Computational Science and Engineering
(International Master's Program)

Technische Universität München

Master's Thesis

Neural Networks on Continuous-Variable Quantum Computers

Martin Knudsen

Computational Science and Engineering
(International Master's Program)

Technische Universität München

Master's Thesis

Neural Networks on Continuous-Variable Quantum Computers

Author:	Martin Knudsen
Examiner:	Prof. Dr. rer. nat. Christian Mendl
Submission Date:	October 15, 2020

I hereby declare that this thesis is entirely the result of my own work except where otherwise indicated. I have only used the resources given in the list of references.

October 15, 2020 Martin Knudsen

Acknowledgments

First and foremost, I wish to thank my supervisor, Prof. Christian Mendl.

I am very grateful for the opportunity to write my thesis in his group. Prof. Mendl had no prior knowledge of me (except for a somewhat desperate email), and he immediately accepted my request to write my thesis in his group. He suggested, that I should look at the continuous-variable approach, which has been a very fruitful and interesting endeavor. I was allowed to freely choose my exact subject, but he answered my questions and gave me support whenever I needed it. He gave me valuable advice as to which problems were interesting and what literature was relevant. Overall, the experience of working in the group was a very pleasant one.

Finally, I want to thank the people at Xanadu for creating their outstanding software and documentation, without which this thesis would have been much harder to complete.

"To me quantum computation is a new and deeper and better way to understand the laws of physics, and hence understanding physical reality as a whole."

-David Deutsch

Abstract

In this work, variational circuits on a continuous-variable (CV) quantum computer are simulated using the PennyLane Framework and used to solve several classic machine learning tasks. Utilizing the CV approach, it is possible to directly encode real numbers into each mode, which is an advantage for more complicated architectures. The necessary background theory in quantum optics and CV quantum computing is presented and used to deduce how neural network inspired architectures can be realized. The parameter shift rule for different gates is deduced and tested with the PennyLane framework. Non-linear 1D regression and 2D function approximation was successfully achieved with a 1 mode and a 2 mode architecture, respectively. Simple classification was performed on the Iris flower dataset to a test-accuracy of 70 %. A 1D linear ordinary differential equation was solved using a three mode architecture and a non-linear ordinary differential equation was solved using a 2 mode architecture. Finally, an idea for the practical implementation of a convolutional neural network on a CV quantum computer building on previous results is presented.

Contents

Part I.

Introduction and Background Theory

1. Introduction

Quantum computing is an ongoing and active field of research, which shows great promise, if it can be efficiently implemented. The constituent of a traditional quantum computer is the *qubit* and one of the strength of a quantum computer is the exponential growth in computing power with the number of qubits [32]. This approach, however, contains serious problems in implementation, especially quantum computing today is error prone, and they have to rely on error codes, meaning it takes several physical qubits to implement a logical qubit, even thousands [22]. Furthermore, many quantum computing paradigms need a sufficiently low temperature to function properly, such as is the case for the ion-trap quantum computer [32, p. 312]. This means a big overhead in terms of size, energy consumption and price.

An interesting novel approach to quantum computing is to switch from a digital qubit domain to a continuous domain [30]. Now, the observables, instead of being 2-state qubits, are represented by continuous variables (CV) such as position or momentum. This approach is interesting because it can be realized by simple optical instruments, which mostly operate at room temperature and is more error resistant than a conventional quantum computer. This is partly due to the fact that the quantum information is carried in light, which is charge-less and therefore does not react that much with the environment [32, p. 287]. Another advantage is the direct representation of real numbers in the input, which needs several qubits to be simulated on a qubit quantum computer. In fact there are some systems which cannot be accurately simulated on a digital computer and in that case this analogue paradigm might prove powerful [15].

One of the most interesting applications of quantum computing is in the field of machine learning. Either as a part of a machine learning method (e.g. optimization) or, as in this thesis, directly embedded on a quantum computer, quantum machine learning is an active area of research. Recently, a way to implement neural networks on a CV quantum computer has been introduced. This work, by *Killoran et al.* [26], was used as a basis for this thesis and several machine learning tasks were solved by similar architectures.

The experiments were simulated using the PennyLane framework from company Xanadu [14].

Notation

Different values of $\hbar$ are set in different part of the work to simplify expressions. In quantum optics, unless explicitly included, $\hbar = 1$. In Quantum the quantum computing sections, the PennyLane convention of using $\hbar = 2$ is followed. The quadrature operators of the quantum electromagnetic oscillator are set to $\hat{x}$ and $\hat{p}$ to underline their similarity to the position and momentum operator respectively. This might overload the x symbol because the x is then used both as a function argument in $f(x)$, an operator $\hat{x}$ and its eigen pairs. Furthermore, whenever *neural networks* are mentioned, *artificial* neural networks are meant and not biological ones. Additionally, the operator $\hat{a}$ will be called the annihilation operator throughout the text, but is sometimes called amplitude, mode or ladder operators in the literature [29] [24]. The modes themselves from the mode expansion are in this work referred to as *modes* and not *qumodes* as in some literature [26]. The phase shift gate, rotating coherent states, is called phase shift gate in this work and a positive phase shift ϕ corresponds to a clockwise rotation as is custom in quantum optics [29, p. 40]. The PennyLane framework uses the opposite convention, but this shouldn't be an issue as the difference can be learned by the machine learning models.

2. Quantum optics

To understand, how optical continuous-variable (CV) quantum computing works, some background in quantum optics is needed. First, the quantum vector potential is introduced and used to derive the mode expansion and the quantum harmonic oscillator view of light.
The quantum gates used in the optical CV paradigm will be introduced and their effect, physical implementation and derivative will be shown. Ultimately, a neural network inspired architecture will be presented.
Quantum Optics from Leonhardt [29] was used extensively for this chapter.

2.1. Maxwell's equations and the quantum vector potential

Classically, light obeys Maxwell's equations, that look like [29, p. 15]

$$\nabla \cdot \mathbf{B} = 0, \quad \nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t}, \quad \nabla \cdot \mathbf{D} = 0, \quad \nabla \times \mathbf{H} = \frac{\partial \mathbf{D}}{\partial t}, \tag{2.1}$$

where $\mathbf{B}$ is the magnetic induction, $\mathbf{E}$ is the electric field, $\mathbf{D}$ is the electric displacement and $\mathbf{H}$ is the magnetic field. In order to translate these equations to the quantum realm, the assumption is made, that the classical fields are expectation values of the underlying quantum fields, so that for all fields the operator version becomes [29, p. 15]

$$\mathbf{F} = \langle \psi | \hat{\mathbf{F}} | \psi \rangle \,, \tag{2.2}$$

where $|\psi\rangle$ is the state-vector of the system or an ensemble if the state is mixed. Maxwell's equations are linear, because the divergence, curl and partial derivative are all linear operations. Because of the linearity of Maxwell's equations, the quantum version of Maxwell's equation are identical, but now the fields are operators

$$\nabla \cdot \hat{\mathbf{B}} = 0, \quad \nabla \times \hat{\mathbf{E}} = -\frac{\partial \hat{\mathbf{B}}}{\partial t}, \quad \nabla \cdot \hat{\mathbf{D}} = 0, \quad \nabla \times \hat{\mathbf{H}} = \frac{\partial \hat{\mathbf{D}}}{\partial t}, \tag{2.3}$$

where each field operator is a 3-dimensional vector of each component operator:

$$\hat{\mathbf{F}} = \begin{bmatrix} \hat{F}_x \\ \hat{F}_y \\ \hat{F}_z \end{bmatrix} . \tag{2.4}$$

These equations can be simplified for the case where the magnetic induction and electric displacement are proportional to the magnetic field and the electric field respectively. This approximation is a good one for non-absorptive, non-dispersive and isotropic media, which is a good approximation fin this case [29, p. 15]. This assumption leads to the so-called constitutive equations [29, p. 15]

$$\hat{\mathbf{D}} = \varepsilon_0 \varepsilon \hat{\mathbf{E}}, \quad \hat{\mathbf{B}} = \mu_0 \mu \hat{\mathbf{H}}, \quad \varepsilon_0 \mu_0 = \frac{1}{c^2}, \tag{2.5}$$

where ε_0, μ_0 and c are constants and ε and μ depend on the material. Again, because these equations are linear, the quantum version takes the same form as the classical version. Equations (2.3) can now be rewritten as

$$\nabla \cdot \hat{\mathbf{B}} = 0, \quad \nabla \times \hat{\mathbf{E}} = -\frac{\partial \hat{\mathbf{B}}}{\partial t}, \quad \varepsilon_0 \varepsilon \nabla \cdot \hat{\mathbf{E}} = 0, \quad \frac{c^2}{\mu\varepsilon} \nabla \times \hat{\mathbf{B}} = \frac{\partial \hat{\mathbf{E}}}{\partial t}. \tag{2.6}$$

These equations can be expressed in terms of a vector potential as [29, p. 16]

$$\hat{\mathbf{E}} = -\frac{\partial \hat{\mathbf{A}}}{\partial t}, \quad \hat{\mathbf{B}} = \nabla \times \hat{\mathbf{A}}. \tag{2.7}$$

Because the divergence of curl is zero, the first equation is satisfied. The second is also satisfied as can be seen by substituting the definition of $\hat{\mathbf{E}}$ and $\hat{\mathbf{B}}$ into equation (2.6). Requiring the Coulomb gauge

$$\nabla \varepsilon \cdot \hat{\mathbf{A}} = 0, \tag{2.8}$$

the third equation is satisfied and only the fourth equation remains. Substituting $\hat{\mathbf{A}}$ in to the third, one obtains

$$\begin{gathered} \frac{c^2}{\mu\varepsilon} \nabla \times \nabla \times \hat{\mathbf{A}} = -\frac{\partial^2 \hat{\mathbf{A}}}{\partial t^2} \\ \frac{1}{\mu\varepsilon} \nabla \times \nabla \times \hat{\mathbf{A}} + \frac{1}{c^2} \frac{\partial^2 \hat{\mathbf{A}}}{\partial t^2} = 0, \end{gathered} \tag{2.9}$$

which is the electromagnetic wave equation [29, p. 16]. Therefore, this equation together with the Coulomb gauge and assuming the constitutive equations (2.5) completely describes how the quantum vector potential evolves.

The Hamiltonian of light is assumed to be the same as the classical version but with operators [29, p. 17]

$$\hat{H} = \frac{1}{2} \int \left(\hat{\mathbf{E}} \cdot \hat{\mathbf{D}} + \hat{\mathbf{B}} \cdot \hat{\mathbf{H}} \right) \mathrm{d}V. \tag{2.10}$$

2.2. Light modes

Having defined the vector potential in terms of the electromagnetic wave equation, the next question becomes how to represent it. One interesting way to do this is through so-called light modes. The idea is to expand the electromagnetic potential in terms of its classical and quantum components

$$\hat{\mathbf{A}}\left(\mathbf{r},t\right)=\sum_k\left(\mathbf{A}_k\left(\mathbf{r},t\right)\hat{a}_k+\mathbf{A}_k^*\left(\mathbf{r},t\right)\hat{a}_k^\dagger\right), \tag{2.11}$$

where $\mathbf{A}_k\left(\mathbf{r},t\right)$ are the classical complex wave functions, called modes, and the $\hat{a}_k$ are operator coefficients, called annihilation operators, carrying the quantum degrees of freedom and symbolizing the quantum equivalent to the classical amplitude of a light wave. The second term is the complex conjugate of the first term.

A scalar product between the different modes can be defined as [29, p. 21]

$$\langle\mathbf{A}_i|\mathbf{A}_j\rangle=-\frac{\varepsilon_0\varepsilon}{i\hbar}\int\left(\mathbf{A}_i^*\cdot\frac{\partial\mathbf{A}_j}{\partial t}-\mathbf{A}_j\cdot\frac{\partial\mathbf{A}_i^*}{\partial t}\right)\mathrm{d}V. \tag{2.12}$$

This scalar product fulfills the typical properties of a scalar product such as conjugate symmetry:

$$\begin{aligned}\langle\mathbf{A}_i|\mathbf{A}_j\rangle^*&=\frac{\varepsilon_0\varepsilon}{i\hbar}\int\left(\mathbf{A}_i\cdot\frac{\partial\mathbf{A}_j^*}{\partial t}-\mathbf{A}_j^*\cdot\frac{\partial\mathbf{A}_i}{\partial t}\right)\mathrm{d}V\\&=\langle\mathbf{A}_j|\mathbf{A}_i\rangle,\end{aligned} \tag{2.13}$$

as well as stability over scalar multiplication and addition (linearity), which is due to the integration and derivative operators both being linear.

There are many potential waves that could satisfy the expansion, but in this work, the convention for normal modes is used [29, p. 23]

$$\langle\mathbf{A}_i|\mathbf{A}_j\rangle=\delta_{ij}\quad\langle\mathbf{A}_i|\mathbf{A}_j^*\rangle=0. \tag{2.14}$$

With these conditions the Bose commutation relations can be derived, which are offered here without proof [29, p. 23]

$$[\hat{a}_i,\hat{a}_j^\dagger]=\delta_{ij},\quad[\hat{a}_i,\hat{a}_j]=0. \tag{2.15}$$

An immediate consequence of these relations, is that two different annihilation operators commute and hence different light modes represent distinct physical systems [29, p. 23].

Interesting in this work, are monochromatic modes that only oscillates at single frequencies ω_k [29, p. 25]

$$\mathbf{A}_k(\mathbf{r},t)=\mathbf{A}_k(\mathbf{r})e^{-i\omega_k t}. \tag{2.16}$$

As derived in section A.1, the light Hamiltonian for monochromatic modes can be written as a sum of harmonic oscillators

$$\hat{H} = \sum_k \omega_k \hbar \left(\frac{1}{2} + \hat{a}_k^\dagger \hat{a}_k \right). \tag{2.17}$$

This is a very important result for several reasons. Firstly, here the so-called *zero-point energy* is encountered, which means that even in the ground state, space still contains the energy $E = \sum_k \frac{\omega_k \hbar}{2}$ [29, p. 26]. Secondly, this Hamiltonian exactly has the shape of a sum of quantum harmonic oscillators [24, p. 44]. The term $\hat{a}_k^\dagger \hat{a}_k$ is Hermitian, and so must represent an observable. In fact, the Hamiltonian consists only of these operators and a constant, so the allowed energies are sums of the eigenvalues of these operators. Because this is the Hamiltonian of light, these operators are called *photon number* operators. They represent the number of photons in a mode and are defined by [29, p. 38]

$$\hat{n} = \hat{a}^\dagger \hat{a}. \tag{2.18}$$

Using the annihilation operator, one can define the important quadrature operators [29, p. 38]

$$\hat{x} = \frac{1}{\sqrt{2}} \left(\hat{a}^\dagger + \hat{a} \right), \quad \hat{p} = \frac{i}{\sqrt{2}} \left(\hat{a}^\dagger - \hat{a} \right), \tag{2.19}$$

which can be seen as the real and imaginary part of the annihilation operator

$$\hat{a} = \frac{1}{\sqrt{2}} \left(\hat{x} + i\hat{p} \right). \tag{2.20}$$

By using the definition (2.19), one can see that they are canonically conjugate [29, p. 38]

$$[\hat{x}, \hat{p}] = i. \tag{2.21}$$

Therefore, $\hat{x}$ and $\hat{p}$ can be regarded as the position and momentum, respectively. It follows, that the momentum can also be written as $\hat{p} = -i\frac{\partial}{\partial x}$, which leads to a different way to write the annihilation operator (a similar result can be achieved for the momentum operator)

$$\hat{a} = \frac{1}{\sqrt{2}} \left(\hat{x} + \frac{\partial}{\partial x} \right). \tag{2.22}$$

Because the energy of a single mode is $\hat{E} = \hat{n} + \frac{1}{2}$ according to equation (2.17), the Hamiltonian can also be written in terms of the quadrature operators as (for unit frequency and $\hbar = 1$)

$$\begin{aligned}\hat{H} &= \hat{a}^\dagger\hat{a} + \frac{1}{2} \\ &= \frac{1}{2}\left(\hat{x}^2 + \hat{p}^2 + i\left(\hat{x}\hat{p} - \hat{p}\hat{x}\right)\right) + \frac{1}{2} \\ &= \frac{1}{2}\left(\hat{x}^2 + \hat{p}^2 - 1\right) + \frac{1}{2} \\ &= \frac{\hat{x}^2}{2} + \frac{\hat{p}^2}{2},\end{aligned} \tag{2.23}$$

where the relation (2.21) was used.

2.3. Fock states

The eigenstates of the photon number operator $\hat{n}$ are called Fock states and have corresponding eigenvalues [29, p. 41]

$$\hat{n}\left|n\right\rangle = n\left|n\right\rangle. \tag{2.24}$$

This means e.g. that the eigenvalue of the Fock state $\left|1\right\rangle$ is $n = 1$, corresponding to the number of photons in that state. The Fock states are orthonormal because each state corresponds to a unique eigenvalue and are assumed to be normalized. The annihilation operator has the effect of decreasing the photon number of a Fock state, which explains the name [29, p. 42]

$$\begin{aligned}\hat{n}\hat{a}\left|n\right\rangle &= \left(\hat{a}^\dagger\hat{a}\right)\hat{a}\left|n\right\rangle \\ &= \left(\hat{a}\hat{a}^\dagger - 1\right)\hat{a}\left|n\right\rangle \\ &= \hat{a}\left(\hat{n} - 1\right)\left|n\right\rangle \\ &= \left(n - 1\right)\hat{a}\left|n\right\rangle,\end{aligned} \tag{2.25}$$

where the third step uses the commutation relation (2.15). From this, one can deduce that $\hat{a}\left|n\right\rangle = c\left|n-1\right\rangle$ for some normalization constant c. Taking the inner product of the annihilated state gives

$$\begin{aligned}\langle n|\hat{a}^\dagger\hat{a}|n\rangle &= n\left\langle n|n\right\rangle \\ &= n,\end{aligned} \tag{2.26}$$

where the last equality follows from the Fock states being orthonormal. Therefore, the effect of the annihilation operator including normalization is

$$\hat{a}\left|n\right\rangle = \sqrt{n}\left|n-1\right\rangle. \tag{2.27}$$

Similarly, the conjugate transposed annihilation operator $\hat{a}^\dagger$ has the effect of increasing the photon number and is therefore called the creation operator. Its effect including normalization is [29, p. 42]

$$\hat{a}^\dagger \left|n\right\rangle = \sqrt{n+1}\left|n+1\right\rangle . \tag{2.28}$$

Using the annihilation operator, one could in principle "empty" out any state until it becomes negative. If the annihilation operator is applied to a negative state $\hat{a}\left|-0.1\right\rangle = -\sqrt{0.1}\left|-1.1\right\rangle$, taking the modulus on both sides leads to a contradiction. Therefore, neither non-integer, nor negative states are allowed.

For the ground state, $\psi_0\left(x\right)$, the eigen equation becomes

$$\hat{n}\psi_0\left(x\right) = 0 \tag{2.29}$$

This is fulfilled if either $\hat{a}\psi_0\left(x\right) = 0$ or $\hat{a}^\dagger\hat{a}\psi_0\left(x\right) = 0$ [29, p. 42]. Looking at the first option and using equation (2.22) leads to the expression

$$\begin{aligned}\frac{1}{\sqrt{2}}\left(x + \frac{\partial}{\partial x}\right)\psi_0 &= 0\\ \frac{\partial\psi_0}{\partial x} &= -x\psi_0,\end{aligned} \tag{2.30}$$

where $\hat{x}$ seizes to be an operator, because it is multiplied with its eigenstate. This equation can be solved by separation of variables

$$\begin{aligned}\int \frac{1}{\psi_0}\,\mathrm{d}\psi_0 &= -\int x\,\mathrm{d}x\\ \ln\psi_0 &= -\frac{1}{2}x^2 + c_1\\ \psi_0\left(x\right) &= c_2 e^{-\frac{1}{2}x^2},\end{aligned} \tag{2.31}$$

where c_1 and c_2 are constants that come from the indefinite integration. c_2 can be found by normalization

$$\begin{aligned}1 &= c_2^2\int_{-\infty}^{\infty} e^{-x^2}\,\mathrm{d}x\\ 1 &= c_2^2\pi^{1/2}\\ c_2 &= \pi^{-\frac{1}{4}},\end{aligned} \tag{2.32}$$

where the Gaussian integral $\int_{-\infty}^{\infty} e^{-x^2}\,\mathrm{d}x = \pi^{\frac{1}{2}}$ was used [42, p. 46]. The positive normalization is used, since the probability distribution ψ_0 per definition can't be negative. The ground state including normalization is therefore

$$\psi_0\left(x\right) = \pi^{-\frac{1}{4}} e^{-\frac{1}{2}x^2}. \tag{2.33}$$

Using equation (2.27), a relation between the ground state and the excited states can be found

$$\begin{aligned} n\left|n\right\rangle &= \sqrt{n}\hat{a}^\dagger \left|n-1\right\rangle \\ \left|n\right\rangle &= \frac{\hat{a}^\dagger}{\sqrt{n}} \left|n-1\right\rangle \\ \left|n\right\rangle &= \frac{\hat{a}^{\dagger n}}{\sqrt{n!}} \left|0\right\rangle, \end{aligned} \tag{2.34}$$

where the last step can be seen by recursively substituting the Fock states.
Combining equation (2.34) and (2.33), Fock states can be derived to be [29, p. 43]

$$\psi_n\left(x\right) = \frac{H_n\left(x\right)}{\sqrt{2^n n! \sqrt{\pi}}} e^{-\frac{x^2}{2}}, \tag{2.35}$$

where $H_n\left(x\right)$ is the nth Hermite polynomial. Important for later calculations is, that these states are not Gaussian.

2.4. Coherent states

Coherent states describe laser light. Therefore, these states have a well-defined amplitude and are eigenstates of the annihilation operator [29, p. 48]

$$\hat{a}\left|a\right\rangle = \alpha\left|a\right\rangle, \tag{2.36}$$

where α is the complex wave amplitude of a mode. By setting $\alpha = 0$, the same equation as for the Fock states (2.29) is achieved, which means they have the same ground state (2.33). Other coherent states can be achieved by applying a displacement operator to the ground state as defined in section 3.2 [29, p. 50]

$$\left|\alpha\right\rangle = \hat{D}(\alpha)\left|0\right\rangle. \tag{2.37}$$

Because the displacement gate is a Gaussian gate, meaning it transforms a Gaussian state to another Gaussian state [26], coherent states are all Gaussian. Coherent states can be represented by a Poissonian sum of Fock states [29, p. 52]

$$\left|\alpha\right\rangle = e^{-\frac{1}{2}|\alpha|^2} \sum_{n=0}^{\infty} \frac{\alpha^n}{\sqrt{n!}} \left|n\right\rangle. \tag{2.38}$$

2.5. Quadrature states and phase space

Yet another way to look at light, is in the so-called phase space picture [26]. The quadrature operators also have eigenstates, that are however not directly observable [29, p. 41]

$$\begin{aligned} \hat{x}\,|x\rangle &= x\,|x\rangle \\ \hat{p}\,|p\rangle &= p\,|p\rangle\,. \end{aligned} \tag{2.39}$$

The eigenvalue x is what is being measured in homodyne detection as explained in section 4.6. The quadratures can be organized in a vector, which is useful in architectures with more than one mode

$$\begin{bmatrix} \hat{\mathbf{x}} \\ \hat{\mathbf{p}} \end{bmatrix} \tag{2.40}$$

The *phase space* formulation of quantum mechanics tracks the evolution of these components [26]. This is mostly the approach, that will be taken in this work. To get an intuitive idea of what the CV gates defined in section 3 do, it often helps to look at their action in phase space. In figure 2.1 phase space is illustrated with different measurements of the quadrature values of identically prepared systems with expectation value close to $x = 1$ and $p = 1$. The underlying probability distributions (quasi) can be described by *Wigner* functions and many of the gates used have simple representations of those functions [29, p. 73], but this is outside the scope of this work.

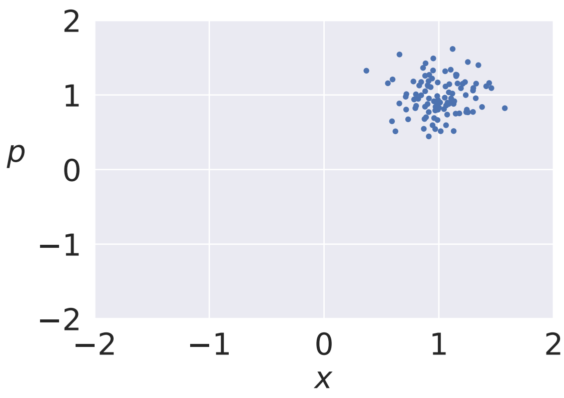


Figure 2.1.: Phase space representation of a mode adapted from [29, p. 39].

3. Optical continuous-variable quantum computation

The gates used in this work can all be achieved by simple optical instruments that are well known and studied in quantum optics. The single mode gates are the phase, displacement, squeeze and Kerr gate. The basic two-mode gate is the beam splitter. These gates are all unitary in the Schrödinger picture, but can produce non-unitary (and nonlinear for the Kerr gate) effects in phase space [26]. In fact this represents a universal gate set, in the sense that any polynomial transformation of arbitrary degree in phase space on the quadrature operators can be achieved [26].

As the effect of these gates are most easily seen on the quadrature and annihilation operators, it is often easier to use the Heisenberg and phase space pictures instead of the Schrödinger picture of quantum mechanics.

3.1. Phase gate

The phase gate changes the phase of a single annihilation operator (mode) in the Heisenberg picture and thereby leads to a rotation in phase space.

It is defined by [29, p. 38]

$$\hat{U}\left(\phi\right) = e^{-i\phi\hat{n}}. \tag{3.1}$$

The effect of the phase operator on the annihilation operator in the Heisenberg picture is

$$\hat{U}^{\dagger}\left(\phi\right)\hat{a}\hat{U}\left(\phi\right) = e^{-i\phi}\hat{a}, \tag{3.2}$$

which can be seen by differentiation [29, p. 38]

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}\phi}\hat{U}^{\dagger}\hat{a}\hat{U} &= i\hat{n}\hat{U}^{\dagger}\hat{a}\hat{U} - i\hat{U}^{\dagger}\hat{a}\hat{U}\hat{n} \\ &= i\hat{U}^{\dagger}\left(\hat{n}\hat{a} - \hat{a}\hat{n}\right)\hat{U} \\ &= i\hat{U}^{\dagger}\left(\hat{n} - \hat{n}^{\dagger}\right)\hat{a}\hat{U} \\ &= i\hat{U}^{\dagger}\left[\hat{a}^{\dagger}, \hat{a}\right]\hat{a}\hat{U} \\ &= -i\hat{U}^{\dagger}\hat{a}\hat{U},\end{aligned} \tag{3.3}$$

using the Bose commutation relation (2.15). Because (3.1) also fulfills the same differential equation and for $\phi = 0$ both sides of (3.2) reduce to $\hat{a}$, (3.1) is the solution.

The effect on the creation operator can be found by taking the hermitian conjugate of (3.2) and using that the phase gate is unitary

$$\hat{U}^\dagger(\phi)\,\hat{a}^\dagger\hat{U}(\phi) = e^{i\phi}\hat{a}^\dagger. \tag{3.4}$$

Because of unitarity, its effect on the square of the annihilation operator is easily derived to be

$$\begin{aligned}\hat{U}^\dagger(\phi)\,\hat{a}^2\hat{U}(\phi) &= \hat{U}^\dagger(\phi)\,\hat{a}\hat{U}(\phi)\,\hat{U}(\phi)^\dagger\,\hat{a}\hat{U}(\phi)\\ &= e^{-2i\phi}\hat{a}^2\end{aligned} \tag{3.5}$$

The effect of this operator on the position quadrature is

$$\begin{aligned}\hat{x}_\phi &= \hat{U}^\dagger(\phi)\,\hat{x}\hat{U}(\phi)\\ &= \frac{1}{\sqrt{2}}\hat{U}^\dagger(\phi)\left(\hat{a}^\dagger + \hat{a}\right)\hat{U}(\phi)\\ &= \frac{1}{\sqrt{2}}\left(\hat{a}^\dagger e^{i\phi} + \hat{a}e^{-i\phi}\right)\\ &= \frac{1}{2}\left(e^{i\phi}(\hat{x} - i\hat{p}) + e^{-i\phi}(\hat{x} + i\hat{p})\right)\\ &= \cos(\phi)\,\hat{x} + \sin(\phi)\,\hat{p}.\end{aligned} \tag{3.6}$$

The effect on the $\hat{p}$ can be found similarly and is [29, p. 40]

$$\hat{p}_\phi = -\sin(\phi)\,\hat{x} + \cos(\phi)\,\hat{p}. \tag{3.7}$$

The effect on the quadratures can therefore be summarized by a rotation

$$U(\phi): \quad \begin{bmatrix}\hat{x}_\phi\\ \hat{p}_\phi\end{bmatrix} = \begin{bmatrix}\cos(\phi) & \sin(\phi)\\ -\sin(\phi) & \cos(\phi)\end{bmatrix}\begin{bmatrix}\hat{x}\\ \hat{p}\end{bmatrix}. \tag{3.8}$$

3.2. Displacement gate

The displacement operator can be described by [29, p. 49]

$$\hat{D}(\alpha) = e^{\alpha\hat{a}^\dagger - \alpha^*\hat{a}}. \tag{3.9}$$

Its effect on the annihilation operator is given by [29, p. 49]

$$\hat{D}(\alpha)^\dagger\hat{a}\hat{D}(\alpha) = \hat{a} + \alpha. \tag{3.10}$$

The effect on the creation operator can be found by taking the hermitian conjugate

$$\hat{D}(\alpha)^{\dagger}\hat{a}^{\dagger}\hat{D}(\alpha) = \hat{a}^{\dagger} + \alpha^{*}. \tag{3.11}$$

The effect on the $\hat{x}$ quadrature is

$$\begin{aligned}
\hat{D}(\alpha)^{\dagger}\hat{x}\hat{D}(\alpha) &= \frac{1}{2}\left(\hat{D}(\alpha)^{\dagger}(\hat{x}+i\hat{p}+\hat{x}-i\hat{p})\hat{D}(\alpha)\right) \\
&= \frac{1}{\sqrt{2}}\left(\hat{D}(\alpha)^{\dagger}\hat{a}\hat{D}(\alpha)+\hat{D}(\alpha)^{\dagger}\hat{a}^{\dagger}\hat{D}(\alpha)\right) \\
&= \frac{1}{\sqrt{2}}(\hat{a}+\alpha+\hat{a}\dagger+\alpha^{*}) \\
&= \frac{1}{\sqrt{2}}(\hat{a}+\hat{a}\dagger)+\sqrt{2}\,\mathrm{Re}(\alpha) \\
&= \hat{x}+\sqrt{2}\,\mathrm{Re}(\alpha)
\end{aligned} \tag{3.12}$$

Similarly, the effect on $\hat{p}$ can be found to be [26]

$$\hat{D}(\alpha)^{\dagger}\hat{p}\hat{D}(\alpha) = \hat{p} + \sqrt{2}\,\mathrm{Im}(\alpha). \tag{3.13}$$

Taking the expectation value of equation (3.12) in the $|x\rangle$ state, the expected outcome of a measurement is

$$\langle x|\hat{D}(\alpha)^{\dagger}\hat{x}\hat{D}(\alpha)|x\rangle = x + 2\,\mathrm{Re}(\alpha), \tag{3.14}$$

For the quadrature ground state $|0\rangle$ this becomes

$$\langle 0|\hat{D}(\alpha)^{\dagger}\hat{x}\hat{D}(\alpha)|0\rangle = 2\,\mathrm{Re}(\alpha), \tag{3.15}$$

which means that if one wants to embed the value $x = 1$ starting in the ground state, this is equivalent of using $\alpha = 0.5$. Therefore, one can talk about embedding a real value x in $|x\rangle$ when in reality one is embedding α in the coherent state. In fact, in the experiments later conducted x is always directly embedded in the coherent state, which leads to an error of factor 2, that nonetheless doesn't matter, cause the parametric circuits will just learn this scaling.

The effect on the quadratures can be summarized by

$$D(\alpha): \quad \begin{bmatrix}\hat{x}_{\alpha}\\ \hat{p}_{\alpha}\end{bmatrix} = \begin{bmatrix}\hat{x}+\sqrt{2}\,\mathrm{Re}(\alpha)\\ \hat{p}+\sqrt{2}\,\mathrm{Im}(\alpha)\end{bmatrix}. \tag{3.16}$$

3.3. Squeeze gate

The squeeze operator for *real* squeezing ξ is given by [28, p. 491]

$$\hat{S} = e^{\frac{\hat{a}_1^2}{2}\xi - \frac{\hat{a}_1^{\dagger 2}}{2}\xi}. \tag{3.17}$$

The effect on the annihilation operator is [29, p. 58]

$$\hat{S}^\dagger \hat{a} \hat{S} = \hat{a}\cosh(\xi) - \hat{a}^\dagger \sinh(\xi). \tag{3.18}$$

The effect of the squeeze operator on the creation operator is easily found by taking the hermitian conjugate

$$\hat{S}^\dagger \hat{a}^\dagger \hat{S} = \hat{a}^\dagger \cosh(\xi) - \hat{a}\sinh(\xi). \tag{3.19}$$

The effect on the $\hat{x}$ quadrature can now be found to be

$$\begin{aligned}
\hat{S}^\dagger \hat{x} \hat{S} &= \frac{1}{2}(\hat{S}^\dagger(\hat{x} + i\hat{p} + \hat{x} - i\hat{p})\hat{S}) \\
&= \frac{1}{\sqrt{2}}(\hat{S}^\dagger \hat{a} \hat{S} + \hat{S}^\dagger \hat{a}^\dagger \hat{S}) \\
&= \frac{1}{\sqrt{2}}(\hat{a}\cosh(\xi) - \hat{a}^\dagger \sinh(\xi) + \hat{a}^\dagger \cosh(\xi) - \hat{a}\sinh(\xi)) \\
&= \frac{1}{\sqrt{2}}(\hat{a}(\cosh(\xi) - \sinh(\xi)) + \hat{a}^\dagger(\cosh(\xi) - \sinh(\xi))) \\
&= e^{-\xi}\frac{1}{\sqrt{2}}(\hat{a} + \hat{a}^\dagger) \\
&= e^{-\xi}\hat{x}.
\end{aligned} \tag{3.20}$$

Similarly, the effect on the $\hat{p}$ quadrature is [29, p. 57]

$$\hat{S}^\dagger \hat{p} \hat{S} = e^{\xi}\hat{p}. \tag{3.21}$$

Therefore, the effect of the real squeeze gate can be summarized by

$$S(r): \quad \begin{bmatrix} \hat{x}_r \\ \hat{p}_r \end{bmatrix} = \begin{bmatrix} e^{-r} & 0 \\ 0 & e^{r} \end{bmatrix} \begin{bmatrix} \hat{x} \\ \hat{p} \end{bmatrix}. \tag{3.22}$$

3.4. Beam splitter gate

The basic two mode gate is the beam splitter [26]. A phaseless beam splitter can be described by (as derived in section 4.1 and A.2)

$$\begin{bmatrix} \hat{a}_1' \\ \hat{a}_2' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} \hat{a}_1 \\ \hat{a}_2 \end{bmatrix}. \tag{3.23}$$

The effect on the quadratures of both modes can be found by looking at the real and imaginary part of the annihilation operator separately

$$\begin{aligned} \hat{x}_1' + i\hat{p}_1' &= \cos(\theta)(\hat{x}_1 + i\hat{p}_1) - \sin(\theta)(\hat{x}_2 + i\hat{p}_2) \\ &= (\cos(\theta)\hat{x}_1 - \sin(\theta)\hat{x}_2) + i(\cos(\theta)\hat{p}_1 - \sin(\theta)\hat{p}_2). \end{aligned} \tag{3.24}$$

Similarly, the effect on the second mode is

$$\begin{aligned} \hat{x}_2' + i\hat{p}_2' &= \sin(\theta)(\hat{x}_1 + i\hat{p}_1) + \cos(\theta)(\hat{x}_2 + i\hat{p}_2) \\ &= (\sin(\theta)\hat{x}_1 + \cos(\theta)\hat{x}_2) + i(\sin(\theta)\hat{p}_1 + \cos(\theta)\hat{p}_2). \end{aligned} \tag{3.25}$$

The effect of a phaseless beam splitter on the quadrature operators in phase space can therefore be summarized by

$$R(\theta): \quad \begin{bmatrix} \hat{x}_1' \\ \hat{x}_2' \\ \hat{p}_1' \\ \hat{p}_2' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & \cos(\theta) & -\sin(\theta) \\ 0 & 0 & \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \\ \hat{p}_1 \\ \hat{p}_2 \end{bmatrix}. \tag{3.26}$$

3.5. Kerr gate

For universality a nonlinearity is needed [26]. This is achieved with the so-called *Kerr* gate, which can be represented by [26]

$$K(\kappa) = e^{i\kappa\hat{n}^2}. \tag{3.27}$$

Its effect on the annihilation operator is [28, p. 308]

$$K^\dagger(\kappa)\hat{a}K(\kappa) = \hat{a}e^{-2i\kappa\hat{n}}. \tag{3.28}$$

The effect on the creation operator is then found by taking the Hermitian conjugate

$$K^\dagger(\kappa)\hat{a}^\dagger K(\kappa) = e^{2i\kappa\hat{n}}\hat{a}^\dagger. \tag{3.29}$$

The effect on the quadratures is, offered without proof [28, p. 309]

$$K(\kappa): \quad \begin{bmatrix} \hat{x}_\kappa \\ \hat{p}_\kappa \end{bmatrix} = \begin{bmatrix} \hat{x}\cosh\phi - i\hat{p}\sinh\phi \\ \hat{p}\cosh\phi + i\hat{x}\sinh\phi \end{bmatrix}, \tag{3.30}$$

where $\phi(\hat{x}, \hat{p}) = K(\hat{x}\hat{p} - \hat{p}\hat{x} - i\hat{x}^2 - i\hat{p}^2)$. The most important thing to notice about this transformation, is that it is nonlinear in phase space, which is important a neural network type transformation.

4. Physical realization

4.1. Beam splitter

As a basic component for two modes a beam splitter can be used, of which a coarse schematic can be seen in figure 4.1.

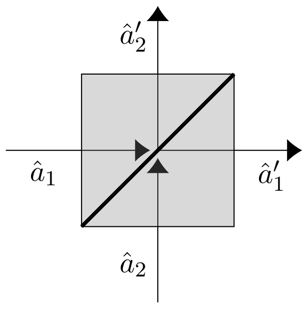


Figure 4.1.: A diagram of an ideal beam splitter adapted from [29, p. 93]. Two different modes interfere and lead to new annihilation operators.

Assuming, the inputs are in coherent states (see section 2.4), the interaction of a beam splitter can be written by [29, p. 94]

$$\begin{aligned}\begin{bmatrix}\hat{a}_1' \\ \hat{a}_2'\end{bmatrix} &= B\begin{bmatrix}\hat{a}_1 \\ \hat{a}_2\end{bmatrix} \\ &= \begin{bmatrix}B_{11} & B_{12} \\ B_{21} & B_{22}\end{bmatrix}\begin{bmatrix}\hat{a}_1 \\ \hat{a}_2\end{bmatrix}.\end{aligned} \tag{4.1}$$

This assumes, that the action of the beam splitter is linear, but that is a good approximation in this case [29, p. 94]. Using the Bose commutation relations (2.15), this transformation can be proven to be unitary, which is done in section A.2.

Using the Z-Y decomposition theorem, any unitary 2×2 gate can be written by [32, p. 175]

$$B = e^{i\alpha}\begin{bmatrix}e^{-i\frac{\beta}{2}} & 0 \\ 0 & e^{i\frac{\beta}{2}}\end{bmatrix}\begin{bmatrix}\cos\left(\frac{\gamma}{2}\right) & -\sin\left(\frac{\gamma}{2}\right) \\ \sin\left(\frac{\gamma}{2}\right) & \cos\left(\frac{\gamma}{2}\right)\end{bmatrix}\begin{bmatrix}e^{-i\frac{\delta}{2}} & 0 \\ 0 & e^{i\frac{\delta}{2}}\end{bmatrix}, \tag{4.2}$$

where α, β, γ and δ are constants that define the transformation. For a beam splitter without phase change (phaseless), the effect becomes [29, p. 95]

$$\begin{bmatrix} \hat{a}_1' \\ \hat{a}_2' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} \hat{a}_1 \\ \hat{a}_2 \end{bmatrix}, \tag{4.3}$$

where $\theta = \frac{\gamma}{2}$.

4.2. Phase change

A phase shift can be achieved by letting the light pass through a medium with a different refraction index as can be seen in figure 4.2 [28, p. 29-30]. The effect is then given by a simple time evolution operator as shown in (3.1) and the effect on the annihilation operator is (according to (3.2))

$$\hat{a}_1' = e^{-i\phi}\hat{a}_1. \tag{4.4}$$

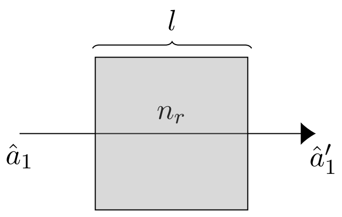


Figure 4.2.: A diagram of a phase gate adapted from [28, p. 296]. The input signal goes through a different medium with refraction index n_r and length l.

Since the phase shift is given by $\phi = n_r k l$, where n_r is the refraction index, k is the wave number and l is the length [28, p. 29], different phase changes can be achieved by altering these variables.

4.3. Displacement

A schematic of a physical implementation of the displacement gate can be seen in figure 4.3 [28, p. 297]. Similar to homodyne detection (see section 4.6), the gate is implemented by interfering with a local laser, which is so powerful in comparison to the signal, that it can be treated classically as having the amplitude β. According to (4.3), the effect on $\hat{a}_1$ o´f the beam splitter is

$$\hat{a}_1' = \cos(\theta)\,\hat{a}_1 - \sin(\theta)\,\beta. \tag{4.5}$$

When $\theta << 1$, this can be approximated using the small angle approximation [9]

$$\begin{aligned}\hat{a}_1' &= \hat{a}_1 - \theta\beta \\ &= \hat{a}_1 + \alpha,\end{aligned} \tag{4.6}$$

where $\alpha = -\theta\beta$. Therefore, by either changing θ or β, one can change the value of the displacement.

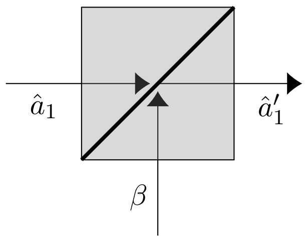


Figure 4.3.: A diagram of a displacement gate adapted from [28, p. 296]. The input signal interferes with a local laser with an amplitude much higher than the signal, so it can be treated classically.

4.4. Squeezing

There are several ways to implement a single mode squeezing operator, but one is degenerate parametric down-conversion [23, p. 165]. A diagram of this approach can be seen in figure 4.4. Along with the signal $\hat{a}_1$, a powerful coherent light is being sent into a nonlinear medium (could be a nonlinear crystal [12, p. 237]) to change its environment (a process called pumping). The total Hamiltonian of the process is given by [23, p. 165]

$$\hat{H} = \hbar\omega\hat{a}^\dagger\hat{a} + \hbar\omega_L\hat{a}_L^\dagger\hat{a}_L + i\hbar\chi^{(2)}\left(\hat{a}_1^2\hat{a}_L^\dagger - \hat{a}_1^{\dagger 2}\hat{a}_L\right), \tag{4.7}$$

where $\chi^{(2)}$ represents the strength of the nonlinear term and is medium dependent. Using the so-called parametric approximation, which holds when the local field is strong enough to never be undepleted of photons [23, p. 165], (4.7) becomes

$$\begin{aligned}\hat{H}_{PA} &= \hbar\omega\hat{a}^\dagger\hat{a} + i\hbar\chi^{(2)}\left(\hat{a}_1^2\hat{a}_L^\dagger - \hat{a}_1^{\dagger 2}\hat{a}_L\right)\\ &= \hbar\omega\hat{a}^\dagger\hat{a} + i\hbar\chi^{(2)}\left(\hat{a}_1^2\alpha_L^* e^{i\omega_L t} - \hat{a}_1^{\dagger 2}\alpha_L e^{-i\omega_L t}\right)\\ &= \hat{H}_0 + \hat{V},\end{aligned} \tag{4.8}$$

where the annihilation operator of the local laser has been split into its polar coordinates $\hat{a}_L = \alpha_L e^{-i\omega_L t}$ and the constant term $\hbar\omega_p|\alpha_L|^2$ has been dropped. The two different parts of the Hamiltonian can be analyzed in the interaction picture [8, p. 166]

$$\begin{aligned}\hat{H}_I &= e^{i\hat{H}_0}\hat{V}e^{-i\hat{H}_0}\\ &= e^{i\omega_L\hat{a}^\dagger\hat{a}t} i\hbar\chi^{(2)}\left(\hat{a}_1^2\alpha_L^* e^{i\omega_L t} - \hat{a}_1^{\dagger 2}\alpha_L e^{-i\omega_L t}\right)e^{-i\omega_L\hat{a}^\dagger\hat{a}t}\\ &= i\hbar\chi^{(2)}\left(\hat{a}_1^2\alpha_L^* e^{i(\omega_L-2\omega)t} - \hat{a}_1^{\dagger 2}\alpha_L e^{-i(\omega_L-2\omega)t}\right),\end{aligned} \tag{4.9}$$

where (3.5) was used.

Using a local laser with mode $\omega_L = 2\omega$, the interaction Hamiltonian become

$$\begin{aligned}\hat{H}_I &= i\hbar\chi^{(2)}\left(\hat{a}_1^2\alpha_L^* - \hat{a}_1^{\dagger 2}\alpha_L\right)\\ &= i\hbar\left(\frac{\hat{a}_1^2}{2}\xi^* - \frac{\hat{a}_1^{\dagger 2}}{2}\xi\right),\end{aligned} \tag{4.10}$$

where $\xi = 2\chi^{(2)}\alpha_L$. This generates the squeeze gate

$$\begin{aligned}\hat{S} &= e^{-\frac{i}{\hbar}\hat{H}_I t}\\ &= e^{\frac{\hat{a}_1^2}{2}\xi^* t - \frac{\hat{a}_1^{\dagger 2}}{2}\xi t}.\end{aligned} \tag{4.11}$$

4.5. Kerr interaction

The Kerr transformation is equivalent to an intensity-dependent phase transformation, called *self-phase modulation* [28, p. 308]. Without going into detail, this can be achieved in a medium with a non-vanishing cubic polarization reaction to the incoming beam, which can be achieved e.g. with an optical fiber [12, p. 345]. After propagation through the Kerr material, the light splits into an ω and a 3ω term, the last of which has to filtered out. A rough diagram of this can be seen in figure 4.5.

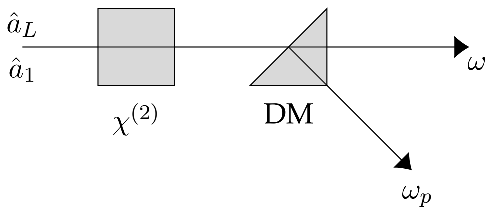


Figure 4.4.: A schematic of a squeeze gate adapted from [28, p. 296]. The input signal goes through a nonlinear medium which is pumped by a powerful local source and then the unwanted frequency is filtered out with a dichoric mirror.

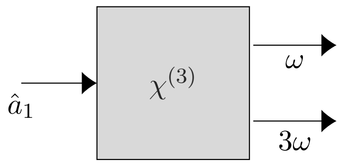


Figure 4.5.: A schematic of a Kerr gate adapted from [28, p. 309]. The input signal goes through a Kerr medium with a $\chi^{(3)}$ interaction and the unwanted 3ω part of the light is discarded.

4.6. Measurement

In this work, the only variable used is the expectation value of the position quadrature. Therefore, only this operator needs to be measured. This can be done e.g. by using so-called balanced homodyne detection of which a coarse diagram can be seen in figure 4.6 [29, p. 110-111]. The mode to be measured interferes with a local laser light, that is shifted with a known phase ϕ compared to the signal. In practice this can be achieved by the signal and the local laser coming from the same master laser. The local oscillator's phase is then shifted using a piezo-electrically movable mirror [29, p.111].

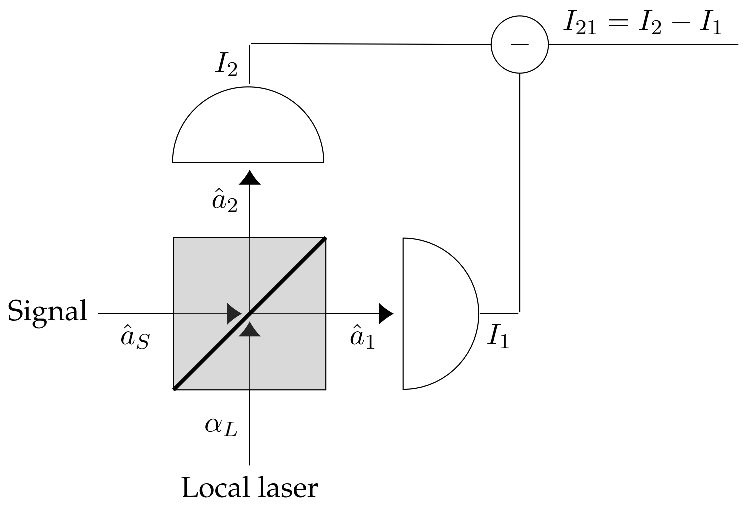


Figure 4.6.: A schematic of homodyne detection adapted from [29, p. 110]. The input signal interferes with a local oscillator in a beam splitter resulting in two output rays, whose light intensity is detected.

A phaseless beam splitter with no loss and $\theta = \frac{\pi}{2}$ has the effect of (looking at equation (3.23))

$$\begin{bmatrix} \hat{a}_1 \\ \hat{a}_2 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} \hat{a}_S \\ \alpha_L \end{bmatrix} . \tag{4.12}$$

Such a beam splitter is used in the detection. The laser is kept strong in comparison with the signal, so it can be treated classically as having a well-defined amplitude α [29, p.111].

It is now possible to define:

$$\begin{aligned}
\hat{n}_{21} &= \hat{n}_2 - \hat{n}_1 \\
&= \hat{a}_2^\dagger \hat{a}_2 - \hat{a}_1^\dagger \hat{a}_1 \\
&= \frac{1}{2}\left(\left(\hat{a}_S^\dagger + \alpha_L^*\right)\left(\hat{a}_S + \alpha_L\right) - \left(\hat{a}_S^\dagger - \alpha_L^*\right)\left(\hat{a}_S - \alpha_L\right)\right) \\
&= \alpha_L \hat{a}_S^\dagger + \alpha_L^* \hat{a}_S \\
&= \frac{1}{\sqrt{2}}\left(\mathrm{Re}\left(\alpha_L\right) + i\,\mathrm{Im}\left(\alpha_L\right)\right)\left(\hat{x}_S - i\hat{p}_S\right) + \left(\mathrm{Re}\left(\alpha_L\right) - i\,\mathrm{Im}\left(\alpha_L\right)\right)\left(\hat{x}_S + i\hat{p}_S\right) \\
&= \sqrt{2}\left(\mathrm{Re}\left(\alpha_L\right)\hat{x}_S + \mathrm{Im}\left(\alpha_L\right)\hat{p}_S\right) \\
&= \sqrt{2}|\alpha_L|\left(\cos\left(\theta\right)\hat{x}_S + \sin\left(\theta\right)\hat{p}_S\right) \\
&= \sqrt{2}|\alpha_L|\hat{x}_\theta,
\end{aligned} \tag{4.13}$$

where θ is the relative phase and the last step uses (3.6). Logically, the intensity difference I_{21} must be proportional to n_{21} because this represents the difference in number of photons. Therefore, the proportionality can be deduced

$$I_{21} \propto \hat{x}_\theta, \tag{4.14}$$

where the proportionality constant can be found by calibration.

4.7. Interferometer

Consisting solely of beam splitters and phase gates, universal multiport interferometers can represent any unitary linear transformation between multiple modes in phase space [17]. An N-mode interferometer uses $N(N-1)/2$ beamsplitters and $N(N-1)/2$ phase shifters. A rough schematic of a 3-mode version can be seen in figure 4.7.

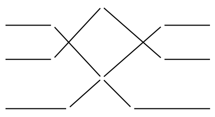


Figure 4.7.: A schematic of a 3-mode interferometer adapted from [17]. Each intersection contains a phase gate and a beam splitter.

5. Parameter shift rule

In order to optimize an optical CV variational circuit with respect to its parameters, it is necessary to differentiate the output with respect to the parameters similar to backpropagation in classical machine learning. Assuming, the outputs are expectation values, this can be done by the so-called parameter-shift rule [6]. The central idea is, that the partial derivative of an expectation value with respect to a circuit parameter can be found by two expectation values at certain shifts to that parameter. The parameter shift rule for a CV quantum gate holds for any Gaussian gate followed by at most logarithmically many non-Gaussian operations [37]. As all used gates except the Kerr gates are Gaussian, this rule is vital for calculating the gradient necessary for optimization.

5.1. Single gate

A single gate quantum circuit with the output being the expectation value of the quadrature $\hat{x}$ as can be seen in figure 5.1. This could represent a phase gate or any other Gaussian single mode gate with a well-defined parameter shift rule.

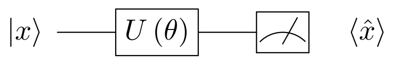


Figure 5.1.: Variational circuit to illustrate a parametrized single gate circuit where the output is the expectation value of the $\hat{x}$ quadrature.

Since the expectation value depends on θ and x it can therefore be seen as a function of those parameters. x is the input and θ is the learnable parameter

$$\begin{aligned} f(x;\theta) &= \langle x|U^{\dagger}(\theta)\,\hat{x}U(\theta)\,|x\rangle \\ &= \langle x|M_{\theta}(\hat{x})\,|x\rangle\,, \end{aligned} \tag{5.1}$$

where $M_{\theta}(\hat{x})$ is the transformation of the operator $\hat{x}$ in the Heisenberg picture. The parameter shift rule then says, that for certain gates the partial derivative of the transformation can be rewritten as [6]

$$\nabla M_{\theta}(\hat{x}) = c\left(M_{\theta+s}(\hat{x}) - M_{\theta-s}(\hat{x})\right), \tag{5.2}$$

where s and θ are unique to the specific type of gate. Because of the product rule and the fact that $|x\rangle$ is independent of θ, the partial derivative of the variational circuit function can be written as

$$\begin{aligned} \nabla_\theta f(x;\theta) &= \langle x|\nabla_\theta M_\theta(\hat{x})|x\rangle \\ &= c\left(\langle x|M_{\theta+s}(\hat{x})|x\rangle - \langle x|M_{\theta-s}(\hat{x})|x\rangle\right) \\ &= c\left(f(x;\theta+s) - f(x;\theta-s)\right), \end{aligned} \tag{5.3}$$

for some $c, s \in \mathbb{R}$ and where equation (5.1) and (5.2). Notice the similarity to finite differences, except this expression is analytical and so the only errors are rounding errors or experimental errors if this is realized in a physical circuit. This rule is very powerful as any circuit, no matter how deep or complicated in principle has a simple analytical derivative of each parameter.

5.2. Multiple gates

The parameter shift rule easily generalizes to multiple parametrized gates as in figure 5.2.

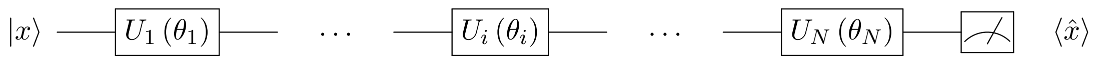


Figure 5.2.: Quantum circuit to illustrate a parametrized multiple gate circuit where the output is the expectation value of the $\hat{x}$ quadrature.

The function in this case is

$$\begin{aligned} f(x;\theta) &= \langle x|U_1^\dagger(\theta_1)\dots U_i^\dagger(\theta_i)\dots U_N^\dagger(\theta_N)\,\hat{x}U_N(\theta_N)\dots U_i(\theta_i)\dots U_1(\theta_1)|x\rangle \\ &= \langle x_{i-1}|U_i^\dagger(\theta_i)\,\hat{x}_{i+1}U_i(\theta_i)|x_{i-1}\rangle \\ &= \langle x_{i-1}|M_{\theta_i}(\hat{x}_{i+1})|x_{i-1}\rangle, \end{aligned} \tag{5.4}$$

where $|x_{i-1}\rangle = U_{i-1}(\theta_{i-1})\dots U_1(\theta_1)|x\rangle$ and the operator $\hat{x}_{i+1} = U_{i+1}^\dagger(\theta_i)\dots U_N^\dagger(\theta_N)\,\hat{x}U_N(\theta_N)\dots U_{i+1}(\theta_i)$. Because only U_i depends on θ_i, the product rule causes the partial derivative of this expression with respect to θ_i to have the same form as the single gate case

$$\begin{aligned} \nabla_{\theta_i} f(x;\theta) &= \langle x_{i-1}|\nabla_{\theta_i} M_{\theta_i}(\hat{x}_{i+1})|x_{i-1}\rangle \\ &= c\left(\langle x_{i-1}|M_{\theta_i+s}(\hat{x}_{i+1})|x_{i-1}\rangle - \langle x_{i-1}|M_{\theta_i-s}(\hat{x}_{i+1})|x_{i-1}\rangle\right), \end{aligned} \tag{5.5}$$

for some $c, r \in \mathbb{R}$.

The transformation for both $\hat{x}$ and $\hat{p}$ can be represented by a matrix M, which makes the notation simpler. Since matrix multiplication is distributive, $\begin{bmatrix}\hat{x}\\ \hat{p}\end{bmatrix}$ can be moved to the end of the circuit. Therefore, it is enough to consider the gate matrix, which is done in the following.

5.3. Phase gate

The derivative of the gate, looking at its effect on the quadratures (section 3.1), is

$$\nabla_\phi M(\phi) = \begin{bmatrix} -\sin(\phi) & -\cos(\phi) \\ \cos(\phi) & -\sin(\phi) \end{bmatrix}. \tag{5.6}$$

The definition of the parameter shift rule for the phase gate becomes

$$\begin{aligned} c\left(M(\phi+s) - M(\phi-s)\right) &= c\left(\begin{bmatrix} \cos(\phi+s) & \sin(\phi+s) \\ -\sin(\phi+s) & \cos(\phi+s) \end{bmatrix} - \begin{bmatrix} \cos(\phi-s) & \sin(\phi-s) \\ -\sin(\phi-s) & \cos(\phi-s) \end{bmatrix}\right) \\ &= c\begin{bmatrix} \cos(\phi+s) - \cos(\phi-s) & -\sin(\phi+s) + \sin(\phi-s) \\ \sin(\phi+s) - \sin(\phi-s) & \cos(\phi+s) - \cos(\phi-s) \end{bmatrix}. \end{aligned} \tag{5.7}$$

If $s = \frac{\pi}{2}$ and the fact that $\cos\left(\phi - \frac{\pi}{2}\right) = \sin(\phi)$, $\cos\left(\phi + \frac{\pi}{2}\right) = -\sin(\phi)$, $\sin\left(\phi - \frac{\pi}{2}\right) = -\cos(\phi)$ and $\sin\left(\phi + \frac{\pi}{2}\right) = \cos(\phi)$ is used, this can be simplified significantly

$$\begin{aligned} c\left(M(\phi+\frac{\pi}{2}) - M(\phi-\frac{\pi}{2})\right) &= c\begin{bmatrix} \cos\left(\phi+\frac{\pi}{2}\right) - \cos\left(\phi-\frac{\pi}{2}\right) & -\sin\left(\phi+\frac{\pi}{2}\right) + \sin\left(\phi-\frac{\pi}{2}\right) \\ \sin\left(\phi+\frac{\pi}{2}\right) - \sin\left(\phi-\frac{\pi}{2}\right) & \cos\left(\phi+\frac{\pi}{2}\right) - \cos\left(\phi-\frac{\pi}{2}\right) \end{bmatrix} \\ &= c\begin{bmatrix} -2\sin(\phi) & -2\cos(\phi) \\ 2\cos(\phi) & -2\sin(\phi) \end{bmatrix}. \end{aligned} \tag{5.8}$$

With $c = \frac{1}{2}$, the parameter shift rule of the phase gate becomes [37]

$$\nabla_\phi M(\phi) = \frac{1}{2}\left(M\left(\phi + \frac{\pi}{2}\right) - M\left(\phi - \frac{\pi}{2}\right)\right). \tag{5.9}$$

5.4. Displacement gate

Looking at (3.16), the Heisenberg representation of the displacement gate on the quadratures is

$$M(a,\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 2a\cos(\phi) & 1 & 0 \\ 2a\sin(\phi) & 0 & 1 \end{bmatrix}, \tag{5.10}$$

where it is applied to the basis $\begin{bmatrix} 1 \\ \hat{x} \\ \hat{p} \end{bmatrix}$. a is the length and ϕ is the phase of the complex displacement. The derivative of this gate with respect to a is

$$\nabla_a M(a,\phi) = \begin{bmatrix} 0 & 0 & 0 \\ 2\cos(\phi) & 0 & 0 \\ 2\sin(\phi) & 0 & 0 \end{bmatrix} \tag{5.11}$$

The parameter shift rule of this gate with respect to a can be derived from the definition (5.2)

$$\begin{aligned} c\left(M(a+s) - M(a-s)\right) &= c\left(\begin{bmatrix} 1 & 0 & 0 \\ 2(a+s)\cos(\phi) & 1 & 0 \\ 2(a+s)\sin(\phi) & 0 & 1 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 \\ 2(a-s)\cos(\phi) & 1 & 0 \\ 2(a-s)\sin(\phi) & 0 & 1 \end{bmatrix}\right) \\ &= c\left(\begin{bmatrix} 0 & 0 & 0 \\ 4s\cos(\phi) & 0 & 0 \\ 4s\sin(\phi) & 0 & 0 \end{bmatrix}\right). \end{aligned} \tag{5.12}$$

One can now see, that any s is allowed if the constant is set to $c = \frac{1}{2s}$. The final rule for the displacement gate is therefore, assuming that the phase is $\phi = 0$ [37]

$$\nabla_r M(r) = \frac{1}{2s}\left(M(r+s) - M(r-s)\right). \tag{5.13}$$

5.5. Squeeze gate

A diagram of single squeeze gate circuit can be seen in figure 5.3.

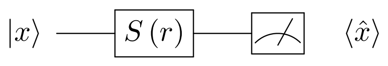


Figure 5.3.: Quantum circuit with a single parametrized squeeze gate where the output is the expectation value of the $\hat{x}$ operator. The argument is set to 0, so the squeezing is real.

Looking at equation (3.22), the gradient of the effect of the squeeze gate with respect to

r in the Heisenberg picture is

$$\begin{aligned}\nabla_r M(r) &= \nabla_r \begin{bmatrix} e^{-r} & 0 \\ 0 & e^{r} \end{bmatrix} \\ &= \begin{bmatrix} -e^{-r} & 0 \\ 0 & e^{r} \end{bmatrix}\end{aligned} \tag{5.14}$$

This can be written in the form (5.2) [6]

$$\begin{aligned}\nabla_r M(r) &= \frac{1}{e^s - e^{-s}} \begin{bmatrix} -e^{-r}\left(e^s - e^{-s}\right) & 0 \\ 0 & e^{r}\left(e^s - e^{-s}\right) \end{bmatrix} \\ &= \frac{1}{e^s - e^{-s}} \left(\begin{bmatrix} e^{-(r+s)} & 0 \\ 0 & e^{r+s} \end{bmatrix} - \begin{bmatrix} e^{-(r-s)} & 0 \\ 0 & e^{r-s} \end{bmatrix} \right) \\ &= c\left(M(r+s) - M(r-s)\right),\end{aligned} \tag{5.15}$$

where $c = \frac{1}{e^s - e^{-s}}$. Looking back at equation (5.3), this leads to a partial derivative of the whole circuit as

$$\nabla_r f\left(x; r\right) = \frac{1}{e^s - e^{-s}} \left(f\left(x; r+s\right) - f\left(x; r-s\right)\right), \tag{5.16}$$

where s has to be as small as possible, as long as the prefactor doesn't blow up.

5.6. Beam splitter gate

The gradient of a phaseless beam splitter with respect to θ is easily seen to be similar to the phase gate version because the gates are similar. The partial derivative of a none-phaseless beam splitter with respect to the phase parameter ϕ is identical. Both rules can be summarized as [37]

$$\nabla_\theta M\left(\theta, \phi\right) = \frac{1}{2}\left(M\left(\theta + \frac{\pi}{2}, \phi\right) - M\left(\theta - \frac{\pi}{2}, \phi\right)\right) \tag{5.17}$$

$$\nabla_\phi M\left(\theta, \phi\right) = \frac{1}{2}\left(M\left(\theta, \phi + \frac{\pi}{2}\right) - M\left(\theta, \phi - \frac{\pi}{2}\right)\right). \tag{5.18}$$

6. Quantum neural networks

6.1. Classic artificial neural networks

In this section, neural networks are introduced together with a necessary abstraction in order to implement them on a CV quantum computer. A rough schematic of a fully connected neural network can be seen in figure 6.1. It consists of an input layer, an output layer and an arbitrary number of hidden layers all with an arbitrary number of so-called *neurons*.

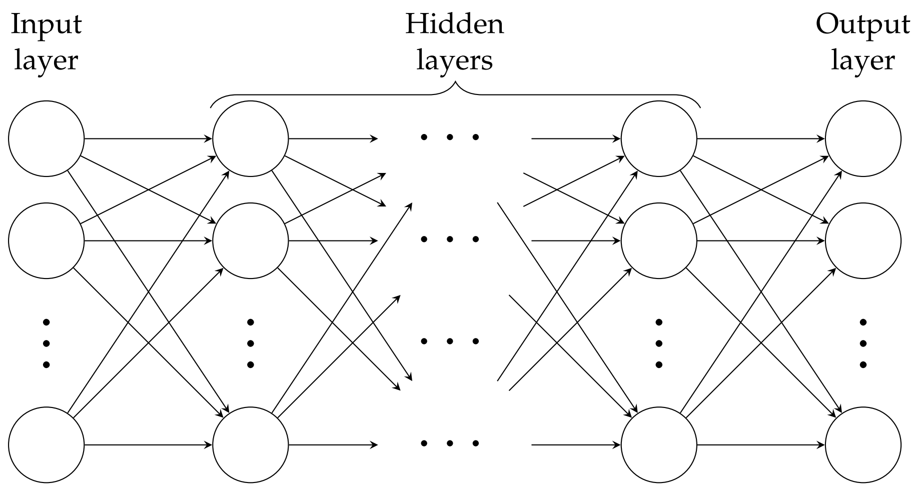


Figure 6.1.: Schematic of a fully connected artificial neural network.

A schematic of an individual neuron can be seen in figure 6.2 [2]. The function of a neuron is to add the inputs weighted by the weights w_i of the edges plus a bias b and then apply an activation function a. A typical activation function is the sigmoid function [2]

$$a\left(x\right) = \frac{1}{1+e^{-x}}, \tag{6.1}$$

which transforms the real axis to the interval $]0,1[$. Another activation function is hyperbolic tangents [2]

$$a\left(x\right) = \tanh(x), \tag{6.2}$$

which transforms the real axis to the interval $]-1, 1[$.

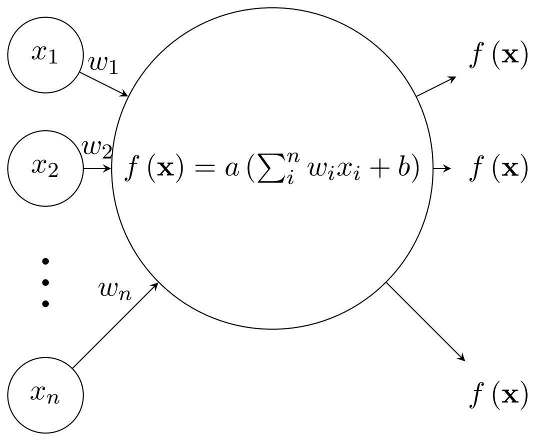


Figure 6.2.: Schematic of a single neuron of a neural network with several input and output neurons.

It can easily be seen, that the sum of the neuron excluding bias is equivalent to an inner product between a weight vector and an input vector

$$y = \begin{bmatrix} w_1 & \dots & w_n \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}. \tag{6.3}$$

Because each neuron in a layer has a different weight vector, the total transformation of one layer *before* the activation function can be represented by a linear transformation

$$\mathbf{y} = W\mathbf{x} + \mathbf{b}, \tag{6.4}$$

where W is the weight matrix. Including the activation function, the total effect in a neuron is

$$f(\mathbf{x}) = a\left(W\mathbf{x} + \mathbf{b}\right). \tag{6.5}$$

Therefore, a fully connected neural network is essentially nothing more than a linear transformation followed by a nonlinear activation function. This abstraction is important because it turns out that both linear transformations and nonlinearities with trainable parameters are realizable on a CV quantum computer. The universal approximation theorems say that neural networks can approximate any function to any precision by an arbitrarily deep or wide neural network [35] [25]. Therefore, being able to realize such networks on a CV-quantum computer could potentially be a big advantage.

6.2. Artificial neural networks on a CV quantum computer

Any Gaussian transformation in phase space (see section 2.5) can be written by [26]

$$\begin{bmatrix} \hat{\mathbf{x}}' \\ \hat{\mathbf{p}}' \end{bmatrix} = M \begin{bmatrix} \hat{\mathbf{x}} \\ \hat{\mathbf{p}} \end{bmatrix} + \begin{bmatrix} \boldsymbol{\alpha}_{\boldsymbol{Re}} \\ \boldsymbol{\alpha}_{\boldsymbol{Im}} \end{bmatrix}, \tag{6.6}$$

where M is a real-valued, so-called *sympletic* matrix and $\left[\begin{smallmatrix} \boldsymbol{\alpha}_{\boldsymbol{Re}} \\ \boldsymbol{\alpha}_{\boldsymbol{Im}} \end{smallmatrix}\right]$ is a vector of complex numbers. A sympletic matrix is defined by [26]

$$\begin{aligned} M^T\Omega M &= \Omega \\ \begin{bmatrix} M_{11} & M_{21} \\ M_{12} & M_{22} \end{bmatrix} \begin{bmatrix} 0 & I \\ I & 0 \end{bmatrix} \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} &= \begin{bmatrix} 0 & I \\ I & 0 \end{bmatrix} \\ \begin{bmatrix} M_{11}M_{21} + M_{21}M_{11} & M_{11}M_{22} + M_{21}M_{12} \\ M_{12}M_{21} + M_{22}M_{11} & M_{12}M_{22} + M_{22}M_{12} \end{bmatrix} &= \begin{bmatrix} 0 & I \\ I & 0 \end{bmatrix} \end{aligned} \tag{6.7}$$

It is evident from equation (6.6), that the block off-diagonal elements of M mix the different quadratures. The transformation (6.6) is very close to the linear transformation needed for a neural network (6.4). One can notice how, looking back at section 3.2, that the addition with the vector can be achieved by a displacement gate on each individual mode. Any sympletic matrix can be decomposed using the Bloch-Messiah decomposition [26]

$$M = K_2 \begin{bmatrix} \Sigma & 0 \\ 0 & \Sigma^{-1} \end{bmatrix} K_1, \tag{6.8}$$

where $\Sigma = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_N)$ is a diagonal matrix of the real, positive singular values and K_1 and K_2 are orthogonal, sympletic and real-valued. To implement a real-valued classic neural network transformation, the matrix M must be block diagonal to not mix $\hat{\mathbf{x}}$ and $\hat{\mathbf{p}}$

$$M = \begin{bmatrix} M_{11} & 0 \\ 0 & M_{22} \end{bmatrix}, \tag{6.9}$$

which leads to the orthogonal matrices also being block diagonal

$$K_1 = \begin{bmatrix} K_{11} & 0 \\ 0 & K_{12} \end{bmatrix} \quad K_2 = \begin{bmatrix} K_{21} & 0 \\ 0 & K_{22} \end{bmatrix}. \tag{6.10}$$

Using this constraint the effect on $\hat{\mathbf{x}}$ becomes

$$\begin{aligned} \hat{\mathbf{x}}' &= M_{11}\hat{\mathbf{x}} + \boldsymbol{\alpha}_{\boldsymbol{Re}} \\ \hat{\mathbf{x}}' &= K_{21}\Sigma K_{11}\hat{\mathbf{x}} + \boldsymbol{\alpha}_{\boldsymbol{Re}}. \end{aligned} \tag{6.11}$$

This decomposition makes the transformation realizable with the simple CV gates described in section 3. Because K_1 and K_2 are orthogonal and therefore unitary, they can be achieved on two different universal interferometers with only phaseless beam splitters as described in section 4.7 [26]. As Σ is a diagonal matrix, its effect can be achieved by a simple squeeze gate for each mode, resulting in a scaling. To achieve the necessary nonlinearity, the so-called Kerr gate is used mode-wise, which is non-Gaussian.

A setup performing the complete transformation is shown in figure 6.3, where $\boldsymbol{\alpha}$, $\mathbf{r}$, $\boldsymbol{\lambda}$, $\boldsymbol{\phi}_1$, $\boldsymbol{\theta}_1$, $\boldsymbol{\phi}_2$ and $\boldsymbol{\theta}_2$ are the parameters of the circuit. Because this setup is parameterized, it can be used to learn a transformation similar to the learning of weights in a neural network.

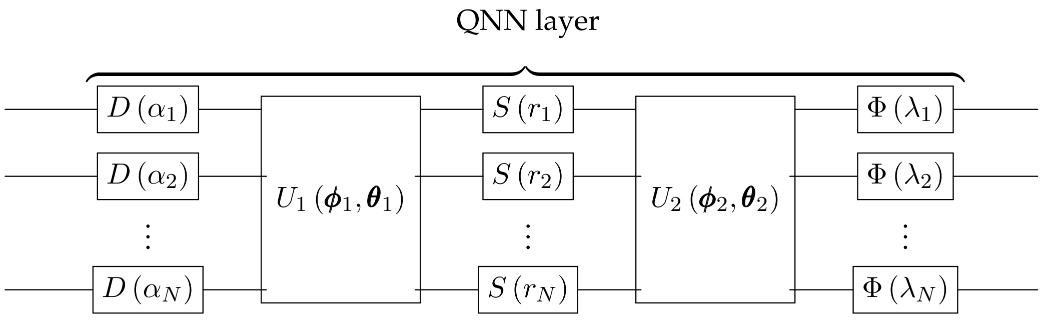


Figure 6.3.: Variational circuit for implementing one neural network layer on a CV quantum computer consisting of two interferometers as well as one displacement, squeeze and Kerr gate for each wire. Adapted from [26].

Part II.

PennyLane Simulation Framework

7. Overview

The Python framework *PennyLane* from Canadian company Xanadu was used to simulate the CV circuits used for numerical experiments [14]. This Python package allows not only simulation of CV quantum computers, but also contains hybrid computation capacity and is therefore perfect for quantum machine learning. Furthermore, this package is compatible with both NumPy and PyTorch, and these interfaces were used extensively.

PennyLane allows insertion of a quantum node (not to be confused with modes) into the directed acyclic graph of a machine learning computation similar to the directed acyclic graph of TensorFlow and PyTorch as in figure 7.1.

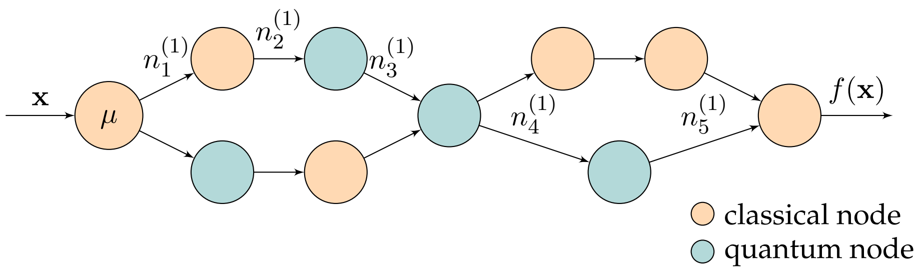


Figure 7.1.: Directed acyclic graph of a PennyLane hybrid computation involving classical and quantum nodes. One of the 4 possible paths through the graph from the first node with parameter μ is indicated.

The gradient or Jacobian can be computed by standard backpropagation using the chain rule. When $n_i^{(p)}$ represents the output of the ith node of the pth path the derivative of the output with respect to the specific parameter μ becomes [14]

$$\frac{\partial f(\mathbf{x})}{\partial \mu} = \sum_{p=1}^{N} \frac{\partial f(\mathbf{x})}{\partial n_m^{(p)}} \frac{\partial n_m^{(p)}}{\partial n_{m-1}^{(p)}} \cdots \frac{\partial n_1^{(p)}}{\partial \mu}, \tag{7.1}$$

where N is the number of paths and m is the number of nodes to the parameter μ. The partial derivatives of classical notes are calculated by automatic differentiation, depending on the interface used as in standard backpropagation. This is not possible for the quantum optimization nodes as these do not use standard mathematical functions with an analytical derivative. However, using the parameter shift rule as explained in section 5, the gradient of all quantum components are found by PennyLane and communicated to the used interface [14]. The quantum nodes are thus treated as black boxes by the interface. Non-Gaussian gates, such as the Kerr gate, do not have a simple parameter shift rule, and so the partial derivative of it is calculated using finite differences [37].

The codes for most numerical experiments of section III can be found in the repository: `https://github.com/martin-knudsen/masterThesis`.

8. Concrete example

To show, how PennyLane is used, one of the numerical experiments presented in section 10 is used which was inspired by [5]. This section relies heavily on the documentation from the PennyLane website [7] and the associated paper [14].

In order to use PennyLane, it needs to be imported, which is done as in listing 8.1. The importing of the NumPy interface from PennyLane allows for automatic differentiation.

```python
import pennylane as qml
from pennylane import numpy as np
from pennylane.optimize import AdamOptimizer
```

Source Code 8.1.: Import statements for importing PennyLane, the NumPy interface of PennyLane and an optimizer.

The simulated quantum computer or *device* is set using the qml.device object. This chooses the particular back end, in this case a Fock back end, which is a CV basis that allows non-Gaussian transformations. For many purposes, the *Gaussian* back end is sufficient, but not in this work because of the usage of non-Gaussian Kerr gates. The `wires` keyword sets the number of modes and `cutoff_dim` sets the maximum allowed dimension of the Fock basis (here allowing a maximum photon number of 10).

```python
dev = qml.device("strawberryfields.fock", wires=1, cutoff_dim=10)
```

Source Code 8.2.: Statement for setting the device as a Fock back end from Strawberryfields with one mode and an allowed dimension of 10 of the Fock basis.

Similar to Qiskit from IBM [19], the different gates are applied sequentially to the device. The states are initialized to the ground state, which in the Fock case is the 0 photon state (2.33). As an example of a sequence, consider the 1 mode variational circuit shown in figure 10.2. Each of the gates takes several arguments, both a gate parameter and an integer indicating the mode to which the gate should be applied. As there is only one mode in this example, all gates are applied to that wire. The argument of the Python definition is a NumPy array in this case, but this is interface dependent. Note, that the rotation gate represents a phase gate with the opposite sign of its argument following the PennyLane convention.

```python
def layer(v):
    # Linear transformation
    qml.Rotation(v[0], wires=0)
    qml.Squeezing(v[1], 0.0, wires=0)
    qml.Rotation(v[2], wires=0)

    # Bias
    qml.Displacement(v[3], 0.0, wires=0)

    # Element-wise nonlinear transformation
    qml.Kerr(v[4], wires=0)
```

Source Code 8.3.: Gates representing a 1 mode variational circuit performing a linear transformation and a non-linearity similar to a neural network. Note that the rotation gate represents a phase gate with the opposite sign of its argument.

As is, the code in listing 8.3 can't run, because it is missing the `qnode` decorator that changes the function according to the particular device used. However, it can be used as a subcircuit in a larger circuit as in listing 8.4. This circuit uses `layer` as a subcircuit and has the necessary decorator. The inputs are again two NumPy arrays, but there is an important difference: `var` is a positional argument and therefore PennyLane can calculate the gradient of the circuit with respect to its components. `x=None` is a keyword argument and is therefore treated as a constant.

As visible in figure 7.1, PennyLane allows for hybrid computations, which is necessary for many tasks including machine learning. For this problem, the loss of the circuit needs to be classically calculated, which is done with the calculation in listing 8.5. The inputs are NumPy arrays.

Next, the optimizer and parameters are initialized as can be seen in listing 8.6. The optimizer takes some keywords, such as learning rate and momentum (`beta1, beta2`).

The final optimization loop can be seen in listing 8.7.

PennyLane provides templates for popular subcircuits. Used in this thesis was mainly the `CVNeuralNetLayers` of which a single layer looks like in listing 8.8 [10].

```python
@qml.qnode(dev)
def quantum_neural_net(var, x=None):
    # Encode input x into quantum state
    qml.Displacement(x, 0.0, wires=0)

    # NN layer subcircuit
    for v in var:
        layer(v)

    # return an expectation value of the x quadrature
    return qml.expval(qml.X(0))
```

Source Code 8.4.: Qunode of PennyLane representing a circuit for the device defined in dev and with the subcircuit layer.

```python
def cost(var, features, labels):
    # output of the circuit of all points
    predictions = np.array([quantum_neural_net(var, x=x) for x in features])

    # find the square loss
    loss = 0
    for l, p in zip(labels, predictions):
        loss = loss + (l - p) ** 2

    loss = loss / len(labels)
    return loss
```

Source Code 8.5.: Classical calculation of the square loss for all points of a variational circuit.

```python
var = 0.05*3 * np.random.randn(num_layers, 5)
opt = AdamOptimizer(0.02, beta1=0.9, beta2=0.999)
```

Source Code 8.6.: Optimizer and initial parameters get defined.

```python
for it in range(steps):
    var = opt.step(lambda v: cost(v, X, Y), var)
```

Source Code 8.7.: Optimization loop consisting of updating the circuit parameters according to the defined optimizer.

```python
def cv_neural_net_layer(
    theta_1, phi_1, varphi_1, r, phi_r, theta_2, phi_2, varphi_2,
     ↪  a, phi_a, k, wires):

    Interferometer(theta=theta_1, phi=phi_1, varphi=varphi_1,
     ↪  wires=wires)

    broadcast(unitary=Squeezing, pattern="single", wires=wires,
     ↪  parameters=list(zip(r, phi_r)))

    Interferometer(theta=theta_2, phi=phi_2, varphi=varphi_2,
     ↪  wires=wires)

    broadcast(unitary=Displacement, pattern="single",
     ↪  wires=wires, parameters=list(zip(a, phi_a)))

    broadcast(unitary=Kerr, pattern="single", wires=wires,
     ↪  parameters=k)
```

Source Code 8.8.: Single CV QNN layer, reprinted from [10].

Part III.

Results and Conclusion

9. Parameter shift rule in a real circuit

The codes for all numerical experiments can be found in the repository: https://github.com/martin-knudsen/masterThesis.

9.1. Accuracy of the expectation value

As the exact value of an observable cannot be measured exactly due to quantum uncertainty, the value has to be estimated by repeated measurements, called "shots", and then the average of the measurements is taken. In PennyLane, an estimate for the expectation value is calculated using the Berry-Essen theorem [4]. To test this out, the circuit in figure 9.1 was used. A histogram of 10000 samples with the resulting expectation value can be

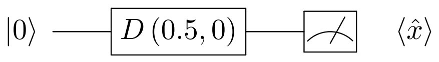


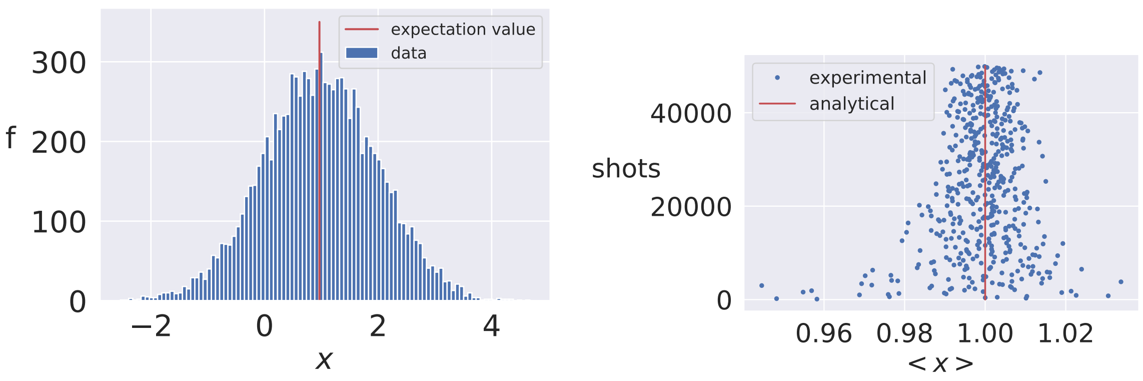
Figure 9.1.: Simple CV variational circuit that displaces the ground state by the real amount $r = 0.5$.

seen in figure 9.2a. The theoretical value of the x eigenvalue is $x = 1$, and as one can see, the resulting expectation value is close to this value. The estimation of the expectation value becomes better and better with increasing shots as can be seen in figure 9.2b. The experimental expectation value seem to follow a normal distribution about the analytical value of the x quadrature.

9.2. Single mode parameter shift

To test how the parameter shift rule would work in a physical circuit, the following function is used: $f(0) = 1$, meaning with an input in the ground state, the expectation value of $\hat{x}$ should become 1. The circuit to perform this task can be seen in figure 9.3. Analytically, the solution to this problem is to set $r = 0.5$, because that gives an $x = 1$ eigenvalue as can be seen in section 3.2.

Using mean square error as defined in equation (10.2) as the loss function, the partial



(a) Frequency of individual measurements and resulting expectation value of the $\hat{x}$ quadrature for the circuit on figure 9.3 using 10000 shots.

(b) Expectation value of $\hat{x}$ in the circuit 9.3 with $r = 0.5$ as a function of shots (opposite axes to highlight the Gaussian curve).

Figure 9.2.

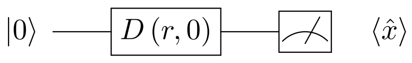


Figure 9.3.: Simple CV variational circuit with only one parametrized displacement gate with the phase set to 0.

derivative of the loss function with respect to r becomes, using the chain rule

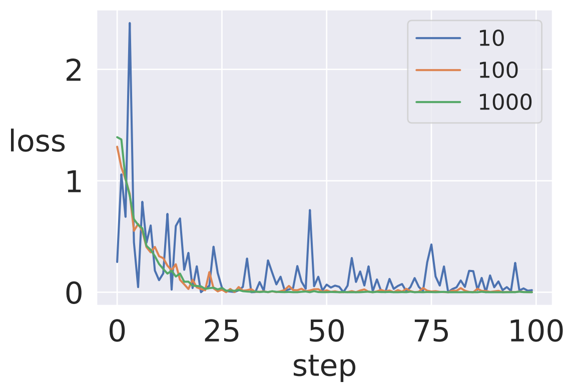
$$\begin{aligned}\nabla_r \mathrm{MSE}\left(x;r\right) &= \left(\frac{\partial \mathrm{MSE}\left(x;r\right)}{\partial f\left(x;r\right)}\right)\left(\frac{\partial f\left(x;r\right)}{\partial r}\right) \\ &= \left(-2\left(0.5 - f\left(x;r\right)\right)\right)\left(\frac{1}{2s}\left(f\left(x;r+s\right) - f\left(x;r-s\right)\right)\right),\end{aligned} \tag{9.1}$$

where the second parenthesis uses the parameter shift rule of the displacement gate (5.13). Now, simple gradient descent can be used to train the circuit [11, p. 93]

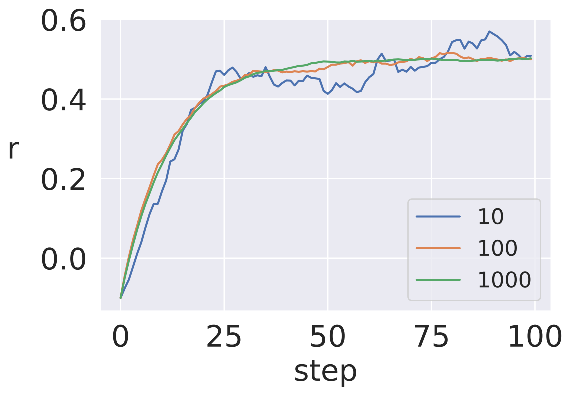
$$r = r - \lambda \nabla_r \mathrm{MSE}\left(x;r\right), \tag{9.2}$$

where λ is the learning rate. PennyLane can directly return an expectation value, so the manual calculation in section 9.1 is not necessary.

The results of this optimization can be seen in figure 9.4a, where three different number of shots were tried. The starting value of the parameter was $r = -0.1$, the learning rate set to $\lambda = 0.01$ and the parameter shift set to $s = 1$, but this is an arbitrary choice.



(a)



(b)

Figure 9.4.: MSE loss function and change in displacement parameter as a function of steps for three different number of shots to approximate the expectation value.

As one can see, the increase in shots, which leads to a better approximation of the true expectation value and therefore the true gradient, results in a faster convergence with less oscillation.

9.3. Two mode parameter shift

To show, that the parameter rule works for a more involved circuit, the circuit in figure 9.5 is used, where the fixed arguments were chosen randomly. The circuit is optimized, so that its function consisting of the sum of the two expectation values becomes $f(\mathbf{0};\theta) = \langle \hat{x}_1 \rangle + \langle \hat{x}_2 \rangle = 1$.

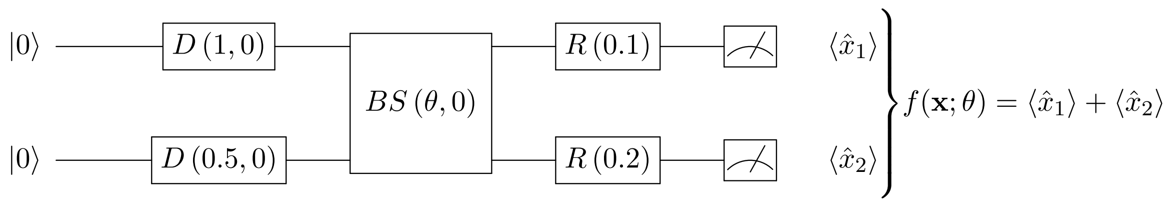

$|0\rangle$
$D(1,0)$
$|0\rangle$
$D(0.5,0)$
$BS(\theta,0)$
$R(0.1)$
$R(0.2)$
$\langle \hat{x}_1 \rangle$
$\langle \hat{x}_2 \rangle$
$f(\mathbf{x};\theta) = \langle \hat{x}_1 \rangle + \langle \hat{x}_2 \rangle$

Figure 9.5.: Simple CV variational circuit with all parameters fixed except θ of the beam splitter. The circuit function is the sum of the two x-quadrature expectation values.

The gradient is very similar to the single mode case, but this time the parameter shift rule is of the beam splitter flavor. Because the function is a simple sum and the derivative is stable over addition, the result has the same form as for the single gate case

$$\begin{aligned}\nabla_\theta \text{MSE}(\mathbf{x};\theta) &= \left(\frac{\partial \text{MSE}(\mathbf{x};\theta)}{\partial f(\mathbf{x};\theta)}\right)\left(\frac{\partial f(\mathbf{x};\theta)}{\partial \theta}\right) \\ &= \left(-2\left(1 - f(\mathbf{x};\theta)\right)\right)\left(\frac{1}{2}\left(f\left(\mathbf{x};\theta + \frac{\pi}{2}\right) - f\left(\mathbf{x};\theta - \frac{\pi}{2}\right)\right)\right).\end{aligned} \tag{9.3}$$

Using simple gradient descent as in (9.2), the loss and change in θ can be seen in figure 9.6. Here a Starting value of $\theta = 0.5$ was used, a learning rate of $\lambda = 0.01$.

As one can see, the optimization is very good for this more involved example as well, so this further corroborates the parameter shift rule for more complex circuits.

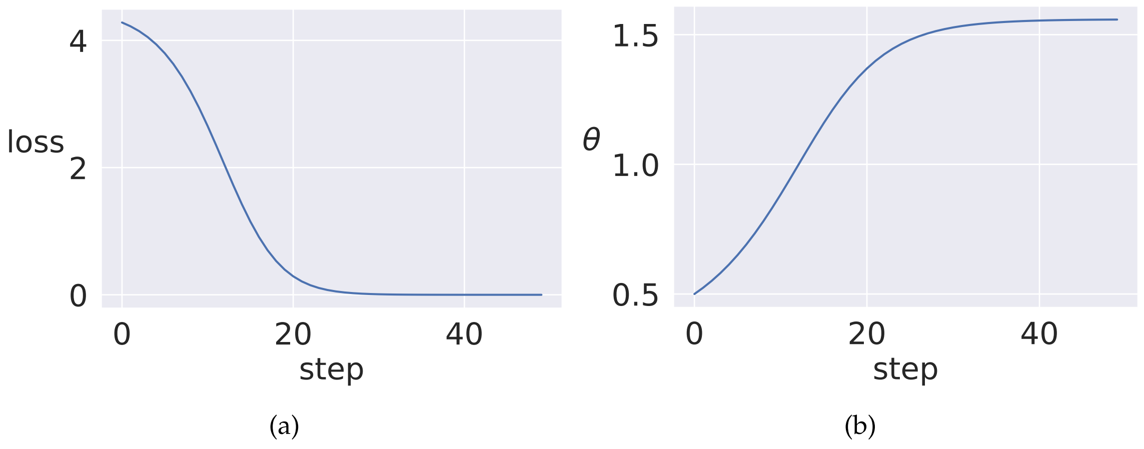


Figure 9.6.: MSE loss function and change of the θ parameter as a function of steps for the circuit 9.5.

10. 1D function regression

A simple task to test the power of variational circuits is nonlinear regression of a function with one input and one output. Because only one input and output are necessary, the natural number of modes is 1. Therefore, the architecture is a little different from the one presented in figure 6.3. A single "layer" is now the one in figure 10.1 [5]. In this case, simple phase shifts were used instead of an interferometer. This is acceptable, because for a single mode an arbitrary Gaussian gate can be defined by $U = D(\alpha)R(\theta)S(r)R(\phi)$ [41]. Using 4 of these layers, the complete transformation can be seen on figure 10.2 .

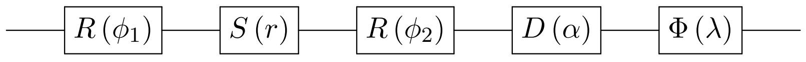


Figure 10.1.: Variational circuit implementing a 1 mode universal transformation. Adapted from [5].

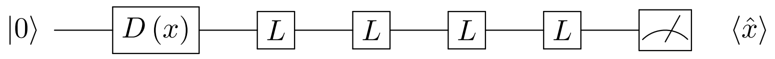


Figure 10.2.: Variational circuit for 1D nonlinear regression using 4 layers such as the one in figure 10.1.

This circuit works by first encoding the x-value using the displacement gate. The variables α, ϕ_1, r, ϕ_2 and λ for each layer are then learned, so that the expectation value of the $\hat{x}$ observable represents the function value at that point. To test this circuit, a Gaussian function with some additional random noise added was used

$$f(x) = e^{\frac{(x-\mu)^2}{2\sigma^2}} + \eta(x), \tag{10.1}$$

with $\eta(x)$ representing random noise and the values $\mu = 0$ and $\sigma = 0.3$. As a loss function, mean square error (MSE) was used, which can be defined by [11, p. 61]

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2, \tag{10.2}$$

where $\hat{y}_i$ is the prediction of the network and y_i is the actual value for the ith data point.

The result of this optimization for different steps can be seen in figure 10.3. The same settings as in [5] were used, meaning the NumPy interface, the Adam optimizer [27] and a cutoff-dimension of 10 was used and the weights were initialized randomly. For the Adam optimizer, the learning rate was set to $\lambda = 0.02$, the momenta to $\beta_1 = 0.9$ and $\beta_2 = 0.999$.

After about 1000 optimization steps, the regression was very accurate with an MSE of 0.0062507. The small "bump" after the steep decline can be explained due to the optimizer initially overshooting the minimum, because of a large step size.

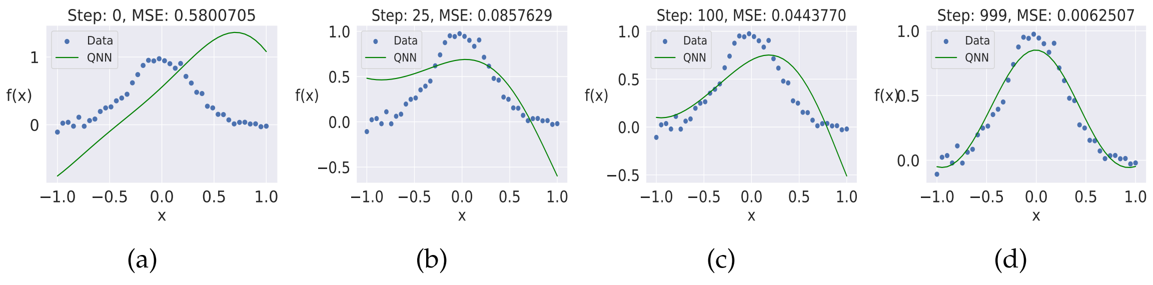


Figure 10.3.: 1D nonlinear regression of equation (10.1) using a 1 mode variational circuit showing the step and the MSE.

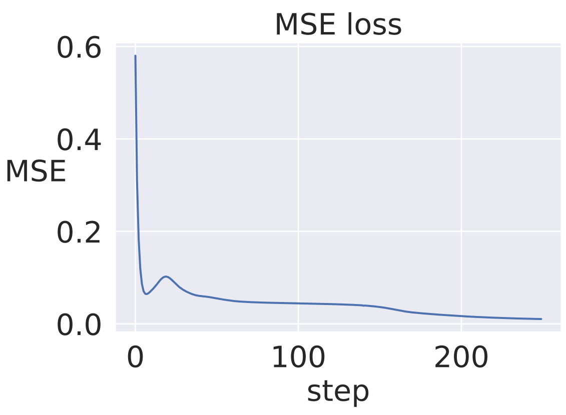


Figure 10.4.: MSE loss function of 1D regression of equation (10.1) as a function of steps.

11. 2D function approximation

In this section, a 2 mode variational circuit is trained to approximate a normalized version of the 2D Rosenbrock function [36]

$$f(x,y) = 100\left(y - x^2\right)^2 + (1-x)^2 . \tag{11.1}$$

A normalized version of this function in the interval $x, y \in [-2, 2]$ can be seen in figure 11.1.

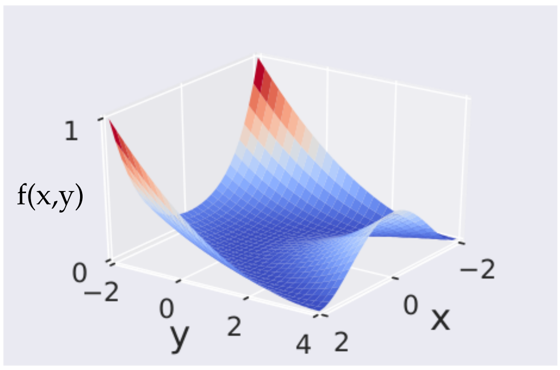


Figure 11.1.: Normalized version of the Rosenbrock test function (equation (11.1)).

The circuit used to solve this task can be seen in figure 11.2. As one can see, this circuit actually uses a neural network layer as presented in section 6.2. Importantly, the exact version used was *not* a normal neural network with real weights, but the complex version allowing mixing of the quadratures (6.8).

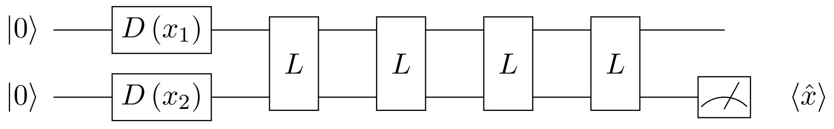
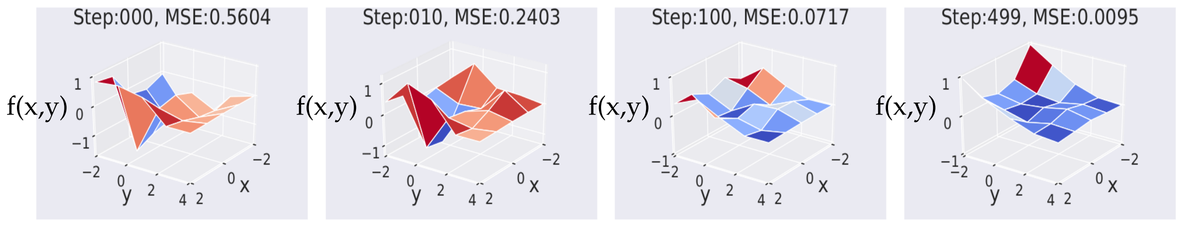


Figure 11.2.: Variational circuit for 2D function approximation using 4 2D QNN layer as defined in figure 6.3 and a measurement of the x-coordinate of the second mode.

A plot of the approximated function for different optimization steps can be seen on figure 11.3. As is evident from the figure, a coarse grid was used for the approximation, to keep runtime low. One can see, how after 500 steps the MSE is low and the function approximation seems to converge.

The settings used in the optimization were: the Numpy interface, cutoff dimension of 30, learning rate $\lambda = 0.01$, Adam optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. Due to the increase in the dimensions, from one to two modes, the computational overhead increased significantly.



(a) (b) (c) (d)

Figure 11.3.: 2D function approximation of equation (11.1) using a 2 input QNN showing the step and the mean square error (MSE).

12. Simple classification

To further test the implementation of neural networks, a simple classification problem is solved, namely the Iris dataset [21]. This dataset consists of 150 samples of flowers with 4 features distributed on 3 classes, *Iris Setosa*, *Iris Verticolor* and *Iris Virginica*. This dataset can be used as a small test for classification algorithms as each of the 4-dimensional feature vectors of each data point belongs to one of 3 classes. The features are the length and width of both the "petal" and "sepal" of the flowers.

In order to visualize the dataset and make the calculation faster, the dimension of the feature vector is decreased using Principal Component Analysis (PCA) [31]. This method reduces the dimensionality of the data by choosing the so-called *principal components*, which are orthogonal axes with the highest possible variance of the data. This is appropriate for classification because the axes with the highest variance show the most difference between the classes and is therefore the easiest to classify. To perform PCA, the Scikit-learn library was used [34]. With 2 principal components, the Iris Dataset looks like in figure 12.1. There is some overlap between the different groups. However, because 3 principal components was used for the classification, this becomes less of an issue.

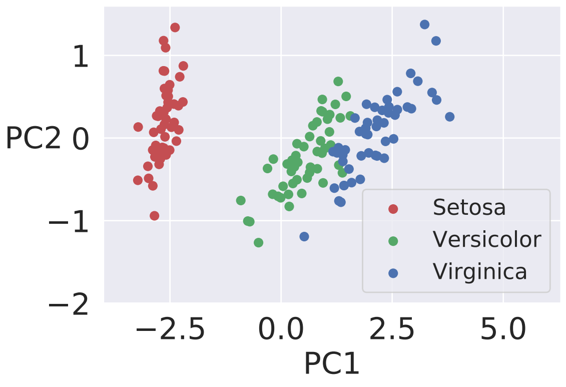


Figure 12.1.: The Iris flower dataset after PCA with 2 principal components.

In order to test how well the circuit has generalized learning outside the training dataset, it has to be split into a test dataset and a training dataset. The standard splitting from Scikit-learn is $\frac{3}{4}$ meaning 112 samples for training and 38 samples for testing and this was used. For the optimization, a quantitative measure of the success of a classification is needed. Using the *softmax* function, a kind of probability can be assigned to each class of the output vector x of the network [3]

$$f_j(x) = \frac{e^{x_j}}{\sum_k e^{x_k}}, \tag{12.1}$$

where x_i is the ith component of x. The f_j are similar to probabilities in that they are between 0 and 1 and sum to 1. Using the softmax function, the so-called cross entropy loss is obtained, which is defined for the ith data point by [3]

$$\begin{aligned} L_i(x) &= -\log f_{y_i}(x) \\ &= -\log \frac{e^{x_{y_i}}}{\sum_k e^{x_k}}, \end{aligned} \tag{12.2}$$

where y_i is the correct label for the ith data point. Therefore, the cross entropy can be understood as the negative logarithm of the softmax probability of the correct class according to the label y_i. The closer the probability of the correct class, the closer the argument within the log is to 1, the closer the log term is to 0. The negative sign comes from the fact that the denominator is always bigger than the numerator and so the log is always smaller or equal to 0. The final loss function is just the sum of all cross-entropies. Another relevant metric for this problem is the accuracy defined as the fraction of correctly labeled images

$$\text{Accuracy} = \frac{\text{number of correctly labeled images}}{\text{total number of images}}. \tag{12.3}$$

In figure 12.2, the circuit used for the solution of the classification task can be seen. Importantly, as in section 11, the actual layer used was the complex, non classical version. The output is a vector of the expectation values of each mode, that are then post processed using the cross-entropy as described above.

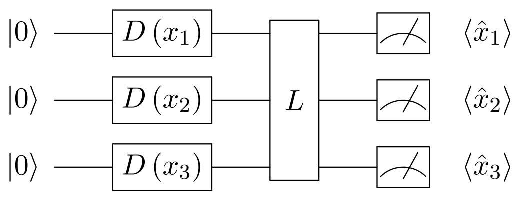
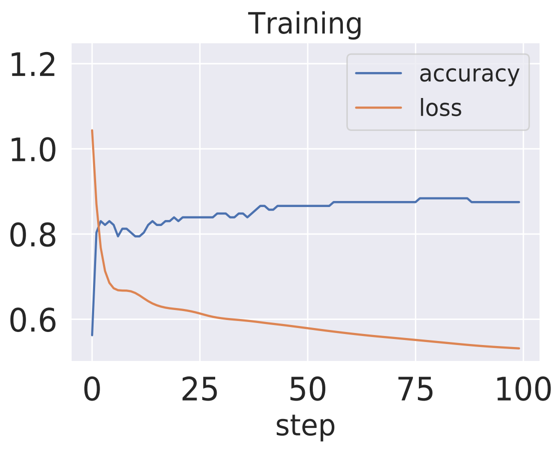
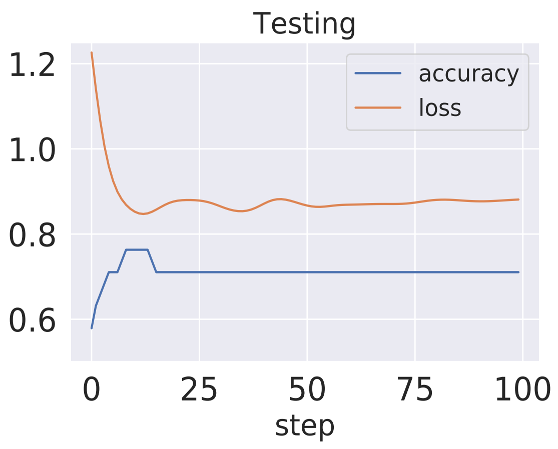

$|0\rangle$
$|0\rangle$
$|0\rangle$
$D(x_1)$
$D(x_2)$
$D(x_3)$
L
$\langle\hat{x}_1\rangle$
$\langle\hat{x}_2\rangle$
$\langle\hat{x}_3\rangle$

Figure 12.2.: 3 mode variational circuit for classification of the Iris dataset using one QNN layer and displacement encoding as well as measurement of the $\hat{x}$ quadrature of each mode.

The results of optimization can be seen in figure 12.3. The settings used for the optimization were: The PyTorch interface, cutoff dimension of 4, learning rate $\lambda = 0.03$ and the Adam optimizer. The low cutoff dimension was due to the calculation being slow, and could be an area of improvement. As one can see the accuracy is increasing in the



(a)



(b)

Figure 12.3.: Loss and accuracy for both training and testing classifying the Iris dataset using a circuit as in figure 12.2.

beginning and then flattening out and opposite for the loss. The training accuracy kept improving and ended close to 90% after about 50 steps. The generalization to the test dataset also increased to about 70% after about 10 steps. Therefore, significant learning has taken place, as the accuracy would be expected to be around $\frac{1}{3}$ if the results weren't statistically significant. The fact that the training accuracy continued improving after the test accuracy flat lined suggests that the optimization is over fitting.

13. Simple ODE

An initial value problem (IVP) of an ordinary differential equation (ODE) can be defined by [16]:

$$y'(x) = f(x, y(x)), \quad y(x_0) = y_0. \tag{13.1}$$

13.1. Classic artificial neural network ansatz

In order to solve a 1D ordinary differential equation (ODE) using a variational circuit, a classic neural network ansatz for this problem is presented in figure 13.1 [20].

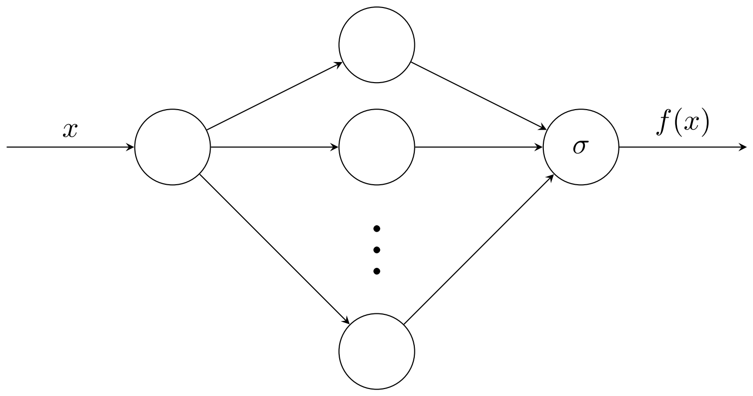


Figure 13.1.: Classic neural network approach to solving an ordinary differential equation where weights and biases are omitted. Adapted from [20].

Because in an ODE, both a function and its derivative are present, the neural network must be differentiated with respect to its input. For the ODE, the relevant loss function is therefore

$$L_{ODE} = (y(x_0) - y_0)^2 + \sum_{i=1}^{N} \left(y'(x_i) - f(x_i, y(x_i))\right)^2, \tag{13.2}$$

where (x_0, y_0) is the initial value, $y(x_i)$ is the function value of the current solution at point x_i, $y'(x_i)$ is the derivative of y in the point x_i and $f(x_i, y(x_i))$ is the right-hand side of the ODE according to the definition (13.1).

Notice the contrast to normal backpropagation of a neural network where the output of the network is optimized (differentiated) with respect to the weights in order to optimize these. To solve an ODE, both the partial derivative of the output with respect to the input *and* the weights are needed. To test the setup, this IVP is solved

$$\frac{\mathrm{d}y}{\mathrm{d}x} = -2xy, \quad y\left(0\right) = 1, \tag{13.3}$$

on the interval $[-1, 1]$.

For comparison, the general solution can be found via separation of variables

$$\begin{aligned} \frac{1}{y}\,\mathrm{d}y &= -2x\,\mathrm{d}x \\ \int \frac{1}{y}\,\mathrm{d}y &= -2\int x\,\mathrm{d}x \\ \ln\left(y\right) &= -x^2 + c_1 \\ y\left(x\right) &= c_2 e^{-x^2}. \end{aligned} \tag{13.4}$$

Inserting the initial value, the value of the constant is found to be $c_2 = 1$. The theoretical solution for this particular IVP is therefore

$$y\left(x\right) = e^{-x^2}. \tag{13.5}$$

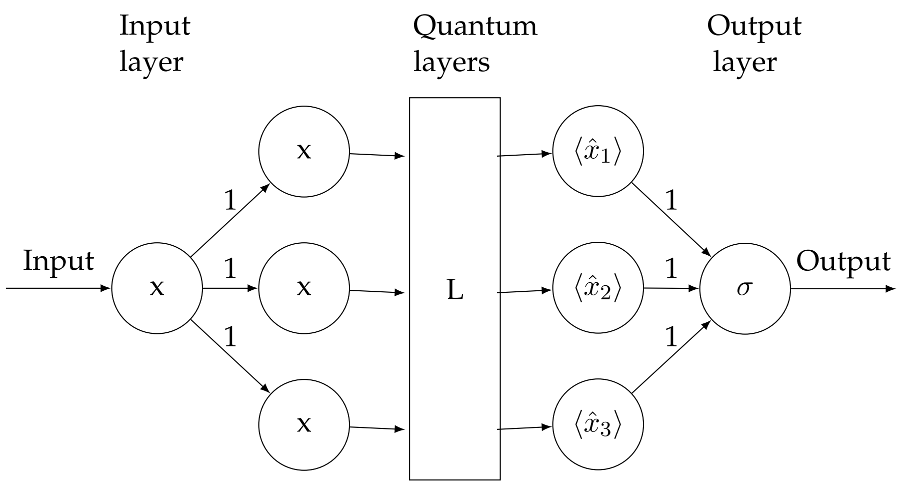
13.2. Neural network inspired variational circuit solution

Inspired by the classical approach, the circuit shown in figure 13.2 is used. As in the other sections, the complex neural network version was used. The activation function used was a sigmoid activation function which is appropriate for this case, because the real solution does not have a negative function value in the chosen interval.

To solve the circuit, the PyTorch interface was used. Several different settings were tried for best possible convergence and the best setting was: learning rate 0.03 and the Adam optimizer, cutoff dimension set to 4. The gradient of the network with respect to its inputs was achieved using PyTorch' automatic differentiation feature, *autograd*.

Several steps in the optimization can be seen in figure 13.3 and the resulting loss curve can be seen in figure 13.4. The circuit did not converge every time but this could be a matter of choosing the right hyper parameters.

As one can observe, the optimization pattern is different when compared to previous tasks. A possible explanation for this behavior is, that there are two sometimes competing interests in the optimization: the differential equation (13.3) and the initial value. Therefore, at step 10 one can see the initial value has been optimized but the differential equation is not that accurate. In step 60, the differential equation (shape) is more fulfilled but the



Input layer
Quantum layers
Output layer
Input
x
x
x
x
1
1
1
L
$\langle\hat{x}_1\rangle$
$\langle\hat{x}_2\rangle$
$\langle\hat{x}_3\rangle$
1
1
1
σ
Output

Figure 13.2.: Hybrid variational circuit for solving an ordinary differential equation. The input is broadcasted to each mode before entering the QNN layer and is post processed by a summation of the expectation values and a sigmoid activation function.

initial value is quite off. Finally, in step 130, both the differential equation and the initial value are fulfilled.

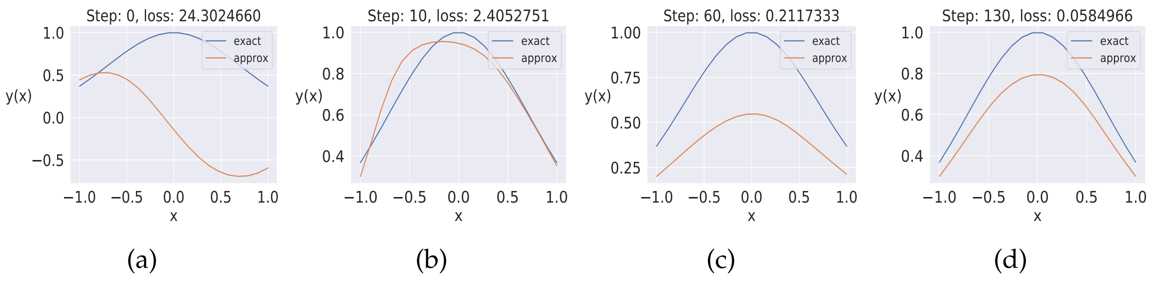


Figure 13.3.: Solution of the ODE (13.3) on a QNN as in figure 13.2.

To test the network on a more involved example, this first order nonlinear ODE was solved [39]

$$y' = x^2 + y^2 - 1\,, \quad y\,(0) = 0. \tag{13.6}$$

This is a so-called Riccati equation [33] and was solved numerically for comparison by using the *odeint* function of the SciPy library [40].

Because of the higher level of complexity, the optimization had difficulty finding the minimum. Therefore, the network to solve this problem was simplified to have fewer free

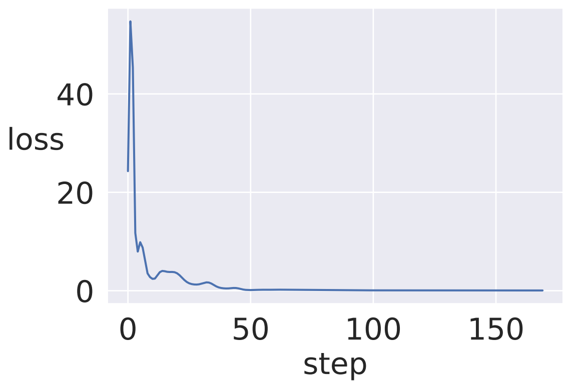


Figure 13.4.: Loss of the ODE in equation (13.3) solved on a QNN as a function of steps.

parameters but still use the complex version of the QNN, as one can see in figure 13.5. This setup used only had 2 modes, a single layer and no *classical* activation function, but just a simple sum of the outputs in the end.

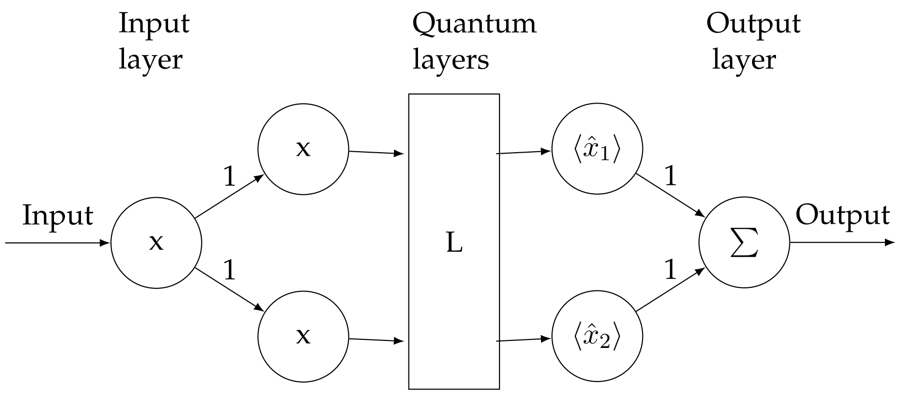
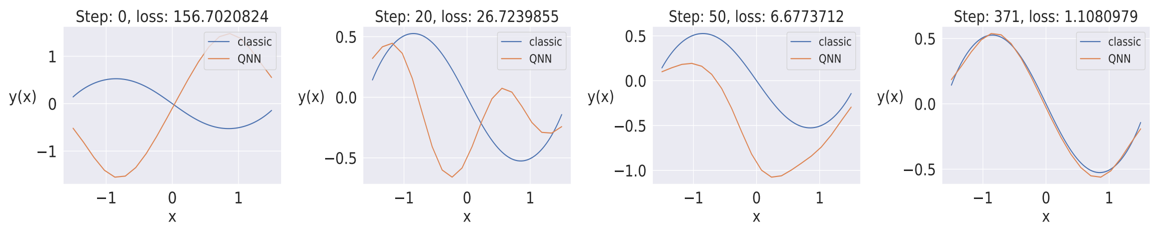


Figure 13.5.: Simplified quantum circuit for solving an ordinary differential equation using only two modes and no classical activation function. The input is broadcasted to each mode before entering the QNN layer and is post processed by a summation of the expectation values.

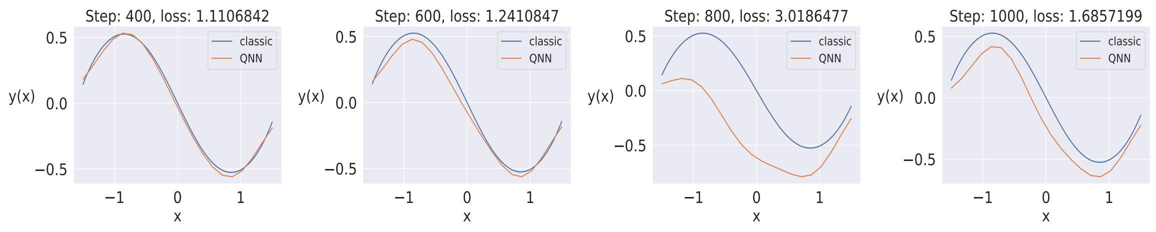
To solve the circuit, the PyTorch interface was used. The result of the optimization using the Adam optimization strategy and a learning rate of $\lambda = 0.02$ can be seen in figure 13.6. As one can see, the final approximation is very good. One can again see the competing interests of the initial value and the shape of the function.

After finding the minimum in step 371 the loss started to increase again and then oscillate, as can be seen in figure 13.7.



(a) (b) (c) (d)

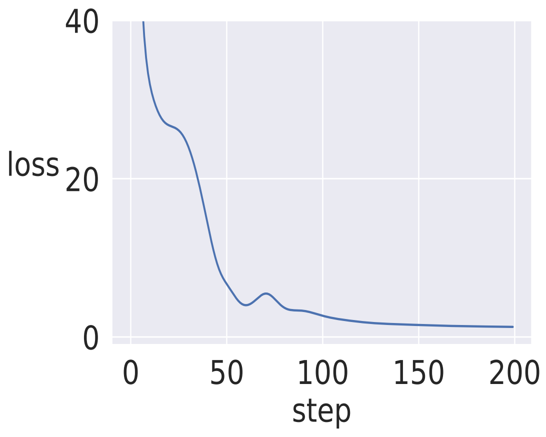
Figure 13.6.: Solution of the ODE (13.6) using the variational circuit from figure 13.5.



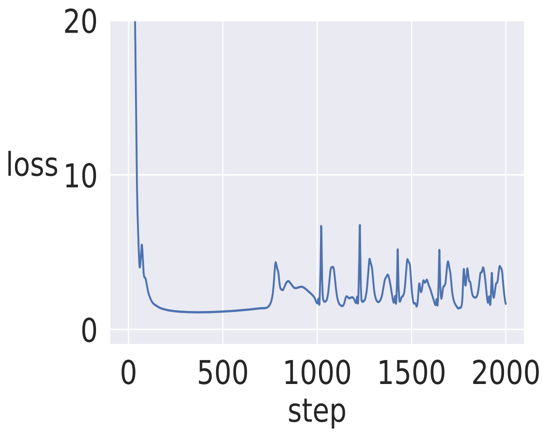
(a) (b) (c) (d)

Figure 13.7.: Divergence of the solution of the ODE (13.6) on a QNN as in figure 13.5 in high iterations.

A graph of the loss function can be seen in figure 13.8. As one can see, the oscillation starts at around step 750 and continuous throughout. One way to explain this is, that the Adam optimizer tends to blow up, if the loss becomes small [1]. To counter this, the accepted error of Adam was increased, as suggested in [1] but this did not discard the oscillation pattern. Another reason, is that the loss landscape might not be convex and therefore have local minima, that are entered after some iterations. Nonetheless, one could choose the parameters of the circuit to be the ones with the smallest loss.



(a)



(b)

Figure 13.8.: Loss of the optimization of the ODE in equation (13.6), solved using the circuit in figure 13.5.

14. Convolutional neural network

Here, a way to realize a periodic convolutional neural network (CNN) on a CV quantum computer is introduced. A CNN on an optical CV quantum computer has already been proposed [26], but here a concrete way to implement it is presented.

A periodic 1D convolution can be represented by a simple circulant matrix [26], which looks like [13]

$$C = \begin{bmatrix} c_1 & c_2 & c_3 & \dots & c_n \\ c_n & c_1 & c_2 & & c_{n-1} \\ c_{n-1} & c_n & c_1 & & \\ \vdots & & & \ddots & \vdots \\ c_2 & c_3 & & \dots & c_1 \end{bmatrix}. \tag{14.1}$$

The similarity between this and the well-known 2D convolution becomes apparent, if all but the first two entries are to 0

$$C = \begin{bmatrix} c_1 & c_2 & 0 & \dots & 0 \\ 0 & c_1 & c_2 & & 0 \\ 0 & 0 & c_1 & c_2 & 0 \\ \vdots & & & \ddots & \vdots \\ c_2 & 0 & & \dots & c_1 \end{bmatrix}. \tag{14.2}$$

When this matrix meets an input vector, every entry becomes the weighted average of itself and its neighboring point. This is the 1D equivalent of a 2×2 (periodic) filter in a 2D convolutional neural network.

Using a circulant matrix is convenient, because it can be diagonalized by the discrete Fourier transform [13]

$$C = \frac{1}{n} F \text{diag}(F\mathbf{c}) F^\dagger, \tag{14.3}$$

where F is the Fourier matrix, diag creates a diagonal matrix of a vector and $\mathbf{c}$ is the first row of C as a column vector. Therefore, a 1D convolution can be understood as transforming the vector to Fourier space, followed by a scaling with the Fourier coefficients of the column vector and ending with a transformation back to normal space. In order to optimize the convolution, the values c_i need to be updated. F stays the same and one only has to calculate the Fourier transform of the row c in every step, which can be achieved using the fast Fourier transform [18].

If the Fourier transformed row of C is set to $\hat{\mathbf{a}} = F\mathbf{c}$, then multiplication with the diagonal matrix results in a complex scaling $\hat{a}_j$ of each component of the Fourier transformed vector. In polar coordinates, this scaling can be written by

$$\hat{a}_j = t_j e^{i\phi_j}, \tag{14.4}$$

where t_j is the modulus and ϕ_j is the phase.

To implement such a transformation on a CV quantum computer, a circuit such as in figure 14.1 can be build, where besides the convolution there is a bias and a nonlinearity as is custom for convolutional neural networks.

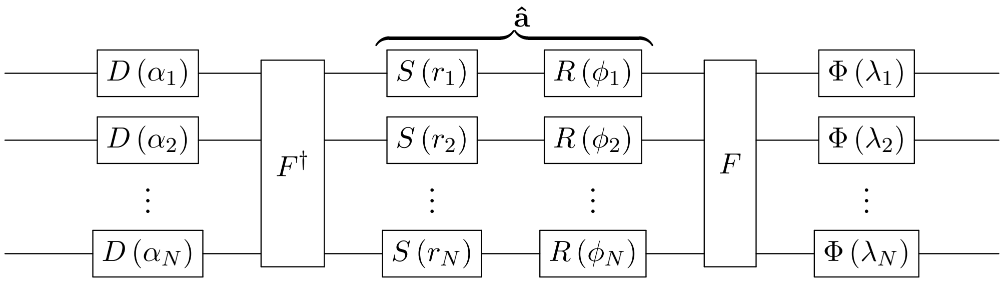


Figure 14.1.: Variational circuit proposal for a convolutional neural network layer on a CV quantum computer consisting of two interferometers as well as one displacement, squeeze, phase shift and Kerr gate for each wire.

Because the effect of the squeeze gate on the $\hat{x}$ quadrature is to scale by the negative exponential of its argument (see eq. (3.22)), its argument has to be

$$\begin{aligned} e^{-r_j} &= t_j \\ r_j &= -\log t_j. \end{aligned} \tag{14.5}$$

The circuit as represented in figure 14.1 was implemented, but the resulting convolution was not accurate. After consulting with the team behind PennyLane, the probable explanation for this, is that squeezing has to be close to 1, because a big scaling results in the state going into a Fock state that might be outside the domain of the simulation (cutoff dimension might be too small). If one wants to use a convolution such as the one from equation (14.2), the question now becomes which constraints on c_1 and c_2 lead to a scaling close to 1. Because the connection between the c_i values and the resulting scaling involves a Fourier transform, the constraints are connected in a non-trivial way. This connection can be seen in figure 14.2, where the infinity norm of the difference between the scaling and a scaling of 1 for all wires is plotted for various values of c_1 and c_2. From this figure, one can see, that the allowed convolutions of the type (14.2) are $|c_1| \approx 0$ and $|c_2| \approx 1$ or vice versa. In conclusion, only convolutions that almost don't have an effect or almost periodically shift each component of the input vector with its neighboring point are allowed. This severely decreases the set of permissible convolutions and therefore handicaps this approach.

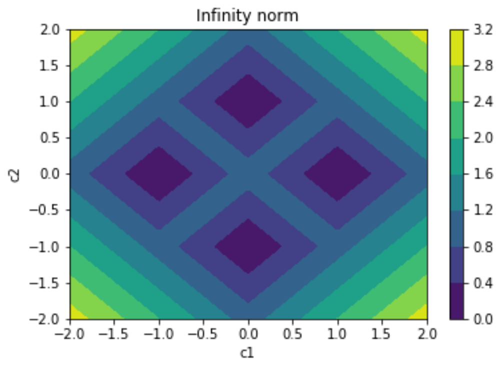


Figure 14.2.: Distance from a scaling of 1 (norm) as a function of convolution coefficients. The closer scaling is to 1, the less the simulation error.

While this method was not simulated successfully, it could in theory be used in a real device where cutoff dimension isn't a problem.

15. Conclusion

Continuous-variable quantum computing has the benefit over the qubit paradigm, that real numbers can easily be represented. This is a big advantage for functions requiring a real input such as neural networks. Such networks can be build using basic optical equipment such as beam splitters and phase changers. These advantages combined make the CV computer a strong candidate for a commercial quantum computer.

In this work, variational circuits were simulated using the PennyLane simulation framework and successfully used to solve several machine learning tasks, such as regression, classification and finding the solution of an ordinary differential equation.

1D regression was performed by a single mode variational circuit that converged, a task that has previously been performed [26]. 2D function approximation using a CV neural network was also performed using a normalized Rosenbrock function and this also converged successfully.

Classification of the Irish flower data set was also successfully performed, however, with only a test-accuracy of 70 %, that is nonetheless statistically significant. Here, a 3 mode variational circuit inspired by a classic neural network architecture was used.

A linear ordinary differential equation was solved using a hybrid computation with a 3 mode circuit inspired by a classic neural network and using a classical sigmoid function for post-processing. A non-linear ordinary differential equation was solved using a simpler 2 mode circuit and no classical activation function. This indicates, that simpler circuits might not only decrease the computational time but also solve the problem more accurately. This is an important insight for quantum computers in general as they are still primitive and the overhead of adding extra modes or qubits is still very high.

Finally, a CV circuit performing a convolutional neural network has previously been proposed [26], but here a way to practically implement this transformation is proposed using the convolution theorem. However, the circuit was not successfully implemented due to uncertainty in the scaling (squeeze gate). As the problem is of the PennyLane simulation, this method could still work in a physical circuit.

Considering, that the numerical experiments conducted for this thesis perform "classical" calculations that can be more easily performed on a classical computer, this is more of a proof of concept than an actual suggestion for quantum supremacy. Future work could focus on how to utilize superposition for computational gain. As all simulation were run locally, another immediate suggestion for future work is to run on a supercomputer to increase the complexity and accuracy of the simulated circuits. Improved performance of the methods used can also be achieved by tuning the hyper parameters. It would also be interesting to verify the circuits in real physical experiments when this is possible.

Appendix

A. Detailed descriptions

A.1. Light Hamiltonian as a sum of quantum harmonic oscillators

the Hamiltonian can be rewritten by first realizing the relationship

$$\begin{aligned}\hat{\mathbf{A}} \cdot \frac{\partial \hat{\mathbf{D}}}{\partial t} &= \hat{\mathbf{A}} \cdot \left(\nabla \times \hat{\mathbf{H}}\right) \\ &= \hat{\mathbf{H}} \cdot \left(\nabla \times \hat{\mathbf{A}}\right) - \nabla \cdot \left(\hat{\mathbf{A}} \times \hat{\mathbf{H}}\right) \\ &= \hat{\mathbf{H}} \cdot \hat{\mathbf{B}} - \nabla \cdot \left(\hat{\mathbf{A}} \times \hat{\mathbf{H}}\right),\end{aligned} \tag{A.1}$$

using Maxwell's equations (2.1) in the first step, a vector identity in the second step [38, p. 123] and the definition of the vector potential (2.7) in the third step. Using this relation and the fact that $\hat{\mathbf{H}}$ and $\hat{\mathbf{B}}$ must commute due to the constitutive equations (2.5), 0he Hamiltonian can be rewritten as

$$\begin{aligned}\hat{H} &= \frac{1}{2}\int \left(\hat{\mathbf{E}} \cdot \hat{\mathbf{D}} + \hat{\mathbf{A}} \cdot \frac{\partial \hat{\mathbf{D}}}{\partial t} + \nabla \cdot \left(\hat{\mathbf{A}} \times \hat{\mathbf{H}}\right)\right) \mathrm{d}V \\ &= \frac{1}{2}\int \left(\hat{\mathbf{E}} \cdot \hat{\mathbf{D}} + \hat{\mathbf{A}} \cdot \frac{\partial \hat{\mathbf{D}}}{\partial t}\right) \mathrm{d}V,\end{aligned} \tag{A.2}$$

where the integral of $\nabla \cdot \left(\hat{\mathbf{A}} \times \hat{\mathbf{H}}\right)$ is zero, because due to the divergence theorem, this is equal to a surface integral of $\hat{\mathbf{A}} \times \hat{\mathbf{H}}$ at infinity (because the volume is over all of space) and here it is 0 [29, p. 22].

Monochromatic modes that only oscillates at single frequencies ω_k are described by [29, p. 25]

$$\mathbf{A}_k(\mathbf{r}, t) = \mathbf{A}_k(\mathbf{r}) e^{-i\omega_k t}. \tag{A.3}$$

For monochromatic modes, the inner product (2.12) becomes

$$\begin{aligned}\langle \mathbf{A}_i|\mathbf{A}_j\rangle &= -\frac{\varepsilon_0\varepsilon}{i\hbar}\int\left(\mathbf{A}_i^*\cdot\frac{\partial\mathbf{A}_j}{\partial t}-\mathbf{A}_j\cdot\frac{\partial\mathbf{A}_i^*}{\partial t}\right)\mathrm{d}V\\ &= \frac{\varepsilon_0\varepsilon}{\hbar}\int\left(\mathbf{A}_i^*\cdot\mathbf{A}_j\omega_j+\mathbf{A}_j\cdot\mathbf{A}_i^*\omega_i\right)\mathrm{d}V\\ &= \delta_{ij},\end{aligned} \tag{A.4}$$

and

$$\begin{aligned}\langle \mathbf{A}_i^*|\mathbf{A}_j\rangle &= -\frac{\varepsilon_0\varepsilon}{i\hbar}\int\left(\mathbf{A}_i\cdot\frac{\partial\mathbf{A}_j}{\partial t}-\mathbf{A}_j\cdot\frac{\partial\mathbf{A}_i}{\partial t}\right)\mathrm{d}V\\ &= \frac{\varepsilon_0\varepsilon}{\hbar}\int\left(\mathbf{A}_i\cdot\mathbf{A}_j\omega_j-\mathbf{A}_j\cdot\mathbf{A}_i\omega_i\right)\mathrm{d}V\\ &= 0,\end{aligned} \tag{A.5}$$

and

$$\begin{aligned}\langle \mathbf{A}_i|\mathbf{A}_j^*\rangle &= -\frac{\varepsilon_0\varepsilon}{i\hbar}\int\left(\mathbf{A}_i^*\cdot\frac{\partial\mathbf{A}_j^*}{\partial t}-\mathbf{A}_j^*\cdot\frac{\partial\mathbf{A}_i^*}{\partial t}\right)\mathrm{d}V\\ &= \frac{\varepsilon_0\varepsilon}{\hbar}\int\left(\mathbf{A}_j^*\cdot\mathbf{A}_i^*\omega_i-\mathbf{A}_i^*\cdot\mathbf{A}_j^*\omega_j\right)\mathrm{d}V\\ &= 0,\end{aligned} \tag{A.6}$$

Using the constitutive (2.5), the mode expansion (2.11) and the definition of the vector potential (2.7), $\hat{\mathbf{E}}\cdot\hat{\mathbf{D}}$ can be written for monochromatic modes as

$$\begin{aligned}\hat{\mathbf{E}}\cdot\hat{\mathbf{D}} &= \left(\sum_k \omega_k\mathbf{A}_k(\mathbf{r},t)\hat{a}_k-\omega_k\mathbf{A}_k^*(\mathbf{r},t)\hat{a}_k^\dagger\right)\cdot\left(\varepsilon_0\varepsilon\sum_{k'}\omega_{k'}\mathbf{A}_{k'}(\mathbf{r},t)\hat{a}_{k'}-\omega_{k'}\mathbf{A}_{k'}^*(\mathbf{r},t)\hat{a}_{k'}^\dagger\right)\\ &= -\varepsilon_0\varepsilon\sum_{kk'}\omega_k\omega_{k'}\left(\mathbf{A}_k(\mathbf{r},t)\hat{a}_k-\mathbf{A}_k^*(\mathbf{r},t)\hat{a}_k^\dagger\right)\cdot\left(\mathbf{A}_{k'}(\mathbf{r},t)\hat{a}_{k'}-\mathbf{A}_{k'}^*(\mathbf{r},t)\hat{a}_{k'}^\dagger\right),\end{aligned} \tag{A.7}$$

and the same for $\hat{\mathbf{A}}\cdot\frac{\partial\hat{\mathbf{D}}}{\partial t}$

$$\begin{aligned}\hat{\mathbf{A}}\cdot\frac{\partial\hat{\mathbf{D}}}{\partial t} &= \left(\sum_k \mathbf{A}_k(\mathbf{r},t)\hat{a}_k+\mathbf{A}_k^*(\mathbf{r},t)\hat{a}_k^\dagger\right)\cdot\left(\varepsilon_0\varepsilon\sum_{k'}\omega_{k'}^2\mathbf{A}_{k'}(\mathbf{r},t)\hat{a}_{k'}+\omega_{k'}^2\mathbf{A}_{k'}^*(\mathbf{r},t)\hat{a}_{k'}^\dagger\right)\\ &= \varepsilon_0\varepsilon\sum_{kk'}\omega_{k'}^2\left(\mathbf{A}_k(\mathbf{r},t)\hat{a}_k+\mathbf{A}_k^*(\mathbf{r},t)\hat{a}_k^\dagger\right)\cdot\left(\mathbf{A}_{k'}(\mathbf{r},t)\hat{a}_{k'}+\mathbf{A}_{k'}^*(\mathbf{r},t)\hat{a}_{k'}^\dagger\right),\end{aligned} \tag{A.8}$$

This can be substituted into the light Hamiltonian (A.2)

$$\begin{aligned}
\hat{H} =&\frac{\varepsilon_0\varepsilon}{2}\sum_{kk'}\omega_{k'}\int -\omega_k\left(\mathbf{A}_k\hat{a}_k - \mathbf{A}_k^*\hat{a}_k^\dagger\right)\cdot\left(\mathbf{A}_{k'}\hat{a}_{k'} - \mathbf{A}_{k'}^*\hat{a}_{k'}^\dagger\right)+\\
&\omega_{k'}\left(\mathbf{A}_k\hat{a}_k + \mathbf{A}_k^*\hat{a}_k^\dagger\right)\cdot\left(\mathbf{A}_{k'}\hat{a}_{k'} + \mathbf{A}_{k'}^*\hat{a}_{k'}^\dagger\right)\mathrm{d}V\\
=&\frac{\varepsilon_0\varepsilon}{2\hbar}\sum_{kk'}\omega_{k'}\hbar\int \hat{a}_k\hat{a}_{k'}(\mathbf{A}_k\mathbf{A}_{k'}\omega_{k'} - \mathbf{A}_k\mathbf{A}_{k'}\omega_k)\hat{a}_k + \hat{a}_{k'}^\dagger(\mathbf{A}_k\mathbf{A}_{k'}^*\omega_k + \mathbf{A}_k\mathbf{A}_{k'}^*\omega_{k'})+\\
&\hat{a}_k^\dagger\hat{a}_{k'}(\mathbf{A}_k^*\mathbf{A}_{k'}\omega_k + \mathbf{A}_k^*\mathbf{A}_{k'}\omega_{k'}) + \hat{a}_k^\dagger\hat{a}_{k'}^\dagger(\mathbf{A}_k^*\mathbf{A}_{k'}^*\omega_{k'} - \mathbf{A}_k^*\mathbf{A}_{k'}^*\omega_k)\,\mathrm{d}V\\
=&\sum_{kk'}\frac{\omega_{k'}\hbar}{2}(\hat{a}_k\hat{a}_{k'}\left\langle\mathbf{A}_k^*|\mathbf{A}_{k'}\right\rangle + \hat{a}_k\hat{a}_{k'}^\dagger\left\langle\mathbf{A}_{k'}|\mathbf{A}_k\right\rangle+\\
&\hat{a}_k^\dagger\hat{a}_{k'}\left\langle\mathbf{A}_k|\mathbf{A}_{k'}\right\rangle + \hat{a}_k^\dagger\hat{a}_{k'}^\dagger\left\langle\mathbf{A}_k|\mathbf{A}_{k'}^*\right\rangle).
\end{aligned} \tag{A.9}$$

Looking at the assumptions for normal modes (2.14), the Hamiltonian becomes

$$\begin{aligned}
\hat{H} =&\sum_{k'}\frac{\omega_{k'}\hbar}{2}\left(\hat{a}_{k'}\hat{a}_{k'}^\dagger + \hat{a}_{k'}^\dagger\hat{a}_{k'}\right)\\
=&\sum_{k'}\frac{\omega_{k'}\hbar}{2}\left([\hat{a}_{k'},\hat{a}_{k'}^\dagger] + 2\hat{a}_{k'}^\dagger\hat{a}_{k'}\right)\\
=&\sum_{k'}\omega_{k'}\hbar\left(\frac{1}{2} + \hat{a}_{k'}^\dagger\hat{a}_{k'}\right),
\end{aligned} \tag{A.10}$$

using the Bose commutation relations (2.15).

A.2. Unitarity of a beam splitter transformation for normal modes

To reiterate, the effect of a beam splitter on the annihilation operators can be written by

$$\begin{aligned}
\begin{bmatrix}\hat{a}_1'\\ \hat{a}_2'\end{bmatrix} &= B\begin{bmatrix}\hat{a}_1\\ \hat{a}_2\end{bmatrix}\\
&= \begin{bmatrix}B_{11} & B_{12}\\ B_{21} & B_{22}\end{bmatrix}\begin{bmatrix}\hat{a}_1\\ \hat{a}_2\end{bmatrix}.
\end{aligned} \tag{A.11}$$

Following the derivation in [29, p. 94]

$$\begin{aligned}1 &= [\hat{a}'_1, \hat{a}'^{\dagger}_1] \\ &= (B_{11}\hat{a}_1 + B_{12}\hat{a}_2)\left(B^*_{11}\hat{a}^\dagger_1 + B^*_{12}\hat{a}^\dagger_2\right) - \left(B^*_{11}\hat{a}^\dagger_1 + B^*_{12}\hat{a}^\dagger_2\right)(B_{11}\hat{a}_1 + B_{12}\hat{a}_2) \\ &= |B_{11}|^2[\hat{a}_1, \hat{a}^\dagger_1] + B_{11}B^*_{12}[\hat{a}_1, \hat{a}^\dagger_2] + B_{12}B^*_{11}[\hat{a}_2, \hat{a}^\dagger_1] + |B_{22}|^2[\hat{a}_2, \hat{a}^\dagger_2] \\ &= |B_{11}|^2 + |B_{12}|^2.\end{aligned} \tag{A.12}$$

Similarly, for $\hat{a}'_2$, it is apparent that $|B_{21}|^2 + |B_{22}|^2 = 1$. From the relation $[\hat{a}_1, \hat{a}^\dagger_2] = 0$ it can be deduced that

$$\begin{aligned}0 &= [\hat{a}'_1, \hat{a}'^{\dagger}_2] \\ &= (B_{11}\hat{a}_1 + B_{12}\hat{a}_2)\left(B^*_{21}\hat{a}^\dagger_1 + B^*_{22}\hat{a}^\dagger_2\right) - \left(B^*_{21}\hat{a}^\dagger_1 + B^*_{22}\hat{a}^\dagger_2\right)(B_{11}\hat{a}_1 + B_{12}\hat{a}_2) \\ &= B_{11}B^*_{21}[\hat{a}_1, \hat{a}^\dagger_1] + B_{11}B^*_{22}[\hat{a}_1, \hat{a}^\dagger_2] + B_{12}B^*_{21}[\hat{a}_2, \hat{a}^\dagger_1] + B_{12}B^*_{22}[\hat{a}_2, \hat{a}^\dagger_2] \\ &= B_{11}B^*_{21} + B_{12}B^*_{22}.\end{aligned} \tag{A.13}$$

Similarly for the reverse order, the result is $B_{21}B^*_{11} + B_{22}B^*_{12} = 0$. These two restrictions mean that the transformation B must be unitary, as one can see by performing the matrix multiplication of B with its hermitian conjugate

$$\begin{aligned}BB^\dagger &= \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22}\end{bmatrix}\begin{bmatrix} B^*_{11} & B^*_{21} \\ B^*_{12} & B^*_{22}\end{bmatrix} \\ &= \begin{bmatrix} |B_{11}|^2 + |B_{12}|^2 & B_{11}B^*_{21} + B_{12}B^*_{22} \\ B_{21}B^*_{11} + B_{22}B^*_{12} & |B_{21}|^2 + |B_{22}|^2\end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 \\ 0 & 1\end{bmatrix},\end{aligned} \tag{A.14}$$

and $BB^\dagger = B^\dagger B$ as one can easily verify.

Bibliography

[1] Adam optimizer goes haywire after 200k batches, training loss grows. https://stackoverflow.com/questions/42327543/adam-optimizer-goes-haywire-after-200k-batches-training-loss-grows?rq=1. Accessed: 2020-08-20.

[2] Cs231n: Convolutional neural networks for visual recognition. https://cs231n.github.io/neural-networks-1/. Accessed: 2020-07-23.

[3] Cs231n: Convolutional neural networks for visual recognition. https://cs231n.github.io/linear-classify/. Accessed: 2020-08-08.

[4] The fock device. https://brilliant.org/wiki/small-angle-approximation/. Accessed: 2020-09-27.

[5] Function fitting with a quantum neural network. https://pennylane.ai/qml/demos/quantum_neural_net.html. Accessed: 2020-06-08.

[6] Parameter-shift rules. https://pennylane.ai/qml/glossary/parameter_shift.html. Accessed: 2020-08-27.

[7] Pennylane documentation, lecture notes. https://pennylane.readthedocs.io/en/stable/. Accessed latest: 2020-10-03.

[8] Quantum theory of radiation interactions, fall 2012, lecture notes. https://ocw.mit.edu/courses/nuclear-engineering/22-51-quantum-theory-of-radiation-interactions-fall-2012/lecture-notes/MIT22_51F12_Ch5.pdf. Accessed: 2020-09-28.

[9] Small-angle approximation. https://pennylane-sf.readthedocs.io/en/latest/devices/fock.html. Accessed: 2020-09-04.

[10] Source code for pennylane.templates.layers.cv_neural_net. https://pennylane.readthedocs.io/en/stable/_modules/pennylane/templates/layers/cv_neural_net.html#CVNeuralNetLayers. Accessed: 2020-10-03.

[11] Yaser S. Abu-Mostafa, Malik Magdon-Ismail, and Hsuan-Tien Lin. *Learning From Data - A Short Course*. AMLBook, 2012.

[12] Hans-A Bachor and Timothy Ralph. *A Guide to Experiments in Quantum Optics*. 3rd edition, 2019.

[13] Bassam Bamieh. Discovering transforms: A tutorial on circulant matrices, circular convolution, and the discrete fourier transform, 2018.

[14] Ville Bergholm, Josh Izaac, Maria Schuld, Christian Gogolin, M. Sohaib Alam, Shahnawaz Ahmed, Juan Miguel Arrazola, Carsten Blank, Alain Delgado, Soran Jahangiri, Keri McKiernan, Johannes Jakob Meyer, Zeyue Niu, Antal Száva, and Nathan Killoran. Pennylane: Automatic differentiation of hybrid quantum-classical computations, 2018.

[15] Holger Boche and Volker Pohl. Turing meets circuit theory: Not every continuous-time lti system can be simulated on a digital computer. *IEEE Transactions on Circuits and Systems I: Regular Papers*, PP:1–14, 08 2020.

[16] Rainer Callies. Numerical programming 2 lecture notes. Summer Term 2019.

[17] William R. Clements, Peter C. Humphreys, Benjamin J. Metcalf, W. Steven Kolthammer, and Ian A. Walmsley. Optimal design for universal multiport interferometers. *Optica*, 3(12):1460–1465, Dec 2016.

[18] J. Cooley and John W. Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation*, 19:297–301, 1965.

[19] Héctor Abraham et al. Qiskit: An open-source framework for quantum computing, 2019.

[20] Diogo R. Ferreira. How to solve an ode with a neural network. https://towardsdatascience.com/how-to-solve-an-ode-with-a-neural-network-917d11918932. Accessed: 2020-06-23.

[21] R. A. FISHER. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188, 1936.

[22] Austin G. Fowler, Matteo Mariantoni, John M. Martinis, and Andrew N. Cleland. Surface codes: Towards practical large-scale quantum computation. *Physical Review A*, 86(3), Sep 2012.

[23] Christopher Gerry and Peter Knight. *Introductory Quantum Optics*. Cambridge University Press, 2004.

[24] David J. Griffiths. *Introduction to Quantum Mechanics*. Pearson Education, Inc, 2 edition, 2014.

[25] Patrick Kidger and Terry Lyons. Universal approximation with deep narrow networks, 2019.

[26] Nathan Killoran, Thomas R. Bromley, Juan Miguel Arrazola, Maria Schuld, Nicolás Quesada, and Seth Lloyd. Continuous-variable quantum neural networks. *Physical Review Research*, 1(3), Oct 2019.

[27] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[28] Pieter Kok and Brendon W. Lovett. *Introduction to Optical Quantum Information Processing*. Cambridge University Press, 2010.

[29] Ulf Leonhardt. *Essential quantum optics: From quantum measurements to black holes*. Cambridge University Press, 2010.

[30] Seth Lloyd and Samuel L. Braunstein. Quantum computation over continuous variables. *Physical Review Letters*, 82(8):1784–1787, Feb 1999.

[31] Andrew Ng. Cs229 lecture notes: Principal components analysis. http://cs229.stanford.edu/notes/cs229-notes10.pdf. Accessed: 2020-08-07.

[32] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 10th edition, 2011.

[33] Encyclopedia of Mathematics. Riccati equation. http://encyclopediaofmath.org/index.php?title=Riccati_equation&oldid=49560. Accessed: 2020-08-19.

[34] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[35] Allan Pinkus. Approximation theory of the mlp model in neural networks. *Acta Numerica*, 8:143–195, 1999.

[36] H. H. Rosenbrock. An Automatic Method for Finding the Greatest or Least Value of a Function. *The Computer Journal*, 3(3):175–184, 01 1960.

[37] Maria Schuld, Ville Bergholm, Christian Gogolin, Josh Izaac, and Nathan Killoran. Evaluating analytic gradients on quantum hardware. *Physical Review A*, 99(3), Mar 2019.

[38] Murray R. Spiegel, Seymour Lipschutz, and John Liu. *Mathematical handbook of formulas and tables,*. McGraw-Hill, fourth edition, 2013.

[39] James Stewart. *Calculus: Concepts and Contexts*. Thomson Brooks/Cole, 3 edition, 2006.

[40] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake Vand erPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1. 0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[41] Christian Weedbrook, Stefano Pirandola, Raúl García-Patrón, Nicolas J. Cerf, Timothy C. Ralph, Jeffrey H. Shapiro, and Seth Lloyd. Gaussian quantum information. *Reviews of Modern Physics*, 84(2):621–669, May 2012.

[42] Graham Woan. *The Cambridge Handbook of Physics Formulas*. Cambridge University Press, 2000.