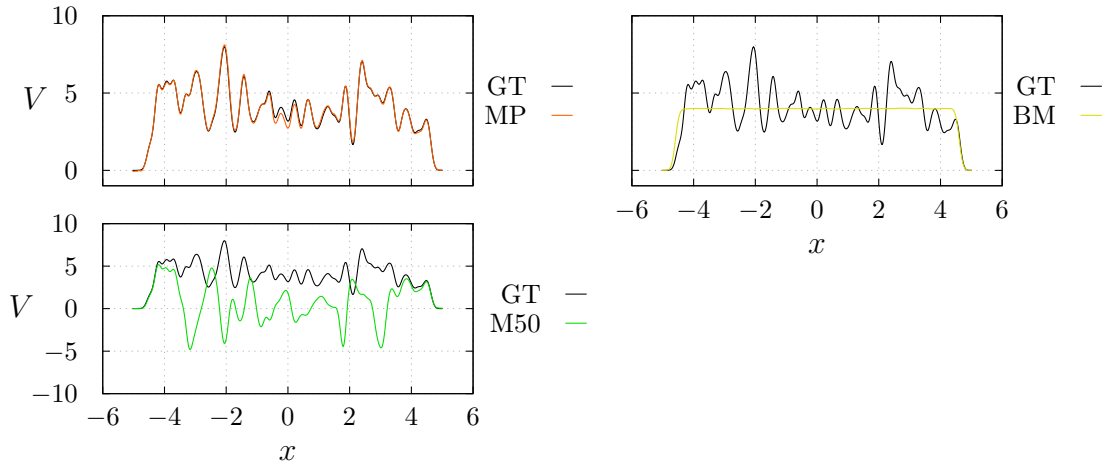

Figure 31: Random sample from the test set. "GT" is the ground truth, "MP" the model prediction, "BM" the benchmark and "M50" the 50^{th} order weighted mean Marčenko approximation.

the 50^{th} order weighted mean Marčenko approximation. The performance difference, already recognizable by the numbers in Table 2, is evident.

Next, we investigate qualitatively the generalizability of the model, i.e. how it copes with samples which lie outside the class it was trained for. We still demand that all considered potentials are nonnegative. We examine six such cases: (1) a smoothened rectangular potential, (2) a triangular potential, (3) a smoothened delta-potential, (4) the standard deviation of the potential values is increased to $\sigma_{\text{pot}} = 15$, (5) the autocorrelation of the potentials is reduced to $\sigma_{\text{ker}} = 0.07$, (6) the autocorrelation of the potentials is increased to $\sigma_{\text{ker}} = 0.2$. The smoothing in (1) and (3) is done with the same Gaussian kernel used in the generation of the training data (i.e. having a standard deviation of $\sigma_{\text{ker}} = 0.1$). The rectangular and the triangular potential both are centered at $x = 0$, have a width of $a = 5$ and a height of $\langle V \rangle = \frac{\sigma_{\text{pot}}}{\sqrt{2\pi}} \approx 4$. The delta-potential (3) is initialized with three delta peaks: two larger peaks and a smaller peak inbetween. Fig. 32 shows the three examples (1), (2), (3) and three random samples for each of the cases (4), (5), (6). The model fails severely with the first three test cases (1-3). Only the edges of the smoothened rectangular potential are captured to a certain extent. The higher potentials (4) are reconstructed quite well on the far left and on the far right, but not in the middle section, where the model underestimates the potential values, but grasps the rough course of the potential. For the less correlated potentials (5) the model can predict not more than the margins on the left and on the right, if at all. The more correlated potentials (6) are reconstructed rather well, albeit not perfectly; also

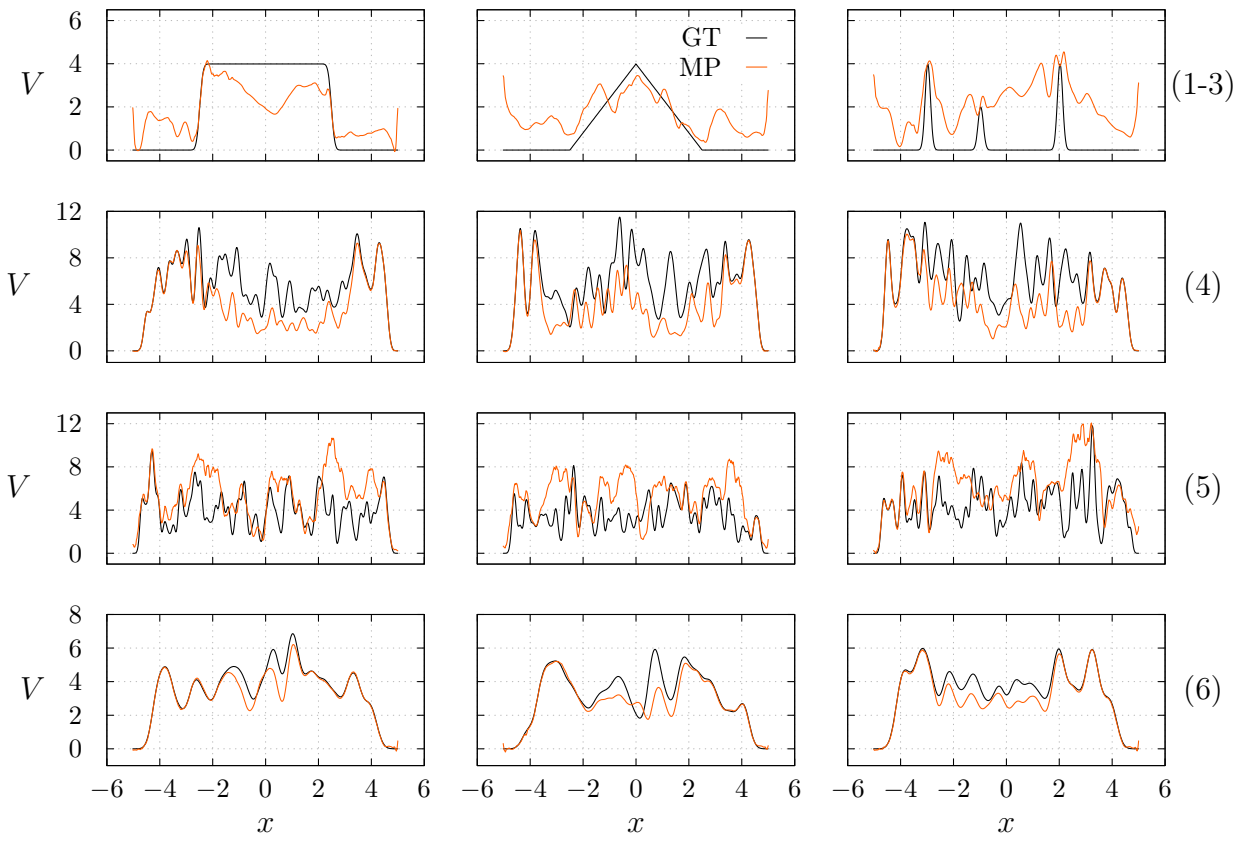

Figure 32: Ground truths (GT) and model predictions (MP) for configurations the model has not been trained for. The key in the top middle plot holds for all other subplots as well. The first row shows three special potentials (from left to right): (1) smoothened rectangular potential of width $a = 5$ and height $\langle V \rangle \approx 4$, (2) triangular potential of width $a = 5$ and height $\langle V \rangle \approx 4$, (3) three smoothened delta-peaks. The other rows show three random examples of the other test cases, indicated by the number on the right: (4) higher standard deviation of potential values $\sigma_{\text{pot}} = 15$, (5) smaller potential autocorrelation $\sigma_{\text{ker}} = 0.07$, (6) larger potential autocorrelation $\sigma_{\text{ker}} = 0.2$. Mind the different scales of the vertical axes.

notice the fluctuations in the model predictions on the left and on the right boundary. From these observations we can conclude that the model prediction is not reliable as soon as one takes a sample which lies outside of the class of potentials, on which the model was trained.
To end this subsubsection, we inspect the model's stability against noise. We introduce multiplicative complex-valued Gaussian noise with a standard deviation of σ_{noise} (around the value 1) onto the reflection amplitudes before doing the backprojection and feeding the result into the model. In Fig. 33 we see that the performance diminishes with increasing σ_{noise}, but the example shown in Fig. 34 reveals that the model is rather robust against noise.
To summarize, the CNN can perform inverse scattering reasonably well for the class it was trained on. It surpasses the Marčenko iteration procedure by far and is resistant against corruption by noise. However, as soon as the potential does not show the correct statistics, i.e. if it lies outside the class $\mathcal{S}^{0.02,20,0.02}_{-256/51,5,5/255}(10, 0.1, +)$, the performance of the CNN rapidly drops.

In the following subsubsection we investigate if a CNN is also able to do inverse scattering for zero-mean potentials.

5.2.2 Zero-mean potentials

For the zero-mean potentials we basically apply the same methods and do the same analysis as with the nonnegative potentials. Therefore this subsubsection is not so extensive in text, though equally comprehensive in content.
We choose the class $\mathcal{S}^{0.02,20,0.02}_{-256/51,5,5/255}(10, 0.1, 0)$, for which we saw in Subsec. 3.4 that the Marčenko iteration is not applicable. The localization length is shown in Fig. 35. The scattering is largely ballistic (for $k \gtrsim 5$).
The chosen CNN architecture is depicted in Fig. 36. The input and the output of the CNN, the cost function (cf. Eq. 70), the handling of the training and validation data as well as the fitting process are chosen just like in Subsubsec. 5.2.1. Only this time we find that the training works best using a learning rate of $2 \cdot 10^{-4}$. The learning curves of the model are plotted in Fig. 37.
From the six test samples in Fig. 38 we see qualitatively that the performance is not as good as the CNN trained for nonnegative potentials (cf. Fig. 30). This can be traced back to the fact that zero-mean potentials possess bound states and therefore

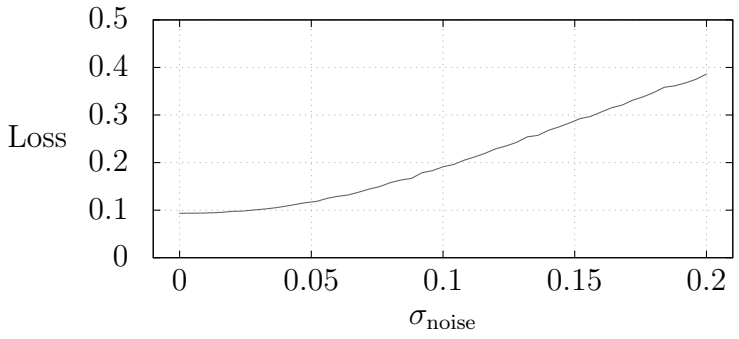

Figure 33: Performance of the model depending on the standard deviation σ_{noise} of the multiplicative Gaussian noise applied to the reflection amplitudes before doing the backprojection. For each value of σ_{noise} the model was tested on 10^4 noisy samples.

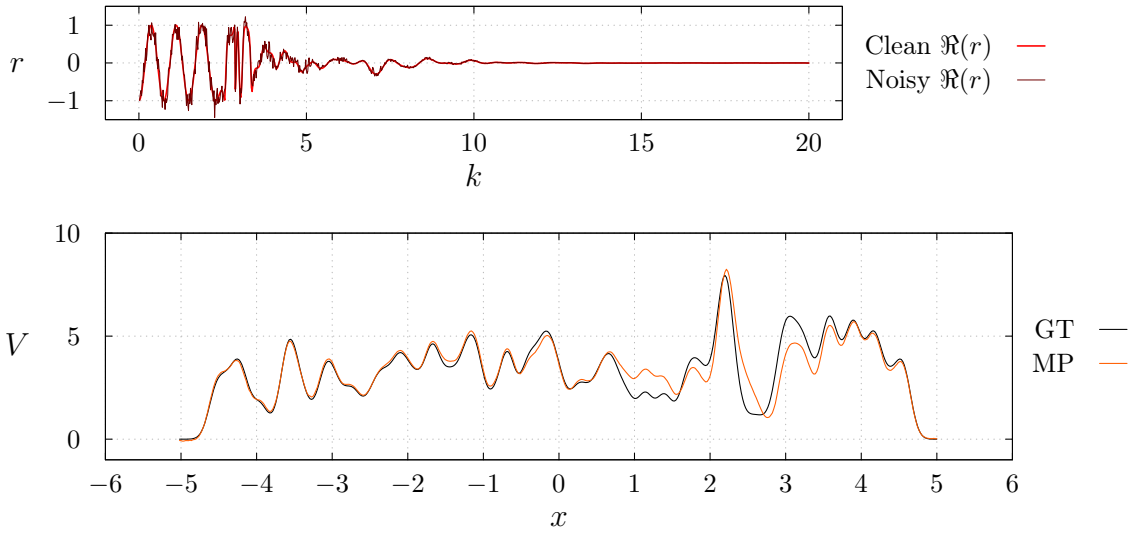

Figure 34: Test sample with $\sigma_{\text{noise}} = 0.2$. On the top the real part of the clean reflection amplitude is compared to the noisy version. On the bottom the ground truth (GT) is compared to the model prediction (MP) based on the noisy signal.

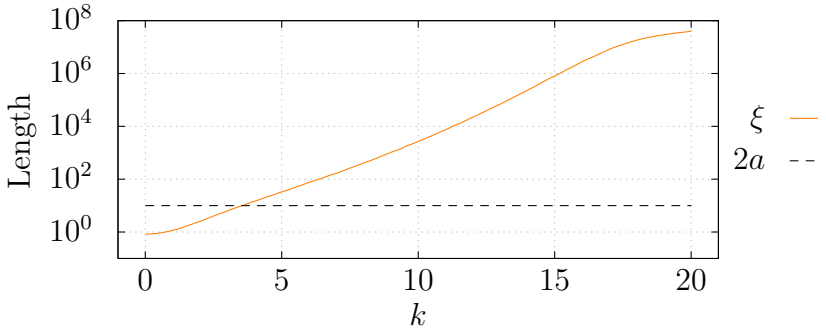

Figure 35: Momentum-dependent localization length $\xi(k)$ of the class $\mathcal{S}^{0.02,20,0.02}_{-256/51,5,5/255}(10, 0.1, 0)$, based on 10^4 samples, compared to the potential length $2a$.

Table 3: Quantitative comparison of the model (CNN) to the benchmark (mean potential in the considered class) and to the 0^{th} order weighted mean Marčenko approximation $W^{(0)}_{\text{Marč}}$. Averaging is done over a test data set consisting of 10^4 samples. The best value is written in bold.

	CNN	Benchmark	Marčenko approx.
Average cost value	**1.4**	4.9	3.5

the inverse scattering problem does not have a unique solution when one is provided with the reflection amplitudes only. Still, the model predictions are better than the Marčenko approximations (cf. Fig. 13).

We want to quantify this statement in the following analysis. Like with the nonnegative potentials, we compare the model to a benchmark and to the zeroth-order weighted mean Marčenko approximation $W^{(0)}_{\text{Marč}}$. The benchmark is the mean potential in the class $\mathcal{S}^{0.02,20,0.02}_{-256/51,5,5/255}(10, 0.1, 0)$, namely $V = 0$. For the Marčenko approximation we choose the zeroth order because there are instances where this is indeed the best approximation. The average cost values of all three methods are listed in Table 3. There is no need for a speed analysis, since the zeroth-order Marčenko approximation is part of the preprocessing of the model input and hence will always be faster than the model prediction, which in turn takes approximately as much time as in the case of nonnegative potentials. This time (as opposed to the nonnegative potentials) the Marčenko approximation is better than the simple benchmark. The CNN however outperforms both of them. In Fig. 39 we directly compare the three methods based on a single randomly chosen sample. Though the model prediction is not flawless, it certainly surpasses the Marčenko approximation.

The generalizability of the model is qualitatively investigated on six test cases: (1) a

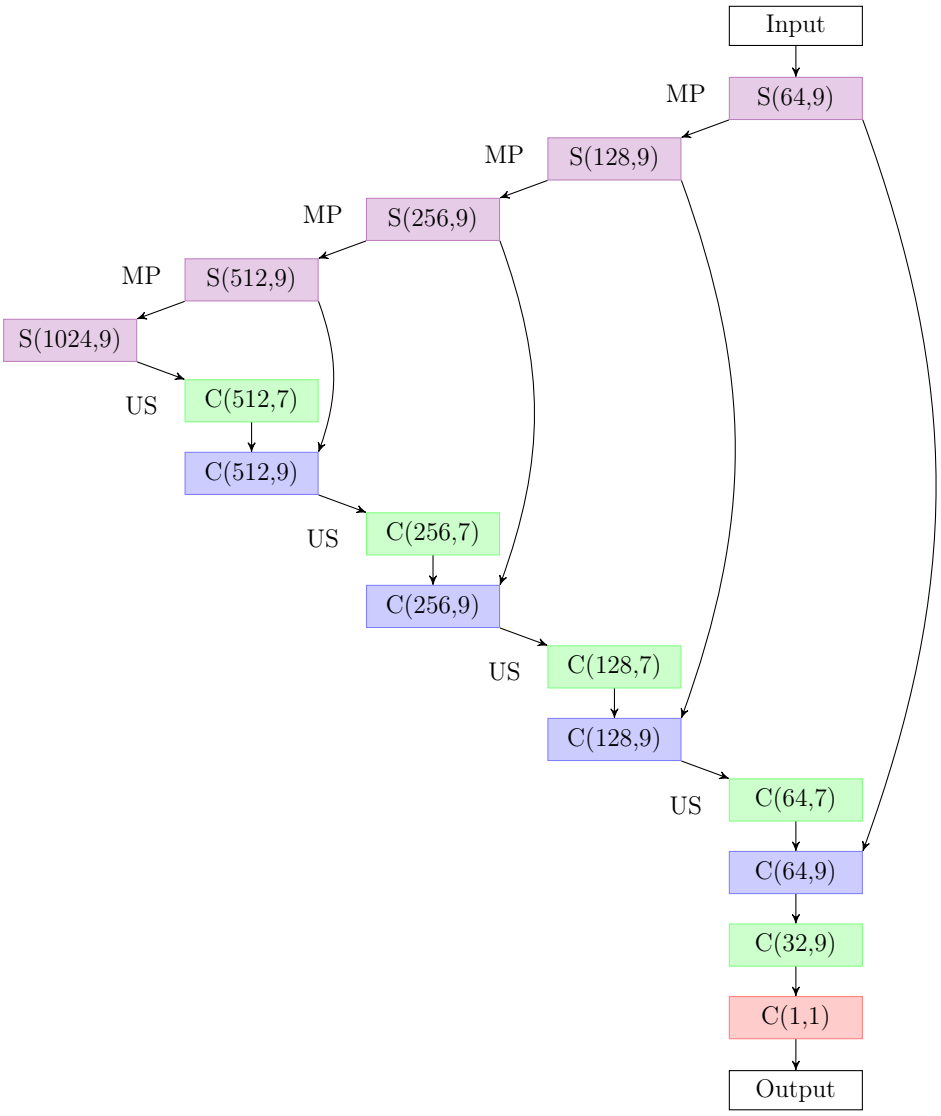

Figure 36: Schematic visualization of the used CNN architecture for zero-mean potentials: Violet block = two 1D depthwise separable convolutional layers, green block = one 1D convolutional layer, blue block = two 1D convolutional layers, red block = one 1D convolutional layer. The two numbers in each block give the number of output channels and the kernel size, respectively. The activation function is always ReLU, except for the red block, where a linear activation is used. The initializers are "He" and "Glorot", respectively. In each convolutional operation the input is padded in such a way that the output has the same size as the input. After each activation function there is a batch normalization layer, except for the red block. "MP" denotes a 1D max pooling layer, which halves the size of the array. "US" denotes a 1D upsampling layer, which doubles the size of the array. The input of each blue block is the concatenation of the outputs of the respectively indicated green and violet block (residual connections). The CNN has $d = 16\,433\,509$ trainable parameters.

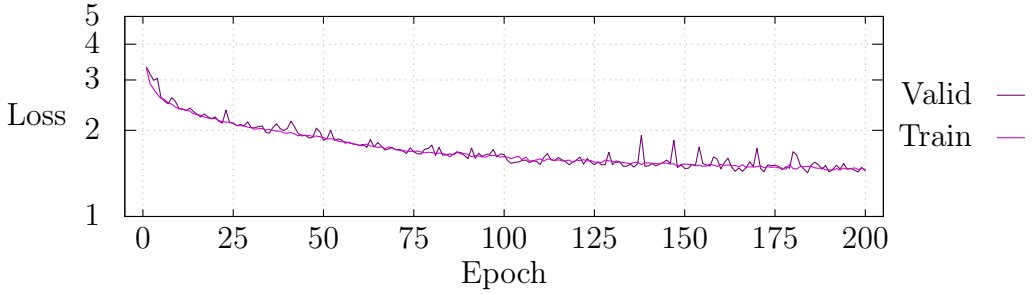

Figure 37: Learning curves of the model used for the inverse scattering of compactly supported smooth zero-mean potentials. "Train" is the training loss – the average cost function evaluated on the training data. "Valid" is the validation loss – the average cost function evaluated on the validation data, to which the model is not fitted.

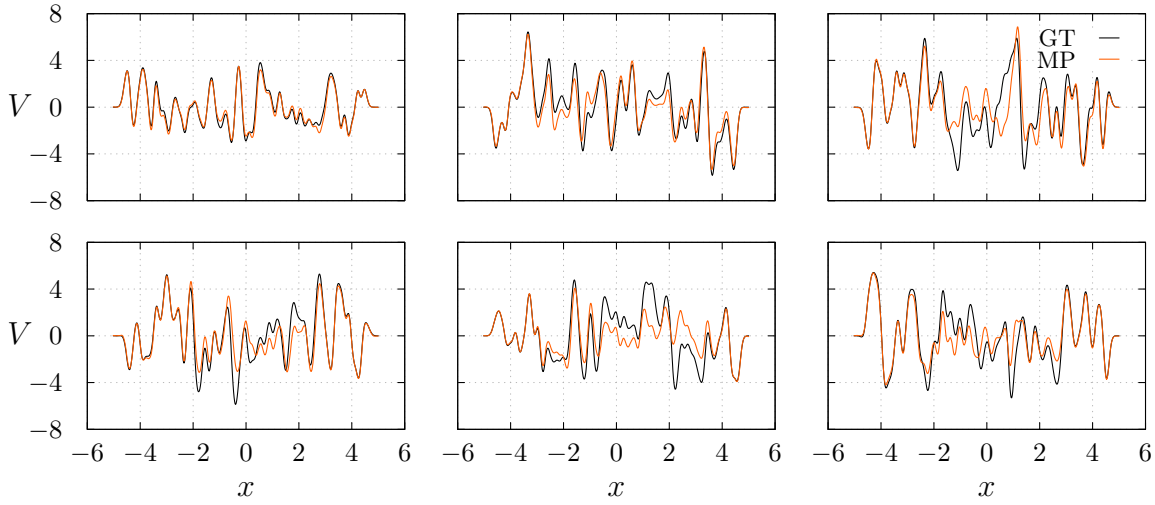

Figure 38: Six randomly generated test examples of the class $\mathcal{S}^{0.02,20,0.02}_{-256/51,5,5/255}(10, 0.1, 0)$. From the ground truths "GT" the reflection amplitudes are calculated, backprojected according to Eqs. (66) – (69), and then fed into the model, which returns the predictions "MP". The key, which is displayed only in the top right subplot, holds for every other subplot as well.

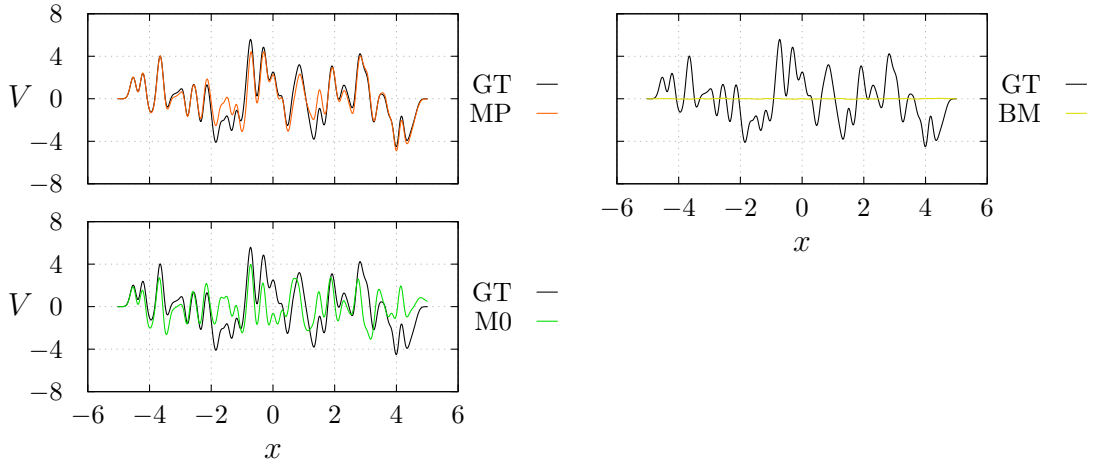

Figure 39: Random sample from the test set. "GT" is the ground truth, "MP" the model prediction, "BM" the benchmark and "M0" the zeroth-order weighted mean Marčenko approximation.

zero-mean smoothened rectangular potential, (2) a zero-mean triangular potential, (3) a zero-mean smoothened delta-potential, (4) the standard deviation of the potential values is increased to $\sigma_{\text{pot}} = 15$, (5) the autocorrelation of the potentials is reduced to $\sigma_{\text{ker}} = 0.07$, (6) the autocorrelation of the potentials is increased to $\sigma_{\text{ker}} = 0.2$. In the cases (1-3) the generated potentials are mapped to zero-mean versions according to Eq. (74). The smoothing in (1) and (3) is done with the same Gaussian kernel used in the generation of the training data (i.e. having a standard deviation of $\sigma_{\text{ker}} = 0.1$). The rectangular and the triangular potential both are centered at $x = 0$, have a width of $a = 5$ and a height of $\sigma_{\text{pot}} = 10$. The delta-potential (3) is initialized with three delta peaks: two larger peaks and a smaller peak inbetween. Fig. 40 shows the three examples (1), (2), (3) and three random samples for each of the cases (4), (5), (6). The model is at least qualitatively able to somehow recognize the edges of the rectangular potential (1) and identify the peak in the middle of the triangular potential (2). The delta-potential (3) on the other hand is reconstructed surprisingly well. The model prediction for the higher/deeper potentials (4) is mediocre. The CNN completely fails at the less correlated potentials (5), but the more correlated potentials (6) are reconstructed correctly in part. From these observations we conclude that the CNN does not perform reliably on samples which lie outside of the class of potentials which it was trained for. We continue with the investigation of the models stability against multiplicative Gaussian noise, just like with the nonnegative potentials. Fig. 41 shows the decay of the model performance as the noise gets stronger.

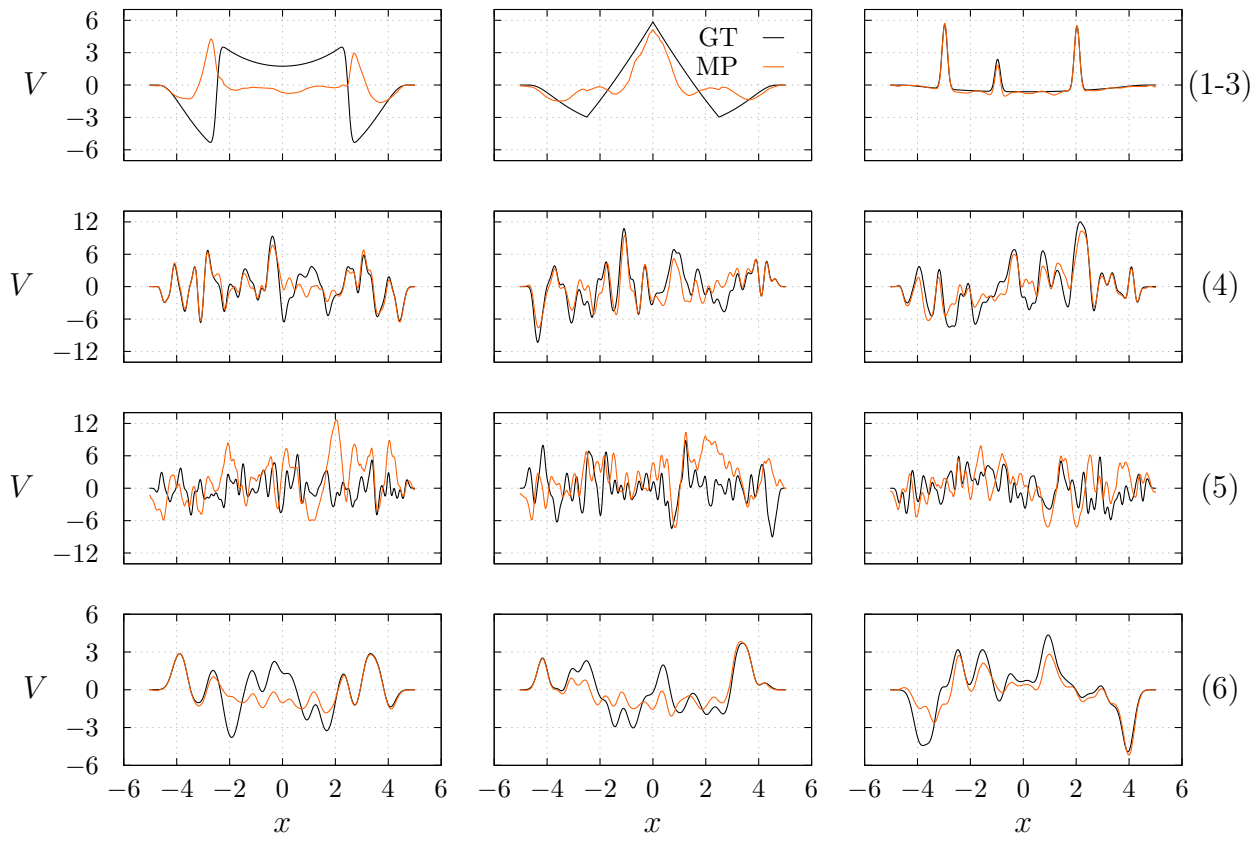

Figure 40: Ground truths (GT) and model predictions (MP) for configurations the model has not been trained for. The key in the top middle plot holds for all other subplots as well. The first row shows three special potentials (from left to right): (1) zero-mean smoothened rectangular potential of width $a = 5$ and height $\sigma_{\text{pot}} = 10$, (2) zero-mean triangular potential of width $a = 5$ and height $\sigma_{\text{pot}} = 10$, (3) three smoothened delta-peaks mapped to a zero-mean version. The other rows show three random examples of the other test cases, indicated by the number on the right: (4) higher standard deviation of potential values $\sigma_{\text{pot}} = 15$, (5) smaller potential autocorrelation $\sigma_{\text{ker}} = 0.07$, (6) larger potential autocorrelation $\sigma_{\text{ker}} = 0.2$. Mind the different scales of the vertical axes.

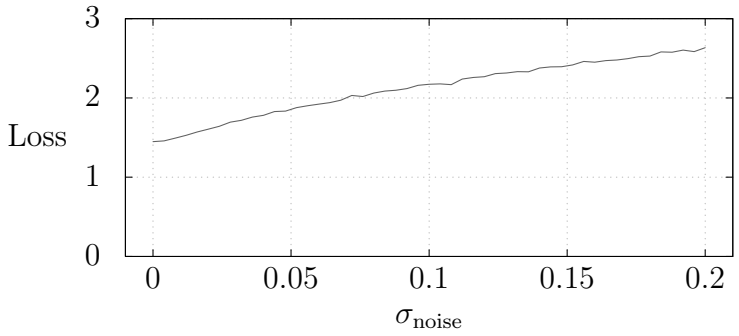

Figure 41: Performance of the model depending on the standard deviation σ_{noise} of the multiplicative Gaussian noise applied to the reflection amplitudes before doing the backprojection. For each value of σ_{noise} the model was tested on 10^4 noisy samples.

In conclusion, the quality of the CNN predictions is far from perfect, but it is still better than the Marčenko approximation. Furthermore, the performance quickly drops as soon as one tests the model on data outside of the class it was trained on.

6 Summary and Outlook

We investigated the inverse Schrödinger scattering problem for one-dimensional Hermitian potentials with compact support. The solution to this problem is unique if one is provided with the reflection amplitude (from the left or from the right), the eigenenergies of the bound states and the normalization constants of the bound states. While the solution can, in principle, be obtained by solving the Marčenko integral equation, this is analytically only possible in very few special cases. Also an iterative approach does not necessarily yield good results. Furthermore, one often does not have access to the entire information about the bound states.
We approach this problem by training Artificial Neural Networks (ANNs) to do the inverse scattering. Each ANN is trained on a different statistical class of potentials. One could ask the question why we do not intend to find an ANN capable of doing inverse scattering for *any* potential. On the one hand, this is simply not possible due to the ill-posedness of the inverse scattering problem in the presence of bound states (if one only knows the scattering matrix of the system). On the other hand, in most cases one knows about the statistical properties of the investigated systems (e.g. biological tissues). This knowledge is an advantage for training ANNs, since it restricts the hypothesis space and therefore the learning process is more efficient.
For a fixed number of pointlike scatterers the Deep Learning approach we put forward here works reasonably well. Also for nonnegative potentials, which do not support bound states, the performance of the ANN is satisfactory, as long as one stays within the class on which the ANN was trained. Zero-mean potentials possess bound states, which renders the inverse scattering problem ill-posed. This translates to a reduced quality of the ANN performance, which we find nonetheless to be superior to the iterated approximate solutions of the Marčenko integral equation.

In future work one could investigate the following scenarios:
One could search for ANN architectures which are suited for potentials with a variable number of delta-peaks. Such potentials cannot be treated with a rigid architecture like the ones used in this thesis.
Bound states, as we have seen, are an ever-present issue in the inverse scattering problem. One could investigate the performance of an ANN trained on a specific class of nonpositive potentials; or figure out a way to include the eigenenergies and the normalization constants of the bound states into the input-architecture of the ANN, such that

the ANN is able to employ this data for the inversion procedure.
One could test other architectures, like bidirectional Recurrent Neural Networks or Convolutional Neural Networks with locality but without shared weights (since the problem at hand is not really translationally invariant). Such a layer is implemented as `keras.layers.LocallyConnected1D` in Keras.
Localizing potentials, i.e. where $\xi(k_{\max})$ is considerably smaller than the potential length, are far more challenging than the smooth potentials considered in this thesis. An interesting aspect would be the quality of the ANN predictions as a function of $k_{\max}$.
One could try to train an ANN on potentials with a range of statistical parameters (like σ_{ker} and σ_{pot}) instead of fixed values.
It is worthwhile trying to apply the proposed method to other physical systems like nonhermitian (i.e. complex-valued) potentials with loss and gain, quasi-1D and higher dimensional systems, or other wave equations like the Helmholtz equation describing electromagnetic scattering or the acoustic wave equation. Eventually such Machine Learning algorithms could help to improve state-of-the-art imaging techniques as used for applications in medicine, geophysics and material science.

7 Acknowledgements

First and foremost, I want to thank my supervisor, Stefan Rotter, for giving me the opportunity to delve into the exciting and cutting-edge field of Deep Learning, and for his time he dedicated to fruitful discussions, which always resulted in valuable input for my work.
I want to thank my colleagues for providing a comfortable yet supportive working atmosphere. Special thanks to Andre Brandstötter for his helpful input.
Last, but not least, I want to thank my family, especially my parents, and my friends, who always supported me in all sorts of ways.

A Technical specifications

All computations were carried out on a machine with the following core components:

- CPU: Intel Xeon E5-1620 v4.
- GPU: NVIDIA GeForce GTX 1080 Ti (one unit).
- RAM: 32GB.

All models were implemented in and trained using Keras 2.2.2 (written in Python) with backend TensorFlow 1.8.0 or Theano 1.0.1 (sometimes the former is faster, sometimes the latter). The functions generating the training data (i.e. solving the forward scattering problems and doing the backprojection) were parallelized using Numba 0.39.0.

B Random potential generation

In order not to always repeat the explanation how we generate random potentials, we introduce a shorthand notation for all the relevant physical parameters involved in a certain setup.

B.1 Delta-potentials

For a potential consisting of delta-peaks, we use the following parameters:

- N = number of delta-peaks.
- $[x_{\min}, x_{\max}]$ = finite space region, in which the N delta-peaks lie. Two consecutive delta-peaks must have a minimum separation of δx. If two delta-peaks could get arbitrarily close together, one would not be able to resolve them.
- $[V_{\min}, V_{\max}]$ = interval, from which the strengths of the delta-peaks are randomly drawn.
- Momentum values $k_j = k_{\min} + j\Delta k$, where $j \in \{0, \ldots, n_k - 1\}$ and $k_{\max} = k_{n_k - 1}$ $\implies$ $n_k = \frac{k_{\max} - k_{\min}}{\Delta k} + 1$.

For such a setup we write the symbol $\mathcal{D}^{k_{\min}, k_{\max}, \Delta k}_{x_{\min}, x_{\max}, \delta x}(N; V_{\min}, V_{\max})$.

B.2 Compactly supported smooth potentials

A compactly supported smooth random potential is generated using the following scheme:

1. Space discretization: $x_i = i\Delta x$. We distinguish three kinds:
   (a) First kind: $i \in \{-I, \ldots, I\}$. $x_{\max} = -x_{\min} = I\Delta x$. This is the usual setup with $2I + 1$ space points. The support of the potential is a subset of $[-a, a]$ with $a = I\Delta x$.
   (b) Second kind: $i \in \{-I, \ldots, I - 1\}$. $x_{\max} + \Delta x = -x_{\min} = I\Delta x$. This setup is used whenever CNNs are involved because then the number of space points is even ($2I$) and this is a desirable feature for pooling (where the number of points is halved, cf. Subsubsec. 4.4.1). The support is a subset of $[-a, a]$ with $a = (I - 1)\,\Delta x$, i.e. $x_{-I} = -I\Delta x = x_{\min}$ is an auxiliary point.

(c) Third kind: $i \in \{0, \ldots, I-1\}$, i.e. only the positive x-axis is considered. $x_{\max} = (I-1)\,\Delta x$. This is choice is convenient for the iterative Marčenko approximation (cf. Subsubsec. 3.3).

2. The potential values $V_{\text{raw}}(x_i)$ are initialized to normally distributed values with a standard deviation σ_{pot} for $x_i \in [-b, b]$ and set to zero for $x_i \notin [-b, b]$. The next bullet point clarifies what b is. If we want to get a strictly nonnegative potential, then all negative $V_{\text{raw}}(x_i)$ are set to zero.

3. The uncorrelated potential V_{raw} is smoothened by convolution with a Gaussian kernel of width (standard deviation) σ_{ker}. This convolution causes the potential to become nonzero outside of $[-b, b]$. The support of the potential grows from $[-b, b]$ to approximately $[-b - 4\sigma_{\text{ker}}, b + 4\sigma_{\text{ker}}]$. Thus we demand that $a = b + 4\sigma_{\text{ker}}$, such that the potential goes to zero at $-a$ and $+a$.

4. If we want the (smoothened) potential V to have zero mean, then it is mapped to $\mathring{V}$ according to Eq. (74) in Subsubsec. B.2.1.

5. Momentum discretization: $k_j = k_{\min} + j\Delta k$, where $j \in \{0, \ldots, n_k - 1\}$ and $k_{\max} = k_{n_k - 1} \implies n_k = \frac{k_{\max} - k_{\min}}{\Delta k} + 1$.

Such a setup is denoted by the symbol $\mathcal{S}^{k_{\min}, k_{\max}, \Delta k}_{x_{\min}, x_{\max}, \Delta x}(\sigma_{\text{pot}}, \sigma_{\text{ker}})$. If the potentials are strictly nonnegative / have zero mean, we write $\mathcal{S}^{k_{\min}, k_{\max}, \Delta k}_{x_{\min}, x_{\max}, \Delta x}(\sigma_{\text{pot}}, \sigma_{\text{ker}}, +/0)$. The three kinds (1.(a), 1.(b) and 1.(c)) can be distinguished by observing the values of $x_{\min}$ and $x_{\max}$. The parameters σ_{pot} and σ_{ker} determine the amplitude and the autocorrelation of the potentials, respectively. For nonnegative potentials it holds that

$$\langle V \rangle = \int_0^\infty v \cdot f_{\text{normal}}\left(v | 0, \sigma_{\text{pot}}^2\right) dv = \frac{\sigma_{\text{pot}}}{\sqrt{2\pi}} \approx 0.4\sigma_{\text{pot}}. \tag{71}$$

As discussed in Subsec. 2.3, the two parameters σ_{pot} and σ_{ker} determine the localization length $\xi(k)$.

B.2.1 Mapping to zero-mean

Suppose we have a potential $V : \mathbb{R} \to \mathbb{R}$ with $\text{supp}(V) \subseteq [-a, a]$ and

$$\int_{-\infty}^{\infty} V(x)\, dx \neq 0. \tag{72}$$

We wish to map this potential V to a potential $\mathring{V} : \mathbb{R} \to \mathbb{R}$ with the properties $\mathrm{supp}(\mathring{V}) \subseteq [-a, a]$ and

$$\int_{-\infty}^{\infty} \mathring{V}(x)\, dx = 0. \tag{73}$$

One could achieve this by taking the Fourier transform $\tilde{V}(k)$ of the original potential, set the values around $k = 0$ to zero and define $\mathring{V}$ as the inverse Fourier transform of this modified function. However, this procedure in general produces a non-compact support of $\mathring{V}$. Therefore we choose another approach and simply add to V a smooth function with the compact support $[-a, a]$, i.e. a so-called test function, namely

$$\mathring{V}(x) = V(x) + A \exp\left(-\frac{a^2}{a^2 - x^2}\right) \Theta(a - |x|). \tag{74}$$

The used test function $\tau_a(x) := \exp\left(-a^2/(a^2 - x^2)\right) \Theta(a - |x|)$ is plotted in Fig. 42. With

$$\beta := \int_{-1}^{1} \exp\left(-\frac{1}{1 - z^2}\right) dz \approx 0.443\,993\,816 \tag{75}$$

one can calculate the parameter A such that condition (73) is fulfilled:

$$A = -\int_{-\infty}^{\infty} V(x)\, dx / (\beta a). \tag{76}$$

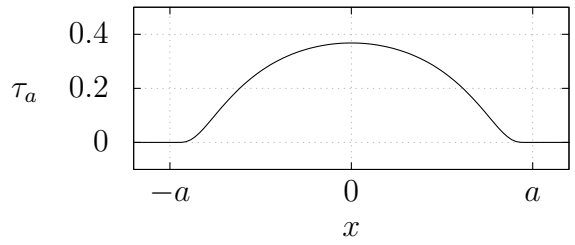

Figure 42: Plot of the test function $\tau_a(x) = \exp\left(-a^2/(a^2 - x^2)\right) \Theta(a - |x|)$, used for the mapping of a potential V to its zero-mean version $\mathring{V}$, cf. Eq. (74).

C One-dimensional forward scattering

C.1 Delta-potential

Consider the potential

$$V(x) = \sum_{i=1}^{N} V_i \, \delta(x - x_i), \tag{77}$$

where $N \in \mathbb{N}$, $x_1 < x_2 < \ldots < x_N$ and $\forall i \in \{1, \ldots, N\} : V_i \in \mathbb{R}$. The stationary Schrödinger equation (2) can be solved analytically with the following ansatz,

$$\forall i \in \{0, \ldots, N\} : \forall x \in (x_i, x_{i+1}) : \psi(k; x) = A_i(k) \, \mathrm{e}^{\mathrm{i}kx} + B_i(k) \, \mathrm{e}^{-\mathrm{i}kx}, \tag{78}$$

i.e. the quantum particle is propagating freely as a superposition of plane waves between the peaks. The points x_0 and x_{N+1} are to be understood as $-\infty$ and $+\infty$, respectively. Take any fixed $i \in \{1, \ldots, N\}$ and integrate the Schrödinger equation (2) over the interval $(x_i - \varepsilon, x_i + \varepsilon)$, where $0 < \varepsilon < \min\{x_{i+1} - x_i, x_i - x_{i-1}\}$:

$$\frac{\partial \psi}{\partial x}(k; x_i + \varepsilon) - \frac{\partial \psi}{\partial x}(k; x_i - \varepsilon) = \frac{2m}{\hbar^2} V_i \psi(k; x_i) - k^2 \int_{x_i - \varepsilon}^{x_i + \varepsilon} \psi(k; x) \, dx. \tag{79}$$

Since $\psi(k; x)$ is continuous in x, we get in the limit $\varepsilon \to 0$:

$$\frac{\partial \psi}{\partial x}\left(k; x_i^+\right) - \frac{\partial \psi}{\partial x}\left(k; x_i^-\right) = \frac{2mV_i}{\hbar^2} \psi(k; x_i). \tag{80}$$

The asymptotic behaviour of the wavefunction in the case "incidence from the left" is given by

$$\begin{aligned} \forall x < x_1 : \psi(k; x) &= \frac{1}{t(k)} \mathrm{e}^{\mathrm{i}kx} + \frac{r(k)}{t(k)} \mathrm{e}^{-\mathrm{i}kx}, \\ \forall x > x_N : \psi(k; x) &= \mathrm{e}^{\mathrm{i}kx}. \end{aligned} \tag{81}$$

This corresponds to $A_0(k) = 1/t(k)$, $B_0(k) = r(k)/t(k)$, $A_N(k) = 1$ and $B_N(k) = 0$. For any $i \in \{1, \ldots, N\}$ it holds that

$$\begin{aligned} \frac{\partial \psi}{\partial x}\left(k; x_i^+\right) &= \mathrm{i}k \left(A_i(k) \, \mathrm{e}^{\mathrm{i}kx_i} - B_i(k) \, \mathrm{e}^{-\mathrm{i}kx_i}\right), \\ \frac{\partial \psi}{\partial x}\left(k; x_i^-\right) &= \mathrm{i}k \left(A_{i-1}(k) \, \mathrm{e}^{\mathrm{i}kx_i} - B_{i-1}(k) \, \mathrm{e}^{-\mathrm{i}kx_i}\right). \end{aligned} \tag{82}$$

Combining these equations with Eq. (80) yields the following recursion formulae:

$$\begin{aligned} A_i(k) &= \left(1 + \frac{\mathrm{i}mV_{i+1}}{\hbar^2 k}\right) A_{i+1}(k) + \frac{\mathrm{i}mV_{i+1}}{\hbar^2 k} B_{i+1}(k)\, \mathrm{e}^{-2\mathrm{i}kx_{i+1}}, \\ B_i(k) &= B_{i+1}(k) + (A_{i+1}(k) - A_i(k))\, \mathrm{e}^{2\mathrm{i}kx_{i+1}}. \end{aligned} \tag{83}$$

Starting with $i = N-1$ one iterates down to $i = 0$. The transmission amplitude is then given by $t(k) = 1/A_0(k)$ and the reflection amplitude by $r(k) = B_0(k)/A_0(k)$.
We now turn to the second case "incidence from the right": The asymptotic boundary conditions read

$$\begin{aligned} \forall x < x_1 : \psi(k;x) &= \mathrm{e}^{-\mathrm{i}kx}, \\ \forall x > x_N : \psi(k;x) &= \frac{r'(k)}{t'(k)} \mathrm{e}^{\mathrm{i}kx} + \frac{1}{t'(k)} \mathrm{e}^{-\mathrm{i}kx}. \end{aligned} \tag{84}$$

As in case "incidence from the left" one can derive recursion formulae for the amplitudes $A_i(k)$ and $B_i(k)$:

$$\begin{aligned} A_0(k) &= 0, \\ B_0(k) &= 1, \\ A_i(k) &= \left(1 - \frac{\mathrm{i}mV_i}{\hbar^2 k}\right) A_{i-1}(k) - \frac{\mathrm{i}mV_i}{\hbar^2 k} B_{i-1}(k)\, \mathrm{e}^{-2\mathrm{i}kx_i}, \\ B_i(k) &= B_{i-1}(k) - (A_i(k) - A_{i-1}(k))\, \mathrm{e}^{2\mathrm{i}kx_i}. \end{aligned} \tag{85}$$

After the iteration from $i = 1$ up to $i = N$ we get $t'(k) = 1/B_N(k)$ and $r'(k) = A_N(k)/B_N(k)$.
In this way the forward scattering problem for delta-potentials can be solved analytically.

C.2 Numerical forward scattering procedures

In Subsec. 2.2 we introduce the Numerov algorithm for numerically solving the forward scattering problem of arbitrary smooth and compactly supported potentials. In the following two subsubsections we present two other numerical procedures and compare them against the Numerov algorithm in Subsubsec. C.2.3.

C.2.1 Lippmann-Schwinger integral equation

Let $V : \mathbb{R} \to \mathbb{R}$ be arbitrary with $\operatorname{supp}(V) \subseteq [-a, a]$. The stationary Schrödinger equation

$$\left[\frac{\partial^2}{\partial x^2} + k^2\right] \psi(k; x) = \frac{2m}{\hbar^2} V(x) \psi(k; x) \tag{86}$$

can be reformulated using the Green's function

$$\left[\frac{\partial^2}{\partial x^2} + k^2\right] G(k; x) = \delta(x) \tag{87}$$

to an integral equation, the so-called Lippmann-Schwinger equation

$$\begin{aligned} \psi(k; x) &= \psi_0(k; x) + \frac{2m}{\hbar^2} \int_{-\infty}^{\infty} G(k; x - x') V(x') \psi(k; x') \, dx' \\ &= \psi_0(k; x) + \frac{2m}{\hbar^2} \int_{-a}^{a} G(k; x - x') V(x') \psi(k; x') \, dx', \end{aligned} \tag{88}$$

where the incident wave ψ_0 satisfies the homogeneous equation

$$\left[\frac{\partial^2}{\partial x^2} + k^2\right] \psi_0(k; x) = 0. \tag{89}$$

In the case "incidence from the left" we choose the advanced Green's function

$$G_{\text{adv}}(k; x) = -\Theta(-x) \frac{\sin(kx)}{k} \tag{90}$$

and $\psi_0(k; x) = \mathrm{e}^{\mathrm{i}kx}$. With this choice we get

$$\psi(k; x) = \mathrm{e}^{\mathrm{i}kx} - \frac{2m}{\hbar^2 k} \int_{-a}^{a} \Theta(x' - x) \sin(k(x - x')) V(x') \psi(k; x') \, dx'. \tag{91}$$

This integral equation satisfies the correct boundary condition:

$$\begin{aligned} x > a &\implies \psi(k; x) = \mathrm{e}^{\mathrm{i}kx}, \\ -a < x < a &\implies \psi(k; x) = \mathrm{e}^{\mathrm{i}kx} - \frac{2m}{\hbar^2 k} \int_{x}^{a} \sin(k(x - x')) V(x') \psi(k; x') \, dx', \\ x < -a &\implies \psi(k; x) = \mathrm{e}^{\mathrm{i}kx} - \frac{2m}{\hbar^2 k} \int_{-a}^{a} \sin(k(x - x')) V(x') \psi(k; x') \, dx' \\ &\overset{!}{=} \frac{1}{t(k)} \mathrm{e}^{\mathrm{i}kx} + \frac{r(k)}{t(k)} \mathrm{e}^{-\mathrm{i}kx}. \end{aligned} \tag{92}$$

With $\sin(\varphi) = \left(\mathrm{e}^{\mathrm{i}\varphi} - \mathrm{e}^{-\mathrm{i}\varphi}\right)/(2\mathrm{i})$ we can read off the transmission and reflection amplitudes:

$$\begin{aligned} \frac{1}{t(k)} &= 1 + \frac{\mathrm{i}m}{\hbar^2 k}\int_{-a}^{a} \mathrm{e}^{-\mathrm{i}kx'} V(x')\,\psi(k;x')\,dx', \\ \frac{r(k)}{t(k)} &= -\frac{\mathrm{i}m}{\hbar^2 k}\int_{-a}^{a} \mathrm{e}^{\mathrm{i}kx'} V(x')\,\psi(k;x')\,dx'. \end{aligned} \tag{93}$$

For the case "incidence from the right" we choose the retarded Green's function

$$G_{\mathrm{ret}}(k;x) = \Theta(x)\,\frac{\sin(kx)}{k} \tag{94}$$

and $\psi_0(k;x) = \mathrm{e}^{-\mathrm{i}kx}$, because then

$$\psi(k;x) = \mathrm{e}^{-\mathrm{i}kx} + \frac{2m}{\hbar^2 k}\int_{-a}^{a} \Theta(x-x')\sin(k(x-x'))\,V(x')\,\psi(k;x')\,dx', \tag{95}$$

which is equivalent to

$$\begin{aligned} x < -a &\implies \psi(k;x) = \mathrm{e}^{-\mathrm{i}kx}, \\ -a < x < a &\implies \psi(k;x) = \mathrm{e}^{-\mathrm{i}kx} + \frac{2m}{\hbar^2 k}\int_{-a}^{x} \sin(k(x-x'))\,V(x')\,\psi(k;x')\,dx', \\ x > a &\implies \psi(k;x) = \mathrm{e}^{-\mathrm{i}kx} + \frac{2m}{\hbar^2 k}\int_{-a}^{a} \sin(k(x-x'))\,V(x')\,\psi(k;x')\,dx' \\ &\overset{!}{=} \frac{r'(k)}{t'(k)}\mathrm{e}^{\mathrm{i}kx} + \frac{1}{t'(k)}\mathrm{e}^{-\mathrm{i}kx}. \end{aligned} \tag{96}$$

The transmission and reflection amplitudes are given by

$$\begin{aligned} \frac{1}{t'(k)} &= 1 + \frac{\mathrm{i}m}{\hbar^2 k}\int_{-a}^{a} \mathrm{e}^{\mathrm{i}kx'} V(x')\,\psi(k;x')\,dx', \\ \frac{r'(k)}{t'(k)} &= -\frac{\mathrm{i}m}{\hbar^2 k}\int_{-a}^{a} \mathrm{e}^{-\mathrm{i}kx'} V(x')\,\psi(k;x')\,dx'. \end{aligned} \tag{97}$$

Eqs. (92), (93), (96) and (97) can be integrated numerically and thus an approximate solution to the FSP is obtained. However, this procedure is rather slow (cf. Subsec. C.2.3) and memory-intensive, since one must store the entire wavefunction during the calculation.

C.2.2 Transfer matrix method

The fundamental building block of the transfer matrix method is, as the name suggests, the transfer matrix $M(k)$. This matrix provides a complete description of a one-dimensional scattering process just like the scattering matrix $S(k)$. The transfer matrix connects the amplitudes of the wavefunction $\psi(k;x)$ at the left of the potential to the amplitudes of $\psi(k;x)$ at the right of the potential (cf. Fig. 1):

$$\begin{pmatrix} \psi_{\text{out}}^{\rightarrow} \\ \psi_{\text{in}}^{\leftarrow} \end{pmatrix} =: M(k) \begin{pmatrix} \psi_{\text{in}}^{\rightarrow} \\ \psi_{\text{out}}^{\leftarrow} \end{pmatrix}. \tag{98}$$

For the sake of readability we suppress the k-dependence of all quantities. The inverse transfer matrix maps the amplitudes at the right of the potential to the amplitudes at the left of the potential:

$$\begin{pmatrix} \psi_{\text{in}}^{\rightarrow} \\ \psi_{\text{out}}^{\leftarrow} \end{pmatrix} = M^{-1} \begin{pmatrix} \psi_{\text{out}}^{\rightarrow} \\ \psi_{\text{in}}^{\leftarrow} \end{pmatrix}. \tag{99}$$

The transfer matrix M contains exactly the same information as the corresponding scattering matrix S, i.e. there is a bijection between both matrices:

$$M = \frac{1}{S_{12}} \begin{pmatrix} S_{12}S_{21} - S_{11}S_{22} & S_{22} \\ -S_{11} & 1 \end{pmatrix} = \frac{1}{t'} \begin{pmatrix} t't - r'r & r' \\ -r & 1 \end{pmatrix} \tag{100}$$

$$\iff S = \frac{1}{M_{22}} \begin{pmatrix} -M_{21} & 1 \\ M_{11}M_{22} - M_{12}M_{21} & M_{12} \end{pmatrix}. \tag{101}$$

The element M_{11} can be simplified in the following way:

$$M_{11} = \frac{t't - r'r}{t'} = t - \frac{r'}{t'}r = t + \frac{r^*}{t^*}r = \frac{|t|^2 + |r|^2}{t^*} = \frac{1}{t^*}. \tag{102}$$

Here we made use of Eq. (7). This allows us to write the transfer matrix in the more symmetric form

$$M = \begin{pmatrix} 1/t^* & r'/t' \\ -r/t' & 1/t' \end{pmatrix}. \tag{103}$$

The unitarity of the scattering matrix S, which is a consequence of the conservation of the probability current, translates into the following condition for the transfer matrix:

$$M^\dagger \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} M = \begin{pmatrix} |M_{11}|^2 - |M_{21}|^2 & M_{11}^* M_{12} - M_{21}^* M_{22} \\ M_{12}^* M_{11} - M_{22}^* M_{21} & |M_{12}|^2 - |M_{22}|^2 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \quad (104)$$

In case of time-reversal symmetry, where the scattering matrix is symmetric, it must hold that

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} M \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = M^* \iff M_{22} = M_{11}^* \wedge M_{21} = M_{12}^*. \quad (105)$$

Thus the transfer matrix is of the form

$$M = \begin{pmatrix} M_{11} & M_{12} \\ M_{12}^* & M_{11}^* \end{pmatrix} = \begin{pmatrix} 1/t^* & -r^*/t^* \\ -r/t & 1/t \end{pmatrix} = \begin{pmatrix} 1/t^* & r'/t \\ -r/t & 1/t \end{pmatrix}. \quad (106)$$

Furthermore it holds that

$$\det(M) = \frac{S_{21}}{S_{12}} = \frac{t}{t'} = 1, \quad (107)$$

$$\operatorname{tr}(M) = M_{11} + M_{11}^* = 2\Re(M_{11}) \in \mathbb{R}. \quad (108)$$

The most interesting property of transfer matrices is their composition: Let M_1 and M_2 be the transfer matrix of a potential with its support being a subset of $[a, b]$ and $[b, c]$, respectively. Then the transfer matrix M of the sum of the two potentials is given by the product

$$M = M_2 M_1. \quad (109)$$

This composition rule generalizes rather straightforwardly and lies at the core of the transfer matrix method: The given potential $V(x)$ is approximated by a piecewise constant potential $\overline{V}(x)$, defined by

$$\forall i \in \mathbb{Z} \, \forall x \in \left[\frac{x_{i-1} + x_i}{2}, \frac{x_i + x_{i+1}}{2} \right) : \overline{V}(x) := V(x_i), \quad (110)$$

where x_i is an arbitrary space discretization. This is schematically depicted in Fig. 43. We can calculate the transfer matrix $M^{(i)}$ of each constant section analytically (as is done shortly). These individual transfer matrices are multiplied together, giving the transfer matrix $\overline{M}$ of the piecewise constant potential $\overline{V}$. This matrix $\overline{M}$ is an

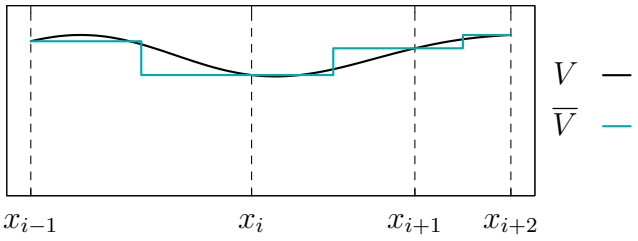

Figure 43: Example of an arbitrary potential V and its corresponding piecewise constant approximation $\overline{V}$, based on the space discretization x_i. (In a numerical calculation one would of course do a much finer discretization – for this figure we pick a rather coarse discretization in order to clearly see what is going on.)

approximation for the transfer matrix M corresponding to the original potential V:

$$M \approx \overline{M} = \prod_{i=0}^{2I} M^{(I-i)} = M^{(I)} M^{(I-1)} \dots M^{(-I+1)} M^{(-I)}. \tag{111}$$

We now calculate the transfer matrix of the rectangular potential

$$V(x) = \begin{cases} V_0 & x \in [b_1, b_2] \\ 0 & x \notin [b_1, b_2] \end{cases} \tag{112}$$

with $V_0 \in \mathbb{R}$. The stationary Schrödinger equation (2) with this potential can be solved analytically with the following ansatz:

$$\begin{aligned} \forall x < b_1 : \psi(k;x) &= \psi_{\text{in}}^{\rightarrow} \mathrm{e}^{\mathrm{i}kx} + \psi_{\text{out}}^{\leftarrow} \mathrm{e}^{-\mathrm{i}kx}, \\ \forall x \in [b_1, b_2] : \psi(k;x) &= \psi_0^{\rightarrow} \mathrm{e}^{\mathrm{i}k_0(k)x} + \psi_0^{\leftarrow} \mathrm{e}^{-\mathrm{i}k_0(k)x}, \\ \forall x > b_2 : \psi(k;x) &= \psi_{\text{out}}^{\rightarrow} \mathrm{e}^{\mathrm{i}kx} + \psi_{\text{in}}^{\leftarrow} \mathrm{e}^{-\mathrm{i}kx}. \end{aligned} \tag{113}$$

In the region of the potential, $[b_1, b_2]$, the quantum particle has the momentum

$$k_0(k) := \frac{\sqrt{2m(E(k) - V_0)}}{\hbar} = \sqrt{k^2 - \frac{2mV_0}{\hbar^2}}, \tag{114}$$

where $E(k)$ is the free-space energy and k is the free-space momentum of the quantum particle. Depending on the energy $E(k)$, the momentum $k_0(k)$ is real and positive for $E(k) > V_0$ (classically allowed) or imaginary with $\Im(k_0) > 0$ (w.l.o.g.) for $E(k) < V_0$

(quantum tunneling). In the latter case one usually defines

$$\kappa_0(k) := -\mathrm{i}k_0(k) = \frac{\sqrt{2m(V_0 - E(k))}}{\hbar} \in \mathbb{R}^+, \tag{115}$$

such that

$$\forall x \in [b_1, b_2] : \psi(k;x) = \psi_0^{\rightarrow} \mathrm{e}^{-\kappa_0(k)x} + \psi_0^{\leftarrow} \mathrm{e}^{\kappa_0(k)x}, \tag{116}$$

emphasizing the exponential behaviour of the wavefunction in the classically forbidden region.
Demanding that $\psi(k;x)$ and $\frac{\partial\psi}{\partial x}(k;x)$ are continuous in x yields the following transfer matrix:

$$\begin{aligned}
M_{11} &= \mathrm{e}^{-\mathrm{i}k(b_2-b_1)} \left(\cos(k_0(b_2-b_1)) + \frac{\mathrm{i}}{2} \left(\frac{k}{k_0} + \frac{k_0}{k} \right) \sin(k_0(b_2-b_1)) \right) \\
&= \mathrm{e}^{-\mathrm{i}k(b_2-b_1)} \left(\cosh(\kappa_0(b_2-b_1)) + \frac{\mathrm{i}}{2} \left(\frac{k}{\kappa_0} - \frac{\kappa_0}{k} \right) \sinh(\kappa_0(b_2-b_1)) \right), \\
M_{12} &= -\mathrm{e}^{-\mathrm{i}k(b_1+b_2)} \frac{\mathrm{i}}{2} \left(\frac{k}{k_0} - \frac{k_0}{k} \right) \sin(k_0(b_2-b_1)) \\
&= -\mathrm{e}^{-\mathrm{i}k(b_1+b_2)} \frac{\mathrm{i}}{2} \left(\frac{k}{\kappa_0} + \frac{\kappa_0}{k} \right) \sinh(\kappa_0(b_2-b_1)), \\
M_{21} &= \mathrm{e}^{\mathrm{i}k(b_1+b_2)} \frac{\mathrm{i}}{2} \left(\frac{k}{k_0} - \frac{k_0}{k} \right) \sin(k_0(b_2-b_1)) \\
&= \mathrm{e}^{\mathrm{i}k(b_1+b_2)} \frac{\mathrm{i}}{2} \left(\frac{k}{\kappa_0} + \frac{\kappa_0}{k} \right) \sinh(\kappa_0(b_2-b_1)), \\
M_{22} &= \mathrm{e}^{\mathrm{i}k(b_2-b_1)} \left(\cos(k_0(b_2-b_1)) - \frac{\mathrm{i}}{2} \left(\frac{k}{k_0} + \frac{k_0}{k} \right) \sin(k_0(b_2-b_1)) \right) \\
&= \mathrm{e}^{\mathrm{i}k(b_2-b_1)} \left(\cosh(\kappa_0(b_2-b_1)) - \frac{\mathrm{i}}{2} \left(\frac{k}{\kappa_0} - \frac{\kappa_0}{k} \right) \sinh(\kappa_0(b_2-b_1)) \right).
\end{aligned} \tag{117}$$

One can show that Eqs. (104), (105), (107) and (108) are fulfilled. From Eq. (106) we can deduce:

$$\begin{aligned}
r(k) &= \frac{\mathrm{e}^{2\mathrm{i}kb_1 \frac{2mV_0}{\hbar^2}}}{2k^2 - \frac{2mV_0}{\hbar^2} + 2\mathrm{i}kk_0(k)\cot(k_0(k)(b_2-b_1))}, \\
r'(k) &= \frac{\mathrm{e}^{-2\mathrm{i}kb_2 \frac{2mV_0}{\hbar^2}}}{2k^2 - \frac{2mV_0}{\hbar^2} + 2\mathrm{i}kk_0(k)\cot(k_0(k)(b_2-b_1))}, \\
t(k) = t'(k) &= \frac{\mathrm{e}^{-\mathrm{i}k(b_2-b_1)}}{\cos(k_0(k)(b_2-b_1)) - \frac{\mathrm{i}}{2}\left(\frac{k}{k_0(k)} + \frac{k_0(k)}{k} \right) \sin(k_0(k)(b_2-b_1))}.
\end{aligned} \tag{118}$$

C.2.3 Comparison of the numerical forward scattering procedures

In (supervised) DL it is essential to have a considerable amount of training data in order for the model to learn well (cf. Subsubsec. 4.3.3). Besides calculating in parallel using a GPU, the procedure used to calculate the scattering data itself shall be fast and accurate.

In Subsec. 2.2 as well as in Subsubsecs. C.2.1 and C.2.2 we presented three procedures for numerically calculating the scattering data of arbitrary smooth potentials. The different procedures are listed in Table 4 along with some of their main properties.

n_x denotes the number of spacial points and n_k the number of k-values. What determines the choice of n_x and n_k? The accuracy of each procedure gets better with increasing n_x (decreasing the grid spacing Δx, a lower bound is determined by the machine precision). Since the scattering problem is solved for each k independently, the number n_k is simply determined by the number of k-values at which one wishes to know the scattering matrix.

The column "No. prop." gives the number of times one has to propagate over the potential in order to get the whole scattering matrix. The last column states if the procedure can calculate and return the wavefunctions. One of the main differences between the Lippmann-Schwinger integration and the Numerov algorithm is that in the former one has to store the whole wavefunction for the entire calculation, whereas the latter is a two step method, i.e. it does not require such a storage.

In order to quantitatively compare the different procedures, we keep the setup fixed to $\mathcal{S}_{-5,5,0.05}^{0.005,5,0.005}(10, 0.3)$ (cf. Sec. B.2). We also want to compare the accuracies of the procedures, thus we need a benchmark. For this purpose we could simply take piecewise constant potentials (where the widths of the constant pieces are much larger than Δx), where the analytical solutions are known. But this would give the transfer matrix method an advantage, since it yields the exact solution per construction. Therefore we proceed as follows: A potential V is created on a grid with $\Delta x = 0.01$. The scattering amplitudes are calculated using the Numerov algorithm. This serves as the

Table 4: List of the forward scattering procedures and some of their properties.

Procedure	Complexity	No. prop.	Wavefunctions
Lippmann-Schwinger integration	$\mathcal{O}(n_x^2 n_k)$	2	yes
Numerov algorithm	$\mathcal{O}(n_x n_k)$	2	yes
Transfer matrix method	$\mathcal{O}(n_x n_k)$	1	no

Table 5: Quantitative comparison of forward scattering procedures. The best values are written in bold.

Procedure	$\langle \text{time} \rangle$	$\langle \text{acc}\,[S] \rangle$	$\langle \text{acc}\,[\psi] \rangle$	$\langle \text{unit}\,[S] \rangle$
Lippmann-Schwinger integration	132 s	$1.7 \cdot 10^{-3}$	$2.8 \cdot 10^{-2}$	$\mathbf{5.6 \cdot 10^{-16}}$
Numerov algorithm	**0.72 s**	$\mathbf{1.1 \cdot 10^{-3}}$	$\mathbf{1.2 \cdot 10^{-2}}$	$4.5 \cdot 10^{-14}$
Transfer matrix method	3.6 s	$2.2 \cdot 10^{-3}$	—	$4.0 \cdot 10^{-15}$

benchmark (regarding accuracy). Then the potential V is downsampled onto a grid with $\Delta x = 0.05$. The three different procedures are evaluated on this downsampled potential. The results are shown in Table 5.

Angle brackets $\langle \cdot \rangle$ without index denote an ensemble average. "$\langle \text{time} \rangle$" is the mean computation time (averaged over 25 sequential runs). For "$\langle \text{acc}\,[S] \rangle$" and "$\langle \text{unit}\,[S] \rangle$" 10^4 samples are processed in parallel and the mean value is taken in the end. The same is done for "$\langle \text{acc}\,[\psi] \rangle$", but with 300 samples only (due to memory issues).

In order to define the accuracies and the deviation from unitarity we introduce the x- and the k-average of a function $f(x)$ or $g(k)$:

$$\langle f(x) \rangle_x := \frac{1}{2a} \int_{-a}^{a} f(x)\, dx \approx \frac{1}{2I} \sum_{i=-I}^{I} f(x_i)\,, \tag{119}$$

$$\langle g(k) \rangle_k := \frac{1}{k_{\max} - k_{\min}} \int_{k_{\min}}^{k_{\max}} g(k)\, dk \approx \frac{1}{n_k - 1} \sum_{j=0}^{n_k - 1} g(k_j)\,. \tag{120}$$

Let $S_b(k)$ be the benchmark scattering matrix and $S(k)$ the scattering matrix calculated with either of the three procedures. We define the accuracy of S as

$$\text{acc}\,[S] := \langle \| S(k) - S_b(k) \|_F \rangle_k\,, \tag{121}$$

where $\|\cdot\|_F$ denotes the Frobenius norm.

Let $\psi_b^{l/r}(k;x)$ be the benchmark wavefunction w.r.t. incidence from the left and right, respectively, and $\psi^{l/r}(k;x)$ the wavefunctions obtained by Lippmann-Schwinger integration or the Numerov algorithm. The wavefunctions are normalized according to

$$\lim_{x \to \pm\infty} \psi_{(b)}^{l/r}(k;x)\, \mathrm{e}^{\mp \mathrm{i}kx} = 1. \tag{122}$$

We define the accuracy of the wavefunctions as

$$\operatorname{acc}[\psi] := \left\langle \left\langle \frac{\left|\psi^l(k;x) - \psi_b^l(k;x)\right| + \left|\psi^r(k;x) - \psi_b^r(k;x)\right|}{2} \right\rangle_x \right\rangle_k . \tag{123}$$

The deviation of the scattering matrix S from unitarity is defined as (dropping the k-dependencies, cf. Eqs. (7) and (8))

$$\operatorname{unit}[S] := \left\langle \frac{|T + R - 1| + |T' + R' - 1| + |r^* t' + t^* r'| + |r^* t + t'^* r'|}{4} \right\rangle_k . \tag{124}$$

Table 5 reveals that the Numerov algorithm outperforms the other procedures in nearly all regards. Only the deviation from unitarity is largest for the Numerov method, but it is still reasonably small. Therefore we choose the Numerov algorithm for all the numerical forward scattering calculations.
One could argue that the results are biased in favour of the Numerov algorithm, since the benchmark is done with it. However, the time and unit $[S]$ do not require any benchmark in the first place. And when using the transfer matrix method as the benchmark, the results for $\langle \operatorname{acc}[S] \rangle$ are the same. The Lippmann-Schwinger integration was not tried as a benchmark due to memory issues.

D One-dimensional inverse scattering

D.1 Proof that all considered potentials lie in L_2^1

In Subsec. 3.1 we state that if a potential $V : \mathbb{R} \to \mathbb{R}$ lies in the set

$$L_2^1 = \left\{ V : \mathbb{R} \to \mathbb{R} \,\middle|\, \int_{-\infty}^{\infty} |V(x)| \left(1 + x^2\right) dx < \infty \right\}, \tag{125}$$

then the corresponding inverse scattering problem has a unique solution, if (additionally to the scattering data) one has enough information about the bound states of V. We want to show that all potentials considered in this thesis lie in this set.
The first category of potentials have a finite number of positive delta-peaks, i.e.

$$V(x) = \sum_{i=1}^{N} V_i \, \delta(x - x_i), \tag{126}$$

where $N \in \mathbb{N}$ is the number of peaks, $x_i \in \mathbb{R}$ are their positions and $V_i \in \mathbb{R}^+$ their strengths. It holds that

$$\int_{-\infty}^{\infty} |V(x)| \left(1 + x^2\right) dx = \sum_{i=1}^{N} V_i \int_{-\infty}^{\infty} \delta(x - x_i) \left(1 + x^2\right) dx = \sum_{i=1}^{N} V_i \left(1 + x_i^2\right) < \infty \tag{127}$$

and hence $V \in L_2^1$.
The other category of potentials under consideration have a finite support, $\operatorname{supp}(V) \subseteq [-a, a]$ with $a < \infty$, and reach only finite values, $\forall x \in \mathbb{R} : |V(x)| \leq F < \infty$. For such potentials we can estimate

$$\int_{-\infty}^{\infty} |V(x)| \left(1 + x^2\right) dx \leq F \int_{-a}^{a} \left(1 + x^2\right) dx = 2Fa \left(1 + \frac{a^2}{3}\right) < \infty \tag{128}$$

and thus $V \in L_2^1$.

D.2 Properties of the normalization constants

There is a link between the normalization constants c_n and c'_n, defined in Eq. (28), and the analytical continuation of the reflection amplitudes $r(k)$ and $r'(k)$: If the potential $V(x)$ vanishes for $x < x_0$ for some $x_0 \in \mathbb{R}$, then the left normalization constants c_n are related to the residues r_n of the left reflection amplitude $r(k)$ at its poles $k = \mathrm{i}\kappa_n$ (cf.

Subsec. 3.1) by [18]

$$c_n = -\mathrm{i} r_n. \tag{129}$$

If the potential $V(x)$ vanishes for $x > x_0$ for some $x_0 \in \mathbb{R}$, then the right normalization constants c'_n are related to the residues r'_n of the right reflection amplitude $r'(k)$ at its poles $k = \mathrm{i}\kappa_n$ by

$$c'_n = -\mathrm{i} r'_n. \tag{130}$$

D.3 Shift and reflection of the potential

In Subsec. 3.2 we assume that the potential $V(x)$ vanishes for $x < 0$. If the potential vanishes on a different half-line, one has to shift and/or mirror the potential in order to get it to a form where it vanishes for $x < 0$.

Suppose the potential $V(x)$ vanishes for $x > 0$, then the mirrored potential $V_{\mathrm{mir}}(x) := V(-x)$ vanishes for $x < 0$. Therefore one can solve the Marčenko integral equation for V_{mir} and mirror it back to obtain the original potential V. In order to do so, one has to find the corresponding function $\mathcal{R}_{\mathrm{mir}}(x)$ (cf. Eq. (29)). It is clear that the left and right reflection amplitudes $r(k)$ and $r'(k)$ change roles when the potential is mirrored, i.e. $r_{\mathrm{mir}}(k) = r'(k)$ and $r'_{\mathrm{mir}}(k) = r(k)$. The energies of the bound states do not change under a parity transformation of the potential, but the normalization constants do change according to $c_{\mathrm{mir},n} = c'_n$ and $c'_{\mathrm{mir},n} = c_n$.

Now suppose that the potential $V(x)$ vanishes only for $x < -\frac{\delta}{2}$ for some $\delta > 0$. The shifted potential $V_{\mathrm{shift}}(x) := V\left(x - \frac{\delta}{2}\right)$ vanishes for $x < 0$. The reflection amplitudes and the normalization constants change according to $r_{\mathrm{shift}}(k) = \mathrm{e}^{\mathrm{i}k\delta} r(k)$, $r'_{\mathrm{shift}}(k) = \mathrm{e}^{-\mathrm{i}k\delta} r'(k)$, $c_{\mathrm{shift},n} = \mathrm{e}^{-\delta\kappa_n} c_n$ and $c'_{\mathrm{shift},n} = \mathrm{e}^{\delta\kappa_1} c'_n$. One can solve the Marčenko integral equation for V_{shift} and shift it back to retrieve the original potential V.

If the potential $V(x)$ vanishes only for $x > \frac{\delta}{2}$ for some $\delta > 0$, then one has to apply both a shift and a reflection.

D.4 Alternative formulations of the Marčenko integral equation

Besides the formulation of the Marčenko integral equation given in Eq. (31), favourable for numerical calculations, there are other common formulations, which come about by simple coordinate transformations. In this subsection we want to show two such formulations. The function $\mathcal{R}(x)$ stays the same, but the function $B(y, x)$ has to be replaced by $\mathcal{K}(\zeta, \eta)$ or $K(x, z)$, respectively. We start with one of the two alternative

formulations, transform it to the second one, which in turn is transformed into the formulation given in Subsec. 3.2.
For the first alternative formulation we do not demand that the potential $V(x)$ has to vanish for $x < 0$, but for $x < -\frac{\delta}{2}$ for some $\delta \geq 0$. Then the Marčenko integral equation for the unknown function $\mathcal{K}(\zeta, \eta)$ reads

$$\mathcal{R}(\zeta + \eta) + \mathcal{K}(\zeta, \eta) + \int_{-(\eta+\delta)}^{\zeta} \mathcal{K}(\zeta, \zeta') \mathcal{R}(\eta + \zeta') \, d\zeta' = 0 \tag{131}$$

together with the restrictions $\zeta \geq \eta$ and $\zeta \geq -(\eta + \delta)$. From the solution $\mathcal{K}(\zeta, \eta)$ of this equation one can retrieve the potential by

$$V(x) = \frac{\hbar^2}{m} \frac{d\mathcal{K}(x, x)}{dx}. \tag{132}$$

In the literature one mostly finds the special case $\delta = 0$, i.e. $V(x)$ vanishes for $x < 0$, which we also want to assume in the following. In the previous subsection D.3 we describe how to transform all the quantities if V is shifted or mirrored.
For the second alternative representation, one introduces the variables

$$x := \frac{\zeta + \eta}{2}, z := \frac{\zeta - \eta}{2} \iff \zeta = x + z, \eta = x - z \tag{133}$$

and defines

$$K(x, z) := \mathcal{K}(x + z, x - z) \iff \mathcal{K}(\zeta, \eta) = K\left(\frac{\zeta + \eta}{2}, \frac{\zeta - \eta}{2}\right). \tag{134}$$

Then the Marčenko equation in Eq. (131) turns into

$$\mathcal{R}(2x) + K(x, z) + 2 \int_0^x K(x' + z, x - x') \mathcal{R}(2x') \, dx' = 0 \tag{135}$$

together with $|x - z| \leq x + z \implies x + z \geq 0$. The potential is then given by

$$V(x) = \frac{\hbar^2}{m} \frac{dK(x, 0)}{dx}. \tag{136}$$

The representation given in Eq. (31) is obtained by keeping x, replacing z by

$$y := x + z \iff z = y - x \tag{137}$$

and defining

$$B(y,x) := K(x,y-x) \iff K(x,z) = B(x+z,x) \tag{138}$$

The restrictions on the variables translate to $y \geq 0$ and $|2x-y| \leq y$, which is equivalent to $x \geq 0$, $y \geq 0$ and $x \leq y$.

D.5 First-order Born approximation

We can use the first-order Born approximation from forward scattering and invert it to get an approximate solution of the inverse scattering problem. In the first-order Born approximation one essentially replaces the wavefunction ψ in the integrand of the Lippmann-Schwinger equation (88) by the incoming wave ψ_0:

$$\psi(k;x) \approx \psi_0(k;x) + \frac{2m}{\hbar^2}\int_{-\infty}^{\infty} G(k;x-x')\,V(x')\,\psi_0(k;x')\,dx'. \tag{139}$$

Here we use a special Green's function, obtained by adding a solution of the homogeneous equation

$$\left[\frac{\partial^2}{\partial x^2} + k^2\right] g_{\text{hom}}(k;x) = 0, \tag{140}$$

which is a linear combination of the functions $\sin(kx)$ and $\cos(kx)$, to a particular solution of the inhomogeneous equation

$$\left[\frac{\partial^2}{\partial x^2} + k^2\right] G(k;x) = \delta(x), \tag{141}$$

namely the retarded Green's function given by

$$G_{\text{ret}}(k;x) = \Theta(x)\,\frac{\sin(kx)}{k}. \tag{142}$$

The new special Green's function is defined as

$$G_{\overline{\text{F}}}(k;x) := G_{\text{ret}}(k;x) + \frac{1}{2\text{i}k}\cos(kx) - \frac{1}{2k}\sin(kx) = \frac{\text{e}^{\text{i}k|x|}}{2\text{i}k}. \tag{143}$$

This Green's function is referred to as the "anti-Feynman Green's function" [61]. We will see shortly why this Green's function is advantageous here.
In the case "incidence from the left" we have the following conditions (note the difference

to Eq. (12)):

$$\begin{aligned}\psi_0(k;x) &= \mathrm{e}^{\mathrm{i}kx} \\ \psi(k;x) &= \begin{cases} \mathrm{e}^{\mathrm{i}kx} + r(k)\,\mathrm{e}^{-\mathrm{i}kx} & x < -a \\ t(k)\,\mathrm{e}^{\mathrm{i}kx} & x > a \end{cases}\end{aligned} \tag{144}$$

with $k \geq 0$. Eq. (139) evaluated at $x < -a$ and $x > a$ yields

$$\begin{aligned} r(k)\,\mathrm{e}^{-\mathrm{i}kx} &\approx \frac{2m}{\hbar^2}\int_{-a}^{a} \frac{\mathrm{e}^{-\mathrm{i}k(x-x')}}{2\mathrm{i}k} V(x')\,\mathrm{e}^{\mathrm{i}kx'}\,dx' \\ \iff r(k) &\approx \frac{m}{\mathrm{i}\hbar^2 k}\int_{-\infty}^{\infty} \mathrm{e}^{2\mathrm{i}kx'} V(x')\,dx' \end{aligned} \tag{145}$$

and

$$\begin{aligned} (t(k)-1)\,\mathrm{e}^{\mathrm{i}kx} &\approx \frac{2m}{\hbar^2}\int_{-a}^{a} \frac{\mathrm{e}^{\mathrm{i}k(x-x')}}{2\mathrm{i}k} V(x')\,\mathrm{e}^{\mathrm{i}kx'}\,dx' \\ \iff t(k)-1 &\approx \frac{m}{\mathrm{i}\hbar^2 k}\int_{-\infty}^{\infty} V(x')\,dx' \end{aligned} \tag{146}$$

respectively. If we had chosen another Green's function, then the complex exponential factors on the left hand sides would not cancel and thus a single x-dependent factor would be left over.

Performing the same calculations in the case "incidence from the right", i.e.

$$\begin{aligned}\psi_0(k;x) &= \mathrm{e}^{-\mathrm{i}kx} \\ \psi(k;x) &= \begin{cases} t'(k)\,\mathrm{e}^{-\mathrm{i}kx} & x < -a \\ \mathrm{e}^{-\mathrm{i}kx} + r'(k)\,\mathrm{e}^{\mathrm{i}kx} & x > a \end{cases}\end{aligned} \tag{147}$$

with $k \geq 0$, yields the following equations:

$$r'(k) \approx \frac{m}{\mathrm{i}\hbar^2 k}\int_{-\infty}^{\infty} \mathrm{e}^{-2\mathrm{i}kx'} V(x')\,dx', \tag{148}$$

$$t'(k) - 1 \approx \frac{m}{\mathrm{i}\hbar^2 k}\int_{-\infty}^{\infty} V(x')\,dx'. \tag{149}$$

As we can see, in the first-order Born approximation the transmission amplitudes contain only a single numerical information about the scattering potential V, namely its integral. The reflection amplitudes however resemble the Fourier transform of the potential and therefore contain much more information about V. Since $r(k)$ and $r'(k)$

are only defined for $k \in \mathbb{R}_0^+$, we introduce the function $\rho : \mathbb{R} \to \mathbb{C}$ by

$$\rho(k) := \begin{cases} r(k) & k \geq 0 \\ r'(-k) & k < 0 \end{cases}. \tag{150}$$

It then holds that

$$\rho(k) \approx \frac{m}{\mathrm{i}\hbar^2 |k|} \int_{-\infty}^{\infty} \mathrm{e}^{2\mathrm{i}kx'} V(x')\, dx'. \tag{151}$$

The inversion of this Fourier transformation gives

$$\begin{aligned} V(x) &\approx \frac{\mathrm{i}\hbar^2}{\pi m} \int_{-\infty}^{\infty} \mathrm{e}^{-2\mathrm{i}kx} |k| \rho(k)\, dk \\ &= \frac{\mathrm{i}\hbar^2}{\pi m} \int_0^{\infty} k \left(\mathrm{e}^{-2\mathrm{i}kx} r(k) + \mathrm{e}^{2\mathrm{i}kx} r'(k) \right) dk =: V_{\mathrm{Born}}(x). \end{aligned} \tag{152}$$

Since the approximate potential $V_{\mathrm{Born}} : \mathbb{R} \to \mathbb{C}$ is just the Fourier transform of $|k| \rho(k)$, it contains the same information as the reflection amplitudes themselves.
The approximate potential V_{Born} shows some desirable features:

- In the absence of any potential, i.e. for $V(x) = 0$, we have $r(k) = r'(k) = 0$ and therefore also $V_{\mathrm{Born}}(x) = 0$.
- If V is symmetric under a parity transformation, i.e. $V(-x) = V(x)$, then $r'(k) = r(k)$ and consequently $V_{\mathrm{Born}}(-x) = V_{\mathrm{Born}}(x)$.

On the other hand, there are also major restrictions, e.g.

$$\int_{-\infty}^{\infty} V_{\mathrm{Born}}(x)\, dx = \frac{\mathrm{i}\hbar^2}{m} \int_0^{\infty} k \left(\delta(k) r(k) + \delta(k) r'(k) \right) dk = 0, \tag{153}$$

since $|r(k)| \leq 1$ and $|r'(k)| \leq 1$.
We highlight a close link between the zeroth-order Marčenko approximations and the real part of the first-order Born approximation. From $\Re(\mathrm{i}z) = -\Im(z)$ and Eq. (152) we get

$$\Re\left(V_{\mathrm{Born}}(x)\right) = -\Im\left(\frac{\hbar^2}{\pi m} \int_0^{\infty} k \left(\mathrm{e}^{-2\mathrm{i}kx} r(k) + \mathrm{e}^{2\mathrm{i}kx} r'(k) \right) dk \right). \tag{154}$$

Using $r(-k) = r^*(k)$ (cf. Eq. (25)) and $\mathrm{i}(z - z^*) = -2\Im(z)$ one can calculate

$$V_{\mathrm{Marč}}^{(0)}(x) = -\Im\left(\frac{2\hbar^2}{\pi m} \int_0^{\infty} r(k)\, k \mathrm{e}^{-2\mathrm{i}kx}\, dk \right) \tag{155}$$

and

$$V'^{(0)}_{\text{Marč}}(x) = -\Im\left(\frac{2\hbar^2}{\pi m}\int_0^\infty r'(k)\, k\mathrm{e}^{2\mathrm{i}kx}\, dk\right). \tag{156}$$

We see that

$$\Re\left(V_{\text{Born}}(x)\right) = \frac{1}{2}\left(V^{(0)}_{\text{Marč}}(x) + V'^{(0)}_{\text{Marč}}(x)\right), \tag{157}$$

i.e. the real part of the first-order Born approximation is the mean of the two zeroth-order Marčenko approximations.

D.6 Comparison of approximate inversion methods

We compare here the two inversion methods presented in Subsubsecs. D.5 and 3.3 (cf. Eqs. (152), (36), (37) and (42)) qualitatively and quantitatively. We introduce the abbreviations

- BA := first-order Born approximation,
- MA := Marčenko approximation,
- MA0 := zeroth-order Marčenko approximation,
- MA1 := first-order Marčenko approximation.

For a qualitative comparison we use six (not strongly scattering) test scenarios: a small, a medium and a large positive rectangular potential, once unmodified (i.e. sharp) and once smoothened. The positivity of the potential means that there are no bound states, i.e. the ISP can be solved uniquely (cf. Subsec. 3.1). The space and momentum parameters are chosen as $\mathcal{S}^{0.01,20,0.01}_{-2,2,0.01}$ (cf. Subsec. B.2). The small / medium / large potential has height 0.25 / 0.5 / 0.75 and width 0.5 / 1.0 / 1.5, all centered at $x = 0$. Smoothing is achieved by convolution with a Gaussian kernel with a standard deviation of 0.05.

Fig. 44 shows the reflection amplitudes of all six potentials as a function of momentum. Since all approximation methods solely build upon the reflection amplitudes, the transmission amplitudes are omitted. Due to the symmetry of the potentials it holds that $r'(k) = r(k)$. Fig. 45 shows the reflection coefficients of all six potentials as a function of momentum. One can see that the reflection decays faster if the potential is smoothened.

The different approximations are compared to the original potentials in Fig. 46. There are several interesting observations:

All the approximations of the sharp potentials show oscillatory behaviour and the Gibbs phenomenon is present at the discontinuities. This can be explained by the finite range

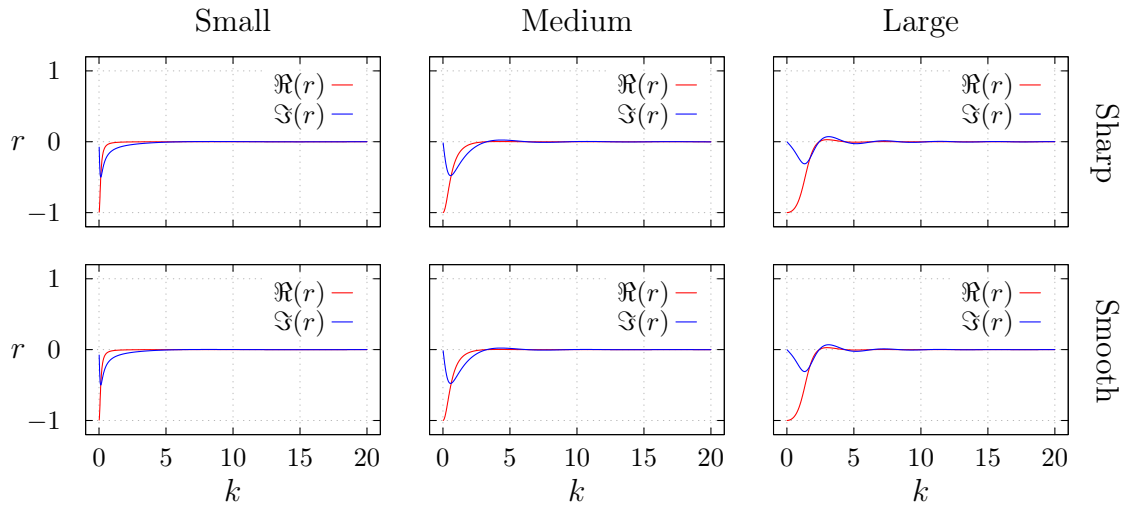

Figure 44: Left reflection amplitudes $r(k)$ of the six rectangular potentials. The top row is for the sharp (unsmoothened) potentials and the bottom row for the smoothened potentials. The left column is for the small (height = 0.25, width = 0.5) potentials, the middle column for the medium (height = 0.5, width = 1.0) potentials and the right column for the large (height = 0.75, width = 1.5) potentials. One can barely see the differences between the sharp and the smoothened potentials, which is why the reflection coefficients $|r(k)|^2$ are shown in Fig. 45 in a logarithmic scale, where the discrepancies are more evident.

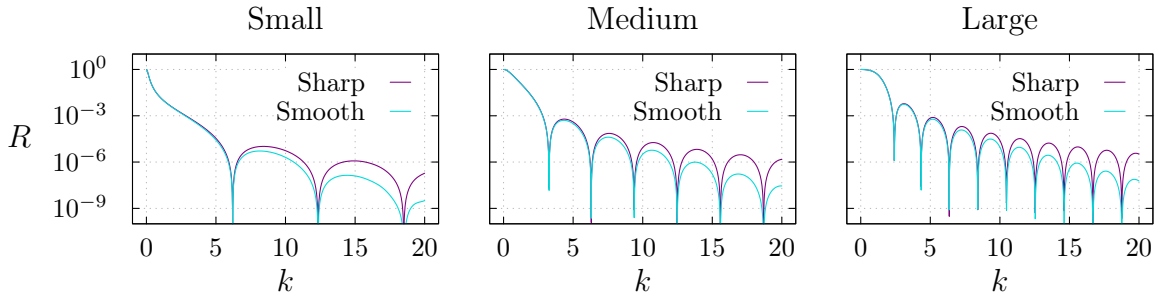

Figure 45: Reflection coefficients $R(k)$ of the six rectangular potentials. Note the logarithmic scale. In each plot the decay of R (with increasing momentum k) of the sharp potential is compared to the one of the corresponding smoothened potential. The former always decays slower than the latter. One can also identify the locations of resonant transmission ($R = 0 \iff T = 1$).

of k-values and the fact that the reflection amplitude converges slower to zero with increasing k if the potential is sharper (cf. Fig. 45).
Both the BA and the MA0's have zero mean (cf. Eqs. (153) and (39)).
The more interesting parts of the BAs are their respective real parts. They are symmetric just like the original potentials, which is why they are "pulled downwards" symmetrically (due to the zero mean property). The heights of the (smoothened) discontinuities however are in good agreement with the ones in the original potentials. As stated in Eq. (157), the real parts of the BAs are exactly the mean values of the respective MA0's.
The MA0's, in turn, are not symmetric and can therefore afford to reconstruct the potential quite well in a certain region until they are also eventually pulled downwards. This region is at the left / right side of the potential for the left / right MA. The heights of the discontinuities are approximately correct.
The MA1's have nonnegative mean (cf. Eq. (40)) and can thus correct for some of the errors of the zeroth order.
We now turn to a more quantitative analysis with the setups $\mathcal{S}^{0.01,20,0.01}_{-10,10,0.02}\,(0.5, 0.05, +/0)$, i.e. nonnegative and zero-mean random smooth potentials. The parameters are chosen such that the scattering is not very strong (as opposed to the classes considered in Subsecs. 3.4 and 5.2). In each case we work with 10^4 samples.
There is an issue with the localization length of strictly nonnegative potentials, which we briefly want to explain. Fig. 47 shows that the average potential of the class $\mathcal{S}^{0.01,20,0.01}_{-10,10,0.02}\,(0.5, 0.05, +)$ has a height of $\langle V\rangle = \frac{\sigma_{\text{pot}}}{\sqrt{2\pi}} \approx 0.2$ (cf. Eq. (71)). The mean logarithmic transmission coefficient $\langle \ln(T)\rangle$ is plotted as a function of the potential length L for $k = 3$ and for $k = 0.5$ in Fig. 48. There are very prominent oscillations in the case $k = 3$, which can be traced back to resonant effects: On average, the wave has the reduced momentum $\langle k\rangle = \sqrt{k^2 - 2\langle V\rangle}$ in the region of the potential. Now, $\langle \ln(T)\rangle$ for a given L is calculated by simply setting the potential to zero after a length of L. There is a resonance in transmission whenever the potential length L is an integer multiple of half of the wave's wavelength $\langle\lambda\rangle = \frac{2\pi}{\langle k\rangle}$:

$$L = n\frac{\langle\lambda\rangle}{2} = n\frac{\pi}{\langle k\rangle} = n\frac{\pi}{\sqrt{k^2 - 2\langle V\rangle}}. \tag{158}$$

For $k = 3$ the distance between the resonances is calculated to $\Delta L \approx 1.07$, which can

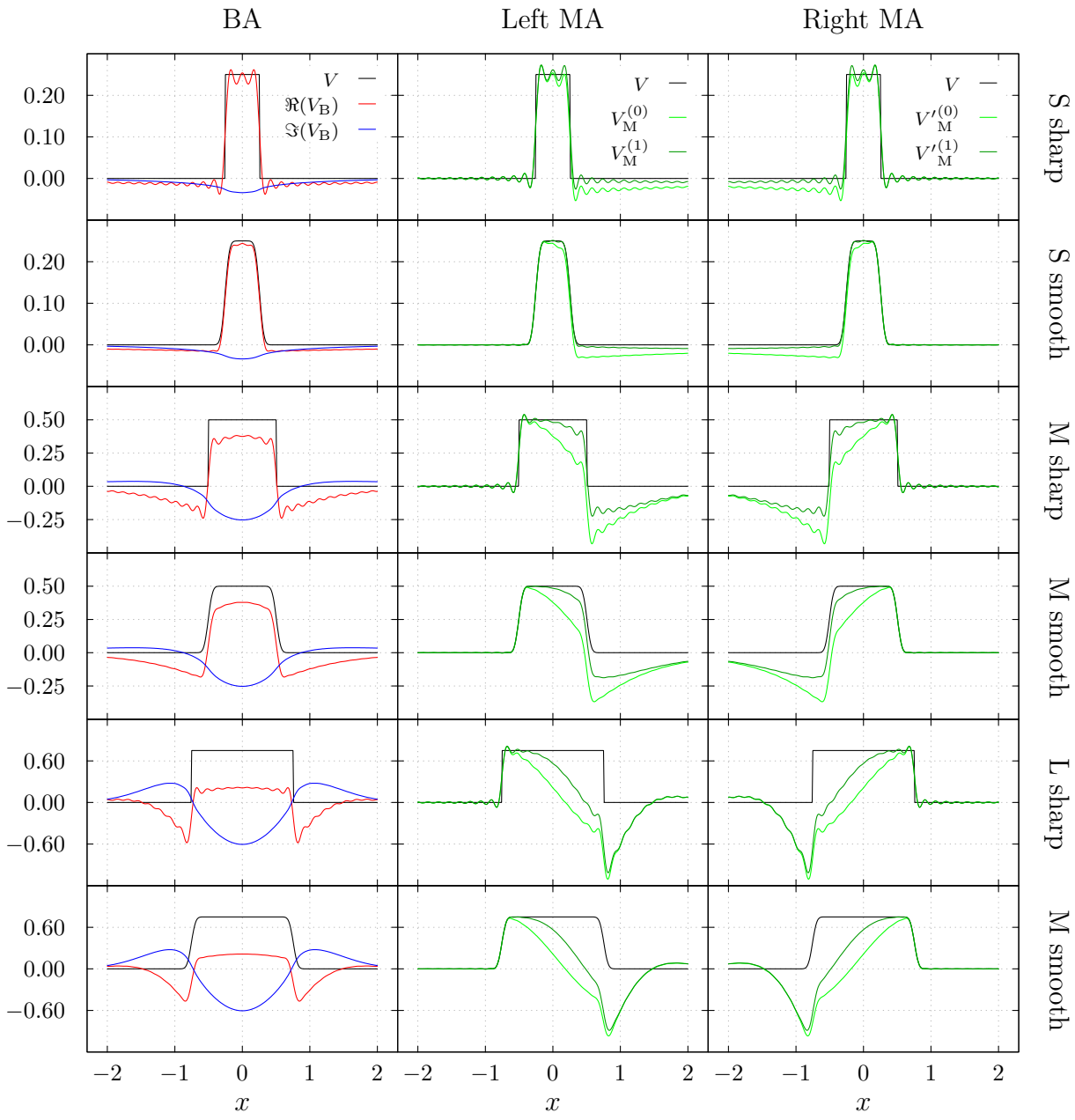

Figure 46: Comparison of the approximations to the original potentials. The rows display the different potentials, indicated by the label on the right side (S = Small, M = Medium, L = Large). The columns show the different approximations, indicated by the label on the top (BA = Born approximation, MA = Marčenko approximation). The BA is split into real and imaginary part. The zeroth and the first order of both MAs is shown. The keys are given only in the first row, they hold for all plots in the respective column. Mind the different scales of the vertical axes.

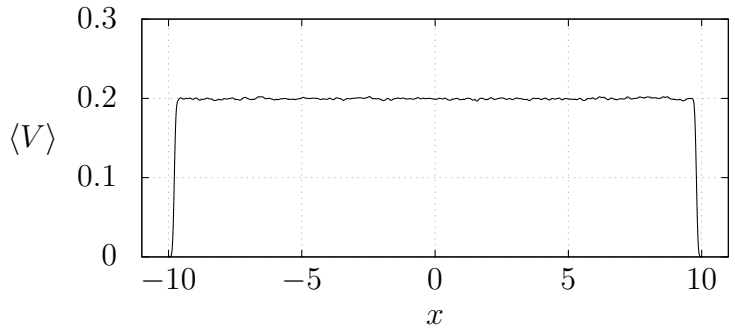

Figure 47: Average potential of the class $\mathcal{S}^{0.01,20,0.01}_{-10,10,0.02}(0.5, 0.05, +)$ calculated from 10^4 samples. The mean height of the potentials is approximately 0.2.

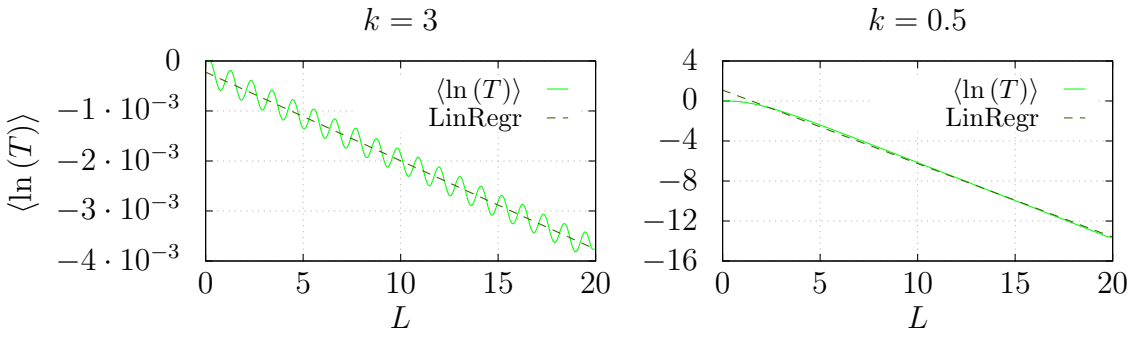

Figure 48: Average of the logarithm of the transmission coefficient $\langle \ln(T) \rangle$ depending on the potential length L for $k = 3$ and $k = 0.5$ for the class $\mathcal{S}^{0.01,20,0.01}_{-10,10,0.02}(0.5, 0.05, +)$, calculated from 10^4 samples. The linear regressions are shown with dashed lines. A striking feature of the curve for $k = 3$ are the oscillations.

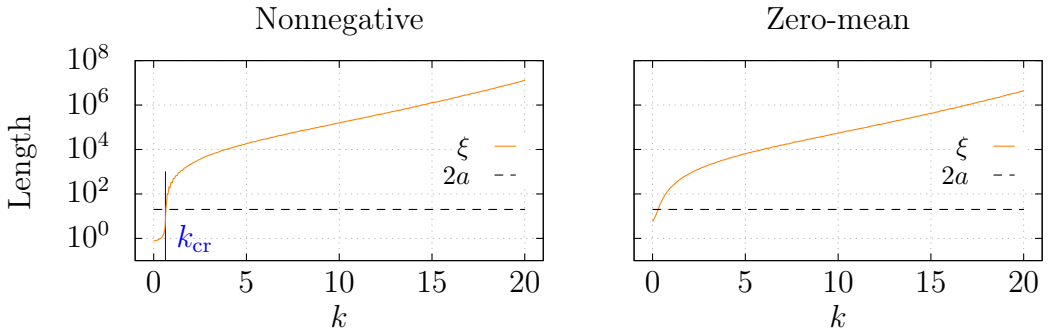

Figure 49: Momentum-dependence of the localization length ξ for the classes $\mathcal{S}^{0.01,20,0.01}_{-10,10,0.02}(0.5, 0.05, +)$ (left) and $\mathcal{S}^{0.01,20,0.01}_{-10,10,0.02}(0.5, 0.05, 0)$ (right), compared to the length of the potentials $2a = 20$. The blue line in the left plot indicates the critical momentum k_{cr} below which ξ does not describe localization, but the exponential decay of the wavefunction inside the classically forbidden region.

be verified in Fig. 48. Below the critical value of

$$k_{\mathrm{cr}} = \sqrt{2\langle V\rangle} = (2/\pi)^{1/4}\sqrt{\sigma_{\mathrm{pot}}}, \tag{159}$$

i.e. $k_{\mathrm{cr}} \approx 0.63$, the wave is (on average) exponentially suppressed as can be seen in Fig. 48 for $k = 0.5$. Despite the oscillations for $k > k_{\mathrm{cr}}$, we do a linear regression in order to obtain the localization length $\xi(k)$, just like in Subsec. 2.3. For $k < k_{\mathrm{cr}}$ one does not really obtain a localization length, but more a penetration depth into the classically forbidden region. The results are shown in the left part of Fig. 49. The blue line marks the critical momentum k_{cr}.

The effect just discussed is absent for zero-mean potentials, which are zero on (ensemble) average. The right part of Fig. 49 shows the localization length for the zero-mean potentials.

In both cases – nonnegative as well as zero-mean potentials – the scattering is ballistic nearly over the entire k-range, namely for $k \gtrsim 1$ (cf. Subsec. 2.3).

For the quantitative analysis we employ the x-average introduced in Eq. (119) to define

$$\mathrm{acc}\left[V_{\mathrm{A}}\right] := \left\langle \left|V_{\mathrm{A}}(x) - V(x)\right|\right\rangle_x, \tag{160}$$

where V_{A} is the approximated potential and V is the exact potential. From our qualitative analysis above we know that the left / right MA gives a good reconstruction in the left / right region of the potential. We can combine them into a weighted mean as

Table 6: Quantitative comparison of approximate inversion methods. The right MA0 $V'^{(0)}_{\text{Marč}}$ performs as good as the left MA0 $V^{(0)}_{\text{Marč}}$ and the right MA1 $V'^{(1)}_{\text{Marč}}$ performs as good as the left MA1 $V^{(1)}_{\text{Marč}}$, which is why the respective values are not listed here. The functions $W^{(\nu)}_{\text{Marč}}$ are weighted averages of the left and the right MAs (cf. Eq. (43)). The ensemble mean value $\langle\cdot\rangle$ is calculated using 10^4 samples. The best values are written in bold.

$\langle \text{acc}\,[V_\text{A}]\rangle$ Class / V_A	$\mathcal{S}^{0.01,20,0.01}_{-10,10,0.02}\,(0.5, 0.05, +)$	$\mathcal{S}^{0.01,20,0.01}_{-10,10,0.02}\,(0.5, 0.05, 0)$
$\Re\,(V_\text{Born})$	$1.96 \cdot 10^{-1}$	$2.54 \cdot 10^{-2}$
$V^{(0)}_{\text{Marč}}$	$1.97 \cdot 10^{-1}$	$2.62 \cdot 10^{-2}$
$V^{(1)}_{\text{Marč}}$	$1.86 \cdot 10^{-1}$	$2.56 \cdot 10^{-2}$
$W^{(0)}_{\text{Marč}}$	$1.93 \cdot 10^{-1}$	$2.44 \cdot 10^{-2}$
$W^{(1)}_{\text{Marč}}$	$\mathbf{1.74 \cdot 10^{-1}}$	$\mathbf{2.36 \cdot 10^{-2}}$

defined in Eq. (43). The computed ensemble averages of the accuracies are arranged in Table 6. The values for the right MAs are essentially the same as for the left ones.
We see that the approximations are in general more accurate if the potentials have zero mean (rightmost column), as opposed to strictly nonnegative potentials. This is explained by the fact that in the lowest orders all approximations have zero mean (cf. Eqs. (153) and (39)).
The weighted MA1 $W^{(1)}_{\text{Marč}}$ performs best in both cases.
Disregarding the weighted MAs, we see the following: The (real part of the) BA is more accurate than the MA0 in both cases. Although the MA1 is always better than the MA0, in surpasses the BA only for nonnegative potentials.

References

[1] van den Berg, P. & Abubakar, A. Inverse Scattering and its Applications to Medical Imaging and Subsurface Sensing. *URSI Radio Science Bulletin* **2002**, 13–26 (2002).

[2] Büyüköztürk, O. Imaging of concrete structures. *NDT&E International* **31**, 233–243 (1998).

[3] Weglein, A. B. *et al.* Inverse scattering series and seismic exploration. *Inverse Problems* **19**, R27–R83 (2003).

[4] de Hoop, M. V. Microlocal Analysis of Seismic Inverse Scattering. In Uhlmann, G. (ed.) *Inside Out: Inverse Problems and Applications*, vol. 47, 219–296 (Cambridge University Press, 2003).

[5] Mackintosh, R. S. Inverse scattering: applications to nuclear physics. `https://arxiv.org/abs/1205.0468v1` (2012).

[6] Müller, P., Schürmann, M. & Guck, J. The Theory of Diffraction Tomography. `https://arxiv.org/abs/1507.00466v3` (2016).

[7] Sun, Y., Xia, Z. & Kamilov, U. S. Efficient and accurate inversion of multiple scattering with deep learning. *Optics Express* **26**, 14678–14688 (2018).

[8] Kamilov, U. S., Liu, D., Mansour, H. & Boufounos, P. T. A Recursive Born Approach to Nonlinear Inverse Scattering. *IEEE Signal Processing Letters* **23**, 1052–1056 (2016).

[9] Wei, Z. & Chen, X. Deep-Learning Schemes for Full-Wave Nonlinear Inverse Scattering Problems. *IEEE Transactions on Geoscience and Remote Sensing* **57**, 1849–1860 (2019).

[10] Kamilov, U. S. *et al.* Learning approach to optical tomography. *Optica* **2**, 517–522 (2015).

[11] Almansouri, H., Venkatakrishnan, S., Buzzard, G., Bouman, C. & Santos-Villalobos, H. Deep neural networks for non-linear model-based ultrasound reconstruction. In *IEEE Global Conference on Signal and Information Processing*, 6–10 (2018).

[12] Li, Y., Xue, Y. & Tian, L. Deep speckle correlation: a deep learning approach toward scalable imaging through scattering media. *Optica* **5**, 1181–1190 (2018).

[13] Li, S., Deng, M., Lee, J., Sinha, A. & Barbastathis, G. Imaging through glass diffusers using densely connected convolutional networks. *Optica* **5**, 803–813 (2018).

[14] Peurifoy, J. *et al.* Nanophotonic particle simulation and inverse design using artificial neural networks. *Science Advances* **4** (2018).

[15] Liu, D., Tan, Y., Khoram, E. & Yu, Z. Training Deep Neural Networks for the Inverse Design of Nanophotonic Structures. *ACS Photonics* **5**, 1365–1369 (2018).

[16] Kay, I. The Inverse Scattering Problem. Research Report EM-74, New York University, Insitute of Mathematical Sciences, Division of Electromagnetic Research (1955).

[17] Deift, P. & Trubowitz, E. Inverse Scattering on the Line. *Communications on Pure and Applied Mathematics* **32**, 121–251 (1979).

[18] Aktosun, T. & Klaus, M. Inverse Theory: Problem on the Line. In Pike, R. & Sabatier, P. (eds.) *Scattering*, chap. 2.2.4, 770–785 (Academic Press, 2002).

[19] Kay, I. The Inverse Scattering Problem When the Reflection Coefficient is a Rational Function. *Communications on Pure and Applied Mathematics* **13**, 371–393 (1960).

[20] Ge, D. B. An iterative technique in one-dimensional profile inversion. *Inverse Problems* **3**, 399–406 (1987).

[21] Müller, C. A. & Delande, D. Disorder and interference: localization phenomena. `https://arxiv.org/abs/1005.0915v3` (2016).

[22] Dorren, H. J. S., Muyzert, E. J. & Snieder, R. K. The stability of one-dimensional inverse scattering. *Inverse Problems* **10**, 865–880 (1994).

[23] Szu, H. H., Carroll, C. E., Yang, C. C. & Ahn, S. A new functional equation in the plasma inverse problem and its analytic properties. *Journal of Mathematical Physics* **17**, 1236–1247 (1976).

[24] Brownstein, K. R. Criterion for existence of a bound state in one dimension. *American Journal of Physics* **68**, 160–161 (2000).

[25] Chollet, F. *Deep Learning with Python* (Manning Publications, 2018), 1st edn.

[26] Klambauer, G., Unterthiner, T., Mayr, A. & Hochreiter, S. Self-Normalizing Neural Networks. `https://arxiv.org/abs/1706.02515v5` (2017).

[27] Cybenko, G. Approximation by Superpositions of a Sigmoidal Function. *Math. Control Signal Systems* **2**, 303–314 (1989).

[28] Housing Values in Suburbs of Boston. URL `https://www.kaggle.com/c/boston-housing/overview`.

[29] Ruder, S. An overview of gradient descent optimization algorithms. `https://arxiv.org/abs/1609.04747v2` (2017).

[30] Gibbs, P. & Hiroshi, S. What is Occam's Razor? URL `http://math.ucr.edu/home/baez/physics/General/occam.html`.

[31] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* **15**, 1929–1958 (2014).

[32] Ioffe, S. & Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. `https://arxiv.org/abs/1502.03167v3` (2015).

[33] Perunicic, A. Understanding Neural Network Weight Initialization. URL `https://intoli.com/blog/neural-network-initialization/`.

[34] LeCun, Y., Bottou, L., Orr, G. B. & Müller, K.-R. Efficient BackProp. In Orr, G. B. & Müller, K.-R. (eds.) *Neural Networks: Tricks of the Trade*, chap. 1, 9–50 (Springer Berlin Heidelberg, 1998).

[35] Glorot, X. & Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, vol. 9, 249–256 (2010).

[36] He, K., Zhang, X., Ren, S. & Sun, J. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. `https://arxiv.org/abs/1502.01852v1` (2015).

[37] Pandey, A. Depth-wise Convolution and Depth-wise Separable Convolution. URL `https://medium.com/@zurister/depth-wise-convolution-and-depth-wise-separable-convolution-37346565d4ec`.

[38] Zeiler, M. D. & Fergus, R. Visualizing and Understanding Convolutional Networks. In Fleet, D., Pajdla, T., Schiele, B. & Tuytelaars, T. (eds.) *Computer Vision – ECCV 2014*, vol. 8689, 818–833 (Springer, Cham, 2014).

[39] Badrinarayanan, V., Kendall, A. & Cipolla, R. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **39**, 2481–2495 (2017).

[40] Zhang, R. *et al.* Real-Time User-Guided Image Colorization with Learned Deep Priors. *ACM Transactions on Graphics* **36**, 119:1–119:11 (2017).

[41] Vincent, P., Larochelle, H., Bengio, Y. & Manzagol, P.-A. Extracting and Composing Robust Features with Denoising Autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, 1096–1103 (2008).

[42] LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE* **86**, 2278–2324 (1998).

[43] Gatys, L. A., Ecker, A. S. & Bethge, M. Image Style Transfer Using Convolutional Neural Networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2414–2423 (2016).

[44] Hochreiter, J. *Untersuchungen zu dynamischen neuronalen Netzen.* Diplomarbeit, Technische Universität München (1991).

[45] Bengio, Y., Simard, P. & Frasconi, P. Learning Long-Term Dependencies with Gradient Descent is Difficult. *IEEE Transactions on Neural Networks* **5**, 157–166 (1994).

[46] Hochreiter, S. & Schmidhuber, J. Long Short-Term Memory. *Neural Computation* **9**, 1735–1780 (1997).

[47] Karpathy, A., Johnson, J. & Fei-Fei, L. Visualizing and Understanding Recurrent Networks. In *International Conference on Learning Representations* (2016).

[48] Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R. & Schmidhuber, J. LSTM: A Search Space Odyssey. https://arxiv.org/abs/1503.04069v1 (2015).

[49] Cho, K. *et al.* Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1724–1734 (Association for Computational Linguistics, 2014).

[50] Chung, J., Gulcehre, C., Cho, K. & Bengio, Y. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. https://arxiv.org/abs/1412.3555v1 (2014).

[51] Józefowicz, R., Zaremba, W. & Sutskever, I. An Empirical Exploration of Recurrent Network Architectures. In *Proceedings of the 32nd International Conference on Machine Learning*, 2342–2350 (2015).

[52] Karpathy, A. The Unreasonable Effectiveness of Recurrent Neural Networks. URL http://karpathy.github.io/2015/05/21/rnn-effectiveness/.

[53] He, K., Zhang, X., Ren, S. & Sun, J. Deep Residual Learning for Image Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 770–778 (2016).

[54] Ronneberger, O., Fischer, P. & Brox, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. In Navab, N., Hornegger, J., Wells, W. M. & Frangi, A. F. (eds.) *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, vol. 9351, 234–241 (Springer, LNCS, 2015).

[55] Das, S. CNN Architectures: LeNet, AlexNet, VGG, GoogLeNet, ResNet and more ... URL https://medium.com/@sidereal/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5.

[56] Abdi, H. & Williams, L. J. Principal component analysis. *WIREs Computational Statistics* **2**, 433–459 (2010).

[57] Goodfellow, I. *et al.* Generative Adversarial Nets. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D. & Weinberger, K. Q. (eds.) *Advances in Neural Information Processing Systems 27*, 2672–2680 (Curran Associates, Inc., 2014).

[58] Karras, T., Aila, T., Laine, S. & Lehtinen, J. Progressive Growing of GANs for Improved Quality, Stability, and Variation. In *International Conference on Learning Representations* (2018).

[59] Silver, D. *et al.* Mastering the game of Go with deep neural networks and tree search. *Nature* **529**, 484–489 (2016).

[60] Kingma, D. P. & Ba, J. L. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations* (2015).

[61] Dereziński, J. & Gérard, C. *Mathematics of Quantization and Quantum Fields*, chap. 18, 499 (Cambridge University Press, 2013), 1st edn.

Index