Learning quantum state properties with quantum and classical neural networks

ARTHUR PESAH

Master in Engineering Physics
Date: May 27, 2019
Supervisor: Peter Wittek (University of Toronto)
Examiner: Val Zwiller (KTH)
School of Engineering Sciences
Host company: University of Toronto
Swedish title: Inlärning av kvanttillståndens egenskaper med kvant och klassiska neurala nätverk

Abstract

While the simulation of quantum systems on near-term quantum devices have witnessed rapid advances in recent years, it is only the first step in understanding these systems. The efficient extraction of useful information from a simulated state represents a second important challenge. The traditional technique is to reconstruct the state using quantum state tomography before analytically computing the desired properties. However, this process requires, in general, an exponential number of measurements and it is inherently inefficient if we are not interested in the state itself, but only in a handful of scalar properties that characterize it.

In this thesis, we introduce several quantum algorithms to estimate quantum state properties directly without relying on tomography. The algorithms are a combination of quantum and classical neural networks, trained to return the desired property. Our contribution is both theoretical and numerical: we prove the universality of several architectures for the class of properties given as polynomial functionals of a density matrix, and evaluate their performance on some particular properties—purity and entropy—using quantum circuit simulators. Furthermore, we provide an extension of each architecture for continuous-variable states.

Sammanfattning

Samtidigt som simuleringen av kvantsystem på kortsiktiga enheter har visat sig snabba framsteg under de senaste åren är det bara det första steget i att förstå dessa system. Den effektiva extraktionen av användbar information från ett simulerat tillstånd representerar en andra viktig utmaning. Den traditionella tekniken är att rekonstruera tillståndet med hjälp av kvantstatstomografi innan man analyserar de önskade egenskaperna analytiskt. Denna process kräver emellertid i allmänhet ett exponentiellt antal mätningar och det är i sig ineffektivt om vi inte är intresserade av kvanttilståndet själv, utan bara i en handfull skalär egenskaper som karakteriserar den.

I denna avhandling introducerar vi flera kvantalgoritmer för att beräkna kvantstatusegenskaper direkt utan att förlita sig på tomografi. Algoritmerna är en kombination av kvant- och klassiska neurala nätverk, tränade för att returnera den önskade egenskapen. Vårt bidrag är både teoretiskt och numeriskt: vi bevisar universaliteten hos flera arkitekturer för klassen av egenskaper som ges som en polynomialfunktion av en densitetsmatris och utvärderar deras prestanda på vissa speciella egenskaper — renhet och entropi — med användning av kvantkretssimulatorer. Dessutom ger vi en förlängning av varje arkitektur för kontinuerliga variabla stater.

Contents

Chapter 1

Introduction

1.1 Motivation

After more than three decades since Richard Feynman's proposal of performing simulations using quantum phenomena [1], the first practical quantum computers are finally being built. The scope of calculations significantly expanded beyond simulations, with a range of promising applications emerging. In the short term, we must accept the presence of noise in the system. Despite this, some applications can still be accelerated using these early quantum computers. In this so-called Noisy Intermediate-Scale Quantum (NISQ) era [2], potential applications that could be accelerated include optimization [3, 4], chemistry [5, 6, 7] and machine learning [8, 9, 10].

In the process of enhancing machine learning using quantum computers, the concept of quantum neural network (QNN) has emerged as a promising application. Like their classical counterpart, QNNs can be used as classification and regression machines [11, 12], as well as generative models [13, 14]. While their advantage to process classical data has not yet been demonstrated, there is one area where the advantage is clear: quantum data processing. When the input of the QNN is itself a quantum state—prepared with another circuit that feeds into it—it has indeed no classical analogue to compete with. Yet, we can identify at least two use cases where QNNs with a quantum input can be useful: quantum simulation and quantum property testing.

Quantum simulation, as originally introduced by Feynman, is based on the idea that the state of n particles is described by exponentially many complex numbers. It would therefore require an exponentially

large memory to be stored and processed on a classical computer. On the contrary, a quantum device can manipulate the quantum state directly, without requiring to store each of its individual components. One important area of science that can be enhanced by quantum simulation is quantum chemistry. In this field, one wants, for instance, to compute the properties of the stable state (or *ground-state*) of a molecule. Many recent advances have allowed to compute the ground-state—as well as the first excited states—of a Hamiltonian on near-term, circuit-based, quantum computers [15, 16]. However, obtaining the ground-state of a system is rarely the last step of a quantum simulation: one wants to extract useful properties from it. In the case of molecules, the charge density, the dipole moment or the exchange-correlation energy are examples of properties interesting to chemist and material scientists [6]. The traditional technique is to reconstruct the entire state using quantum state tomography before analytically computing the desired properties. However, this process requires in general an exponential number of measurements and is inherently inefficient if we are interested in only a few scalar properties. Training a QNN to take the state as input and its properties as outputs is therefore one way to alleviate this issue.

Another area where QNNs can provide an advantage is quantum property testing. When building a quantum computer—or more generally any type of quantum technology—testing your device is of paramount importance, since current devices are subject to many sources of noise: decoherence, gate and measurements errors, loss, etc. One way to test it is to verify that, given a circuit, the output state corresponds to the theoretical one. In order to avoid once again the exponential process of quantum state tomography, another idea could be to check that several properties of the state hold, such as its entanglement or its purity. A QNN trained to output those values, before comparing them to the theory, would therefore allow an efficient testing of the device.

In this thesis, we study the possibility of efficiently estimating properties of two kinds of state: qubit states and continuous-variable (CV) states. While the former lays at the core of the most common quantum computing paradigm, CV states have recently regained popularity, with the creation of Python libraries to manipulate them [17, 18] and several algorithms for near-term devices using them [19, 20, 21].

1.2 Previous work

The idea of using quantum circuits to estimate properties of a state without full reconstruction goes back to [22], which introduced a class of circuits, based on the controlled-SWAP gate, to evaluate all the functionals $\mathrm{tr}\left[\rho^k\right]$ (for $k \in \mathbb{N}$), using k copies of ρ but the measurement of only one qubit in the σ_z-basis. They noted that getting the value of $\mathrm{tr}\left[\rho^k\right]$ for all $k \in \{2, ..., n\}$ (with n the number of qubits) is sufficient to obtain the spectrum of ρ, and therefore estimate any functional of the form $\mathrm{tr}[f(\rho)]$. The main downside of this approach is that it requires to build a circuit with n^2 qubits. Around the same time [23] presented a class of circuits that can compute any observable, i.e. linear functional of the form $\mathrm{tr}[O\rho]$. [24] and [25] generalized those circuits in a coherent framework, showing that any polynomial functional of a density matrix can be written $\mathrm{tr}\left[O\rho^{\otimes k}\right]$ and directly estimated with a controlled unitary gate and two Hadamard gate. However, the question of how to obtain this unitary in practice was still considered open by the authors.

Direct measurements of CV state functionals of the form $\mathrm{tr}\left[\rho^k\right]$ have also been considered in the literature. [26] showed that performing a discrete Fourier transform on k copies of the state, followed by a parity measurement was enough to evaluate those polynomials. [27] discovered the same method independently and generalized the circuit to arbitrary multi-partite states. A clear summary of those two approaches is carried through in [28]. However, as far as we know, the most general case of efficiently evaluating $\mathrm{tr}\left[O\rho^{\otimes k}\right]$ has not yet been considered in the literature.

The notion of quantum neural network has been developed more recently and can refer to very different quantum algorithms or circuits inspired by classical neural networks. As suggested in [29], quantum machine learning algorithms can be classified into three categories: classical machine learning (ML) to solve quantum tasks ("QC" ML) [30, 31, 32], quantum algorithms to enhance classical ML ("CQ" ML) [11, 12, 33] and quantum algorithms to solve quantum tasks ("QQ" ML) [29, 34, 35]. The last case is the one of interest for this thesis. In [35], the authors developed an architecture of QNN that respect certain symmetries, and use it to estimate ground-state topological phase of a many-body system. [34] trained parametrized circuits (with a varying number of gates) to compute the overlap $\mathrm{tr}[\rho\sigma]$ between two states ρ and σ (which becomes the purity when $\rho = \sigma$) with less gates than the traditional SWAP test. Fi-

nally, [29] developed a universal ansatz and training algorithm for learning unitary circuits, and evaluates them on random unitary matrices. However, none of those papers framed the problem as learning general, non-linear functionals of a state. Closer to our approach is [33], which constructed an architecture of QNN able to learn polynomial functions of real data (for classical ML tasks).

The papers mentioned above considered QNNs acting on discrete states (qubits). More general architectures working for continuous-variable states have been introduced in [36, 37]. However, by the time of writing, we haven't found any theoretical or numerical result for learning quantum properties with them.

1.3 Contributions

In this thesis, we introduce three different quantum neural network architectures designed to compute polynomial functionals $f(\rho)$ of a state ρ:

1. Quantum Polynomial Network with ancilla (PolyNet-ancilla): universal circuit that can estimate $f(\rho)$ as the average value $\langle Z_a \rangle$ of an ancilla qubit

2. Quantum Polynomial Network with correlations (PolyNet-corr): universal circuit whose predicted value for $f(\rho)$ is a linear combination of all the correlations $\langle Z_{i_1} ... Z_{i_j} \rangle$ of the output qubits.

3. Quantum Polynomial Network with averages (PolyNet-avg): circuit whose predicted value for $f(\rho)$ is a non-linear function (given by a classical neural network) of the averages $\langle Z_1 \rangle, ..., \langle Z_n \rangle$ of all the output qubits

We prove the universality of the first two circuits in the qubit-case and exhibit the challenges faced by the three of them in the continuous-variable case. We present an algorithm to train those three circuits, and evaluate them numerically on purity $\mathrm{tr}[\rho^2]$ and von Neumann entropy $\mathrm{tr}\,(\rho \log(\rho))$ estimation, for both qubits and continuous-variable states, using a simulator. While we achieve compelling performance in the qubit case, the results obtained for continuous-variable states, combined with the theory, can be seen as a no-go result.

1.4 Outline of the thesis

This thesis is divided into three main sections.

Background We start by an extensive introduction to the world of continuous-variable quantum computing, assuming the reader familiar with the basics of discrete quantum computing (qubits, gates, measurements). We then review the two main concepts combined in this thesis: quantum neural networks and property estimation. In particular, we exhibit some previous theoretical work that we rely on in the rest of the thesis.

Method We introduce here our three architectures independently, along with the theory that goes with them. We then present the training procedure and the dataset preparation

Experiments Finally, we show and analyze the practical results obtained with the different architectures, properties to estimate, and quantum computing paradigms.

Chapter 2

Background

2.1 Continuous-Variable quantum computing

Most introductions to quantum computing consider the *qubit* as the unit of information. A qubit is a quantum state describing a two-level system, such as the spin of an electron, the polarization of light, or the energy level of an atom. If we define $|0\rangle$ and $|1\rangle$ as respectively the ground state and the excited state of the system, a general 1-qubit state can be written $|\psi\rangle = a|0\rangle + b|1\rangle$ where $|a|^2 + |b|^2 = 1$. A natural extension of the qubit is the *qudit*, a state describing a d-levels system and which can be written $|\psi\rangle = a_0|0\rangle + ... + a_d|d\rangle$.

However, many systems in nature are described by a Hamiltonian with infinitely many eigenstates. One example of such system is the quantum harmonic oscillator, described by the Hamiltonian $H = \frac{1}{2}(p^2 + \omega^2 q^2)$, where q and p are the position and momentum operators. A general solution of this system is called a *qumode* and can be written $|\psi\rangle = \sum_{n=0}^{\infty} a_n|n\rangle$, where $|n\rangle$ is the eigenstate of the Hamiltonian with energy $n\hbar\omega$. The Hilbert space of qumodes is radically different than the space of qubits and therefore requires the use of a different paradigm of quantum computing: continuous-variable.

This section will be dedicated to the study of continuous-variable systems. We will start by motivating the use of qumodes in quantum computing, showing that such states correspond to states of light—i.e. solutions of the quantized Maxwell's equation—and can therefore be used in photonic experiments. The following parts will be dedicated to the different components of a CV system:

- the **qumode** and its different basis and representation.

- the **gates** that allow qumodes to evolve and getting entangled
- the **measurement** process, which can be carried in three different ways.

We assume that the reader is familiar with qubit-based quantum computing (gates, Bloch sphere, measurements) and will try to draw analogies with this traditional framework whenever it is relevant.

2.1.1 Physical motivation: quantizing light

The continuous-variable framework of quantum computing directly emerges from one of its experimental realizations: the photonic quantum computer. In this device, the vector of information—the qumode—is carried out by photons. This section is dedicated in finding the quantum state associated to a photon.

In classical physics, light is described by an electromagnetic field $(\boldsymbol{E}, \boldsymbol{B})$, solution of Maxwell's equations. In empty space, those are given by:

$$\begin{aligned} \nabla \times \boldsymbol{E} &= \frac{\partial \boldsymbol{B}}{\partial t} \\ \nabla \times \boldsymbol{B} &= \mu_0 \epsilon_0 \frac{\partial \boldsymbol{E}}{\partial t} \\ \nabla \cdot \boldsymbol{E} &= 0 \\ \nabla \cdot \boldsymbol{B} &= 0 \end{aligned} \tag{2.1}$$

The usual process to turn classical equations into quantum is to find a pair of variables, usually called $\boldsymbol{q}$ and $\boldsymbol{p}$, such that the energy function $H(\boldsymbol{q}, \boldsymbol{p})$ follows Hamilton's equations:

$$\begin{aligned} \frac{\partial H}{\partial \boldsymbol{p}} &= \dot{\boldsymbol{q}} \\ \frac{\partial H}{\partial \boldsymbol{q}} &= -\dot{\boldsymbol{p}} \end{aligned} \tag{2.2}$$

The quantization process then consists in turning $\boldsymbol{q}$ and $\boldsymbol{p}$ into operators, such that they obey the so-called *canonical commutation relations*:

$$[q_i, p_j] = i\hbar\delta_{ij} \tag{2.3}$$

For any quantum state following Schrödinger's equation, it is then possible to show, using Ehrenfest theorem, that Hamilton's equation will be respected in average for $\boldsymbol{q}$ and $\boldsymbol{p}$, recovering the classical case.

With this objective in mind, let us start by examining the solutions of (2.1) in the simple case of a one-dimensional field confined inside a cavity with perfectly conducting walls at $z = 0$ and $z = L$. Assuming the electric field is polarized in the x-direction, the single-mode solution of Maxwell's equations is given by:

$$\begin{aligned} \boldsymbol{E}(z,t) &= \left(\frac{2\omega^2}{V\epsilon_0}\right)^{\frac{1}{2}} q(t) sin(kz)\boldsymbol{e_x} \\ \boldsymbol{B}(z,t) &= \frac{\mu_0\epsilon_0}{k}\left(\frac{2\omega^2}{V\epsilon_0}\right)^{\frac{1}{2}} \dot{q}(t) cos(kz)\boldsymbol{e_y} \end{aligned} \tag{2.4}$$

where ω is the frequency of the mode, V the effective volume of the cavity, $k = \frac{\omega}{c}$ the wave number, and q a function encapsulating the time dependency of the fields. We will show that $q(t)$ and $p(t) = \dot{q}(t)$ can be used as our canonical variables.

It is known that the energy of an electromagnetic field is given by

$$H = \frac{1}{2}\int \mathrm{d}V \left[\epsilon_0 \boldsymbol{E}^2(\boldsymbol{r},t) + \frac{1}{\mu_0}\boldsymbol{B}^2(\boldsymbol{r},t)\right] \tag{2.5}$$

Inserting our solution (2.4) in (2.5), we get:

$$H = \frac{1}{2}(p^2 + \omega^2 q^2) \tag{2.6}$$

We can see that Hamilton's equations (2.2) are met if we choose q and p as our canonical variables. Our system is therefore ready to be quantized:

$$\hat{H} = \frac{1}{2}(\hat{p}^2 + \omega^2 \hat{q}^2) \tag{2.7}$$

where $[\hat{q}, \hat{p}] = i\hbar$. The operators $\hat{q}$ and $\hat{p}$ are often called position and momentum operators in the literature. However, despite this denomination, they do not correspond to the position and momentum of any physical object in our case, but rather to the amplitudes of the electric and magnetic fields. Another convenient way to write our Hamiltonian is using the so-called **quadrature operators** $\hat{X} = \sqrt{\frac{\omega}{2\hbar}}\hat{q}$ and $\hat{P} = \frac{1}{\sqrt{2\hbar\omega}}\hat{p}$, which correspond to position and momentum without unit:

$$\hat{H} = \hbar\omega(\hat{X}^2 + \hat{P}^2) \tag{2.8}$$

Operators	Definition	Commutation relation
Position and momentum	Quantization of $q(t)$ and $p(t)$	$[\hat{q}, \hat{p}] = i\hbar$
Quadratures	$\hat{X} = \sqrt{\frac{\omega}{2\hbar}}\hat{q}$ $\hat{P} = \frac{1}{\sqrt{2\hbar\omega}}\hat{p}$	$[\hat{X}, \hat{P}] = \frac{i}{2}$
Annihilation and creation	$\hat{a} = \hat{X} + i\hat{P}$ $\hat{a}^\dagger = \hat{X} - i\hat{P}$	$[a, \hat{a}^\dagger] = 1$

Table 2.1: Summary of the different canonically conjugate operators

This Hamiltonian describes a quantum harmonic oscillator, whose solution is detailed in every quantum mechanics textbook. Finding the eigenvalues of $\hat{H}$ essentially consists in the following steps:

1. Defining the operators $\hat{a} = \hat{X} + i\hat{P}$ and $\hat{a}^\dagger = \hat{X} - i\hat{P}$, called respectively *annihilation* and *creation* operators, as well as $\hat{N} = \hat{a}^\dagger\hat{a}$, called the *number* operator.

2. Showing that, for every eigenstate $|x\rangle$ of $\hat{H}$ associated to the positive eigenvalue x, $\hat{a}^\dagger|x\rangle = \sqrt{x+1}|x+1\rangle$, $\hat{a}|x\rangle = \sqrt{x}|x-1\rangle$ and $\hat{N}|x\rangle = x|x\rangle$

3. Proving that all eigenvalues are in fact positive integer

4. Showing that $\hat{H} = \hbar\omega(\hat{N} + \frac{1}{2})$ and deducing that the eigenvalues of $\hat{H}$ are given by $\hbar\omega(n + \frac{1}{2})$ where n is a positive integer.

To sum it up, single-mode photons are described by a Hamiltonian $\hat{H}$ which has a discrete but infinite number of eigenvectors, as shown in Figure 2.1. The eigenstates $|n\rangle$ of $\hat{H}$ are called *Fock states* and constitute an orthonormal basis of our Hilbert space. Therefore, every single-mode photon state $|\psi\rangle$ can be written $|\psi\rangle = \sum_n a_n|n\rangle$, with $\langle n|m\rangle = \delta_{nm}$. We call such a state a *qumode*.

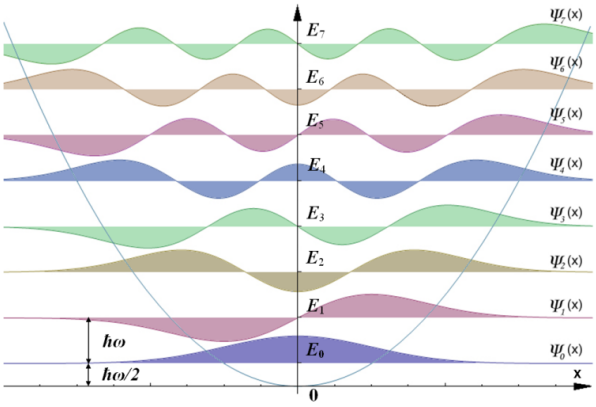


Figure 2.1: The quantum harmonic oscillator is characterized by a discrete but infinite number of equally spaced energy levels. A state in one of those levels is called a Fock state

2.1.2 Classification of qumodes

At this stage of the presentation, a question arises: if a qumode is described using a discrete basis, why do we call this computational model "continuous-variable"? We will answer that question in this section, showing that there exists several other bases to represent a photonic state, described by continuous variables. In the same way as the Fock basis was extracted from the spectrum of the Hamiltonian $\hat{H}$, other bases will be constructed by looking at the eigenstates of different operators.

$\hat{X}$ and $\hat{P}$ eigenstates

The operators $\hat{X}$ and $\hat{P}$ are both Hermitian observables and can therefore be diagonalized:

$$\hat{X}|x\rangle = x|x\rangle$$
$$\hat{P}|p\rangle = p|p\rangle$$

The spectrum of those operators is continuous and the eigenstates $\{|x\rangle\}$ and $\{|p\rangle\}$ form a continuous basis of the state space: for any state $|\psi\rangle$,

$$|\psi\rangle = \int \psi(x)|x\rangle dx$$

$$|\psi\rangle = \int \phi(p)|p\rangle dp$$

.

Since $[\hat{X}, \hat{P}] = \frac{i}{2}$, there is no eigenstate $|x, p\rangle$ of both $\hat{X}$ and $\hat{P}$: that is the uncertainty principle. Therefore, it is not possible to associate a probability distribution $p(x, p)$ to the whole phase space. However, it is possible to construct so-called *quasi-probability distributions* on the phase space: real functions of x and p that implicitly contain the probability associated with each variables. The most common quasi-probability distribution is the Wigner function, which motivates the definition of a very important category of states: Gaussian states.

Wigner's representation and Gaussian states

In classical probability theory, a probability distribution can be defined in two ways: with a density function or with its Fourier transform, the characteristic function. More precisely, for a random variable X, the characteristic function is defined as $\chi(t) = \mathbb{E}\left[e^{iXt}\right]$.

Although it is not possible to associate a density function to the quantum phase space, one can construct the equivalent of the characteristic function for a general density matrix ρ:

$$\begin{aligned} \chi(\alpha, \beta) &= \mathbb{E}_\rho\left[e^{i(\alpha\hat{X}+\beta\hat{P})}\right] \\ &= \mathrm{tr}\left[\rho e^{i(\alpha\hat{X}+\beta\hat{P})}\right] \end{aligned} \tag{2.9}$$

for $\alpha, \beta \in \mathbb{R}$. The Wigner function is defined as the inverse Fourier transform of the characteristic function:

$$W(x, p) = \frac{1}{\pi^2}\int e^{-i(\alpha x+\beta p)}\chi(\alpha, \beta)d\alpha d\beta \tag{2.10}$$

This process is summarized in the commutative diagram in Figure 2.2.

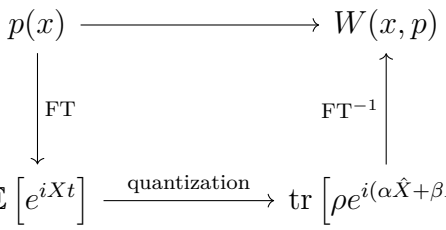


Figure 2.2: Construction of the Wigner function

If interpreting the Wigner function in general is not straightforward, there are several properties that can help us to do so:

1. $W(x,p) \in \mathbb{R}$ and is normalized:

$$\int W(x,p)dxdp = 1$$

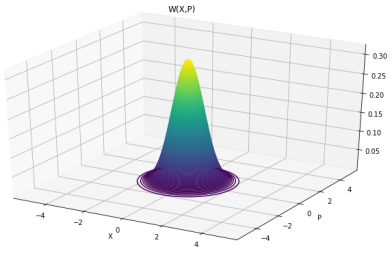
2. The marginal distributions over x and p correspond to the actual probability distributions:

$$\int W(x,p)dp = |\psi(x)|^2$$

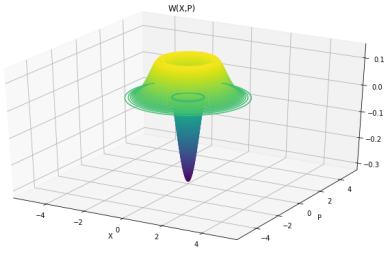
$$\int W(x,p)dx = |\phi(p)|^2$$

3. $W(x,p) \geq 0 \iff W(x,p)$ is a Gaussian distribution

States whose Wigner function respects condition 3 are called *Gaussian states*. An example of Gaussian state is the vacuum Fock state $|0\rangle$, whose Wigner function is represented below:



All other Fock states $|n\rangle$ with $n \geq 1$ are non-Gaussian. Here is the Wigner function of $|1\rangle$ for example:



So why do we care about Gaussian states? Here are some insights:

1. They are the easiest to implement experimentally. The light produced by a laser is a Gaussian state and can be transformed into any other Gaussian state by means of linear optics and squeezing. Obtaining non-Gaussian states is one of the main challenges of experimental quantum optics.

2. Gaussian operations, i.e. operations which turn Gaussian states into Gaussian states, correspond to affine transformations of the phase space. Transformations such as rotations, scaling and translations preserve Gaussian distributions. On the contrary, non-Gaussian operations are non-linear in the phase space. We will use this property when talking about quantum neural networks.

3. They can be simulated efficiently by classical computers. Therefore, any CV quantum algorithm with a speed-up over classical computation must contain non-Gaussian operations.

Due to Heisenberg's uncertainty principle, not all Gaussian distributions can be associated to a Gaussian state. There exists actually only two types of valid Gaussian states: coherent states and squeezed states.

Coherent states

Informally, coherent states are Gaussian states with the same variance on the x and p axes. Here are examples of coherent states (in the Wigner representation):

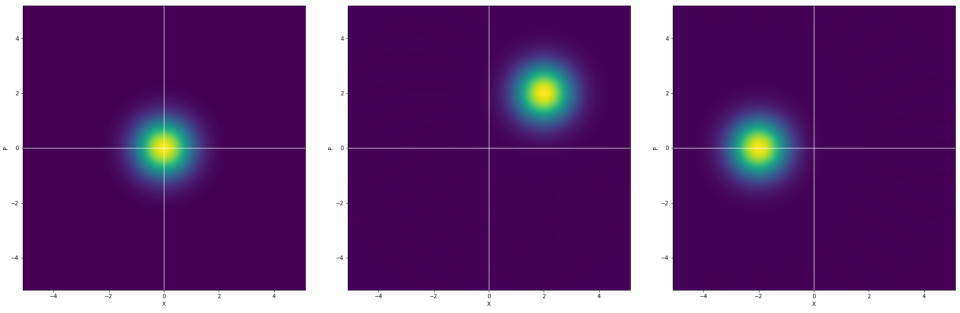


Figure 2.3: Wigner representation of several coherent state. **Left:** $|0\rangle$. **Middle:** $|\alpha = (2, 2)\rangle$. **Right:** $|\alpha = (-1, 0)\rangle$

All coherent states have the same variance and are therefore only characterized by the two components of the mean.

Formally, coherent states are defined as eigenstates of the annihilation operator $\hat{a}$:

$$\hat{a}\,|\alpha\rangle = \alpha\,|\alpha\rangle$$

The first example of $\hat{a}$ eigenstate is the Fock state $|0\rangle$: $\hat{a}|0\rangle = 0$ by definition of $\hat{a}$. To obtain other coherent states, we need to introduce the *displacement operator*, defined as

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

for $\alpha \in \mathbb{C}$. We will now prove three important facts about coherent states:

1. $|\alpha\rangle$ have the following decomposition in the Fock basis:

$$|\alpha\rangle = e^{-\frac{|\alpha|^2}{2}} \sum_{n=0}^{\infty} \frac{\alpha^n}{\sqrt{n!}}\,|n\rangle$$

2. Every coherent state can be written $|\alpha\rangle = D(\alpha)\,|0\rangle$

3. Coherent states are Gaussian states with mean $(\operatorname{Re}\alpha, \operatorname{Im}\alpha)$ and variance $\frac{1}{4}$ on the $\hat{X}$ and $\hat{P}$ axis (as introduced in the first paragraph)

Proposition 1. *For all $\alpha \in \mathbb{C}$, the coherent state $|\alpha\rangle$ have the following decomposition in the Fock basis:*

$$|\alpha\rangle = e^{-\frac{|\alpha|^2}{2}} \sum_{n=0}^{\infty} \frac{\alpha^n}{\sqrt{n!}}\,|n\rangle$$

.

Proof. Since the Fock states form a basis of our space, every $|\alpha\rangle$ can be written

$$|\alpha\rangle = \sum_{n=0}^{\infty} C_n\,|n\rangle\,.$$

Applying the definition of $|\alpha\rangle$ as an eigenstate of $\hat{a}$ and the relation $\hat{a}\,|n\rangle = \sqrt{n}\,|n-1\rangle$, we have:

$$\begin{aligned}
\hat{a}\,|\alpha\rangle &= \alpha\,|\alpha\rangle \\
\sum_{n=0}^{\infty} C_n \hat{a}\,|n\rangle &= \sum_{n=0}^{\infty} \alpha C_n\,|n\rangle \\
\sum_{n=1}^{\infty} C_n \sqrt{n}\,|n-1\rangle &= \sum_{n=0}^{\infty} \alpha C_n\,|n\rangle \\
\sum_{n=0}^{\infty} C_{n+1} \sqrt{n+1}\,|n\rangle &= \sum_{n=0}^{\infty} \alpha C_n\,|n\rangle
\end{aligned} \tag{2.11}$$

By identifying the coefficients on both sides, we get the recurrent relation

$$\forall n \in \mathbb{N}, C_{n+1}\sqrt{n+1} = \alpha C_n \tag{2.12}$$

whose solution is:

$$C_n = C_0 \frac{\alpha^n}{\sqrt{n!}} \tag{2.13}$$

We can obtain C_0 using the normalization condition

$$\begin{aligned} \langle\alpha|\alpha\rangle &= 1 \\ |C_0|^2 \sum_{n=0}^{\infty} \frac{|\alpha|^{2n}}{n!} &= 1 \\ |C_0|^2 e^{|\alpha|^2} &= 1 \\ C_0 &= e^{-\frac{|\alpha|^2}{2}} \end{aligned} \tag{2.14}$$

which leads to the expected results:

$$|\alpha\rangle = e^{-\frac{|\alpha|^2}{2}} \sum_{n=0}^{\infty} \frac{\alpha^n}{\sqrt{n!}} |n\rangle \tag{2.15}$$

□

Proposition 2. *Every coherent state* $|\alpha\rangle$ *can be written* $|\alpha\rangle = D(\alpha)|0\rangle$ *where* $D(\alpha) = e^{\alpha\hat{a}^\dagger - \alpha^*\hat{a}}$ *is the displacement operator*

Proof. To prove this proposition, we will show that the states $D(\alpha)|0\rangle$ have the same expansion as coherent states in the Fock basis. Let us first notice that the displacement operator can be rewritten

$$D(\alpha) = e^{-\frac{1}{2}|\alpha|^2} e^{\alpha\hat{a}^\dagger} e^{-\alpha^*\hat{a}} \tag{2.16}$$

Indeed, the Baker-Campbell-Hausdorff formula tells us that

$$e^{X+Y} = e^X e^Y e^{\frac{1}{2}[X,Y]}$$

if $[X, Y]$ is a multiple of the identity. In our case, $X = \alpha\hat{a}^\dagger$, $Y = \alpha^*\hat{a}$ and $[X, Y] = |\alpha|^2[\hat{a}^\dagger, \hat{a}] = -|\alpha|^2\mathbb{I}$. We used the fact that $[\hat{a}, \hat{a}^\dagger] = \mathbb{I}$, which can be proven using the canonical commutation relation $[p, q] = i\hbar\mathbb{I}$ (introduced in Section 2.1.1) and the definition of the annihilation and creation operators.

Applying (2.16) to the vacuum state leads to

$$e^{-\frac{|\alpha|^2}{2}} e^{\alpha\hat{a}^\dagger} e^{-\alpha^*\hat{a}} |0\rangle = e^{-\frac{|\alpha|^2}{2}} e^{\alpha\hat{a}^\dagger} |0\rangle \tag{2.17}$$

since all terms $(\alpha^*)^k \hat{a}^k |0\rangle$ of the Taylor expansion of $e^{-\alpha^* \hat{a}}$ will be 0 except for $k = 0$. Taylor expanding the second exponential and using $\hat{a}^\dagger |n\rangle = \sqrt{n+1}\,|n+1\rangle$ gives us

$$\begin{aligned} e^{-\frac{|\alpha|^2}{2}} e^{\alpha \hat{a}^\dagger} |0\rangle &= e^{-\frac{|\alpha|^2}{2}} \sum_{n=0}^{\infty} \frac{\alpha^n (\hat{a}^\dagger)^n}{n!} |0\rangle \\ &= e^{-\frac{|\alpha|^2}{2}} \sum_{n=0}^{\infty} \frac{\alpha^n}{n!} \sqrt{n!}\, |n\rangle \\ &= e^{-\frac{|\alpha|^2}{2}} \sum_{n=0}^{\infty} \frac{\alpha^n}{\sqrt{n!}} |n\rangle \end{aligned} \tag{2.18}$$

Therefore, the states $D(\alpha)|0\rangle$, also called *displaced vacuum states* are exactly the coherent states with eigenvalue α. □

Proposition 3. *$|\alpha\rangle$ is a Gaussian state with mean $(\operatorname{Re}\alpha, \operatorname{Im}\alpha)$ and variance $\frac{1}{4}$ on both the $\hat{X}$ and $\hat{P}$ axes.*

Proof. We need to prove that the Wigner function of $|\alpha\rangle$ is a Gaussian distribution. As a reminder, the Wigner function of a density matrix ρ is defined as the Fourier transform of the characteristic function

$$\chi(\lambda_x, \lambda_p) = \operatorname{tr}\left[\rho e^{i(\lambda_x \hat{X} + \lambda_p \hat{P})}\right]$$

for $\lambda_x, \lambda_p \in \mathbb{R}$. Noticing that

$$\begin{aligned} i(\lambda_x \hat{X} + \lambda_p \hat{P}) &= i\left(\lambda_x \frac{1}{2}(\hat{a} + \hat{a}^\dagger) + \lambda_p \frac{1}{2i}(\hat{a} - \hat{a}^\dagger)\right) \\ &= \frac{1}{2}(\lambda_p + i\lambda_x)\,\hat{a} + \frac{1}{2}(\lambda_p - i\lambda_x)\,\hat{a}^\dagger \\ &= \lambda^* \hat{a} - \lambda \hat{a}^\dagger \end{aligned} \tag{2.19}$$

where $\lambda \coloneqq \frac{1}{2}(\lambda_p - i\lambda_x)$, we can rewrite the characteristic function as

$$\chi(\lambda) = \operatorname{tr}\left[\rho D(\lambda)\right], \tag{2.20}$$

which is often taken as the textbook definition of the characteristic function.

In our case, $\rho = D(\alpha)|0\rangle\langle 0|\,D^\dagger(\alpha)$ and

$$\chi(\lambda) = \operatorname{tr}\left[D(\alpha)|0\rangle\langle 0|\,D^\dagger(\alpha) D(\lambda)\right]. \tag{2.21}$$

Using $D^\dagger(\alpha) = D(-\alpha)$ and $D(\alpha)D(\beta) = e^{i\,\mathrm{Im}\,\alpha\beta^*}D(\alpha+\beta)$ (eq. (3.38) and (3.42) of [38]),

$$\begin{aligned}
\chi(\lambda) &= \langle 0|\, D(-\alpha)D(\lambda)D(\alpha)\,|0\rangle \\
&= e^{-i\,\mathrm{Im}(\alpha\lambda^*)}\,\langle 0|\, D(\lambda-\alpha)D(\alpha)\,|0\rangle \\
&= e^{-i\,\mathrm{Im}(\alpha\lambda^*)}e^{i\,\mathrm{Im}((\lambda-\alpha)\alpha^*))}\,\langle 0|\, D(\lambda)\,|0\rangle \\
&= e^{2i\,\mathrm{Im}\,\lambda\alpha^*}\langle 0|\lambda\rangle \\
&= e^{2i\,\mathrm{Im}\,\lambda\alpha^*}e^{-\frac{|\lambda|^2}{2}} \\
\chi(\lambda) &= e^{-\frac{1}{2}\left(|\lambda|^2-4i\,\mathrm{Im}\,\lambda\alpha^*\right)}
\end{aligned} \tag{2.22}$$

Completing the square gives us

$$\begin{aligned}
\chi(\lambda) &= e^{-\frac{1}{2}\left(|\lambda-2i\alpha|^2-4|\alpha|^2\right)} \\
&= e^{-\frac{1}{2}|\lambda-\beta|^2}e^{\frac{1}{2}|\beta|^2}
\end{aligned} \tag{2.23}$$

where $\beta \coloneqq 2i\alpha$.

We previously defined the Wigner function as the Fourier transform of $\chi(\lambda_x, \lambda_p)$:

$$W(x,p) = \frac{1}{\pi^2}\int e^{-i(\lambda_x x+\lambda_p p)}\chi(\lambda_x,\lambda_p)d\lambda_x d\lambda_p \tag{2.24}$$

However, as in (2.19), we can rewrite the exponential by defining $\lambda \coloneqq \frac{1}{2}(\lambda_p - i\lambda_x)$ and $z = x + ip$. We can now calculate the Wigner function:

$$\begin{aligned}
W(z) &= e^{\frac{1}{2}|\beta|^2}\int \chi(\lambda)e^{z\lambda^*-z^*\lambda}d^2\lambda \\
&= e^{\frac{1}{2}|\beta|^2}\int e^{-\frac{1}{2}|\lambda-\beta|^2}e^{-2i\,\mathrm{Im}\,\lambda z^*}d^2\lambda \\
&= e^{\frac{1}{2}|\beta|^2}\int e^{-\frac{1}{2}\left(|\lambda-\beta|^2+4i\,\mathrm{Im}\,\lambda z^*\right)}d^2\lambda \\
&= e^{\frac{1}{2}|\beta|^2}\int e^{-\frac{1}{2}\left(|\lambda-\beta|^2+4i\,\mathrm{Im}(\lambda-\beta)z^*+4i\,\mathrm{Im}\,\beta z^*\right)}d^2\lambda \\
&= e^{\frac{1}{2}|\beta|^2}\int e^{-\frac{1}{2}\left(|\lambda-\beta+2iz|^2-|z|^2+4i\,\mathrm{Im}\,\beta z^*\right)}d^2\lambda \\
&= e^{\frac{1}{2}|\beta|^2}e^{\frac{1}{2}|z|^2}e^{-2i\,\mathrm{Im}\,\beta z^*}\int e^{-\frac{1}{2}|\lambda-\beta+2iz|^2}d^2\lambda \\
&= e^{\frac{1}{2}(|\beta|^2-4i\,\mathrm{Im}\,\beta z^*)}e^{\frac{1}{2}|z|^2}\int e^{-\frac{1}{2}|\lambda-\beta+2iz|^2}d^2\lambda \\
&= e^{\frac{1}{2}|\beta-2iz|^2}e^{-\frac{1}{2}|z|^2}e^{\frac{1}{2}|z|^2}\int e^{-\frac{1}{2}|\lambda-\beta+2iz|^2}d^2\lambda \\
W(z) &= e^{\frac{1}{2}|\beta-2iz|^2}\int e^{-\frac{1}{2}|\lambda-\beta+2iz|^2}d^2\lambda
\end{aligned} \tag{2.25}$$

Noticing that the integrand is a (non-normalized) two-dimensional Gaussian distribution with variance 1 and mean $\beta - 2iz$, we get for the Wigner function:

$$\begin{aligned} W(z) &= \frac{1}{\pi^2} 2\pi e^{\frac{1}{2}|\beta - 2iz|^2} \\ &= \frac{2}{\pi} e^{\frac{1}{2}|2iz - 2i\alpha|^2} \\ &= \frac{1}{2\pi\sigma^2} e^{\frac{|z-\alpha|^2}{2\sigma^2}} \end{aligned} \tag{2.26}$$

where $\sigma \coloneqq \frac{1}{4}$. Therefore, $W(z) = W(x + ip)$ is a Gaussian distribution with mean α and variance $\frac{1}{4}$. □

Remark 1. *The uncertainty relation for the quadrature operators is given by*

$$\langle (\Delta X)^2 \rangle \langle (\Delta P)^2 \rangle \geq \frac{1}{16} \tag{2.27}$$

For coherent states, we showed that $\langle (\Delta X)^2 \rangle = \langle (\Delta P)^2 \rangle = \frac{1}{4}$*, which means that* $\langle (\Delta X)^2 \rangle \langle (\Delta P)^2 \rangle = \frac{1}{16}$*. Therefore, coherent states are said to saturate, or equalize, the uncertainty relation.*

Remark 2. *It can be shown that coherent states* $|\alpha\rangle$ *form a basis of our Hilbert space. However, this basis is NOT an orthogonal basis:*

$$\langle \alpha_1 | \alpha_2 \rangle = e^{-|\alpha_1 - \alpha_2|^2}$$

Squeezed states

To define the second type of Gaussian states, we give back freedom to the variance. However, due to the uncertainty principle, decreasing the noise on one axis necessarily increases the noise on the orthogonal axis with the same amount. States with different standard deviations on two axes are called *squeezed states* and are represented in Figure (2.4).

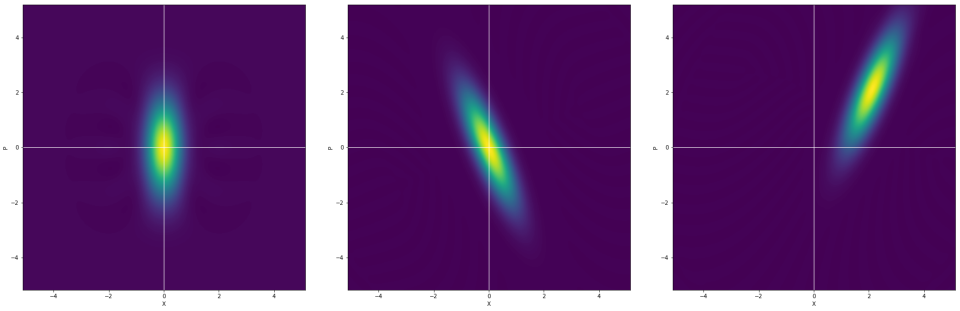


Figure 2.4: Wigner representation of several squeezed state. **Left:** $|\alpha = 0, z = (0.5, 0)\rangle$. **Middle:** $|\alpha = 0, z = (0.5, 0.5)\rangle$. **Right:** $|\alpha = (1.5, 1.5), z = (0.5, -0.5)\rangle$

As for coherent states, formally defining squeezed states requires the introduction of a new operator, called *squeezing operator* and given by

$$S(\xi) = e^{\frac{1}{2}\left(\xi^* \hat{a}^2 - \xi (\hat{a}^\dagger)^2\right)}$$

for $\xi \in \mathbb{C}$. Squeezed states are then defined as the states

$$|\xi, \alpha\rangle = D(\alpha) S(\xi) |0\rangle .$$

Writing $\xi = re^{i\theta}$, we can interpret θ as the angle of the axis in which the squeezing is performed, r as the amount of squeezing, and α as the position of the mean, as highlighted by the following proposition:

Proposition 4. *$|\alpha, \xi\rangle$ is a Gaussian state with mean $(\operatorname{Re}\alpha, \operatorname{Im}\alpha)$ and covariance matrix*

$$C(r, \theta) = \frac{1}{4} R\left(-\frac{\theta}{2}\right) \begin{pmatrix} e^{-2r} & 0 \\ 0 & e^{2r} \end{pmatrix} R\left(\frac{\theta}{2}\right)$$

where $R(\theta) \coloneqq \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}$.

Remark 3. *$\hat{X}$ and $\hat{P}$ eigenstates can be seen as a limit of squeezed states:*

$$\begin{aligned} |x\rangle &= \lim_{r \to \infty} |\alpha = x, \xi = r\rangle \\ |p\rangle &= \lim_{r \to \infty} |\alpha = ip, \xi = -r\rangle \end{aligned} \tag{2.28}$$

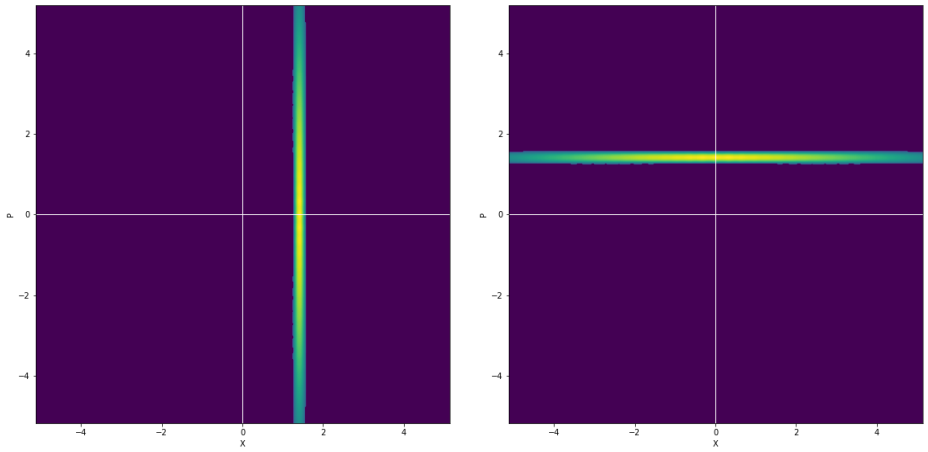


Figure 2.5: Wigner representation of $\hat{X}$ and $\hat{P}$ eigenstates. **Left:** $\hat{X}$-eigenstate $|x = 1.5\rangle$. **Right:** $\hat{P}$-eigenstate $|p = 1.5\rangle$.

2.1.3 Quantum gates in a CV system

We will first review the general formalism of CV operations, before describing the most common gates used in CV quantum computing.

The two representations of CV gates

As we've seen, there are two ways to understand operations on CV quantum states: either as unitary operations acting on the state vector, or as transformations of the phase space. For instance, the displacement of a state $|\psi\rangle$, with Wigner function $W(x, p)$, can be represented both by $D(\alpha)\,|\psi\rangle$ and $W(x + \mathrm{Re}\,\alpha, p + \mathrm{Im}\,\alpha)$.

In the Hilbert space picture, unitary operators can always be written as the exponential of a skew-symmetric operator[1]:

$$U = e^S$$

For instance, $S = \alpha\hat{a}^\dagger - \alpha^*\hat{a}$ corresponds to a displacement. In practice, most unitaries considered in CV quantum computing consist in the exponential of polynomial functions of $\hat{a}$ and $\hat{a}^\dagger$ (or equivalently $\hat{x}$ and $\hat{p}$). In particular, second-order polynomials correspond to Gaussian transformations (i.e. transformations that turn Gaussian states into Gaussian

[1]Since the Lie algebra of the unitary group consists in the skew-symmetric operators

states), while higher order polynomial correspond to non-Gaussian transformations.

In the phase space picture, the distinction between Gaussian and non-Gaussian transformations is also very clear. Gaussian operations are affine transformations of the phase space:

$$\begin{pmatrix}\mathbf{x}\\ \mathbf{p}\end{pmatrix} \mapsto M\begin{pmatrix}\mathbf{x}\\ \mathbf{p}\end{pmatrix} + \begin{pmatrix}\boldsymbol{\alpha_R}\\ \boldsymbol{\alpha_I}\end{pmatrix} \tag{2.29}$$

However, not all linear transformations M are allowed, but only those preserving the uncertainty principle, called *symplectic matrices*.

The symplectic formalism

Preserving the uncertainty principle comes down to preserving the canonical commutation relations. In the general multimode case, noting $\hat{\mathbf{x}} = (\hat{x}_1, ..., \hat{x}_n)^T$, $\hat{\mathbf{p}} = (\hat{p}_1, ..., \hat{p}_n)^T$ and $\hat{\mathbf{r}} = \begin{pmatrix}\hat{\mathbf{x}}\\ \hat{\mathbf{p}}\end{pmatrix}$ the canonical commutation relations can be written

$$[\hat{\mathbf{r}}, \hat{\mathbf{r}}^T] = i\Omega \tag{2.30}$$

where the Lie bracket between vectors of operators is defined as

$$[\hat{\mathbf{a}}, \hat{\mathbf{b}}] = \hat{\mathbf{a}}\hat{\mathbf{b}} - (\hat{\mathbf{a}}\hat{\mathbf{b}})^T$$

and

$$\Omega = \begin{pmatrix}0 & -\mathbb{I}\\ \mathbb{I} & 0\end{pmatrix}$$

is the $2n \times 2n$ symplectic form. Therefore, any matrix M that preserves the canonical commutation relations must verify the relation

$$[M\hat{\mathbf{r}}, \hat{\mathbf{r}}^T M^T] = i\Omega \tag{2.31}$$

Developing the LHS gives us

$$\begin{aligned}[M\hat{\mathbf{r}}, \hat{\mathbf{r}}^T M^T] &= M\hat{\mathbf{r}}\hat{\mathbf{r}}^T M^T - \left(M\hat{\mathbf{r}}\hat{\mathbf{r}}^T M^T\right)^T \\ &= M\left(\hat{\mathbf{r}}\hat{\mathbf{r}}^T - \left(\hat{\mathbf{r}}\hat{\mathbf{r}}^T\right)^T\right)M^T \\ &= iM\Omega M^T\end{aligned} \tag{2.32}$$

This motivates the definition of a symplectic matrix: a matrix M is symplectic if

$$M^T\Omega M = \Omega.$$

All the Gaussian gates described below can be shown to respect this property.

Displacement gate

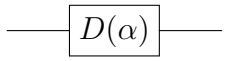


Figure 2.6: Symbol of a displacement gate

The displacement gate $D(\alpha) = e^{\alpha\hat{a}^\dagger - \alpha^*\hat{a}}$ (for $\alpha \in \mathbb{C}$) is a single mode Gaussian gate which translates the phase space the following way:

$$\begin{pmatrix} x \\ p \end{pmatrix} \mapsto \begin{pmatrix} x \\ p \end{pmatrix} + \begin{pmatrix} \operatorname{Re}\alpha \\ \operatorname{Im}\alpha \end{pmatrix}$$

Squeezing gate

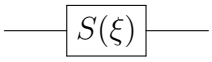


Figure 2.7: Symbol of a squeezing gate

The squeezing gate $S(\xi) = e^{\frac{1}{2}\left(\xi^*\hat{a}^2 - \xi(\hat{a}^\dagger)^2\right)}$ (for $\xi = re^{i\theta} \in \mathbb{C}$) is a single mode Gaussian gate which squeezes the phase space the following way:

$$\begin{pmatrix} x \\ p \end{pmatrix} \mapsto R\left(\frac{\theta}{2}\right) \begin{pmatrix} e^{-r} & 0 \\ 0 & e^{r} \end{pmatrix} R\left(-\frac{\theta}{2}\right) \begin{pmatrix} x \\ p \end{pmatrix}$$

Rotation gate

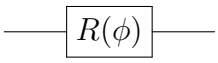


Figure 2.8: Symbol of a rotation gate

The rotation gate $R(\theta) = e^{i\theta\hat{a}^\dagger\hat{a}}$ (for $\theta \in [0, 2\pi)$) is a single mode Gaussian gate which rotates the phase space:

$$\begin{pmatrix} x \\ p \end{pmatrix} \mapsto \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix} \begin{pmatrix} x \\ p \end{pmatrix}$$

Beam splitter

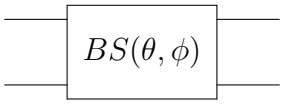


Figure 2.9: Symbol of a beam splitter

A beam splitter $BS(\theta, \phi) = e^{\theta\left(e^{i\phi}\hat{a}_i\hat{a}_j^\dagger - e^{-i\phi}\hat{a}_i^\dagger\hat{a}_j\right)}$ (for $\theta, \phi \in [0, 2\pi]$) is a two-modes Gaussian gate which entangles the phase space the following way:

$$\begin{pmatrix} x_1 \\ x_2 \\ p_1 \\ p_2 \end{pmatrix} \mapsto \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & \cos(\theta) & -\sin(\theta) \\ 0 & 0 & \sin(\theta) & \cos(\theta) \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ p_1 \\ p_2 \end{pmatrix}$$

Examples of non-Gaussian gates

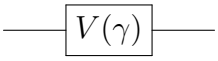


Figure 2.10: Symbol of a cubic gate

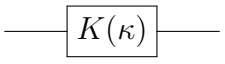


Figure 2.11: Symbol of a Kerr gate

Non-Gaussian gates are unitaries of the form $e^{P(\hat{a},\hat{a}^\dagger)}$, where P is a polynomial of degree at least three. Examples are the cubic gates $e^{i\frac{\gamma}{3\hbar}x^3}$ and the Kerr gates $e^{i\kappa\hat{a}_1^\dagger\hat{a}_1\hat{a}_2^\dagger\hat{a}_2}$.

More advanced non-Gaussian gates can be simulated using all the Gaussian gates presented above and at least one non-Gaussian gate.

2.1.4 Measuring a CV state

In most applications of traditional quantum computing, a measurement consists in projecting the state on the computational basis (usually the Z-Pauli eigenstates). On the other side, CV states make use of many different bases and therefore also require to perform several different types

of measurements. The most common types of measurements are homodyne, heterodyne, and photon-counting, each corresponding to one of the different bases presented above.

Photon-counting measurement

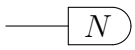


Figure 2.12: Symbol of a Fock measurement

Photon-counting, also called *Fock measurement* or *photon-number resolving measurement*, consists in projecting the state into the Fock basis. It is therefore represented by the set $\{|n\rangle\langle n|\}_{n\in\mathbb{N}}$ of projective operators. When one mode of a multimode Gaussian state is measured in the Fock basis, the remaining state will become non-Gaussian. It can therefore be used as a method to create non-Gaussian states [39].

Homodyne measurement

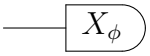


Figure 2.13: Symbol of a homodyne measurement on the eigenstates of $\hat{X}_\phi$

A *homodyne measurement* consists in projecting the state into a rotated quadrature, i.e. an eigenstate of $\hat{X}_\phi = \cos(\phi)\hat{X} + \sin(\phi)\hat{P}$. In the Wigner representation, it means projecting it on a a rotated axis. Homodyne measurements are represented by the set $\{|x_\phi\rangle\langle x_\phi|\}_{x_\phi\in\mathbb{R}}$ of projective operators.

A multimode Gaussian state remains Gaussian after a homodyne measurement on one mode [17].

Heterodyne measurement

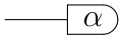


Figure 2.14: Symbol of a heterodyne measurement

We saw that the coherent states form another (non-orthogonal) basis of the Hilbert space. Therefore, it is also possible to define a measurement that projects the state into a coherent state $|\alpha\rangle$: that's what we call a *heterodyne measurement*. They can be seen as a simultaneous measurement of $\hat{x}$ and $\hat{p}$ with some uncertainty [17]. They are represented by the set $\{|\alpha\rangle\langle\alpha|\}_{\alpha\in\mathbb{C}}$ of projective operators.

A multimode Gaussian state remains Gaussian after a heterodyne measurement on one mode [17].

2.2 Quantum neural networks

2.2.1 General idea

Most machine learning tasks can be framed as function approximation problems, whether the function is the class of an input image, the translation of a sentence or the best action to take in a game. Neural networks are a particular type of function approximators that have been particularly successful to process images, texts and sounds. In the same way, the notion of quantum neural networks has appeared recently to designate several different types of quantum function approximators.

Formally, one can define a *quantum neural network (QNN)*, also called *variational circuit* or *parametrized circuit*, as a circuit depending on a set of parameters that can be trained to approximate a certain set of quantum functions. In general, a QNN can be written $f_\theta(\rho) = \text{tr}\left[\hat{B}U^\dagger(\theta)\rho U(\theta)\right]$ where $\hat{B}$ is an observable and $U(\theta)$ a parametrized unitary matrix. A circuit architecture $U(\theta)$ is called a *variational form*, or *ansatz*. If an ansatz on n qubits (or qumodes in the CV case) can approximate any unitary matrix acting on those qubits (or qumodes), it is said to be *universal*. Examples of universal ansatz for one and two qubits are given in Figure 2.15 and 2.16.

One of the main challenges of QNNs is to approximate non-linear quantum functions, since typical quantum circuits are unitary, and therefore linear. In practice, non-linear quantum functions arise when you either consider quantum properties such as the purity $\rho \mapsto \text{tr}[\rho^2]$, or embed classical data into the quantum realm to speed the approximation of classical functions: $\rho(x) \mapsto f(x)$. To solve it, one can either introduce measurements in the circuit [35] or have the circuit act on several copies of the input [33].

Another challenge is to find the best ansatz for a given class of functions. Universal ansatz can be constructed in general in both the discrete [40] and CV [37] cases, but they might not be the most efficient if the functions have some known symmetries. Examples of ansatz respecting some symmetries are convolutional QNN for quantum many-body systems [35] and UCCSD for chemistry [41].

2.2.2 Training

Training a QNN consists in finding the parameters that minimize a certain cost function. If the problem is a supervised learning problem, i.e. we have access to a dataset of input/output states $\{(\rho_{in}^{(i)}, \rho_{out}^{(i)}\}_i$, and we call our QNN f_θ, an example of objective is

$$\min_\theta \frac{1}{2} \sum_i ||f_\theta(\rho_{in}^{(i)}) - \rho_{out}^{(i)}||_1$$

where $\frac{1}{2}||\rho - \sigma||_1 = \frac{1}{2} \operatorname{tr}\left[\sqrt{(\rho-\sigma)^\dagger(\rho-\sigma)}\right]$ is the trace distance.

Calling our cost function $C(\theta)$, there are two ways to optimize it:

- Non-differentiable methods: evaluating $C(\theta)$ at several nearby points and deducing a good update for the parameters. Examples of algorithms include Nelder-Mead and gradient descent with numerical gradient calculation.
- Differentiable methods: evaluating the gradient of $C(\theta)$ analytically before performing any variant of gradient descent.

For QNNs, the main challenge of differentiable methods is to efficiently compute the gradient $\nabla_\theta f_\theta(\rho)$. The library Strawberry Fields [17] implements it for CV gates on its simulator using automatic differentiation (since all operations involved are made of linear transformations and elementary functions). However, when $f_\theta(\rho)$ is evaluated on a real quantum computer, $\nabla_\theta f_\theta(\rho)$ must as well be evaluated on the quantum computer. A general method to compute the gradient of a quantum circuit on a quantum computer has recently been introduced in [42] and implemented in the library PennyLane [18]. We will briefly describe the method here for the discrete case, since we use it extensively in our experiments.

Let $\mu \in \theta$ a scalar parameter affecting only one gate $\mathcal{G}(\mu)$, such that the circuit can be written $U(\theta) = V\mathcal{G}(\mu)W$. The unitary $\mathcal{G}(\mu)$ can be written in its exponential form

$$\mathcal{G}(\mu) = e^{-i\mu G} \tag{2.33}$$

where G is Hermitian. Assuming that G has two distinct eigenvalues a and b (degenerate in the case of a multi-qubit gate), we can multiply G by a phase factor without loss of generality (since the global phase is not observable), such that its eigenvalues become $\pm r$. The partial derivative of $f_\theta(\rho) = \text{tr}\left[\hat{B}U^\dagger(\theta)\rho U(\theta)\right]$ is then given by the following theorem [42] :

Theorem 1. *If $\mathcal{G}(\mu) = e^{-i\mu G}$ and G has two distinct eigenvalues $\pm r$, we have*

$$\partial_\mu f_\theta(\rho) = r\left(f_\theta(\rho + s) - f_\theta(\rho - s)\right) \tag{2.34}$$

where $s = \frac{\pi}{4r}$

Computing each component of the gradient therefore comes down to evaluating the circuit twice, with shifted parameters. For example, if $\mathcal{G}(\mu)$ is a rotation $R_x(\phi) = e^{-i\phi\hat{X}/2}$, then $r = \frac{1}{2}$ and $s = \frac{\pi}{2}$. Similar forms of the gradient can be derived in the case where there are more than two distinct eigenvalues, as well as in the CV case [42].

Once we have a method to compute the gradient, most first-order optimization algorithms used in machine learning can be used here, such as Adam [43] or AMSGrad [44].

2.2.3 Discrete ansatz

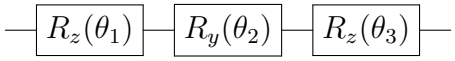


Figure 2.15: Universal ansatz for one qubit: any 2x2 unitary matrix can be constructed with this circuit [45]

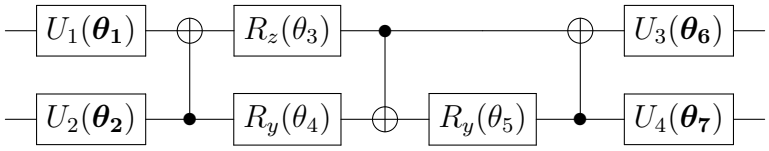


Figure 2.16: Universal ansatz for two qubits, that can be shown to be optimal in the number of continuous parameters and CNOT gates [45, 46]. Here, U is a universal ansatz for one qubit, such as the one represented in Figure 2.15.

In the discrete setting, there are several ways to build universal ansatz. For one and two qubits—the maximum number of qubits simulated in

our experiments—we considered the ansatz proposed in [46], made of rotations and CNOT, and represented in Figures 2.15 and 2.16. A more general universal ansatz is proposed in [40], working for any number of qubits and also made of rotations of CNOT.

2.2.4 Continuous ansatz

We will present here a very general ansatz for continuous-variable, introduced in [37]. It consists of a succession of layers (that we will call Nathan's layers), each layer being made of linear optics (rotations and beam splitters), displacement, squeezing and non-Gaussian gates, as displayed in Figure 2.17. This architecture can approximate any CV unitary matrix by adding enough layers, which makes it universal (in a weaker sense than for qubits).

This CV QNN can be seen as a classical neural network acting on the phase space. Indeed, as we saw in Section 2.1.3, the Gaussian component of each layer acts as a linear transformation of the phase space with a bias term, while the non-Gaussian gates act as a non-linearity:

$$\begin{pmatrix} \mathbf{x} \\ \mathbf{p} \end{pmatrix} \mapsto \Phi \left(M \begin{pmatrix} \mathbf{x} \\ \mathbf{p} \end{pmatrix} + \begin{pmatrix} \boldsymbol{\alpha_R} \\ \boldsymbol{\alpha_I} \end{pmatrix} \right) \tag{2.35}$$

where M is a symplectic matrix and Φ a non-linear function. Moreover, the succession of gates that constitutes the Gaussian part of the layer is shown in [37] to generate the whole space of symplectic matrices, so all the possible linear transformations of the phase space.

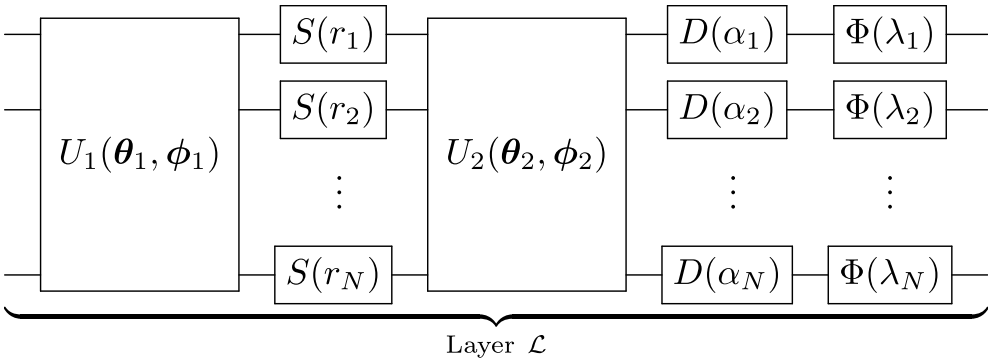


Figure 2.17: Nathan's layer acting on N qumodes, where U_1 and U_2 are two universal linear interferometers [37]

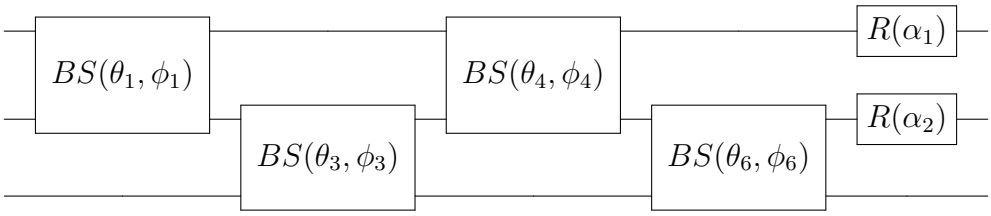


Figure 2.18: Linear interferometer acting on 3 qumodes

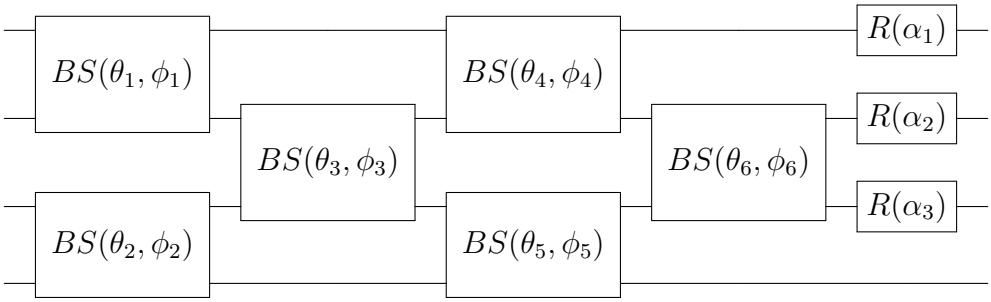


Figure 2.19: Linear interferometer acting on 4 qumodes

2.3 Quantum property estimation

In this work, we leverage quantum neural networks to solve the problem of estimating properties of quantum states. A property can be defined as a functional $f : \mathcal{S}(n) \to \mathbb{R}$ from the space $\mathcal{S}(n)$ of n-qubit (or n-qumode) density matrices. The problem of quantum property estimation is to find an algorithm that compute $f(\rho)$ when ρ is the output of quantum device. When the algorithm is a quantum circuit whose output is $f(\rho)$, it is called a *direct estimation* [24], as opposed to full quantum state tomography. This section introduces different methods to directly estimate polynomial properties, and present the two functionals considered in this work: purity and von Neumann entropy.

2.3.1 Approximating polynomials

The first assumption is that f is a polynomial function of ρ. Examples include the purity function, defined by $f(\rho) = \mathrm{tr}[\rho^2]$, as well as all the functions of the form $f(\rho) = \mathrm{tr}[\rho^m]$, which can be used to compute Rényi entanglement entropies $H(\rho, m) = \frac{1}{1-m} \log(\mathrm{tr}[\rho^m])$. Considering all the functions that can be approximated as polynomial functions through their Taylor expansion, this case is rather general.

The first thing to notice is that if f is at least quadratic, it is not possible to find a unitary matrix U and an observable O such that $f(\rho) = \mathrm{tr}\left[OU^\dagger \rho U\right]$, since the RHS is linear in ρ while the LHS is not. Therefore, there is no circuit that takes a quantum state as input and returns $f(\rho)$ as the average of one of the qubits. A first idea could be to add ancilla qubits to the circuit, but since it does not create any non-linearity, it cannot solve the problem neither.

To alleviate this issue, [25] and [24] propose to use m copies of ρ as input, where m is the degree of f. In the case of discrete states, [25] proves two important statements:

Proposition 5. *Let $f : \mathcal{S}(n) \to \mathbb{R}$ a polynomial functional from the set $\mathcal{S}(n)$ of n-qubit density matrices. Then, for all $\rho \in \mathcal{S}(n)$:*

1. *there exists an observable O_f such that $f(\rho) = \mathrm{tr}[O_f \rho^{\otimes m}]$*
2. *there exists a unitary matrix U_f such that $f(\rho) = \sum_j o_j \langle j|U_f^\dagger \rho^{\otimes m} U_f|j\rangle$ where $|j\rangle$ is the j^{th} element of the computational basis and $\{o_j\}$ the eigenvalues of O_f (the ordering being determined in the construction of U_f)*

Let us study the consequences of those two statements. The first point reduces the computation of a functional to the computation of an observable. It means that if we can find a circuit that measures O_f, $f(\rho)$ will simply be equal to the average measurement. The second point proves the existence of an actual circuit U_f and algorithm to approximate $f(\rho)$ from this circuit: estimate the probability $p(j) = \langle j|U_f^\dagger \rho^{\otimes m} U_f|j\rangle$ of getting the output j, and compute $f(\rho) = \sum_j o_j p(j)$. The main downside of this algorithm is that in general estimating the probability distribution of the output is almost as demanding as performing a whole quantum state tomography.

Inspired by [24] and [23], [25] analyses another circuit—that we will call *Ekert circuit*—which requires this times only the average of one qubit in the computational basis:

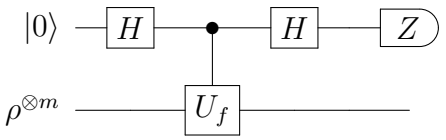


Figure 2.20: Ekert circuit to estimate polynomials using measurements of an ancilla qubit

The gate U_f can be any unitary such that $O_f = \frac{1}{2}\left(U_f + U_f^\dagger\right)$, for instance $U_f = O_f + i\sqrt{I - O_f^2}$. It is possible to show that

$$f(\rho) = \mathrm{tr}\left((Z \otimes I \otimes ... \otimes I)U\left(|0\rangle\langle 0| \otimes \rho^{\otimes m}\right)U^\dagger\right)$$

where U corresponds to the circuit above [25]. It means that taking the average of the observable Z on the ancilla qubit gives us the answer directly.

If this approach has the advantage to prove the existence of a circuit to estimate polynomials without tomography, its practical advantage is limited: except in some special cases, there is no guarantee that U_f can be prepared efficiently, as stated in [25]:

> *While such a circuit always exists, there is no guarantee that it will achieve the desired unitary transformation efficiently. Indeed, most unitary transformations cannot be approximated efficiently. A protocol of this nature would only be worthwhile if it were substantially more efficient than performing quantum state tomography on ρ and then calculating the function directly. [...]*
>
> *There is no known efficient algorithm for finding the simplest circuit that produces a given unitary transformation. Indeed, that is almost certain a computationally intractable problem in itself.*

This piece was written in 2002 before the introduction of variational circuits, which are now commonly begin used to find the simplest circuit that produces a given unitary transformation [45, 34]. The goal of our work is precisely to find this unitary using variational circuits.

2.3.2 Purity

The first experiments considered in this thesis concerned a particular polynomial: the purity. The purity of a state ρ is defined as $\mathrm{tr}[\rho^2]$ and quantifies how much a state is mixed. It has several important properties:

1. For a pure state ρ, $\mathrm{tr}[\rho^2] = \mathrm{tr}[\rho] = 1$
2. If ρ is an n-qubit state, $\frac{1}{2^n} \leq \mathrm{tr}[\rho^2] \leq 1$

The most common technique to compute purity is called the SWAP test [47, 48] and is defined by the circuit represented in Figure 2.21.

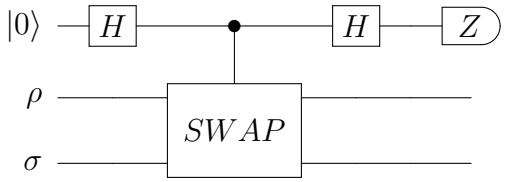


Figure 2.21: SWAP test: circuit to compute the state overlap of discrete states ρ and σ [47, 48]. When $\rho = \sigma$, the purity is computed

This circuit computes the state overlap between two states ρ and σ, defined as $\mathrm{tr}[\rho\sigma]$, using the average of Z in the ancilla qubit as the output. It can therefore evaluate the purity when $\rho = \sigma$. It can be seen as a particular case of the circuit presented in Figure 2.20. Indeed, the SWAP operator can be written

$$\hat{S} = \sum_{i,j} |i\rangle \langle j| \otimes |j\rangle \langle i| \tag{2.36}$$

which gives us

$$\begin{aligned} \mathrm{tr}\left[\hat{S}\rho^{\otimes 2}\right] &= \sum_{i,j} \mathrm{tr}\left[|i\rangle \langle j| \rho \otimes |j\rangle \langle i| \rho\right] \\ &= \sum_{i,j} \mathrm{tr}\left[|i\rangle \langle j| \rho\right] \left[|j\rangle \langle i| \rho\right] \\ &= \sum_{i,j} \rho_{ji}\rho_{ij} \\ &= \mathrm{tr}\left[\rho^2\right] \end{aligned} \tag{2.37}$$

Since $\hat{S}$ is both unitary and Hermitian, it proves that $U_f = O_f = \hat{S}$ for $f : \rho \mapsto \mathrm{tr}[\rho^2]$ (following the notations of Proposition 5)

Direct estimation of the purity is also possible for continuous-variable states, and also comes down to computing the average $\mathrm{tr}\left[\hat{S}\rho^{\otimes 2}\right]$ of the SWAP operator. [26] shows that a circuit made of a 50:50 beam splitter $e^{i\frac{\pi}{4}(\hat{a}_1\hat{a}_2^\dagger + \hat{a}_1^\dagger\hat{a}_2)}$ and the measurement of the parity observable $(-1)^{\hat{N}}$ on the first qumode—as represented in Figure 2.22—does the job.

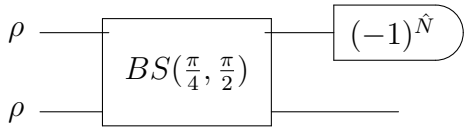


Figure 2.22: Circuit to compute the purity of a CV state ρ [26]

Chapter 3

Method

In this thesis, we propose three models to compute polynomials of states using quantum circuits. The first one, *Quantum Polynomial Network with ancilla (PolyNet-ancilla)*, is an adaptation of the Ekert circuit presented in Section 2.3.1 where the central part is a variational circuit. The second one, *Quantum Polynomial Network with correlation (PolyNet-corr)*, uses n-points correlations between each qubit of a quantum circuit as input of a linear network. The final one, *Quantum Polynomial Network with averages (PolyNet-avg)*, uses the average of all the qubits as input of non-linear neural network. In this section, we will describe the three architectures in both the discrete and CV cases, give some universality results, and specify how to train those circuits using an artificially generated dataset.

In this section, we will call n the number of qubits of our input states, $f : \mathcal{S}(n) \to \mathbb{R}$ the polynomial function we try to approximate, with $S(n)$ the space of n-qubit density matrices, and m the degree of f.

3.1 Quantum Polynomial Network with ancilla

3.1.1 Discrete-variable case

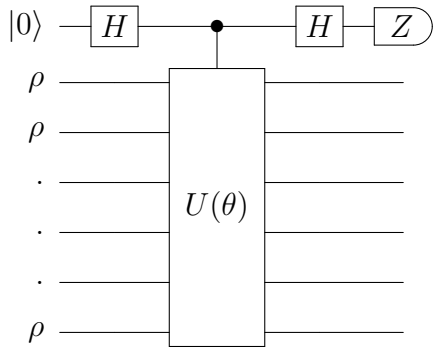


Figure 3.1: PolyNet with ancilla, an adaptation of the Ekert Circuit where the unitary $U(\theta)$ is a parametrized circuit trained to approximate the polynomial f

The first architecture considered in this project is an adaptation of the Ekert circuit introduced in Section 2.3.1, with a variational part. The circuit, represented in Figure 3.1, consists of two Hadamard gates and a controlled unitary part $U(\theta)$ depending on some parameters θ. To evaluate a polynomial f of degree m, it requires m copies of ρ as input. The predicted value of $f(\rho)$ is read by measuring the ancilla qubit in the Z-basis and taking the average.

We saw the existence of a unitary matrix U_f corresponding to every polynomial f. So, if $U(\theta)$ is a universal ansatz, we can find a vector θ^* such that $U(\theta) = U_f$. Therefore, PolyNet-ancilla is a universal polynomial approximator.

The two main advantages of this architecture are its universality and the measurement of only one qubit for the output. The main downside is the necessity to add an ancilla qubit, which can be a real constraint for near-term devices.

3.1.2 Continuous-variable case

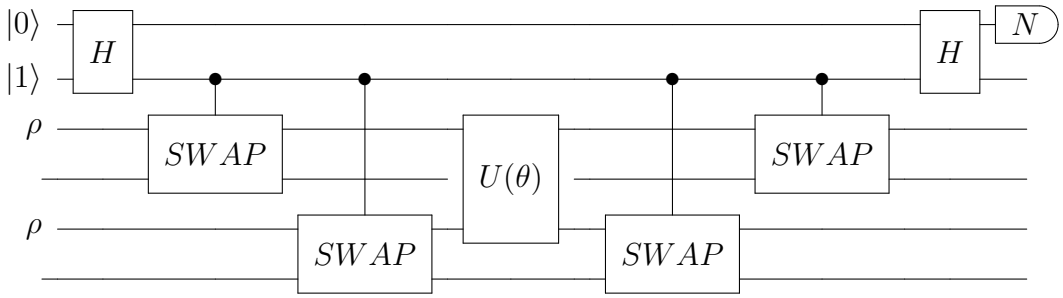


Figure 3.2: PolyNet with ancilla generalized to CV systems. The two first qumodes simulate an ancilla qubit which controls the application of $U(\theta)$ through the controlled-SWAP, generalizing the technique proposed in [49] for hybrid discrete/CV systems. Circuits for polynomials of higher degrees can be constructed following the same pattern with more copies of ρ

The Ekert circuit can also be adapted in the continuous-variable case, by using a hybrid qubit/qumode architecture. As in the discrete setting, the variational component takes m copies of the state ρ as input. However, ρ is this time continuous and so is the unitary $U(\theta)$. On the other hand, the ancilla part is made of Hadamard gates and a control, and must therefore remain a two-level system.

In a full CV device, the qubit component must be simulated with qumodes. Several methods exist for that purpose [50]. The simplest approach is to encode the logical qubits $|0_L\rangle$ and $|1_L\rangle$ in two elements of a CV basis: the Fock states $|0\rangle$ and $|1\rangle$, coherent states $|\alpha\rangle$ and $|-\alpha\rangle$, etc. However, most discrete gates are challenging to simulate using this encoding and usual CV gates. In the so-called *dual-rail encoding*, each qubit is instead simulated using two qumodes:

$$\begin{aligned} |0_D\rangle &\coloneqq |0_L 1_L\rangle \\ |1_D\rangle &\coloneqq |1_L 0_L\rangle\,, \end{aligned} \tag{3.1}$$

for any two single-mode basis states $|0_L\rangle$ and $|1_L\rangle$. If we take $|0_L\rangle$ and $|1_L\rangle$ to be the Fock states $|0\rangle$ and $|1\rangle$, [51] noticed that the rotation of a dual-rail qubit in the Bloch sphere can be performed using beam splitters of the form $BS\left(\theta, \frac{\pi}{2}\right) = e^{i\theta(\hat{a}_1\hat{a}_2^\dagger + \hat{a}_1^\dagger\hat{a}_2)}$:

$$BS\left(\theta, \frac{\pi}{2}\right)|0_L 1_L\rangle = \cos(\theta)\,|0_L 1_L\rangle + i\sin(\theta)\,|1_L 0_L\rangle \tag{3.2}$$

In particular, a Hadamard gate corresponds to the operator $BS\left(\frac{\pi}{4}, \frac{\pi}{2}\right)$.

In order to adapt the Ekert circuit for CV states, we finally need to construct hybrid controlled unitaries. In the qubit setting, [49] suggests a procedure to turn any unitary operation into a controlled unitary. It consists in associating an ancilla qubit to each input qubit and applying a controlled-SWAP to each pair, before and after the unitary. That way, if the control qubit is $|0\rangle$, the SWAP is not applied and the unitary is. If it is $|1\rangle$, the state is temporary stored on an other set of qubits via the SWAP, and restored onto the original qubits after the unitary has been applied to some trash qubits. To have a usual controlled unitary operation where the unitary is applied when the control qubit is $|1\rangle$, a NOT can be applied on the control at the beginning.

Fortunately, the controlled-SWAP is one of the only hybrid gates considered in the literature. For instance, [51] proposes an implementation using beam splitters and quartic gates.

The final circuit is given in Figure 3.2. One last difficulty is the preparation of the single-photon state $|1\rangle$ in the second qumode of the circuit. A simple way to do it is to learn this preparation. [14] showed a high fidelity in the preparation of single-photon states with a CV QNN.

3.2 Quantum Polynomial Network with correlations

3.2.1 Discrete-variable case

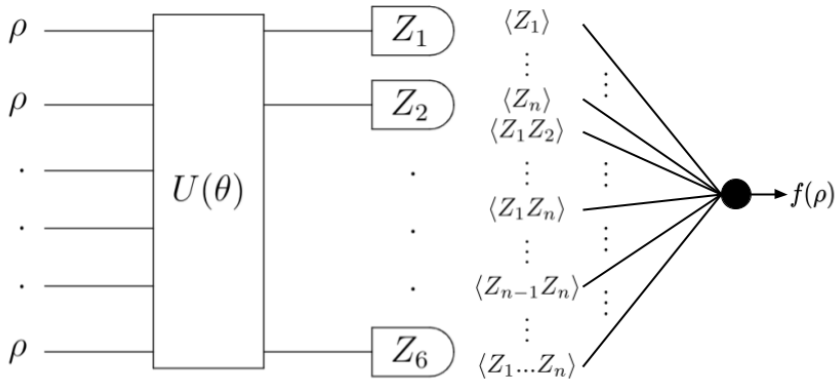


Figure 3.3: Quantum Polynomial Network with correlations (PolyNet-corr). The first part consists in a quantum circuit $U(\theta)$ that takes $\rho^{\otimes m}$ as input. Then, a succession of measurements is performed on the Z basis, from which can be extracted all the n-points correlations. Those 2^{nm} correlations form the input of a linear neural network whose output is an estimation of $f(\rho)$

Our second architecture is a QNN $U(\theta)$ followed by measurements of each qubit in the computational basis. Using those measurements, one can approximate all the correlation functions $\langle M_1 \otimes ... \otimes M_{nm} \rangle$ where $M_i = \mathbb{I}$ or Z. Those 2^{nm} averages are then passed as input to a classical linear neural network whose output is an estimation of $f(\rho)$. Proving that such an architecture is able to compute $f(\rho)$ is possible and comes down to proving the following theorem:

Theorem 2 (Universality of PolyNet-corr)**.** *Let $f : \mathcal{S}(n) \to \mathbb{R}$ a polynomial function of degree m from the space $\mathcal{S}(n)$ of n-qubit density matrices.*

Then:

$$\exists U_f \in \mathcal{U}(2^{nm}), \exists \alpha \in \mathbb{R}^{2^{nm}}, \forall \rho \in \mathcal{S}(n), f(\rho) = \sum_i \alpha_i \operatorname{tr}\left[M^{(i)} U_f \rho^{\otimes m} U_f^\dagger\right] \tag{3.3}$$

where each $M^{(i)}$ *is an observable from the set* $\{M_1 \otimes ... \otimes M_{nm} | \forall j, M_j \in \{\mathbb{I}, Z\}\}$

Proof. By Proposition 5, we know that there exists O_f such that $f(\rho) = \operatorname{tr}[O_f \rho^{\otimes m}]$. Therefore, proving (3.3) is equivalent to proving the following:

$$\begin{aligned}\operatorname{tr}\left[O_f \rho^{\otimes m}\right] &= \operatorname{tr}\left[\left(\sum_i \alpha_i M^{(i)}\right) U_f \rho^{\otimes m} U_f^\dagger\right] \\ &= \operatorname{tr}\left[U_f^\dagger \left(\sum_i \alpha_i M^{(i)}\right) U_f \rho^{\otimes m}\right]\end{aligned}$$

In particular, this equality is true if

$$O_f = U_f^\dagger \left(\sum_i \alpha_i M^{(i)}\right) U_f \tag{3.4}$$

Since $\sum_i \alpha_i M^{(i)}$ is diagonal (as a linear combination of tensor products of diagonal matrices), (3.4) is simply a statement about the diagonalization of O_f. To prove that there exists U_f and α such that (3.4) is true, we therefore need to show that $\sum_i \alpha_i M^{(i)}$ can represent any diagonal matrix. It comes down to proving the following lemma:

Lemma 1. *The set* $\{M_1 \otimes ... \otimes M_{nm} | \forall j, M_j \in \{\mathbb{I}, Z\}\}$ *forms a basis of* $\mathcal{D}(2^{nm})$ *(space of diagonal matrices in dimension* 2^{nm}*).*

To prove Lemma 1, one can consider the vectors m_j consisting of the diagonal elements of each matrix M_j and reduce the lemma to a statement about the elements $m_1 \otimes ... \otimes m_{nm}$. Let $h : \mathbb{R}^2 \to \mathcal{D}(2)$ the isomorphism that associates to any vector the corresponding diagonal matrix: $h(m_j) = M_j$. By definition, each m_j is either $\begin{pmatrix}1\\1\end{pmatrix}$ (if $M_j = \mathbb{I}$) or $\begin{pmatrix}1\\-1\end{pmatrix}$ (if $M_j = Z$) and those two vectors forms a basis of $\mathbb{R}^2$. Therefore, the set consisting of all the tensor products $m_1 \otimes ... \otimes m_j$ forms a basis of $\mathbb{R}^2 \otimes ... \otimes \mathbb{R}^2 \cong \mathbb{R}^{2^{nm}}$. By construction, $M_1 \otimes ... \otimes M_{nm} = h(m_1) \otimes ... \otimes h(m_{nm}) = (h \otimes ... \otimes h)(m_1 \otimes ... \otimes m_{nm})$. Since the function $h \otimes ... \otimes h$ is

an isomorphism from $\mathbb{R}^{2^{nm}}$ to $\mathcal{D}(2^{nm})$, it maps a basis to another basis. Therefore $\{M_1 \otimes ... \otimes M_{nm} | \forall j, M_j \in \{\mathbb{I}, Z\}\}$ forms a basis of $\mathcal{D}(2^{nm})$

□

The advantages of this architecture are its universality and the absence of any ancilla qubit. The main downside is that the input of the linear network is exponentially large, since it contains all the 2^{nm} k-point correlations. The number of parameters to learn grows therefore exponentially as well. Another drawback is that estimating higher-order correlation functions with a low variance can require a large number of measurements.

3.2.2 Continuous-variable case

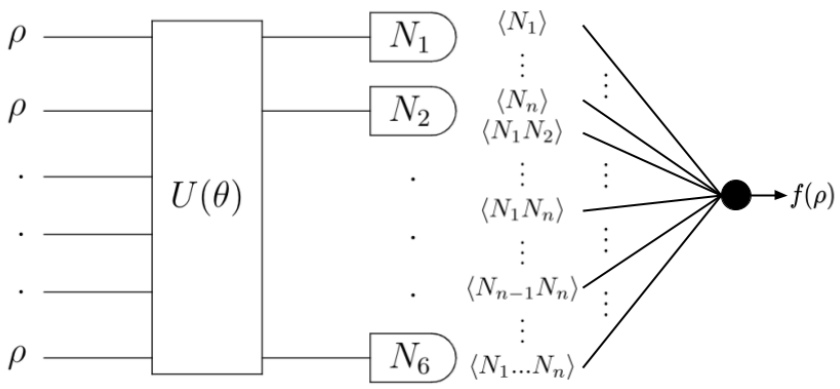


Figure 3.4: CV PolyNet-corr. The first part consists in a quantum circuit $U(\theta)$ that takes $\rho^{\otimes m}$ as input. Then, for all the qubits, the average of the observable N is computed and fed into a classical neural network whose output is an estimation of $f(\rho)$

To adapt PolyNet-corr for CV states, we replaced the measurements in the computational basis by measurements in the Fock basis. However, the universality property does not hold anymore. Indeed, the set of observables $\{M_1 \otimes ... \otimes M_{nm} | \forall j, M_j \in \{\mathbb{I}, N\}\}$ does not form a basis of all diagonal operators anymore, since this space is infinite and the set is

finite. Hence, 1 does not hold for CV states. We can therefore consider this architecture as a no-go result.

3.3 Quantum Polynomial Network with averages

3.3.1 Discrete-variable case

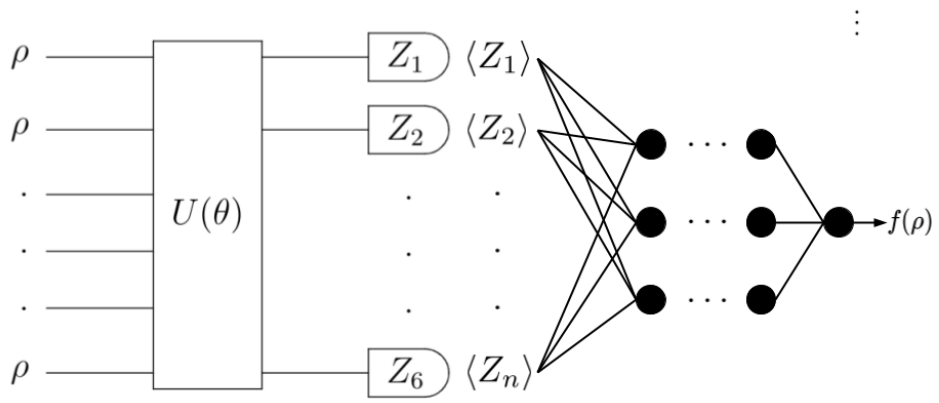


Figure 3.5: Quantum Polynomial Network with averages (PolyNet-avg). The first part consists in a quantum circuit $U(\theta)$ that takes $\rho^{\otimes m}$ as input. Then, for all the qubits, the average of the observable Z is computed and fed into a classical neural network whose output is an estimation of $f(\rho)$

The last architecture considered in this work is a QNN $U(\theta)$ followed by measurements of each qubit in the computational basis. The averages of all the measurements are passed into a classical non-linear neural network. At the time of this writing, the universality has not been proved theoretically, even though it is suggested by our experiments (see Chapter 4). We therefore enounce it as a conjecture:

Conjecture 1 (Universality of PolyNet-avg)**.** *Let $f : \mathcal{S}(n) \to \mathbb{R}$ a polynomial functional of degree m from the space $\mathcal{S}(n)$ of n-qubit density*

matrices (2^n dimensional Hermitian semi-positive matrices of trace 1). Then:

$$\begin{gathered}\exists U \in \mathcal{U}(2^{nm}), \exists g \in \mathcal{C}^0(\mathbb{R}^{nm}), \forall \rho \in \mathcal{S}(n), \\ f(\rho) = g\left(\mathrm{tr}\left[Z_1 U \rho^{\otimes m} U^\dagger\right], ..., \mathrm{tr}\left[Z_{nm} U \rho^{\otimes m} U^\dagger\right]\right)\end{gathered} \tag{3.5}$$

where each Z_i is a $Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$ operator applied to the i^{th} qubit: $Z_i = \mathbb{I} \otimes ... \otimes Z \otimes ... \otimes \mathbb{I}$

This architecture has the benefits to use only a linear number of observables and not to use any ancilla. However, it requires to learn both a classical and a quantum neural network at the same time, which can make the training part harder than the two previous architectures (as we also noticed in the experiments).

3.3.2 Continuous-variable case

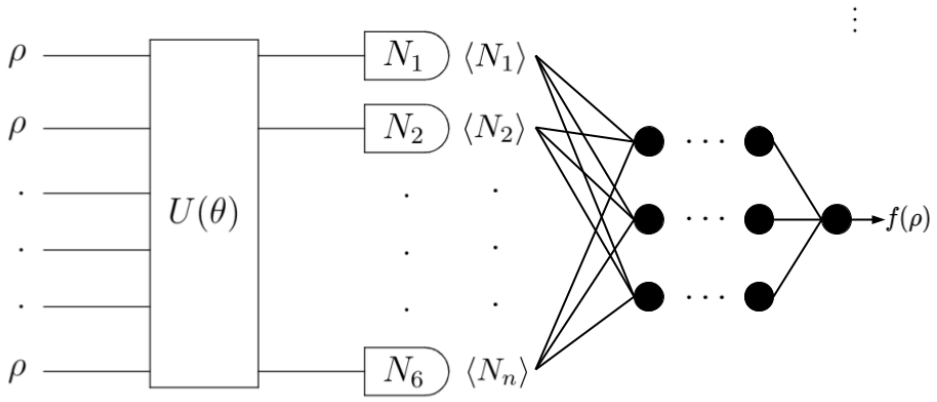


Figure 3.6: CV PolyNet-avg. The first part consists in a quantum circuit $U(\theta)$ that takes $\rho^{\otimes m}$ as input. Then, for all the qubits, the average of the observable N is computed and fed into a classical neural network whose output is an estimation of $f(\rho)$

As for PolyNet-corr, the adaptation of PolyNet-avg for CV states simply consists in replacing the measurements in the Pauli basis by measure-

ments in the Fock basis. The universality of this CV architecture remains an open question at the time of writing.

3.4 Training procedure

3.4.1 Dataset preparation

In order to train our model, we need a dataset of elements $\{(\rho_i, f(\rho_i))\}_i$, where the density matrices ρ_i are the inputs of our network and $f(\rho_i)$ the target outputs. Since PolyNet takes quantum data as input, every density matrix needs to be prepared with a circuit. Therefore, creating the dataset requires two steps: generating density matrices on a classical computer and building quantum circuits that prepare each state.

Generating the dataset

Learning a model with the best generalization capabilities requires a dataset that covers the input and target spaces as uniformly as possible. For that, we generate random density matrices using the following construction (d is the size of the Hilbert space in consideration):

1. Randomly generate $(\lambda_i)_{1\leq i\leq d} \in [0,1]^d$ such that $\sum_i \lambda_i = 1$ using the procedure below:
   (a) Sample λ_1 uniformly in $[0,1]$
   (b) At step $i \geq 2$, sample λ_i uniformly in $[\lambda_{i-1}, 1]$
2. Sample a unitary matrix U with the Haar measure.
3. $\rho = U \begin{pmatrix} \lambda_1 & & (0) \\ & \ddots & \\ (0) & & \lambda_d \end{pmatrix} U^\dagger$

Then, in order to obtain balanced target values, we perform a rejection sampling step. We first discretize the output set $f(S(n))$: let $t_0, ..., t_N \in f(S(n))$ such that $t_0 = \min f(S(n))$ and $t_{i+1} = t_i + \epsilon$ (where $\epsilon = \frac{\max f(S(n)) - \min f(S(n))}{N}$ is a hyperparameter, fixed at 0.01 in our experiments). Then, for all t_i, we generate density matrices ρ using the method above and only keep those for which $f(\rho) \in [t_i, t_{i+1}]$, By preparing the same number of states for all t_i, we ensure that the dataset will be balanced.

In the CV case, the library Strawberry Fields requires a cutoff c in the Fock space to simulate CV circuits, meaning that $|c\rangle$ will be the eigenstate of higher energy. Therefore, the density matrices will be of size $d = c^n$ (with n the number of qumodes) and can be generated using the method above.

Preparing the states

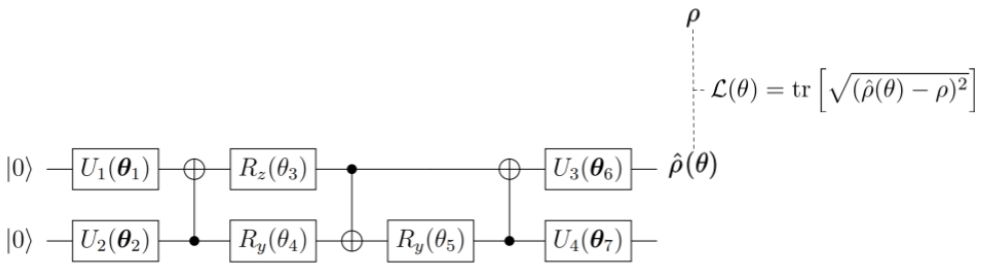


Figure 3.7: State preparation procedure for one qubit, using the universal ansatz presented in Section 2.2.3

The next step is to prepare each density matrix of our dataset with a quantum circuit. Since the output of a circuit is always a pure state, we need to use a common trick to encode generic density matrices: *purification*. For every state ρ acting on a Hilbert space H_A, there exists another system H_B (with the same dimension as H_A) and a pure state $|\psi\rangle \in H_A \otimes H_B$, such that $\mathrm{tr}_B\left[|\psi\rangle\langle\psi|\right]$ [52]. Therefore, to prepare a state ρ with n qubits, we can use a variational circuit with $2n$ qubits and learn to prepare a state $|\psi\rangle$ such that $\rho = \mathrm{tr}_B\left[|\psi\rangle\langle\psi|\right]$ approximates ρ, as presented in Figure 3.7 for 1-qubit states.

As a cost function, we use the trace distance between the prepared state $\hat{\rho}(\theta)$ and the real state ρ:

$$\begin{aligned} \mathcal{L}(\theta) &:= \mathrm{tr}\left[\sqrt{(\hat{\rho}(\theta) - \rho)^2}\right] \\ &= \sum_i |\lambda_i| \end{aligned} \tag{3.6}$$

where the λ_i are the eigenvalues of $(\hat{\rho}(\theta) - \rho)$.

Since our goal is to generate a dataset of prepared states with balanced property values (e.g. balanced purity), conserving the property of

each state is particularly important. Therefore, we also add a regularization term in the loss in order to minimize the difference in property of the real and generated states:

$$\mathcal{L}(\theta) := \operatorname{tr}\left[\sqrt{(\hat{\rho}(\theta)-\rho)^2}\right] + \lambda\left(f(\hat{\rho}(\theta)) - f(\rho)\right)^2 \tag{3.7}$$

3.4.2 Training

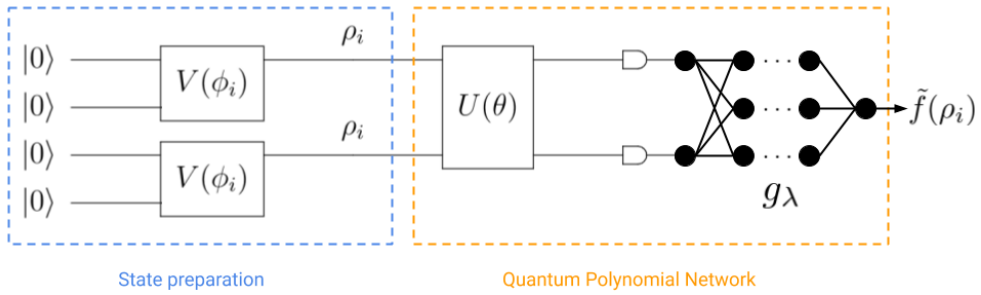


Figure 3.8: Complete circuit used during training. The state preparation part processes each input state ρ_i. The second part is one of the three polynomial network presented above: it's composed of a QNN $U(\theta)$, measurements and a classical network g_λ.

Once every state ρ_i of the dataset has been prepared, each ρ_i can be associated to a circuit $V(\phi_i)$. The polynomial network is trained to minimize the following loss function:

$$\mathcal{L}(\theta, \lambda) = \sum_i \left(\tilde{f}(\rho_i) - f(\rho_i)\right)^2 \tag{3.8}$$

where θ represents the parameters of the quantum circuit $U(\theta)$ and λ of the classical neural network g_λ. The optimization procedure can be performed using gradient descent, as explained in Section 2.2.

Chapter 4

Experiments

We performed experiments with the three proposed architectures for both discrete and continuous variables. Two functionals were considered in those experiments: purity and entropy.

The code used to perform the experiments is available at github.com/artix41/quantum-polynomial-network.

4.1 Software stack

All the experiments were performed in the programming language Python, using a set of four different libraries for quantum computing:

- PyQuil [53]
- Strawberry Fields [17]
- Pennylane [18]
- QuTiP [54]

PyQuil is a qubit-based circuit simulator. We used it to prepare discrete-variables states and to simulate the different architectures (through the PyQuil module in Pennylane).

Strawberry Fields is a library to simulate and optimize CV circuits, based on the machine learning library TensorFlow [55]. We used it to simulate and train all our CV QNN.

Pennylane is a framework made to optimize the parameters of both CV and discrete circuits by gradient descent. It allows to compute the gradient of quantum circuits and hybrid architectures directly on the

quantum computer. The circuit simulation part is carried out through modules corresponding to other libraries (PyQuil, Qiskit, Strawberry Fields, etc.). We used it to optimize our discrete-variables architectures.

QuTiP is a toolbox for manipulating quantum states. We used it to compute the Wigner representation of CV states in the different figures of Section 2.1.2, as well as to generate random density matrices in the state preparation part.

Apart from those quantum computing libraries, the usual scientific computing packages NumPy [56], SciPy [57] and MatplotLib [58] were also used extensively in this project.

4.2 State preparation

The first step to conduct our experiments was to prepare a dataset of states. Since the generation method as well as the loss function depend on the property we want to predict, we prepared different datasets for purity and entropy. In the CV case, since we did not perform any experiment on entropy, only a purity dataset was prepared.

For the preparation of CV states, we had to use a cutoff in the Fock space, as required by the library Strawberry Fields in order to simulate CV circuits. We tested our method with two cutoffs—3 and 5—since higher cutoff lead to much slower simulations.

For each dataset, we show three histograms to evaluate the performance of the preparation:

- The distribution of the property value (e.g. purity) in the dataset. Since the goal of the generation method presented in Section 3.4.1 was to have a balanced dataset, we expect a uniform distribution.

- The distribution of the trace distance (first term of the cost function) over all the prepared density matrices. We expect a distribution concentrated around zero if the preparation step has worked as desired.

- The distribution of the mean squared error between the property of the generated and the real density matrices (second term of the cost function). We expect a distribution concentrated around zero if the preparation step has worked as desired.

4.2.1 Discrete states

In the discrete state preparation experiments, we prepared a dataset of 400 samples, training the ansatz presented in Figure 2.16 with the cost function of equation 3.7. We used $\lambda = 1$ as our regularization coefficient and minimized the cost function using the non-differentiable version of the optimization algorithm L-BFGS-B (through its implementation in Scipy).

We can see in Figures 4.2 and 4.4 that the preparation procedure was particularly efficient: we reached a precision of less than 10^{-8} for the two parts of the cost function, for both the purity and the entropy. Figures 4.1 and 4.3 show that the distribution of the purity and the entropy over each dataset is very close to a uniform distribution, as expected.

Purity

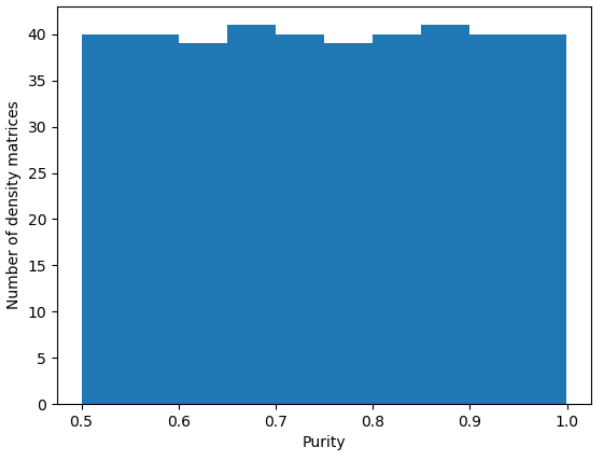


Figure 4.1: Distribution of the purity of density matrices generated using the method described in Section 3.4.1

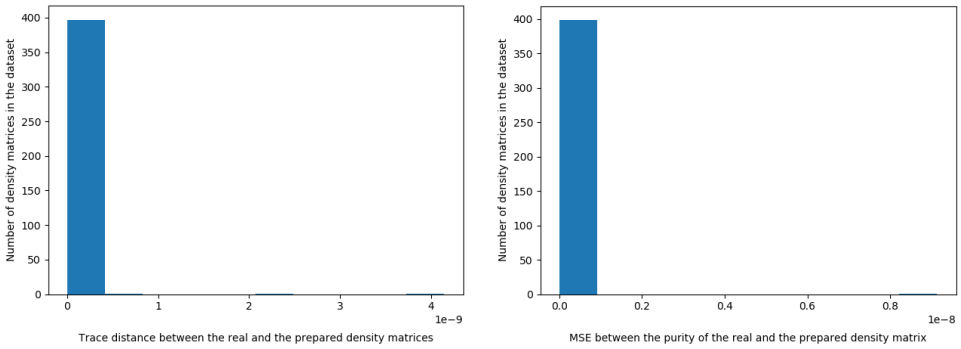


Figure 4.2: Distribution of the difference between real and prepared density matrices in our dataset, by comparing them using trace distance (left) and their difference in terms of purity (right)

Entropy

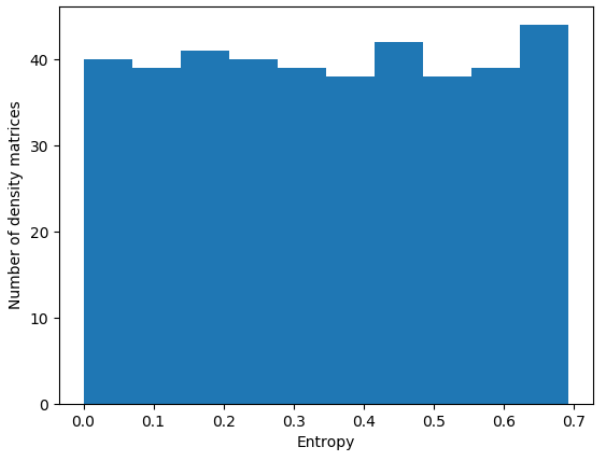


Figure 4.3: Distribution of the purity of density matrices generated using the method described in Section 3.4.1

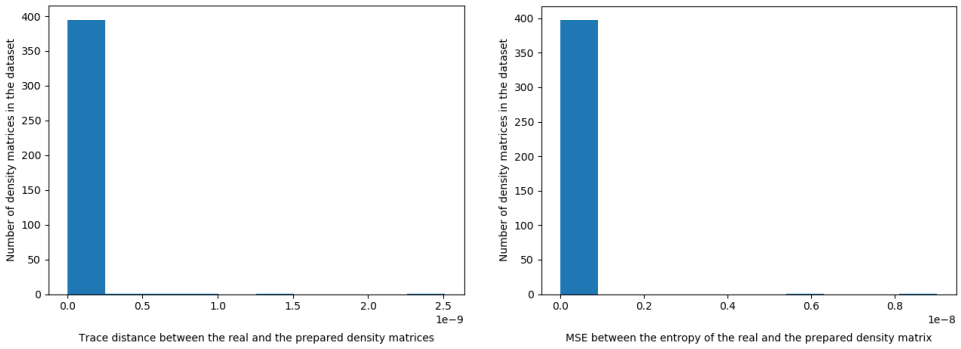


Figure 4.4: Distribution of the difference between real and prepared density matrices in our dataset, by comparing them using trace distance (left) and their difference in terms of entropy (right)

4.2.2 CV states

4.3 Discrete-variable Polynomial Network

Once we had a dataset of discrete density matrices and their corresponding property, we were able to train our three polynomial networks. As for the state preparation, we used the ansatz presented in Figure 2.16 as our QNN. We split our dataset with 300 training samples and 100 test samples. We trained each architecture until convergence with the library Pennylane, using the algorithm Adam with a learning rate of $\lambda = 5 \cdot 10^{-2}$ and a batch size of 16.

For each experiment, we show the graph of predicted vs true value of the property after training, for both the training and test sets. In those graphs, each point corresponds to a density matrix of the dataset. On top of each graph is displayed the value of the cost function. In a totally accurate predictor, all the points should be on the line $y = x$

4.3.1 PolyNet with ancilla

Learning purity

Learning purity with PolyNet-ancilla was our most successful experiment. As you can see in Figure 4.5, the cost function is below 10^{-7} and all the points seem to lie on the line $y = x$.

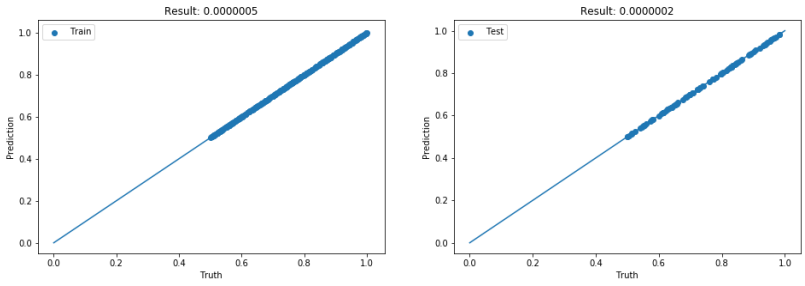


Figure 4.5: Results for PolyNet-ancilla for purity in the discrete case. **Left**: training set ; **Right**: test set

Learning entropy

Entropy is a non-linear functional of the state. Using only a quadratic function approximator, we therefore expect less accurate results than for purity. However, Figure 4.6 shows that our architecture came up with a relatively good approximation, with a cost function of $4 \cdot 10^{-4}$ on the test set. It sets a promising path for approximating non-linear functionals using polynomial estimators.

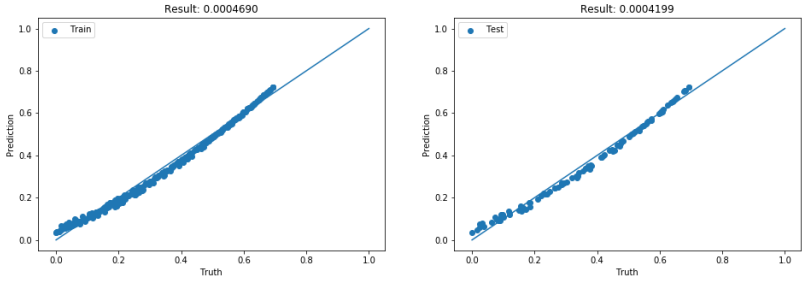


Figure 4.6: Results for PolyNet-ancilla for entropy in the discrete case. **Left**: training set ; **Right**: test set

4.3.2 PolyNet with correlations

Learning purity

Using PolyNet-corr to estimate the purity of qubit states was also a very successful experiment. As we can see in Figure 4.7, the cost function converges to 10^{-6} and all the points look aligned.

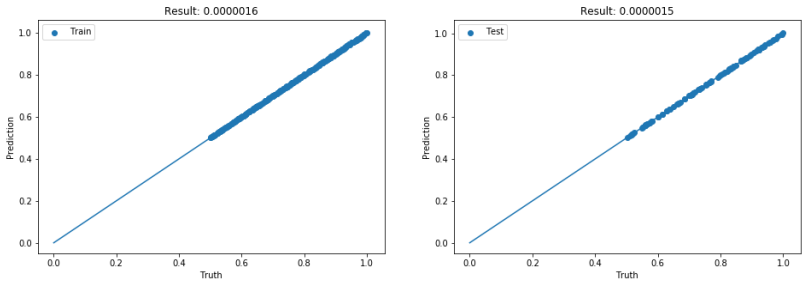


Figure 4.7: Results for PolyNet-corr for purity in the discrete case. **Left**: training set ; **Right**: test set

Learning entropy

The results for entropy are also very similar to PolyNet-ancilla, with a convergence of the cost function to $4 \cdot 10^{-4}$

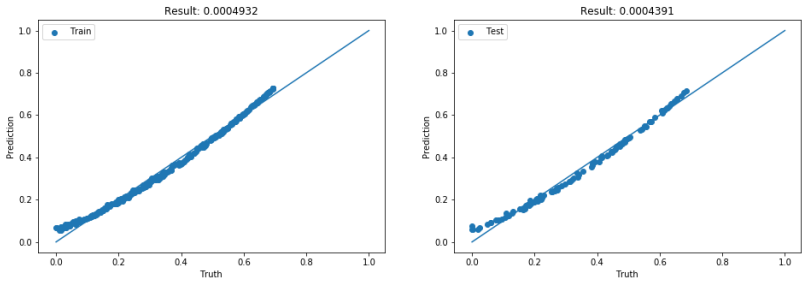


Figure 4.8: Results for PolyNet-corr for entropy in the discrete case. **Left**: training set ; **Right**: test set

4.3.3 PolyNet with averages

We only trained PolyNet-avg with purity: the experiment for entropy is let as future work. The last component of this architecture is a classical neural network. We used 4 hidden layers, 20 hidden nodes per layer, and ReLU as an activation function, except for the output layer where we used a sigmoid.

The results are less encouraging than for the two previous architectures, with a convergence of the cost function to $2 \cdot 10^{-4}$. Several reasons may account for this phenomenon:

- We did not prove the universality of PolyNet-avg. Therefore, it is

possible that it is not universal, and in that case that purity cannot be estimated exactly by this architecture.

- Training non-linear neural networks is a harder task than the linear neural network PolyNet-corr. It is therefore possible that a better tuning of the hyperparameters (number of nodes/layers, learning rate, etc.) or a longer training would have improved the performance.

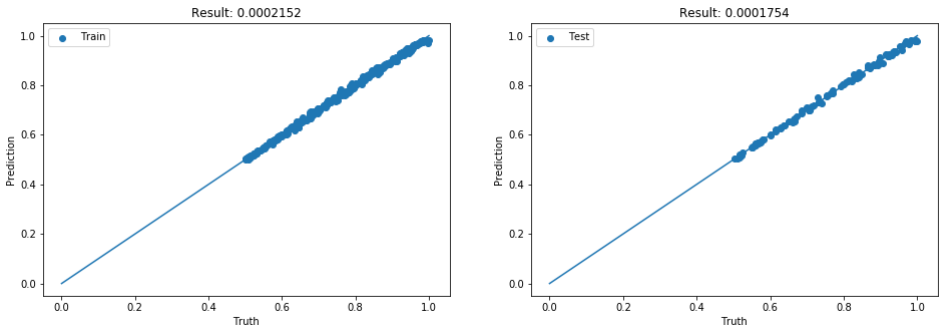


Figure 4.9: Results for PolyNet-avg for purity in the discrete case. **Left**: training set ; **Right**: test set

4.4 Continuous-variable Polynomial Network

We also trained our architectures to predict the properties of CV states. However, as explained in the state preparation part, a cutoff on Fock space was required for the simulation, and we used 3 by default for most experiments. One experiment with cutoff 5 was performed, for PolyNet-avg.

4.4.1 PolyNet with ancilla

Since the CV version of PolyNet-ancilla has been invented at a very late stage of the thesis, I did not have time to perform experiments on it. Those experiments are let as future work.

4.4.2 PolyNet with correlations

With a convergence of the cost function to $2 \cdot 10^{-4}$, the results for CV PolyNet-corr were encouraging. However, they are worst than in the

qubit case and the apparent convergence might be due to the small cutoff.

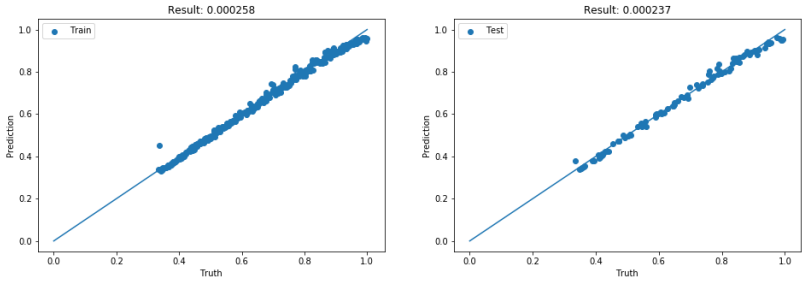


Figure 4.10: Results for PolyNet-corr for purity in the CV case. **Left**: training set ; **Right**: test set

4.4.3 PolyNet with averages

We tested PolyNet-avg with two different cutoffs on the Fock space: 3 and 5. As you can see in Figures 4.11 and 4.12, there is a big difference between the two experiments: increasing the cutoff leads to decreased performance. It suggests that the cutoff plays an important role when running our CV architectures. Simulating CV states with cutoff 3 is similar to simulating qutrits, and the task of estimating the property of qutrits is easier than for general CV states.

Cutoff 3

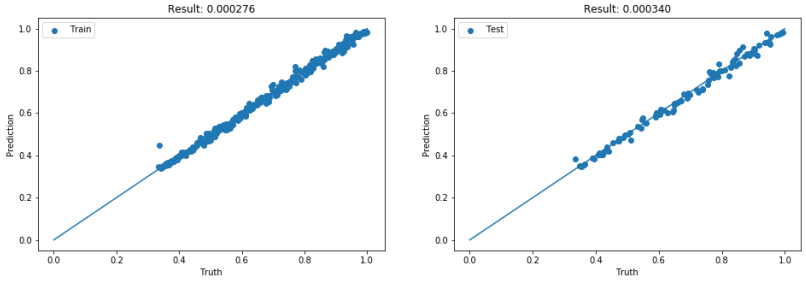


Figure 4.11: Results for PolyNet-avg for purity in the CV case with cutoff 3. **Left**: training set ; **Right**: test set

Cutoff 5

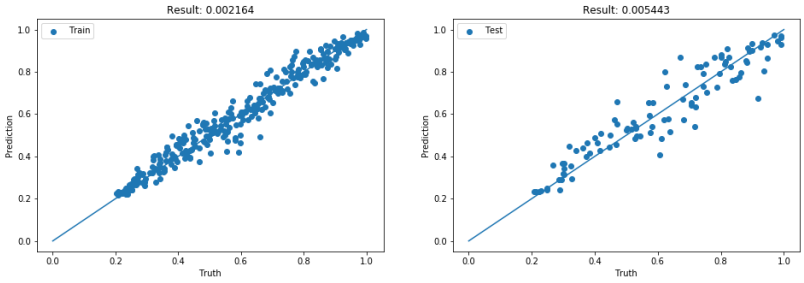


Figure 4.12: Results for PolyNet-avg for purity in the CV case with cutoff 5. **Left**: training set ; **Right**: test set

Chapter 5

Conclusion

We introduced three circuit architectures based on quantum and classical neural networks that can be used to compute polynomial properties of a state. For two of them, PolyNet with ancilla and PolyNet with correlations, we proved that they have the capacity to learn polynomial properties of qubit states, and conjecture that it is also the case for PolyNet with averages. In the CV case, we introduced a generalization of PolyNet with ancilla, the only architecture to generalize well for CV states. Except for this last architecture—that requires more computational resources than available during the writing of this thesis—we tested all the other circuits for both CV and qubit states on simulators. We showed that for one-qubit states, the three circuits are able to learn the purity almost perfectly and the entropy with some error, due to its non-polynomial character. For one-qumode states, the results degraded with the increase of the cutoff, which suggests that they would not generalize well to real CV states, as predicted by the theory.

Several possible tracks can be considered for future work. First, proving that PolyNet with averages is universal (Conjecture 1). Then, experimentally testing PolyNet with ancilla in the CV case, as well as the other architectures with a biggest number of qubits and other functionals.

Bibliography

[1] Richard P. Feynman. "Simulating physics with computers". In: *International Journal of Theoretical Physics* 21.6 (1982), pp. 467–488. DOI: 10.1007/BF02650179 (cit. on p. 1).

[2] John Preskill. "Quantum Computing in the NISQ era and beyond". In: *Quantum* 2 (2018), p. 79. DOI: 10.22331/q-2018-08-06-79 (cit. on p. 1).

[3] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. "A quantum approximate optimization algorithm". In: *arXiv:1411.4028* (2014). URL: https://arxiv.org/abs/1411.4028 (cit. on p. 1).

[4] Fernando G.S.L. Brandao and Krysta Svore. "Quantum Speed-ups for Semidefinite Programming". In: *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)* (2017). URL: https://arxiv.org/abs/1609.05537 (cit. on p. 1).

[5] Sam McArdle et al. "Quantum computational chemistry". In: *arXiv:1808.10402* (2018). URL: https://arxiv.org/abs/1808.10402 (cit. on p. 1).

[6] Yudong Cao et al. "Quantum Chemistry in the Age of Quantum Computing". In: *arXiv:1812.09976* (2018). URL: https://arxiv.org/abs/1812.09976 (cit. on pp. 1, 2).

[7] Chris Sparrow et al. "Simulating the vibrational quantum dynamics of molecules using photonics". In: *Nature* 557.7707 (2018), pp. 660–667. DOI: 10.1038/s41586-018-0152-9 (cit. on p. 1).

[8] Jacob Biamonte et al. "Quantum machine learning". In: *Nature* 549 (2017), 195 EP. DOI: 10.1038/nature23474 (cit. on p. 1).

[9] Peter Wittek. *Quantum Machine Learning: What Quantum Computing Means to Data Mining*. Academic Press, 2014 (cit. on p. 1).

[10] Maria Schuld and Francesco Petruccione. *Supervised Learning with Quantum Computers*. Springer International Publishing, 2018 (cit. on p. 1).

[11] Nathan Wiebe, Ashish Kapoor, and Krysta M Svore. "Quantum Perceptron Models". In: *30th Conference on Neural Information Processing Systems (NIPS 2016)* (2016). URL: https://arxiv.org/abs/1602.04799 (cit. on pp. 1, 3).

[12] Edward Farhi and Hartmut Neven. "Classification with Quantum Neural Networks on Near Term Processors". In: *arXiv:1802.06002* (2018). URL: https://arxiv.org/abs/1802.06002 (cit. on pp. 1, 3).

[13] Seth Lloyd and Christian Weedbrook. "Quantum Generative Adversarial Learning". In: *Phys. Rev. Lett.* 121 (4 2018), p. 040502. DOI: 10.1103/PhysRevLett.121.040502 (cit. on p. 1).

[14] Juan Miguel Arrazola et al. "Machine learning method for state preparation and gate synthesis on photonic quantum computers". In: *Quantum Science and Technology* 4.2 (2018), p. 024004. URL: https://arxiv.org/abs/1807.10781 (cit. on pp. 1, 36).

[15] Alberto Peruzzo et al. "A variational eigenvalue solver on a quantum processor". In: *Nature Communications volume 5, Article number: 4213* (2018). URL: https://arxiv.org/abs/1304.3061 (cit. on p. 2).

[16] Ryan LaRose et al. "Variational Quantum State Diagonalization". In: *arXiv:1810.10506* (2018). URL: https://arxiv.org/abs/1810.10506 (cit. on p. 2).

[17] Nathan Killoran et al. "Strawberry Fields: A Software Platform for Photonic Quantum Computing". In: *Quantum* 3 (2019), p. 129. DOI: 10.22331/q-2019-03-11-129 (cit. on pp. 2, 24–26, 45).

[18] Ville Bergholm et al. "PennyLane: Automatic differentiation of hybrid quantum-classical computations". In: *arXiv:1811.04968* (2018). URL: https://arxiv.org/abs/1811.04968 (cit. on pp. 2, 26, 45).

[19] Guillaume Verdon et al. "A Quantum Approximate Optimization Algorithm for continuous problems". In: *arXiv:1902.00409* (2019). URL: https://arxiv.org/abs/1902.00409 (cit. on p. 2).

[20] Patrick Rebentrost, Brajesh Gupt, and Thomas R. Bromley. "Photonic quantum algorithm for Monte Carlo integration". In: *arXiv:1809.02579* (2018). URL: https://arxiv.org/abs/1809.02579 (cit. on p. 2).

[21] Leonardo Banchi et al. "Molecular Docking with Gaussian Boson Sampling". In: *arXiv:1902.00462* (2019). URL: https://arxiv.org/abs/1902.00462 (cit. on p. 2).

[22] Paweł Horodecki and Artur Ekert. "Method for Direct Detection of Quantum Entanglement". In: *Phys. Rev. Lett.* 89 (12 2002), p. 127902. DOI: 10.1103/PhysRevLett.89.127902 (cit. on p. 3).

[23] J.P. Paz and A. Roncaglia. "A quantum gate array can be programmed to evaluate the expectation value of any operator". In: *Phys. Rev. Lett. 88, 217901* (2003). URL: https://arxiv.org/abs/quant-ph/0306143 (cit. on pp. 3, 30).

[24] Artur K. Ekert et al. "Direct estimations of linear and non-linear functionals of a quantum state". In: *Phys. Rev. Lett. 88, 217901* (2002). URL: https://arxiv.org/abs/quant-ph/0203016 (cit. on pp. 3, 29, 30).

[25] Todd Brun. "Estimating Polynomial Functions of a Quantum State". In: *AIP Conference Proceedings 734, 71* (2004). URL: https://arxiv.org/abs/quant-ph/0401067 (cit. on pp. 3, 30, 31).

[26] K. L. Pregnell. "Measuring Nonlinear Functionals of Quantum Harmonic Oscillator States". In: *Phys. Rev. Lett.* 96.6 (2006). DOI: 10.1103/physrevlett.96.060501 (cit. on pp. 3, 32).

[27] A. J. Daley et al. "Measuring Entanglement Growth in Quench Dynamics of Bosons in an Optical Lattice". In: *Phys. Rev. Lett.* 109 (2 2012), p. 020505. DOI: 10.1103/PhysRevLett.109.020505 (cit. on p. 3).

[28] Hyunseok Jeong et al. "Detecting the degree of macroscopic quantumness using an overlap measurement". In: *J. Opt. Soc. Am. B* 31.12 (2014), pp. 3057–3066. DOI: 10.1364/JOSAB.31.003057 (cit. on p. 3).

[29] Edward Farhi and Hartmut Neven. "Efficient Learning for Deep Quantum Neural Networks". In: *arXiv:1408.7005* (2018). URL: https://arxiv.org/abs/1408.7005 (cit. on pp. 3, 4).

[30] Giuseppe Carleo and Matthias Troyer. "Solving the quantum many-body problem with artificial neural networks". In: *Science* 355.6325 (2017), pp. 602–606. DOI: 10.1126/science.aag2302 (cit. on p. 3).

[31] M. Tiersch, E. J. Ganahl, and H. J. Briegel. "Adaptive quantum computation in changing environments using projective simulation". In: *Scientific Reports* 5 (2015), 12874 EP. DOI: 10.1038/srep12874 (cit. on p. 3).

[32] Giacomo Torlai et al. "Neural-network quantum state tomography". In: *Nature Physics* 14.5 (2018), pp. 447–450. DOI: 10.1038/s41567-018-0048-5 (cit. on p. 3).

[33] K. Mitarai et al. "Quantum circuit learning". In: *Phys. Rev. A* 98 (3 2018), p. 032309. DOI: 10.1103/PhysRevA.98.032309 (cit. on pp. 3, 4, 25).

[34] Lukasz Cincio et al. "Learning the quantum algorithm for state overlap". In: *New J. Phys. 20 113022* (2018). URL: https://arxiv.org/abs/1803.04114 (cit. on pp. 3, 31).

[35] Iris Cong, Soonwon Choi, and Mikhail D. Lukin. "Quantum Convolutional Neural Networks". In: *arXiv:1810.03787* (2018). URL: https://arxiv.org/abs/1810.03787 (cit. on pp. 3, 25, 26).

[36] Michael Broughton Guillaume Verdon Jason Pye. "A Universal Training Algorithm for Quantum Deep Learning". In: *arXiv:1806.09729* (2018). URL: https://arxiv.org/abs/1806.09729 (cit. on p. 4).

[37] Nathan Killoran et al. "Continuous-variable quantum neural networks". In: *arXiv:1806.06871* (2018). URL: https://arxiv.org/abs/1806.06871 (cit. on pp. 4, 26, 28).

[38] Christopher Gerry and Peter Knight. *Introductory Quantum Optics*. Cambridge University Press, 2004. DOI: 10.1017/CBO9780511791239 (cit. on p. 17).

[39] Daiqin Su, Casey R. Myers, and Krishna Kumar Sabapathy. "Conversion of Gaussian states to non-Gaussian states using photon number-resolving detectors". In: *arXiv:1902.02323* (2019). URL: https://arxiv.org/abs/1902.02323 (cit. on p. 24).

[40] P. B. M. Sousa and R. V. Ramos. "Universal Quantum Circuit for N-qubit Quantum Gate: A Programmable Quantum Gate". In: *Quantum Info. Comput.* 7.3 (2007), pp. 228–242. URL: http://dl.acm.org/citation.cfm?id=2011717.2011721 (cit. on pp. 26, 28).

[41] Panagiotis Kl. Barkoutsos et al. "Quantum algorithms for electronic structure calculations: Particle-hole Hamiltonian and optimized wave-function expansions". In: *Phys. Rev. A* 98 (2 2018), p. 022322. DOI: 10.1103/PhysRevA.98.022322 (cit. on p. 26).

[42] Maria Schuld et al. "Evaluating analytic gradients on quantum hardware". In: *arXiv:1811.11184* (2019). URL: https://arxiv.org/abs/1811.11184 (cit. on pp. 26, 27).

[43] Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. "Adam: A Method for Stochastic Optimization". In: *arXiv:1412.6980* (2014). URL: https://arxiv.org/abs/1412.6980 (cit. on p. 27).

[44] Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. "On the Convergence of Adam and Beyond". In: *arXiv:1904.09237* (2019). URL: https://arxiv.org/abs/1904.09237 (cit. on p. 27).

[45] Farrokh Vatan and Colin Williams. "Quantum-assisted quantum compiling". In: *Phys. Rev. A 69, 032315* (2018). URL: https://arxiv.org/abs/1807.00800 (cit. on pp. 27, 31).

[46] Farrokh Vatan and Colin Williams. "Optimal Quantum Circuits for General Two-Qubit Gates". In: *Phys. Rev. A 69, 032315* (2004). URL: https://arxiv.org/abs/quant-ph/0308006 (cit. on pp. 27, 28).

[47] Daniel Gottesman and Isaac Chuang. "Quantum Digital Signatures". In: *arXiv:quant-ph/0105032* (2001). URL: https://arxiv.org/abs/quant-ph/0105032 (cit. on p. 32).

[48] Harry Buhrman et al. "Quantum fingerprinting". In: *Phys. Rev. Lett. 87, 167902* (2001). URL: https://arxiv.org/abs/quant-ph/0102001 (cit. on p. 32).

[49] Xiao-Qi Zhou et al. "Adding control to arbitrary unknown quantum operations". In: *Nature Communications* 2 (2011), 413 EP. DOI: 10.1038/ncomms1392 (cit. on pp. 35, 36).

[50] Hoi-Kwan Lau and Martin B. Plenio. "Universal Quantum Computing with Arbitrary Continuous-Variable Encoding". In: *Phys. Rev. Lett.* 117 (10 2016), p. 100501. DOI: 10.1103/PhysRevLett.117.100501 (cit. on p. 35).

[51] Hoi-Kwan Lau et al. "Quantum Machine Learning over Infinite Dimensions". In: *Phys. Rev. Lett.* 118 (8 2017), p. 080501. DOI: 10.1103/PhysRevLett.118.080501 (cit. on pp. 35, 36).

[52] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010. DOI: 10.1017/CBO9780511976667 (cit. on p. 43).

[53] Robert S Smith, Michael J Curtis, and William J Zeng. "A Practical Quantum Instruction Set Architecture". In: *arXiv:1608.03355* (2016). URL: https://arxiv.org/abs/1608.03355 (cit. on p. 45).

[54] J.R. Johansson, P.D. Nation, and F. Nori. "QuTiP: An open-source Python framework for the dynamics of open quantum systems". In: *Computer Physics Communications* 183.8 (2012), pp. 1760–1772. DOI: 10.1016/j.cpc.2012.02.021 (cit. on p. 45).

[55] Martin Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/ (cit. on p. 45).

[56] Travis E Oliphant. *A guide to NumPy*. Vol. 1. Trelgol Publishing USA, 2006 (cit. on p. 46).

[57] Eric Jones, Travis Oliphant, Pearu Peterson, et al. *SciPy: Open source scientific tools for Python*. [Online; accessed <today>]. 2001. URL: http://www.scipy.org/ (cit. on p. 46).

[58] J. D. Hunter. "Matplotlib: A 2D graphics environment". In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55 (cit. on p. 46).