FACULTY OF ENGINEERING AND ARCHITECTURE

Photonic quantum reservoir computing

Robbe De Prins
Student number: 01503121

Supervisors: Prof. Peter Bienstman, Prof. Guy Van der Sande (Vrije Universiteit Brussel)

Master's dissertation submitted in order to obtain the academic degree of
Master of Science in Engineering Physics

Academic year 2020-2021

GHENT UNIVERSITY

Preface

Throughout the writing of this thesis I have received a great deal of support and assistance.

First and foremost, I would like to thank my supervisors, Professor Peter Bienstman and Professor Guy Van der Sande, whose expertise was invaluable. Your insightful feedback pushed me to sharpen my thinking and elevated my work and knowledge to a higher level.

I would like to thank my counsellors Emmanuel Gooskens, Fabian Böhm and Floris Laporte, for their remarks, interest, valuable guidance and insights.

Thank you to Lander Burgelman, Jasper de Witte and Tom Vanackere, for listening to me when I needed a soundboard, for your support and interest.

I would like to acknowledge my colleagues from my internships at imec IDLab and imec-MICT–UGent. You have introduced me to the wonderful world of machine learning. Thank you for giving me the chance to learn the basic tools I needed in order to work on this thesis.

In addition, I would like to thank my parents for their wise counsel and sympathetic ear. Finally, I could not have completed this thesis without the support, understanding and patience of my girlfriend.

Robbe De Prins
14 June 2021

Photonic quantum reservoir computing

Written by: Robbe De Prins
Supervisors: Prof. Peter Bienstman, Prof. Guy Van der Sande
Counsellors: Emmanuel Gooskens, Fabiam Böhm

Master's dissertation submitted in order to obtain the academic degree of
Master of Science in Engineering Physics

Faculty of Engineering and Architecture
Academic year of 2020-2021

Abstract

Machine learning aims to derive patterns from data in order to apply what it has learned to previously unseen data. Examples of such tasks are speech and image recognition. As its number of application grows, so does its research. In the last years, new machine learning proposals have arisen that try to improve existing algorithms by performing them using quantum hardware rather than classical hardware. This combines machine learning with the field of quantum computing. The intersection of these two fields is called quantum machine learning. One machine learning strategy that is naturally suited for this purpose is reservoir computing, in which a part of the computational cost of solving a temporal task can be outsourced to a nonlinear, dynamical system, such as a quantum system. This approach only requires a simple training procedure and low-quality components, making it suitable for existing noisy intermediate-scale quantum computers. In this thesis, we present a quantum optical model to perform reservoir computing by making use of an existing quantum optical neural network. The construction of the system will first be discussed, after which we perform an analysis of its performance and discuss its possible extensions and improvements.

Keywords

Quantum machine learning, quantum optics, reservoir computing, boson sampling

Photonic quantum reservoir computing

Robbe De Prins

Supervisor(s): Prof. Peter Bienstman, Prof. Guy Van der Sande

Counsellor(s): Emmanuel Gooskens, Fabian Böhm

Abstract—**Machine learning and quantum computing are rapidly growing fields. In recent years, the common ground between these two fields has attracted a lot of attention, possibly enabling existing machine learning algorithms to be performed more efficiently than is possible on classical computers. One proposal to do so is quantum reservoir computing, which exploits the quantum nature of physical systems and combines it with a simple training strategy in order to solve temporal tasks such as speech recognition. In this thesis, we present a new quantum optical reservoir computing model, based on an existing quantum optical neural network. We discuss the construction of this model, show that it can solve a simple task and study how its performance varies for different values of the system parameters.**

Keywords—**Quantum machine learning, quantum optics, reservoir computing, boson sampling**

I. Introduction

Machine learning (ML) is taking the world by storm, showing superior data processing capabilities. A promising branch of this field is neuromorphic computing, of which neural networks (NNs) are one of the most well-known examples. NNs consist of several parameterized layers of so-called nodes that can be used to process information. By optimizing its parameters, the network can learn to solve a particular task. In the past, this has led to state-of-the-art performance of tasks such as image recognition [1]. Research into neuromorphic information processing techniques is fueled by the desire for techniques with superior computational power and energy efficiency to the human brain.

The search for such efficient systems leads us to unconventional computing schemes that go beyond the Von Neumann architecture [2], a very general and well-known paradigm characterised by the splitting of memory and processing units. In the brain however, all of these units work closely together, which suggests that their co-location is a gateway to high-efficiency techniques.

An example of such a computing scheme is reservoir computing (RC), which has its origins in the development of recurrent neural networks (RNNs). RNNs are NNs to which recurrent connections are added. These loops grant the system memory that can be used to process temporal data. In contrast to the normal training procedure of an RNN (which is similar to that of an NN), it turns out that these systems can also engage in useful forms of information processing without optimizing their internal parameters. Due to their inherent memory and rich dynamics, a random initialisation of such a system can perform a mapping from its input data to a higher dimensional space. After doing so, it suffices to optimize a simple linear readout layer in order to make the system perform certain tasks [3].

This means that we can outsource part of our time-dependent signal processing to a system that does not require any optimization and we only need to perform a simple linear training procedure. In ML, such a mapping is called a temporal feature expansion and the system that performs it in RC is called a reservoir. However, we are not limited to the group of RNNs in our search for suitable reservoirs. On the contrary, as long as the dynamics of a system are rich enough for the task at hand and the system has adequate memory, that system is a possible candidate for this computing scheme [4]. One type of system that has already proven to be especially successful at RC, are classical photonic systems [5].

However, the range of reservoirs is not restricted to classical systems either. Instead, we can utilise the rich dynamics of quantum physics to boost the efficiency of these computing schemes, leading us to the field of quantum reservoir computing (QRC). Quantum systems generally possess a large number of degrees of freedom that can be exploited for RC. However, simulating quantum systems on classical computers rapidly becomes challenging. Moreover, it is widely believed [6] that certain tasks, for example the factorisation of integers [7], are intractable for classical computers, while they can be solved more efficiently on hardware that is inherently based on the laws of quantum physics, leading to so-called quantum supremacy [8].

An example of a task that has already been used to demonstrate quantum supremacy, is boson sampling [9]. Boson sampling is the act of sampling indistinguishable photons, using single-photon detectors, after they have travelled through a linear interferometer. This process is popular because its hardware implementation requires less resources than other proposals to demonstrate quantum supremacy.

Building on the idea of boson sampling, a quantum optical neural network (QONN) was introduced in Ref. [10]. Here, photons are fed through a number of parameterized interferometers with Kerr nonlinearities added in between. By optimizing the phase shifters of the interferometers, this network can be trained to perform (non-temporal) tasks.

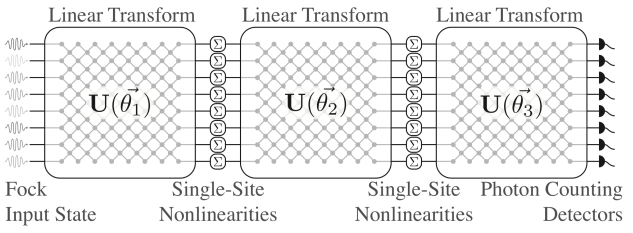


Fig. 1: QONN layout [10]. Inputs are Fock states and the linear transforms are linear interferometers. The single-site nonlinearities are given by Kerr-type interactions, applying a constant phase shift for each additional photon present. Readout is performed by photon-number-resolving detectors at each output mode.

In this work, we introduce a new QRC model that is based on such a QONN. We will discuss the structure of this model and show how its simulation is performed. Afterwards, its behaviour will be studied.

II. Construction of a QRC model

A. Set-up and operating procedure

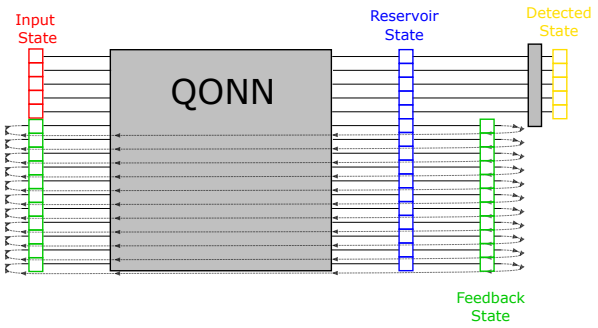


Fig. 2: Layout of the quantum reservoir that is introduced in this thesis, comprising of input modes (red) and a randomly initialised QONN (grey square), leading to a reservoir state (blue). The detector (grey rectangle) is a combination of single-photon detectors on each mode. It measures a certain detection state (yellow). The feedback state (green) is looped back to the input of the QONN.

A schematic representation of our quantum reservoir is given in Figure 2. The QONN in this set-up is randomly initialised. Generally, there are m_A modes that lead to the detector. Recurrent connections are added to the other m_B modes of the reservoir. All modes are numbered from top (mode 1) to bottom (mode m_A+m_B). The recurrent connections grant the reservoir memory, such that it can learn to solve temporal tasks, i.e. to prepare a desired output, depending on multiple past inputs.

In order to do so, we send photons into the system at a constant rate. These photons encode input data from the train set. There are many ways to perform such an encoding, but for now we assume that $m_A > 2$ and that we only want to encode classical bits, i.e. 0 and 1. In order to encode a 0-bit, we choose to send n_{in} photons into the first input mode. Conversely, for a 1-bit, we choose to send n_{in} photons into the second input mode.

After the input state $|\psi_{\text{in}}\rangle$ is prepared, it is combined with a certain feedback state $|\psi_{\text{fb}}\rangle$. The feedback state did not leave the system in the previous iteration and contains information about multiple previous inputs. As $|\psi_{\text{in}}\rangle$ and $|\psi_{\text{fb}}\rangle$ are defined on different modes (labeled with A and B respectively), the combined state is equal to $|\psi_{\text{in}}\rangle \otimes |\psi_{\text{fb}}\rangle$. This combined state is fed into the QONN where its photons interfere. Consequently, the resulting reservoir state $|\psi_{\text{res}}\rangle$ also contains information about previous inputs. A part of this information is retained in a new feedback state, while another part is detected.

Although $|\psi_{\text{res}}\rangle$ generally is a superposition of Fock states, only one of its Fock states can be detected. As is common in quantum computing, this stochastic process can be repeated multiple times in order to estimate the expectation values of all detections [11]. This requires us to make multiple copies of the reservoir in which we send the same input data. In a simulation, these copies can be created in parallel, but in experiments, this is done sequentially using a single reservoir.

As the resulting expectation values contain information about multiple past inputs, the reservoir performs a temporal feature expansion. Furthermore, these temporal features need to be mapped onto the desired output of the task. This can be performed by a linear regressor [12]. After training, a similar feature expansion is carried out for the test data in order to estimate the performance of the system.

B. Temporal tasks

As noted in the previous section, at the moment we only want to encode classical bits into the input state. The reason for this is that we focus on performing a simple classical task: the temporal XOR task. The purpose of this task is to combine subsequent input bits according to XOR logic, as is shown in Table I.

TABLE I: An example of the desired output for the temporal XOR task. The input bits of iterations $k-1$ and k are combined, leading to a 0-bit if both inputs are equal. Otherwise, the output is a 1-bit.

Iteration	0	1	2	3	4	5	6	7
Input bit sequence	1	1	1	0	0	1	0	1
Output bit sequence	1	0	0	1	0	1	1	1

C. Simulation of a single reservoir

C.1 QONN

Having introduced the general structure of our model and the task that we want to tackle, let us consider how we can simulate the QONN of a specific reservoir in the ensemble. As we know that the different interferometers in the N layers of the QONN are composed of beamsplitters (BS) and phase shifters (PS), they can be characterised by unitary matrices $\boldsymbol{U}(\boldsymbol{\theta}_i)$ ($i \in \{1, ..., N\}$), that are constructed by combining individual transformation matrices of its components:

$$\boldsymbol{BS} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \quad (1) \qquad \boldsymbol{PS} = \begin{pmatrix} e^{i\theta_1} & 0 \\ 0 & e^{i\theta_2} \end{pmatrix} \quad (2)$$

Note that $\boldsymbol{U}$ only operates on a basis of single-photon Fock states. However, generally we are working with multi-photon Fock states and we want to derive the unitary matrix $\boldsymbol{W}$ that transforms these states. For each $\boldsymbol{U}$, there exists a corresponding $\boldsymbol{W}$. Both matrices are connected through a homomorphism Φ, where $\boldsymbol{W} = \Phi[\boldsymbol{U}]$. Given a certain $\boldsymbol{U}$, the calculation of $\boldsymbol{W}$ boils down to simulating the boson sampling process that we introduced earlier. This calculation can be performed by the so-called Ryser's algorithm [13]. As mentioned before, boson sampling, and hence the execution of Ryser's algorithm, is a hard problem for classical computers that can be solved naturally on quantum photonic hardware.

In between the resulting $\boldsymbol{U}(\boldsymbol{\theta}_i)$, nonlinear Kerr interactions

are applied to each mode. These interactions are described by:

$$\boldsymbol{\Sigma}(\phi) = \sum_{n=0}^{\infty} e^{in(n-1)\phi/2} |n\rangle\langle n| \tag{3}$$

where n is the number of photons in the mode that we consider and ϕ quantifies the strength of the Kerr nonlinearity. Finally, this leads to the following transformation matrix of the QONN:

$$\begin{aligned} \boldsymbol{S}(\boldsymbol{\Theta}) =& \Phi\left[\boldsymbol{U}\left(\boldsymbol{\theta}_N\right)\right] \prod_{i=1}^{N-1} \boldsymbol{\Sigma}(\phi)\ \Phi\left[\boldsymbol{U}\left(\boldsymbol{\theta}_{N-i}\right)\right] \\ =& \boldsymbol{W}\left(\boldsymbol{\theta}_N\right) \prod_{i=1}^{N-1} \boldsymbol{\Sigma}(\phi)\ \boldsymbol{W}\left(\boldsymbol{\theta}_{N-i}\right) \end{aligned} \tag{4}$$

where we have dropped the dependence on the Kerr interaction strength ϕ for ease of notation. Also note that the nonlinear layers, which result from applying Equation (3) to all modes of the system, are also written as $\boldsymbol{\Sigma}(\phi)$ to simplify the notation. The size of $\boldsymbol{S}$ is defined by the Fock basis that it operates on, and consequently by the number of photons that are present in the QONN. Each iteration, this photon number is given by:

$$n_{\text{res}}(k) = n_{\text{res}}(k-1) - n_{\text{out}}(k-1) + n_{\text{in}} \tag{5}$$

where $n_{\text{res}}(k)$ and $n_{\text{out}}(k)$ respectively represent the number of photons in the QONN and in the detected state at iteration k. Due to the probabilistic nature of the detectors, n_{res} can fluctuate between different iterations. Also note that n_{res} can differ between reservoir copies in the ensemble.

As a result of these photon fluctuations, the size of the Hilbert space on which a reservoir operation is performed, also fluctuates. This behaviour is very unusual in comparison with QRC approaches on other platforms. In some cases, this makes it difficult to transfer results and conclusions from literature.

As we currently do not consider photon loss, Equation (5) provides a value for n_{res} at each iteration. Hence, we can calculate $\Phi[\,\boldsymbol{U}(\boldsymbol{\theta}_i)\,]$ using a Fock basis that consists of n_{res} photons, spread over m modes. As the QONN is kept fixed, we can store and re-use the resulting $\boldsymbol{S}(\boldsymbol{\Theta})$ for different values of n_{res}. This shifts the computational load at runtime from executing Ryser's algorithm to matrix multiplications.

C.2 Partial trace and detection

Furthermore, let us describe how we can simulate the separation of the A and B modes at the output of the QONN. Normally, such a process is described by a partial trace of a density matrix, which generally leads to a mixed state in the feedback modes [14]. In this case, however, it can be shown that the process simplifies because of the detection that is performed after separating the modes. The resulting feedback state is again a pure state that can be represented by a state vector. This state is calculated by first simulating a measurement, where the outcome $|\mu\rangle_A$ occurs with probability $P(\mu)$. By summing over all terms in the reservoir superposition that can be written as $a_\sigma^{(\mu)} |\mu\rangle_A \otimes |\sigma\rangle_B$, it can be shown that $P(\mu) = \sum_\sigma \left|a_\sigma^{(\mu)}\right|^2$ and that the resulting feedback state is given by:

$$|\psi_{\text{fb}}^{(\mu)}\rangle = \frac{1}{\sqrt{P(\mu)}} \sum_\sigma a_\sigma^{(\mu)} |\sigma\rangle \tag{6}$$

D. *Feature convergence*

Knowing how we can simulate each reservoir in the ensemble, we shift our focus to the input features of the linear regressor, further simply called the features. These features are approximations of the expectation values of the detections, calculated at the end of each iteration by dividing the number of times a detection occurred by the ensemble size. The regressor interprets the feature values of different iterations as different samples that need to be mapped to the correct output.

Figure 3 shows the convergence of the feature values that were obtained at a certain iteration as a function of the ensemble size. In this figure, the expectation values are repeatedly estimated, each time using a larger ensemble size. Eventually, in this example, the values for an ensemble size of 600 are sent to the regressor.

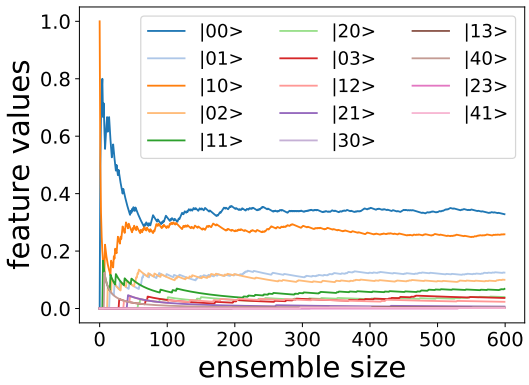


Fig. 3: Feature values as a function of the ensemble size, at iteration 750 of the training procedure. At a certain ensemble size, each colored line shows an approximation of the expectation value of a certain detection (indicated by the legend), acquired using an ensemble of that size.

E. *Fading memory*

The resulting feature values contain information about multiple past inputs. However, the memory of a reservoir in an RC approach should be limited to a number of iterations. This requirement is called the fading memory principle. It states that the influence of input information on the current reservoir state should become smaller the longer ago it was injected. If so, the reservoir asymptotically forgets its initial conditions and only information that was acquired during the last iterations shapes the current reservoir state.

However, as was already made clear in the introduction, the range of systems that qualify for an RC approach is wide. Indeed, in reality, this additional fading memory constraint proves not to be very restrictive [4].

Let us now also investigate the fading memory of our system by generalizing the definition of the temporal XOR task, increasing the memory time that it requires. We do so by introducing a certain delay D between its input bits. Instead of combining the input bits of the iterations $k-1$ and k (as was shown in Table I), the objective is now to combine the input bits of iterations $k-D$ and k.

We perform this task for different values of D and average the resulting test bit error rate (BER) for 10 different reservoir initialisations. Doing so, we find the following figure:

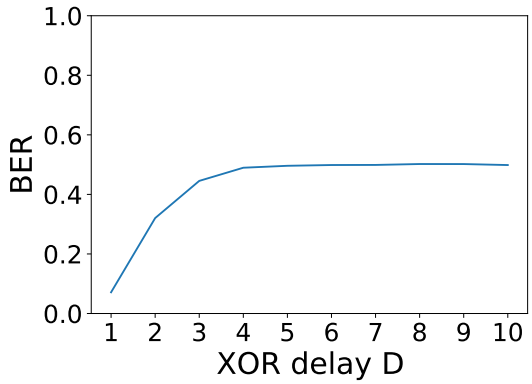


Fig. 4: Mean test BER as a function of D. For each value of D, the average is calculated over 10 reservoir initialisations.

This shows that the injected information is indeed forgotten over a number of iterations. Let us denote this number by M. For the specific model that was used for this simulation, M is equal to 4. However, we note that tuning the fading memory is an important practice in RC [4], as the memory should be adapted to the task at hand. Therefore, the study of M in function of the system parameters is part of the future objectives of this thesis.

III. Results and discussion

A. *Study of the system parameters*

Now that our model is constructed, we further study its behaviour and evaluate its performance. As we do not optimize any hyperparameters at our current research stage, we only distinguish between a train and test set, without the introduction of a validation set. For all further simulations, the test set size is equal to 10 000, allowing us to safely present the resulting test BER up to an accuracy of 1% [15].

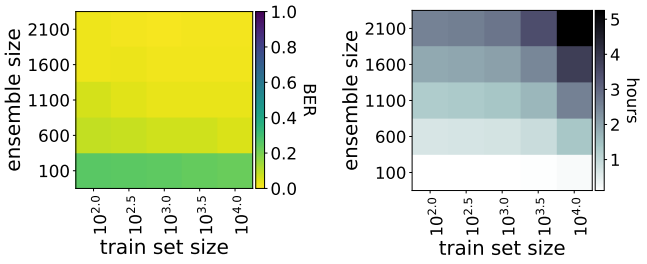
A.1 Train set size and ensemble size

Figure 5 shows simulation results for different values of the train set size and the ensemble size. For each combination of these sizes, 50 different reservoir initialisations are performed, after which their resulting BER and simulation time are averaged.

We would expect that both a larger train set and a larger ensemble would enhance the performance of the system. The reason for this is that more data is generally better for training machine learning models [12], while a bigger ensemble results in more accurate estimations of the expectation values.

In Figure 5a we notice that the dependency of the BER on the ensemble size is more pronounced than the dependency on the train set size. Note however that this observation is task-dependent and that more data could be required to perform more challenging tasks.

The increase in simulation time of Figure 5b results from the fact that the total number of reservoir iterations scales linearly with the the train set size and the ensemble size and that the test set size is fixed to a value of 10 000.



(a) mean BER (b) mean simulation time

Fig. 5: Visualisation of the mean test BER and mean simulation time for different train set size and ensemble size. For each combination of the train set size and the ensemble size, 50 different reservoir initialisations are performed.

A.2 Kerr interaction strength ϕ

Let us now investigate the influence of ϕ (as defined in Equation (3)). For each value of ϕ that we consider, we perform 100 reservoir initialisations with an ensemble size of 600. For each initialisation, we store the maximum value of n_{res}, i.e. the maximum over all reservoir copies and over all training iterations. The resulting values are shown in Figure 6. For each value of ϕ, reservoir initialisations that lead to the same value of $\max(n_{\text{res}})$ are grouped together in one dot. The area of the dots is proportional to the number of initialisations that it represents. Figure 7 depicts the test BER for those same reservoir initialisations, where only positive values of ϕ are retained. For each BER value ϵ on the x-axis, the fraction of initialisations is plotted that lead to a test BER lower than ϵ.

We notice that the removal of the nonlinear layers from the QONN ($\phi = 0$), both influences the photon number fluctuations and the performance of the system. More specifically, at $\phi = 0$, $\max(n_{\text{res}})$ and the test BER are more initialisation-dependent. Although the underlying reason for this is unclear at the moment, we remark that this behaviour is unwanted as we want to initialise the QONN randomly. Moreover, note that higher values of $\max(n_{\text{res}})$ correspond with long simulation times, as we need to perform Ryser's algorithm.

What is also unclear at the moment, is whether there is a direct link between the photon number fluctuations and the performance of the system. If such a link were to exist, it could give us more insight in the behaviour of our system and possibly show us ways to boost the performance. Consequently, a further study of this topic is part of the future objectives of this thesis.

Also note that the initialisation dependence of the error is still rather high for ϕ values different from zero. In classical RC systems, after fully optimizing the system (not the case yet in this thesis), typically 95% to 99% of all reservoir initialisations lead to a near perfect test BER ($\approx 0\%$) [16]. In Figure 7, the initialisation dependence is noticeably stronger, resulting at best (for $\phi = 0.94$) in an ϵ value of 16% if we consider the best performing 95% of all initialisations. Increasing the ensemble size (not shown here) lowers this initialisation dependence, but further research is needed to show whether this problem can be

completely resolved by further optimization of the system.

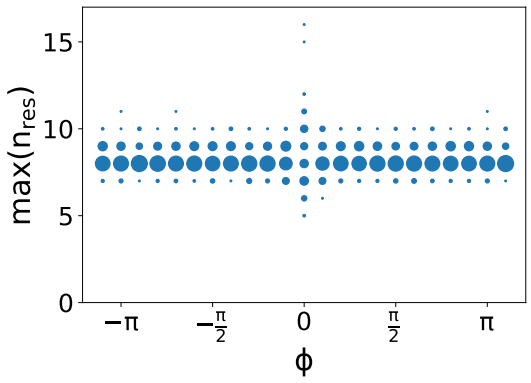


Fig. 6: The maximum of n_{res} over all reservoir copies and over all training iterations, for different values of ϕ. For each value of ϕ, 100 reservoir initialisations are performed and the area of the dots is proportional to the number of initialisations that lead to that value of max(n_{res}).

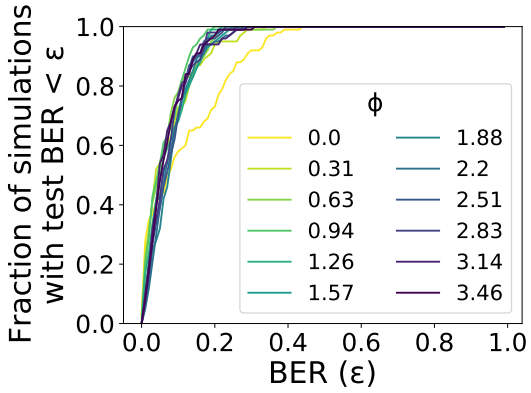


Fig. 7: Cumulative distribution function (CDF) of the fraction of simulations (with different reservoir initialisation) below a certain BER value. For each BER value ϵ on the x-axis, the fraction of simulations is plotted that leads to a test BER lower than ϵ.

B. Collapse as part of the dynamics

Until now, we have simulated a measurement at the end of every iteration. In this section, however, we will consider a modified version of our model where we remove all detections, except for the last one.

As is explained in Ref. [17], a hardware implementation of such a model can still be used to extract information from the reservoir at all iterations. In order to do so, the QRC procedure needs to be restarted repeatedly, each time running the system for a longer number of iterations, say K iterations, before making the final detection and stopping the system. Note that since we do always perform a measurement in the end, this new procedure still requires us to create an ensemble of reservoir copies. If we create these reservoir copies sequentially, this means that for each of the possible values of K and for each reservoir copy that we want to create, the first K inputs of the data set are sent into the reservoir.

It goes without saying that this is a lengthy procedure which limits the capacity of the system. Moreover, for the model depicted in Figure 2, the removal of the measurements means that the detection modes still leave the system, but are not detected. Consequently, the decoherence of the photons in these modes should be prevented until the end of each experiment. Because of these reasons, we do not want to perform such a procedure using a *hardware implemented* version of our model. However, as will become clear in the remainder of this section, the removal of all intermediate detections will lead us to a more efficient implementation of our current *simulation*.

In such a simulation, in order to describe the separation of the A and B modes, we need to shift from the state vector representation to the density operator formalism [14] and calculate a partial trace. As this separation process generally is not followed by a detection anymore, the expression for the feedback state no longer simplifies to the pure state. Instead, we get a mixed state.

It can be proven that this mixed state describes a statistical ensemble of all the possible 'collapsed' states that could occur in the approach with intermediate detections (described by Equation (6)). In other words, by making an intermediate detection, we follow a specific branch from the probability tree that is defined by the state in front of the detector. Without these intermediate measurements, all the possible branches are kept into account until the last iteration. Finally, it turns out that both approaches, with and without intermediate detections, lead to the same expectation values. Since we use these expectation values as input for the linear regressor, both approaches lead to equally performing systems.

However, a simulation without intermediate measurements turns out to be more efficient. As we use density matrices to describe the mixed states, we no longer have to simulate different realisations of our system. Instead we can simulate a single reservoir and calculate the expectation values from its reservoir state $\boldsymbol{\rho}_{\text{res}}$ at the final iteration. Remember that the creation of reservoir copies would however be needed in a hardware implementation of the system. Moreover, during simulations, we can also copy our density matrices, which is forbidden in real life by the no-cloning theorem [18]. Consequently, we can iterate over the complete data set, without restarting the system and without intermediate measurements, while constantly making a copy of $\boldsymbol{\rho}_{\text{res}}$. Afterwards, we can calculate the desired expectation values from these copies.

B.1 Photon number threshold

In contrast to a simulation with intermediate detections, where $n_{\text{res}}(k)$ was given by Equation (5), in this new simulation method, $\boldsymbol{\rho}_{\text{res}}$ no longer contains a specific number of photons. Instead, we need to perform Ryser's algorithm using a basis of Fock states that contain up to n_{thr} photons, spread over m modes. Here, n_{thr} is a chosen threshold value.

Although the use of such a threshold value is common when simulating a quantum computing procedure with Fock states, the fact that it needs to be chosen becomes more problematic in the context of temporal training procedures. The reason for this is that, in theory, the possibility always exist that no photons

are detected when performing intermediate detections. In an approach without intermediate detections, this 'branch' is hence also accounted for by the mixed state, leading to a density matrix that, in theory, needs to be described using a threshold value that rapidly increases with the number of iterations.

As Ryser's algorithm scales exponentially with the photon number, this would lead to unfeasible simulation times. Although we expect that a finite threshold value could also lead to accurate results, it is not obvious to choose such a value that combines these accurate results with fast simulations. In order to do so, further research should be carried out, for which a comparison with the simulation with intermediate detections would come in handy.

IV. Future outlook

Following this last discussion, it would be worthwhile to make a more in-depth study about the photon number fluctuations. As discussed in Section III-A.2, this could also lead to new insights in the behaviour of the system and possibly lead to new methods to shape the photon number fluctuations, enabling us to boost the performance. In line with this study, future simulations could also be made to account for optical loss and other sorts of decoherence. For this purpose, we could make use of existing decoherence models [19].

A different objective is the study of how possible encodings of the input data could lead to better performance. These encodings are known to heavily impact the results of QML models [20]. For this purpose, a starting point could be to include an extra *optimizable* QONN on the first m_A modes, in front of the current QONN.

An issue that was not addressed up to now, is that it is difficult to generate Fock states without deterministic single-photon sources. For this reason, Gaussian states are often used as alternative input states to perform boson sampling. The resulting 'Gaussian boson sampling' process falls within the same complexity class as regular boson sampling. As the required set-up is however more practical, it could lead to superior hardware implementations.

Finally, instead of only considering a simple classical tasks, the Information Processing Capacity (IPC) [21] could serve as a better measure to quantify the potential of our system to exploit its high-dimensional Hilbert space. Moreover, we could also consider quantum tasks and perform a similar study using a quantum capacity measure [22].

V. Conclusions

In this paper, we introduced a new photonic QRC method and provided a proof-of-principle using a simple classical task. The performance of this model appeared to improve by using a larger ensemble. A similar improvement is also expected to occur as a function of the train set size, for more challenging tasks. A study of the Kerr interaction strength ϕ additionally showed that this parameter influences both the photon number fluctuations and the performance of the system, increasing the initialisation-dependence at $\phi = 0$. Further research is required to reveal the connections between these observations and to check whether the overall initialisation-dependence can be lowered to the same level as classical RC systems. Finally, we proposed a more efficient simulation method for our system and stressed some of the technicalities that need to be taken into account in order to implement such a simulation.

References

[1] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[2] J. Von Neumann, "First draft of a report on the edvac," *IEEE Annals of the History of Computing*, vol. 15, no. 4, pp. 27–75, 1993.

[3] W. Maass, T. Natschläger, and H. Markram, "Real-Time Computing Without Stable States: A New Framework for Neural Computation Based on Perturbations," *Neural Computation*, vol. 14, no. 11, pp. 2531–2560, 11 2002. [Online]. Available: https://doi.org/10.1162/089976602760407955

[4] G. Tanaka, T. Yamane, J. B. Héroux, R. Nakane, N. Kanazawa, S. Takeda, H. Numata, D. Nakano, and A. Hirose, "Recent advances in physical reservoir computing: A review," *Neural Networks*, vol. 115, pp. 100–123, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608019300784

[5] D. Brunner, M. C. Soriano, and G. Van der Sande, *Photonic Reservoir Computing: Optical Recurrent Neural Networks*. Walter de Gruyter GmbH & Co KG, 2019.

[6] S. Cook, "The p versus np problem," *The millennium prize problems*, pp. 87–104, 2006.

[7] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM review*, vol. 41, no. 2, pp. 303–332, 1999.

[8] A. W. Harrow and A. Montanaro, "Quantum computational supremacy," *Nature*, vol. 549, no. 7671, pp. 203–209, 2017.

[9] S. Aaronson and A. Arkhipov, "The computational complexity of linear optics," *Proceedings of the Annual ACM Symposium on Theory of Computing*, no. 0844626, pp. 333–342, 2011.

[10] G. R. Steinbrecher, J. P. Olson, D. Englund, and J. Carolan, "Quantum optical neural networks," *npj Quantum Information*, vol. 5, no. 1, pp. 1–9, 2019. [Online]. Available: http://dx.doi.org/10.1038/s41534-019-0174-7

[11] P. Mujal, R. Martínez-Peña, J. Nokkala, J. García-Beni, G. L. Giorgi, M. C. Soriano, and R. Zambrini, "Opportunities in quantum reservoir computing and extreme learning machines," *arXiv preprint arXiv:2102.11831*, 2021.

[12] E. Alpaydin, *Introduction to machine learning*. MIT press, 2020.

[13] H. J. Ryser, "Combinatorial mathematics. the carus mathematical monographs, no. 14. published by the mathematical association of america; distributed by john wiley and sons," *Inc., New York*, 1963.

[14] M. A. Nielsen and I. Chuang, "Quantum computation and quantum information," 2002.

[15] M. Jeruchim, "Techniques for estimating the bit error rate in the simulation of digital communication systems," *IEEE Journal on selected areas in communications*, vol. 2, no. 1, pp. 153–170, 1984.

[16] E. Gooskens, F. Laporte, C. Ma, S. Sackesyn, J. Dambre, and P. Bienstman, "The wavelength dimension in waveguide-based photonic reservoir computing," *Optics Express*, vol. 1, no. 1, p. 10, 2021.

[17] J. Chen, H. I. Nurdin, and N. Yamamoto, "Temporal information processing on noisy quantum computers," *Physical Review Applied*, vol. 14, no. 2, p. 1, 2020. [Online]. Available: https://doi.org/10.1103/PhysRevApplied.14.024065

[18] G. Lindblad, "A general no-cloning theorem," *Letters in Mathematical Physics*, vol. 47, no. 2, pp. 189–196, 1999.

[19] M. Schlosshauer, "Quantum decoherence," *Physics Reports*, vol. 831, pp. 1–57, 2019.

[20] J. Nokkala, R. Martínez-Peña, G. L. Giorgi, V. Parigi, M. C. Soriano, and R. Zambrini, "Gaussian states of continuous-variable quantum systems provide universal and versatile reservoir computing," *Communications Physics*, vol. 4, no. 1, pp. 1–11, 2021. [Online]. Available: http://dx.doi.org/10.1038/s42005-021-00556-w

[21] J. Dambre, D. Verstraeten, B. Schrauwen, and S. Massar, "Information processing capacity of dynamical systems," *Scientific reports*, vol. 2, no. 1, pp. 1–7, 2012.

[22] L. G. Wright and P. L. McMahon, "The capacity of quantum neural networks," *Optics InfoBase Conference Papers*, vol. Part F181-CLEO-AT 2020, pp. 1–12, 2020.

Contents

Chapter 1

Introduction

Machine learning (ML) [1][2] is a subfield of computer science where a system is created that learns to perform a certain task by repeatedly improving itself, making use of relevant data instead of being programmed explicitly. These days, ML is taking the world by storm, showing superior data processing capabilities. Consequently, there is a constant search to improve these existing methods.

In recent years, several ideas have emerged that combine ML with another promising information-processing field: quantum computing (QC) [3][4]. QC is a subfield of quantum information science where collective properties of quantum states, such as superposition and entanglement, are exploited to perform a certain computation. QC systems have been demonstrated to show great potential and it is widely believed that these systems are even able to outperform classical systems [5], leading to so-called quantum supremacy [6]. A typical example of a task that is believed to be intractable for classical computers, but not for hardware that is inherently based on the laws of quantum physics, is the factorisation of integers, for which a well-known quantum algorithm was developed by Peter Shor [7].

This leads us to the field of quantum machine learning (QML) [8], which is schematically represented by Figure 1.1. The purpose of QML is to boost the efficiency of existing machine learning algorithms, both in terms of computational power and power consumption, by exploiting the potential of quantum systems.

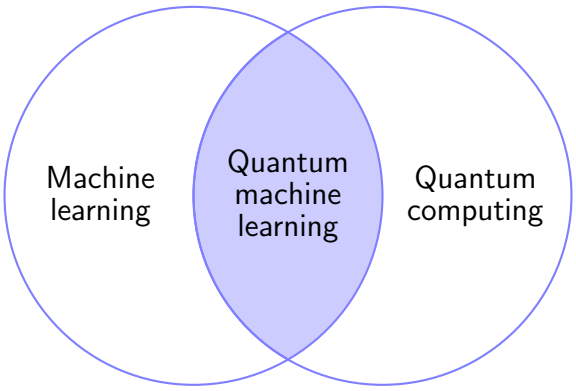


Figure 1.1: Schematic positioning of quantum machine learning with respect to its related technologic domains

A promising branch of ML that could be used for this purpose is neuromorphic computing, of which neural networks (NNs) [9] are one of the most well-known examples. NNs consist of several parameterized layers of so-called nodes that can be used to process information. More specifically, by optimizing its parameters, the network can learn to solve a particular task. In the past, this has led to state-of-the-art applications for tasks such as image recognition [10]. The research into neuromorphic information processing techniques is fuelled by the desire for computing systems with superior computational power and energy efficiency to the human brain.

The search for such systems leads us to unconventional computing schemes that go beyond the Von Neumann architecture [11]. This is a very general and well-known paradigm characterised by the splitting of memory and processing units. In the brain, all of these units work closely together, which suggests that their co-location is a gateway to highly efficiency techniques.

An example of such a computing scheme is reservoir computing (RC) [12][13], which has its origins in the development of recurrent neural networks (RNNs) [14]. As the name suggests, RNNs with recurrent connections. These loops grant the system memory, which can be used to process temporal data. In contrast to the normal training procedure of an RNN (which is similar to that of an NN), it turns out that these systems can also engage in useful forms of information processing without optimizing their parameters. Due to their inherent memory and rich dynamics, a random initialisation of such a system can perform a mapping from its input data to a higher dimensional space. After this mapping, it suffices to optimize a simple linear readout layer in order to make the system perform certain tasks.

This means that we can outsource part of our time dependent signal processing to a system that does not require any optimization and we only need to perform a simple linear training procedure ourselves. In ML, such a mapping is called a temporal feature expansion and the system that performs it in RC is called a reservoir. An additional advantage of this computational scheme is that it does not imply strong quality requirements on the components of the reservoir, as its non-idealities

are also taken into account when training the readout layer.

Note that we are not limited to the class of RNNs in our search for suitable reservoirs. On the contrary, as long as the inherent dynamics of a system are rich enough for the task at hand and the system has adequate memory, it is a possible candidate for this computing scheme [15]. One type of systems that has already proven to be especially successful at reservoir computing, is the group of classical photonic systems [16]. These systems are a source of inspiration for this thesis.

However, keeping in mind the idea of QML, there is also no reason to restrict the range of possible reservoirs to classical systems. Instead, we can utilise the rich dynamics of quantum physics to boost the efficiency of RC. By doing so, we are led to a specific subset of QML systems, called quantum reservoir computing (QRC) [17] systems. In conclusion, based on the success of classical reservoir computing, QRC aims to exploit the quantum nature of physical systems by combining it with this easy training strategy and these low quality requirements for the components.

In this thesis, we will introduce and simulate a new QRC model that is based on photonic quantum computing. Before we move on to its construction, let us first elaborate on the structure of this thesis.

1.1 Structure of the thesis

- **Chapter 2** introduces a number of relevant machine learning concepts, which we will further use to discuss RC more in-depth. Building on this, we will discuss QRC and provide an overview of the existing literature in this field.
- **Chapter 3** focuses on quantum optics. It introduces some relevant concepts from this field that we will use to perform QRC.
- **Chapter 4** introduces our new QRC model and the task that it will solve. Afterwards, this chapter will describe how the simulation of the model is performed.
- **Chapter 5** analyses the model that was introduced in chapter 4 as a function of its parameters and also highlights some elements of its behaviour.
- **Chapter 6** sums up the possible extensions of the model and indicates the future objectives of this thesis.
- **Chapter 7** gives a summary of the most important findings.

Chapter 2

Machine learning concepts

2.1 Learning strategies

The field of machine learning can broadly be subdivided into three strategies: supervised learning, unsupervised learning and reinforcement learning. Supervised learning is the most common subbranch of machine learning and makes use of a labelled data set, i.e. a data set consisting of input-output pairs of the task that we want the machine to learn. This type of data set is not used in both other learning strategies. In unsupervised learning, the aim is instead to look for underlying patterns in a given data set. In other words, this training strategy is said to be self-organized. Finally, the goal of reinforcement learning is to train an intelligent agent, for example a robot, to take actions in an environment, based on a certain reward that is linked to the successful completion of the task. In the remainder of this thesis, we will only focus on the first of these three strategies, supervised learning. Let us therefore discuss it in more detail.

The intention of supervised learning is to learn a function that maps a certain input to a certain output. This procedure is called training and it is performed by considering a set of example input-output pairs. The main purpose of this procedure is to be able to apply the trained function to unseen input data, thereby solving the task at hand. This desired behaviour is called generalization. Supervised learning consists of two subcategories, namely regression and classification.

Classification focuses on data sets where each input belongs to a specific class. The goal is to correctly classify each of these data points. A typical example of such a task is image classification, where the input data consists of pictures and the task is to recognise what is shown. For example, a data set of animal pictures can be labelled with a number of animal names. Applying a ML algorithm to this data set, a system can be created that maps the given pictures to the correct animal names. The system is said to learn a certain mapping function. Afterwards, when provided with an unseen photo, the system will also use this function to determine what animal is depicted. Note that the pictures in this example are composed of pixel values. These are examples of the so-called (input) features that the system

receives.

The second category, regression, focuses on predicting continuous label values. An example of this is the prediction of house prices. In this example, the available features are different house properties, such as its surface area and its number of rooms. After training, the optimized system can predict the value of a house that was not previously considered, based on its properties.

The simplest form of regression is linear regression, for which the mapping function takes the following form:

$$\hat{y} = \sum_{j=1}^{d} w'_j x'_j + b = \boldsymbol{w'}^T \cdot \boldsymbol{x'} + b = \boldsymbol{w}^T \cdot \boldsymbol{x} \tag{2.1}$$

Here, $\hat{y}$ is the predicted output and $\boldsymbol{x'}$ contains the input features of a certain data point, such as the house properties in the example mentioned above. By expanding $\boldsymbol{x'}$ with a constant value of 1, we form the vector $\boldsymbol{x}$. This allows us to combine $\boldsymbol{w'}$ and b into the so-called weight vector $\boldsymbol{w}$, which contains all optimizable parameters of the linear regressor.

In general, for a supervised ML approach, the optimization of such parameters can be formalised by introducing a cost function (also called a loss function). The optimization problem seeks to minimize this cost function, often by using an iterative process such as gradient descent [18]. In this case, for linear regression, the loss function is given by:

$$||\boldsymbol{y} - \boldsymbol{X} \cdot \boldsymbol{w}||_2^2 \tag{2.2}$$

Here, $\boldsymbol{X}$ is the feature matrix containing the feature values for all data points in the training set (i.e. for all so-called training samples) and $\boldsymbol{y}$ is the desired output (also called the label) for all of these samples. Linear regression is one of the few cases where a closed-form solution exists for this problem, which can be used alternatively to the previously mentioned iterative processes.

In the following two sections, we will introduce two ML models: neural networks and reservoir computing. At a later stage in this thesis, we will substitute both of these two concepts for their quantum mechanical counterparts.

2.2 Neural network

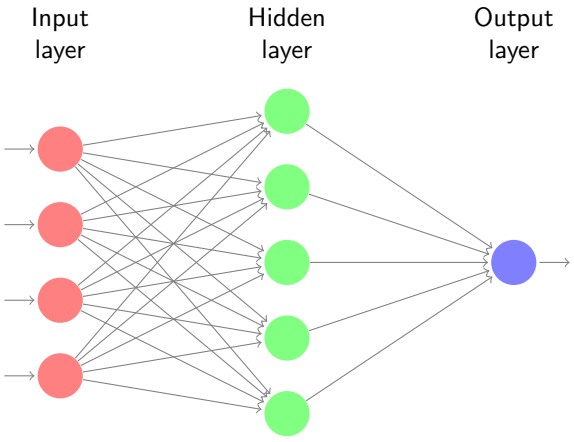


Figure 2.1: Schematic representation of a simple feedforward neural network, comprising of an input layers, a single hidden layer and an output layer. The arrows depict connections between the nodes of these layers. All connections are directed towards the output layer and are assigned a certain weight.

A neural network (NN) [9] is a computing system that is vaguely inspired by a biological brain and therefore falls within the category of neuromorphic computing. A schematic representation of such a network is shown in Figure 2.1. In general, a neural network consists of a collection of nodes that are grouped into different layers. The first and last of these layers are respectively referred to as the input and output layers. All layers in between are so-called hidden layers. The nodes of different layers are connected to each other via interconnections. All of these connections are assigned a certain weight. For now, we only consider the case where these connections do not form cycles and are directed towards the output layer, leading to the concept of a feedforward neural network.

Let us first consider how such a network processes information. Starting at the input layer and moving towards the output layer, we repeatedly consider the connections in between two consecutive layers. Each time, the weights of these connections are applied to the node values of the first layer and the results are transferred to the second layer. The results of connections that lead to the same node are summed. Before assigning the values to the nodes of the second layer, an activation function is applied that controls the magnitude of these values. This activation function can be of many types [19], but most often it is a rectified linear activation function (ReLU), a logistic sigmoid function or a hyperbolic tangent (tanh).

The weights form the set of optimizable parameters of the network. They can be updated iteratively to get the desired values at the output layer. In order to perform such a training procedure, once again, we define a certain cost function.

Furthermore, we can apply the so-called back propagation technique [20] to minimize this cost function. In order to do so, the gradients of the cost function are calculated with respect to the weights of the network. These calculations are performed efficiently by making use of the chain rule. Starting from the output layer an propagating backwards through the network, the gradients are calculated layer per layer. In order to minimize the cost function, the resulting gradients can be used to update the weight values according to a certain gradient method. Common examples of such methods are gradient descent [18][21] and its variants, such as stochastic gradient descent [22].

Generalizing the concept of feedforward neural networks, we also define recurrent neural networks (RNNs) [14]. These networks additionally allow for cycles in their structure. These cycles are also called recurrent connections or feedback connections. Transferring information via the interconnections of the network, these cycles ensure that input information is not immediately directed towards the output layer. Instead, the cycles grant the system memory and enable it to learn temporal tasks.

2.3 Reservoir computing

A ML approach that finds its origins in the study of RNNs, is reservoir computing (RC) [12][13]. A typical problem with RNNs is that they are often hard to train [23]. RC can help in this respect as it appears that it can be sufficient to train only the connections leading to the final output layer of an RNN, without noticeable loss of computational power [24].

In RC, this finding is extended to a wider range of nonlinear, dynamical systems. These systems are kept fixed and are used to map input signals into a higher dimensional space. In other words, the intrinsic and rich dynamics of both software-implemented and physical systems can be used to outsource a part of a computation. By doing so, the computational power of such systems is used to reduce the effective computational cost of the total training procedure. Before discussing examples of such systems, let us first have a look at a general RC procedure.

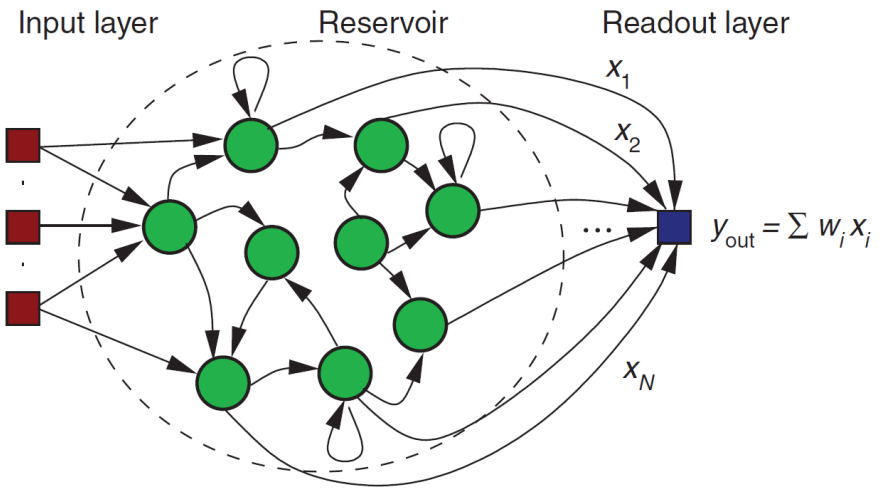


Figure 2.2: Standard layout of a reservoir computer, comprising an input layer (red), the reservoir (green) with randomized but fixed connections, and the linear readout layer (blue). Here, for simplicity a one-dimensional readout layer is drawn that applies weights $\{w_1, w_2, ..., w_N\}$ to the nodes $\{x_1, x_2, ..., x_N\}$, also called the true nodes. This figure is drawn from Ref. [13]

Figure 2.2 shows a schematic representation of a reservoir computer. Its input layer is used to encode the training data. The reservoir itself is treated as a black box with fixed parameters. Only the weights of the readout layer are optimized in order to train the system. For this purpose, the previously introduced linear regressor can be used. Its weights $\{w_1, w_2, ..., w_N\}$ are optimized such that it maps a subset of the reservoir nodes $\{x_1, x_2, ..., x_N\}$, the so-called true nodes, onto the desired output of the task y_{out}. After the optimization process, a similar procedure can be performed in order to validate the performance of the system.

The state of the reservoir at a certain iteration k can generally be described by the following update equation:

$$\boldsymbol{x}(k) = f\left(\boldsymbol{W}_{\text{res}} \cdot \boldsymbol{x}(k-1) + \boldsymbol{W}_{\text{in}} \cdot \boldsymbol{x}_{\text{in}}(k)\right) \tag{2.3}$$

Here $\boldsymbol{x}_{\text{in}}(k)$ is the injected input data at iteration k, which is transformed according to the matrix $\boldsymbol{W}_{\text{in}}$ before entering the reservoir. $\boldsymbol{W}_{\text{res}}$ and f respectively describe the internal connectivity and the nonlinear dynamics of the reservoir.

Feedback connections in the reservoir enable the system to learn temporal tasks. Each iteration, new information is fed into the system which influences the state of the reservoir according to Equation (2.3). It is however necessary that the influence of the information on the reservoir state becomes smaller the longer ago it was injected. If so, the state of the system is only determined by its recent input history. This concept is referred to as the fading memory principle. Although it is a necessary property for a dynamical system in order to perform reservoir computing, this constraint proves not to be restrictive [15] and it is naturally satisfied by most

systems that contain recurrent connections. In fact, the difficulty in this respect often lies in assuring that the memory of the reservoir is long enough for the task at hand.

One of the key advantages of reservoir computing is that the fabrication tolerances for the components of the reservoir are less important, because for each reservoir a bespoke set of weights is acquired that partly takes these non-idealities into account. In the past, it has been shown that reservoirs with such noisy components can still be used to learn complex, non-linear tasks [15].

Although software-implemented reservoirs, such as RNNs, are often used, the actual range of suitable dynamical systems is extremely broad. Physical systems can be used as long as their dynamics are sufficiently rich and complex to allow them to be employed for information processing [15]. This includes that the system leads to a state that is nonlinear in its inputs to achieve nontrivial information processing [25] and that it exhibits fading memory. A good illustration of the generality of these requirements is that ripples on a water surface, due to their nonlinear nature, have been used as an analog physical system to perform reservoir computing [26]. Examples of physical systems that are more frequently used, are classical photonic systems [16]. As mentioned before, we will also consider a photonic system in this thesis.

In the past, classical RC has already proven to lead to state-of-the-art performance in tasks such as speech recognition [27], where speech is the temporal input signal and the task is for example to recognise individual words. Other examples of such tasks are chaotic time series prediction [28] and radar signal forecasting [29].

2.4 Quantum reservoir computing

Based on the success of classical RC, quantum reservoir computing (QRC) aims to boost the performance of reservoir computers further by exploiting the quantum nature of physical systems. As mentioned before, RC is characterised by an easy training strategy and low quality requirements for the reservoir components. This is also the case in QRC. It can be noted that quantum computing systems are currently subject to strong hardware constraints. Although a universal, fault-tolerant and efficient quantum computer is the long-term goal in the field of quantum computing, this will potentially take decades to achieve. One of the key limitations in making these systems is that the information encoded in a state of the quantum system easily and uncontrollably becomes entangled with its environment, a concept that is referred to as decoherence [30]. This stresses the importance of error correction schemes in the construction of a long-term quantum computer. At the moment however, noisy intermediate-scale quantum (NISQ) [31] computers already exist that use hundreds of qubits, often without error correction, to perform imperfect operations. In this context, QML techniques like QRC can be seen as a alternative way to exploit the computational power of quantum systems, making use of the currently available resources to perform classically challenging tasks and possibly removing the need for error correction in the process.

In QRC, one of the main goals is to improve upon the efficiency of existing

classical schemes, both in terms of computational power and power consumption. However, QRC also aims to achieve other goals. Adding to the set of classical tasks that was mentioned in the previous section, QRC enables us to train quantum tasks, i.e. to exploit the quantum information in either or both the input and the output states. An interesting example of such a task would be to determine whether a quantum channel is Markovian or not. Similar to the concept of a Markov chain [32], which describes a sequence events in which the probability of each event depends only on the state attained in the previous event, Markovian quantum channels are not subject to memory effects. In contrast, non-Markovian quantum channels are affected by such memory effects, which results from interaction with their environment. These effects may be exploited to increase the capacity of a quantum channel [33]. Another quantum task would be to determine whether noise from different subsystems of a quantum system (e.g. different subsets of qudits) is correlated or uncorrelated. This is important in the context of error correction [34].

2.4.1 QRC platforms

The number of platforms that have peen proposed for QRC has increased significantly in recent years, owing both to the wide range of suitable RC reservoirs (as explained in Section 2.3) and to the widespread developments in the field of QC. An important characteristic that all of these platforms have in common is that they show an exponential growth in Hilbert space dimensions when increasing the number of quantum elements. This provides them with a large state space which plays a decisive role in the performance of QRC systems [35].

Before moving on to our own photonic set-up, let us summarize the most important of these QRC platforms. At the moment, spin-based QRC approaches such as nuclear magnetic resonance (NMR) systems [35][36] and trapped ions [37] are studied the most, both regarding simulations and physical experiments. NMR quantum computing is particularly interesting as it relies on an ensemble of identical molecules of which the nuclear spin degrees of freedom can be employed to perform calculations. For this reason, it naturally possesses interesting measurement characteristics. More specifically, because a measurement of a particular observable is taken over the entire ensemble, it has little impact on individual spins. This detection can be seen as a type of monitoring without disturbance [38], which contrasts with the strong projective measurements, i.e. probabilistic measurements which disturb the state of the system that they measure, that are common on most other quantum computing platforms.

On the one hand, in the context of quantum computing, this property of NMR computing is often seen as problematic. Strong projective measurements are for example often used to perform qubit initialisation and error correction. On the other hand, in the context of QRC, this detection property also seems to be very useful. The reason for this can be found in the repetitive output extraction that is often performed for temporal tasks. Indeed, due to the stochastic nature of strong projective measurements, we are generally required to prepare several copies of the system or to perform multiple measurements in order to extract information about the observables. Typically, this information is a set of expectation values. As

on the NMR platform such an ensemble is naturally present, these system copies do not need to be created. Moreover, these noninvasive measurements can also reduce other unwanted back-actions of the detector that result from its imperfect realisation rather than from its stochastic behaviour.

Another interesting QRC approach is introduced in Ref. [39], performing QRC using superconducting qubits on an IBM quantum processor [40]. Further, Ref. [39] also gives an analytical derivation of a universal approximation property, meaning that it defines a class of systems that can approximate a class of functions with arbitrary precision.

Finally, we also mention that photonic systems are promising candidates to perform QRC. This is due to the fact that photonics has already shown to be especially successful at classical reservoir computing [16], as well as the fact that this platform is seen as a path towards scalable quantum computers [41]. One such photonic QRC approach is proposed in [42], making use of Gaussian states to encode continuous variables and also leading to a universal approximation property for the class of reservoirs. Another photonic model, based on the continuous monitoring of a single Kerr nonlinear oscillator was introduced in Ref. [43].

In this thesis, we will also make use of photonics to define a new QRC system. However, before we can introduce its configuration, we first have to introduce the quantum optical concepts that it relies upon.

Chapter 3

Quantum optical concepts

In this chapter, we introduce the quantum states that we will use to describe quantum information in this thesis: Fock states. After elaborating on how these states can be generated and detected, we will show how they can be used to perform computational tasks that are hard for classical computers. Eventually, we discuss a QML approach that was previously introduced in literature, a quantum optical neural network (QONN), which will be at the heart of our own QRC model.

3.1 Fock states

Whereas bits are the known units of information in classical computing, in quantum information science we make use of qudits. Unlike classical bits, which always have a value of either 0 or 1, qudits can be in a superposition between the possible states of a d-level system. In the case of a two-level system, we often use the more widely known term qubits.

In this thesis, qudits will be further desribed by Fock states, which are elements of a Fock space. Fock states consist of a well-defined number of identical particles. In photonics, these particles are single photons that can be present in a number of optical modes. If the system under consideration for example contains 2 non-interacting photons in 3 optical modes, the set of Fock states that form a Fock basis for this system is given by:

$$\{|2,0,0\rangle, |0,2,0\rangle, |0,0,2\rangle, |1,1,0\rangle, |0,1,1\rangle, |1,0,1\rangle\}$$

In general, when dealing with n photons and m optical modes, these Fock states can be written as $|n_1, n_2, ..., n_m\rangle$, where $\sum_{i=1}^{m} n_i = n$. It further follows from combinatorics that the dimension of the Fock basis equals:

$$\begin{pmatrix} n+m-1 \\ m \end{pmatrix}$$

Although this thesis will only simulate these Fock states, rather than performing experiments, in the next two sections we will briefly introduce how they can be generated and detected.

3.2 Single-photon sources

Generating a photonic Fock state requires the use of reliable single-photon sources [44][45]. An ideal single-photon source would be one that only emits indistinguishable, single photons at an arbitrary time and at an arbitrarily fast repetition rate. In reality, several technologies exist to implement these sources, but they all exhibit deviations from this ideal behaviour. Based on whether or not these sources can emit single photons at arbitrary times, also called generation 'on demand', we can divide them into two groups: deterministic photon sources and probabilistic photon sources.

Deterministic sources ideally produce single photons whenever they are triggered. They can be implemented for example using color centers, quantum dots, 2D materials, single molecules, atoms or ions. These all function as two-level systems that can be excited with a laser pulse. As the lifetime of the excited state of such a system is finite, a single photon can be generated with a short pulse. Although this generation on demand is very desirable, deterministic sources are not the most widely used sources at the moment and research is still being done to find easily implementable models.

The second group of sources, probabilistic sources, are not able to generate single photons on demand. Instead, they utilise lasers to perform photon pair creation with a certain probability, either via parametric down-conversion (PDC) in bulk crystals and waveguides or via four-wave mixing (FWM) in optical fibers.

Currently, the single photon generation technique that is most widely used, is PDC. Let us therefore discuss this process in more detail. In PDC, a nonlinear crystal is used to convert one photon of higher energy, coming from a pump laser, into a pair of photons of lower energy, in accordance with the law of conservation of energy and the law of conservation of momentum. The initial photon is called the pump photon, while the resulting photons are called the signal photon and the idler photon. By detecting the idler photon, we can deduce whether we successfully created a single photon and if so, at what moment this process happened. This process is better known as heralding. Additionally, from the phase matching condition we can derive the relative polarization of the signal and the idler, which enables us to identify the state of the signal photon. Although PDC is most often used to generate a single photon, there are also extensions that allow, for example, the generation of two and three photons in the same mode [46].

3.3 Single-photon detectors

Knowing how Fock states can be generated, let us consider how they can be detected. For this purpose, we need to make use of single-photon detectors [44][47]. Similar to the discussion on single-photon sources, let us first describe what the ideal characteristics of such a detector would be. Whenever a photon is incident, we want to detect it with a probability of 100% (detection efficiency), without variance in the delay between the optical input and the electrical output (jitter time). Moreover, we do not want any dark counts, i.e. electrical output without optical

input. After a detection, we also want to be able to detect a new photon arbitrarily fast (without so-called dead time). Again multiple techniques exist to manufacture such detectors, but in reality they all suffer from non-idealities. The most common examples of such techniques are: single-photon avalanche photodiodes (SPADs), photomultiplier tubes (PMTs), superconducting nanowire single-photon detectors (SNSPDs) and up-conversion single-photon detectors. Note that a lot of variations and even hybrids exist that are based on these methods. One characteristic that the mentioned techniques have in common, however, is that they cannot reliably detect exactly how many photons are present. They are therefore not photon-number-resolving and if they are triggered, we only know, with a certain confidence, that at least one photon must have been present.

3.3.1 Number-resolving detectors

In some applications, as is also the case in this thesis, it is additionally required to detect the actual number of photons that are present. Once again, different techniques exist to tackle this problem of which we list the most common ones: superconducting tunnel junction (STJ)-based detectors, Quantum-dot field-effect transistor (QDFET)-based detectors, superconducting transition edge sensors, visible light detectors and variants of the aforementioned SNSPDs.

Apart from intrinsically number-resolving detectors, we can also combine a set of detectors that are not photon-number-resolving in order to approximately detect the number of photons that is present. This concept is called a threshold detector [48]. It is constructed by connecting the mode that we want to measure to a cascade or tree of fibre splitters, the ends of which are each connected to non-number-resolving detectors. As the size of the tree grows, so does the probability of detecting all photons that were initially bunched into one mode.

3.4 Linear interferometer

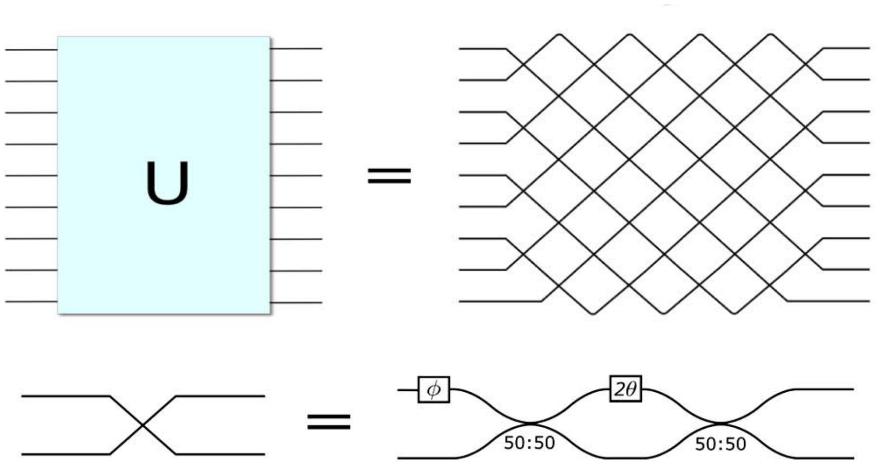


Figure 3.1: Schematic representation of an interferometer according to the configurations of Clements *et al*. This figure is drawn from Ref. [49]

In between the generation of Fock states and their detection, these states can be manipulated. This can for example be done using a linear interferometer. This component is a combination of beamsplitters and phase shifters that are combined in a certain configuration and act on a number m of optical modes. The configuration that will be used in this report was proposed by Clements *et al*. [49] and is depicted in Figure 3.1. Note that it only uses beamsplitters with equal splitting ratio, also called 50/50 beamsplitters.

This figure also shows that the beamsplitters and phase shifters are combined in elementary components that act on neighbouring modes. By applying a phase shift between these modes and leading them both to a beamsplitter, interference of the light is made possible. The combination of two such phase shifters and beamsplitters (shown at the bottom of the figure) is better known as a Mach-Zehnder interferometer.

In the scheme of Clements *et al*, the linear interferometer is characterized by a unitary transformation $\boldsymbol{U}(\boldsymbol{\theta})$, which is parameterized by m^2 phase shift parameters $\theta_j \in (0, 2\pi]$. The matrix itself has a shape of $m \times m$ and is constructed by combining the individual transformation matrices of the beamsplitters and the phase shifters, which are given by:

$$\boldsymbol{BS} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{3.1}$$

$$\boldsymbol{PS} = \begin{pmatrix} e^{i\theta_1} & 0 \\ 0 & e^{i\theta_2} \end{pmatrix} \tag{3.2}$$

The linear interferometer that corresponds with $\boldsymbol{U}$ induces a transformation $\boldsymbol{W}$ on its multi-photon input states. In other words, $\boldsymbol{U}$ operates on a single-photon Fock space, while $\boldsymbol{W}$ operates on a larger, multi-photon Fock space. As explained in Section 3.1, the number of basis states of this multi-photon Fock space is given by $\binom{n+m-1}{m}$, which scales exponentially in n and m. As will be discussed further in the next section, $\boldsymbol{U}$ and $\boldsymbol{W}$ are connected through a homomorphism Φ.

3.5 Boson sampling

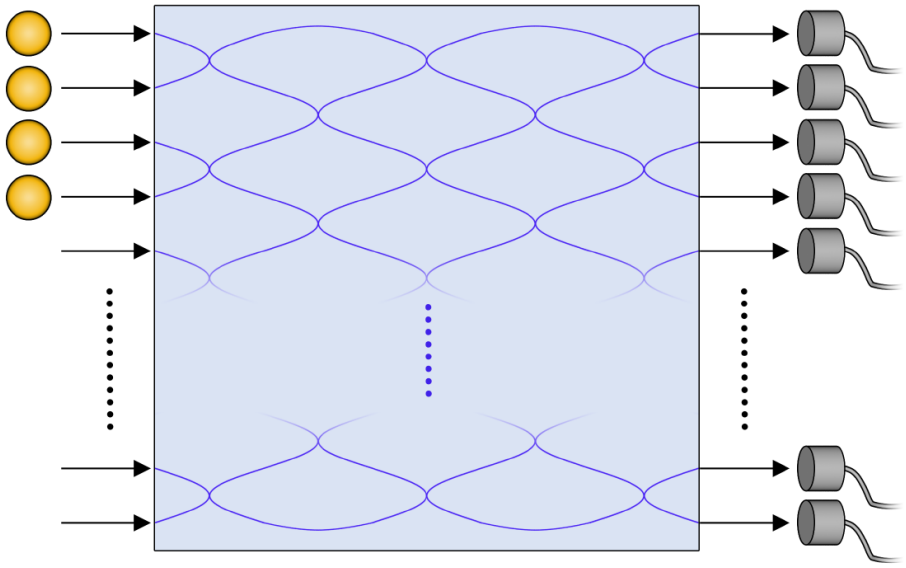


Figure 3.2: Schematic representation of the boson sampling procedure. Indistinguishable photons enter a linear interferometer and are guided to single-photon detectors. For simplicity, the figure only shows a single photon per input mode (yellow-filled circles) and the detectors can be assumed to be number-resolving. This figure is drawn from Ref. [50].

Having introduced the linear interferometer, let us consider how this component can be used to perform boson sampling [51], which is a well-known proposal to demonstrate quantum supremacy. Boson sampling is the act of sampling indistinguishable photons using photon-number-resolving detectors after they have travelled through a linear interferometer. We note that in reality, however, this procedure is often carried out with detectors that cannot resolve the photon number. This is possible because of the bosonic birthday paradox [52], which states that the probability of encountering photons in the same mode is low when they are scattered over $m >> n^2$ modes, where n represents the number of photons that are present and

m is the number of optical modes.

Although this quantum computing operation is non-universal, meaning that it cannot be used to express any quantum algorithm, it is strongly believed to be computationally hard for classical computers. This statement can be understood from the form of the probability distribution that is sampled by the single-photon detectors. This probability distribution is proportional to the permanent of complex matrices that are directly derived from matrix $\boldsymbol{U}$ [53]. At first glance, a permanent strongly resembles a determinant. It is formed by similar polynomials in the entries of the matrix under consideration, where the only difference is that no additional signs are added to the polynomial of the permanent. The calculation of a permanent is however substantially more difficult than the calculation of a determinant.

In more mathematical terms, it is said that the calculation of determinants falls within the P-hard complexity class, while the calculation of permanents falls within the #P-hard complexity class, which is in turn strictly harder than the better-known NP-complexity class. Here, P is the set of problems that are solvable in polynomial time by a classical computer, while NP is the class of problems for which a solution can be verified in polynomial time by a classical computer.

This discussion can therefore be related to the P versus NP problem [5], which is a major unsolved problem in computer science. In short, this problem is concerned with the question as to whether the P and NP complexity classes coincide or not. If it turned out that P $\neq$ NP, which is widely believed but not proven, it would mean that there are problems in NP that are harder to compute for a classical computer than they are to verify, leading to a possible advantage of quantum computers. As a result, the discussion above indicates that the calculation of permanents is at the source of the hardness of the boson sampling problem.

When simulating a boson sampling procedure on a classical computer, this computational hurdle emerges when transforming $\boldsymbol{U}$ of the linear interferometer to $\boldsymbol{W}$ which acts on the multi-photon input states. As noted earlier, $\boldsymbol{U}$ can be easily constructed by combining the individual matrices of the beamsplitters and phase shifters. We know that there exists a homomorphism Φ that transforms $\boldsymbol{U}$ to the multi-particle unitary $\boldsymbol{W}$, but its evaluation depends on calculations of the permanents. The asymptotically fastest known algorithm that exactly computes the permanent of an $m \times m$ matrix is Ryser's algorithm [54]. It has a complexity of $O(2^{m-1}m)$. This algorithm will also be used in this thesis.

It has to be noted that a physical boson sampler is significantly more straightforward to build than any universal quantum computer proposed so far, making it a feasible way to demonstrate quantum supremacy. Therefore, we can conclude that while being considered intractable for classical computers, boson sampling can naturally be solved on a photonic quantum substrate, while requiring a relatively low amount of resources. An interesting illustration of these statements, is the recent demonstration of quantum supremacy using boson sampling, for which 50 photons were guided in 100 optical modes [55]. This shows that quantum supremacy can already be attained by near-term quantum computers, since it does not require the performed task to be useful and it neither requires the quantum computer to perform high-quality quantum error correction, both of which are long-term objectives.

3.6 Quantum optical neural network

Now that we have introduced some important concepts of both machine learning and quantum computing, let us look at a certain class of quantum machine learning systems: quantum neural networks (QNN) [56][57]. These networks are a type of variational quantum algorithms [58], which employ a classical optimizer to train a parameterized quantum circuit. QNNs consist of gates whose parameters can be optimized to perform a desired transformation on the data that is sent in the network. Often, these networks are capable of performing a broad range of tasks [59], both classical and quantum mechanical. They can be used to experimentally discover algorithms that are faster or require smaller networks than previously used classical or quantum algorithms. Another advantage is that they often can be implemented using noisy intermediate scale quantum (NISQ) technology [60], for which there is an ongoing search for suitable algorithms.

A wide variety of QNNs exist and they can be defined on different platforms. One such example that can be implemented on a photonic platform is the quantum optical neural network (QONN) [61], which is depicted in Figure 3.3.

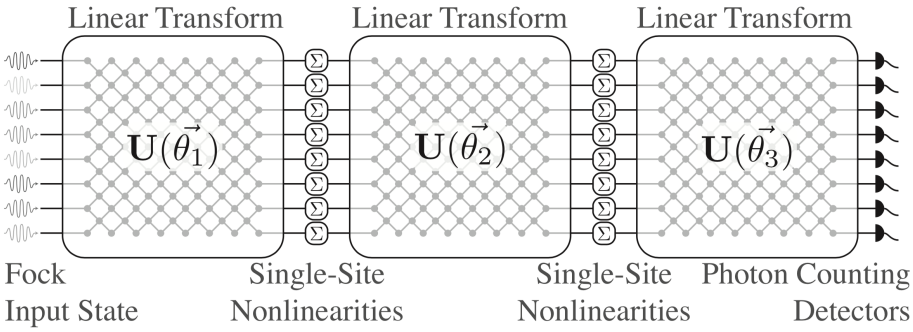


Figure 3.3: QONN layout. Inputs are Fock states and the linear transforms are parameterized linear interferometers. The single-site nonlinearities are given by $\chi^{(3)}$ functions: a Kerr-type interaction applying a constant phase for each additional photon present. Readout is performed by single-photon detectors which measure the photon number at each output mode. This figure is drawn from Ref. [61].

The QONN can be subdivided into three parts. Firstly, single-photon sources are used to encode the training data. If the training data is not defined on a Fock basis, as it could be classical data for example, it has to be encoded into a Fock basis following a well-defined procedure. Secondly, the resulting Fock states are fed to a succession of layers. One such layer consists of a linear interferometer and a combination of single-site nonlinearities $\mathbf{\Sigma}$. These nonlinearities are single-mode Kerr interactions in the monochromatic approximation, applying a phase that is quadratic in the number of photons that is present [62], as shown in the following

equation:

$$\boldsymbol{\Sigma}(\phi) = \sum_{n=0}^{\infty} e^{in(n-1)\phi/2} |n\rangle\langle n| \tag{3.3}$$

The strength of these nonlinearities is defined by the interaction strength ϕ, but is typically fixed to π. The resulting transformation that is performed by a QONN consisting of N layers, is given by:

$$\begin{aligned} \boldsymbol{S}(\boldsymbol{\Theta}) =& \Phi\left[\boldsymbol{U}\left(\boldsymbol{\theta}_N\right)\right] \prod_{i=1}^{N-1} \boldsymbol{\Sigma}(\phi)\ \Phi\left[\boldsymbol{U}\left(\boldsymbol{\theta}_{N-i}\right)\right] \\ =& \boldsymbol{W}\left(\boldsymbol{\theta}_N\right) \prod_{i=1}^{N-1} \boldsymbol{\Sigma}(\phi)\ \boldsymbol{W}\left(\boldsymbol{\theta}_{N-i}\right) \end{aligned} \tag{3.4}$$

Here, $\boldsymbol{S}$ is parameterized by $\boldsymbol{\Theta}$, the Nm^2 phase parameters of the linear interferometers. Note that further we will not write ϕ (the Kerr interaction strength) in the argument of $\boldsymbol{S}$ to simplify its notation. Also note that the nonlinear layers, which result from applying Equation (3.3) to all modes of the system, are also written as $\boldsymbol{\Sigma}(\phi)$ to simplify the notation. Further, recall that Φ is the homomorphism between $\boldsymbol{U}$ and $\boldsymbol{W}$. The boson sampling result $\boldsymbol{W}\left(\boldsymbol{\theta}_N\right)$ is placed outside of the product to account for the fact that nonlinearities are only added in between interferometers.

Finally, the transformed Fock states are detected by single-photon detectors. The QONN can be trained by defining a cost function and optimizing over $\boldsymbol{\Theta}$. In Ref. [61], QONNs are trained to perform a range of quantum information processing tasks, including newly developed protocols for quantum optical state compression, reinforcement learning, black-box quantum simulation, and one-way quantum repeaters. Consequently, this thesis will try to make use of the demonstrated computational power of such a QONN by including it in a new QRC approach.

Also in Ref. [61], a Python library was developed that allows to calculate the matrix $\boldsymbol{S}$ of such a QONN configuration. Its key functionality, namely the execution of Ryser's algorithm, was implemented in Cython (a compiled rather than an interpreted version of Python, designed to give C-like performance with code that is written mostly in Python) to increase the efficiency of the simulation. This Python library served as a useful starting point for the simulations performed in this thesis.

Chapter 4

Construction of a QRC model

In this chapter we introduce a QML approach that combines a quantum optical reservoir with a classical optimizer. It can be used to solve both classical and quantum tasks. In the first section of this chapter, we will sketch the configuration and the operating procedure of this system. From this discussion, it will follow that we need to combine multiple copies of the same reservoir into an ensemble in order to extract reproducible results. In Section 4.2 we will discuss the task that will be tackled. In the remainder of the chapter, we will elaborate on the different parts of the system and show how they are simulated. In Sections 4.3, 4.4 and 4.5, we deepen our understanding of the technicalities that build up the simulation of a single quantum reservoir in the ensemble. Afterwards, in Section 4.6 we discuss how the features are extracted from an ensemble of such reservoirs and how these features take shape for ensembles of different size. In Sections 4.7 and 4.8 we propose two different improvements of the current system, which will lead us to the final model that will be analysed in Chapter 5.

4.1 Configuration and operating procedure

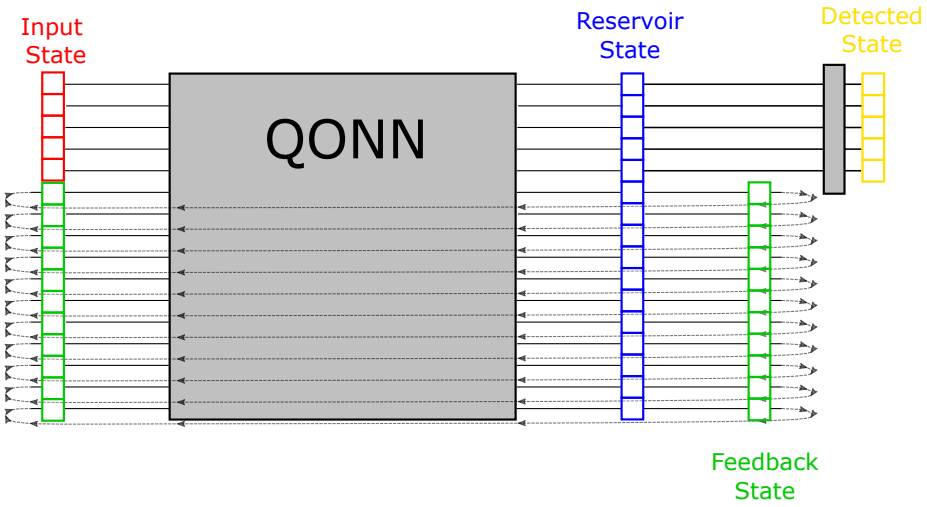


Figure 4.1: Layout of the quantum reservoir that is introduced in this thesis, comprising of input modes (red) and a randomly initialised QONN (grey square), leading to a reservoir state (blue). The detector (grey rectangle) leads to a certain detected state (yellow). The feedback state (green) is looped back to the input of the QONN.

Figure 4.1 schematically represents the quantum optical reservoir that we will consider in this thesis. The grey square in this configuration depicts the QONN that was introduced in Section 3.6. The phase parameters $\boldsymbol{\Theta}$ of this QONN are initialised randomly. Further, the train data (and later the test data) is encoded in the input modes, leading to an input state. The QONN performs a transformation $\boldsymbol{S}$ on this input state, as defined by Equation (3.4). This leads to a so-called reservoir state. Similar to the concept of true nodes that was introduced in Section 2.3, we do not measure the complete reservoir state. Instead, only a subset of the optical modes are guided to the detector, while the other modes are redirected towards the input of the QONN, forming recurrent connections. These two subsets of modes are respectively named the detection modes and the feedback modes. From now on, we will number the modes in Figure 4.1 from top (mode 1) to bottom (mode $m_A + m_B$), where m_A is the number of detection modes (also equal to the number of input modes) and m_B is the number of feedback modes.

The introduction of recurrent connections enables the system to learn temporal tasks. That is, it can learn to prepare the desired output, depending on multiple past inputs. In order to do so, we send n_{in} photons into the system at a constant rate. These photons encode inputs from the data set. There are many ways to perform such an encoding, but for now we assume that we want to be able to encode a discrete set of classical inputs and that the size of this set is smaller than

the number of input modes of the QONN. These assumptions hold for instance when the input data consists of classical bits and when the QONN has at least two input modes. The classical inputs can be encoded by associating each of its possible values with a different input mode. Knowing the value that we want to encode, we can send all of the n_{in} photons into the corresponding mode. Note that, by spreading the photons over the modes, this scheme can be extended to encode a larger set of possible data values with equal numbers of modes and photons. Moreover, it is possible to encode continuous data by preparing superpositions of Fock states, where the data is encoded in either or both the magnitude and the phase of the complex coefficients.

After preparing the input state, it is combined with the feedback state that results from the previous iteration. The resulting state is fed through the QONN. Here, the photons repeatedly interact via the network of Mach-Zehnder interferometers and the non-linear layers of Kerr interactions. As the feedback state is a result of multiple previous iterations, the reservoir state that we obtain also contains information about previous inputs. A part of this information is retained in the feedback state, while another part of the information is read.

The above leads to an important consideration. The state in front of the detector generally is a superposition of Fock states. Due to the stochastic nature of quantum measurements, only one of these Fock states can be detected. Apart from information loss, this results in different measurement outcomes for different simulations of the same reservoir. As a result, it is common for quantum (reservoir) computers to combine the measurement information of multiple copies of the same system in order to estimate the expectation values of the different possible detections [17]. These copies can either be created in parallel or by performing sequential measurements using the same, unaltered system. The sequential approach is often preferred for hardware implementations as it requires less resources. In a simulation, however, the parallel approach, which is depicted in Figure 4.2 for our quantum reservoir, can also be carried out. Although the simulations of different reservoirs are not currently performed in parallel, this schematic representation of an *ensemble* of reservoir copies helps to understand the system.

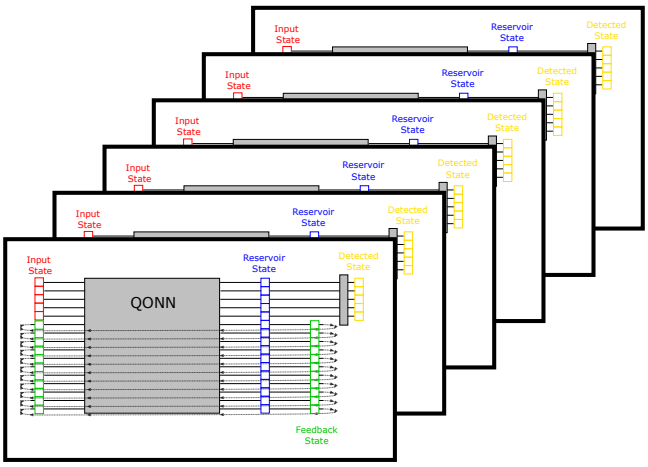


Figure 4.2: A schematic representation of an ensemble of reservoir copies. All parallel reservoir operations were initialised equally, but they can lead to different detected states as a result of the probabilistic measurement procedure.

Until now, the system can be interpreted as a temporal feature expansion that leverages the highly dimensional Hilbert space of the quantum network. Furthermore, we want to map the obtained features onto the desired output of the task. This is done by using a classical optimizer, more specifically a linear regressor. As is the case in classical reservoir computing, it is only required to optimize the weights of a linear readout layer. However, in this quantum approach, the construction of the readout layer is remarkably different. Where in Figure 2.2 weights were directly applied to the measured values of the true nodes, we now first combine the measurement results of multiple identical reservoirs in order to approximate the ensemble statistics (i.e. the expectation values). Afterwards, these statistics are fed into a regressor. During training, the weights of the regressor are optimized and subsequently stored. Afterwards, a similar procedure is carried out in order to validate the performance of the system, using the stored values of the weights.

4.2 Temporal tasks

Having introduced our QRC model, we also discuss the task that we want to perform. As was explained in the previous section, the optimization is currently carried out by a classical linear regressor, rather than by a quantum component. As a result, this model can only be used to perform tasks of which the desired output is classical.

Although the system can still be used to perform both classical and quantum tasks, for now we will focus on a well-known simple classical task: the temporal XOR task. The most basic form of this task is depicted in Table 4.1. It shows that all pairs of subsequent input bits are combined according to XOR logic.

Table 4.1: An example of the desired output for the temporal XOR task. The input bits of iteration $k-1$ and k are combined, leading to a 0-bit if both inputs are equal. Otherwise, the output is a 1-bit.

Iteration	0	1	2	3	4	5	6	7
Input bit sequence	1	1	1	0	0	1	0	1
Output bit sequence	1	0	0	1	0	1	1	1

As explained earlier, n_{in} photons are fed into the system at each iteration, depending on the value that we want to encode. As this task is defined using classical bits, we only need to use two of the input modes. From now on, we choose to send the n_{in} photons in the first mode if we want to encode a 0-bit. Conversely, we send the n_{in} photons in the second mode if we want to encode a 1-bit.

Note that the input bit sequence is a sequence of random bits. In the simulations of our system, we can therefore easily generate a data set with a random bit generator and calculate the desired output sequence. Additionally, as the required output for the task is a classical bit string as well, the bit error rate (BER), i.e. the fraction of all bits that were wrongly predicted, is an appropriate quality measure for the predicted output of our model.

4.3 Photon number fluctuations

In Sections 4.3, 4.4 and 4.5, we focus on describing the simulation of a single quantum reservoir in the ensemble. In the first of these three sections we discuss how the number of photons in such a reservoir can fluctuate between different iterations and between different reservoirs. These considerations will make it possible to reduce the number of times we perform Ryser's algorithm to implement the homomorphism Φ that was introduced in Sections 3.4 and 3.5.

Let us first describe how the Fock basis (introduced in Section 3.1) can take shape, depending an the number of photons that are present in our system. As explained in Sections 3.4 and 3.5, the interferometers in the QONN are characterised by unitary matrices $\boldsymbol{U}(\boldsymbol{\theta}_i)$, where $i \in \{1, ..., N\}$. These matrices act on a basis of single-photon Fock states. Using Ryser's algorithm, we are able to transform these unitary matrices $\boldsymbol{U}$ to the unitary matrices $\boldsymbol{W}$ that work on a basis of multi-photon Fock states. A multi-photon Fock basis can be built knowing the specific number of photons (n) and number of modes (m) that are present in the system. If for example $n = 2$ and $m = 3$, the basis on which the matrix $\boldsymbol{W}$ operates, is given by:

$$\text{basis}(n{=}2, m{=}3) = \Big\{ |200\rangle, |110\rangle, |101\rangle, |020\rangle, |011\rangle, |002\rangle \Big\} \tag{4.1}$$

However, if the photon number is uncertain, such that the superposition in front of the QONN contains Fock terms with different n, we could also combine multiple Fock bases. Assume for example that we want to account for all possible n values

up to a certain threshold value n_{thr}. We call the resulting set of bases a 'lossy basis', which can generally be written as:

$$\text{lossy\_basis}(n_{\text{thr}}, m) = \Big\{\text{basis}(n_{\text{thr}}, m), \text{basis}(n_{\text{thr}} - 1, m), ..., \text{basis}(1, m)\Big\} \quad (4.2)$$

When using this lossy basis, Ryser's algorithm returns a block-diagonal multi particle matrix that combines the boson sampling results $\boldsymbol{W}$ that follow from the individual bases. Note that this increases the number of times the algorithm should be performed and consequently the number of permanents that need to be calculated.

As will be explained in the remainder of this section, in our current simulation we do not have to use a lossy basis. However, we note that a lossy basis was required in an alternative reservoir configuration that is described in Appendix A. Although this alternative model turns out to be disadvantageous, its discussion strengthens the design choices of the configuration shown in Figure 4.1.

To see why we can use a 'regular' basis (such as the one in Equation (4.1)) in our current model, assume that the system does not contain any photons at the start of the first iteration and that we send n_{in} photons to the input modes at each iteration. Looking back to Figure 4.1, we see that the photons that build up the reservoir state can either be detected or can be directed to the feedback modes of the system. Hence, after the first measurement a certain number of photons leaves the system. Due to the probabilistic nature of this measurement, this number can differ between parallel reservoirs of the ensemble. However, what is important in this discussion, is that the number of photons that is present in each QONN is unambiguously known. For every reservoir in the ensemble, this photon number can be recursively calculated using the following formula:

$$n_{\text{res}}(k) = n_{\text{res}}(k-1) - n_{\text{out}}(k-1) + n_{\text{in}} \quad (4.3)$$

Here $n_{\text{res}}(k)$ and $n_{\text{out}}(k)$ respectively represent the number of photons in the QONN and in the detected state at iteration k. We see that, due to the probabilistic sampling of the detectors, the number of photons can also fluctuate within one reservoir of the ensemble, between different iterations.

The average value of the photon number can however easily be estimated. Assume for example that $n_{\text{in}} = 1$ and that the number of detection modes is equal to the number of feedback modes. Also assume that the randomly initialised phase parameters of the QONN are such that the photons are not preferentially redirected to the detector or to the feedback modes. Under these assumptions and after some transient period (that will be described in Section 4.7), the average photon number is given by:

$$1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + ... = 2 \quad (4.4)$$

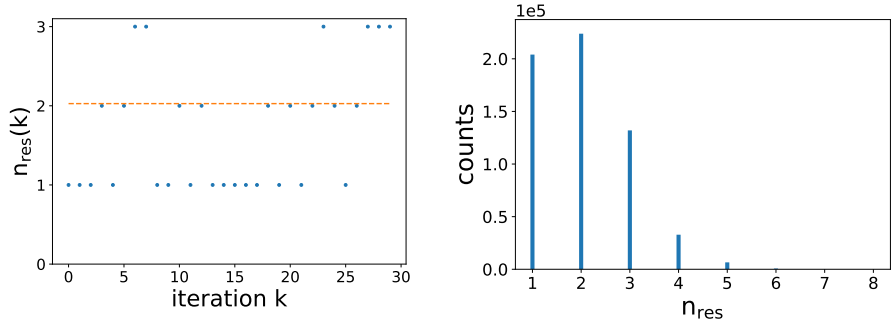
Here, the first term corresponds to the n_{in} photons that were injected at the last iteration. The second term corresponds to the average number of photons that

were injected at the penultimate iteration and that remained in the system and so on. In general, we have a geometric series of which the average value is given by:

$$n_{\text{in}} \sum_{k=0}^{\infty} r^k = \frac{n_{\text{in}}}{1-r} \tag{4.5}$$

Here, r is the average splitting ratio of the photons, which we assume to be close to $1/2$ for $m_A = m_B$.

Figure 4.3a shows the number of photons in one reservoir of an ensemble of size 600, during the first training iterations. We emphasise that these are iterations of the reservoir operating procedure in which we use data from the train set. The actual optimisation is only carried out by the linear regressor after all the data from the train set has been processed by the reservoir. In this simulation, we use values of $n_{\text{in}} = 1$ and $m_A = m_B = 2$, such that we expect the average photon number to be described by Equation (4.4). The orange, dashed line shows the averaged value of n_{res}, calculated over the total ensemble and over all training iterations. As can be seen, there is a good correspondence with the result from the geometric series. Figure 4.3b shows the distribution of n_{res}, also calculated over the total ensemble and training set. The distribution during testing is similar and is left out here.



(a) Blue dots: n_{res} values in a single reservoir during the first 30 training iterations. Orange dashed line: mean(n_{res}) of the same reservoir over all training iterations

(b) Histogram of n_{res} over all reservoir copies and over all training iterations

Figure 4.3: Simulated number of photons in the reservoir state (n_{res}) during training for an ensemble consisting of 600 reservoir copies. Other parameters: $m_A = m_B = 2$, $N = 6$ (as defined in Equation (3.4)), train set size $= 1000$, $n_{\text{in}} = 1$, $\phi = \pi$ (as defined in Equation (3.3))

As we know how many photons are in each reservoir at all iterations, we can boson sample the single-photon matrices $\boldsymbol{U}(\boldsymbol{\theta}_i)$ using specific values of n_{res} (recursively determined) and m (defined by the architecture of the system). What makes this observation all the more useful, is that the phase parameters $\boldsymbol{\Theta}$ of the QONN are identical throughout the ensemble and that they are kept fixed during

the simulation. This enables us to store and re-use the boson sampling results $\boldsymbol{S}(\boldsymbol{\Theta})$ of the QONN (as defined in Equation (3.4)), for different values of n_{res}. Doing so avoids repeated calculations of the same permanents. After each detection, Equation (4.3) gives a new value for n_{res} that can be used during the next iteration. If the boson sampling result for that value of n_{res} is already stored, it can be loaded from memory. If not, it is calculated explicitly and stored afterwards. This programming method is called 'memoization' [63]. This addition to the operating procedure shifts the computational load at runtime from calculations of permanents to matrix multiplications. This results in a clear advantage over the simulation of a regular QONN training procedure, where the previous boson sampling results are practically useless once the phase parameters are updated. This does however not imply that boson sampling results which require a lot of permanents to be evaluated, cannot be problematic anymore. These matrices still increasingly add to the computational load of the system as their size is given by:

$$\begin{pmatrix} n+m-1 \\ m \end{pmatrix} \times \begin{pmatrix} n+m-1 \\ m \end{pmatrix}$$

Note that, as a result of the photon number fluctuations, the Hilbert space on which the reservoir operation is performed has different dimensions between different iterations and between different reservoirs of the ensemble. This behaviour is very unusual when comparing this system with QRC approaches on other platforms and in some cases makes it difficult to transfer results and conclusions that were postulated there.

4.4 Combination of input and feedback

In this section we still focus on the description of a single reservoir, but now we describe how the input state ($|\psi_{\text{in}}\rangle$) and the feedback state ($|\psi_{\text{fb}}\rangle$) are combined at the beginning of each iteration. The input state is present in the first m_A modes of the system and consists of n_{in} photons, while the feedback state is present in the remaining $m - m_A = m_B$ modes and consists of n_{fb} photons. These modes are respectively indicated by the red and green boxes in Figure 4.1. As these subsets of modes do not overlap, the combined state has the following form: $|\psi_{\text{comb}}\rangle = |\psi_{\text{in}}\rangle \otimes |\psi_{\text{fb}}\rangle$. Here, $\otimes$ is the tensor product between the two Fock spaces on which $|\psi_{\text{in}}\rangle$ and $|\psi_{\text{fb}}\rangle$ were defined, namely $\text{basis}(n_{\text{in}}, m_A)$ and $\text{basis}(n_{\text{fb}}, m_B)$.

Consider for example the case where $m_A = m_B = 2$, $n_{\text{in}} = 2$ and $n_{\text{fb}} = 1$. Generally, the input and feedback state can be described by the following superpositions:

$$|\psi_0\rangle = a_1 |20\rangle_A + a_2 |11\rangle_A + a_3 |02\rangle_A \tag{4.6}$$

$$|\psi_{\text{fb}}\rangle = a_4 |10\rangle_B + a_5 |01\rangle_B \tag{4.7}$$

Here, we have introduced the labels A and B to denote the first m_A modes and

the last m_B respectively. This leads to the following combined state:

$$\begin{aligned}|\psi_{\text{comb}}\rangle =& \Big(a_1 |20\rangle_A + a_2 |11\rangle_A + a_3 |02\rangle_A \Big) \otimes \Big(a_4 |10\rangle_B + a_5 |01\rangle_B \Big) \\ =& \, a_1 a_4 |2010\rangle + a_1 a_5 |2001\rangle + a_2 a_4 |1110\rangle \\ &+ a_2 a_5 |1101\rangle + a_3 a_4 |0210\rangle + a_3 a_5 |0201\rangle \end{aligned} \tag{4.8}$$

Note that this procedure can further be simplified if we choose to follow the encoding scheme where all n_{in} photons are sent into the same input mode. Re-shifting the focus to the temporal XOR task, we put $a_1 = 1$ and $a_2 = a_3 = 0$ in the previous example if we want to encode a classical 0-bit. Conversely, we put $a_3 = 1$ and $a_1 = a_2 = 0$ if we want to encode a classical 1-bit.

4.5 Separation of the modes and detection

The remaining segment of the quantum reservoir that we have not yet thoroughly discussed, is the separation of the QONN output modes into the A and B modes and the subsequent detection. By performing a detection, wavefunction collapse occurs in the A modes of the system, which influences the remaining state in the B modes. To understand this procedure, let us first consider the following example. Assume that the reservoir state consists of two photons and that there are 4 reservoir modes. The first two of these modes will be detected, while the last two modes become the feedback modes. As we take $n_{\text{res}} = 2$ and $m = 4$, the dimension of the Fock basis before separating the A and B modes is equal to $\binom{5}{4} = 10$. Consequently, the reservoir superposition can generally be written as:

$$\begin{aligned}|\psi_{\text{res}}\rangle =& \, a_1 |2000\rangle + a_2 |1100\rangle + a_3 |1010\rangle + a_4 |1001\rangle + a_5 |0200\rangle + a_6 |0110\rangle \\ &+ a_7 |0101\rangle + a_8 |0020\rangle + a_9 |0011\rangle + a_{10} |0002\rangle \\ =& \, |00\rangle_A \otimes \Big(a_8 |20\rangle_B + a_9 |11\rangle_B + a_{10} |02\rangle_B \Big) \\ &+ |10\rangle_A \otimes \Big(a_3 |10\rangle_B + a_4 |01\rangle_B \Big) + |01\rangle_A \otimes \Big(a_6 |10\rangle_B + a_7 |01\rangle_B \Big) \\ &+ a_1 |20\rangle_A \otimes |00\rangle_B + a_2 |11\rangle_A \otimes |00\rangle_B + a_5 |02\rangle_A \otimes |00\rangle_B \end{aligned} \tag{4.9}$$

Here we have separated the detection modes (A) from the feedback modes (B) and grouped terms that have corresponding numbers of photons in the detection modes. As the feedback superpositions in between brackets still consist of orthogonal terms, the probability to detect $|00\rangle_A$ is equal to $|a_8|^2 + |a_9|^2 + |a_{10}|^2$. In general, the probability to sample a certain detection can thus easily be calculated by searching for all terms in the reservoir superposition that contain those specific detection values and summing over the corresponding norm squared complex coefficients.

Afterwards, the superposition in between brackets can be re-normalized to form the feedback state for the next iteration. By doing so, the new norm squared

complex coefficients add up to one. In the example above, after the detection of $|00\rangle$, the remaining feedback state becomes:

$$|\psi_{\text{fb}}\rangle = \frac{1}{\sqrt{|a_8|^2 + |a_9|^2 + |a_{10}|^2}} \Big(a_8 \, |20\rangle_B + a_9 \, |11\rangle_B + a_{10} \, |02\rangle_B \Big) \tag{4.10}$$

4.6 Input features of the linear regressor

Having discussed the simulation of a single reservoir, we combine multiple copies of the same reservoir into an ensemble. As explained in Section 4.1, the ensemble can be used to approximate the expectation values of the possible outcomes at each detection. These values serve as input features for the linear regressor. In Section 4.6.1 we discuss this process and explain how the stochastic nature of the photon fluctuations can lead to a different number of input features (further simply called features) during training and testing. As is clear from Section 2.1, this prevents the regressor from making a prediction of the desired output of the test data. Hence, we describe how this problem can be resolved. Afterwards, in Section 4.6.2 we show that the obtained feature values converge as the size of the ensemble is increased and we study the fading memory of our system.

4.6.1 Feature extraction

Each iteration, all reservoirs in the ensemble lead to a specific detection in the form of a Fock state that consists of n_{out} photons, spread over m_A modes. While m_A is clearly the same for all detections, n_{out} is not. In order to store a feature value for all possible detections and for all possible iterations, we store a feature matrix. The first axis of this matrix covers all iterations, while the second axis covers all basis states in $\text{lossy\_basis}(n_{\text{max}}, m_A)$, following the definition of the lossy basis in Section 4.3. Here, n_{max} is the maximum number of photons that is detected at once during the training procedure. Each iteration, we record the number of times the different states in the lossy basis are observed. Those numbers are divided by the ensemble size and stored in the feature matrix at their corresponding positions.

After constructing the feature matrix, we feed it to a linear regressor where the different iterations are interpreted as different samples. Doing so, a weight matrix is optimized that is used to transform the features of a certain iteration to the desired output of the task at that iteration. After training, the test data is sent through the reservoir and a new feature matrix is constructed. Using the optimized weight matrix from the training procedure, we can evaluate the performance of the model.

As this last feature matrix can differ in size from the feature matrix that was constructed during training, this leads us to a problem. Due to the probabilistic nature of the photon number fluctuations, it is possible that a Fock state is detected during testing that contains more photons than any of the detections did during training. In that case, n_{max} is higher during testing than it was during training. Consequently, each sample that is presented to the regressor has a different number of features at the testing stage than it had during training, which is not possible.

A first possible solution to this problem is to add a number of extra photons n_{extra} to the n_{max} of the training procedure, before constructing the feature matrix. Note that the value of n_{extra} must not depend on n_{max} of the testing procedure. As, in that case, the processing of the training set would depend on the test data, this would be a form of data leakage. Data leakage means that the model uses information from outside the training set to perform the training, which can turn the further estimation of the performance invalid. Alternatively, we could choose an arbitrary constant, for example $n_{\text{extra}} = n_{\text{in}}$, but it has to be noted that this approach also is not ideal. It is always possible that photon number fluctuations lead to a test detection having more photons than $n_{\text{max}} + n_{\text{extra}}$. These simulations should then be discarded, losing all progress. For this reason, it is hard to predict a good value for n_{extra} and it is even impossible to find a value for which we are sure that no simulation will ever be excluded.

A second solution to this problem, that will be used in the remainder of the thesis, can be constructed by putting n_{extra} back to zero. This approach relies on the fact that the train set must be representative for the actual data. If the number of train detections is large enough, the occurrence of test detections with an unseen number of photons should be limited. This can be confirmed using simulations. Figure 4.4 shows the fraction of all test detections that contain a number of photons that was not previously recorded during training. The figure shows this fraction for different values of the train set size and the ensemble size. As the total number of train detections scales linearly with both the train set size and the ensemble size, we see that the train set is more representative for the actual photon number fluctuations if we increase both sizes. Hence, we see that the fraction of unexpected test detections lowers accordingly. Moreover, we see that less than 0.005% of all test detections could not be captured in $\mathrm{lossy\_basis}(n_{\text{max}}, m_A)$. As such a small fraction of the test detections should not influence the final feature values, we further choose to ignore these detections in the feature extraction process. In order to calculate the fractional occurrence of a certain detection at a given iteration, we still monitor its number of appearances and divide that number by the ensemble size, but now we adjust the denominator by subtracting the number of ignored detections at that iteration.

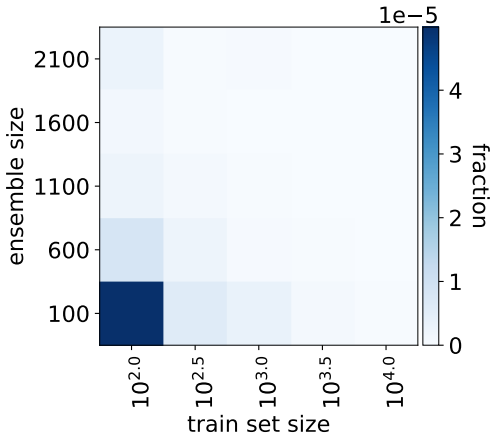


Figure 4.4: Fraction of the test detections that contain a number of photons that was not previously recorded during training. The results are shown as a function of the train set size and the ensemble size. For each combination of the train set size and the ensemble size, 10 separate reservoir intialisations were performed and the results were averaged over these initialisations. Other parameters: $m_A = m_B = 2$, $N = 6$, $n_{\text{in}} = 1$, $\phi = \pi$, test set size $= 10\,000$

Note that these observations are constricted to a specific part of the parameter space and that we do not state a priori that the number of ignored detections will always be negligible for $n_{\text{extra}} = 0$. The simulation of the system does however indicate the frequency of this behaviour and also still allows n_{extra} to be different from zero, enabling the user to tune the rate of detection omissions to a reasonable extent such that it does not influence the results. In the simulations that will be discussed in the remainder of this thesis, $n_{\text{extra}} = 0$ however proved to be a good value.

4.6.2 Convergence and fading memory

Let us consider the values that follow from this feature extraction process as a function of the ensemble size. Figure 4.5 shows these values at an arbitrarily chosen iteration of the training process (iteration 750). In this figure, the estimation of the expectation values is repeated, each time using a larger ensemble of reservoir copies. In this example, the process is repeated up to an ensemble size of 600, after which the feature values are sent to the regressor.

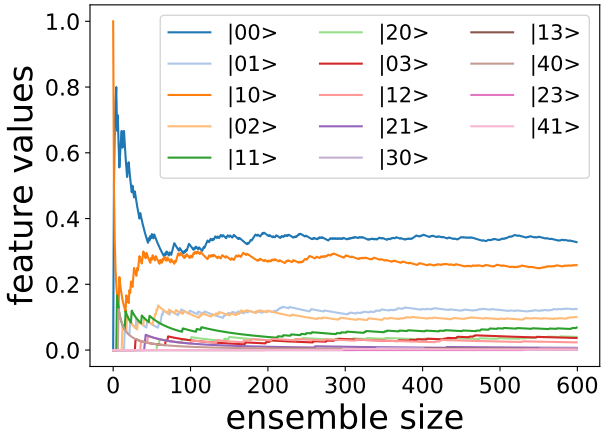


Figure 4.5: Feature values as a function of the ensemble size, at iteration 750 of the training procedure. Each line depicts the estimated expectation value of a certain detection. This figure is made from a single simulation with an ensemble size of 600, where we apply the feature extraction process of Section 4.6.1 incrementally to the first number of reservoir copies. At the end of this procedure (right on the figure), we find the feature values that serve as input for the regressor. Other parameters: $m_A = m_B = 2$, $N = 6$, $n_{\text{in}} = 1$, train set size $= 1000$, $\phi = \pi$

Knowing that the feature values converge as a function of the ensemble size, let us describe what information these values actually contain. For this purpose, we have to shift our attention back to the fading memory principle. As explained in Section 2.3, this principle also is a key concept in classical reservoir computing and it means that the influence of input information on the reservoir state becomes smaller the longer ago it was injected. This implies that the reservoir asymptotically forgets its initial conditions and that only the information that was acquired during the last iterations shapes the current reservoir state.

Moreover, this principle is a requirement for a nonlinear, dynamical system to be used in a reservoir computing approach. As was already made clear in Section 2.3, the range of systems that qualify for such an approach is however wide. Indeed, in reality, this fading memory constraint proves not to be restrictive [15] and is naturally satisfied by most systems that contain recurrent connections. In fact, the difficulty often lies in assuring that such systems have long enough memory times.

Furthermore, let us investigate the fading memory of our system. Looking back to the definition of the temporal XOR task, we see that, until now, the system only needed to combine subsequent bits in order to achieve its goal. This was shown in Table 4.1. This task can however be re-defined such that it may also require longer memory times. We define the *delayed* temporal XOR task as the temporal XOR task between the input bit of the current iteration and the input bit of D iterations ago, where D the so-called delay. The first D bits of the output sequence are arbitrarily chosen to be 1. As an example, the delayed XOR task with $D = 2$ is

shown in Table 4.2.

Table 4.2: An example of the desired output for the delayed temporal XOR task, with a delay of 2. The input bits of iteration $k-2$ and k are combined, leading to a 0-bit if both inputs are equal. Otherwise, the output is a 1-bit.

Iteration	0	1	2	3	4	5	6	7
Input bit sequence	1	1	1	0	0	1	0	1
Output bit sequence	1	1	0	1	1	1	0	0

Figure 4.6 shows the mean test BER for different values of the XOR delay. These averaged values are calculated over 10 different simulations that use a different reservoir initialisation. At low values of the delay, we see that the resulting BER increases with the delay. When the delay exceeds 3 iterations, the system always scores equally well as random guesses of the output bit values. This indeed shows that the injected information is forgotten over a number of iterations. Let us denote this number by M. For the specific configuration of this simulation, M seems to be equal to 4. However, we note that tuning the fading memory is known to be an important practice in RC [15], as it should be adapted to the task at hand. Therefore, the study of the fading memory as a function of the system parameters is also part of the future objectives of this thesis.

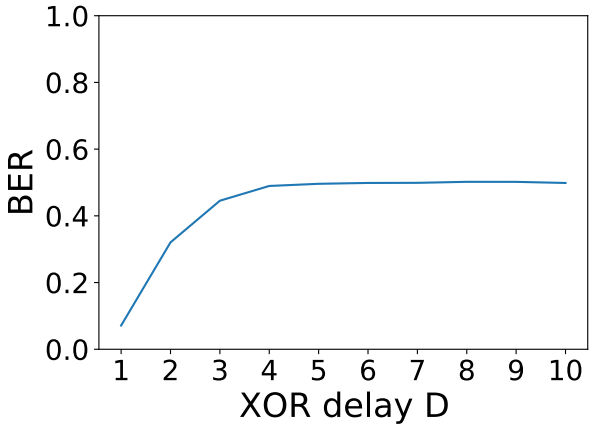


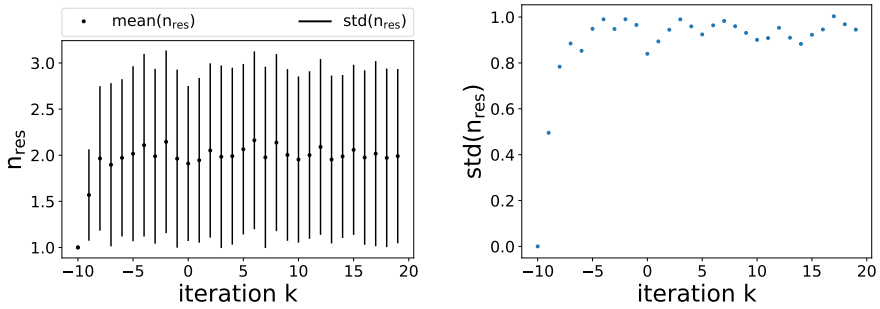
Figure 4.6: Mean test BER as a function of the delay between the input bits of the trained temporal XOR task. For each value of D, 10 separate reservoir initialisations were performed. Other parameters: $m_A = m_B = 2$, $N = 6$, $n_{\text{in}} = 1$, ensemble size $= 600$, train set size $= 1000$, test set size $= 10\,000$, $\phi = \pi$, $\alpha = 10^{-4}$ (introduced in Section 4.8)

4.7 Boot-up cycle

Before studying how our QRC model is influenced by its parameters, we make two different improvements to the system. The first of these improvements will be discussed in this section and aims to remove transient phenomena at the start of the operating procedure. These phenomena are also common in classical RC and are unwanted as they distract the regressor from learning the task at hand. As will become clear, this problem can simply be resolved by introducing a boot-up cycle.

During a boot-up cycle, random input data is fed into the system that is encoded using the same number of photons (n_{in}) that is used in the regular operating procedure. The reservoir simulation runs as usual, but the resulting detections are disregarded. Afterwards, the actual train and test data is fed into the system and the detections are stored.

Figure 4.7 shows the photon number fluctuations in the system during the first training steps, where the bootup-cycle is indicated by negative iteration numbers. In the simulations that will be discussed in the remainder of this thesis, a boot-up cycle of 10 iterations proved to be sufficient to exclude the transient phenomenon. Figure 4.7a shows the mean n_{res}, taken over an ensemble of 600 reservoirs. Equation (4.5) gives an approximation of the average value of n_{res}. As n_{in} was taken to be 1 here and $m_A = m_B = 2$, this approximation is equal to 2. In Figure 4.7a we observe that the average value of n_{res} fluctuates around this predicted value after completing the boot-up cycle. This same figure also shows the standard deviation of n_{res} using vertical bars. As could be expected, the standard deviation is zero at the first iteration as all reservoirs have $n_{\text{res}} = n_{\text{in}}$ at that point. During the subsequent iterations, the spread on n_{res} rises. This is made more clear by Figure 4.7b, where the standard deviation of n_{res} is plotted separately. We observe that the standard deviation rises during the boot-up cycle, after which it fluctuates around a constant value.

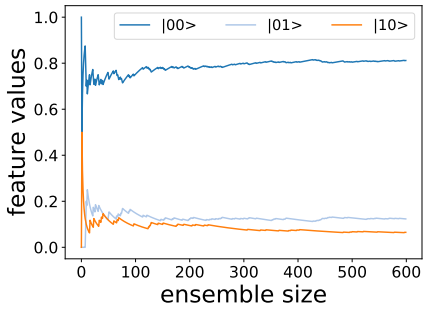


(a) mean and standard deviation

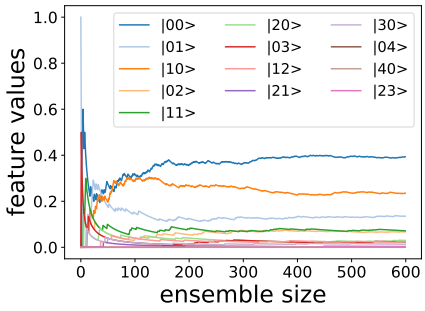
(b) standard deviation

Figure 4.7: Visualisation of n_{res} in an ensemble of 600 reservoir copies during first training steps. (a) shows the averaged value of n_{res} over the ensemble (dots) and the associated standard deviation (bars). (b) shows the standard deviation of n_{res} separately. Other parameters: $m_A = m_B = 2$, $N = 6$, $n_{\text{in}} = 1$, $\phi = \pi$

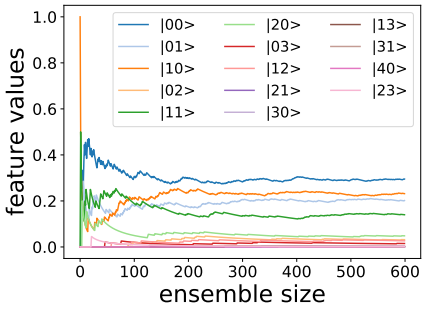
Let us now illustrate why the transient behaviour would limit the system if the boot-up cycle was not added. Figure 4.8 shows the convergence of the features as a function of the ensemble size at three different iterations. At iteration -10 we see, in accordance with Figure 4.7a, that the number of photons that can be detected is restricted to 1. At iteration 0, we see that the number of different states that were detected is considerably larger. In the subsequent iterations, as can be seen for example at iteration 500, the range of different detections remains approximately the same. We conclude that, during the boot-up cycle, the system results in deviating feature values with respect to later iterations and that these values need to be removed.



(a) Iteration -10



(b) Iteration 0



(c) Iteration 500

Figure 4.8: Feature values as a function of the ensemble size, at different iterations of the training procedure. Other parameters: $m_A = m_B = 2$, $N = 6$, $n_{\text{in}} = 1$, train set size $= 1000$, $\phi = \pi$

4.8 Regularization

The second improvement that we make to the system aims to remove its sensitivity to small feature changes. In this section, we describe how such changes can be induced and how they can be countered by rounding the feature values before

sending them to the linear regressor. Afterwards, a more general and common solution to these types of problems is introduced: regularization.

As explained earlier, one of the advantages of reservoir computing is that it only requires a simple linear regression algorithm to optimize the parameters of the readout function. During development, this readout function however showed to be sensitive to small changes in the input features of the regressor. These changes could for example be induced during the feature extraction procedure, when excluding test detections that have more than $n_{\text{max}} + n_{\text{extra}}$ photons. As was explained in Section 4.6.1, the fraction of test detections that are excluded should be small, such that they can be ignored. Note that choosing a higher value for n_{extra} will only change the feature values if states with $n_{\text{max}} + n_{\text{extra}}$ photons actually will be detected. If not, only zeros are added to the feature matrix with respect to a simulation with a lower value of n_{extra}.

As the fraction of exclusions is assumed to be small, this behaviour should only have a small influence on the feature matrix as well. This statement could be confirmed by directly comparing feature values of simulations with and without exclusion of detections, using different values n_{extra}. Note that these simulations were carried out with equally seeded random generators such that both simulations, apart from having equal QONN phase parameters, also have equal sampling events at the detector. As a result, the observed feature changes were solely due to the exclusion of test detections.

Using a linear regressor, these small feature changes did however result in noticeable differences in the train and test errors. Further analysis showed that even smaller feature changes could also influence the performance of the system. A first possible solution to this problem is to round the features to a certain number of decimals before sending them to the regressor.

Figure 4.9 shows the change in train and test BER that is induced by adding noise to the feature matrices, both after the feature extraction of the training procedure and the testing procedure. The noise is uniformly distributed over $[0, 10^{-10})$. 50 different reservoir initialisations are performed and for each of those initialisations, 4 optimization processes were performed, either adding noise or not and either rounding the features to 5 decimals or not. The boxplots show the BER differences for the different initialisations. Note that the whiskers of the boxplots stretch to the highest and lowest calculated differences, so no outliers are shown. Also note that the difference in BER is an actual difference, rather than a relative difference.

We observe that by rounding the features, the train and test BER are not influenced by adding noise. This behaviour is expected as in both cases (with and without noise) we end up with the same feature values. However, if we do not round the feature values, the train and test BER are influenced. This behaviour should not occur, as there is no valuable information encoded in the small feature changes.

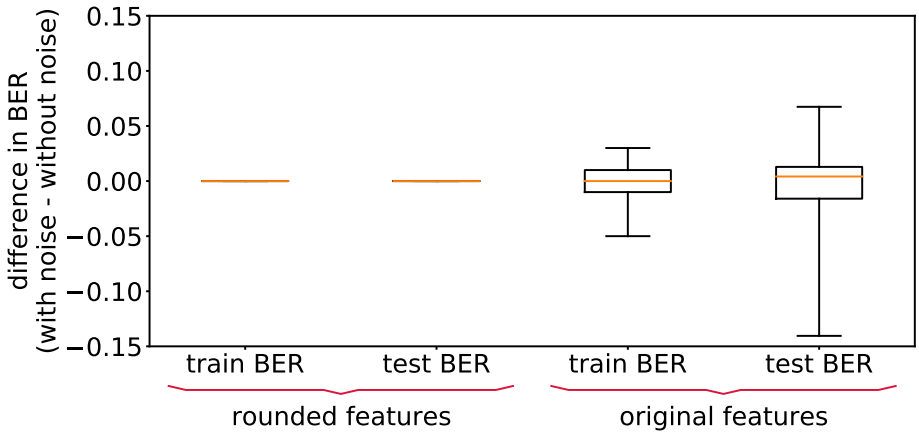


Figure 4.9: The change in train and test BER that is induced by adding noise to the feature values. The boxplots show the BER differences for 50 different reservoir initialisations. The whiskers of the boxplots stretch to the highest and lowest calculated differences, so no outliers are shown. The noise is uniformly distributed over $[0, 10^{-10})$. The first two boxplots were obtained by rounding all feature values to 5 decimals before sending them to the regressor. The last two boxplots were obtained without rounding the feature values. Other parameters: $m_A = m_B = 2$, $N = 6$, $n_{\text{in}} = 1$, ensemble size $= 600$, train set size $= 100$, test set size $= 10\,000$, $\phi = \pi$

This shows that the rounding approach works as expected, but as the number of remaining decimal places is chosen arbitrarily, it needs to be noted that this is not a generally applicable solution. The rounding procedure did however serve as a valuable reference point when comparing it with the next approach: regularization.

Regularization is the act of adding extra penalty terms to the cost function of an optimization problem in order to prevent that the resulting model is tuned too closely to its given training data set. In other words, regularization prevents overfitting [64], such that the model is better at generalizing to previously unseen data. A type of regularization that is often used for linear regressors, is ridge regression. Starting from Equation (2.2), the cost function is altered as follows:

$$||\boldsymbol{y} - \boldsymbol{X} \cdot \boldsymbol{w}||_2^2 \; + \; \alpha \, ||\boldsymbol{w}||_2^2 \tag{4.11}$$

Here, α is a new optimizable parameter, also called the regularization parameter. During training, the extra term stimulates the regressor to reduce the size of the weights. The reason for this is that very large weights may indicate overfitting behaviour, as those weights draw excessive attention to a select number of features. In Section 5.1.2 the influence of α on the performance of the system will be analysed. Anticipating the results of that section, we mention that for low values of α, regularization yields similar results as the rounding method.

Chapter 5

Results and discussion

In this chapter we will make use of the model that was constructed in Chapter 4. First, in Section 5.1, we will look at the influence of different parameters of the system with the purpose of better understanding its behaviour. Afterwards, in Section 5.2, a discussion will ensue on the role of the repeated detections that are performed during operation of the system. This discussion will highlight different characteristics of our current model and will enable us to further improve the simulation of this model.

5.1 Study of the system parameters

Before discussing the different parameters of the system, we need to note that we currently do not optimize hyperparameters. Hyperparameters are parameters of the system that are used to *control* the training process, rather than being *derived* during training. As machine learning approaches often do optimize these parameters, it is common practice to subdivide the available data set not only in a train and test set, but also in a validation set. This validation set is used to validate the performance of the system for different hyperparameter values and to determine an optimal set of those values. However, since the hyperparameters are derived using the validation data, we need to use a third, unseen data set, the test set, to make an unbiased evaluation of a final model.

At the moment, however, we do not optimize any hyperparameters of our current model and only distinguish between a train and test set. The reason for this is that we do not focus on finding the optimal behaviour of our system, but we rather want to show that the current model can be trained to solve a simple task. Further, we want to get an understanding of the properties of this model and of where they originate from.

Also note that we only perform the temporal XOR task with $D = 1$ in the remainder of this section. Further analysis is needed to verify whether the results in this section can be generalized to more challenging tasks.

The simulations that are discussed below were acquired using a test set size of

10 000. This enables us to safely present the resulting test BER up to an accuracy of 1%. This follows from the rule of thumb that we need to use at least $10/R$ bits to present a certain resolution R [65]. Note that the size of the train set, as opposed to the test set, will be changed in the analysis below.

All calculations were performed on the high performance computing infrastructure of the UGent, enabling the parallelization of simulations with different reservoir initialisations. Note that due to the ensemble approach in principle also the simulations of the reservoir copies could be parallelized. This would however only drastically speed up the process if the total number of reservoir initialisations would be low. At this moment, the parallelization of the reservoir copies itself is not implemented yet.

5.1.1 Train set size and ensemble size

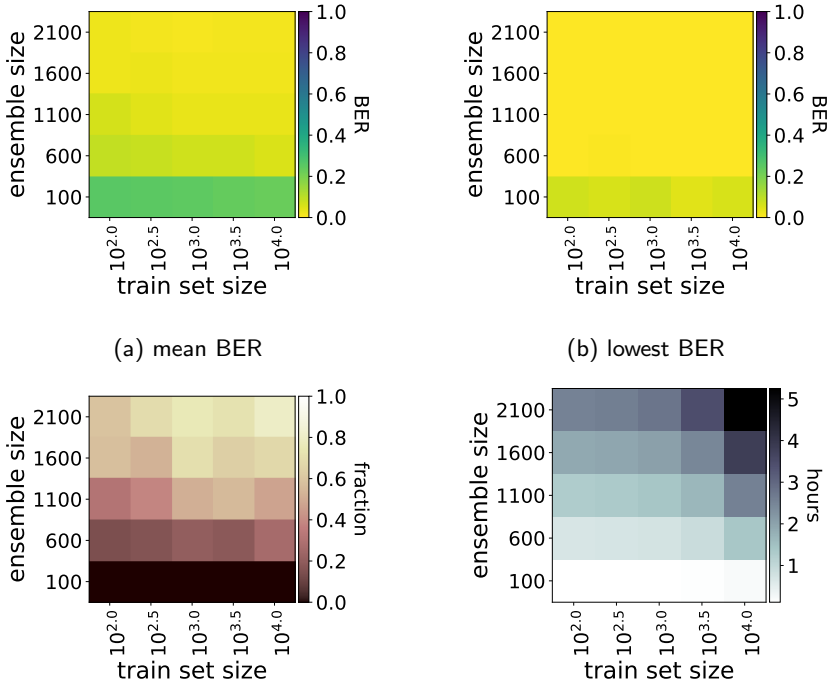
Figure 5.1 shows simulation results for different values of the train set size and the ensemble size. For every combination of these two sizes, 50 simulations are performed, each time using a different reservoir initialisation. Figures 5.1a and 5.1b respectively show the mean test BER and the lowest test BER, calculated over the different initialisations. Figure 5.1c shows the fraction of reservoir initialisations that resulted in a test BER lower than an arbitrary value of 2%. Remember that a BER of 0% corresponds with a perfectly trained system, while a BER of 50% means that the system scores equally well as random guesses of the output. Finally, Figure 5.1d shows the average simulation time, also calculated over the different initialisations.

We could expect that both a larger train set size and a larger ensemble size would enhance the performance of the system. The reason for this is that more data is generally better for training machine learning models [66], while a bigger ensemble succeeds in finding more accurate estimations of the expectation values. In Figures 5.1a, 5.1b and 5.1c, the expected relation with respect to the ensemble size can definitely be observed, but the dependency of the test BER on the train set size is less pronounced. However, it has to be noted that these observations are task-dependent and that more data could be required to perform more challenging tasks.

The increase in simulation time of Figure 5.1d results from the fact that the total number of reservoir iterations scales linearly with the the train set size and the ensemble size. As the test set size is fixed to a value of 10 000, the simulation time of the reservoir operation during testing is approximately equal for all simulations. Also note that the execution of Ryser's algorithm requires approximately the same time for all simulations as we use the memoization technique that was described in Section 4.3.

We conclude that for the current model and task, a train set size larger than 1000 does not improve the performance by much. For train set sizes that are noticeably smaller than the test set size of 10 000, say from 100 up to 1000, we observe in Figure 5.1d that the simulation time is approximately the same. A larger ensemble size seems to have more influence on the performance of the system, but this also raises the simulation time. In a hardware implementation, more reservoir copies

(for example by performing the same experiment multiple times) will also come at a cost, leading us to a trade-off when choosing the size of the ensemble.



(a) mean BER

(b) lowest BER

(c) fraction of simulations with BER $< 2\%$

(d) mean simulation time

Figure 5.1: Test BER and simulation time for different train set size and ensemble size. For each combination of the train set size and the ensemble size, 50 separate reservoir initialisations were performed. Other parameters: $m_A = m_B = 2$, $N = 6$, $n_{\text{in}} = 1$, $\alpha = 10^{-4}$, $\phi = \pi$, test set size $= 10\,000$

5.1.2 Regularization parameter α

In this section, the influence of the regularization parameter α on the performance of the system will be analysed. For each of the values of the train set size and the ensemble size in Section 5.1.1, 10 reservoir initialisations are performed. The results for a few of these initialisations are shown in Figure 5.2, in order to discuss the main trends that we observe. The orange and red lines respectively show the train and test BER of the system, where we used a ridge regressor for different values of α. These results are compared with a system that does not perform any regularization procedure, but instead rounds the input features to 5 decimals before sending them to a linear regressor. The train and test BER of this last regressor are respectively depicted in green and blue.

Let us first consider Figure 5.2a. This is a result of a reservoir initialisation that

is used to process a train set of size 10 000. The ensemble size is 2100. We see that the train and test error approximately coincide, both for the ridge regressor as well as for the linear regressor that uses rounded features. For the ridge regressor, we see a rise in error as α exceeds 10^{-2}. Although for high values of test set size and ensemble size, this rise can be small and right-shifted (i.e. only occur at high values of α), it will always be present. This can be understood from Equation (4.11). When α is chosen too large, the regularization term in this cost function overpowers the importance of the actual targets during training.

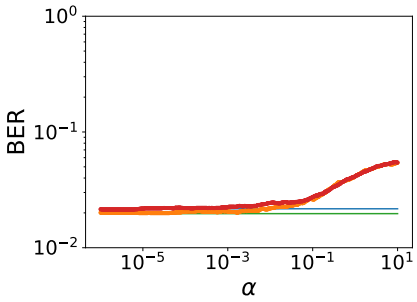
If we keep the training set at 10 000, but lower the ensemble size, we know from Figure 5.1a that the average test BER, calulated over different reservoir initialisations, rises. Consequently, we see in Figure 5.2b that both the train and test BER are higher when using an ensemble size of 100. Remember that this figure is the result of a single reservoir initialisation, but on average the same conclusions can be drawn for other initialisations. Also note that this figure is an example of where the rise in error at high α values is right-shifted.

Lowering the train set size to 3162 ($\approx 10^{3.5}$) and 1000, regardless of the ensemble size, we still see that the train and test errors coincide, both for the ridge regressor as for the approach with rounded features. An example for a train set size of 1000 and an ensemble size of 1100 is shown in Figure 5.2c. In some cases, the rise in error at high α becomes left-shifted and can reach higher error values.

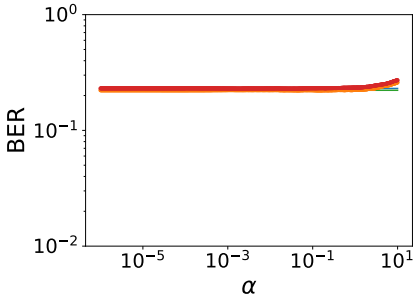
Lowering the train set size further to 316 and 100, the same trends are still observed, but more cases arise where the train and test errors are substantially different. Two such examples are shown in Figure 5.2d and Figure 5.2e. Since the gap between train and test error is a measure for how much the model overfits, we see that the smaller train set causes more overfitting. Moreover, in some of these cases (of which two are shown in Figure 5.2d and Figure 5.2e), the ridge regressor shows to improve on the approach with rounded features for a certain range of α values.

In general, we can conclude that high α values (higher than 10^{-2}) should be avoided such that the regularization term in Equation (4.11) does not overpower the training term. If the train set is relatively large, a small α value (smaller than 10^{-2}) suffices to score equally well as the approach with rounded features. In that case, a secure optimization of the α parameter does not add much value to the system. At relatively low train set sizes, without making conclusions about optimal α values, we however need to note that there is more room for improvement. In this part of the parameter space, the model would benefit from a secure optimization process where data is split in a train, validation and test set, as was explained in the introduction of Section 5.1.

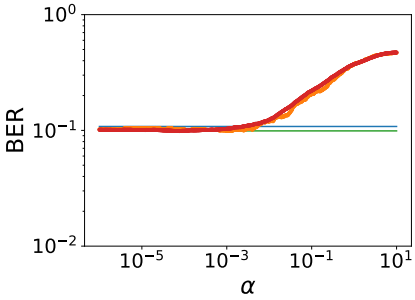
Note that regularization is a more general and commonly used method than rounding the features. For unexplored parts of the parameter space and for other tasks, regularisation may turn out to be more important than it appears to be here. In the simulations that will be discussed in the remainder of this thesis, we use ridge regularisation with $\alpha = 10^{-4}$, which for a train set size of 1000 gives similar results as the rounding method.



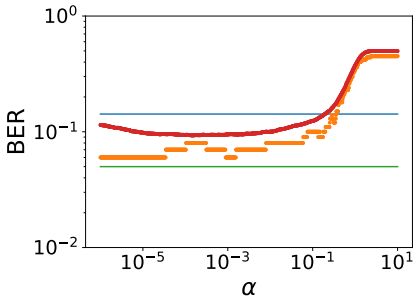
(a) train set size: 10 000
ensemble size: 2100



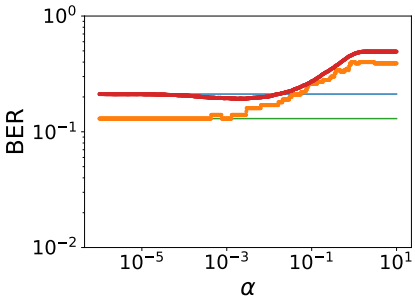
(b) train set size: 10 000
ensemble size: 100



(c) train set size: 1000
ensemble size: 1100



(d) train set size: 100
ensemble size: 600



(e) train set size: 100
ensemble size: 100

Figure 5.2: BER as a function of α, for different values of the train set size and the ensemble size. The train BER and the test BER of the ridge regressor are respectively colored orange and red. The train BER and the test BER of the linear regressor with rounded features are respectively colored green and blue. Other parameters: $m_A = m_B = 2$, $N = 6$, $n_{\text{in}} = 1$, $\phi = \pi$, test set size $= 10\,000$

5.1.3 Kerr interaction strength ϕ

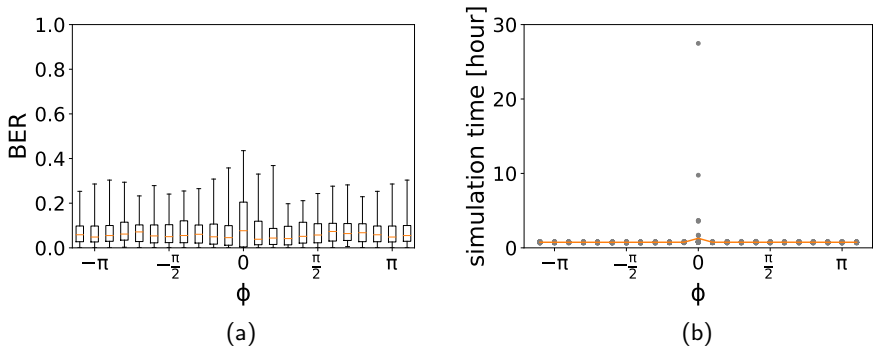


Figure 5.3: The test BER (a) and simulation time (b) of 100 different reservoir initialisations as a function of ϕ. The whiskers of the boxplots in (a) stretch to the highest and lowest value that were observed at each value of ϕ, so no outliers are shown. In (b) the dots depict simulation times of different reservoir initialisations, while the orange line depict the average value over different initialisations that have the same value of ϕ. Other parameters: $m_A = m_B = 2$, $N = 6$, $n_{\text{in}} = 1$, ensemble size $= 600$, train set size $= 1000$, test set size $= 10\,000$, $\alpha = 10^{-4}$

In this section, we perform simulations for different values of ϕ, which is the Kerr interaction strength as was defined in Equation (3.3). To be more specific, for each value of ϕ that we consider, we perform 100 reservoir initialisations with an ensemble size of 600. Figure 5.3a shows the test BER as a function of ϕ. Note that the whiskers of the boxplots stretch to the highest and lowest value that were observed at each value of ϕ, so no outliers are shown. The grey dots depict the separate values, while the blue line depicts the average value over those different reservoir initialisations. Figure 5.3b shows the simulation time of the different reservoir initialisations as grey dots, while their average value over 100 initialisations is depicted by the orange line. In both figures, we see that the behaviour of the system is different at $\phi = 0$. As is clear from Equation (3.3), $\phi = 0$ corresponds with the situation where there are no single-site nonlinearities in the QONN.

Note that most of the dots in Figure 5.3b coincide. This figure is alternatively plotted using a cumulative distribution function in Figure B.3a of Appendix B. However, for now, we do not focus on the actual distribution of the different values at the same value of ϕ. Instead, we want to stress the significantly higher spread of values for different initialisations at $\phi = 0$. Without nonlinearities in the QONN, the initialisation-dependence of both the test BER as the simulation time seems to be stronger.

Moreover, the removal of the nonlinearities also appears to influence the fluctuations of the photon number of the reservoir state (n_{res}), as is depicted in Figure 5.4. In Figure 5.4a and Figure 5.4b we respectively see the average value of n_{res} and the size of its fluctuations (quantified by the standard deviation of n_{res}), calcu-

lated over the complete training procedure. Again, boxplots are used to show how mean(n_{res}) and std(n_{res}) vary for different reservoir initialisations and we see that the behaviour at $\phi = 0$ stands out. We notice that mean(n_{res}) is remarkably less initialisation-dependent at $\phi = 0$, while the corresponding values of std($\phi = 0$) are more initialisation-dependent. In Figure 5.4c we see the maximum value of n_{res} that occurred during training. As these values are integers, different reservoir initialisations are grouped into a single dot if they coincide on the figure. The area of the dots is proportional to the number of reservoir initialisations that they represent. Again, we see an increased initialisation-dependence at $\phi = 0$. max(n_{res}) is interesting, as high values for n_{res} lead to more time-consuming boson sampling procedures and matrix multiplications. This therefore explains the outlier values with long simulation times in Figure 5.3b.

We conclude that the single-site nonlinearities in the QONN influence the photon number fluctuations and the initialisation dependence of the test BER and the simulation time. Although the underlying reason for this is unclear at the moment, we remark that this behaviour is unwanted as we want to initialise the QONN randomly.

What is also unclear at the moment, is whether there is a direct link between the photon number fluctuations and the performance of the system. If such a link were to exist, it could give us more insight in the behaviour of our system and possibly show us ways to boost the performance. Consequently, it is required to perform a more thorough analysis of photon number distribution as a function of the different system parameters.

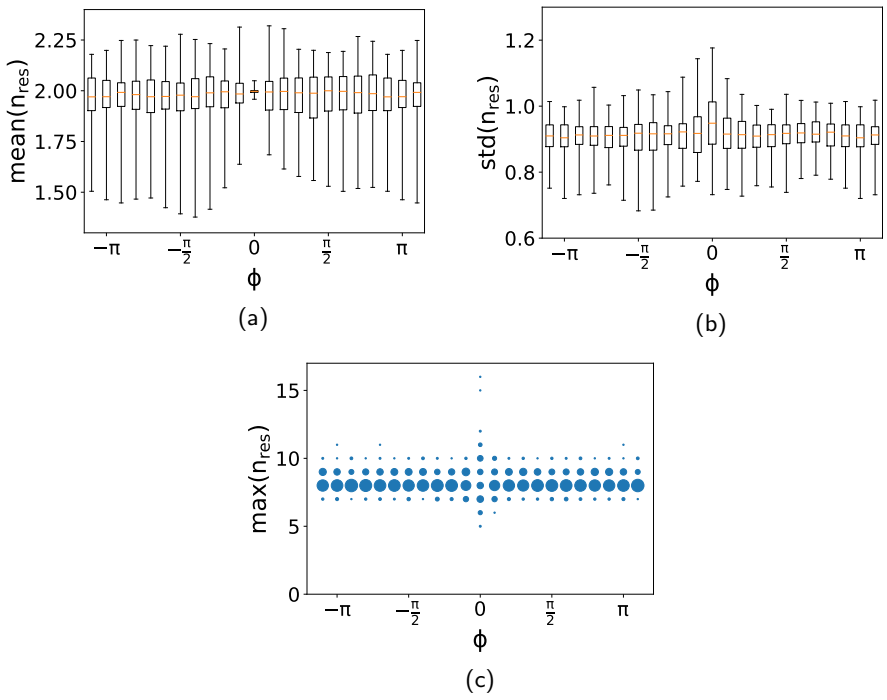


Figure 5.4: The mean (a), standard deviation (b) and maximum (c) of the number of photons in the reservoir state, calculated over all reservoir copies and over all training iterations. The results for 100 different reservoir initialisations are plotted as a function of ϕ. The whiskers of the boxplots in (a) and (b) stretch to the highest and lowest value that were observed at each value of ϕ, so no outliers are shown. In (c) the results of different reservoir initialisations are grouped in a single dot if they lead to the same value for max(n_{res}). The area of the dots is proportional to the number of reservoir initialisations that they represent. Other parameters: $m_A = m_B = 2$, $N = 6$, $n_{\text{in}} = 1$, ensemble size $= 600$, train set size $= 1000$, test set size $= 10\,000$, $\alpha = 10^{-4}$

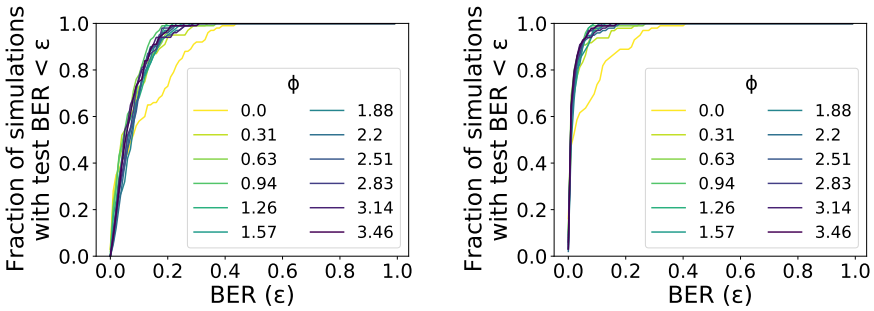
Let us now also take a more careful look at the distribution of the test BER for different reservoir initialisations. In Figure 5.3a it seems like the overall initialisation-dependence, regardless of the value of ϕ, is rather high. In order to study this better, we look at Figure 5.5a, where a cumulative distribution function is given as an alternative, more accurate representation of the boxplots in Figure 5.3a, using the same ensemble of 600 reservoir copies. As the results are similar for positive and negative ϕ values, only positive ones are depicted. For each BER value ϵ on the x-axis, the fraction of reservoir initialisations is plotted that leads to a test BER lower than ϵ.

In classical RC systems, after fully optimizing the system (not the case yet in this thesis), typically 95% to 99% of all initialisations lead to a near perfect

test BER ($\approx 0\%$) [67]. In Figure 5.5a, the initialisation dependence however is noticeably stronger. At best (for $\phi = 0.94$), ϵ only lies at 16% if we consider the best performing 95% of all initialisations.

Figure 5.5b shows the same cumulative distribution function, where all parameters are unchanged, except for the ensemble size, which is put at 2100. For this larger ensemble, we observe similar behaviour as in Figure 5.3 and Figure 5.4. We see, in accordance with Figure 5.1c, that the initialisation-dependency weakens by increasing the ensemble size. For example, here we find that at best (again for $\phi = 0.94$) 95% of all reservoirs result in a test BER that is lower than 6%.

Note again that the system is not fully optimized at this point and that further analysis will need to performed in order to derive whether the initialisation-dependence can be reduced to equal standards as in classical systems or whether this dependency is an an intrinsic feature of this system.



(a) ensemble size: 600

(b) ensemble size: 2100

Figure 5.5: Cumulative distribution function (CDF) of the fraction of simulations (with different reservoir initialisation) below a certain BER value. For each BER value ϵ on the x-axis, the fraction of simulations is plotted that leads to a test BER lower than ϵ. Other parameters: $m_A = m_B = 2$, $N = 6$, $n_{\text{in}} = 1$, train set size $= 1000$, test set size $= 10\,000$, $\alpha = 10^{-4}$

5.1.4 Number of QONN layers N

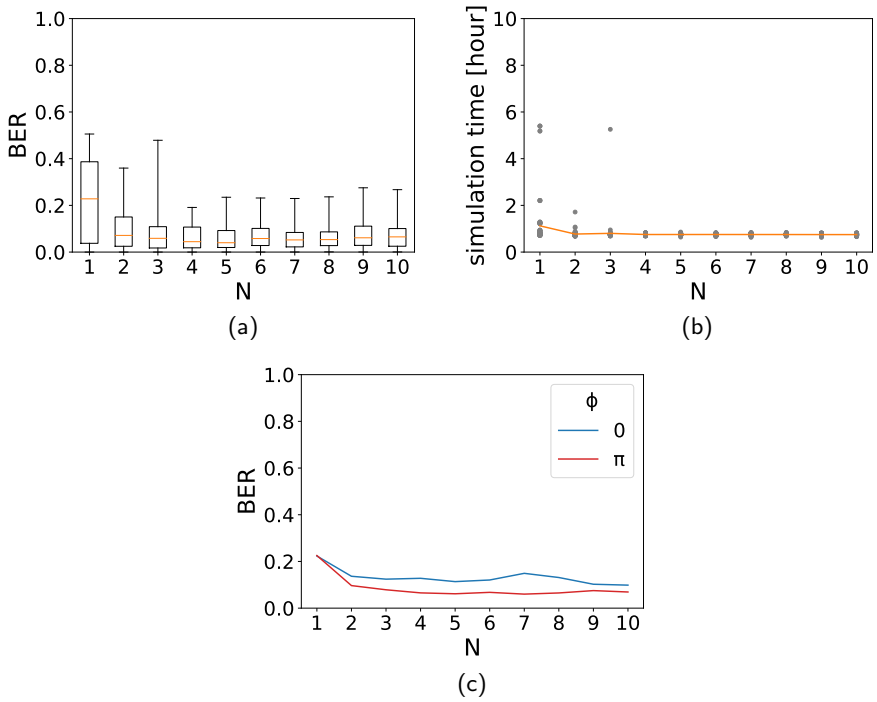


Figure 5.6: The test BER (a,c) and simulation time (b) of 100 different reservoir initialisations as a function of N. For (a) and (b), $\phi = \pi$. The whiskers of the boxplots in (a) stretch to the highest and lowest value that were observed at each value of N, so no outliers are shown. In (b) the dots depict simulation times of different reservoir initialisations, while the orange line depict the average value over different initialisations that have the same value of N. In (c) the mean test BER of (a) is compared with the mean test BER of simulations that are performed similar to (a), but where ϕ is put to zero. Other parameters: $m_A = m_B = 2$, $n_{\text{in}} = 1$, ensemble size $= 600$, train set size $= 1000$, test set size $= 10\,000$, $\alpha = 10^{-4}$

Let us again put ϕ equal to π and see how the system behaves for a different numbers of layers in the QONN (N). Figure 5.6 shows the test BER and the simulation time as a function of N, for 100 different reservoir initialisations. Similar to Figure 5.3b, for each value of N the test BERs of different initialisations are presented in a boxplot, while the simulation time for different initialisations is given by grey dots. The orange line again shows the average simulation time. An alternative representation of the simulation time, using a CDF, is shown in Figure B.3b of Appendix B. Note that the range of values on the y-axis of Figure 5.6b is made smaller than was the case in Figure 5.3b in order to differentiate more easily between different outliers.

Figure 5.6c shows the mean of the boxplots in Figure 5.6a and compares them

with the mean test BER of a similar set of simulations where the only difference is that $\phi = 0$

We see that the initialisation-dependence of the results is stronger at $N = 1$. This can be associated with the discussion in Section 5.1.3, as the QONN configuration is defined by Equation (3.4), where each layer is composed of a linear interferometer and a nonlinear Kerr layer. As single-site nonlinearities are only added in between interferometers, the number of times we perform a single-site nonlinearity on all of the modes is equal to $N - 1$. Consequently, for $N = 1$ there are no nonlinearities and the situation is similar to the one at $\phi = 0$ in Figure 5.3b. Consequently, the results in Figure 5.6c coincide at $N = 1$.

The photon number fluctuations that correspond with Figure 5.6 are depicted in Figure 5.7, following the same conventions as Figure 5.4. Again, we see that the removal of the nonlinearities from the QONN leads to stronger initialisation-dependence of the test BER and the simulation time, while having a weaker initialisation-dependence in mean(n_{res}) and a stronger initialisation-dependence in std(n_{res}) and max(n_{res}).

Additionally, note that for $\phi = 0$ in Figure 5.6c the mean test BER also is higher at $N = 1$. As was explained in [61], a possible explanation for this would be that a QONN can perform more complex transformations if it possesses more layers. We conclude that the increase in mean test BER at $N = 1$ for $\phi = \pi$ in Figure 5.6c is due to two superimposed effects and that N should be chosen large enough to avoid them both. Knowing this, we further expect that more difficult tasks would require increasingly large values of N.

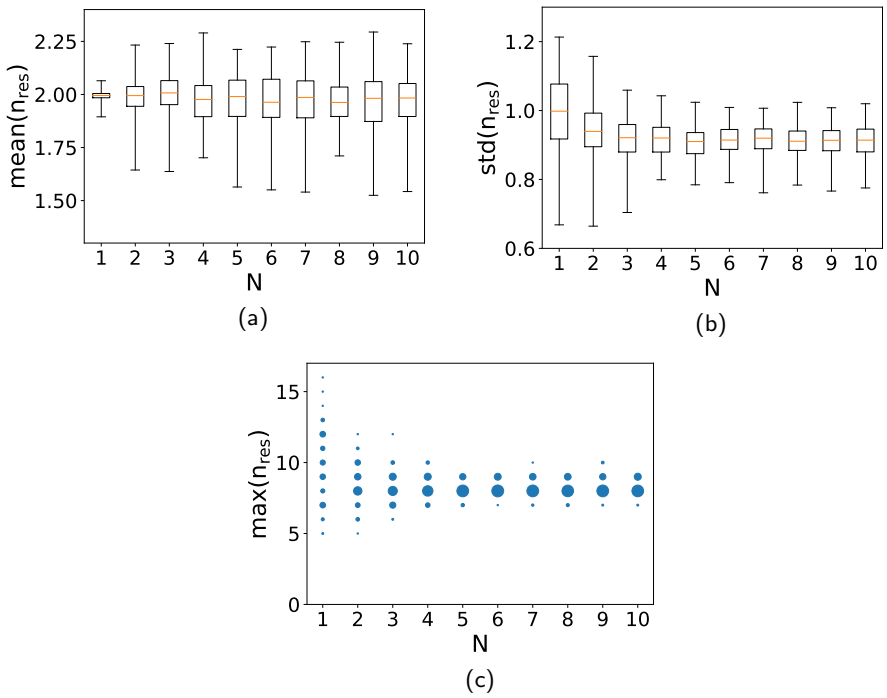


Figure 5.7: The mean (a), standard deviation (b) and maximum (c) of the number of photons in the reservoir state, calculated over all reservoir copies and over all training iterations. The results for 100 different reservoir initialisations are plotted as a function of N. The whiskers of the boxplots in (a) and (b) stretch to the highest and lowest value that were observed at each value of N, so no outliers are shown. In (c) the results of different reservoir initialisations are grouped in a single dot if they lead to the same value for max(n_{res}). The area of the dots is proportional to the number of reservoir initialisations that they represent. $m_A = m_B = 2$, $n_{\text{in}} = 1$, ensemble size $= 600$, train set size $= 1000$, test set size $= 10\,000$, $\alpha = 10^{-4}$

5.2 Collapse as part of system dynamics

As is clear from the previous sections, wavefunction collapse is an important part of the reservoir dynamics. However, these repetitive and probabilistic measurements form a particularly unconventional mechanism in the context of classical reservoir computing. Here in our quantum case, the measurements mainly have two consequences. The first one, as was explained in Section 4.3, is that n_{res} is exactly known at each iteration, allowing us to simplify the boson sampling procedure. The second one, stemming from the probabilistic nature of these detections, is that exact duplicates of a reservoir give rise to different results when the same input data is fed into them, which led us to an ensemble approach.

Until now, we have simulated such a measurement at the end of every iteration.

In this section, however, we will consider a modified version of our model where we remove all detections, except from the last one. We call these detections the intermediate detections. At first glance, it seems that by doing so there is no longer a way to extract information at each iteration. However, the opposite is true. In the remainder of this section, we first discuss a procedure that does manage to do this and that was previously introduced in Ref. [39]. Although it will soon become clear that this procedure does not contribute to the efficiency of a hardware implementation of our system, its discussion will show us a number of characteristics that are typical for the platform that we use. Moreover, after a mathematical comparison of the procedure with and without intermediate detections, we will be able to deduce how we can improve our current simulation and how a hardware implementation without intermediate detections can serve as a useful research tool.

5.2.1 Procedure without intermediate detections

Assume that we remove all intermediate detections from the procedure and only detect at the very end. A way to still extract information from the system at all iterations is to repeatedly restart the simulations, and letting the system each time run for a longer number of iterations before making the final detection and stopping the system. In other words, we perform as many reservoir initialisations as there were iterations in the previous procedure and we always postpone the measurement until the end of the experiment. Note that, as we eventually still perform a measurement, we still need to prepare an ensemble of reservoir copies.

In Ref. [39], this procedure was performed using superconducting qubits on an IBM quantum processor [40]. It goes without saying that it is a lengthy procedure which limits the capacity of the system. In the paper, this approach was however followed because of hardware restrictions. More specifically, the IBM quantum processor did (at that time [68]) not allow for 'qubit reset', which is the action of re-using a qubit after it is measured. Instead, all measurements on this processor were supposed to be performed at the end of each experiment. In reservoir computing, qubit reset can for instance be desired in order to re-use qubits that were detected at a certain iteration to encode input data of the subsequent iteration. By doing so, all qubits of the system are utilised during every iteration. Similar to the IBM quantum processor, some other NISQ devices also do not support qubit reset, limiting the number of measurements that can be performed and therefore also limiting the optimal use of the resources when performing QRC.

Although the proposed procedure is lengthy, Ref. [39] also proposes an important improvement to simplify it, based on the so-called 'convergence property'. In essence, this convergence property is nothing more than the fading memory principle, meaning that the initial conditions of the reservoir are forgotten over M iterations. However, as we do not detect intermediately here, reservoirs with different initial conditions now lead to approximately equal reservoir states after M iterations, assuming that the same input data is fed into them during that period of time. As a result, the operating procedure does not always need to be restarted from iteration 1. Instead, it can be restarted M iterations before the iteration at which the measurement is performed, making use of an arbitrary initial reservoir

state.

Furthermore, Ref. [39] emphasises that intermediate detections, which could not be used as qubit reset could not be performed, would greatly reduce the total number of reservoir operations and therefore boost the efficiency. Consequently, also in this thesis a hardware implementation without intermediate detections will not boost the performance of our system. However, by removing the intermediate detections, it will turn out that its simulation can be made more efficient.

5.2.2 Comparison of both procedures: with and without intermediate detections

Density operators and the partial trace

Let us return our attention to our current set-up. Before we can give a mathematical description of this system, both with and without intermediate detections, we are required to introduce two concepts: the density matrix and the partial trace. In order to introduce these concepts in a simple way and to introduce some notations that we will use in the remainder of this section, we first briefly look back to the detection procedure that was presented in Section 4.5. In that section, we assumed the following expression for the reservoir state:

$$\begin{aligned}|\psi_{\text{res}}\rangle =& a_1 |2000\rangle + a_2 |1100\rangle + a_3 |1010\rangle + a_4 |1001\rangle + a_5 |0200\rangle \\ &+ a_6 |0110\rangle + a_7 |0101\rangle + a_8 |0020\rangle + a_9 |0011\rangle + a_{10} |0002\rangle \end{aligned} \tag{5.1}$$

Assume that this state is the reservoir state in the first iteration of the process. Following the example in Section 4.5, we detect $\mu = |00\rangle_A$ with a probability of:

$$P(\mu) = |a_8|^2 + |a_9|^2 + |a_{10}|^2 \tag{5.2}$$

which leads to the following feedback state:

$$|\psi_{\text{fb}}^{(00)}\rangle = \frac{1}{\sqrt{|a_8|^2 + |a_9|^2 + |a_{10}|^2}} \Big(a_8 |20\rangle_B + a_9 |11\rangle_B + a_{10} |02\rangle_B \Big) \tag{5.3}$$

Here, we have introduced the notation $|\psi_{\text{fb}}^{(\mu)}\rangle = |\psi_{\text{fb}}^{(n_1 n_2)}\rangle = |\psi_{\text{fb}}^{(n_1 n_2)}\rangle_B$ to describe the remaining feedback state after detecting $\mu = |n_1 n_2\rangle_A$. Note that we drop the subscript B for simplicity and that all Fock terms in $|\psi_{\text{fb}}^{(\mu)}\rangle$ contain an equal number of photons, as was discussed in Section 4.3.

It also has to be noted that this process could equivalently be described using density matrices. A density matrix is a generalized representation of a quantum state that is often used in quantum computing. While state vectors can only describe so-called 'pure states', density matrices can also describe 'mixed states'. Mixed states can arise when the preparation of the system is not fully known, leading to a so-called statistical ensemble of possible preparations. As a result, the density operator of a pure state can be written as an outer product of a single state vector $\boldsymbol{\rho} = |\psi\rangle \langle\psi|$, while a mixed state is described by a weighted average of such outer

products $\boldsymbol{\rho} = \sum_j p_j \left|\psi_j\right\rangle \left\langle\psi_j\right|$. Here, ψ_j is prepared with a probability p_j, as defined by the statistical ensemble. As will become even more clear in the mathematical description of the next section, this 'statistical ensemble of possible preparations' is closely related to the ensemble of reservoir copies that we introduced in Section 4.1 in order to account for the different possible measurement outcomes at the detector.

While the density operator is an alternative mathematical representation that can be used to account for multiple preparations of the same system, it generally also requires us to simulate the detection procedure. As we already know from Section 4.5, if we perform intermediate detections, we can continue to use the state vector representation if we want to. This shows that we are working with pure states, as the state vector only gives rise to a single outer product in the density operator.

Let us consider how the description of the detection procedure is performed in the density operator formalism [3][69]. As was the case in Section 4.5, we first measure a certain state μ and then calculate the remaining feedback state. It turns out this is described by the following equation:

$$P(\mu) = tr\big[\boldsymbol{\rho}_{\text{res}}\boldsymbol{O}\big] = tr\big[\boldsymbol{\rho}_{\text{res}}\left|\mu\right\rangle\left\langle\mu\right|\big] \tag{5.4}$$

Here, $\boldsymbol{\rho}_{\text{res}} = \left|\psi_{\text{res}}\right\rangle\left\langle\psi_{\text{res}}\right|$ if we consider the pure state from Equation (5.1) and $\boldsymbol{O} = \left|\mu\right\rangle\left\langle\mu\right|$ is a so-called measurement operator. $tr[\boldsymbol{\rho}] = tr_{AB}[\boldsymbol{\rho}] = \sum_i \boldsymbol{\rho}_{ii}$ is called the trace of $\boldsymbol{\rho}$ and sums all its diagonal elements. Note that the subscript AB can be added to stress the fact that the trace is performed on all A and B modes here. The remaining feedback state turns out to be given by:

$$\boldsymbol{\rho}_{\text{fb}} = \frac{\boldsymbol{O}\boldsymbol{\rho}_{\text{res}}\boldsymbol{O}}{tr[\boldsymbol{\rho}_{\text{res}}\boldsymbol{O}]} \tag{5.5}$$

Finally, in order to introduce the last concept of the density operator formalism that we will need, briefly assume that we do not perform intermediate detections anymore, but that the A modes still leave the system. In order to calculate the remaining feedback state, we need to calculate a partial trace. This is a generalized form of the trace that was introduced earlier and it allows us to describe density matrices of bipartite systems, i.e. systems that operate on a Hilbert space that is formed by taking a tensor product of two smaller Hilbert spaces. In this particular case, the smaller Hilbert spaces are defined on the A and B modes. In order to consider the state on the B modes, we need to trace over the A modes of the system:

$$\boldsymbol{\rho}_{\text{fb}} = \text{tr}_A\left[\boldsymbol{\rho}_{AB}\right] = \sum_{\mu} {}_A\!\left\langle\mu\right|\boldsymbol{\rho}_{AB}\left|\mu\right\rangle_A \tag{5.6}$$

Here, $\boldsymbol{\rho}_{AB} = \boldsymbol{\rho}_{\text{res}}$. Note that ${}_A\!\left\langle\mu\right|\cdots\left|\mu\right\rangle_A$ only operates on the A modes, such that we get a feedback density matrix in the B modes, rather than a number (as was the case in Equation (5.4)). Furthermore, μ runs over all possible Fock states in the A modes. If we know for example that $n_{\text{res}} = 2$ and $m_A = 2$, then $\mu \in \{\left|00\right\rangle, \left|10\right\rangle, \left|01\right\rangle, \left|20\right\rangle, \left|11\right\rangle, \left|02\right\rangle\}$.

Mathematical comparison

Let us generalise the results from Section 4.5 starting from a general pure reservoir state instead of the specific state of Equation (5.1) and subsequently shift to the density operator formalism. This will facilitate the comparison with the approach without intermediate detections further in this section. Although the description below is partly based on Section 4.5, afterwards the same results will be derived generally and independently in Appendix C.

In the first iteration we will always have a pure reservoir state. Generally we can describe this state by:

$$|\psi_{\mathsf{res}}(k=1)\rangle = \sum_{\kappa} |\kappa\rangle_A \otimes \big(\sum_{\sigma} a_{\sigma}^{(\kappa)} \, |\sigma\rangle_B \big) \tag{5.7}$$

Here, κ is a basis state of the Fock basis on the A modes, while σ is a basis state of the Fock basis on the B modes. Note that the complex coefficients are labelled with the basis states they are associated with, rather than with arbitrary integers. Also note that we will no longer write the iteration number as an argument in the following equations. This notation will however be repeated at the start of the second iteration.

Assume that we detect a certain μ on the A modes. Similar to Equation (5.2), the probability of this event is given by:

$$P(\mu) = \sum_{\sigma} \left| a_{\sigma}^{(\mu)} \right|^2 \tag{5.8}$$

Similar to Equation (5.3), this leads to the following feedback state:

$$|\psi_{\mathsf{fb}}^{(\mu)}\rangle = \frac{1}{\sqrt{\sum_{\sigma} \left| a_{\sigma}^{(\mu)} \right|^2}} \sum_{\sigma} a_{\sigma}^{(\mu)} \, |\sigma\rangle_B \tag{5.9}$$

Combination with the input state of the second iteration and transformation of the QONN (described by Equation (3.4)) leads to:

$$|\psi_{\mathsf{res}}(k=2)\rangle = \boldsymbol{S}\big(\, |\psi_{\mathsf{in}}\rangle \otimes |\psi_{\mathsf{fb}}^{(\mu)}\rangle \, \big) \boldsymbol{S}^{\dagger} \tag{5.10}$$

As we performed an intermediate detection, this state is again pure and can hence alternatively be described by the following density matrix:

$$\boldsymbol{\rho}_{\mathsf{res}}^{(\mu)}(k=2) = \boldsymbol{S} \left(|\psi_{\mathsf{in}}\rangle \langle \psi_{\mathsf{in}}| \otimes |\psi_{\mathsf{fb}}^{(\mu)}\rangle \langle \psi_{\mathsf{fb}}^{(\mu)}| \right) \boldsymbol{S}^{\dagger} = \boldsymbol{S} \left(\boldsymbol{\rho}_{\mathsf{in}} \otimes \boldsymbol{\rho}_{\mathsf{fb}}^{(\mu)} \right) \boldsymbol{S}^{\dagger} \tag{5.11}$$

Note that, after detecting a certain μ, we know how many photons are present in $\boldsymbol{\rho}_{\mathsf{in}} \otimes \boldsymbol{\rho}_{\mathsf{fb}}^{(\mu)}$ and we can calculate the matrix $\boldsymbol{S}$ using a certain $\mathrm{basis}(n_{\mathsf{res}}, m)$. According to Equation (5.4), we find the following expectation values:

$$\langle \boldsymbol{O} \rangle^{(\mu)} = tr_{AB} \left[\boldsymbol{\rho}_{\mathsf{res}}^{(\mu)} \boldsymbol{O} \right] \tag{5.12}$$

where $\boldsymbol{O} = |\mu'\rangle_{A \, A} \langle \mu'|$.

Note that, although Equation (5.8) and Equation (5.12) both describe the probability of a certain detection (of μ and μ' respectively), we adapt a different notation in both equations to stress the fact that we are interested in the expectation values of the second iteration. The idea is to compare these expectation values with their corresponding expression in the procedure without intermediate detections.

The process described above is summarized in Figure 5.8 using a probability tree. With a probability of $P(\mu)$, we are led to the outcome $|\mu\rangle_A$ at the end of the first iteration. Knowing this detection has taken place, we are led to the detection $|\mu'\rangle_A$ at the end of the second iteration with a probability of $\langle \boldsymbol{O} \rangle^{(\mu)}$. In this figure, only five possible measurement outcomes are drawn at each iteration. Note that in general, this will not be the case, as the number of possible outcomes depends on the values of n_{res} and m_A. Keeping into account all possible states μ that can be detected at the intermediate measurement, we find the (unconditional) expectation values of μ':

$$\langle \boldsymbol{O} \rangle = \sum_{\mu} P(\mu)\, \langle \boldsymbol{O} \rangle^{(\mu)} \tag{5.13}$$

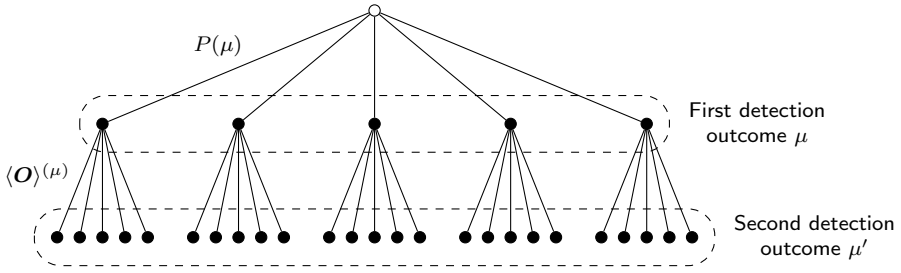


Figure 5.8: Schematic representation of the procedure with initermediate detections. The outcome $|\mu\rangle_A$ is obtained with probability $P(\mu)$. Given a certain $|\mu\rangle_A$, the probability of detecting $|\mu'\rangle_A$ at the end of the second iteration is $\langle \boldsymbol{O} \rangle^{(\mu)}$.

In order to complete this comparison, we remove all intermediate detections, following a similar procedure as the one introduced in Ref. [39]. More specifically, we assume that the detection modes still leave the system, but are only detected at the final iteration, keeping in mind that we need to reset the system afterwards.

Again, we start from a general pure reservoir state state. As we need to perform a partial trace to describe the separation of the A and B modes, we first write the reservoir state as a density operator:

$$\boldsymbol{\rho}_{\text{res}}(k=1) = |\psi_{\text{res}}\rangle\, \langle \psi_{\text{res}}| \tag{5.14}$$

According to Equation (5.6), the feedback density matrix is given by:

$$\begin{aligned} \boldsymbol{\rho}_{\text{fb}} &= tr_A\left[\boldsymbol{\rho}_{\text{res}}\right] = \sum_{\mu} {}_A\langle \mu\, |\boldsymbol{\rho}_{\text{res}}|\, \mu \rangle_A \\ &= \sum_{\mu} \left(\sum_{\sigma} \sum_{\tau} a_{\sigma}^{(\mu)} a_{\tau}^{(\mu)^*}\, |\sigma\rangle_{B\ B}\langle \tau| \right) = \sum_{\mu} P(\mu)\, |\psi_{\text{fb}}^{(\mu)}\rangle\, \langle \psi_{\text{fb}}^{(\mu)}| \end{aligned} \tag{5.15}$$

where we used the expression of Equation (5.7) for $|\psi_{\text{res}}\rangle$. Note that τ also is a basis state of the Fock basis on the B modes. $P(\mu)$ and $|\psi_{\text{fb}}^{(\mu)}\rangle$ are respectively the same as in Equation (5.8) and Equation (5.9). However, note that $\boldsymbol{\rho}_{\text{fb}}$ is not a pure state anymore, but rather behaves like a statistical ensemble of 'collapsed states'.

Combination with $\boldsymbol{\rho}_{\text{in}} = |\psi_{\text{in}}\rangle \langle\psi_{\text{in}}|$ of the second iteration and transformation by the QONN leads to:

$$\boldsymbol{\rho}_{\text{res}}(k=2) = \boldsymbol{S}\left(\boldsymbol{\rho}_{\text{in}} \otimes \boldsymbol{\rho}_{\text{fb}}\right)\boldsymbol{S}^{\dagger} = \sum_{\mu} P(\mu)\boldsymbol{S}\left(\boldsymbol{\rho}_{\text{in}} \otimes \boldsymbol{\rho}_{\text{fb}}^{(\mu)}\right)\boldsymbol{S}^{\dagger} \tag{5.16}$$

Note that this expression again contains the boson sampling result of the QONN. For ease of notation, this matrix is simply still denoted by $\boldsymbol{S}$. In reality, this matrix however differs between terms as it operates on a different number of photons, leading to Hilbert spaces of different size. Also note that removing the intermediate detections requires us to use the density operator formalism in order to perform the partial trace. With intermediate detections, the use of these density matrices was also possible, but not required.

Finally, by applying Equation (5.4), we find that the resulting expectation values can be written as a function of $\langle\boldsymbol{O}\rangle^{(\mu)}$, as defined in Equation (5.12):

$$\langle\boldsymbol{O}\rangle = tr_{AB}\left[\boldsymbol{\rho}_{\text{res}}\boldsymbol{O}\right] = \sum_{\mu} P(\mu)\; tr_{AB}\left[\boldsymbol{\rho}_{\text{res}}^{(\mu)}\boldsymbol{O}\right] = \sum_{\mu} P(\mu)\; \langle\boldsymbol{O}\rangle^{(\mu)} \tag{5.17}$$

where $\boldsymbol{O} = |\mu'\rangle_{A\,A}\langle\mu'|$ and $\boldsymbol{\rho}_{\text{res}}^{(\mu)}$ is the same as in Equation (5.11). Comparing Equation (5.13) and Equation (5.17), we conclude that the final expectation values are identical in both procedures, regardless of whether we perform intermediate detections.

However, as we have already noted, the remaining feedback state is now mixed rather than pure. Hence, if we want to prove that the expectation values are identical for procedures longer than two iterations, we have to repeat this mathematical description, starting from a general mixed state at a general iteration k. As this proof is similar to the one above, it is provided in Appendix C.

Interpretation

The acquired result can be interpreted as follows: when performing an intermediate measurement, in effect we follow a specific branch from the probability tree that is defined by the complex coefficients of the state in front of the detector. Without these intermediate measurements, all these possible branches are kept into account until the last iteration, by means of a mixed state which is connected to a statistical ensemble. As both approaches lead to the same expectation values and as these expectation values are the input of the regressor, both approaches also lead to the equally performing systems.

Note that this result was not expected to be true in advance. For example, Ref. [42] also uses intermediate detections and although it does not go into detail about its consequences, it mentions that detections can also be a source of computational

power. Moreover, they can be a source of nonlinearity, as is also the case in Ref. [70]. The fact that detectors can induce nonlinearity is for example also applicable to classical photonic systems, where complex amplitudes are converted into real-valued power levels by squaring the magnitude of the complex representation of the coherent light signal [71]. Nonlinearity is important, because it has already proven to be beneficial to tackle different computational tasks in classical reservoir computing [72]. The controlled erasure of information can even be used to tune the fading memory, a technique that was also implemented in Ref. [39].

From the earlier mathematical comparison we see that the removal of the intermediate detections does in fact not remove any nonlinearity in this model. As nonlinearity is connected with the removal of information, as is for example the case when you intentionally remove information to create nonlinearity [39], the reason why our system is unaltered can be attributed to the fact that the A modes still leave the system. In other words, as we eventually describe all possible preparations of the system, both in the approach with and without intermediate detections, we lose as much information when performing a partial trace as when performing a detection.

Note that other quantum computing approaches often have stationary quantum states instead of states that move in space. In that case no information leaves the system without intermediate detections. Consequently, it can be assumed that the fact that we lose as much information with as without intermediate detections is a characteristic of this QRC approach.

5.2.3 Consequences for the simulation

A more efficient simulation of the ensemble

As follows from the discussion above, the removal of the intermediate detections has two major implications for the simulations.

On the one hand, we notice that the value of n_{res} no longer is exactly known in each of the reservoir copies, since we no longer extract a certain number of photons at the end of each iteration. As a result, we are required to use the lossy basis that was introduced in Section 4.3.

On the other hand, we are not able to describe the system with state vectors anymore, but instead we use density matrices. As was discussed in the previous section, the reason for this can be found in the mixed states that emerge at the first partial trace. These mixed states, which appear to behave like a statistical ensemble of collapsed states, can only be described by density operators. By doing so, we can account for the different possible preparations of the system by only using a single density matrix. This contrasts with the approach with intermediate detections, as there, the detector ensured that the reservoir state remained pure. Note however that, although we no longer need to describe different reservoir copies during the simulation, these copies do still need to be prepared in reality as we perform a measurement at the final iteration.

Also remember that in reality, if we do not perform intermediate detections, we need to restart the system after performing the final measurement, preventing us

from extracting information in further iterations. Luckily, we are able to circumvent this problem during the simulation. Although, in reality, we are bound to the no-cloning theorem [73] which states that it is impossible to create an independent and identical copy of an arbitrary unknown quantum state, in our simulation we can make such copies. This leads us to an alternative simulation method for which we iterate over the complete data set, without restarting the system and without intermediate measurements, while constantly making a copy of the reservoir density matrix. Afterwards, we can calculate the expectation values at each iteration from these copies.

Photon number threshold

As mentioned before, this new simulation method utilises the lossy basis that was introduced in Section 4.3. As a result, we need to choose a value for n_{thr}. Even though the use of such a threshold is common in photonic quantum computing, we notice here that its arbitrary nature becomes more problematic in the context of temporal training procedures, such as QRC.

As we do not perform intermediate detections, at first sight, we need to account for all possible measurement outcomes at the detector, no matter how small their probability. Each iteration, the possibility exists that no photons leave the system, in theory leading to a value of n_{thr} that rises with the number of iterations. As both the computational cost of boson sampling and the dimensions of the boson sampling results scale exponentially in n, this would lead to unfeasible computing times.

However, if we do introduce a finite threshold value, we need to exclude events artificially from the density matrix that contain more photons than n_{thr}, after which we need to re-normalize the resulting density matrix. As compared to the approach with intermediate detections, it is however more difficult to estimate which values of n_{res} are realistic. Although the reservoir state dictates the probability of a certain preparation, these probabilities are only calculated as we progress through the simulation. In general, as we are working with an approximation of the real density matrix, it is harder to know when the threshold starts to influence the results of our simulation. We expect the approximation to become worse for lower thresholds, while higher thresholds can result in long simulation times.

Consequently, when implementing the new simulation method, it would be necessary to have a thorough understanding of the photon number fluctuations. For this purpose, the currently existing simulation (with intermediate detections) can be used to simulate realistic values for n_{res}, using the same parameter values in both simulations. Ultimately, the threshold can be set at n_{max}, the maximum number of photons during training with intermediate detections.

As our system utilises a quantum state that moves in space, these photon number fluctuations and the arbitrary choice of the threshold value seem to be typical for the platform that we use. For that reason, it is part of our future objectives to study these mechanisms more.

5.2.4 Consequences for a hardware implementation

To conclude this chapter, let us leave the simulations behind and reconsider a hardware implementation without intermediate detections. As is clear by now, such an implementation would require us to reset the system a number of times. Although this restriction is relaxed by the convergence property (as explained in Section 5.2.1), it still limits the efficiency of the system.

However, the implementation without intermediate detections can be useful for another reason. As was discussed in Section 2.4.1, strong projective measurements can, in addition to their expected probabilistic behaviour, also induce back-action due to non-idealities in the implementation of the detector. At the moment, it is however unclear to what extent these non-idealities would influence the performance of the system. As the set-up with and without intermediate detections ideally would lead to the same expectation values, a comparison of their hardware implementations could serve as a useful tool to study this effect.

A hardware implementation without intermediate detections would however not be easy to implement. Note that photons leaving the system still remain entangled with the reservoir. If these emitted photons are unwillingly manipulated of measured, the state of the reservoir can be altered. Ideally, the decoherence of these photons would therefore be prevented until the end of the experiment, but no further research has been done here to determine if and how this can be done. This shows that, as opposed to quantum platforms with stationary reservoir states, on this platform a hardware implementation without intermediate detections is less obvious.

Chapter 6

Future outlook

In this chapter, we take a look at the future objectives of this thesis. We will present some extensions that could increase the efficiency of the current model and also discuss some elements of the the system that would benefit from a more in-depth analysis.

6.1 Photon fluctuations

As is clear by now, the photon number fluctuations, which are typical for the platform that we use, greatly impact the simulation time of our model. Moreover, as appears to be the case in Section 5.1.3 and Section 5.1.4, they also seem to influence the performance of our system. However, whether this actually is the case and how this link is established is unsure at the moment.

Therefore, it would be interesting to study these fluctuations for different parameters of our system. Doing so could give more insight in the behaviour of the system and possibly lead to new methods to shape these fluctuations and boost the performance.

Also in Section 5.2.3 we emphasised the relevance of such a study, as the alternative simulation with density matrices requires us to choose a realistic value for n_{thr}. Even though the currently existing simulation could be used to estimate such a threshold value, it would be beneficial to perform such an estimation without performing an additional simulation.

6.2 Optical loss and decoherence

An important factor in real photonic systems is that they can suffer from optical losses. In order to estimate the influence of these losses on the performance, it could be simulated that photons leave the system with a certain probability.

At first glance, it might be expected that this will limit the performance of the system severely. However, as is known to be the case for photonic quantum neural networks (QNNs) [74], it can be expected that our system can compensate for this

effect. In fact, deliberately adding loss to a QRC system has also proven to enable for tuning of the system's memory, similar to the effect of resetting qubits in a qubit-based system [39]. Moreover, it could prevent overfitting [75]. These remarks emphasise the relevance of including photon loss in our simulations.

Note that photon loss is a form of 'amplitude damping', which in turn is a type of 'decoherence'. As explained earlier, decoherence is the loss of quantum coherence, meaning that information encoded in the state of the system uncontrollably becomes entangled with its environment. The most important counterpart of amplitude damping, is 'dephasing', where no photons are lost, but the environment adjusts the relative phases of the system's state. Similar to photon loss, dephasing also removes information from the system and it can therefore be expected to have a similar influence on the performance.

How we will implement these phenomena depends on the simulation method. In the current simulation method, using reservoir copies, a simple first step would be to introduce loss events explicitly, removing a number of photons from a certain reservoir with a certain probability. For this purpose, we would need to put forward a scheme that changes the superposition in that reservoir to a new superposition that only contains Fock states with the new photon number. By doing so, we ensure that in each reservoir we still know the exact value of n_{res} and we do not need to use a lossy basis. Additionally, the scheme could also change the relative phases of the Fock terms, simulating the dephasing process. How such a scheme should be constructed to provide a realistic description of both processes, would in this case become an important additional research question.

However, if we would use the alternative simulation method of Section 5.2.3 instead, we would need to model these phenomena in another way. As here, we are working with density matrices that cannot be written as a single superposition of Fock states, we would need to put forward a more advanced operation scheme. Although creating such a scheme may seem like a burden, luckily, we can also make use of existing decoherence models [30] that have already proven to give an accurate description of these phenomena.

Note that these existing decoherence models are more common in quantum optics and should be considered the correct standard. Although, in the light of Section 5.2, we expect that the previously proposed 'loss events' also give realistic results, without introducing uncertainty of n, further research must be carried out to show whether this actually is the case.

6.3 Input encoding

As was explained in Section 2.4.1, an important property of all QRC platforms is that they show an exponential growth in Hilbert space dimensions when increasing the number of quantum elements, which plays a decisive role in their performance [35]. The main objective of QRC is to boost the efficiency of a computation by exploiting this large number of degrees of freedom (DOF) that is present even for small networks. The degree to which it is possible to exploit such DOFs however is not only determined by the size of the Hilbert space.

An important example of this are input encodings, which are known to play a decisive role in QML [76]. Although in this thesis, we have not focused strongly on maximally optimizing the system (remember, for example, that we did not fit any hyperparameters), this is a future goal. By doing so, a study of the possible input encodings will also become necessary. A first starting point for this could be to introduce an additional QONN in front of the current QONN, operating on the A modes and hence transforming the input states before guiding them to the reservoir. Furthermore, the parameters of this new QONN can be optimized.

6.4 Gaussian boson sampling

One of the main issues of boson sampling with Fock states is that it can be difficult to generate these states. This results from the fact that most set-ups use spontaneous parametric down-conversion to generate single photons. As was introduced in Section 3.2, these single-photon sources utilise heralded photon pairs that are generated with a certain probability. As a result, we do not receive a single photon whenever we trigger the source, but we do know when the single photon is generated. This contrasts sharply with the class of deterministic photon sources that can generate photons 'on-demand'. However, these deterministic sources are more difficult and costly to make.

As a result, when using probabilistic single-photon sources, we have to wait until all photon sources emit a photon simultaneously in order to generate a certain Fock state. The average time to generate such a Fock state hence scales exponentially with the number of photons it contains.

However, boson sampling can also be performed without Fock states. Instead, we can use Gaussian states, more specifically single mode squeezed states, which can be generated simultaneously and on-demand. These states are used as input for a boson sampling set-up, using a linear interferometer and single-photon detectors. This procedure is called Gaussian boson sampling. It was first introduced in Ref. [77].

In contrast to regular boson sampling, in this case the probability of a certain output pattern of photons is connected to the hafnian function, which is a generalization of the permanent function that we introduced before. The calculation of such hafnians also falls within the P# complexity class, showing that Gaussian boson sampling produces problems that are 'equally hard' as boson sampling with Fock states.

As a result, Gaussian boson sampling can be seen as a more practical version of boson sampling with Fock states, that is implemented with a very similar set-up. For this reason, it would be interesting to study how the introduction of Gaussian states would influence the results that were acquired in this thesis.

6.5 Capacity measures

Up to now, we only considered a simple classical task in order to study the system. This does not suffice to quantify the potential of the system to exploit its Hilbert space. Such an analysis would require the use of a so-called capacity measure. Capacity measures enable us to study what kind of linear and nonlinear memory functions a given system can approximate (often called the 'expressive power' of a system). The most common example of such a measure, is the Information Processing Capacity (IPC) [78].

Recently, in Ref. [79], a capacity analysis of various parameterized quantum circuits was published, also showing how the capacity scales with the number of qudits that are used. This analysis could be used to compare a future capacity study of our own system.

Finally, we note that since capacity measures such as the IPC are defined in the context of classical tasks, they cannot be used to quantify the ability of a system to perform quantum tasks. However, it would also be interesting to study the capacity of our own system to solve such tasks. Fortunately, although research into quantum alternatives of capacity measures is still very limited, [80] recently provided a capacity measure that can be used both in the context of classical and quantum tasks. Although in this paper its use for quantum tasks is discussed, it is however not yet applied in this manner.

Chapter 7

Conclusions

In this thesis, we introduced an new photonic QRC method and provided a proof-of-principle using a simple classical task. It was discussed how the different elements of the system can be simulated, after which the performance of the system was tested.

The performance appeared to improve by using a larger ensemble. A similar improvement is also expected to occur as a function of the train set size, for more challenging tasks. A study of the Kerr interaction strength ϕ of the nonlinearities in the reservoir additionally showed that this parameter influences both the photon number fluctuations and the performance of the system, increasing the initialisation-dependence if we remove the nonlinearities. Further research is required to reveal the mutual links between these observations and to check whether the over-all initialisation-dependence can be lowered to the same level as classical RC systems.

Afterwards, we showed that an alternative operating procedure without intermediate detections leads to the same expectation values at the detector and hence leads to an equally performing system. This enabled us to propose a more efficient simulation method for our system and to stress some of the technicalities that need to be taken into account to implement such a simulation.

Finally, we stress that our model is not fully optimised at the moment and that it would benefit from different extensions, of which a few were discussed in Chapter 6. As is discussed in Ref. [17], it is premature to make quantitative comparisons of QRC models with their much more advanced classical counterparts, but the study of our model did highlight some paths that can be researched with the aim of constructing a worthy competitor for such classical systems.

Appendices

A An alternative reservoir set-up

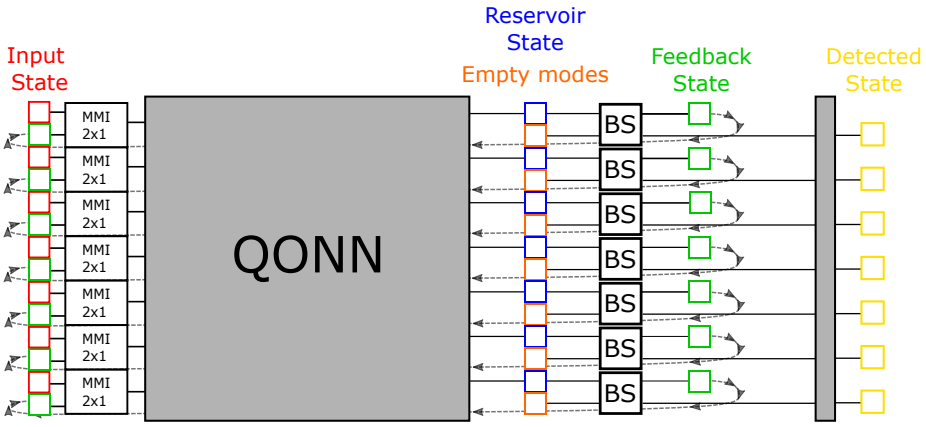


Figure A.1: Layout of an alternative realization of the quantum reservoir in Figure 4.1. The new elements in this set-up are the 2x1 multi-mode interferometers (MMI), the empty modes (orange) in which no photons are sent and the beamsplitters (BS).

Figure A.1 shows an alternative realization of a quantum reservoir. This set-up was constructed in parallel with the set-up of Figure 4.1, but later appeared to be disadvantageous for several reasons. In this section, these reasons are explained after discussing the layout of this reservoir. Rather than simply addressing a previously excluded step of the research process, this appendix serves to strengthen the choices in the final reservoir configuration of Figure 4.1.

As is the case in Figure 4.1, the input state is transformed by a randomly initialised QONN. The output modes of the QONN however are not immediately separated. Instead, they are combined with an equal number of 'empty modes'. As the name suggests, no photons are sent into these newly added modes. The empty modes interact with the output modes of the QONN via beamsplitters. Each of these beamsplitters acts on two neighbouring modes. In order to add recurrent connections, the upper output modes of the beamsplitters are redirected to the input modes, while the lower output modes are led to the single-photon detectors.

A first problem that we encounter in this set-up is the addition of the beamsplitters. Apart from calculating the transformation matrix $\boldsymbol{S}(\boldsymbol{\Theta})$ of the QONN, we also have to boson sample the layer of beamsplitters, using twice the number of modes. In more mathematical terms, this model requires us to calculate $\Phi[\mathbf{B}]$, where $\mathbf{B}$ is a $2m \times 2m$ matrix, given by:

$$\mathbf{B} = \begin{bmatrix} \mathbf{BS} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{BS} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{BS} \end{bmatrix} \tag{A.1}$$

Here, $\mathbf{BS}$ follows the definition of Equation (3.1), while $\mathbf{0}$ is a zero matrix. As mentioned before, the computational load of the boson sampling algorithm scales strongly with n and m. As a result, the empty modes strongly limit the size of this model.

A second problem that is induced by this set-up appears when combining the input state with the feedback state. Practically, this can be implemented by making use of 2x1 multi-mode interferometers (MMIs), of which a schematic representation is shown in Figure A.2. This component essentially is a broad optical waveguide with a number of guided eigenmodes that propagate independently from one another and at different velocities. When such an MMI is excited by an incident wave, the field profile is decomposed into these eigenmodes, resulting in an interference pattern that changes along the length of the MMI. By making an MMI with specific dimensions, it can combine the optical power of its input modes into its output modes. However, 2x1 MMIs are known to induce optical losses in classical applications, which is a result of the lower number of output modes with respect to the input modes. As a result, the component will induce decoherence in quantum applications. In other words, nature will end up adding another output channel to the device, i.e. a radiative mode, allowing photons to escape in order to preserve unitarity. Due to this behaviour, the MMIs will perform poorly when processing quantum information.

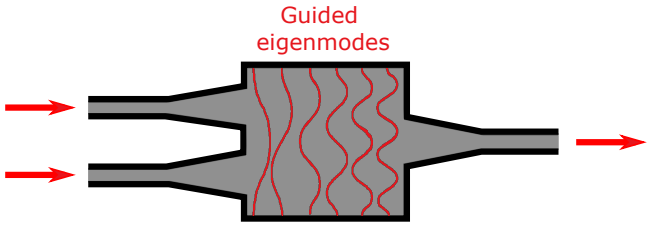
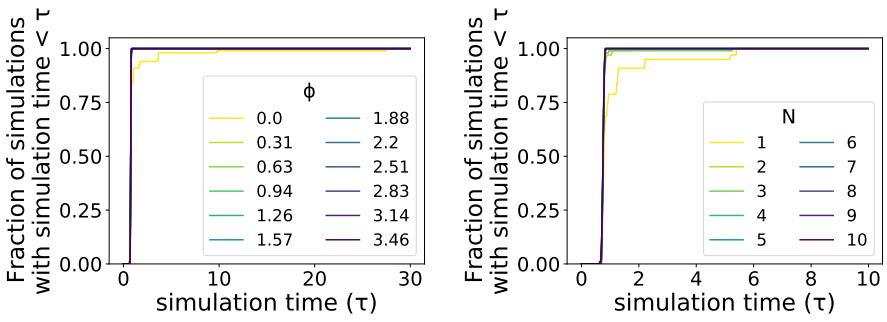


Figure A.2: 2x1 multi-mode interferometer

Building on this second problem, we note that the inherent decoherence of the MMIs also induces uncertainty on the photon number in the QONN (n_{res}). As long as the radiative modes of the MMIs are not measured, n_{res} is not exactly known and we are obliged to resort to the lossy basis for the simulations, as was explained in Section 4.3.

B Additional figures for the study of system parameters



(a) (b)

Figure B.3: The simulation time of 100 different reservoir initialisations for different values of ϕ (a) and N (b), depicted by a cumulative distribution function. For each simulation time τ on the x-axis, the fraction of simulations (with different reservoir initialisation) is plotted that lead to a simulation time lower than τ. For (a), $N = 6$, while for (b) $\phi = \pi$. Other parameters: $m_A = m_B = 2$, train set size $= 1000$, ensemble size $= 600$, $\alpha = 10^{-4}$.

C Derivation of the expectation values, starting from a general mixed state

In Section 5.2.2, a mathematical comparison was given of the simulation of the system (using a classical optimizer), with and without intermediate detections. Both derivations however started from a pure reservoir state. In the approach with intermediate detections, the resulting reservoir state in the second iteration still is a pure state as a result of the strong projection. In the case that we do not perform intermediate detections, we however end up with a mixed state after performing the partial trace. In order to make conclusions about the expectation values in both approaches, we generalise this example starting from a general mixed state:

$$\boldsymbol{\rho}_{\text{res}}(k) = \sum_{\kappa,\sigma} \sum_{\nu,\tau} a_{\sigma\tau}^{(\kappa)(\nu)} \left(|\kappa\rangle_A \otimes |\sigma\rangle_B \right) \left({}_A\langle\nu| \otimes {}_B\langle\tau| \right) \tag{C.1}$$

Here, κ and ν are basis states of the Fock basis on the A modes, while σ and τ are basis states of the Fock basis on the B modes.

First assume that we do perform an intermediate measurement. According to Equation (5.4), the probability of a certain outcome μ is given by:

$$P(\mu) = tr_{AB}\left[|\mu\rangle_A \, {}_A\langle\mu| \, \boldsymbol{\rho}_{\text{res}} \right] = \sum_{\sigma} a_{\sigma\sigma}^{(\mu)(\mu)} \tag{C.2}$$

According to Equation (5.5), this results in the following feedback density matrix:

$$\boldsymbol{\rho}_{\text{fb}}^{(\mu)} = \frac{1}{\sum_{\sigma} a_{\sigma\sigma}^{(\mu)(\mu)}} \sum_{\sigma,\tau} a_{\sigma\tau}^{(\mu)(\mu)} \left|\sigma\right\rangle_{B\ B}\!\left\langle\tau\right| \tag{C.3}$$

Combination with the input state and transformation by the QONN leads to:

$$\boldsymbol{\rho}_{\text{res}}^{(\mu)}(k+1) = \boldsymbol{S}\left(\boldsymbol{\rho}_{\text{in}} \otimes \boldsymbol{\rho}_{\text{fb}}^{(\mu)}\right)\boldsymbol{S}^{\dagger} \tag{C.4}$$

Applying Equation (5.5) once again, this results in the following expectation values:

$$\langle O\rangle^{(\mu)} = tr_{AB}\left[\boldsymbol{\rho}_{\text{res}}^{(\mu)} O\right] \tag{C.5}$$

Starting from the same general mixed state of Equation (C.1), we go through the parallel calculations in the case that we do not measure intermediately. By doing so, we get the following feedback density matrix:

$$\boldsymbol{\rho}_{\text{fb}} = tr_A\left(\boldsymbol{\rho}_{\text{res}}\right) = \sum_{\mu}\left(\sum_{\sigma,\tau} a_{\sigma\tau}^{(\mu)(\mu)} \left|\sigma\right\rangle_{B\ B}\!\left\langle\tau\right|\right) = \sum_{\mu} P(\mu)\ \boldsymbol{\rho}_{\text{fb}}^{(\mu)} \tag{C.6}$$

Here, $P(\mu)$ and $\boldsymbol{\rho}_{\text{fb}}^{(\mu)}$ are respectively given by Equation (C.2) and Equation (C.3). We see that $\boldsymbol{\rho}_{\text{fb}}$ again behaves like an ensemble of collapsed states. The only difference we have by not starting with a pure state, is that $\boldsymbol{\rho}_{\text{fb}}^{(\mu)}$ no longer is a pure state either. Combination with the input state and transformation by the QONN leads to:

$$\begin{aligned}\boldsymbol{\rho}_{\text{res}}(k+1) =& \boldsymbol{S}\left(\boldsymbol{\rho}_{\text{in}} \otimes \boldsymbol{\rho}_{\text{fb}}\right)\boldsymbol{S}^{\dagger} \\ =& \sum_{\mu} P(\mu)\ \boldsymbol{S}\left(\boldsymbol{\rho}_{\text{in}} \otimes \boldsymbol{\rho}_{\text{fb}}^{(\mu)}\right)\boldsymbol{S}^{\dagger} = \sum_{\mu} P(\mu)\ \boldsymbol{\rho}_{\text{res}}^{(\mu)}\end{aligned} \tag{C.7}$$

Here, we also recognize $\boldsymbol{\rho}_{\text{res}}^{(\mu)}$ from Equation (C.4). Finally, this leads to expectation values that we can write as a function of the $\langle \boldsymbol{O}\rangle^{(\mu)}$ from Equation (C.5):

$$\langle \boldsymbol{O}\rangle = tr_{AB}\left[\boldsymbol{\rho}_{\text{res}}\boldsymbol{O}\right] = \sum_{\mu} P(\mu)\ tr_{AB}\left[\boldsymbol{\rho}_{\text{res}}^{(\mu)}\boldsymbol{O}\right] = \sum_{\mu} P(\mu)\ \langle \boldsymbol{O}\rangle^{(\mu)} \tag{C.8}$$

where $\boldsymbol{O} = \left|\mu'\right\rangle_{A\ A}\!\left\langle\mu'\right|$.

This generally shows that we find the same expectation values, with and without intermediate detections.

Bibliography

[1] E. Alpaydin, *Introduction to machine learning*. MIT press, 2020.

[2] J. Alzubi, A. Nayyar, and A. Kumar, "Machine learning from theory to algorithms: an overview," in *Journal of physics: conference series*, vol. 1142, no. 1. IOP Publishing, 2018, p. 012012.

[3] M. A. Nielsen and I. Chuang, "Quantum computation and quantum information," 2002.

[4] N. S. Yanofsky, "An introduction to quantum computing," 2007.

[5] S. Cook, "The p versus np problem," *The millennium prize problems*, pp. 87–104, 2006.

[6] A. W. Harrow and A. Montanaro, "Quantum computational supremacy," *Nature*, vol. 549, no. 7671, pp. 203–209, 2017.

[7] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM review*, vol. 41, no. 2, pp. 303–332, 1999.

[8] M. Schuld, I. Sinayskiy, and F. Petruccione, "An introduction to quantum machine learning," *Contemporary Physics*, vol. 56, no. 2, p. 172–185, Oct 2014. [Online]. Available: http://dx.doi.org/10.1080/00107514.2014.964942

[9] A. Krenker, J. Bešter, and A. Kos, "Introduction to the artificial neural networks," *Artificial Neural Networks: Methodological Advances and Biomedical Applications. InTech*, pp. 1–18, 2011.

[10] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[11] J. Von Neumann, "First draft of a report on the edvac," *IEEE Annals of the History of Computing*, vol. 15, no. 4, pp. 27–75, 1993.

[12] B. Schrauwen, D. Verstraeten, and J. Van Campenhout, "An overview of reservoir computing: theory, applications and implementations," in *Proceedings of the 15th european symposium on artificial neural networks. p. 471-482 2007*, 2007, pp. 471–482.

[13] G. V. D. Sande, D. Brunner, and M. C. Soriano, "Review article Advances in photonic reservoir computing," *Nanophotonics*, vol. 6, no. 3, pp. 561–576, 2017.

[14] A. Sherstinsky, "Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network," *CoRR*, vol. abs/1808.03314, 2018. [Online]. Available: http://arxiv.org/abs/1808.03314

[15] G. Tanaka, T. Yamane, J. B. Héroux, R. Nakane, N. Kanazawa, S. Takeda, H. Numata, D. Nakano, and A. Hirose, "Recent advances in physical reservoir computing: A review," *Neural Networks*, vol. 115, pp. 100–123, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608019300784

[16] D. Brunner, M. C. Soriano, and G. Van der Sande, *Photonic Reservoir Computing: Optical Recurrent Neural Networks*. Walter de Gruyter GmbH & Co KG, 2019.

[17] P. Mujal, R. Martínez-Peña, J. Nokkala, J. García-Beni, G. L. Giorgi, M. C. Soriano, and R. Zambrini, "Opportunities in quantum reservoir computing and extreme learning machines," *arXiv preprint arXiv:2102.11831*, 2021.

[18] S. Ruder, "An overview of gradient descent optimization algorithms," 2017.

[19] S. Sharma and S. Sharma, "Activation functions in neural networks," *Towards Data Science*, vol. 6, no. 12, pp. 310–316, 2017.

[20] D. E. Rumelhart, R. Durbin, R. Golden, and Y. Chauvin, "Backpropagation: The basic theory," *Backpropagation: Theory, architectures and applications*, pp. 1–34, 1995.

[21] A. Ng, K. Katanforoosh, and Y. Bensouda Mourri, "Lecture on gradient descent from the coursera neural networks and deep learning course," https://www.coursera.org/learn/neural-networks-deep-learning/lecture/A0tBd/gradientdescent, 2021, accessed 18/05/21.

[22] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*. Springer, 2010, pp. 177–186.

[23] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International conference on machine learning*. PMLR, 2013, pp. 1310–1318.

[24] W. Maass, T. Natschläger, and H. Markram, "Real-Time Computing Without Stable States: A New Framework for Neural Computation Based on Perturbations," *Neural Computation*, vol. 14, no. 11, pp. 2531–2560, 11 2002. [Online]. Available: https://doi.org/10.1162/089976602760407955

[25] M. Lukoševičius and H. Jaeger, "Reservoir computing approaches to recurrent neural network training," *Computer Science Review*, vol. 3, no. 3, pp. 127–149, 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1574013709000173

[26] C. Fernando and S. Sojakka, "Pattern recognition in a bucket," in *European conference on artificial life*. Springer, 2003, pp. 588–597.

[27] Y. Paquot, F. Duport, A. Smerieri, J. Dambre, B. Schrauwen, M. Haelterman, and S. Massar, "Optoelectronic reservoir computing," *Scientific reports*, vol. 2, no. 1, pp. 1–6, 2012.

[28] L. Larger, M. C. Soriano, D. Brunner, L. Appeltant, J. M. Gutiérrez, L. Pesquera, C. R. Mirasso, and I. Fischer, "Photonic information processing beyond turing: an optoelectronic implementation of reservoir computing," *Optics express*, vol. 20, no. 3, pp. 3241–3249, 2012.

[29] F. Duport, A. Smerieri, A. Akrout, M. Haelterman, and S. Massar, "Fully analogue photonic reservoir computer," *Scientific reports*, vol. 6, no. 1, pp. 1–12, 2016.

[30] M. Schlosshauer, "Quantum decoherence," *Physics Reports*, vol. 831, pp. 1–57, 2019.

[31] J. Preskill, "Quantum computing in the nisq era and beyond," *Quantum*, vol. 2, p. 79, 2018.

[32] P. A. Gagniuc, *Markov chains: from theory to implementation and experimentation*. John Wiley & Sons, 2017.

[33] B. Bylicka, D. Chruściński, and S. Maniscalco, "Non-markovianity and reservoir memory of quantum channels: a quantum information theory perspective," *Scientific reports*, vol. 4, no. 1, pp. 1–7, 2014.

[34] J. P. Clemens, S. Siddiqui, and J. Gea-Banacloche, "Quantum error correction against correlated noise," *Physical Review A*, vol. 69, no. 6, p. 062313, 2004.

[35] K. Fujii and K. Nakajima, "Harnessing disordered-ensemble quantum dynamics for machine learning," *Phys. Rev. Applied*, vol. 8, p. 024030, Aug 2017. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevApplied.8.024030

[36] M. Negoro, K. Mitarai, K. Fujii, K. Nakajima, and M. Kitagawa, "Machine learning with controllable quantum dynamics of a nuclear spin ensemble in a solid," 2018.

[37] J. M. Pino, J. M. Dreiling, C. Figgatt, J. P. Gaebler, S. A. Moses, M. S. Allman, C. H. Baldwin, M. Foss-Feig, D. Hayes, K. Mayer, and et al., "Demonstration of the trapped-ion quantum ccd computer architecture," *Nature*, vol. 592, no. 7853, p. 209–213, Apr 2021. [Online]. Available: http://dx.doi.org/10.1038/s41586-021-03318-4

[38] D. Lu, A. Brodutch, J. Park, H. Katiyar, T. Jochym-O'Connor, and R. Laflamme, "Nmr quantum information processing," *Electron Spin Resonance (ESR) Based Quantum Computing*, pp. 193–226, 2016.

[39] J. Chen, H. I. Nurdin, and N. Yamamoto, "Temporal information processing on noisy quantum computers," *Physical Review Applied*, vol. 14, no. 2, p. 1, 2020. [Online]. Available: https://doi.org/10.1103/PhysRevApplied.14.024065

[40] "Ibm quantum experience," https://www.ibm.com/quantum-computing/, accessed: 2021-05-26.

[41] T. Rudolph, "Why i am optimistic about the silicon-photonic route to quantum computing," *APL Photonics*, vol. 2, no. 3, p. 030901, 2017.

[42] J. Nokkala, R. Mart, G. L. Giorgi, V. Parigi, M. C. Soriano, and R. Zambrini, "Gaussian states provide universal and versatile quantum reservoir computing," *arXiv preprint arXiv:2006.04821*, pp. 1–13, 2020.

[43] L. C. G. Govia, G. J. Ribeill, G. E. Rowlands, H. K. Krovi, and T. A. Ohki, "Quantum reservoir computing with a single nonlinear oscillator," *Phys. Rev. Research*, vol. 3, p. 013077, Jan 2021. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevResearch.3.013077

[44] M. D. Eisaman, J. Fan, A. Migdall, and S. V. Polyakov, "Invited review article: Single-photon sources and detectors," *Review of scientific instruments*, vol. 82, no. 7, p. 071101, 2011.

[45] B. Lounis and M. Orrit, "Single-photon sources," *Reports on Progress in Physics*, vol. 68, no. 5, p. 1129, 2005.

[46] M. Cooper, L. J. Wright, C. Söller, and B. J. Smith, "Experimental generation of multi-photon fock states," *Opt. Express*, vol. 21, no. 5, pp. 5309–5317, Mar 2013. [Online]. Available: http://www.opticsexpress.org/abstract.cfm?URI=oe-21-5-5309

[47] R. H. Hadfield, "Single-photon detectors for optical quantum information applications," *Nature photonics*, vol. 3, no. 12, pp. 696–705, 2009.

[48] J. Carolan, J. D. Meinecke, P. J. Shadbolt, N. J. Russell, N. Ismail, K. Wörhoff, T. Rudolph, M. G. Thompson, J. L. O'brien, J. C. Matthews *et al.*, "On the experimental verification of quantum complexity in linear optics," *Nature Photonics*, vol. 8, no. 8, pp. 621–626, 2014.

[49] W. R. Clements, P. C. Humphreys, B. J. Metcalf, W. S. Kolthammer, and I. A. Walmsley, "Optimal design for universal multiport interferometers," *Optica*, vol. 3, no. 12, pp. 1460–1465, 2016.

[50] R. García-Patrón, J. J. Renema, and V. Shchesnovich, "Simulating boson sampling in lossy architectures," *Quantum*, vol. 3, p. 169, 2019.

[51] S. Aaronson and A. Arkhipov, "The computational complexity of linear optics," *Proceedings of the Annual ACM Symposium on Theory of Computing*, no. 0844626, pp. 333–342, 2011.

[52] A. Arkhipov and G. Kuperberg, "The bosonic birthday paradox," *Geometry & Topology Monographs*, vol. 18, no. 1, pp. 10–2140, 2012.

[53] S. Scheel, "Permanents in linear optical networks," *arXiv preprint quant-ph/0406127*, 2004.

[54] H. J. Ryser, "Combinatorial mathematics. the carus mathematical monographs, no. 14. published by the mathematical association of america; distributed by john wiley and sons," *Inc., New York*, 1963.

[55] H. S. Zhong, H. Wang, Y. H. Deng, M. C. Chen, L. C. Peng, Y. H. Luo, J. Qin, D. Wu, X. Ding, Y. Hu, P. Hu, X. Y. Yang, W. J. Zhang, H. Li, Y. Li, X. Jiang, L. Gan, G. Yang, L. You, Z. Wang, L. Li, N. L. Liu, C. Y. Lu, and J. W. Pan, "Quantum computational advantage using photons," *Science*, vol. 370, no. 6523, pp. 1460–1463, 2021.

[56] M. Schuld, I. Sinayskiy, and F. Petruccione, "The quest for a quantum neural network," *Quantum Information Processing*, vol. 13, no. 11, pp. 2567–2586, 2014.

[57] A. A. Ezhov and D. Ventura, "Quantum neural networks," in *Future directions for intelligent systems and information sciences*. Springer, 2000, pp. 213–235.

[58] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio *et al.*, "Variational quantum algorithms," *arXiv preprint arXiv:2012.09265*, 2020.

[59] K. Beer, D. Bondarenko, T. Farrelly, T. J. Osborne, R. Salzmann, D. Scheiermann, and R. Wolf, "Training deep quantum neural networks," *Nature communications*, vol. 11, no. 1, pp. 1–6, 2020.

[60] J. Preskill, "Quantum computing in the nisq era and beyond," *Quantum*, vol. 2, p. 79, 2018.

[61] G. R. Steinbrecher, J. P. Olson, D. Englund, and J. Carolan, "Quantum optical neural networks," *npj Quantum Information*, vol. 5, no. 1, pp. 1–9, 2019. [Online]. Available: http://dx.doi.org/10.1038/s41534-019-0174-7

[62] R. Loudon, *The quantum theory of light*. OUP Oxford, 2000.

[63] "Memoization with decorators," https://www.python-course.eu/python3_memoization.php, accessed: 2021-05-25.

[64] D. M. Hawkins, "The problem of overfitting," *Journal of chemical information and computer sciences*, vol. 44, no. 1, pp. 1–12, 2004.

[65] M. Jeruchim, "Techniques for estimating the bit error rate in the simulation of digital communication systems," *IEEE Journal on selected areas in communications*, vol. 2, no. 1, pp. 153–170, 1984.

[66] A. Ajiboye, R. Abdullah-Arshah, and Q. Hongwu, "Evaluating the effect of dataset size on predictive model using supervised learning technique," *International Journal of Software Engineering and Computer Sciences*, 2015.

[67] E. Gooskens, F. Laporte, C. Ma, S. Sackesyn, J. Dambre, and P. Bienstman, "The wavelength dimension in waveguide-based photonic reservoir computing," *Optics Express*, vol. 1, no. 1, p. 10, 2021.

[68] "How to measure and reset a qubit in the middle of a circuit execution," https://www.ibm.com/blogs/research/2021/02/quantum-mid-circuit-measurement/, accessed: 2021-05-24.

[69] "Quantum information 2019 by prof. gabriel teixeira landi," http://www.fmt.if.usp.br/~gtlandi/courses/qinfo19.html, accessed: 2021-05-24.

[70] N. Killoran, T. R. Bromley, J. M. Arrazola, M. Schuld, N. Quesada, and S. Lloyd, "Continuous-variable quantum neural networks," *Physical Review Research*, vol. 033063, pp. 1–22, 2019.

[71] K. Vandoorne, P. Mechet, T. Van Vaerenbergh, M. Fiers, G. Morthier, D. Verstraeten, B. Schrauwen, J. Dambre, and P. Bienstman, "Experimental demonstration of reservoir computing on a silicon photonics chip," *Nature Communications*, vol. 5, pp. 1–6, 2014. [Online]. Available: http://dx.doi.org/10.1038/ncomms4541

[72] F. A. Araujo, M. Riou, J. Torrejon, S. Tsunegi, D. Querlioz, K. Yakushiji, A. Fukushima, H. Kubota, S. Yuasa, M. D. Stiles *et al.*, "Role of non-linear data processing on speech recognition task in the framework of reservoir computing," *Scientific reports*, vol. 10, no. 1, pp. 1–11, 2020.

[73] G. Lindblad, "A general no-cloning theorem," *Letters in Mathematical Physics*, vol. 47, no. 2, pp. 189–196, 1999.

[74] N. Killoran, T. R. Bromley, J. M. Arrazola, M. Schuld, N. Quesada, and S. Lloyd, "Continuous-variable quantum neural networks," *Phys. Rev. Research*, vol. 1, p. 033063, Oct 2019. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevResearch.1.033063

[75] S. Dasgupta, K. E. Hamilton, P. Lougovski, and A. Banerjee, "Designing a nisq reservoir with maximal memory capacity for volatility forecasting," *arXiv preprint arXiv:2004.08240*, 2020.

[76] J. Nokkala, R. Martínez-Peña, G. L. Giorgi, V. Parigi, M. C. Soriano, and R. Zambrini, "Gaussian states of continuous-variable quantum systems provide universal and versatile reservoir computing," *Communications Physics*, vol. 4, no. 1, pp. 1–11, 2021. [Online]. Available: http://dx.doi.org/10.1038/s42005-021-00556-w

[77] C. S. Hamilton, R. Kruse, L. Sansoni, S. Barkhofen, C. Silberhorn, and I. Jex, "Gaussian boson sampling," *Physical review letters*, vol. 119, no. 17, p. 170501, 2017.

[78] J. Dambre, D. Verstraeten, B. Schrauwen, and S. Massar, "Information processing capacity of dynamical systems," *Scientific reports*, vol. 2, no. 1, pp. 1–7, 2012.

[79] T. Haug, K. Bharti, and M. S. Kim, "Capacity and quantum geometry of parametrized quantum circuits," 2021.

[80] L. G. Wright and P. L. McMahon, "The capacity of quantum neural networks," *Optics InfoBase Conference Papers*, vol. Part F181-CLEO-AT 2020, pp. 1–12, 2020.