Denoising with Quantum Machine Learning

Joséphine Pazem

Information
Band / Volume 82
ISBN 978-3-95806-641-0

Mitglied der Helmholtz-Gemeinschaft

JÜLICH
Forschungszentrum

Schriften des Forschungszentrums Jülich
Reihe Information / Information Band / Volume 82

Forschungszentrum Jülich GmbH
Peter Grünberg Institut (PGI)
Theoretische Nanoelektronik (PGI-2/IAS-3)

Denoising with Quantum Machine Learning

Joséphine Pazem

Schriften des Forschungszentrums Jülich
Reihe Information / Information Band / Volume 82

ISSN 1866-1777 ISBN 978-3-95806-641-0

Bibliografische Information der Deutschen Nationalbibliothek.
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der
Deutschen Nationalbibliografie; detaillierte Bibliografische Daten
sind im Internet über http://dnb.d-nb.de abrufbar.

Herausgeber und Vertrieb:	Forschungszentrum Jülich GmbH Zentralbibliothek, Verlag 52425 Jülich Tel.: +49 2461 61-5368 Fax: +49 2461 61-6103 zb-publikation@fz-juelich.de www.fz-juelich.de/zb
Umschlaggestaltung:	Grafische Medien, Forschungszentrum Jülich GmbH
Druck:	Grafische Medien, Forschungszentrum Jülich GmbH
Copyright:	Forschungszentrum Jülich 2022

Schriften des Forschungszentrums Jülich
Reihe Information / Information, Band / Volume 82

D 82 (Master RWTH Aachen University, 2022)

ISSN 1866-1777
ISBN 978-3-95806-641-0

Vollständig frei verfügbar über das Publikationsportal des Forschungszentrums Jülich (JuSER)
unter www.fz-juelich.de/zb/openaccess.

Ich versichere, dass ich die Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Aachen, den

Aknowledgements

I would like to warmly thank my supervisor, Mohammad Ansari, for his kind guidance and availability for regular, fruitful, and instructive discussions. His support throughout my master?s project has fostered my growth and interest in completing this project. He nurtured my curiosity by introducing me to the world of scientific research and the physics community. My group members were also of great help during our weekly group meetings and occasional get-togethers: thank you to Alwin, Xuexin, Arne, Manab, Zhongyi, Tobias, Jennifer, and Jiaqi. Our former group member, Pia, also guided me in my first steps in the project and taught me many of my skills in coding. Thanks to exchanges with Polina Feldmann and Dmitri Bondarenko, Maria Schuld, and Nana Liu, this project could also progress.

I am also grateful to my examiners for their interest and stimulating discussions in my project.

This work would not have been possible without the care and support from my friends and family. I would like to thank Leicai for always lending an ear and encouraging me to gain confidence in my work, my parents for their loving care and attention in difficult moments, and sharing my ups and downs in this project. I am incredibly grateful to my sister, Tatiana, for reaching out to me and always providing unconditional support.

Abstract

This master thesis explores aspects of quantum machine learning in the light of an application to dampen the effects of noise on NISQ processors. We investigate the possibility of designing machine learning models that can be accommodated entirely on quantum devices without the help of classical computers. With Dissipative Quantum Neural Networks, we simulate a quantum feed-forward neural network for denoising: the Quantum Autoencoder. We assign it to correct bit-flip noise in states that can exist only quantum mechanically, namely the highly entangled GHZ-states. The numerical simulations report that the QAE can recover the target states up to some tolerance threshold on the noise intensity. To understand the limitations, we investigate the mechanisms behind the completion of the denoising task with quantum entropy measures. The observations reveal that the latent representation is key to reconstructing the desired state in the outputs. Consequently, we propose an inexpensive modification of the original QAE: the brain box-enhanced QAE. The addition of complexity in the intermediate layers of the network maximizes the robustness of the QAE in a setting where only a finite-size training data set is available. We close the argument with a discussion on the generalization properties of the network.

Contents

Introduction

"Quantum theory is the soul of theoretical physics. It is not just a theory of specific physical systems but a new framework with universal applicability."

G.M.D'Ariano, G.Chiribella, P.Perinotti [1]

One of the significant challenges in the quest for a universal quantum computer is noise on the bit states. A reason for this is the essence of the information that is manipulated on these devices?namely, its quantum properties. Quantum theory offers new ways to process information potentially advantageous to classical methods. It provides a new framework for a compact representation of information in superposition states and makes it possible to handle exponentially large data sets. These quantum properties are fundamental in accelerating algorithms on complex computational problems. Nevertheless, quantum information has the downside of being short-lived and fragile: a slight disturbance is mainly detrimental to its wave function. Due to its quantum nature, the qubits that convey quantum information interact with spurious noise sources. Unwanted entanglement and interactions with the environment lead to the drastic reduction of the lifetime of quantum states. When this issue represents a daunting task, machine learning could be an efficient approach to automate perfecting qubit states. It has proven helpful to find particular classical states in many-body problems. Similarly, quantum states search can be automated by quantum machine learning. The gates on NISQ (Noisy Intermediate-Scale Quantum) devices [2] are yet far from perfect, and such algorithms must be implemented in a noise-tolerant way. Machine learning techniques are sometimes referred to as the most promising invention in computer science. They have evolved to handle problems of increasing complexity. Therefore, implementing machine learning with quantum computations could jointly harness the power of quantum information and machine learning. After translating these techniques into the language of quantum theory, the realization with fault-tolerant gates paves the way for the proof of quantum advantage.

This thesis attempts to contribute to developing a programmable quantum computer. It adopts a holistic view where the system performs better than when it is divided into separated building blocks. When Feynman first revealed his idea of a quantum computer in 1981 [3], he already recognized that the realization of such a quantum machine would challenge research and science across disciplines. Quantum machine learning can be seen as the fusion of outstanding thoughts formulated in the 20th century. The stepping

stone of quantum mechanics was laid by physicists such as Max Planck and Einstein to cope with the catastrophic mismatch between black body radiation and classical theory. It was followed by a constant effort to improve, remodel and reformulate the new paradigm, involving Schrödinger, Hilbert, Heisenberg, Born, von Neumann, and so many more whose names are engraved in quantum theory. About twenty years later, Shannon laid the foundation for a theory of information [4], giving the community the tools for the analytical treatment of data. Another two decades later, the concept of the perceptron emerged in the unexpected field of psychology [5]. Twenty years after Feynman's proposition, the first qubits were realized [6] and quantum algorithms with an advantage over classical computers were invented.

We can now combine these beautiful advances into a unified approach to progress toward fault-tolerant quantum hardware. Nowadays, quantum computations and simulations can work on platforms that accommodate and manipulate qubits. Running actual algorithms on them, however, remains challenging. Even though the question of real quantum advantage is still open, the rivalry between quantum and classical computers cannot be addressed equally. The noise impinging upon quantum systems makes it impossible to run algorithms on large scales in current devices. By adding more qubits, unexpected entanglement becomes stronger. The quantum nature of the hardware is to be blamed for such imperfections. As time goes by, it distorts the information encoded on qubits. The same superposition and entanglement expected to speed up the information processing can be harmful. The qubits cannot be isolated from each other [7] and from external noise [8]. Noise is caused by both spurious interactions in the physical systems and imprecise controls. Overcoming the former requires a new system designed to remove noise sources based on accurate identification. This way, qubit states become more isolated from external environments and matter imperfections. Quantum control theory elaborates techniques such as the Characterization, Verification, and Validation (CVV) method to reduce systematic noise. It takes noisy circuits as inputs to an algorithm that optimizes the controls without altering the underlying physics. Reducing errors on the current quantum devices is a challenge since the fabrication of new platforms is costly. While quantum error correction codes exist, the devices available do not reach the suitable scales to accommodate them. Meanwhile, error mitigation and noise cancellation protocols attenuate noise effects. They consist of both classical and quantum techniques. They bring the NISQ devices closer to achieving fault-tolerant quantum computing.

When most control theory techniques focus on reducing leakage and dephasing in multi-qubit states, we employ a machine learning technique to address noise on more extensive hardware. We take advantage of the data coming directly from the device to train the model. Ultimately, the circuit learns how to produce some delicate target state on a subset of its qubits. This hardware-friendly method is expected to perform despite noise on qubits and control gates.

This thesis is structured as follows: the first chapter discusses the basis for classical machine learning. We explain the concept of neural networks and give an algorithmic meaning to the idea of learning. Moreover, we introduce two models that enable the

implementation of neural networks on a quantum processor, namely the Variational Quantum Algorithms (VQA) and the Dissipative Quantum Neural Networks (DQNN). The second chapter applies quantum neural networks to denoise corrupted states with the Quantum Autoencoder (QAE) architecture. Starting with a short overview of the challenges of noise correction, we show that the QAE can filter noise out of its inputs, thereby facilitating noise cancelation on quantum hardware. In addition, to move away from the theory of classical machine learning, we propose an alternative, inherently quantum measure to point out the role of quantum properties in the denoising process: the quantum Rényi entropy. Finally, in a third and final chapter, we examine the limitations of the Quantum Autoencoder and show the importance of its intermediate representation to recover desired states on its outputs. We show that an absolute tolerance to noise stems from the amount of data available to train the system. This upper bound can be saturated by choosing a suitable structure for the intermediate representation. As a final argument, we discuss the meaning of generalization for Quantum Autoencoders and propose a cross-testing approach.

Chapter 1

From classical to quantum: Introduction to Quantum Machine Learning

1.1 What is machine learning?

1.1.1 A (very) short overview of classical machine learning

Machine learning has become a tool used across disciplines with various applications. It encompasses a large ensemble of computational and statistical methods and algorithms to learn a task on unknown data. Arthur Samuel created the term in his 1959 paper [9] on a program to play checkers: "Lacking such knowledge [about machine learning techniques], it is necessary to specify methods of problem-solving in a minute and exact detail, a time-consuming and costly procedure. Programming computers to learn from experience should eventually eliminate the need for much of this detailed programming effort.". One of the simplest examples is the recognition of handwritten patterns. For example, the digits contained in the MNIST library [10] is a common benchmark for machine learning models. Some algorithms can produce pictures of unknown people [11]. More interestingly, for physics, they can generate some new optics experiments based on the entanglement analysis of quantum states [12], and classify phases of matter [13]. These techniques provide information about physical features inaccessible in the current state of the art.

Machine learning is trained to solve two problems: 1) classification tasks assign labels to unknown data based on its analysis. Such issues range from binary to multi-label classification. 2) Regression finds the best fit for relations between variables and makes predictions on new data sets. The interpretation of regression as classification over continuous classes reconciles both problems [14]. In the following, we will mainly explain machine learning from the classification angle for simplicity, but both are valid approaches for the same task.

Machine learning is commonly separated into supervised, unsupervised, and rein-

forcement learning. Supervised learning trains an algorithm with pairs of data points. One presents new examples, and the other its solution, for example, the corresponding label in a classification task. Even though supervised learning is compelling and can lead to significant advances in sciences, its training requires immense efforts to build large data sets where each sample is carefully associated with its target. Unsupervised learning attempts to circumvent this problem and identifies patterns in the data itself to group points in a common category. Reinforcement learning is famous for its ability to play complex games, such as Go or Chess. It elaborates strategies by adapting its policy based on the rewards and penalties it gets from its actions.

1.1.2 Deep Learning with neural networks

In the collective imagination, machine learning is often merged with the concept of deep learning and summons the picture of the so-called "artificial neural networks". It dates back to 1957 when in his psychology research on artificial intelligence, Rosenblatt [5] introduced the notion of perceptrons. It was defined as a functioning object that mimics a neuron in the brain. A biological neuron processes the stimuli it receives. It reacts with the "firing response" corresponding to a signal emitted by a small voltage, or its absence. Artificial neural networks of perceptrons translate this scheme in mathematical terms and implement it algorithmically on a programmable computer.

Let us look at the perceptrons first, and then at the operations they enable in a neural network. A perceptron is represented in figure 1.1. Its main goal is to output a signal that encodes a solution to the task. The inputs of the perceptron are d-dimensional vectors $\{\overrightarrow{x^j} = (x_1^j, \cdots, x_d^j)\}$ belonging to the set X. They correspond to the initial stimuli in biological neurons. Formally, their components are combined as a single input by a linear transformation $z = \sum_i w_i x_i^j + b$. The weights w_i are real scalars tuned to attain the optimal response. The weighted sum enters the processing unit, the node. Mathematically, it is the input of a non-linear activation function $f(z) = y$. Its output y stands for the firing response and is encoded on the output neuron. In such perceptrons, the training optimizes the weights and biases to process the data according to the task. When properly trained, the network should generalize well to unknown data. The computational meaning of the training is explained in the next section 1.1.3.

Let us look at a classification task with two categories. The Heaviside step function is an ideal non-linear function to provide binary outputs for each neuron: $\theta(z) = 0$ when $z \leq 0$, $\theta(z) = 1$ otherwise. This function corresponds to the integrate-and-fire mechanism in biological neurons: it fires when the signal is larger than a given threshold. The training modifies the weights in the linear transformation. In turn, the form of the activation function f remains fixed. Once the learning phase ends, for a data vector $\overrightarrow{x}$, the neuron is expected to output either 0 if it belongs to category A, and 1 otherwise:

$$f(\overrightarrow{x}) = \theta(z = \sum_i w_i x_i + b) = \begin{cases} 0 & \text{if } \overrightarrow{x} \in \text{category A} \\ 1 & \text{if } \overrightarrow{x} \in \text{category B} \end{cases} \tag{1.1}$$

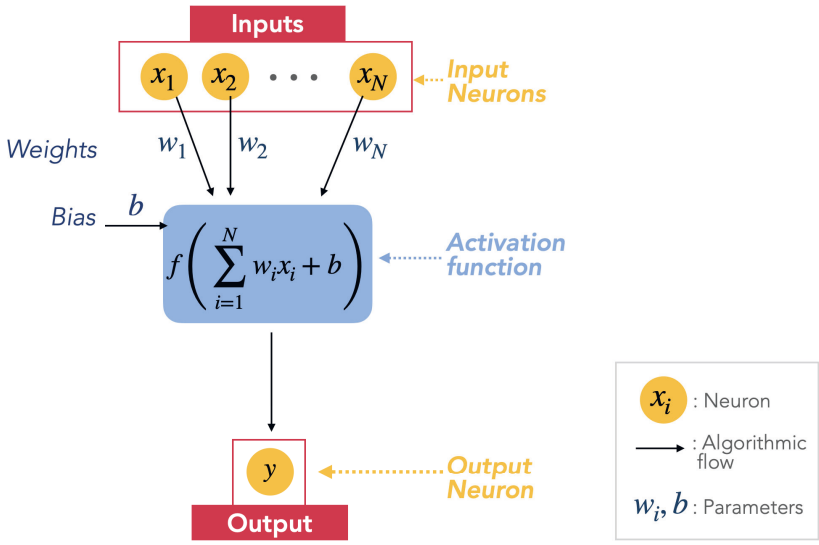


Figure 1.1: A perceptron is the building unit of a neural network: it simulates the action of a biological neuron. It receives an input encoded in neuron units and processes it by computing a biased linear combination of its components and feeding the result in the activation function. In turn, the latter presents its output on a new neuron unit.

To implement complex tasks, multiple perceptrons are brought together to form a neural network, as depicted in figure 1.2. They are generally organized in layers, with various neurons (or perceptrons) each. In Feed-Forward Neural Networks (FFNN), only neurons in two successive layers are linked. No connections are allowed within the same layer or the earlier ones. The number of layers is called the depth of the network. The width designates the number of neurons per layer and can vary from one layer to another. In such a layout, the first layer, or "input layer?? is initialized with the data. It is then processed through the successive layers of perceptrons. The final output is read out on the last layer, or "output layer". For each neuron in the intermediate layers, the inputs correspond to the previous layer's outputs; similarly, a stimulus is propagated from one neuron to another via neurotransmitters and receptors in the nervous system and the brain. The intermediate layers whose outputs remain unknown are named the hidden layers. A perceptron alone can only compute a limited set of functions since it only entails a single non-linear function. However, the combination of neurons into a network becomes a powerful algorithm. Indeed, a network with a single hidden layer can approximate any function to arbitrary precision, provided enough perceptrons in the hidden layer [15]. Therefore, neural networks can perform universal computations.

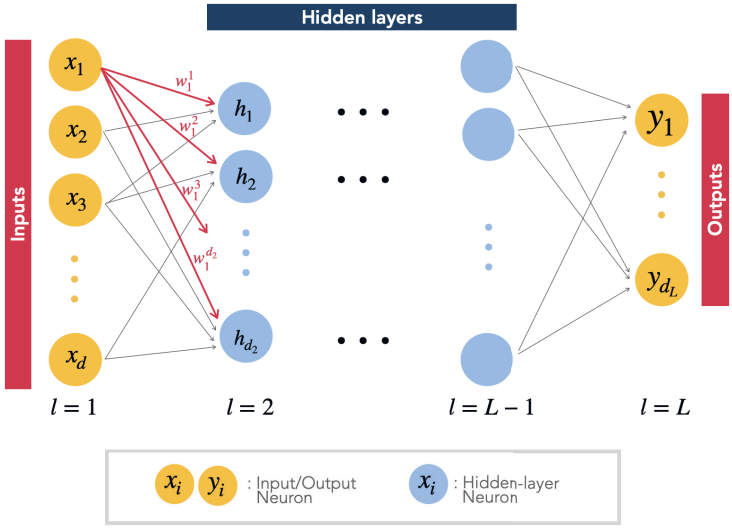


Figure 1.2: An FFNN is composed of L layers of perceptrons, one layer serving as the input for the next by processing its inputs, except for the first and last layers, also known as input and output layers. The hidden layers have unknown outputs. The depth in this network is L, and the width of a layer l is d_l, corresponding to the number of neuron units in the layer.

1.1.3 Algorithmic meaning of learning: gradient descent

We explain the meaning of learning in an algorithmic setting. This section paves the way for the translation of neural networks to the quantum framework. So far, learning was formulated as tuning weights and biases. The update technique itself was omitted and how the network understands the task. To fill these gaps, we introduce the cost function $C(\{\vec{w_i}, b_i\}, \{\vec{x}^j\})$, also referred to as loss or objective function. The notation $\vec{w_i}$ designates the vector of weights for the inputs of neuron i, while $\vec{x}^j$ corresponds to the j-th data point. The cost function encodes the task to complete in the form of an optimization problem. The parameters $\{\vec{w_i}^*\}$ that solve the task minimize the cost function:

$$\vec{w_i}^* = \arg\min_{\vec{w_i}} C(\{\vec{w_i}\}, \{\vec{x}^j\}) \tag{1.2}$$

where the biases b_i have been merged in the weight vector $\vec{w_i}$. This trick necessitates the addition of a $d+1$-th component to the data vector $\vec{x}^j$, that is set to 1. The exact form of the cost function is flexible: common choices are the quadratic and the sigmoid functions. A smooth function of the weights and biases is desirable to implement iterative learning techniques [16]. Indeed, the gradient descent method presented below relies on continuous, differentiable cost functions. Steep gradients can accelerate the training and

avoid the vanishing gradient trap. Most importantly, the objective function must encode the solutions to the problem in its minima.

To reconnect with the classification task in equation 1.1, a textbook example is the mean squared distance between the predicted labels and the actual labels known from the training data: $C(\{\vec{w_i}, b_i\}, \{\vec{x}^j\}) = 1/N \sum_{j=1}^{N} ||\vec{y}^j - \hat{y}^j||^2$, where $\vec{y}^j$ is the label given by the algorithm for a corresponding data point $\vec{x}^j$, and $\hat{y}^j$ is the estimated label. The distance is the Euclidian norm: $||\vec{v}|| = \sqrt{\sum_i v_i^2}$. The cost function penalizes erroneous outputs by making them more positive. The cost evaluation as a function of parameters defines a cost landscape with hills and valleys. In this picture, the value of the cost function during the optimization is equivalent to a ball rolling downhill: the optimal combination of parameters lies at the bottom of a valley, where the cost function is minimal. Gradients with respect to the parameters guide the exploration of the landscape to find this solution.

This is the approach adopted by the gradient descent algorithm. The intuition behind it is the following. For simplicity, we omit the arrows in the parameters $\vec{w_i}$. In the cost landscape, the gradient $\nabla_{w_i} C$ points in the direction where the function increases when w_i increases by a small amount. Therefore, to land on smaller values, one must move opposite to the gradient, that is along $-\nabla_{w_i} C$. More formally, at the training step t, the update is given by:

$$w_{i,t+1} \leftarrow w_{i,t} - \eta \nabla_{w_{i,t}} C \tag{1.3}$$

where η is a positive real number that scales the update. This learning rate is a sensitive hyperparameter. A small value of η reduces the update's size and slows down the convergence to a minimum. It may render the algorithm inefficient. In contrast, choosing large values for η fosters the exploration of the landscape, but causes overshooting: the update jumps over the minimum without reaching it. The learning rate is tuned by hand for each network. The algorithm does not update it in 1.3. Still, it can be designed to vary throughout the training according to a predefined schedule and adapt to the training data.

Gradient descent is a powerful method and has shown convergence to a (local) minimum for (non-)convex landscapes in most applications, provided that the function is differentiable. However, large data sets lead to computational difficulties. Calculating gradients for each data point at each training step becomes an expensive task. The vanilla gradient descent, also called batch or deterministic gradient descent [17], entails unnecessary redundancy when the training data set possesses similar data points. Therefore, alternative algorithms seek more efficient uses of the data. Stochastic gradient descent (SGD) computes the gradients based on a single data point at a time and accelerates the learning process considerably. It comes at the expense of high variance in the learning curve, which can cause overshooting. Mini-batch gradient descent emerges from the compromise between the two methods. It uses a subset of data points for each training step, thereby smoothing the learning curve while preserving the desired speed up ([17], chapter 8).

To lay some foundations for the machine learning model we will introduce in chapters 2 and 3, some deeper insights on gradient descent algorithms are presented here. A comprehensive and operational overview of them is provided in [18]. Indeed, despite the notable gain in speed and precision in SGD or mini-batch gradient descent, the algorithms can be improved by bringing two modifications. First, the method to tune the hyperparameter η is still unclear: one must do a trade-off between speed and fluctuations, and in 1.3 the learning rate remains constant at all times. At the same time, a training schedule could make it more adaptive and ensure convergence to a minimum. Second, the data properties are still under-exploited, for the learning rate is the same for all weights. However, sparse data would require larger updates in the direction of rare points, and damping in the most recurrent ones. This is the reason why we use the NADAM gradient descent method [19, 18] to train the network in chapters 2 and 3.

We first attempt to provide some intuition on the NADAM algorithm. To make learning more efficient, it combines multiple techniques: momentum [20], ADAM (Adaptive Moment estimation) gradient descent [21], as well as Nesterov accelerated gradient descent [22]. It uses information on past gradients' first and second moments for a given parameter. It accelerates the updates in the direction of the minimum and tunes learning rates independently for each weight. At the same time, it enables the implementation of a learning schedule that modifies the learning rates adaptively. It exploits most of the information at hand at step t to compute gradients based on estimating the future parameter value at step $t+1$. If we label the parameter update Δw_t at the training step t, the NADAM rule gives:

$$\begin{aligned} w_{t+1} &= w_t + \Delta w_t \\ \Delta w_t &= -\frac{\eta}{\sqrt{\hat{v}_t}+\varepsilon}\left(\beta_1 \hat{m}_t + \frac{(1-\beta_1) g_t}{1-{\beta_1}^t}\right) \end{aligned} \tag{1.4}$$

where η and β_1 are hyperparameters, ε is a small constant to regularize division by zero. The recommended values for them are $\eta = 0.002$ and $\beta_1 = 0.9$. The hyperparameter β_2 hidden in the variance $\hat{v}_t$ in equation 1.5c is generally set at $\beta_2 = 0.999$. We now decompose and explain the role of each component of the rule. The three important terms are:

Gradient vector:
$$g_t = \nabla_{w_t} C(w_t) \tag{1.5a}$$

Momentum (first moment):
$$\begin{aligned} \hat{m}_t &= \frac{m_t}{1-{\beta_1}^t} \\ &= \frac{1}{1-{\beta_1}^t}\left(\beta_1 m_{t-1} + (1-\beta_1)\cdot g_t\right) \end{aligned} \tag{1.5b}$$

Variance (second moment):
$$\begin{aligned} \hat{v}_t &= \frac{v_t}{1-{\beta_2}^t} \\ &= \frac{1}{1-{\beta_2}^t}\left(\beta_2 v_{t-1} + (1-\beta_2)\cdot g_t^2\right) \end{aligned} \tag{1.5c}$$

Let us go through these terms one by one. In equation 1.5a, we recognize the same gradient as in the vanilla gradient descent 1.3. In the update rule 1.4, it appears weighted by the usual learning rate η, as well as the variance term. The second term 1.5b is defined recursively: at each step t in the training, the new momentum is a linear combination of the past momenta and the gradient. It corresponds to a decaying running average of the past gradients. It is designed to help optimize asymmetric slopes in the landscape, such as ravines or saddle points, where the slope is steep in some directions, and shallow in others. If the directions of the previous momentum and the new gradient are identical, the update step along this direction is larger, while opposite directions dampen the amplitude of the update. Hence, the momentum term reduces fluctuations in the optimization and adopts a more direct path to the minimum. Finally, the variance term 1.5c provides the algorithm with an adaptive learning rate. It is defined recursively as well. Instead of accumulating gradient terms, it computes the decaying, running average of the squared gradients, corresponding to the second moment. When the gradients are far apart from one another, the variance becomes large and the next gradient is damped, while when they tend to become too small, the variance boosts the future gradients to prevent them from vanishing. The renormalization pre-factor in 1.5c depends explicitly on the "discrete time" t: as the training progresses and t increases, the denominator comes closer to unity, canceling the amplifying effect it had in the early learning steps. This automatic variance scaling implements a clear learning schedule without manual programming. It builds upon the Adagrad method and solves the problem of vanishing updates at the end of the training. The variance term can be understood as keeping only squared gradients for a few training steps in the past and discarding the previous ones. This adaptive rule cancels the biases due to the initialization at zero of the momentum and variance. Note that the gradient and momentum are vectors: the update rule has become different for each weight w_i.

A final point is left to understand in the training rule: the inclusion of future estimates for the parameters to improve the direction of the update. Nesterov accelerated gradients aim at improving the Momentum method by building the update with an approximation of the next position of the weight. It is seen by comparing the formulas for the "plain" momentum and Nesterov accelerated gradient methods:

$$\text{Momentum method:} \quad \Delta w_t = m_t = \gamma m_{t-1} + \eta \cdot \nabla_{w_t} C(w_t) \tag{1.6}$$

$$\text{Nesterov accelerated gradients:} \quad \Delta w_t = \gamma m_{t-1} + \eta \cdot \nabla_{w_t} C(w_t - \gamma m_{t-1}) \tag{1.7}$$

$$\text{and the update:} \quad w_{t+1} = w_t - \Delta w_t \tag{1.8}$$

In this case, m_{t-1} corresponds to the weighted accumulation of gradients up to step t. In equations 1.6 and 1.7, the gradient is calculated at different positions of the parameters. In Nesterov's method, it is computed at the estimated position in the next step, approximated by the difference of the last parameter with the momentum m_{t-1}: it only differs from the actual parameter at step $t+1$ by the gradient term. One can include this future prediction in the NADAM update rule by computing the gradient at the predicted parameter instead of its actual value. This however requires one to compute the momentum term twice, in equation 1.7, and Dozat [19] notices that one

can instead implement the following trick. Looking at the update rule 1.4 without the prediction, (corresponding to the 'ADAM" rule [21]):

$$\begin{aligned}
\hat{m}_t &= \frac{1}{1-{\beta_1}^t}\left(\beta_1 m_{t-1} + (1-\beta_1) g_t\right) && (1.9)\\
\Delta w_t &= -\frac{\eta}{\sqrt{\hat{v}_t}+\varepsilon}\hat{m}_t \\
&= -\frac{\eta}{\sqrt{\hat{v}_t}+\varepsilon}\left(\beta_1 \frac{m_{t-1}}{1-{\beta_1}^t} + (1-\beta_1)\frac{g_t}{1-{\beta_1}^t}\right) \\
&= -\frac{\eta}{\sqrt{\hat{v}_t}+\varepsilon}\left(\beta_1 \hat{m}_{t-1} + (1-\beta_1)\frac{g_t}{1-{\beta_1}^t}\right) && (1.10)\\
&\xrightarrow{\text{Nadam}} -\frac{\eta}{\sqrt{\hat{v}_t}+\varepsilon}\left(\beta_1 \hat{m}_t + (1-\beta_1)\frac{g_t}{1-{\beta_1}^t}\right) && (1.11)
\end{aligned}$$

where in the last line 1.11, the renormalized momentum in step $t-1$ has been swapped for the updated one at step t to boost the ADAM update rule 1.10 with Nesterov acceleration.

We make a short remark here. Even though the gradient descent algorithm should ultimately converge towards a minimum or a good approximation of it after enough update steps, it is expensive to implement in practice in a neural network. Indeed, the cost function derivative concerning the weights is not trivial to compute. For this reason, the idea of a neural network was partly forgotten by the community during the "winter of machine learning". It was only in 1986 that Rumelhart, Hinton, and Williams revived the backpropagation algorithm [23] and made training possible. Based on the chain rule for derivatives, this technique calculates gradients using only a single forward pass through the network, keeping the outputs at each layer in memory, and deduces the gradient with a single backward pass.

1.1.4 Future challenges in machine learning

So far, we have discussed one of the most famous machine learning techniques: neural networks. In this framework, we have understood that gradient descent, combined with the backpropagation algorithm, provides networks with the ability to learn. Their outcome can be impressively close to the solutions inquired. However, machine learning is not limited to neural networks and encompasses various algorithms, some free of gradients. Rather than optimizing parameters this way, some techniques keep the memory of the training data and realize their task by comparing unknown data to their memory. For example, the k-nearest-neighbors algorithm [24] classifies new data according to their proximity to an identified data point. Another approach simulates agents' behavior, building policies to guide their actions. Indeed, in reinforcement learning, the algorithm reacts to rewards it receives from implementing a particular set of policies [25]. Therefore, machine learning encompasses a wide range of algorithms whose applications are up-and-coming.

However, machine learning tools are often implemented as black boxes despite this effervescence and diversity and remain an enigma from many perspectives. Their success in learning is often still obscure, especially when the models gain in complexity, some models leveraging millions of parameters. Consequently, some authors [26] emphasize the lack of understanding of the learning dynamics in neural networks. For example, understanding the reason behind the presence of vanishing gradients in a general setting would solve one of the pressing issues of gradient-based learning.

Phenomenological rules have been discovered for the dependence of the learning on initial parameters and hyperparameters, but few general or analytical properties are advertised. This slows down progress for the choice of initialization impacts the final solution, different parameters generally resulting in very different solutions [27]. These points are also related to a "good" data set. Complex models require large data sets, making them expensive to train and implement. However, their amount alone is no guarantee for successful training. The training set must provide the network with the relevant features to identify to reach a satisfactory generalization. In other words, a balance must be achieved. The set must entail enough diversity so that various patterns while fluctuations must be limited to a specific range to let the algorithm recognize similarities between points. Consequently, the scale and complexity of the models are bottlenecks in the performance of machine learning techniques [27].

1.2 A model for quantum machine learning

1.2.1 Bring quantum mechanics to machine learning

As quantum computing and the theory of quantum information are improving, new hopes are emerging to find more efficient ways to implement machine learning with quantum devices. At first sight, the laws of quantum theory are not incompatible with the framework of machine learning. For example, the execution of a machine learning algorithm relies primarily on the cost function, the resulting parameters, and the training data. The requirement in quantum theory to identify canonical conjugate variables is, therefore, no impediment to its realization. Better than this, the description of quantum states as wave functions implies the creation of superposition states. This opens new possibilities to efficiently encode large amounts of data and process them together. Similarly, entanglement can extend the action of one neuron to other neurons. This delocalization of interactions brings information processing in a neural network beyond classicality. Therefore, the fusion of quantum computing with machine learning, or Quantum Machine Learning (QML) is expected to provide a "quantum speed up", or "quantum advantage" over its classical counterpart. So does Shor's quantum algorithm for factorization compared to all known classical methods [28]. In other words, the hope is to reduce the number of resources required to realize the same task by exploiting quantum phenomena. However, such claims of advantage are usually challenging to prove rigorously. In many cases, one can only compare the complexity of the quantum algorithm with the best known classical for the same task.

QML can fuse machine learning with quantum mechanics in three ways depending on the type of data used and the hardware to process it [14]. Processing quantum data with classical computers is often a difficult task due to the exponential scaling in the dimensions of the data sets required to represent quantum states and the exotic probability distributions that they can realize. In the opposite case, a prerequisite to treating classical data with quantum algorithms is designing a suitable and efficient encoding method to translate it to quantum states. Unfortunately, no such algorithm is known to the author that realizes this conversion efficiently. In these conditions, data encoding overtakes the speed-ups realized in the learning algorithms. Finally, processing quantum data on quantum hardware skips the encoding phase and reunites with the first intention of quantum computing to simulate quantum with quantum. From this point of view, it is of particular interest for fundamental research. However, it is possible only within the limit of large enough, fault-tolerant platforms. Note that quantum platforms can process the data entirely on their own, some approaches even aiming at implementing gradient descent-like algorithms directly at the scale of the physical system [29] to create a self-learning machine, or perform subroutines only, working in pair with a classical computer, as in [30] where only the complex task of kernel computation is delegated to the quantum machine. The latter case is of particular interest for early benchmarking on NISQ devices since such algorithms can work at smaller scales and make error mitigation more accessible thanks to the possibility of implementing post-processing. The following content will focus on the quantum-quantum setting and exploit quantum data directly with a quantum circuit.

1.2.2 Build a Quantum Neural Network

Prepared with a bit of background on machine learning from section 1.1, and with a concrete idea of what learning means in this setting, we attempt to transpose it to the setting of quantum mechanics. The intention is to run such algorithms on quantum computers, provided that a platform can accommodate it. For completeness, we mention that many popular machine learning techniques apart from neural networks have been successfully reformulated for quantum algorithms, the Principal Component Analysis [31] being one example among many others. Here, we focus on creating a quantum version of a feed-forward neural network or a Quantum Neural Network (QNN). To realize the correspondence, we must find:

1. a system that encodes and conveys information in the same way a neuron enables loading and reading data,

2. a non-linear activation function that manipulates information stored in qubits according to a set of parameters,

3. a method for updating these variables based on an objective function.

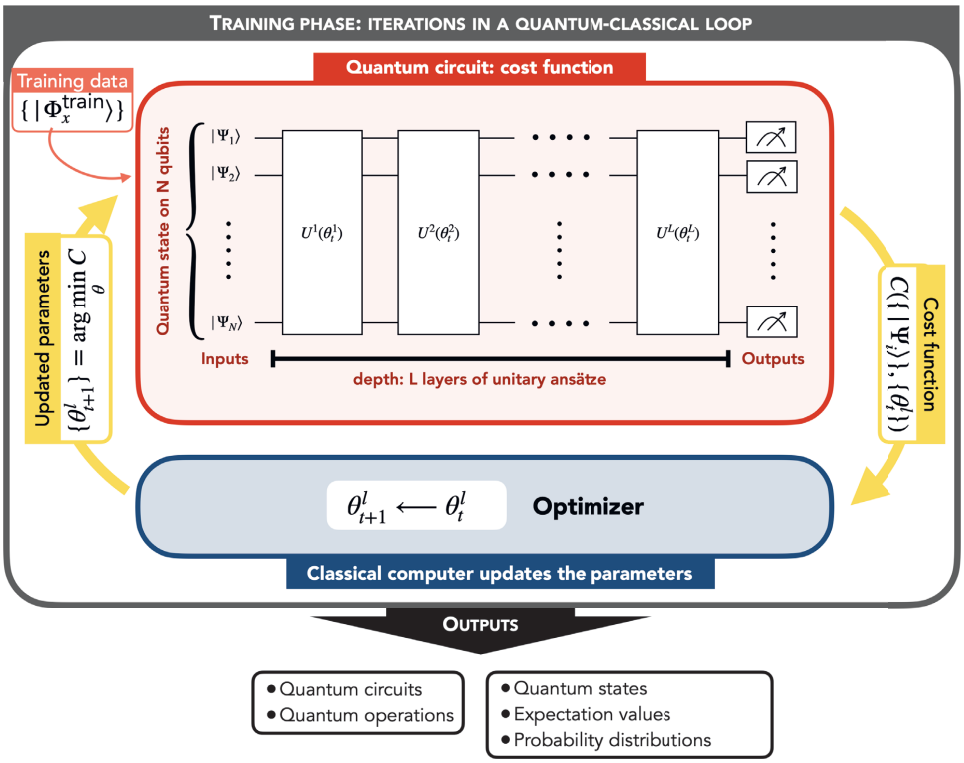


Figure 1.3: The Variational Quantum Algorithm is a hybrid quantum-classical algorithm capable of simulating QNN. The quantum part, depicted in red, acts on some inputs with parametrized quantum gates. It acts on N qubits and applies L layers of unitaries, selected according to some ansatz. The outputs are read out and used to compute the cost function. The optimization takes place on a classical platform using a classical optimizer algorithm: it outputs parameters that minimize the cost function. The updated angles are communicated to the quantum device to modify the gates. This quantum-classical loop is repeated until a stopping criterion is reached. The outputs consist of a trained circuit with optimized parameters or an approximation and a representation of quantum states resulting from applying this circuit to some inputs. These inputs can be purely quantum or correspond to classical data encoded in quantum states.

1.2.2.1 QNN with variational circuits

The ultimate goal is to run the final algorithm partially on a quantum computer. Most quantum models for feed-forward neural networks embed it in a quantum circuit. This

representation of the algorithm facilitates the compilation into physical gate sequences and possible implementations on NISQ devices. In addition, the circuit model for quantum computation already satisfies the first requirement for a neural network: the qubits are systems that can encode information and be manipulated. Measurements enable the read-out of the outputs. Because of the nature of quantum mechanics, reading out the outcomes requires building statistics on the states and running the algorithm multiple times. The results can be represented as expected values of some measured observable or quantum states reconstructed through state tomography. In the latter case, reading out the outputs alone is already a complex task: without accelerating techniques, it scales exponentially in the size of the output state.

To obtain a quantum perceptron and design a network, the gates must be defined with tunable parameters. In their seminal 2018 paper [32], Farhi and Neven propose to use quantum variational circuits to build a quantum FFNN for a classification task. These circuits are also referred to as parametrized circuits. They apply a succession of gates on $N+1$ qubits, where N qubits accommodate the inputs, while the last qubit is an ancilla. The gates, in turn, are defined by unitary matrices, that can be represented using parameters θ, such as rotation angles, for example. For this task, Farhi and Neven use a single qubit to encode the outputs, thereby reducing the number of resources required to calculate expectation values. This model, however, has been used in more flexible forms within the broad family of Variational Quantum Algorithms (VQA) [33, 14]. It is considered one of the most promising applications for fault-tolerant quantum computing. The task is formulated in the cost function, whose minimum defines optimal parameters in the cost landscape. Despite this similarity with classical learning in 1.1.3, the scope of VQAs goes beyond machine learning: it is a popular method in quantum chemistry to identify the ground states of molecules using the Variational Quantum Eigensolver (VQE) [34, 35], or constitutes an approach to combinatorial optimization problems, such as the Quantum Approximate Optimization Algorithm (QAOA) [36], among other applications.

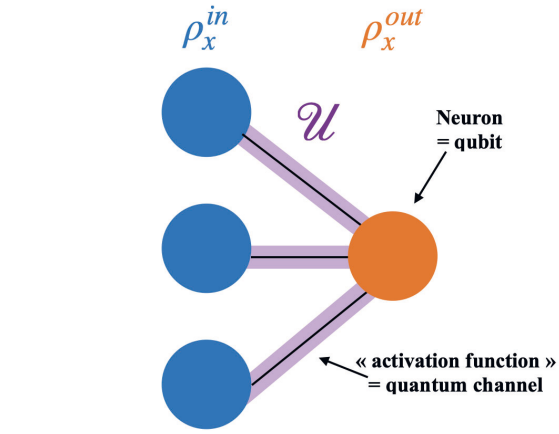
Due to the formulation of the task as the minimization of an objective function and its versatility, the VQA constitutes an appealing candidate for realizing quantum neural networks. As shown in figure 1.3, qubits play the role of neurons. They are connected through parametrized unitaries $U(\theta)$ to be optimized. Compared with classical neural networks, the layers are composed of the unitaries rather than neurons. Each layer can have multiple unitaries, such that the operation applied at layer l composed of N_l unitaries can be conveniently denoted as $U^l(\theta^l) = \prod_{n=N_l}^{1} U_n^l(\overrightarrow{\theta}_n^l)$. As in the classical case, learning is realized during the training phase with an iterative algorithm. The parameters are updated according to the direction of the cost landscape that leads to its minimum. VQA is particularly suitable for problems hosted on a few qubits because it works in a hybrid fashion. The cost function is calculated quantum mechanically by applying the quantum circuit to the inputs: it generally entails the expectation value of a well-chosen observable. However, the updates of the parameters take place on a classical computer using an optimization technique that matches the task's requirements. Hence, during the training, the VQA goes back and forth in a loop between the classical and

quantum computers. In the training round, the data is first encoded on the qubits. Then the successive layers of parametrized unitaries act on the inputs. Once read-out, the outputs are collected to calculate the cost as the parameters and inputs function. The classical optimizer uses this classical information to update the parameters of the unitaries. Moving back to the quantum circuit, the next round can start with the modified unitaries.

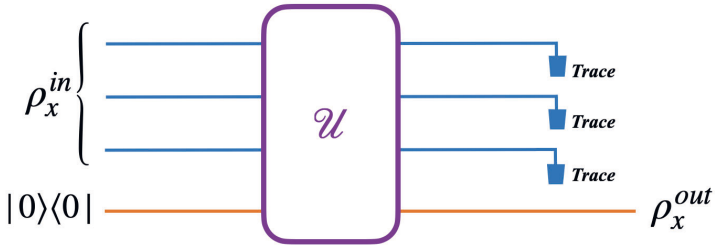
In VQAs, the choice of the unitaries' form determines the circuit's expressivity or universality. Good ansätze must be selected to match the cost function and the device on which the QNN gets trained. An ansatz corresponds to a set of gates whose form is constrained. They are organized according to some layout on the qubits in the network. An ansatz can combine single- and multi-qubit gates and parametrized and fixed gates. For efficiency and to reduce the cost of the simulations, most ansätze use repetition and alternating layers of similar unitaries, with independent or identical parameters. For example, the hardware-efficient ansatz is a popular choice: it resorts to native gates only, thereby reducing the depth of the circuits to implement. In contrast, the variational Hamiltonian ansatz is problem-friendly but loses computational efficiency. It aims to find the ground states of trotterized Hamiltonians: each evolution step is represented by a trained parametrized unitary. As such, it covers a wide variety of physical problems. The hardware-efficient ansatz belongs to the hardware-aware type of ansätze, while the variational Hamiltonian ansatz is problem-aware. A suitable ansatz should enable the model to explore all the directions of the cost landscape to find its minimum. Consequently, the selection of the ansatz is at the center of one of the most notorious problems of quantum machine learning [37]: the gradients vanish in some areas of the optimization landscape. This makes gradient-based optimization impossible and leaves the system stuck in a sub-optimal local minimum. Such areas of the cost landscape are called barren plateaus. Leveraging arguments from quantum control theory, some authors [38] advocate the use of training-aware ansätze instead: they result from the analysis of the coverage of dynamic Lie algebra underlying the model and the ansätze.

1.2.2.2 Dissipative Quantum Neural Networks

To explore the benefits of quantum properties for machine learning in their entirety, the community seeks self-contained QNN models. Their implementation on quantum devices at larger scales is a first step to looking for a quantum advantage. Throughout the remaining chapters, we will focus mainly on the so-called Dissipative Quantum Neural Networks (DQNN) [39, 40]. Even though they still leverage parametrized circuits, their architecture has a straightforward comparison to the classical neural networks introduced in figure 1.2. Qubits and their connections by unitaries still represent the neurons. As shown in figure 1.4 (a), the axone corresponds to a unitary that connects the input qubits in layer l to the single output qubit in layer $l+1$. Instead of describing layers as a group of unitaries defined by some ansatz as in the VQAs, they are composed of the neurons or qubits themselves. This way, the architecture of the network resembles that of FFNN introduced in section 1.1.2 and can be qualified as a multi-layer perceptron.



(a)



(b)

Figure 1.4: Representation of a quantum perceptron for dissipative quantum neural networks. The structure (a) imitates that of a classical perceptron, where qubits represent neurons, and the connections correspond to the application of unitary channels, followed by a suitable trace over the inputs. The perceptron has an equivalent circuit representation shown in (b). Dissipation is caused by the action of discarding the input qubits after applying the quantum channel $\mathcal{U}$ to propagate it forward to the output layer.

Equivalently, quantum circuits can represent the perceptrons in DQNNs as in figure 1.4 (b): such networks are suitable for circuit-based quantum computation. As depicted in figure 1.5, the qubits on the input layer are initialized with the quantum data in quantum states. To propagate the states forwards through the network, unitaries act on the qubits in two successive layers l and $l+1$. More precisely, the layer of unitaries $\mathcal{U}^l$ is composed of a number N_{l+1} of unitary perceptrons that act on all qubits in layer l, and a single qubit in layer $l+1$: $\mathcal{U}^l = \prod_{i=N_l}^{1} U_i^l$, where the unitaries U_i^l have been implicitly

extended by identity operations on the remaining qubits in layer $l+1$. Subsequently, to ensure that the states live on a single layer at a time, the qubits in layer l are discarded by tracing them out, creating dissipation in the forward pass. Then, for an input ρ_x, the propagated state at layer $l+1$ takes the form:

$$\rho_x^{l+1} = \operatorname*{Tr}_{q \in l} \left\{ \mathcal{U}^l (\rho_x^l \otimes |0\rangle\langle 0|^{\otimes N_{l+1}}) \mathcal{U}^{l\dagger} \right\} \tag{1.12}$$

where layer $l+1$ has been initialized in the computational ground state, and all qubits q in layer l are discarded. This ensures that the activation function produces more non-linearity. In the case of VQAs, non-linearity emerged from the Hilbert spaces in which the unitaries were living. Indeed, an individual Hilbert space $\mathcal{H}_i$ is associated to each qubit $i \in \{1, \cdots, N\}$ entering the circuit. Even though quantum mechanics is inherently formulated in terms of linear algebra, acting on multiple qubits with a single operation brings Hilbert spaces together in a tensor product $\bigotimes_{i \in U} \mathcal{H}_i$, thereby creating non-linearity. The addition of dissipation in the DQNN accomplishes non-linearity in a more evident way.

This quantum realization of a perceptron can be readily understood in terms of parametrized circuits. In fact, [40] reconciles this new approach with the concept of VQAs. To find a direct link, the perceptron is represented with a parametrized circuit using one of the ansätze known from variational algorithms. This direct mapping to multiple VQAs enables them to generalize the knowledge about VQAs. They propose guides to design models that avoid barren plateaus [41]. In particular, DQNNs inherit the expressivity of the hardware-efficient ansatz, opening the door to promising applications. In contrast, the "parameter matrix multiplication" optimization [39] parts with the parametrized circuits. It understands each entry of the unitary as a parameter. As the unitary acts on the qubits, all parameters are applied simultaneously. Similarly, they are updated together with single matrix multiplication. In this framework, the unitary operations are not constrained to follow any fixed model: the update rule is free to optimize the parameters in any direction compatible with the initialization choice. A set of random unitaries is generally selected to initialize the perceptrons.

Since the network introduced in the following chapters utilizes the parameter matrix multiplication approach for the perceptrons and training, we explain the learning rule of such QNNs in more detail. The reader can refer to appendix B for a complete derivation of the update formula. The challenge is to express the rule with two constraints: (1) like in gradient descent in section 1.1.3, its computation should root in the cost function, (2) the update must have a formulation that is compatible with quantum mechanics and that can be applied efficiently. The most suitable form of the update is a Hamiltonian-evolution term:

$$U_j^l(t+\varepsilon) \longleftarrow e^{i\varepsilon K_j^l(t)} U_j^l(t) \tag{1.13}$$

to optimize the unitary acting on qubits in layer $l-1$ and qubit j in layer l, at the training step t. In this expression, ε reflects a notion of discrete time and the parameter matrix $K_j^l(t)$ provides information about the gradient, for each unitary $U_j^l(t)$. The latter has a "built-in" learning rate η, as will be shown in the derivation of the update rule.

It depends on the channels $\{\mathcal{U}^l\}_{l=1}^L$ at step t, as well as on the training data $\{\rho_x^{\text{in}}\}_{x=1}^N$ in the training phase. The gradient of the cost function is defined as the time derivative of the cost function:

$$\frac{dC(t)}{dt} = \lim_{\varepsilon \to 0} \frac{C(t+\varepsilon) - C(t)}{\varepsilon}. \tag{1.14}$$

The cost function is expressed as the expectation value of some Hermitian observable $\hat{\mathcal{O}}$ with respect to the density matrix ρ_x^{out} on the output layer, averaged over the training data $x \in \{1, \cdots, N\}$:

$$\begin{aligned} C(\{\rho_x\}, \{U_j^l\}) &= \frac{1}{N} \sum_{x=1}^{N} C_x \\ &= \frac{1}{N} \sum_{x=1}^{N} \underset{\text{out}}{\text{Tr}} \left\{ \hat{\mathcal{O}} \rho_x^{\text{out}} \right\}. \end{aligned}$$

For supervised training, the observable $\hat{\mathcal{O}}$ is taken to be the target state corresponding to input x. In this setting, the data is indeed organized in training pairs $\left\{\rho_x^{\text{in}}, \rho_x^*\right\}$. The target state ρ_x^* must have the same dimension as the output layer of the DQNN, but need not live in the same Hilbert space of the inputs ρ_x^{in}. Since the density matrices for the target states comply with the hermicity requirements of an observable, setting $\hat{\mathcal{O}} = \rho_x^*$ is allowed and results in the cost function being the fidelity:

$$\widetilde{F}(\rho_x^{\text{out}}, \rho_x^*) = \text{Tr} \left\{ \sqrt{\sqrt{\rho_x^*} \rho_x^{\text{out}} \sqrt{\rho_x^*}} \right\}. \tag{1.15}$$

For the task we intend to realize in chapter **??**, we seek pure output states $\rho_x^* = |\Psi_x^*\rangle\langle\Psi_x^*|$. In addition, equation 1.15 causes computational instabilities due to the application of the square root to multiple matrices. Therefore, we simplify the expression of fidelity to:

$$F(\rho_x^{\text{out}}, \rho_x^*) = \langle \Psi_x^* | \rho_x^{\text{out}} | \Psi_x^* \rangle. \tag{1.16}$$

In both expressions, fidelity is valued between 0 and 1 and is maximized for identical states. Therefore, one can either maximize the fidelity during the training or equivalently minimize the reconstruction error $1 - F$.

Finally, the missing element to express the learning rule is the dependence of the cost function upon the parameters, namely the unitaries $\mathcal{U}^l$, which was implicit in the ρ_x^{out}. In equation 1.12, the propagation as a whole constitutes a quantum map $\mathcal{E}^l(\bullet)$ made of unitary gates, and traces over the qubits in the previous layer. The Hilbert space of the inputs is different from the one of the outputs. As a result, the state on layer l for an input x during the forward pass can be written as:

$$\begin{aligned} \rho_x^l &= \underset{1,\cdots,l-1}{\text{Tr}} \left\{ \prod_{k=l}^{1} \mathcal{U}^k \left(\rho_x^{\text{in}} \otimes |0\rangle\langle 0|^{\otimes \sum_{k=2}^{l} N_k} \right) \prod_{k=1}^{l} \mathcal{U}^{k\dagger} \right\} \\ &= \mathcal{E}^l \left(\mathcal{E}^{l-1} \left(\cdots \left(\mathcal{E}^1 \left(\rho_x^{\text{in}} \right) \right) \right) \right) \end{aligned}$$

where we have combined the operations of adding new qubits, applying the unitaries on the qubits in two successive layers and tracing out the previous layer in a single operation: $\mathcal{E}^l(\rho^{l-1}) = \mathrm{Tr}_{l-1}\left\{\mathcal{U}^l\left(\rho_x^{l-1}\otimes|0\rangle\langle 0|^{\otimes N_l}\right)\mathcal{U}^{l\dagger}\right\}$. To access the target states for the optimization of each unitary U_j^l, a backward channel is also defined. Its derivation is given in appendix A. It plays the role of a backpropagation algorithm: it enables one to compute the all the gradients by running only one forward and one backward pass at each iteration. This assumes the existence of a quantum memory. The adjoint channel $\mathcal{F}^l$ propagates backwards states on layer $l+1$ to layer l. The output layer is initialized with the target state ρ_x^*. The resulting state σ_x^l on layer l reads:

$$\sigma_x^l = \mathcal{F}^l(\cdots(\mathcal{F}^L(\rho_x^*))) \tag{1.17}$$

where $\mathcal{F}^l(\bullet)$ can be found analytically from the Kraus decomposition of the forward map $\mathcal{E}^l(\bullet)$, as shown in Appendix A:

$$\mathcal{F}^l(\rho_x^l) = \operatorname*{Tr}_l\left\{\mathbb{I}_{l-1}\otimes|0\rangle\langle 0|^{\otimes N_l}\,\mathcal{U}^{l\dagger}\left(\mathbb{I}_{l-1}\otimes\sigma_x^l\right)\mathcal{U}^l\right\} \tag{1.18}$$

The availability of a backpropagation method makes the training possible. The discrepancy between the target states and the actual outputs guides the update of the parameters. For each unitary U_j^l, the difference corresponds to the mismatch between the inputs propagated forwards up to qubit j in layer l on the one hand, and the target states propagated backward until the same location. Such a discrepancy can be appreciated with a non-zero commutator in quantum mechanics. When trained over a data set $\{\rho_x^{\text{in}}\}_{x=1}^N$, the complete update rule is formulated as:

$$U_j^l(t+\varepsilon) = e^{i\varepsilon K_j^l(t)}U_j^l(t) \tag{1.19}$$

$$K_j^l(t) = \eta\frac{2^{N_{l-1}}}{N}\sum_{x=1}^{N}\operatorname*{Tr}_{q\notin U_j^l}\left\{M_j^l(t)\right\} \tag{1.20}$$

$$M_j^l(t) = \left[\prod_{q=j}^{1}U_q^l(t)\left(\rho_x^{l-1}\otimes|0\rangle\langle 0|^{\otimes N_l}\right)\prod_{i=1}^{j}U_i^{l\dagger}(t)\;,\;\prod_{i=j+1}^{N_l}U_i^{l\dagger}(t)\left(\mathbb{I}_{l-1}\otimes\sigma_x^l\right)\prod_{i=N_l}^{j+1}U_i^l(t)\right] \tag{1.21}$$

where in equation 1.20, q stands for all qubits that are not included in U_j^l.

Together with 1.13, equations 1.20 and 1.21 complete the quantum formulation of a gradient descent algorithm in DQNNs. A summary is shown in figure 1.5. As in the classical case, the update matrix entails a learning rate η that determines the size of the update step. It remains a hyperparameter to tune to control fluctuations and ensure the convergence of the cost function. Furthermore, the matrices M_j^l play the role of the gradient in finding the direction of the update, given by the difference between the current operation, and the ideal output it should provide. In equation (1.21), the first term in the commutator is precisely the state propagated forwards up to the unitary to

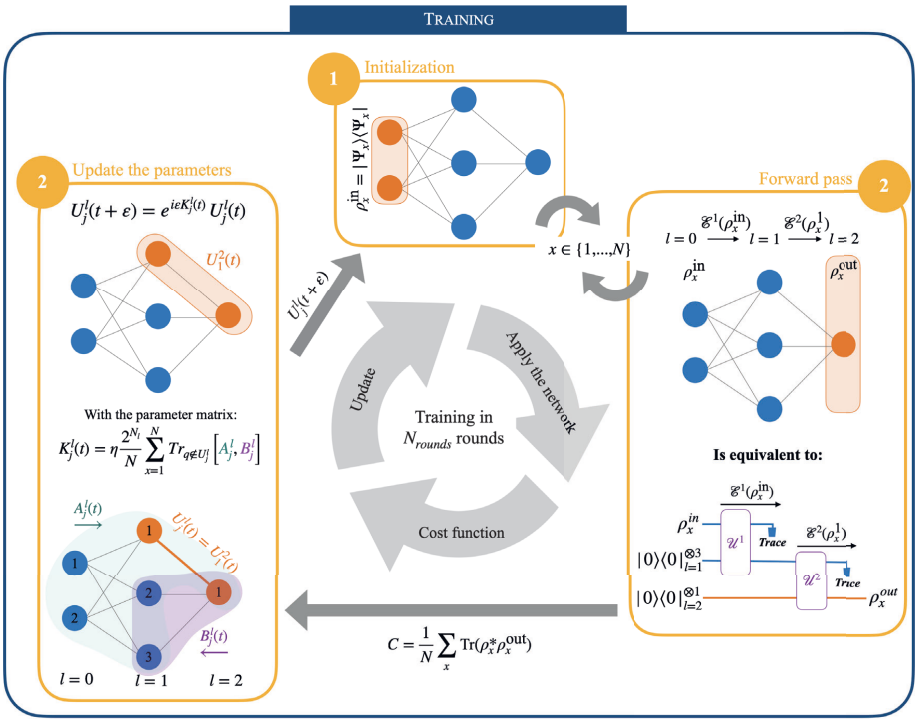


Figure 1.5: Training process for the DQNN. The training takes place in N_{rounds} training rounds. Each iteration has three steps. 1) The network is initialized with a training sample. 2) The state is propagated forwards until the output layer by the successive application of the layers of neurons. Steps 1) and 2) are repeated for each state in the training set. The read-outs of the outputs are inputs to the cost function. In step 3), the perceptrons are updated using the parameter matrix update rule. The modified unitaries are used in the next training round.

update. Conversely, the second term reflects the ideal states transformed with the adjoint channel up to the same point. In a loose interpretation, the commutator quantifies the error generated by the map tuned by the parameters at step t.

Finally, we show that the quantum training algorithm can be improved in the same fashion as in the NADAM rule in equation 1.11. For this purpose, equations **??** are complemented with a Nesterov accelerated momentum and renormalization term for the learning rate. Such a method is employed to train the model we present in the following chapters and is based on the code in [42]. The quantum version of this algorithm defines a momentum $\hat{m}_t$ and variance $\hat{v}_t$ based on the parameter matrix K_j^l. These

three variables are initialized to be zero-matrices in the first training step. First, the momentum term computes the vanishing running average of the past gradients, weighted with hyperparameter β_1, to avoid vanishing gradients:

$$m_{t+1} = \beta_1 \left(\sum_{s<t} \right) m_s + (1-\beta_1) K_j^l(t) \tag{1.22}$$

$$\hat{m}_{t+1} = \frac{m_{t+1}}{1-{\beta_1}^t}. \tag{1.23}$$

The momenta, renormalized or not, are matrices. The weighted average gives a more significant role in the momenta closest to time step t, while the earliest momenta vanish progressively. The renormalization in $\hat{m}_{t+1}$ in equation 1.23 regularizes the bias towards the zero-matrix due to the initialization. Using the notation $U_j^l(t+\varepsilon) = e^{i\varepsilon \Delta W_j^l(t)} U_j^l(t)$, we can formulate a partial momentum parameter matrix:

$$\widetilde{\Delta W_j^l}(t) = \beta_1 \hat{m}_{t+1} + \frac{1-\beta_1}{1-{\beta_1}^t} K_j^l(t). \tag{1.24}$$

To complete the momentum term, a scalar standing for the variance must be incorporated to adapt the learning rate and control fluctuations. It is done by considering the spectral range of the parameter matrix $K_j^l(t)$: it corresponds to the gap between the absolute values of its most prominent and smallest eigenvalues $\Delta_\lambda K_j^l(t)$. This variance is adjusted at each iteration by adding the vanishing running average of the variances of the past steps, weighted with hyperparameter β_2, as for the momentum:

$$v_j^l(t) = \beta_2 \left(\sum_{s<t} v_j^l(s) \right) + (1-\beta_2)(\Delta_\lambda K_j^l(t))^2 \tag{1.25}$$

$$\hat{v}_j^l(t) = \frac{v_j^l(t)}{1-{\beta_2}^t} \tag{1.26}$$

Bringing all components of the update rule together, the quantum NADAM update rule for DQNNs can be written as:

$$U_j^l(t+\varepsilon) = e^{i\varepsilon \Delta W_j^l(t)} U_j^l(t) \tag{1.27}$$

$$\begin{aligned} \Delta W_j^l(t) &= \frac{\widetilde{\Delta W_j^l}(t)}{\sqrt{\hat{v}_j^l(t)} + \delta} \\ &= \frac{1}{\sqrt{\hat{v}_j^l(t)} + \delta} \left(\beta_1 \hat{m}_{t+1} + \frac{1-\beta_1}{1-{\beta_1}^t} K_j^l(t) \right) \end{aligned} \tag{1.28}$$

where δ is a small scalar to regularize possible cases of division by zero. The direct equivalence between the classical and quantum NADAM update rule in equations 1.4 and 1.28 respectively, is straightforward.

1.3 What are the promises of quantum machine learning?

Before applying the QNN model to a concrete task, we discuss some questions underlying their use and design, both from a computer scientist and physicist perspective. In the collective imagination, machine learning is associated with the field of big data. This is the greed of the current deep learning algorithms for vast data sets to learn faster and better. For some computer scientists and industrial purposes, this is a challenge. Applying models with up to millions of parameters to extensive data is equivalent to implementing linear algebraic operations on high dimensional data vectors and parameter matrices. Therefore, even some simple models on paper are hard to implement at large scales. They have a long training time, and entail a massive amount of computational power, as the "quantum" Netflix algorithm for content recommendations, based on the well-known principal component algorithm that takes a whole day to update its parameters [43].

Therefore, quantum computing and the perspectives it opens for machine learning, trigger hopes in the community to bypass this scaling issue, by providing the so-called quantum advantage. Though this concept is not limited, QML is listed among the promising candidates to prove a hierarchy between classical and quantum information processing techniques. This raises the question: what does it mean to be better for algorithms or computing platforms? In the literature, authors often claim that "speed-ups" are expected by using quantum processing for entire algorithms or their subroutines [44]. The formalism of computational complexity is a reference to tackle these questions that constitute the "benchmarking problem" in [45]. For this purpose, robust measures must be developed to adequately compare quantum and classical models on an equal footing, ideally in the most general way, or more realistically, with the best known algorithms on both sides.

Thanks to the possibility in quantum mechanics to represent data in superposition states, it is believed that QML models can deal with many data points simultaneously, instead of looking at them one at a time [27]. This has already been shown, in theory, with the help of the so-called qBLAS (quantum Basic Linear Algorithm Subroutines), such as the Fourier transform mentioned earlier or the HHL (Harrow, Hassidim, Lloyd) algorithm [46] to solve sparse linear systems. The native formulation of quantum mechanics with linear algebra makes it suitable to embed machine learning problems. When provided with enough non-linearity, not only should the computation time be reduced, but more complex patterns hidden in the data could also be discovered.

The latter aspect makes QML particularly attractive to the physics community. Indeed, training quantum models on quantum data, as generated from experiments, can help find more complex patterns underlying the outcomes and discover new features of quantum mechanics. Fundamentally practical applications entail directly simulating complex quantum systems with QML models. For instance, the Variational Quantum Eigensolver (VQE) [34] has become a practice in condensed matter physics and quantum chemistry, and builds upon the VQA approach of QML to analyze and discover new molecules and their ground states. Moreover, the formalism of some QML models

makes them particularly suitable to implement on existing, special-purpose quantum processors such as quantum annealers, that offer a large number of noisy qubits [45]. These experiments give a glimpse into the future abilities of QML.

Nevertheless, even though it has already become desirable for industrial and financial applications [47], the bright future of QML is shadowed by the lack of knowledge about these models. The implementations on imperfect hardware provide only limited insights into their actual abilities. Some unexpected behaviors may occur when scaling up to more significant instances that are out of reach with the current NISQ devices. Furthermore, the theoretical formalism for QML is still in its early steps. For example, the precise role of quantum properties such as coherence and entanglement remains unclear. The analysis of quantum models lies at the core of elaborating powerful and relevant applications. Fundamental knowledge about QML models may result in a proof (or refutation) of quantum advantage in machine learning. For example, even though linear algebra is native to quantum mechanics, the non-linearity required to achieve QNN is non-trivial and must be carefully included in the models. As in classical machine learning, the tuning of hyperparameters often relies on experience and test-and-trial approaches. Another concern is the presence of barren plateaus in the cost landscape. They originate from the vanishing gradients and cause the training to fail. The last years have seen the rise of active research to determine their connections with the choice of ansätze in VQAs and the properties of the cost function. Bounds on the generalization and trainability of quantum models were derived using tools of optimal control theory: locality and expressivity were pointed out as mechanisms reducing barren plateaus [40, 41, 38, 37]. The birth of this field of research represents an opportunity for discoveries in quantum information theory and the development of new tools to describe quantum phenomena.

Some computer scientists argue that the scaling problem in classical methods can be solved thanks to cloud computing and supercomputing. Instead, they pinpoint adversary attacks as the weak point of machine learning models [48, 49, 50], to the extent that an undetectable deviation from the original data distribution leads to drastically different outcomes. For example, it can cause two almost indistinguishable points to be classified under various labels in a classification task. The underlying problem lies in selecting and filtering the valuable features to identify patterns within the data. The model should discard those that are too sensitive to noisy distributions. For this purpose, the robustness of quantum classification models has been analyzed using tools of quantum hypothesis testing in [51], proving that an upper bound can be formulated in QML classification algorithms regarding the achievability of robustness against adversary attacks. This result would hint that quantum models suffer from the same shortcoming as classical models.

Until now, the absence of a quantum device with the correct scale and connectivity represents the bottleneck to the blossom of QML and the proof of its superiority over traditional methods. Moreover, fully quantum models such as the DQNN introduced in 1.2.2 often require storing and accessing quantum information in memory, as for the efficient implementation of the backpropagation algorithm, for example. For this

reason, the availability of a stable QRAM (Quantum Random Access Memory) is a pre-requirement to reach the complexity bounds derived from the theory.

Furthermore, comparing classical and quantum machine learning reveals obstacles inherent to the latter's quantum nature. The input and output data must be expressed explicitly and in the same language for a fair comparison. [45] discusses this question and points out the so-called "input" and "output" problems. The former refers to the necessity to efficiently embed classical data in a quantum object to apply QML methods. According to the data, techniques exist that encode data in basis states and amplitudes or tune angles in quantum circuits. Nevertheless, these methods scale poorly with the size of the data sets, thereby canceling the speed-up expected from the QML algorithm. Furthermore, it was found in [52] where a Fourier-inspired encoding is designed, that despite the universality of the model chosen, the encoding can limit the expressivity of the model to a small subset of functions. As its name suggests, the output problem is related to the read-outs of the outcomes of the algorithm. Indeed, [53] warns about the accessibility of the information contained in the outputs of QML algorithms. For example, the collapse of the quantum states once measured erases part of its information. Suppose the solution to a task is encoded in the amplitudes of a superposition state. In that case, the algorithm must either be repeated many times, or the outputs must be processed further, quantum-mechanically or classically, to reconstruct the information. For example, if state tomography is required, the number of repetitions of the experiment and measurements scales exponentially with the size of the data. Once more, read-out cancels out the complexity reduction announced based on the QML algorithm only.

Chapter 2

Automate denoising: the Quantum Autoencoder

2.1 Why is quantum denoising needed?

2.1.1 Noise on quantum computers

The concept of denoising originates in classical computing, and is a task many machine learning models are designed for: it aims at cleaning some data from diverse sources of noise. It is particularly developed in the realm of speech [54, 55, 56] and image [57] recognition. However, it is also increasingly used outside computer science, to filter out the noise in experimental data such as the data collected by the LIGO [58]. Indeed, these data sets cannot escape the presence of a background that hides the properties probed with the experiment. As a result, the goal of denoising is to remove the unwanted features of a data set, without harming the important ones. This implies that the machine learning algorithm can differentiate between the relevant and irrelevant parts of the data. As a by-product, it may identify new parasitic elements that were not included in hand-made denoising algorithms.

Moving to quantum computing, denoising techniques play a crucial role in better available NISQ devices. The average error rate for two-qubit gates in state-of-the-art is of order 10^{-1} for quantum processors. In contrast, classical computers reach lower rates by 16 orders of magnitude, about 10^{-17} [59]. In the latter, despite some slight noise affecting the information processing devices, computations are possible, and errors are too rare to be significant. The errors per qubit accumulate when scaling up the number of qubits in a quantum processor. Eventually, the outputs become too faulty. In this respect, the high quantum error rates represent a critical obstacle to realizing programmable quantum computers. The quantum error correction (QEC) codes known to this date require a significant overhead of quantum and classical resources. To protect quantum information, it is encoded in logical qubits. The latter is composed of multiple physical qubits and is a pre-requisite for many QEC codes [60, 61, 62, 63, 64]. Moreover, identifying the error demands building look-up tables whose size scales ex-

ponentially with the number of qubits involved. Operational challenges such as fast classical-quantum communication are currently central in implementing such codes. It enables real-time post-processing of the measurement outcomes to adapt the recovery operations in error correction codes [65].

Noise in quantum processors can be manifested in two ways. First, as Shor explains in [8], the quantumness of states is both a blessing and a curse. While superposition and entanglement can lead to the superiority of ideal quantum devices over Turing computation models, they are both extraordinarily delicate and short-lived. Decoherence destroys information stored in superpositions. This can be modeled by letting the qubits interact with other parasitic quantum systems in the environment, which perform measurements on the information units. In addition, quantum computing is connected to analog computation models: the gates in a quantum circuit depend on continuous parameters such as rotation angles. Consequently, their perfect implementation is almost impossible. This situation is aggravated by the accumulation of such imprecisions that considerably shortens the accessible circuits and computation times, and decreases the fidelity of the output states. Quantum errors can also propagate and affect other qubits through gates. For example, when noise impinges on the control qubit of a Controlled-NOT gate, it also makes the target qubit's state noisy.

Second, classical methods employed to handle noise generally rely on the duplication of information. If the same features are encoded on multiple bits and an error deteriorates one of them, the ideal features can be recovered by a simple majority rule on the copies. However, the no-cloning theorem [66] forbids such duplication in the quantum framework, and a substitute encoding must be found to "replicate" quantum information [60, 61, 62, 63, 64]. Furthermore, qubits have more degrees of freedom than classical bits since both amplitudes and phases define the superpositions. This apparent flexibility to store information can jeopardize the states, for noise affects them more diversely. It acts on the two types of degrees of freedom cumulatively, triggering bit-flips and phase noise simultaneously. Unfortunately, building a quantum error correction code that covers multiple error schemes at the same time requires an overhead in quantum resources, lower bounded by the quantum Hamming bound [67].

To summarize, noise sources affecting quantum hardware can stem from the imperfection of the control of the qubit, such as the physical processes underlying the realization of a gate. Diverse spurious interactions with an environment can lead to faulty states, including but not restricted to qubit-qubit interactions, connections with a thermal bath causing thermal excitations, and imperfect modeling of qubits by ignoring excited states out of the computational space. Even though error correction codes are already available in theory, their implementation is intended for more evolved quantum hardware that encompasses more qubits and has smaller gate and measurement error rates. Meanwhile, error mitigation and noise reduction techniques are constantly proposed to render the hardware fault-tolerant.

2.1.2 Design protocols for noise

The conception of quantum denoising protocols is generally tricky because it relies on two expensive steps. First, measures are employed to quantify as many noise sources on the hardware as possible. We refer to this process as the characterization task. Second, the measurement outcomes from the characterization are re-employed to find a protocol, classical, quantum, or hybrid, able to dampen the effects of the experimental noise to a significant extent. Both sides of the protocol are related to Characterization, Verification, and Validation (CVV). A summary diagram is shown in figure 2.1. We now take a look at both sub-tasks separately.

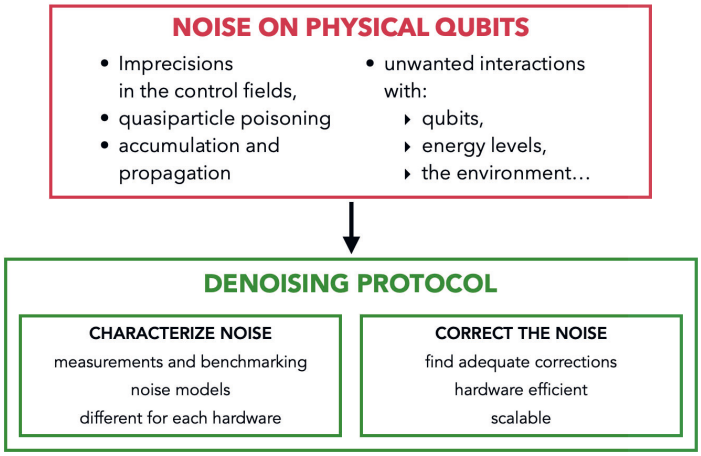


Figure 2.1: The design of denoising protocols is a challenging task on the current NISQ devices. A non-exhaustive list of possible noise sources is shown in red and includes noise due to the hardware control and to the physical system itself with the interactions it permits. Because these noise sources are numerous and some of them are unknown, noise correction and mitigation are difficult. Such techniques can be divided into two sub-tasks: the device must be characterized to identify noise sources. Based on quantitative and qualitative measures, algorithms and protocols can be found to dampen the observed noise.

2.1.2.1 Noise characterization

In the characterization step, the challenge is to build a complete measure that will provide significant insights about and differentiate between various sources of noise, whether they are systematic or not, coherent or incoherent. In addition, a desirable feature of

these measures is the generalization to all types of hardware. It can compare different devices and conclude their technological evolution. In the NISQ era, the nature of noise varies from one device to another, and some platforms are more suitable than others for implementing certain tasks. Therefore, instead of seeking a general evaluation of the hardware for universal programming, task-oriented benchmarking brings to light the advantages of each device for a particular set of tasks. A bad score with the general approach can hide uneven weights among noise sources that cause only a subset of gates to be faulty. The device performs well for applications that use only the robust set of gates. Hence, task-oriented benchmarking can open opportunities to experiment with quantum algorithms despite the presence of noise on some gates [68, 69].

More general techniques have been developed that are transferable to any task. They support the evolution of quantum computers in the direction of a universally programmable device rather than a task-specific platform. The general intuition behind such measures is to compare the results of a particular circuit run on quantum hardware with the ideally expected ones. A corollary is that such perfect outputs must be calculated outside the quantum device. Hence, the circuit must be simulatable classically. This consequence restricts the types of gates and the number of qubits that can be leveraged and evaluated. In contrast, even though process tomography does not necessarily require the classical outcome of some circuit, it scales exponentially in the number of qubits, and it remains sensitive to State Preparation and Measurement Errors, also known as SPAM.

Therefore, Clifford gates circuits play a significant role in implementing general benchmarking scenarios. According to Gottesman-Knill theorem [70], Clifford gates can be simulated using only classical resources. Hence, such circuits fulfill the primary constraint explained above.

Randomized benchmarking [71, 72] employs this observation to derive a parameter α that estimates some error probability resulting in the Error Per Clifford gate rate (EPC), and makes it possible to evaluate the gates classically. A quantum circuit is designed out of $m+1$ randomized Clifford gates $\{C_{i_j}\}_{j=1}^{m}$, such that $\prod_{j=1}^{m+1} C_{i_j} = \mathbb{I}$, where the index i designates a gate in the Clifford group. The noiseless channel evolves the initial state in m steps and returns it to its original state in the $m+1$-th step. The noise is modeled by applying a noise superoperator Λ_{i_j}, which can be different for each gate i and step j. The noisy circuit is built out of the gates $\{\Lambda_{i_j}(C_{i_j})\}_{j=1}^{m}$. Any noise affecting a gate can be represented universally using the depolarizing channel [71] described by a parameter α as:

$$\rho^{out} = (1-\alpha)\frac{\mathcal{I}}{2^n} + \alpha\rho^{in} \tag{2.1}$$

with the same value of α for all m gates. $\mathcal{I}/2^n$ stands for the maximally mixed state of n qubits. After running the random circuits made by combining different gates, a POVM measurement $E_{|\Psi^*\rangle}$ is performed, ideally corresponding to the state obtained without noise $E_{|\Psi^*\rangle} = |\Psi^*\rangle\langle\Psi^*|$. This helps to define the process fidelity F:

$$F(m, |\Psi^*\rangle) = \mathrm{Tr}\left\{E_{|\Psi^*\rangle}\mathcal{S}_m(\rho^{in})\right\} = \langle\Psi|\mathcal{S}_m(\rho^{in})|\Psi^*\rangle \tag{2.2}$$

where the quantum map $\mathcal{S}_m$ corresponds to the random Clifford channel averaged over the many realizations, and the right hand-side of Eq 2.2 is the fidelity with the initial state. By repeating these steps for different circuit depths m, the fidelity can be extrapolated as a function of m and α:

$$F(m, |\Psi^*\rangle) = A_0 \cdot \alpha^m + B_0. \tag{2.3}$$

Thanks to the determination of α, the Error Per Clifford (EPC) is calculated for n qubits:

$$EPC = \frac{2^n - 1}{2^n}(1 - \alpha). \tag{2.4}$$

In comparison, quantum state tomography repeatedly projects states on a basis and averages the outcomes over many trials. Therefore, it is sensitive to SPAM errors. RB is robust to these errors thanks to the extra parameters A_0 and B_0 that capture them. Because it is tractable and tolerates measurement and preparation errors, RB is an adaptable industrial measure to characterize devices. However, it remains too limited to understand larger systems. Firstly, its implementation is limited to 6-qubit circuits. Secondly, it can only provide partial information about multi-qubit gates. For example, it cannot reflect the noise accumulated in the idling time when some qubits wait for the next round of gates while others are processed. Coherent and incoherent sources of noise are not distinguishable. The coherent noise refers to the unitary noise sources stemming, for example, from miscalibrations of control pulses on a superconducting chip. At the same time, the latter describes stochastic noise events that lead to decoherence [73]. Even though methods involving the classical simulation of circuits are inefficient on large scales, they still catch the community's interest in designing unified measures for the current devices.

The Quantum Volume [74] uses randomly generated circuits as well. However, the building blocks are alternating layers of two-qubit gates. This measure is particularly attractive to the community because it bears a useful operational meaning with direct consequences on the hardware and its evolution. It gives predictions on the algorithms it can run. Namely, it describes the size of the largest square circuit that a device can run fault-tolerantly: the size corresponds interchangeably to the number of qubits and the depth of the circuit.

Recently, alternative benchmarking methods have been proposed [75] using hardware-efficient gates. The outputs are binned according to their probability after performing classical and quantum simulations of multiple circuits. The Binned Output Generation (BOG) method can distinguish between coherent and incoherent types of errors. It can also evaluate error rates on multi-qubit gates without having to break them down into 2-qubit gates [75]. This protocol has revealed the importance of new behaviors of errors stemming from considering multiple qubits together. Therefore, it provides more detailed information about the underlying noise mechanisms affecting the hardware during a real quantum algorithm to improve future devices.

2.1.2.2 Noise correction

After estimating the noise impinging on a quantum device, protocols must be designed to bring the outputs closer to the noiseless expected states. As mentioned in the opening of this section, quantum error correction codes have been designed and can face both bit- and phase-flip errors. However, they require many physical qubits to encode the logical ones. They must operate with quick and clean gates to avoid decoherence. They should still allow classical-quantum communication to process measurements and adapt their gates accordingly. QEC codes are, therefore, often destined for the next era of quantum computing with more qubits and better gates fidelities. Meanwhile, noise reduction and error mitigation on the current NISQ devices are promising to prove a quantum advantage on these early noisy machines. But they come along with their difficulties.

Error mitigation techniques can apprehend the problem from two angles. It can attempt to modify a general circuit to minimize the recurrence of noisy processes in the computation. For example, suitable compiling associated with machine learning techniques can optimize the gates of a circuit [76, 77]. Such circuit optimization can be realized directly on a quantum device using VQA to reduce the depth of the circuit [33]. Indeed, the least gates in a circuit, the slightest noise is injected due to the noise in gates themselves and unwanted interactions during the idling times.

From another point of view, noise cancelation can directly target the outputs of circuits. The noiseless expectations values can be deduced via extrapolation based on some data. Two popular methods are currently the zero-noise extrapolation (ZNE) [78, 79] and the Clifford data regression (CDR)[80]. In the former, the data consists of the same circuit with variable noise intensities controlled by some parameter ε. This can be achieved by increasing the gate times, thereby extending exposure to unwanted interactions for decoherence. The so-called fixed identity insertion method varies noise by extending circuits with identity operations. They are formulated as two CNOT gates applied one after another. While this extension leaves the states unchanged in the ideal circuit, the CNOT gates corrupt them. The noise parameter is then controlled by adding more or less of these identity blocks in the circuit. Using a Richardson extrapolation, an estimate for the expectation value $\hat{\mu}$ of the observable of interest is deduced at zero noise intensity. More concretely, to obtain the estimation of the expectation value μ in the noise-free case, different noise levels $\{c_0, \cdots, c_n\}$ such that $c_j < c_{j+1}\ \forall j$ define the noise amplification εc_j implemented in the noisy device. For each run with noise level c_j, an estimate $\hat{\mu}_j$ is calculated. The extrapolation yields the approximate expectation value in the noise-free scenario:

$$\hat{\mu} = \sum_{j=0}^{n} \gamma_j \hat{\mu}_j \tag{2.5}$$

where the coefficients γ_j are constrained as follows: $\sum_{j=0}^{n} \gamma_j = 1$ and $\sum_{j=0}^{n} \gamma_j c_j^k = 0,\ \forall k \in \{1, \cdots, n\}$. The downsides of this technique are that it entails running the same circuit repeatedly to gather enough statistics on the outputs in order to build the expectation values at each noise level. In addition, the form of the final result is usually highly non-trivial and difficult to model with low-degree polynomials. Despite a

relatively good scaling, the accuracy of the estimations drops as the accessible levels of noise rise far away from zero intensity.

Instead of varying the noise intensity in the same circuit, CDR generates Clifford circuits that can be simulated classically, and that resemble the initial non-Clifford one. A linear regression is then performed with a simple least-squares optimization on two parameters a_1, a_2, such that the noise mitigated expectation value $\hat{\mu}$ takes the form:

$$\hat{\mu} = f(\tilde{\mu}) = a_1\tilde{\mu} + a_2 \tag{2.6}$$

where $\tilde{\mu}$ is the noisy estimate. In practice, each of the m Clifford is simulated twice, classically and on the device, with ideal outputs $\{y_i\}$ and the noisy outputs $\{x_i\}$. The parameters for the estimation of found by minimizing the mean-squared error:

$$(a_1, a_2) = \arg\min_{a_1,a_2} \sum_{i=1}^{m} [y_i - (a_1 x_i + a_2)]^2. \tag{2.7}$$

As in the case of randomized benchmarking, the collection of data necessary to obtain precise estimates becomes an obstacle to scaling this method to larger circuits. ZNE and CDR have been combined in the so-called variable noise CDR (vnCDR) technique [81] to extract the most advantageous features of the data used in each method separately.

This brief and incomplete overview of noise mitigation methods shows how machine learning can improve the next generation of NISQ devices. All techniques mentioned above rely on data to learn a better version of the circuits or their outputs to reach noise-free estimates of outputs. Their implementation exclusively on quantum hardware would eliminate potentially expensive post-processing and reduce decoherence during the idling times it can cause. Therefore, we attempt to use the QML models introduced in section 1.2.2 to realize such a noise cancellation protocol.

2.2 Apply machine learning to the concrete task of denoising quantum states

One may argue that a computational model such as machine learning loses meaning if it cannot contribute to finding the solution to any problem. Therefore, the main focus of this work is the design of a QNN to facilitate the denoising of certain states on a quantum device. As the previous section argues, this task is mainly demanded to scale up and eliminate noise impinging on current NISQ devices. It may pave the road to the next era of fault-tolerant quantum computing. We introduce a model that goes a step closer to efficient noise cancellation: the Quantum Autoencoder.

2.2.1 The autoencoder: FFNN for denoising

QML algorithms often draw inspiration from classical models and make them compatible with quantum platforms. The invention of quantum perceptrons, described in 1.2.2

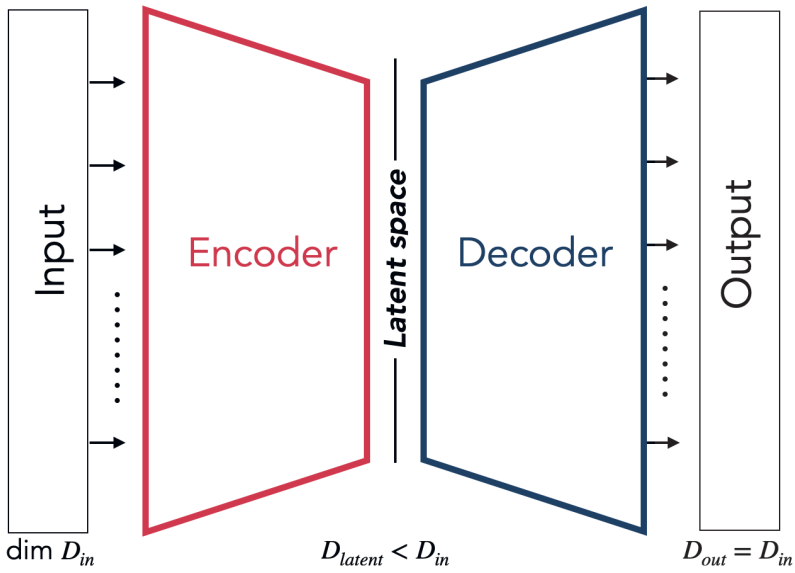


Figure 2.2: The autoencoder is a feed-forward neural network, divided into two subsystems trained together. On the left, the encoder reads the data to extract patterns that define it. On the left, the decoder can reconstruct the inputs in the output layer from the information previously encoded. The intermediate representation of the data is often called the latent vector and lives in the latent or code space. To guarantee that the training results in non-trivial mappings, the undercomplete architecture is depicted here: the dimension of the latent space is smaller than that of the actual inputs

supports this approach. Let us discuss the example of an FFNN which can be used in an autoencoder to denoise quantum states.

The autoencoder is a well-studied model and has witnessed the emergence of deep learning from its early steps onwards [82, 83, 84]. Figure 2.2 describes the autoencoder as a feed-forward neural network that can be separated into two primary constituents: the encoder and the decoder. Their role becomes clear when we look closer at the problem it solves: the network is trained to reproduce its inputs but must avoid learning an identity map. In other words, the network must learn the essential components of the training data to be able to reproduce it. For this purpose, the encoder has the mission to find a suitable way to represent the inputs. It maps them to a different space, while the decoder reads and translates them back to their original form. Several methods can drive the network away from the trivial map, while processing the inputs until the out-

put layer, going through an intermediate representation [17]: the cost function can be regularized to penalize such a scenario, perturbing the training data by a small amount forces the network to extract the characteristic information for the unspoiled distribution and dimension variations in the architecture can prevent leaving the inputs untouched in the forward pass. Our study focuses on the latter technique, particularly when the input space is larger than the intermediate representation space. This bottleneck layout is referred to as the undercomplete autoencoder. In classical applications [85, 86], many shallow autoencoders are often stacked and trained together. This accelerates the training by pre-training the weights on smaller parts of the network, thereby initializing them closer to their optimized value for the autoencoder itself.

The task the autoencoder is trained for is usually less desired than its by-product, the feature extraction. Already in their early moments, autoencoders were shown to be equivalent to Principal Component Algorithms [83, 87]. These techniques earned great attention in the realm of data compression since they grasp the main axes of variations within the data sets: the features extracted in the D_{latent}-dimensional latent space are similar to the D_{latent} first main singular vectors resulting from the PCA when the decoder is set to be linear. The loss function is the mean square error. The introduction of non-linearity in the encoder and decoder produces a generalization of the PCA. Still, it must be constrained, so the network does not learn a simple memory and directly maps data points from indices without processing characteristic features.

The reproduction task is mediated by encoding an internal representation of the inputs in the so-called latent or code space. It empowers the autoencoder in various applications, such as data compression or information retrieval. For example, autoencoders contribute to the implementation of generative models [17]. Instead of finding an underlying distribution to understand the data, these models aim at generating a new distribution that reproduces it. The characteristics of this distribution shine light on the complex features hidden in the data. A popular generative model is the Generative Adversarial Network (GAN) [11]. This technique has earned great attention in the last years, and research in this field has been flourishing with some applications to physics [12]. It is expected to identify complex natural phenomena that are not yet accessible to the human mind, such as entanglement characteristics. The latent variable produced by the autoencoder is an unknown intermediate vector of features and serves as a basis to define probability distributions [17]. This example shows the importance of the latent vector in the autoencoder. More generally, the encoding found by the autoencoder can be leveraged to boost other techniques, such as classification or compression of data.

Thanks to the intermediate representation designed by the encoder, an algorithm trained to classify data can do so while relying on sharper features. Reference [88] showed that autoencoders tend to reorganize inputs in space: points that are semantically related to each other are mapped in the same neighborhood with similar vectors. Consequently, the allocation of latent vectors to different parts of code space already reflects their similarity and labels. The optimized encoding facilitates the classification task. In undercomplete autoencoders, this efficient compressed representation can be leveraged to improve memories. For this purpose, deeper networks, despite potentially harder

training, can yield better data compression [89].

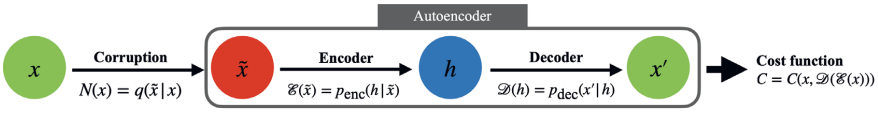


Figure 2.3: The denoising autoencoder is a variation of the standard autoencoder. Before the training samples are read, a stochastic noise process represented by a function $N(x)$ of its inputs, or equivalently as a conditional probability distribution $q(\tilde{x}|x)$. After the corruption step, the autoencoder is trained by applying the encoder and decoder functions $\mathcal{E}$ and $\mathcal{D}$ respectively, yielding successively the intermediate representation h and the reconstruction x'. The training deviates from the traditional approach by using both ideal and intact data sample to calculate the cost function $C(x, \mathcal{D}(\mathcal{E}(x)))$.

However, to reduce noise on quantum devices, the so-called "denoising autoencoder" [90, 86] is the most promising. As shown in figure 2.3, instead of learning to reproduce exactly its input data x, a step is added before the input layer to let the training samples undergo a stochastic corruption process, $\tilde{x} = N(x)$, such as complete erasure of some vector components or Gaussian noise. From this imperfect training data set, it is expected that the network will capture robust characteristics in the inputs. They remain invariant under the application of noise to recover the noiseless states. The loss function is expressed with respect to the noiseless inputs and the outputs recovered from the noisy samples. The loss function, in turn, determines the gradients in the backpropagation algorithm. This way, it educates the network about the elimination of noise. These denoising autoencoders are a common practice in classical machine learning. Early results showed that they can denoise MNIST data set successfully when it is polluted by an erasure noise process [90]. In this implementation, the features retained in the latent representation have a better quality than those obtained otherwise. This way, the autoencoders improve the performance of subsequent networks in classification tasks. Similarly, the same architecture is applied to sound recognition, where [85] filters noise out of recorded phone conversations in Cantonese.

2.2.2 Build a Quantum Autoencoder (QAE)

2.2.2.1 The QAE architecture

The Dissipative Quantum Neural Networks in section 1.2 become the basis for the quantum denoising model we propose. Indeed, the quantum autoencoder (QAE) has been one of the first network architectures to be implemented using QNN [91]. However, in this setting, the QAE was used for quantum data compression and to re-discover the

quantum teleportation protocol between two agents. The QAE is therefore not only able to tackle classical tasks, but it can solve purely quantum problems. Feldmann and Bondarenko first achieved the realization of denoising QAE with DQNNs in [42].

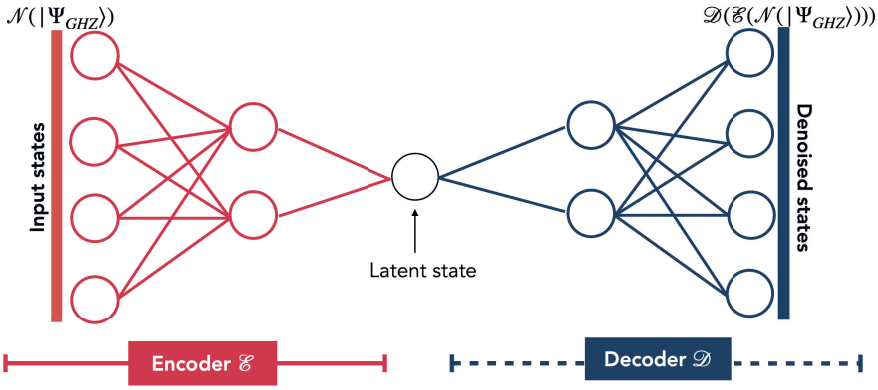


Figure 2.4: The quantum autoencoder is embedded in the dissipative QNN model. The qubits represent the neurons and are organized in layers. We allow for full connectivity between two successive layers, i.e. each layer of unitaries encompasses N_{l+1} $(N_l + 1)$-dimensional unitary couplings. To guarantee a non-trivial quantum map, the denoising QAE is undercomplete, and the dimension of latent space is smaller than the inputs' Hilbert space: 4-qubit states are compressed in a single qubit.

The denoising autoencoder is organized in layers of qubits whose width decreases until the latent space and increases until the original dimension is recovered. This symmetric layout simulates undercomplete autoencoders and guarantees that the network does not learn a trivial map. The encoder, shown in red, implements a quantum map $\mathcal{E}(\bullet)$ that compresses the N_{in}-dimensional inputs onto the N_{latent}-dimensional latent space. The latent state is then converted back to its original format by the decoder in blue with the map $\mathcal{D}(\bullet)$. The QAE implements a reversible compression of the input data. Compared to VQA formulations of the QAE where the decoder is often set to be the adjoint of the encoder $\mathcal{D}(\bullet) = \mathcal{E}^\dagger(\bullet)$ [92, 93, 94, 95], the encoder and decoder are trained simultaneously. During the learning phase, the distinction between the two channels provides more expressivity and resilience to the operations that can be reached through optimization. In addition, maximal connectivity between successive layers is favored. The unitary channel between two successive layers l and $l+1$ contains N_{l+1} unitaries acting on $N_l + 1$ qubits. In simulations of the network, the maximum number of qubits does not exceed $\max_l N_l + N_{l+1}$, for $l \in [1, L-1]$, as long as reinitialization of the qubits

is available on the device.

2.2.2.2 Learn denoising with the QAE

The application of the QAE takes place in 6 steps:

1. The input layer is initialized with the data. Let us consider such a state preparation for both a quantum implementation and the classical simulation:

   a) When the network is running on a quantum computer, the inputs ρ_{in} are prepared on a quantum device to its best accuracy. As such, it already includes the effect of noise channels and the ideal state $\rho^* = |\Psi^*\rangle\langle\Psi^*|$ cannot be realized. We note $\rho^{\text{in}} = \tilde{\rho}^{\text{in}}$, where $\tilde{\rho}^{\text{in}}$ is the altered state.

   b) When the network is simulated numerically on a classical computer, the ideal state ρ^* is first prepared. It is then distorted by applying a noise channel, whose intensity is tuned by a noise parameter p:

$$\rho^{\text{in}} = \mathcal{N}(\rho^*, p) \tag{2.8}$$

2. Subsequently, the quantum map corresponding to the encoder propagates inputs forward until the latent space at layer k is reached:

$$\begin{aligned} \rho^{\text{latent}} &= \mathcal{E}(\rho^{\text{in}}) \\ &= \text{Tr}_{l=1,\cdots,k-1}\left\{\prod_{l=k-1}^{1}\mathcal{U}^l\left(\rho^{\text{in}}\otimes|0\rangle\langle 0|^{\otimes\sum_{l=2}^{k}N_l}\right)\prod_{l=1}^{k-1}\mathcal{U}^{l\dagger}\right\} \end{aligned} \tag{2.9}$$

   In the first learning step, a random quantum map is applied.

3. From the information contained in the latent state only, the decoder reconstructs an output that comes closer to the perfect state ρ^* as the training progresses. The output state is then:

$$\begin{aligned} \rho_{out} &= \mathcal{D}(\mathcal{E}(\rho\text{in})) \\ &= \begin{cases} \mathcal{D}(\mathcal{E}(\tilde{\rho}^{\text{in}})) & \text{when implemented on a quantum computer} \\ \mathcal{D}(\mathcal{E}(\mathcal{N}(\rho^*, p))) & \text{when simulated numerically on a classical computer.} \end{cases} \end{aligned} \tag{2.10}$$

4. Repeat steps (1-3) for each state ρ_x^{in} in the training set and collect the outputs for the update of the parameters.

5. Implement the parameter update following the training rule in 1.2.2.2.

6. Repeat steps (1-4) until a convergence criterion is reached. It can be a fixed number N_{rounds} of training rounds. Alternatively, the algorithm can stop when the learning curve is stable enough, and the update does not decrease the cost function anymore.

2.2.2.3 What is the training data?

In contrast to the QAE for data compression, the denoising QAE does not have access to the ideal state in equation 2.10 for each training state. Recalling the initial classification of machine learning techniques shifts the QAE from the supervised type to the unsupervised one. Even though we are still training the QAE with pairs of states, one for the inputs and one to represent the desired states, target states become noisy too and indicate partial directions for completing the task. This difference will become important in section 3.3. Nevertheless, the training rule 1.19 is kept identical. The only difference lies in the initialization of the output layer as an imperfect target state in the definition of $M_j^l(t)$. Therefore, we implement an approximate version of the supervised learning algorithm in equation 1.19 for an unsupervised task.

In addition, this approximation gives rise to subtleties when monitoring the network's learning process. For this reason, authors in [42] establish a distinction between two different functions. The cost function is defined as the fidelity of the outputs with the noisy target states $\widetilde{\sigma_x^{\text{out}}}$:

$$F_{\text{cost}} = \frac{1}{N}\sum_{x=1}^{N} \text{Tr}\left\{\rho_x^{\text{out}}\, \widetilde{\sigma_x^{\text{out}}}\right\}. \tag{2.11}$$

In contrast, to evaluate the performance of the QAE, a validation function is proposed that compares the outputs with the ideal noiseless target state σ_x^*:

$$F_{\text{val}} = \frac{1}{N}\sum_{x=1}^{N} \text{Tr}\left\{\rho_x^{\text{out}}\, \widetilde{\sigma_x^*}\right\}. \tag{2.12}$$

The former reflects the learning progress during the training phase where ideal states are unavailable. The latter is computed in the testing phase and evaluates the generalization of the map found by the network to different data sets. For numerical simulations, both of them can be used interchangeably during the training: they possess the same convergence behavior, but the saturation value will differ. When using the cost function for finite noise parameters p, the best achievable fidelity is lower than 1 and decreases with p. With the validation function, a successful training must tend to 1 at the end of the training. For the denoising QAE, the update rule in equation 1.19 is derived from the cost function F_{cost} with noisy target states. Thanks to its explicit formulation in terms of layers of quantum channels, we need not compute the cost function to obtain the parameter matrices. Therefore, the training progress can be followed by both cost and validation functions interchangeably, without interfering with the updates.

After describing the core of the algorithm itself, we clarify the content of the data set. For a quantum version of the denoising autoencoder, the training data is composed of pure quantum states selected to solve the task. To reproduce and understand the results of [42], the training set is made of N_{data} Greenberger-Horne-Zeilinger (GHZ) states [96] on N_{in} qubits:

$$|\Psi_{\text{GHZ}}\rangle = \frac{|00\cdots0\rangle + |11\cdots1\rangle}{\sqrt{2}}. \tag{2.13}$$

The preparation of these states on a quantum device is of particular interest from various points of view. They constitute a generalization of Bell states and represent a quantum resource for strong entanglement. Moreover, they maximize the quantum bound of the Mermin polynomials[1] [97, 98]. They are the perfect states to evaluate the possibility of creating entanglement and quantify it on the quantum hardware available. It has been done on 5-qubit, and 53-qubit superconducting chips [99, 100]. Their high entanglement also makes them perfect resources for quantum algorithms such as quantum teleportation and communication [101], quantum computing [102], quantum metrology [103], quantum information [104]...

For the following numerical simulations, noise is modeled by a noise channel $\mathcal{N}(\bullet, p)$. In particular, we train the network to correct the bit-flip channel, whose intensity is tuned by the probability p for each independent qubit to be flipped. The GHZ-states $\rho_{\text{GHZ}} = |\Psi_{\text{GHZ}}\rangle\langle\Psi_{\text{GHZ}}|$ undergoing the effects of the bit-flip channel are then defined as:

$$\begin{aligned} \rho^{\text{in}} &= \mathcal{N}_{N_{\text{in}}}(\cdots \mathcal{N}_1(\rho_{\text{GHZ}}, p)) \\ \text{where } \mathcal{N}_i(\rho, p) &= (1-p)\,\mathbb{I}_i\rho\mathbb{I}_i + p\,X_i\rho X_i \end{aligned} \tag{2.14}$$

According to this definition of the average effect of the bit-flip channel on the inputs, the density matrices for the noise strengths p and $1-p$ are identical, due to the symmetry of the GHZ states. Therefore, when controlling the generalization of denoising, it is sufficient to test for noise parameters $p \leq 0.5$.

2.2.2.4 Numerical simulation of the QAE

To understand the interest of the denoising QAE, the network has been simulated numerically. The code was written in two languages. We used Python as in [105]. With the implementation of real hardware in mind, the code has been developed with the pyquil library offered by Rigetti. In this language, users can create gates and circuits that can be directly sent to their devices to test quantum algorithms. Similar to the original approach [42], the Qetlab package on Matlab considerably accelerates the classical simulations by optimizing quantum matrix operations. This makes simulations of larger networks more efficient in investigating the behavior of larger input states. Nevertheless, it cannot be compiled into the native language of quantum devices. Due to the length of the training for deep networks, most simulations were run on the Jülich cluster.

To enable the reader to reproduce our results, we provide a short summary of the parameters in the program. Unless mentioned otherwise, the network was trained on 200 training pairs of noisy GHZ-states. To ensure convergence of the cost function, 200 rounds of parameter updates were simulated. For the gradient descent, the NADAM update rule in 1.28 was chosen to accelerate and stabilize the training. The momentum

[1] Mermin polynomials generalize Bell's inequalities to any arbitrary amount of qubits in a deterministic way. The gap between the values of the polynomials under quantum and local reality hypotheses is maximized by the GHZ-states. Consequently, Mermin inequalities refute the existence of local hidden variables to instrument communication between states. They are evidence of the non-locality of quantum states.

hyperparameter β_1 was set to 0.8, and the variance hyperparameter β_2 to 0.999. The inverse learning rate $\lambda = 1/\eta$ was 0.0175 and had to be adapted according to the number of states in the training set.

2.3 What does a denoising QAE learn?

The network can be simulated numerically by completing a model for denoising QAE trained with the NADAM rule to denoise GHZ-states. First, we attempt to reproduce the results provided in the main reference [42].

2.3.1 Evolution of the training

In this first simulation, we aim to train a $[4, 2, 1, 2, 4]$ network. We describe the structure of the network layer by layer with the notation $[N_1, \cdots, N_L]$ where N_l designates the number of qubits in layer l, or equivalently its width. This network is challenged to reproduce 4-qubit GHZ-states on its output layer. Its training inputs were affected by the bit-flip channel with noise intensities $p \in [0, 0.5]$ described in equation 2.14.

To understand the evolution of the training and its learning progress, we draw the objective function throughout the learning phase. For this purpose, we choose to look at the validation function defined in 2.12 for a more intuitive interpretation. We remember from the introduction to machine learning that the cost function is minimized. More rigorously, the task in the denoising QAE is to minimize the infidelity of the output state with its desired state, $C(\rho^{\text{out}}, \rho^*) = 1 - F_{\text{val}}(\rho^{\text{out}}, \rho^*)$. However, for a more intuitive but strictly equivalent formulation, the objective function can employ the fidelity $C(\rho^{\text{out}}, \rho^*) = F_{\text{val}}(\rho^{\text{out}}, \rho^*)$ directly and maximize it instead. Since $F_{\text{val}} \in [0, 1]$, a fidelity close to 1 is the indicator of successful training and the output state is expected to recover the ideal state accurately. In contrast, the lower the fidelity, the further apart the target and the output states are. The fidelity can therefore serve as a distance metric for QNNs. This quantity is still needed and debated to compare quantum and classical neural networks.

The learning curves are presented in figure 2.5 over the range of bit-flip probabilities per qubit $p \in [0, 0.5]$. They agree with those found in [42]. The graph shows a typical behavior across probabilities. After an initial phase with strong gradients, the behavior of fidelity changes and it plateaus at some fixed value. All learning curves for noise intensities below $p = 0.35$ are superposed and follow the same exponential growth pattern during the first 50 iterations until they stagnate at high fidelity, typically above 0.999. This is evidence that the denoising QAE can learn the nature of the ideal states even when its training data is mixed with polluted inputs. In contrast, past 35% of bit-flip probability, the fidelity lands on finite values far below 1 after a possibly long-awaited exponential rise or does not converge altogether. In both cases, the network's outputs cannot be used since there is no certainty that they are approximations of the ideal

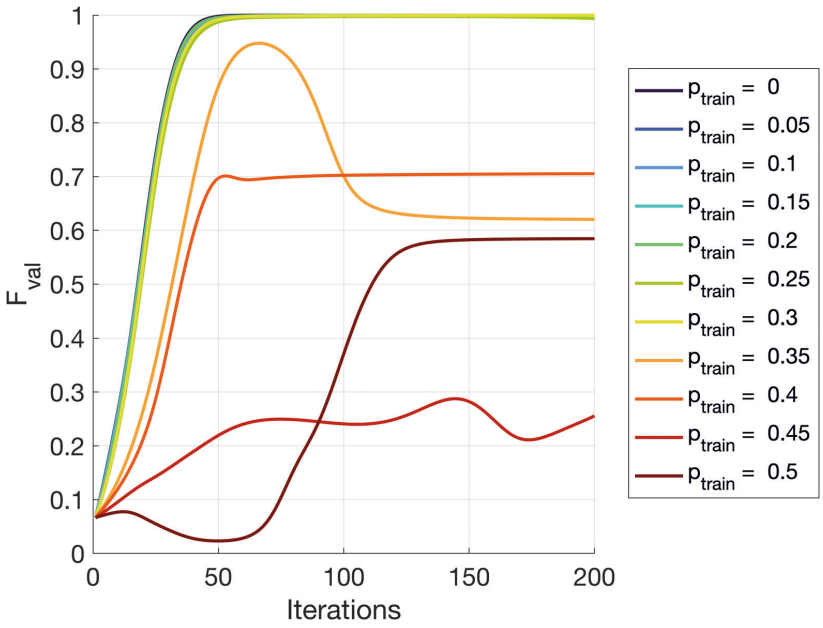


Figure 2.5: Learning curve for a [4,2,1,2,4] network, trained with Matlab. We represent the fidelity F_{val} of the denoised states with the ideal states. The inputs of the QAE were spoiled with the bit-flip channel tuned with the noise parameter p. During the early training steps, the fidelity rises exponentially and converges to some fidelity. As the noise intensity increases beyond 30%, the cost function stagnates at lower values, indicating that the network is unable to recover the ideal state.

states. We refer to the last noise intensity for which the inputs can be denoised as the "tolerance threshold" for the network.

Looking closely at the objective function, we realize that the fidelity can drop for two reasons. First, the network produces an imperfect state for any input, noisy or ideal. Alternatively, the network cannot recover the ideal state for only some noise realizations. The inspection of the outputs of the network in the strong noise regime reveals that when fidelity converges to intermediate values as for $p = 0.4$ in figure 2.5, most states are denoised, and only a subset of the possible noise realization remains faulty. In contrast, all the states are mapped to flawed states when the fidelity converges to zero, as we see in later sections. In either case, the imperfect outputs are not any random state, but they usually reproduce the symmetry of the noisy GHZ states and even particular noise realizations found in the training set.

2.3.2 Reproducibility of the results

Even though we obtained learning curves in figure 2.5 that are consistent with the literature [42], the reproduction of the tolerance threshold gives rise to variations. Indeed, the network is sensitive to the initialization conditions. First, random unitaries are selected to provide an initial proposition for the interaction parameters to update in the network. As seen from the learning curve, where the same initial set of unitaries was utilized for all probabilities, the initial fidelity is below 10%, indicating that the map implemented initially is far from the solution. Training the network with various sets of initial parameters should change the value of the fidelity after convergence in the strong noise regime. However, we expect that the tolerance threshold is equal for all of them. However, the initialization of the interactions does influence the network's performance, the leading cause for variations being the composition of the training set.

Algorithm 1 Generation of the training states

```
for x ∈ {1, ..., N_data} do
    |Ψ_x⟩ ← |GHZ⟩
    R = [r_1, r_2, ..., r_{N_in}]      ▷ Sample a random number out of U(0,1) for each qubit
    for q ∈ {1, ..., N_in} do          ▷ Apply noise on each qubit
        if r_q ≤ p then
            |Ψ_x⟩ ← X_q |Ψ_x⟩
        else
            |Ψ_x⟩ ← 𝕀_q |Ψ_x⟩
        end if
    end for
end for
```

Indeed, the translation of the equation for the bit-flip channel in 2.14 provides an averaged appreciation of the effects of noise in the limit where an infinite amount of states are considered. This description, however, is incompatible with the training data set being composed of a finite number of discrete realizations of the channel, both in the numerical simulations and on the hardware. Instead of states with amplitudes depending on the noise parameter p, the training samples only reflect the effects of some qubits being flipped or not. For example, the state $X_1|GHZ\rangle = 1/\sqrt{2}(|1000\rangle + |0111\rangle)$ can be identified as noisy, but shows no indication of the noise intensity of the channel. Instead, p is implemented in the statistics of the learning set. The pseudo-code formulation of the input generation algorithm is shown in Algorithm 1. To build a training set of N_{data} states on N_{in} qubits with noise strength p, each state is determined by a vector $\overrightarrow{R} = (r_1, \cdots, r_{N_{\text{in}}})$. It samples N_{in} values between 0 and 1 out of a uniform distribution $\mathcal{U}(0,1)$. Noise is then applied to a qubit according to the corresponding entry in $\overrightarrow{R}$ in

comparison to the noise parameter:

$$\begin{cases} |\Psi_{\text{GHZ}}\rangle \to X_i|\Psi_{\text{GHZ}}\rangle, & \text{if } r_i \leq p \\ |\Psi_{\text{GHZ}}\rangle \to \mathbb{I}_i|\Psi_{\text{GHZ}}\rangle, & \text{if } r_i > p. \end{cases} \tag{2.15}$$

where X_i is the bit-flip operator acting on qubit i.

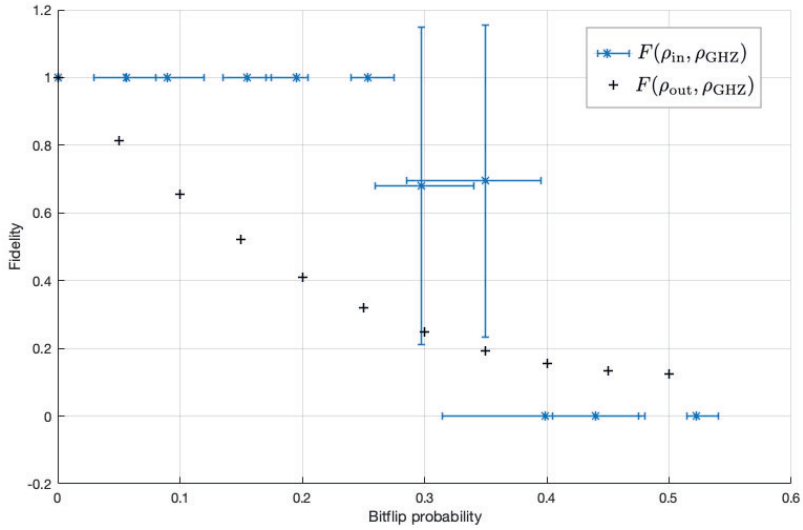


Figure 2.6: Average fidelity of the outputs of a [4,2,1,2,4] QAE during the training phase. For low noise intensities, the fidelity is close to 1 and the ideal states are almost perfectly recovered. Fidelity drops at high intensities. The definition of a clear threshold becomes blurry due to the fluctuations in the probability distribution. The horizontal error bars indicate the smallest and largest bit-flip probabilities implemented on the qubits of the inputs in the training data following Algorithm (1)

When the size of the data set tends to infinity, it is then composed of a mixture of noise realizations, each noise realization with n noisy qubits amounting for a proportion $p^n(1-p)^{N_{\text{in}}-n} + p^{N_{\text{in}}-n}(1-p)^n$ in the set. The second member in the sum results from the symmetry of the GHZ-state under the bit-flip channel, flipping a group of n qubits yielding the same state as flipping the $N_{\text{in}} - n$ qubits in its complement. This algorithm generates an accurate data set when the amount of samples it contains is large. However, the data set the network is trained with is limited in size, and variations occur in the definition of the noise intensity. The precision achieved with this algorithm may cause disturbances in the training results. In particular, two successive probabilities can overlap. The definition of the tolerance threshold is then imprecise. The fidelity

displayed at the end of the training for some probability p differs strongly from one qubit to the other. Such fluctuations are more visible for large noise parameters and become critical around the tolerance threshold. For example, if the fluctuations bring p closer to a higher value, the threshold is underestimated because the noise implemented is larger than intended. As a result, the network is unable to denoise such noise intensities. We show an example of the fidelity achieved in the testing phase on a [4,2,1,2,4] QAE in figure 2.6 and indicate the error bars on the probability distribution of the training data set. In this instance of the training, the probabilities around the threshold overlap. In particular, the largest qubitwise bit-flip probability implemented for the target noise strength at $p = 0.30$ comes close to the next one at $p = 0.35$, where the network was expected to fail. Consequently, the QAE could not denoise all states accurately.

Algorithm 2 Modified approach for the generation of the training states

for $x \in \{1, \ldots, N_{\text{data}}\}$ **do**
  $|\Psi_x\rangle \leftarrow |GHZ\rangle$
  $R = [r_1, r_2, \ldots, r_{N_{\text{in}}}]$
  $F = [0, \cdots, 0]$ ▷ Record the frequency of noise for each qubit
  for $q \in \{1, \ldots, N_{\text{in}}\}$ **do**
    if $r_q \leq p$ **then**
      $|\Psi_x\rangle \leftarrow X_q |\Psi_x\rangle$
      $f_q \leftarrow f_q + 1$ ▷ Increment on the counter for each bit-flip
    else
      $|\Psi_x\rangle \leftarrow \mathbb{I}_q |\Psi_x\rangle$
    end if
  end for
end for
$F = [f_1, \cdots, f_{N_{\text{in}}}]/N_{\text{data}}$
if $\forall q \in \{1, \ldots, N_{\text{in}}\}$, $f_q \in [p - \varepsilon, p + \varepsilon]$ **then** ▷ Add a condition to keep the set
  Keep the training set
else
  Start again from the beginning
end if

To obtain robust tolerance thresholds, we must reduce the uncertainty of the noise strength. A straightforward approach would be to increase the size of the training set. However, this is not a desirable direction since letting the number of samples grow to the thermodynamic limit increases the computational time exponentially. All training pairs take part in calculations at each iteration of the training. Another solution could be to implement stochastic gradient descent, such that only a subset of this more extensive training set is used in computations at each iteration. The convergence behavior for this learning rule is unknown.

We opt for a different approach in numerical simulations and add a verification step to the Algorithm (1). As shown in Algorithm (2), we modify the original algorithm by

keeping the final training set only under the condition that all qubitwise probabilities are accurate with precision ε. This constraint is applied to the bit-flip frequencies f_q for each qubit $q \in [1, \cdots, N_{\text{in}}]$. The data set is retained if, for all input qubits, the noise frequency falls into the interval $[p-\varepsilon, p+\varepsilon]$. Otherwise, it is discarded, and the process must start over. New random vectors R are generated for each state in the training set. This trial-and-error augmented algorithm enables us to confine the fluctuations of the probabilities within a small interval around the target noise strengths and thereby guarantee the absence of overlap in the probabilities. Consequently, a tolerance threshold can be defined univocally. We note, however, that this solution also has its limitations. If ε is chosen too small, the probabilities cannot reach the right degree of accuracy with a small number of samples, and the set is always discarded in the verification step. Therefore, the algorithm runs indefinitely. For our purpose, it suffices to choose $\varepsilon = 0.01$ for the loop to end quickly. For the same reason, a minimum number of training states must be present to enable the required precision on the probability. Finally, when the noise channel allows for a larger variety of noise realizations, or when the number of qubits increases, the condition becomes harder to fulfill, requiring longer runtime. A more clever modification may be required in these cases, as the SGD mentioned above. Since we mostly use the bit-flip channel and cannot go above inputs with 6 qubits, this approach is enough for our simulations.

With a data set generated by the modified algorithm, we train the [4,2,1,2,4] QAE for the same initial set of unitaries, as shown in figure 2.7. We confirm a tolerance threshold of $p = 0.3$ for all initial conditions on unitaries and training samples generated this way. In contrast, the non-modified algorithm limited the threshold to $p = 0.25$.

2.3.3 Quantum characterization of the QAE: Entropy

2.3.3.1 Introduction to quantum entropy

The previous section showed that a [4,2,1,2,4] QAE could recover 4-qubit GHZ states from noisy versions spoiled with bit-flips. This QML technique works for noise intensities up to 30% qubitwise probabilities. However, the reason these technique works is still obscure, and the way the QNN can denoise the states is unknown. Therefore, we need a relevant measure to shed light on how information and noise are processed in the network and how they relate to the quantum nature of the QAE.

In Shannon's classical theory of information, entropy is a central quantity to quantify information: it determines how many bits are required to encode it. It defines the capacity of communication channels that describes how fast and precise communication can be [106]. Entropy is a relevant measure to probe classical neural networks and understand how they process information.

Classically, Shannon entropy is defined for a random variable X with n possible outcomes $\{x_i\}_{i=1}^n$ occurring with respective probabilities $\{p_i\}_{i=1}^n$:

$$S_{\text{Shannon}} = \sum_{i=1}^{n} p_i \log_2(p_i), \tag{2.16}$$

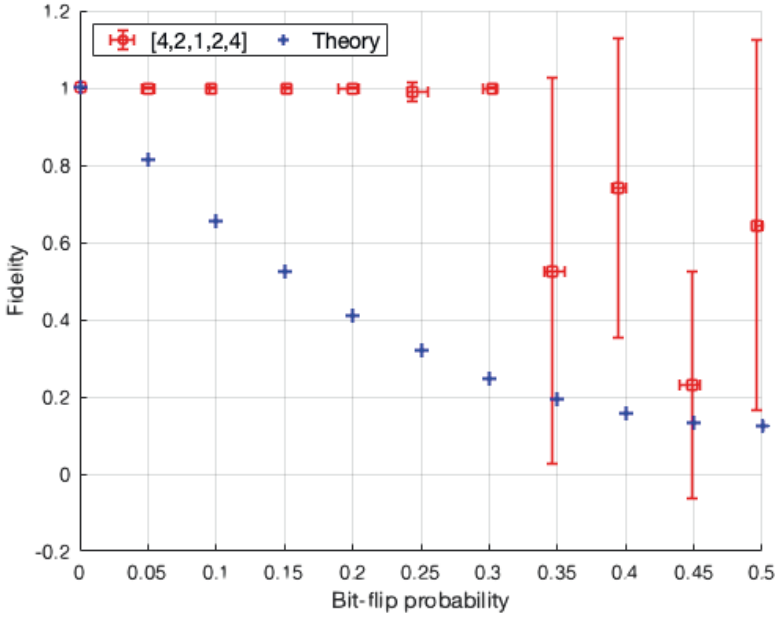


Figure 2.7: Average fidelity of the outputs during the testing phase of the [4,2,1,2,4] QAE with the same initial unitary set as in 2.6. Thanks to the modification of the algorithm to generate the training states, the actual noise strengths are now confined around their intended value and do not overlap anymore. In addition, the tolerance threshold is confirmed at $p = 0.3$ for all initial conditions. The blue + represent the theoretical fidelity without denoising using the definition of the input density matrix in equation 2.14.

where the logarithm is taken in base 2 to reflect that the information is encoded in bits. S_{Shannon} is often interpreted as a measure of disorder or surprise: it describes the unexpectedness of the data. Physically, it is proportional to the thermodynamic entropy $S_{\text{thermodyn}} = k_B S_{\text{Shannon}}$. Hence, by the second law of thermodynamics, it can only increase. Alfred Rényi's approach to entropy aims to unify different definitions of entropy. For this purpose, he defines Rényi entropy, associated with a degree M:

$$S^{(M)} = \frac{1}{1-M} \log_2(\sum_{i=1}^{n} {p_i}^M). \tag{2.17}$$

The constant prefactor is often dropped to simplify the expression [107]. The role of this constant is to rescale the logarithm such that $S^{(M)}$ does not increase with the degree

M. From equation , the definition for Shannon's entropy can be recovered by letting M tend to 1: $S_{\text{Shannon}} = \lim_{M\to 1} S^{(M)}$.

To analyze quantum information in QML, these definitions are adapted to quantum states and their density matrices. In contrast to the bra-ket formulation of the quantum states, density matrices provide the possibility to deal with states that are not pure. They live inside the Bloch sphere for a single qubit state instead of on its surface. Von Neumann entropy takes on the role of Shannon's entropy in determining the number of qubits to encode quantum information:

$$S_{\text{von Neumann}}(\rho) = -\operatorname{Tr}\{\rho \ln(\rho)\}. \tag{2.18}$$

Its natural generalization into the quantum Rényi entropy is [108, 109]:

$$S^{(M)}(\rho) = -\ln\left(\operatorname{Tr}\left\{\rho^M\right\}\right) \tag{2.19}$$

where ρ^M is the M-th power of the density matrix, and we have used the simplified formulation without the constant. Neither von Neumann nor Rényi entropy are physical observables [107]. Indeed, a physical observable should be measurable in real-time in a lab. For this reason, it must depend linearly on the density matrix. Since neither equation 2.18 nor 2.19 are linear in the density matrix, they cannot be observed. In this sense, computing them reveals non-trivial characteristics of the states. For example, second-order Rényi entropy is closely related to the purity of the states $\operatorname{Tr}\{\rho^2\}$, since the argument of the monotone logarithm is precisely this quantity. Therefore, computing Rényi entropy can provide information about dissipation in the same way purity does [68, 110]. We also notice that using second-order entropy instead of von Neumann entropy is computationally advantageous. Instead of taking the logarithm of a matrix which entails many caveats and raises various exceptions, $S^{(2)}$ only relies on matrix multiplication followed by a trace, such that the argument of the logarithm is scalar.

Apart from its interpretation as a measure for surprise and disorder, Rényi entropy bears a second important meaning in quantum physics. One of its characteristics is that it is a conserved quantity for a closed system evolving unitarily: $dS^{(M)}(\rho)/dt = 0$ in this case. Similarly, for a bipartite system where the two subsystems are not allowed to interact, the entropy of each individual system is conserved as well. However, when they exchange interactions, for example, when a qubit interacts with an environment or a heat bath as in [107], the entropy of the subsystem of interest is not conserved anymore. The following inequality holds for bipartite systems with subsystems A and B:

$$S^{(2)}(\rho_A) = S^{(2)}(\rho_B) \geq S^{(2)}(\rho_{AB}), \tag{2.20}$$

where $\rho_A = \operatorname{Tr}_B\{\rho_{AB}\}$ is the reduced density matrix for subsystem A. The equality follows from the Schmidt decomposition of the global state ρ_{AB}: the two reduced representations share the same Schmidt coefficients. Since the entropy of the reduced density is at most that of the entire system, Rényi entropy can also be understood to quantify entanglement between a subsystem and the rest of the system. In particular, in the case of the QAE, since no environment is considered, the quantum network as a whole

preserves its purity, and $S^{(2)}(\rho_{\text{net}}) = 0$, indicating that the network is in a pure state and does not share entanglement with any other system. In contrast, the entropy of each individual neuron or layer must be larger than or equal to that of the whole network: $S^{(2)}(\rho_l) \geq S^{(2)}(\rho_{\text{net}})$. Similarly, the summation of distinct entropies for multiple layers of a multipartite system is non-trivial since entangling mechanisms may be shared and repeated between many subsystems. Since the entropy of the whole network is null, a finite Rényi entropy for one layer reflects the presence of entanglement between this layer and the rest of the network.

2.3.3.2 Entropy in the network

Thanks to the Rényi entropy, we possess a measure for the evolution of information and entanglement in the network. We apply it to the QAE, starting by measuring it on the outputs. Indeed, the purpose of the autoencoder is to recover the ideal states such that it can be employed in further computations. This implies that the output states must be pure and have zero entropy. Figure 2.8 shows the average entropy of the output layer with respect to the rest of the network when the optimized unitaries are implemented. For relevant comparison, the initial conditions are the same as in figure 2.5, with tolerance to up to 30% of bit-flip noise.

At first sight, the range of noise intensities can be divided into two regions. For $p \leq 0.20$, the entropy lies at zero, and the standard deviation is also null. This indicates that the network can reach a high fidelity for any noise realization, and the output state is pure. It guarantees that no entanglement is left with the ancilla qubits in the hidden layers. In contrast, for $p > 0.20$, the entropy takes finite values, and entanglement is shared with the network. The finite values of entropies at $p = \{0.25, 0.30\}$ are of particular concern because, despite seemingly high fidelity, output states preserve some spurious entanglement with the network. This causes noise in the evolution of the states in further applications: states may escape the control of the users to the advantage of dissipation to this environment. The hidden layers would become a source of noise in subsequent usages.

These observations revive the discussion on the expression of a robust distance value in the realm of QML and quantum computing [111] since it proves that fidelity is not sufficient to report the similarity between quantum states fully. An alternative distance measure must have two properties. First, it should reflect the similarity between the density matrices. For this purpose, a common quantity found in classical machine learning can be the Kullback-Leibler distance. It provides an estimate of the distance between two discrete probability distributions with n outcomes with finite probabilities, $\{p_i\}_{i=1}^n$ and $\{q_i\}_{i=1}^n$ in the following way:

$$D_{\text{KL}}(P||Q) = \sum_{i=1}^{n} p_i \log \frac{p_i}{q_i}. \tag{2.21}$$

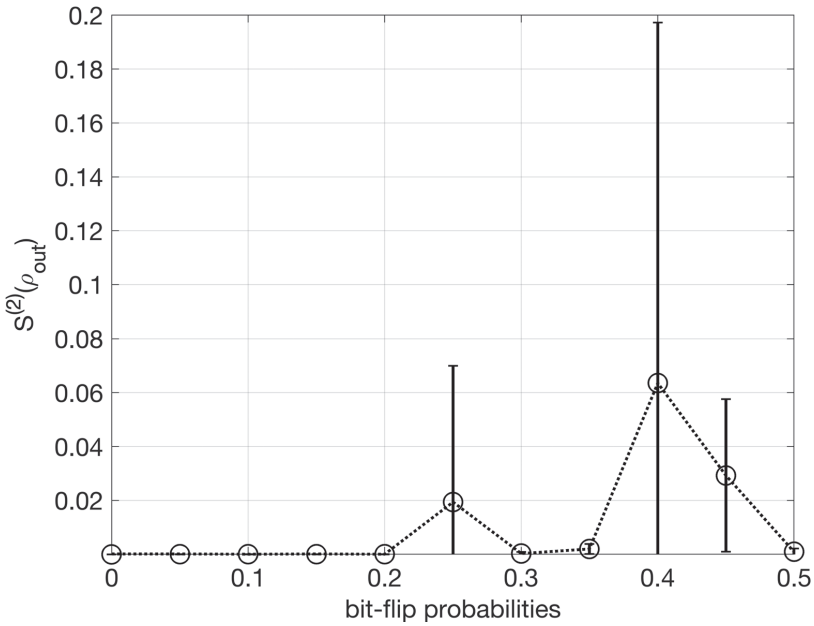


Figure 2.8: Averaged second-order Rényi entropy in the outputs of the [4,2,1,2,4] QAE against the bit-flip probabilities, during the test phase. The error bars represent the standard deviation across the 200 testing pairs that were used. The entropy in the outputs grows in a non-trivial way with the noise intensity. Its behavior changes around the tolerance threshold, defining zero and finite noise regions. In particular, even though the tolerance threshold for this network lies at $p = 30\%$, entropy becomes non-negligible at lower noise values, namely at $p = 0.25$. Therefore, even though the network was seemingly able to denoise the GHZ-states perfectly at these values when looking solely at the fidelity, recovered states at this intensity lead to noisy computations.

Second, it is desirable to include the effects of entanglement and dissipation in the measure, in the same fashion as in the Rényi entropy. Therefore we introduce the Rényi divergence between two density matrices ρ and σ, depending on an order α [112]:

$$D_\alpha(\rho||\sigma) = \begin{cases} \frac{1}{\alpha-1}\log\left(\mathrm{Tr}\left\{\sigma^{(1-\alpha)/(2\alpha)}\rho\sigma^{(1-\alpha)/(2\alpha)}\right\}^\alpha\right) & \text{if } \alpha \in (0,1)\cup(1,\infty) \\ \mathrm{Tr}\left\{\rho(\log\rho - \log\sigma)\right\} & \text{if } \alpha = 1 \\ \log||\sigma^{-1/2}\rho\sigma^{-1/2}||_\infty & \text{if } \alpha = \infty, \end{cases} \tag{2.22}$$

where ρ is such that $\mathrm{Tr}\,\rho = 1$, i.e. it is a valid quantum density matrix. We evaluate the outputs for the same [4,2,1,2,4] network with this newly introduced measure in figure 2.9, where $\alpha = 2$. The numerical accuracy of the simulation may be the cause of the fluctu-

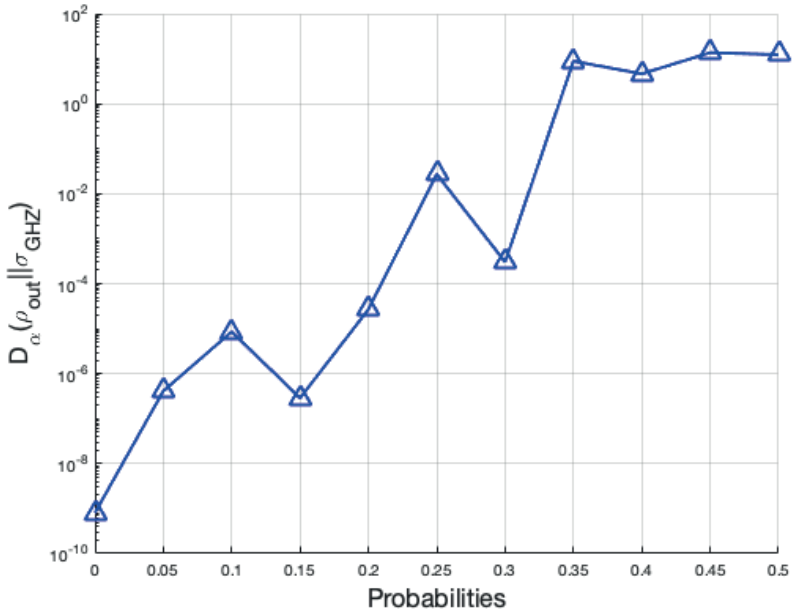
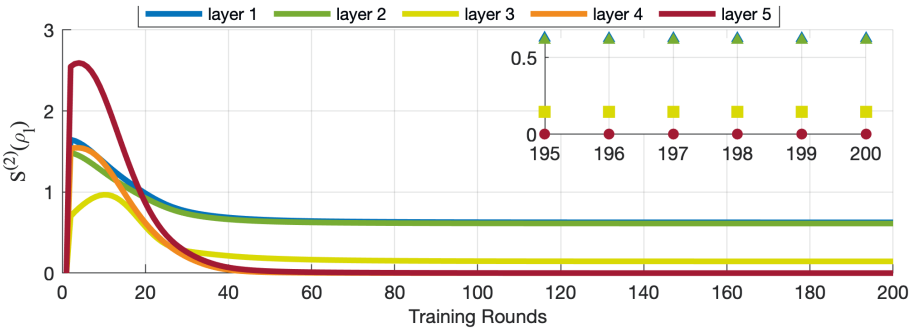


Figure 2.9: Renyi divergence at the output of the [4,2,1,2,4] in the test phase, with $\alpha = 2$. The Renyi divergence efficiently combines the high similarity and purity requirements for the output states. It increases steadily with the noise intensity, culminates at high values in the high noise regime, and is non-negligible around the tolerance threshold. As a result, it predicts that outputs in the intermediate noise regime remain fragile.

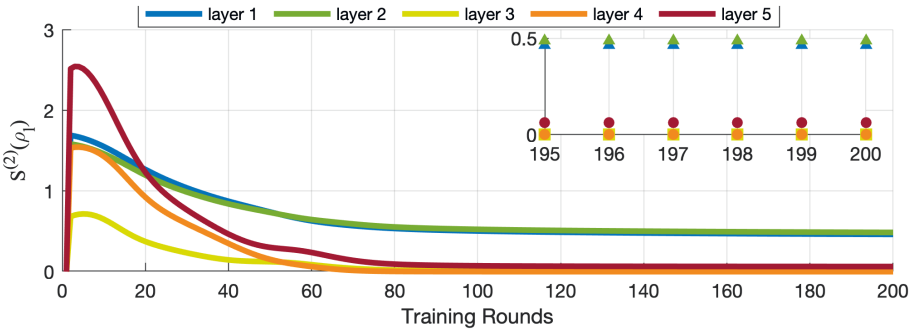
ations in the low noise regime, where it may be difficult for the computer to express and distinguish such low values with high precision. Therefore, under a threshold of 10^{-4}, we consider the divergence negligible and very close to 0. In contrast, the divergence is constant for high noise intensities and stagnates at values close to the unity. It indicates that states are different from the desired outputs and possess high entanglement with the network. In the intermediate range of noise, around the tolerance threshold, though the divergence is low, it remains finite: it reflects a high fidelity regime with spurious entanglement.

To go one step further, it is also possible to compute the evolution of Rényi entropy at each step of the training for each layer with respect to the rest of the network. In this approach, we expect (1) to understand how the map redistributes noise within the ancilla qubits in the hidden layers and uses dissipation to cancel it out on the output

layer at each iteration; (2) to witness the role of entanglement to analyze and correct the inputs. For this purpose, second-order Rényi entropy is calculated for each layer l of the network after applying the channel that propagates it to layer $l+1$. In the simulation, the state of the two successive layers is kept. The two reduced density matrices are deduced: the one on layer l is used for the entropy calculation, and the one on layer $l+1$ represents the state of being forwarded to the next layer.



(a)



(b)

Figure 2.10: Evolution of the second-order Rényi entropy for each layer at each step of the learning phase. Figure (a) represents an example in the weak noise regime, at $p = 0.1$ and (b) in the strong noise region, at $p = 0.4$. In both cases, the entropy of the output layer starts by increasing drastically in the early iterations and is exponentially suppressed throughout the training. The other layers imitate this behavior with a smaller amplitude. A notable difference across layers is the convergence value: While the encoder's layer, i.e., layers 1 and 2 maintain a high entanglement, noise is suppressed in layers 4 and 5 of the decoder. The insets zoom on the last 5 iterations and give a clear view of the last values. In particular, the gap between the latent and the output layers distinguishes successful or failed trainings.

We represent the evolution of $S^{(2)}(\rho_l)$ for each layer l in the network throughout the training in figure 2.10. Looking first at the weak noise regime at $p = 0.1$, in (2.10 a), we note that the evolution can be divided into two phases, similar to the corresponding learning curves in figure 2.5. It undergoes large variations in the first 40 steps, followed by stagnation as the number of rounds increases. In the first part, all layers gain entropy, the output layer (layer 5) at most and the latent layer (layer 3) at least. After this steep

rise, an exponential decay occurs. The entropy of the output layer is heavily suppressed, indicating that noise is disappearing from the recovered states. A softer decline is visible in the latent layer that ultimately possesses higher entanglement than the two layers of the decoder, i.e., layers 4 and 5. In contrast, the entropy of layers 1 and 2, constituting the encoder, is only mildly reduced. In the event of strong noise intensities, as in figure 2.10(b) where $p = 0.4$, the evolution follows the same steps with two exceptions. First, the decay rate of the entropy in all layers is smaller, and the first part of the training stretches over more than 80 rounds instead of 40. Stronger noise is, therefore, harder to apprehend for the QAE. Larger input variations slow down gradient descent since the direct path to the solution is less straightforward. Second, the training cannot reduce the noise of the output layer to 0 as in (a). In the last rounds, it converges to a finite value, while the entropy of the bottleneck has completely vanished. This will become important in the next chapter.

These typical evolutions can lead to a tentative interpretation of the mechanisms at play to implement denoising with a QAE. In the first phase, noise in the inputs is magnified by the random unitaries that were initially selected. It flows to the output layer. At this stage, the outputs have low fidelity. As the training progresses, entropy is reorganized between the layers, and entanglement is used as an aid to complete the task. It vanishes from the decoder, whose mission is to produce pure states. As such, entanglement in this part of the network would be detrimental, and the interpretation of entropy as a measure of disorder becomes relevant. In contrast, entropy remains high in the first half of the network. Since the encoder must extract the useful information hidden in the inputs to find the essential features of the target states, entanglement between layers can magnify its analytical power. Acting like successive filters, it extracts the noisy parts of the inputs and propagates only the relevant ones to the latent space. In this sense, entropy is beneficial in the encoder and can be treated as a measure for entanglement. This highlights the role of quantum properties in boosting noise tolerance. In the middle, the third layer corresponding to the latent space is the turning point of the network: it coordinates desirable and spurious entropies by blocking entropy in the encoder. This insulating wall against the disorder protects the decoder and guarantees the quality of the outputs.

Chapter 3

Combat the limitations of the QAE

3.1 Scaling up

The precedent chapter shows that the denoising autoencoder could be successfully converted into a quantum version. In simulations, using ideal gates trained within the network, it was possible to teach a [4,2,1,2,4] network to extract the noise from GHZ-states spoiled with the bit-flip channel. These results are auspicious to the extent that an inherently quantum network can denoise and reproduce a highly entangled state.

However, one of the main challenges of noise cancellation and error correction is the implementation at larger scales on the available quantum processors. The model should consider noise within the network and denoise states on more significant amounts of qubits. Propositions have been made to cope with the former. The authors of [113] propose to integrate noise directly in the model to train. In other words, they introduce the VQA and the noise correction algorithm simultaneously. To achieve this, they add parameters to the ansatz in the VQA. They do not interfere with the task but contribute to noise cancellation only. In this manner, the machine learning algorithm adapts to the presence of noisy gates by anticipating noise in the parameters that minimize the cost function. A suitable ansatz is needed to model the noise impinging on the hardware. This brings the problem down to the characterization issue introduced in section 2.1.2.1. We set this important question aside to focus on the scaling issues in the following.

To investigate the ability to denoise larger states, we simulate a [6,2,1,2,6] network with 6-qubit input states disturbed by the bit-flip channel as in section 2.2.2.3. We measure the fidelity of the recovered GHZ states. For consistency with the existing literature on denoising QAE [42, 105], we keep the fidelity measure to evaluate the performance of the QAE. The robustness of the QAE decreases when the size of the inputs increases. Figure 3.1 shows that while the weak noise regime for a [4,2,1,2,4] network stretched from $p \in [0, 0.3]$, it has been reduced to $p \in [0, 0.2]$ for the [6,2,1,2,6] network. This indicates that it becomes harder for the network to learn the denoising

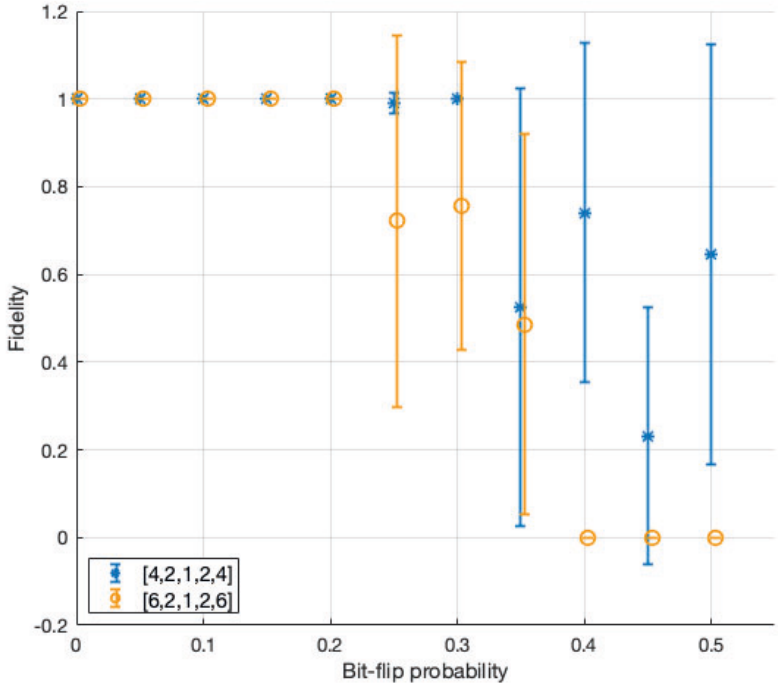


Figure 3.1: Average fidelity with the ideal GHZ-state on 4 and 6 qubits, in blue and orange respectively, measured during the testing phase. The vertical error bars represent the standard deviation across states in the testing set. When scaling up, the noise tolerance of the network drops from 0.3 to 0.2.

task on larger states. The growth of the size of the inputs results in larger diversity in the training set: there exist more noise realizations that are different than for small states. The proportion for each of them in the data set is smaller, and the generalization from patterns observed in the data becomes more difficult. Unfortunately, it was not possible to go beyond the limit of 6 qubits during the classical simulations because the training would become too expensive: we would expect a duration in the order of a week for larger scales. Despite this computational issue, it has already become clear that understanding the mechanisms underlying the tolerance of the network is crucial to scaling this technique for denoising purposes.

3.2 Structural limitation

3.2.1 Entropy flow and noise

To understand the limitations of the denoising QAE, we use the entropy measures employed in section 2.3.3. In figure 2.8, the entropy evolutions were understood as measures for the flow of noise in the network, entropy remaining high in the encoder, and vanishing in the decoder. In the middle, the latent layer possesses an intermediate value that draws a border between the two entropy regimes. The single-qubit layer is an insulating wall against the heat, which remains confined in the encoder. The latent space reduces the "temperature" of the decoder, or in other words, it makes it less disordered.

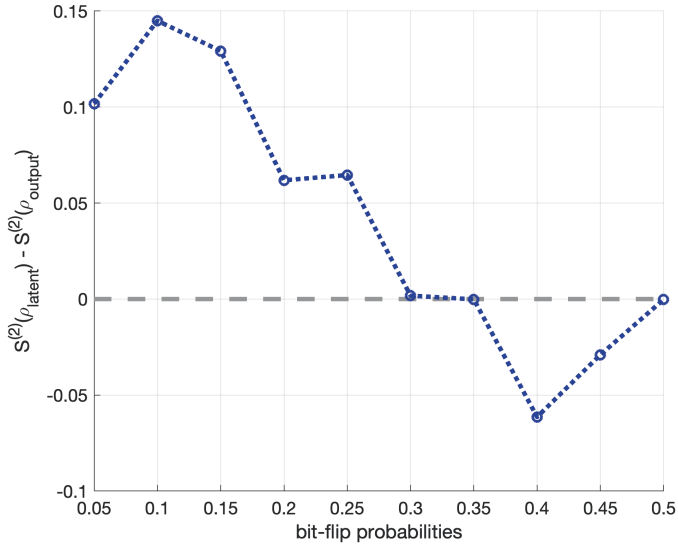


Figure 3.2: Entropy gap, evaluating the difference in the Rényi entropies between the latent layer (layer 3) and the output layer (layer 5) in a [4,2,1,2,4] network. The differences are plotted against the bit-flip probabilities p. The entropy difference remains positive in the weak noise regime, below 30%, and the sign is flipped when entering the strong noise regime. Consequently, this entropy gap is predictive of the success of the training of the denoising QAE.

Following this logic and going beyond the fidelity, an indicator of the success of the training can be the difference between entropies in the output and latent layers. We refer to the quantity as the entropy gap.

In figure 3.2, the entropy gap is positive in the weak noise regime. It culminates at a sweet spot by $p = 0.1$. In contrast, as the noise intensity reaches the threshold, the

gap closes progressively, almost becoming null at the point itself. The transition to the intense noise regime is reflected by the sign change of the entropy gap. Its negative value indicates that noise in the outputs has become more significant than that in the latent state. In the light of QAE for data compression, a possible interpretation for such results is that the compressed form is less noisy than the outputs, and the decoding operation injects more noise in the recovered state.

We propose a second measure to refine the description of the connection between the latent and output states given by the entropy gap. We construct a training set with random, pure states during the test phase. The reason for this approach and its success will become clearer in section 3.4 where a generalization test is proposed, the "cross-testing" technique, based on an idea in [105]. We apply the QAE maps obtained with different training probabilities $p_{\text{train}} \in [0, 0.5]$. The training probabilities describe the noise strength of the bit-flip channel, as in the previous sections. At each application of the quantum map, we record the second-order Rényi entropy of the latent and output states.

Figure 3.3 shows that both before (a) and after (b) the tolerance threshold, the entropy of the output states is linear in the entropy of the latent states, where each point in the plots corresponds to a different input state. These linear relations vary with the noise intensity encountered during the training. At $p_{\text{train}} = 0.10$, the entropy of the latent state can grow arbitrarily large, enabling it to absorb noise and prevent perturbations from flowing to the output layer. The entropy of the outputs is "locked" at zero, as desired to obtain an output that is separable from the network. The relation is reversed in the strong noise regime: the latent state retransmits noise that flows to the output state, where entropy can grow as large as possible. An overview of the correlations for all training probabilities is provided in the appendix C. This entropy correlation follows an interesting pattern across noise probabilities, as shown in figure (3.3 c). We perform a simple linear regression based on the mean-square error on the entropies:

$$S^{(2)}(\rho_5) = \alpha S^{(2)}(\rho_3) + \beta, \tag{3.1}$$

with α and β two scalars. The linear coefficient α between the output and latent entropies for different states is null in the weak noise regime. But its behavior changes abruptly as soon as it crosses the threshold, where it takes on finite values and culminates at $p = 0.4$. The reason for this behavior of the linear coefficient in the strong noise regime is unknown. In a loose sense, this sudden variation resembles a second-order phase transition. Until a critical noise parameter, the correlation remains zero, and it continuously changes its course past the tolerance threshold to take on finite values.

The entropy gap and the phase transition in the entropy correlations provide reasonable predictions for the success of the denoising since the threshold they indicate coincides with the tolerance defined with fidelity. Furthermore, they emphasize the importance of the latent representation for proper recovery from noise. In particular, robustness to noise of the single-qubit layer determines the success of the denoising algorithm. These results are supported by the emphasis on the latent representation in the

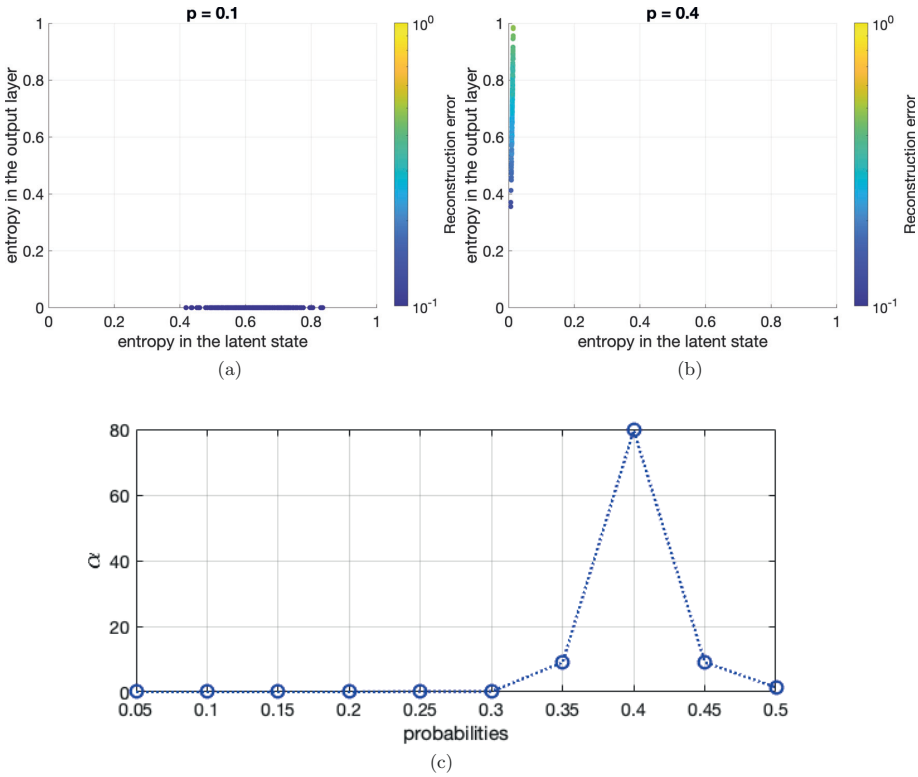


Figure 3.3: Correlations between the second-order Rényi entropies of the latent and output states in the testing stage. The test states are random, pure quantum states. The [4,2,1,2,4] denoising QAE was trained with bit-flip probabilities of $p = 0.10$ in (a) and $p = 0.40$ in (b). The color scale represents the outputs' reconstruction error $1?F$. The linear correlations are inverted before and past the threshold: in the weak noise regime, the latent state absorbs noise and ensures that the entropy of the outputs stays at zero, while the reverse happens in the strong noise regime. In (c), we represent the linear coefficient α obtained from a linear fit on the plots above: the linear factor is zero until the threshold. It takes finite values afterward, indicating that noise can flow to the outputs.

context of classical neural networks. We mentioned in section 2.2.1 that the intermediate vector at the center of the double bottleneck structure is comparable to the result of generalized principal component analysis.

Similar results have been found for the QAE based on the VQA model used for data compression. First, the authors of [93] showed that the cost function could be calculated interchangeably at the output layer or in the latent space, where it relies on the so-called trash state, i.e., the state that is discarded before obtaining the compressed state. They train a quantum autoencoder with an ensemble of pure states $\{p_i, |\phi_i\rangle_{\mathrm{AB}}\}$ living on a system AB with a classical learning rule. Their goal is to compress these states reversibly in subsystem A, by learning a unitary $U^{\overrightarrow{p}}$ depending on parameters $\overrightarrow{p}$, that encodes the states in latent space, and whose adjoint recovers the original state. In the latent representation, system B has been discarded and is reinitialized for the decoding. Equivalently, the action of discarding B is replaced by swapping it with a clean system B' initialized with state $|a\rangle_{\mathrm{B}'}$ that is the starting point for complete decoding. The output of their QAE for an input $|\phi_i\rangle_{\mathrm{AB}}$ is:

$$\rho_i^{\mathrm{out}} = (U^{\overrightarrow{p}}_{\mathrm{AB}'})^\dagger \left(\underset{B}{\mathrm{Tr}}\{U^{\overrightarrow{p}}_{AB}\, \rho_{AB}\, (U^{\overrightarrow{p}}_{AB})^\dagger\} \otimes |a\rangle\langle a|_{B'} \right) U^{\overrightarrow{p}}_{\mathrm{AB}'}. \tag{3.2}$$

The cost function is then the average fidelity for all states of the ensemble:

$$C_{\mathrm{out}} = \sum_i p_i F(|\phi_i\rangle_{\mathrm{AB}}, \rho_i^{\mathrm{out}}) \tag{3.3}$$

where F is the fidelity in equation 1.16. It amounts to comparing the two states $\mathrm{Tr}_{\mathrm{B}'}\{U^{\overrightarrow{p}}|\phi_i\rangle\langle\phi_i|_{\mathrm{AB}}(U^{\overrightarrow{p}})^\dagger \otimes |a\rangle\langle a|_{\mathrm{B}'}\}$ and $\mathrm{Tr}_{\mathrm{B}'}\{(U^{\overrightarrow{p}})^\dagger|\phi_i\rangle\langle\phi_i|_{\mathrm{AB}'}U^{\overrightarrow{p}} \otimes |a\rangle\langle a|_{\mathrm{B}}\}$. The former state is the state on system AB at the end of the encoder, while the latter is the state on AB, resulting from the propagation backwards from the output layer to the input layer of the decoder, starting from the target state. The authors show that instead of evaluating the cost function in the output layer, the fidelity of the trash system B' can be equivalently considered, and system AB can be discarded in the cost function instead of B' as in the states above. The alternative cost function becomes:

$$C_{\mathrm{trash}} = \sum_i p_i F\left(\underset{A}{\mathrm{Tr}} \left[U^{\overrightarrow{p}}|\phi_i\rangle\langle\phi_i|_{\mathrm{AB}}(U^{\overrightarrow{p}})^\dagger \right], |a\rangle_{\mathrm{B}} \right) \tag{3.4}$$

Therefore, the comparison of the states at the end of the encoder and the input of the decoder can implement the same task as the natural cost function in equation 2.11, but with fewer resources to read out the states. This property highlights that the intermediate latent representation is key to the success of the training and the full recovery of the state because it selects the relevant features for the task, or equivalently, discards the ones that it judges irrelevant. Therefore, the task is equivalently to identify the main component of the data or discard the insignificant ones. The compressed states are composed of the information that was retained.

The authors in Ref. [92] show that the purity of the latent representation sets an upper bound on the fidelity in the outputs. They implement a similar circuit for the same task, except they allow more flexibility in the decoder by training it independently of the encoder unitary. It is shown that the reversibility of the compressed state depends

on the rank of the inputs and the dimension of the latent space. Namely, a proof is provided to show that for inputs with spectral decomposition $\rho_{\text{AB}} = \sum_{j=1}^{k} p_j |\psi_j\rangle\langle\psi_j|$, such that the probabilities $\{p_j\}$ are in decreasing order and $\sum_j p_j = 1$, the upper bound on the final fidelity is given by :

$$F \leq \sum_{j=1}^{d_A} p_j, \tag{3.5}$$

where d_A is the dimension of the latent space. This bound can be achieved when the rank of the input is smaller than the dimension of the latent space, $k \leq d_A$, but has a zero probability of being attained otherwise. Therefore, the flatter the spectrum of the input, the more qubits are required in the latent space, and the worse the representation when the code space is too small. These results imply that the size of the latent space in relation to the spectral characteristics of the inputs is key to the success of the QAE. Nevertheless, this approach becomes ill-defined when looking at the denoising QAE, to the extent that the inputs and outputs do not have the same rank: the ideal GHZ-states have rank 1, while the rank of the density matrices defined in equation 2.14 vary with the probability, as shown in figure 3.4. In particular, the finite noise intensities have rank $2^{N_i n-1}$, for a 4-qubit GHZ-state with bit-flips. The uniformity of the spectrum must also be accounted for: the larger the noise becomes in the GHZ-states, the more uniform the spectrum, and the larger the error when the latent space is too small. Since the rank of the density matrix is related to its purity, a pure state corresponding to a rank one density matrix, the reversibility of the compression depends on the purity of the compressed state in latent space. More precisely, the QAE studied in this paper can reproduce only as much purity in the outputs as has been encoded in code space. In the DQNN-based QAE, this phenomenon is visible with the rise of second-order Rényi entropy past the tolerance threshold. As a result, the loss of purity in the compressed state in latent space can be responsible for the fall of fidelity in the output layer.

Finally, the authors in [95] go one step further in the spectral analysis, in the same setting as above, with a VQA-based QAE for data compression, with the constraint that the decoder is the adjoint of the encoder. They show that for such a network, the optimal unitaries obtained with singular value decomposition exactly encode the eigenvalues of the inputs in the compressed state, given that the latent space is large enough. Formally, for an input with spectral decomposition $\rho_{\text{AB}} = \sum_{j=1}^{k} p_j |\psi_j\rangle\langle\psi_j|$, the ideally compressed state σ^* in its diagonal representation is:

$$\sigma^* = \sum_{i=1}^{k} p_i |\omega_i\rangle\langle\omega_i| \tag{3.6}$$

where k is the rank of both ρ_{AB} and σ^* with spectrum $\{p_i\}_{i=1}^{k}$. The set $\{|\omega_i\rangle\}_{i=1}^{k}$ is an orthonormal basis of the latent space. If this statement still holds for denoising QAE based on DQNN, we expect the latent representation to be composed of the eigenvalues of either the ideal or the noisy state at the input of the decoder.

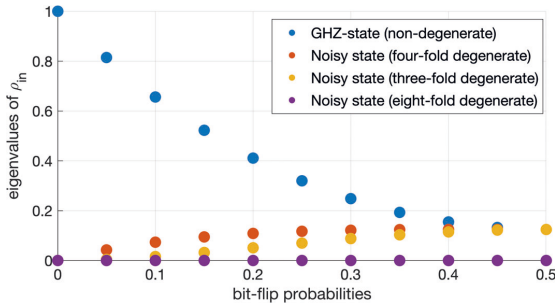


Figure 3.4: Eigenvalues of the density matrices ρ_{in} resulting from the application of the bit-flip channel on a 4-qubit GHZ-state. The eigenvalue of the ideal GHZ-state, depicted in blue, is non-degenerate. The noisy eigenvalues, in orange and yellow, are degenerate. The rank of the density matrix increases from 1 to $2^{N_{\text{in}}-1}$ when the noise intensities become finite. In addition, the spectrum becomes more uniform when the bit-flip probabilities increase. Therefore, we expect the upper bound on the output fidelity to drop in the strong noise regime, when the size of latent space is smaller than $2^{N_{\text{in}}-1}$.

This section shows that the intermediate representation of the denoising QAE is key to understanding its limitations. Its entropy acts as a wall against noise, and its failure is an obstacle to the training by letting noise propagate to the output layer. This phenomenon is embodied by a second-order phase transition at the tolerance threshold, in the linear coefficient relating entropies of latent and output states. In addition, the selection and discarding of the appropriate pieces of information make the state's recovery reversible. The network can be trained equivalently with the discarded information or the state recovered on the output layer. The reversibility of the compression is essentially related to the spectral properties of the inputs and latent representations: the latent space must be large enough to encode all eigenvalues of the inputs, and the optimal unitaries encode precisely those in the compressed state.

3.2.2 QAE and brain boxes

3.2.2.1 Boost noise tolerance with brain boxes

The observations in the previous section 3.2.1 shed light on the importance of the latent state to improve the robustness of the QAE to strong noise. Improving its architecture by creating a wider and deeper code space could increase tolerance to noise.

We propose a modification of the internal representation: the brain boxes. The underlying idea is to allow for more complex structures at the intersection of the two wings of the QAE. On the one hand, we allow the latent states to live on a more extensive system with more qubits. Following the arguments in [92, 95], this should provide sufficient degrees of freedom to encode the different eigenvalues of the average noisy state. We expect a space with N qubits to faithfully encode $2^N - 1$ eigenvalues,

where the 2^N-th degree of freedom is constrained by the normalization requirement for valid quantum states. Four qubits must match the rank criterion for complete encoding of the noisy states without information loss. On the other hand, we aim to improve the purity of the inputs of the decoder to compute pure denoised states. Since each new layer is initialized in a well-defined pure state, the addition of intermediate layers could boost purity in the compressed representations, and ultimately in the recovered states.

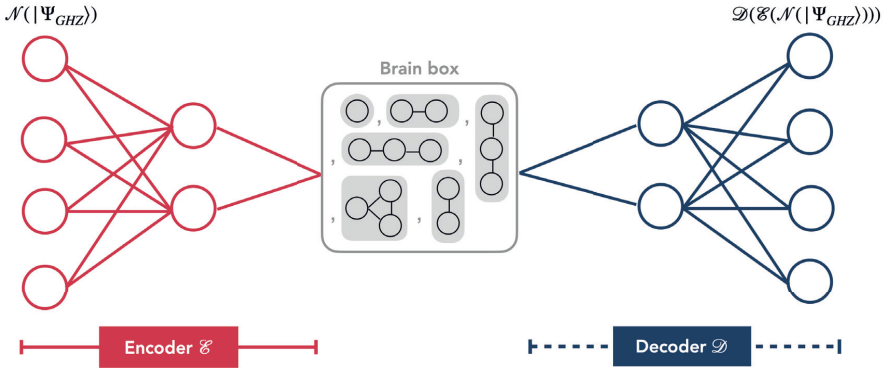


Figure 3.5: To boost the noise tolerance of the QAE, we introduce the brain box in latent space. By adding more qubits, and increasing the depth of the latent space, it is possible to encode more information and increase the purity of the intermediate representation.

We show the QAE architecture enhanced with the brain boxes (BB) in figure 3.5. To minimize the overhead of computational resources, both in time and quantum bits and gates, we limit the number of qubits in the brain box to four. They offer a flexible layout: the qubits can be organized in many or a single layer(s), symmetrically or asymmetrically. We simulated the brain boxes shown in table 3.1, resulting in [4,2,BB,2,4] and [6,2,BB,2,6] architectures. Though the initialization of the rest of the network is left unchanged, we combine the addition of complexity with another constraint. We leave the qubits independent of one another. This is equivalent to setting the interactions between qubits of successive layers within the brain box to the identity. Indeed, when looking at the compressed states, one finds that some regions of latent space act as faulty attractors during the training. If the latent state falls in these regions, it stays trapped, and the learning fails. Since the user chooses the initial unitaries, we let the unitaries in the brain boxes start from the identity to cancel any bias towards the noise sinks. Then, the learning rule updates the interactions and single-qubit rotations to find proper operations and good representations for the reconstruction. Since the parameter matrix multiplication technique in the learning rule (cf section 1.2.2.2) does not constrain the form of the unitaries, in contrast to the parametrized circuit approach, where the updates are confined within the selected ansatz, the training can still bring the parameters

closer to these sinks.

1 qubit	2 qubits	3 qubits	4 qubits
[1]	[1,1]	[1,1,1]	[1,1,1,1]
	[2]	[1,2]	[1,2,1]
		[2,1]	[2,1,1]
		[3]	[1,1,2]

Table 3.1: Brain boxes implemented to boost the noise tolerance of the QAE. The rest of the network is unchanged, and the resulting architecture is [N_{in}, 2, BB, 2, N_{in}].

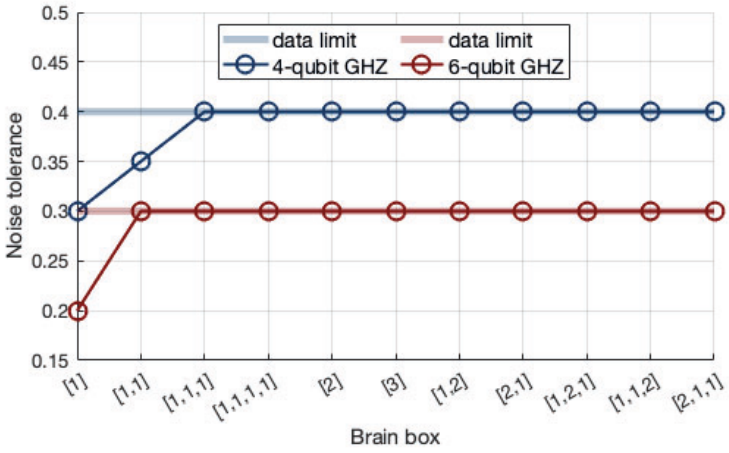


Figure 3.6: Record of the tolerance thresholds for the QAE augmented with the brain boxes shown in table 3.1 when trained with 200 states for 200 iterations. Adding complexity in the box increases the noise tolerance for both input sizes of 4 and 6 qubits. Not only the number of qubits but also their layout matters to maximize the tolerance. The computation of the data limit is explained in section 3.3. The addition of the brain boxes maximizes the noise tolerance given the data it was trained with.

Figure 3.6 summarizes the tolerance thresholds we obtained. The graphs for the testing results are presented in appendix D. The same training sets were used as in figure 3.1 except for the addition of identity operations in the brain boxes to make a direct comparison possible. The tolerance threshold is determined by selecting the most significant noise parameter to recover the GHZ-states with more than 99% of fidelity. For 4- and 6-qubit inputs, using a brain box instead of the single-qubit layer improves the tolerance. Thanks to this modification, the threshold gained ten percentage points for

both input sizes. Despite the addition of parameters and qubits, this improvement only comes at the cost of a limited overhead in resources. No apparent scaling with the size of the inputs appears in your results for four and six qubits. Indeed, looking closely at the training algorithm, one realizes that the maximal number of qubits required to simulate it is the largest cumulated amount in two successive layers, namely $\max_l N_l + N_{l+1}$, for $l = 1, \cdots, L-1$. Therefore, the brain boxes do not induce any overhead in quantum resources, because their width does not exceed the size of the inputs. In addition, the results of the [4,2,BB,2,4] network in figure 3.6 suggest that for the same amount of qubits, different thresholds can be realized. In particular, two qubits in a linear chain fail to saturate the data upper bound, while stacking them in a single layer maximizes the achievable robustness.

Another essential property to consider when implementing an algorithm, especially when simulated classically, is the time resources it involves. For this reason, we provide an overview of the time required to train the network, both in the number of iterations it takes to converge to high fidelities and in real-time. In the latter quantity, we divide time by the number of layers in the network, to make it possible to compare different layouts. Figure 3.7 hints that almost all brain boxes reduce the time required for the training, except for the configuration [3]. The simpler the operations, i.e. the more linear the layout, the least time the training takes. In particular, when the brain box ends with a single qubit layer, the unitaries that connect the brain to the decoder are smaller and easier to update. Hence, the number of iterations to converge to a high fidelity is smaller. From another perspective, one can think that the sole degree of freedom of the single-qubit layer can only capture the single eigenvalue relevant to the pure, ideal GHZ-state, thereby making the reconstruction easier. These properties on the time resources may vary when the network is run quantum mechanically.

3.2.2.2 Perspectives on the brain box

Without providing a definite answer as to why the brain box-enhanced QAE improves the tolerance threshold, we propose two possible directions to approach the question. First, the addition of parameters thanks to the brain boxes can improve under some conditions. In particular, the change of behavior for different brain boxes can be connected to overparametrization, a well-known topic in the recent research in machine learning. Overparametrization corresponds to a regime where the number of parameters is huge [114]. The model can precisely capture all training data features thanks to their fine-tuning. In the classical interpretation, the overparametrized model is expected to overfit the training data. In other words, the model should perform by memorizing the learning data instead of extracting the relevant features for the task. A low error detects overfitting during the training, i.e. low values of the cost function to minimize, contrasted with a high validation error in the testing phase.

This view on overparameterization is broken by the so-called double-descent phenomenon [115]: when the number of parameters keeps increasing, the model ceases to overfit. The generalization error is considered from the perspective of the variance-bias

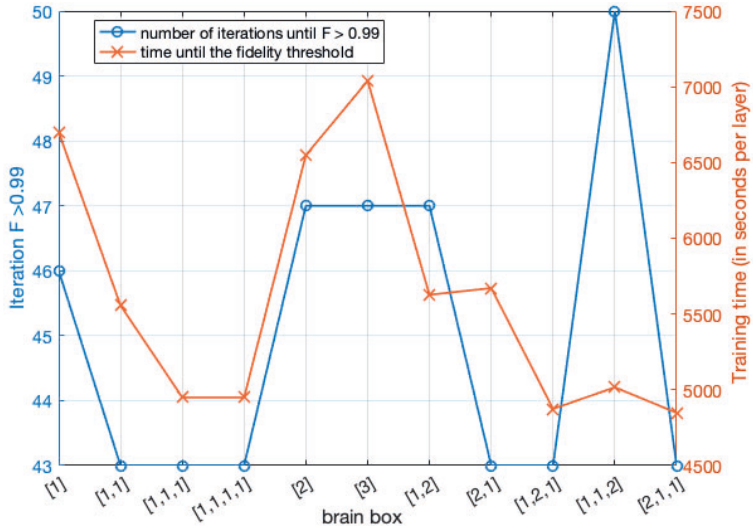


Figure 3.7: Time required for the training of brain box enhanced QAE with 6-qubit inputs. Time is measured by the number of training rounds to converge to high fidelity states, and seconds per layer.

trade-off. It takes the squared bias, variance, and some noise contributions. The bias corresponds to the distance between the estimates and the targets and therefore accounts for the training error. The variance evaluates the sensitivity of the results to variations in the data set [24, 17]. As such, it is equivalent to a testing error. The traditional approach is to choose the number of parameters that balances these two sources of error. As the number of parameters increases, the bias drops because the model captures more details about the data. At the same time, the variance increases, and the model starts to overfit. Consequently, the classical perspective defines a sweet spot to balance the two contributions and minimize the generalization error. Nevertheless, this picture is incomplete. As the number of parameters increases, a new "phase" emerges. The generalization error drops again and undergoes a second descent. In contrast to the classical regime, the so-called "modern interpolating regime" [115] avoids overfitting despite the expressivity of the model.

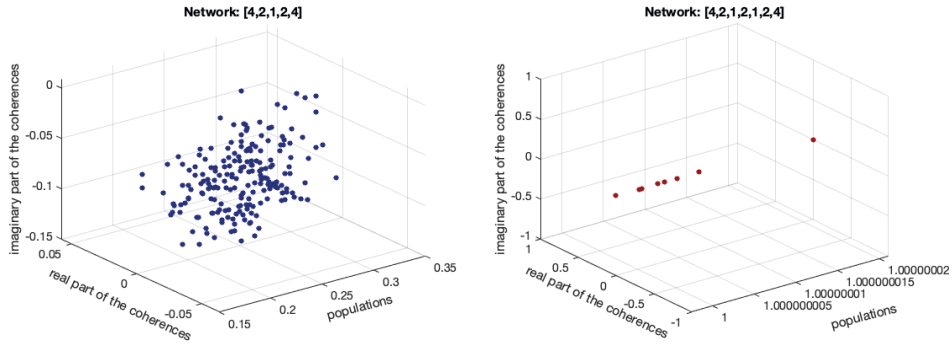
The number of parameters in our brain-box enhanced QAE has increased, possibly bringing the model to the overparameterization regime. The identification of the parameter regime depends on the demands on generalization for the QAE, as discussed later in section 3.4. The most demanding definition of generalization requires the network to perform denoising on many inputs affected by various noise sources. According to this approach, the performance of the QAE is poor on new data and worse with the

brain boxes. This is a signature of the detrimental overparameterization regime: the model overfits its training states. The number of parameters is too small to reach the second descent. In this case, the network can correct GHZ-states only and does not help any other state. In an alternative perspective on generalization, the QAE copes with any type of noise occurring in the inputs and recovers the desired flawless GHZ-state. The brain box is sufficient to reach the second descent region, i.e., the modern interpolating regime. In addition, the study of overparametrization in the context of VQA-based QAE [116] suggested that the bound on the number of parameters to reach the second descent is not tight. For an unexplained reason, the QAE goes to the current interpolating regime earlier than expected, when other types of networks come close to the bound. Therefore, only a few parameters are required for the computational phase transition. This is facilitated by the exponential scaling of the number of parameters with the number of qubits in the brain boxes, and ultimately, in the whole network.

We present a second argument besides overparameterization to explain improvements in the robustness thanks to the brain boxes. We look in detail at the states in the latent space that serve as inputs to the decoder. To represent it in three dimensions, we compare the brain boxes [1] corresponding to the original QAE, with a [1,2,1] box, providing a maximal noise tolerance in a minimum of time, and ending with a single-qubit layer. We plot the density matrix in a 3D plot to show the correlations between the (classical) populations on the diagonal entries, and the (quantum) coherences on the off-diagonal terms. For the real diagonal components, a single entry suffices thanks to the normalization of the quantum state. The two complex off-diagonal coherences are adjoints with one another. Therefore, three dimensions give a complete representation of the state.

In figure 3.8, we represent such latent states for the two networks when the input data are random quantum states. The simplification of the representation is striking. While the latent states in the original QAE build a fuzzy cloud around some approximate linear correlation between the populations and coherences, the linear mapping is impressively well-defined in the [4,2,1,2,1,2,4] network. This remapping of the input data into a different space can be related to the feature space encoding in classical machine learning. In this framework, some data is mapped to some different space to visualize some inner structure it may possess. For example, data that is not linearly separable can sometimes become separable. Similarly, one can consider that in the case of the QAE, the network must decide whether the state is noisy or not, or which noise realization occurred. In this case, a linear mapping as in 3.8(b) reduces the dimension of the decision boundary from 3-dimensional to 1-dimensional, and thereby facilitates the distinction and recovery.

The question of the distinguishability between the state in the context of the QAE can be confusing though. Indeed, while one wants to distinguish between the noisy and ideal states in the inputs to denoise only the former, the ultimate task given to the QAE is to recover the ideal state only. The question of distinguishability is hence two-sided. To reconcile these two perspectives on the distinction between states, we appeal



(a) (b)

Figure 3.8: Latent states in the [4,2,1,2,4] (a) and last layer of the brain box of the [4,2,1,2,1,2,4] (b) network. Each point corresponds to a different, random input in the network, that is recovered with a fidelity larger than 99.9%. The addition of the brain box enables the network to find a more suitable intermediate representation of the inputs, that concentrates them on a line.

to Quantum Hypothesis Testing (QHT) as in [51]. Indeed, the authors of this paper use QHT tools to prove the robustness of QML algorithms for classification tasks subject to adversarial attacks. To tolerate such attacks, a minor change in the inputs must not be able to change the output label for the corresponding noiseless input. Therefore, the noisy state must be mistaken for the ideal one. In contrast, QHT is concerned with classifying a state as some desired state and telling two states apart in the most optimal way relying on the suitable Helstrom measurement. QHT and the QAE can meet when looking at table 3.2 where the possible outcome of QHT are summarized. When the outcome of the measurement and the actual state coincide, no error happens. However, two types of errors can happen: type I error occurs when the target state is mistaken for a noisy one, and conversely, type II occurs when the noisy state is labeled as ideal. A robustness bound can be determined according to the distance between two states, such that type II error takes place for this state, and the state is treated as the ideal one.

In order to use a QHT argument, we plot the latent states on the Bloch sphere for both [4,2,1,2,4] and [4,2,1,2,1,2,4] networks in figure 3.9. This visualization emphasizes the region of the qubit Hilbert space occupied by the latent states. While they spread in a relatively large region in the southern hemisphere in the original network (3.9 a), they concentrate in a tight region for the [4,2,1,2,1,2,4] network (3.9 b). In this case, no matter what the inputs are, they are all redirected to a point-like area. Therefore, in terms of robustness and QHT, the states become less distinguishable, and the latent representation creates a bias towards the desired type-II error. In turn, it leads to

		Verdict	
		Ideal	*Noisy*
Reality	*Ideal*	no error	type I
	Noisy	type II	no error

Table 3.2: Outcomes of a QHT protocol. The columns represent the verdict based on the measurements, and the rows the actual nature of the states. On the diagonals, the classification of the state coincides with their true nature, while the off-diagonal cells reflect faulty judgement.

the reconstruction of the ideal GHZ-state in the outputs. As a result, by shrinking the space where the inputs are mapped in the latent space to a tiny region, the noisy states are mistaken for the ideal ones and the network reconstructs only the ideal states. In addition, we note that the point corresponds exactly to the diagonal matrix with eigenvalues 0 and 1, as predicted in the ideal cases [92, 95].

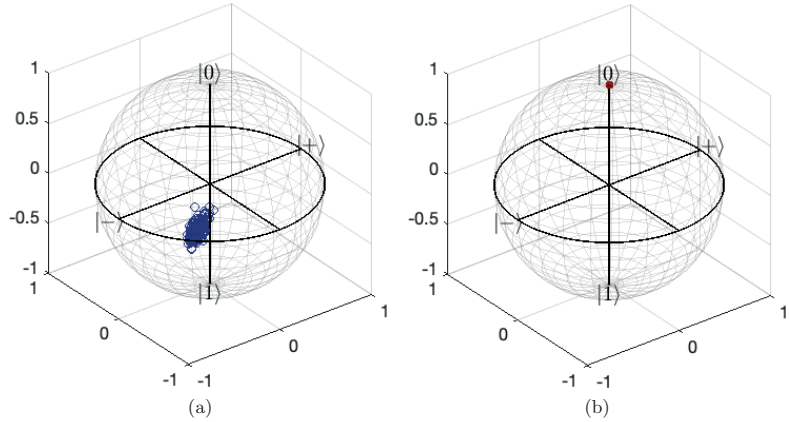


Figure 3.9: Comparison of the latent states on the Bloch sphere, for the [4,2,1,2,4] (a) and [4,2,1,2,1,2,4] QAE (b). The states in the original network occupy a larger region of the Hilbert space of the latent representation, while the brain box-enhanced QAE concentrates around the maximally mixed state.

3.3 Training data and absolute upper bound on the tolerance

In the previous section 3.2.2.1, we successfully improved the robustness to noise of the QAE by complementing the architecture with brain boxes. However, these results justify the following question: Why does not increasing the complexity of brain boxes increase the noise tolerance beyond some limit? The uniformity of the upper bound on the tolerance is striking because it falls when the inputs grow in size. Therefore, identifying this limitation could further contribute to understanding the denoising QAE and perfecting it.

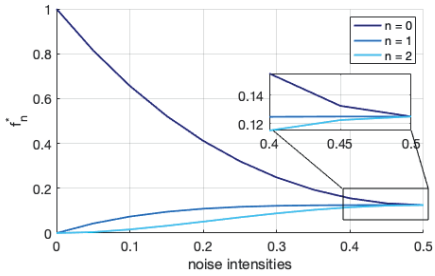
To answer this question, we draw inspiration from classical machine learning, where the amount and quality of the data available for the training are critical to the success of the algorithms [27]. The training algorithm for DQNN was initially formulated for supervised learning. We expect that the approximation of the learning rule with unsupervised learning leads to limitations in completing the task due to missing information in the training dataset. In our implementation of the QAE, the training data is best described by the proportion of each noise realization in the set. Therefore, when the cost function "teaches" the network, the only information it can provide to steer the training to the desired state is encoded in the data set itself.

We start the investigation by visualizing the training data in two limits in figure 3.10. For each noise realization with n bit-flips occurring on N_n samples among the N_{data} states in the data set, we define the recurrence $f_n = N_n/N_{\text{data}}$. We check in figure (3.10 a) what an ideal training set would look like, in the limit where infinitely many states would be available. We denote this regime as the thermodynamic limit. It is obtained from the equation of the bit-flip noise channel in equation 2.14. For states with N_{in} qubits, we cumulate the recurrences for those with n and $N_{\text{in}} - n$ bit-flips due to the symmetry of the GHZ-states under the bit-flip channel: states with n flipped qubits can be obtained by flipping the $N_{\text{in}} - n$ qubits in the rest of the state. Hence, the analytical form of the frequencies for the noise parameter p is:

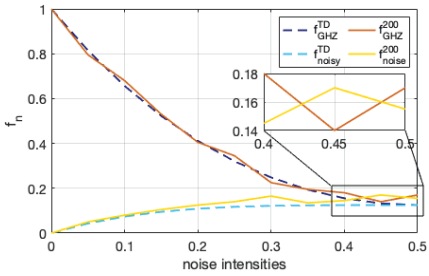
$$f_n^* = p^n (1-p)^{(N_{\text{in}}-n)} + p^{(N_{\text{in}}-n)} (1-p)^n \text{ , such that} \tag{3.7}$$

$$\sum_{n=0}^{\lfloor N_{\text{in}}/2 \rfloor} \binom{N_{\text{in}}}{n} (p^n (1-p)^{(N_{\text{in}}-n)}) + \binom{N_{\text{in}}}{N_{\text{in}}-n} (p^{(N_{\text{in}}-n)} (1-p)^n) = 1. \tag{3.8}$$

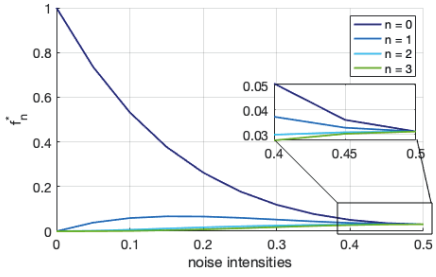
By observing the results in figure 3.10 (a) and (c), we notice that over the whole range of noise intensities, the ideal GHZ-state amounts for the most significant proportion in the training set compared to the other noise realizations, for both input sizes of 4 and 6 qubits, in (a) and (c). Its dominance vanishes only at $p = 0.5$ where the second most frequent noise realization represents the same proportions as the ideal state. In contrast, limiting the size of the training set to 200 states as in 3.10 (b) and (d) introduces fluctuations around the frequencies defined in the thermodynamic limit. These variations



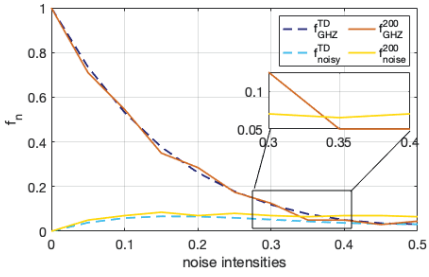
(a) Thermodynamic limit for 4-qubit inputs



(b) Training set with 200 states on 4 qubits



(c) Thermodynamic limit for 6-qubit inputs



(d) Training set with 200 states on 6 qubits

Figure 3.10: Frequencies of the noise realizations with n qubits flipped, in the thermodynamic limit ((a) and (c)), and frequencies for the two dominant noise realizations in training set with 200 training pairs in (b) and (d). While the ideal GHZ-state remains dominant until $p = 0.5$ in the case of an infinitely large data set for both input sizes namely 4 in the first, and six qubits in the second row, it is outnumbered earlier at $p = 0.45$ and $p = 0.35$ for the set with finite size, for 4- and 6-qubit inputs respectively.

become essential in the strong noise regime because the frequencies of the ideal and most probable noise realizations are very close to one another; minor variations can reverse the dominance: the noisy state is more frequent than the perfect state. To the extent that no other hint is given to the network to point out the desired output of the network, the fluctuations play an important role and can cause the failure of the training.

We check whether this concurrence between noisy and ideal state is indeed at the origin of the absolute bound seen in the threshold in figure 3.6. For this purpose, we artificially compose a training set of 200 states with proportions close to those of the thermodynamic limit, such that the crossing takes place only at $p = 0.5$. This approach is necessary to the extent that sets with more than 1000 training samples would be required to limit the variations to a minimum. However, such a network is very time-

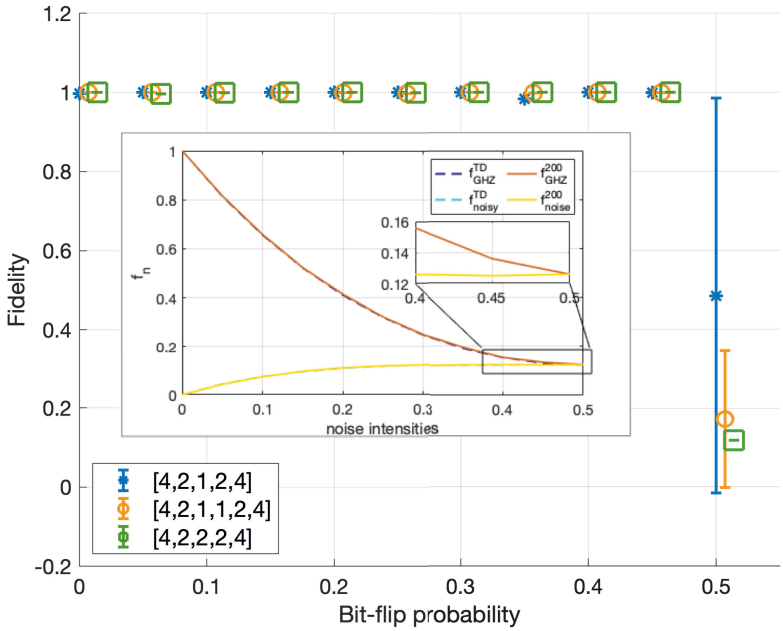


Figure 3.11: Average fidelity during the testing phase, measure for a network optimized with a training data set of 200 training pairs manually engineered to be a good approximation of the thermodynamic limit. The error bars correspond to the standard deviation over data samples. The inset displays the recurrences of the perfect GHZ and the most frequent noise realization in the data set. When the data set is close to the thermodynamic limit, the tolerance threshold amounts to 50% and the original and brain box-enhanced QAE are equivalent in performance.

consuming to train. Therefore, we keep the finite size of the set and adjust the frequencies manually. The distribution we obtain, shown in figure 3.11 (a), is smooth and displays only minor variations around the thermodynamic limit. In particular, as intended, no crossing takes place between the ideal GHZ-state and the possible noise realizations, except at 50% bit-flip probability.

We show the results of the trainings for three networks in figure 3.11 (b): the original [4,2,1,2,4] QAE, and two different brain boxes, [4,2,1,1,2,4] and [4,2,2,2,4]. In the three cases, the tolerance thresholds are extended to 45%, and the strongest noise parameter $p = 0.5$ is the only intensity the QAE cannot correct. To allow for comparison with the previous results in figure 3.6, the network was initialized with the same unitaries and trained for the same duration. We conclude from this simulation that the training set imposes an absolute bound on the noise tolerance of the QAE. When the frequency of

the ideal GHZ-state does not dominate the training set, and some other state is more recurrent, the QAE identifies the most recurrent state as the target state. Indeed, in the approximate supervised learning rule that we implement, the outputs have been replaced by noisy states as well, and the network has no other resource to understand the task it is given.

We check the consistency of our interpretation against results in the literature. The authors of [13] derive a scaling between the generalization error in VQA-based QML algorithms and the size of the training data sets. They consider a model that includes T trainable local gates parametrized by a variable α and that can be combined in G_T different configurations. Each gate is repeated at most M times in the circuit, and the t-th gate changes at most by an amount Δ_t during the learning phase, with N_{data} training samples. It holds that with high probability, the generalization error scales as follows:

$$\text{gen}(\alpha) \in \mathcal{O}\left(\min_{K=0,\cdots,T} f(K) + \sqrt{\frac{\log G_T}{N_{\text{data}}}}\right) \tag{3.9}$$

$$\text{where } f(K) := \sqrt{\frac{K\log(MT)}{N_{\text{data}}}} + \sum_{k=K+1}^{T} M\Delta_k, \tag{3.10}$$

where $K \leq T$ reflects the number of gates that have undergone a significant change Δ_t during the training. This means that the generalization error is proportional to $\sqrt{1/N_{\text{data}}}$ and decreases when more samples are added to the data set. Therefore, the scaling of the generalization error in our network seemz consistent with these results.

Furthermore, we note in figure 3.11 (b) that the tolerance of the original QAE is as good as the brain box-enhanced QAE, and the variations between different inputs vanish as well for the denoised states. It indicates that if the training data is inexpensive to produce and the computational resources for the training do not grow exponentially with its size, the addition of the brain boxes becomes irrelevant, to the extent that it only adds overhead quantum resources and time, without changing the quality of the output over the valid noise range. Nevertheless, since up to this day, producing these vast data sets is costly, and finite-size training sets are more advantageous, the brain box can compensate for this lack of data and saturate the maximum bound on the noise tolerance. In this case, the original QAE would underperform.

3.4 Generalization performance

The amount of noise tolerated by the network depends on the data set it was trained with can raise some concerns. Indeed, this can be addressed with the decomposition of the generalization error [24, 17] into the sum of the squared bias, the variance, and some additional noise term as in section 3.2.2.2:

$$\text{generalization error} = (\text{bias})^2 + \text{variance} + \text{noise}. \tag{3.11}$$

The second term is directly connected to the sensitivity to variations in the inputs: the QAE yields different fidelities in its outputs according to the data samples it acts

on after the training. In particular, it would enable the recovery of a subset of states only. However, an explicit criterion for generalization seems difficult to formulate for QAEs since its primary task is precisely to reproduce the inputs, and in the case of the denoising QAE, to recover some target state. Consequently, the question arises: what is a desirable generalization for the denoising QAE?

This interrogation goes alongside the formulation of the task. Indeed, as seen in section 3.3, the training lands on different results in the weak and strong noise regimes because the target state is no longer dominant in the training set in the latter. Consequently, since the cost function in equation 2.11 that gave rise to the update rule is based mainly on noisy states, the task of the QAE has been changed unwittingly. Instead of asking to reproduce the ideal GHZ-state, the cost function designates a noisy state as the desired output, because it weighs more in the sum over states ρ_x^{in} in the cost function. Therefore, a univocal reformulation of the cost function might be a way to improve noise tolerance and generalization. This new task expression should perform well despite the ideal states not accessible on noisy quantum hardware.

The definition of the cost function might instead seek a balance between the information that must be kept or discarded. Similar to generative adversarial models, the cost function could be composed of two competing terms: one to tell states apart from one another, and a second to generate accurate copies of the inputs. The network is aware of the features that make two states distinguishable, e.g. a bit-flip, while the generative term embraces diversity in the targeted states. This approach could be investigated with the Quantum Generative Adversarial Networks (QGAN) architecture, as proposed in [117, 118, 119, 120], or find a formulation in terms of optimal measurement as in Quantum Hypothesis Testing in an adversarial setting, as shown in section 3.2.2.2. The information bottleneck framework would also penalize memory effects on the training data while rewarding the selection of proper features to reach the desired outputs [121].

In addition, refining the formulation of the learning task requires, first and foremost clarity on the type of generalization sought with the QAE. Namely, does generalization mean (1) that the network can handle any noise channel acting on the desired state? (2) can it filter bit-flips out of a wide diversity of states? (3) can it remove any kind of noise in any state? This ill-defined definition of generalization is a well-known problem for generative models and becomes particularly important when comparing classical and quantum algorithms. To address this issue, Ref. [47] proposes a quantitative evaluation of generalization based on three criteria. First, the model's pre-generalization ability characterizes the network's capacity to explore the space outside its training samples and produce novel outputs. The second criterion looks at the validity of the generated samples. After passing the pre-generalization test, the outputs produced by the network must belong to the solution space. Thirdly, the quality of the solution is appreciated by its ability to minimize some cost functions. For example, if the network is asked to output touristic destinations as cities, the pre-generalization step would require that the network can output locations on a map that it did not see during the training. A valid output would have to be an existing city. Finally, a quality output should be attractive to tourists and minimize the cost function associated with their dissatisfaction.

Since the autoencoders have been shown to work as generative models [90], such criteria can be used to evaluate the generalization in denoising QAE as well. However, in the first stage, the QAE fails to explore states out of the data set it has been trained with. This requirement would correspond to option (3) above: all states could be completely denoised. However, this is not possible to the extent that if it can work with all states, a state and its noisy version would be valid target states, and the network must be given additional clues to determine whether noise occurred or not. Ideally, noise must map target states in a subspace orthogonal to the desired states. We therefore eliminate option (3) as a valid way of understanding generalization for the denoising QAE in our approach.

The second view of generalization is investigated in Ref. [42]. The authors trained a [3,1,3] QAE to denoise bit-flips on 3-qubit states. During the training, the inputs are ϕ-GHZ states $(|000\rangle + e^{i\phi}|000\rangle)/\sqrt{2}$ with three different phases. In contrast, a continuous range of phases is implemented during the testing. Their results indicate that the QAE can still recover the desired states accurately. Nevertheless, the fidelity drops close to 0.8 when the noise parameter increases, even below the tolerance threshold. Therefore, the denoising QAE trained with a small variety of samples can recover diverse targets when they are all impacted by the same noise channel. As a result, the second view on generalization we proposed is fulfilled to some extent by denoising QAE.

Finally, we examine option (1) for generalization in the QAE, namely whether the network can filter noise from various noise channels for the same target state. For this purpose, we introduce a new test, the "cross-test". As in a usual test, or "symmetric test", we apply the trained map onto a new set of unseen data and measure the average fidelity of the outputs. However, we construct asymmetric the data sets, similar to [105], where the map is trained with a bit-flip probability of $p_{\text{train}} = 0.2$, but the fidelities during the testing are reported for test probabilities in the range $p_{\text{test}} \in [0, 0.5]$ for the bit-flip channel. We generalize this approach beyond the bit-flip channel and design testing data with noisy GHZ-states perturbed with (1) the bit-flip channel in equation 2.14 with different noise intensities as explained above, (2) the depolarizing channel (cf eq. 3.12 below) and (3) an approximation for the erasure channel. We provide the results for two different brain boxes, [1] and [2] in table 3.3.

Starting with the bit-flip channel in the first row, a shared feature for both networks is the insensitivity of the trained map to the noise intensity, as suggested by the nearly horizontal reconstruction error. Once the quantum channel has been trained, it handles different bit-flip probabilities similarly and produces the same outputs. This can be related to the fact that the noise parameter in the training set represents frequencies in the set, but the states themselves are the same. Therefore, the network has been familiarized with all the states that can be reached with the bit-flip channel starting from the GHZ-state. The comparison of the two architectures reveals that the tolerance thresholds are maintained respectively at 20% and 30% for the original and brain box enhanced-QAE. However, the reconstruction error in the latter becomes non-negligible, since fidelities take values between 99.99% and 99% close to the tolerance threshold.

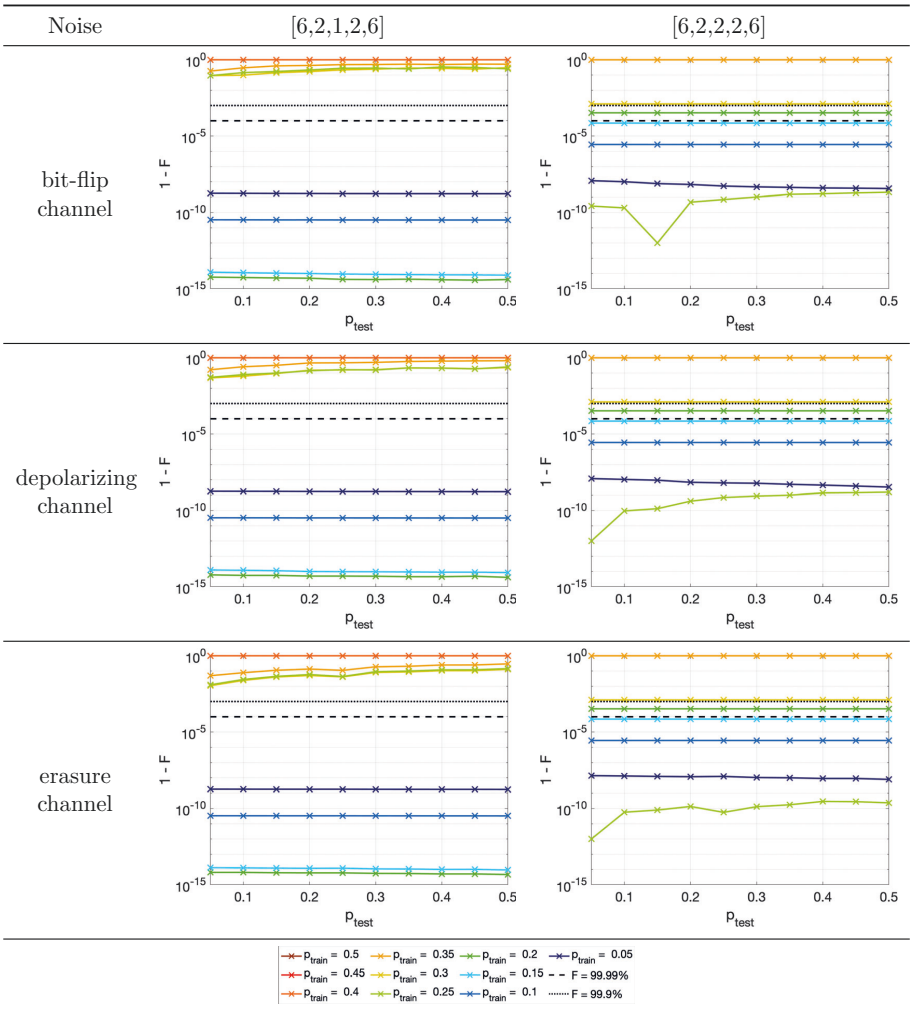


Table 3.3: Reconstruction error for the cross-testing simulations of two QAE architectures with 6-qubit inputs: the original one in the first column, and the one enhanced with a brain box [2]. Three types of noise are investigated, the bit-flip channel, the depolarizing channel and the erasure channel. Once trained with the bit-flip channel with a noise parameter below the tolerance threshold, the QAE can denoise any noise intensity in the same channel, as shown by the linearity of the results.

Despite a seemingly small drop, the 1% of fidelity that is missing may cause damage in the evolution of the outputs when employed in other computations. The accumulation of such error rates in an algorithm would lead to erroneous results.

We also implement different noise channels that can lead to a wider variety of states, particularly those not present in the training set. Firstly, we simulate the depolarizing channel with noise parameter p on qubit i in the state ρ as follows:

$$\mathcal{N}_i^{\text{depol}}(\rho, p) = \frac{1-3p}{4}\,\mathbb{I}_i\rho\,\mathbb{I}_i + \sum_{k=\{x,y,z\}} \frac{p}{4}\,\sigma_i^k\,\rho\,\sigma_i^k \tag{3.12}$$

where the σ_i^k correspond to the Pauli matrices $\{\sigma^x, \sigma^y, \sigma^z\}$ acting on qubit i and $\mathbb{I}_i$ to the identity operation. As in equation 2.14, the channels for individual qubits are concatenated to provide the final state. To implement this channel in the code, we ensure the accuracy of the noise parameter as in Algorithm (2). The checks apply solely on the proportion of noisy states in the set and no trial-and-error is used to balance the three possible noise types, despite the use of a second random number to sample them. The second row in table 3.3 reveals that the QAE generalizes astonishingly well to the depolarizing channel: despite the exponential increase in the variety of states in the inputs, the reconstruction error remains equal to that in the bit-flip case.

This observation is confirmed by the application of a third noise channel, inspired by the quantum erasure channel [106, 122]. This noise model can represent the spontaneous emission in an atom. It divides the Hilbert space of a quantum system into the computational space, and a Hilbert space that is orthogonal to it:

$$\mathcal{H}_{\text{syst}} = \mathcal{H}_{\text{comp}} \bigoplus \mathcal{H}_{\text{comp}}^{\perp}. \tag{3.13}$$

In the ideal case without noise, the state remains in the computation Hilbert space $\mathcal{H}_{\text{comp}}$ with a probability $1-p$. In contrast, noise arises when leakage to the orthogonal Hilbert space $\mathcal{H}_{\text{comp}}^{\perp}$ happens. Therefore, the information in the computational basis is erased, but a signal notifies about this event. For example, the emitted photon in the previous example can be detected with suitable methods. The addition of the orthogonal Hilbert space is not available in our model, since the unitaries have been trained on the computational space only. Since no interaction with non-computational levels has been included, they cannot bring the evaded state back to its original space. To mimic the quantum erasure channel, we replace the ideal GHZ-states with random states, that can differ at each noise realization. It is created by normalizing a vector of valid complex amplitudes. The noise parameter p represents the proportion of random states in the testing set and could stand for a decay rate in the example mentioned above. This simulation is a confirmation of the previous observations. Since the probability of two identical noisy states is low, the variance in the testing set is maximal. Even in these conditions, high fidelity in the outputs is maintained for both architectures. As long as the quantum channel has been trained successfully, the QAE can recover the desired GHZ-state in its outputs with high fidelity.

Therefore, even though the QAE cannot correct any noise channel in any state, it can generalize its denoising effect in two senses. First, it can correct bit-flips on ϕ-GHZ-states, for any value of ϕ while seeing only three of them during the training. Second, the cross-testing approach reveals that it can recover the GHZ-state from any noise channel with high fidelity when trained in the weak-to-intermediate noise regime of the bit-flip channel. While this result may hint at overfitting, in the sense that the network can only create GHZ-states starting from any quantum state, it is also attractive for some applications. First, it presents the operational advantage of being device-independent. Indeed, regardless of the nature of the noise and its intensity, the QAE can produce valid GHZ-states in its outputs as long as the quantum map can be trained. Therefore, no knowledge about noise sources on the device is required to generate high-quality GHZ-states. This can also be advantageous for devices that must produce entangled states, such as repeaters to build a quantum internet, and whose calibration is expensive but necessary: by being insensitive to noise variations in time, the QAE adapts to such devices. Finally, this generalization ability is of theoretical interest. Indeed, the training of the QAE finds a universal quantum map from any quantum state to the GHZ-state. We note that this is possible because the GHZ-state has rank one and thereby majorizes any other quantum state [123]. The elaboration of such a map is known to be complex to find.

Conclusion and outlook

In this thesis, we explore the tools of quantum machine learning in light of its application to revert the effects of noise on quantum computers. The QML technique is elaborated to improve the quantum processor that accommodates it. Starting from the basics of machine learning, we progressively move from quantum machine learning to the construction of a Quantum Autoencoder model that can learn denoising. By simulating it numerically, we investigate its performance, capacities, and limitations by borrowing methods at the intersection between computer science and quantum theory.

We have introduced two possible ways to implement QML on quantum hardware, among many other attempts. Even though both methods rely on the circuit representation of quantum computations, they understand the notion of parameters from different points of view. The Variational Quantum Algorithms tune quantum gates with trained and updated parameters in a hybrid, quantum-classical manner. Widely studied and already realizing some of its promises in some disciplines such as quantum chemistry or condensed matter physics, the success of the VQA lies in the suitable choice of an ansatz to train and parametrize. Despite potentially limiting the operations the model can express, this model can take advantage of the device's strengths to be more robust against noise and scale better. In contrast, the Dissipative Quantum Neural Networks model was recently introduced in the community and offered a fully quantum algorithm that mimics classical feed-forward neural networks. The parameters are envisioned as unitaries applied between network layers, and dissipation enhances the non-linearity achievable in the activation functions. Thanks to its structure, some areas of the knowledge developed in classical machine learning can be transferred to these networks. The direct comparison with classical networks is a requirement to investigate the question of quantum advantage in QML. Despite being potentially expensive to implement, its quantum learning rule makes it possible to reach desired quantum maps.

We employ the latter model for the practical application of denoising. We connect QML and quantum computing to design a quantum algorithm capable of mitigating the effects of noise on quantum hardware, thereby connecting knowledge of both fields. The study of the Quantum Autoencoder uncovers the challenges its implementation represents. It emphasizes the need for a quantum reformulation of functional properties of artificial neural networks and the investigation of the apparition of some well-known phenomena in quantum neural networks. Indeed, the notion of distance between two states must be redefined in the quantum setting to enable a precise formulation of the

task in the cost function. The effects of initialization are questioned, and the importance of the quality and quantity of the training data is brought to light. The data quality is shown to be a primary bottleneck in realizing a DQNN for denoising to the extent that it guides the network towards the desired state in an unsupervised setting. Finally, we open the discussion of the concept of generalization, which is essential to benchmark the performance of a QML algorithm against other quantum algorithms and classical machine learning methods.

Studying the denoising Quantum Autoencoder, these insights are gained to correct bit-flips on GHZ-states. The DQNN is given a task that can only succeed if the quantum properties of the states are understood. We show in numerical simulations that the QAE can recover the desired states with high fidelity until some noise tolerance threshold, after which fidelity drops, and the similarity with the target states is lost. Our results are consistent with the existing literature and are invariant in the random initialization of the parameters. To understand the apparition of these limitations, we use tools of quantum information and quantum theory. We compute second-order Rényi entropy to measure noise and entanglement in the outputs, and, more generally, in the network. We show that Rényi divergence is a suitable measure of the distance between the output and target states, including quantum effects such as purity loss. Moreover, Rényi entropy is crucial to understanding the network dynamics during the training. The network learns to confine entropy in the encoder to perform the denoising task, while entanglement vanishes in the decoder that delivers the outputs. In the process, the latent space appears as the key ingredient to the success of the training. The tolerance threshold arises when the intermediate layer no longer blocks noise in the first half of the network, thereby inducing a behavior comparable to a second-order phase transition in the correlations between the entropies of the latent and output states.

With these insights, we attempt to improve the noise tolerance of the QAE by modifying it with brain boxes. Introducing more complex and potentially more significant latent spaces makes the intermediate layer robust against noise. Our numerical simulations show that the tolerance upper bound imposed by the training data is saturated thanks to the addition of the brain boxes. Not only is the size of the latent space important, but its layout also plays a role in the improvement of robustness. We relate this increased tolerance to the phenomenon of overparametrization and the field of representation learning. The latent state selects features that are more favorable to the recovery of the target state. However, we question its ability to generalize its performance to new data and propose three perspectives. The literature has shown that QAE can denoise bit-flips on some novel states that are remarkably similar to the training data. We implement the cross-testing approach to probe for the robustness of the trained map to various noise mechanisms. Our numerical results reveal the denoising effect of the QAE beyond the bit-flip channel: it can denoise states affected by the depolarizing channel and an approximate quantum erasure channel. Nevertheless, the QAE fails to fulfill broader generalization criteria.

The model of the QAE has shown promising results in numerical simulations. However, its transferability to a quantum processor remains unknown. Running this algo-

rithm would require modifications in the training rule to constrain the operations according to the possibilities offered by the device. First, it requires operating multi-qubit gates on large numbers of qubits. It would be helpful to investigate how to implement this technique on devices with minor connectivity. Second, the trained and learned unitaries are complex and probably unsuitable for compiling in a finite-sized, efficient quantum circuit. Adapting the training rule to this constraint and allowing only specific terms to appear in the final unitaries could prevent overheads in resources and computational time. In addition, we have shown that the formulation of the problem in an unsupervised way led to some dependence of the training on the data. The main obstacle to improving noise tolerance is access to good data sets. Therefore, one could take advantage of the existing quantum tools in quantum information and computer science to root the expression of the task in selecting relevant information to obtain robust representations in the latent states instead of evaluating the dominant form in the data set. This reformulation of the job goes hand in hand with the discussion of generalization to provide more precise insights into the goal to reach in the task. The ideal design of the QAE would be able to cope with various noise channels affecting a large diversity of quantum states.

Bibliography

[1] G. M. D'Ariano, G. Chiribella, and P. Perinotti, *Quantum Theory from First Principles: An Informational Approach*. Cambridge, England: Cambridge University Press, 2017.

[2] J. Preskill, *"Quantum Computing in the NISQ era and beyond"* Quantum, vol. 2, p. 79, 2018.

[3] R. P. Feynman, *"Simulating physics with computers"*, Int. J. Theor. Phys., vol. 21, no. 6-7, pp. 467–488, 1982.

[4] C. E. Shannon, *"A mathematical theory of communication"*, Bell Syst. Tech. J., vol. 27, no. 3, pp. 379–423, 1948.

[5] F. Rosenblatt, *"The perceptron: a probabilistic model for information storage and organization in the brain"*, Psychological review, vol. 65, no. 6, pp. 386–408, 1958.

[6] I. L. Chuang, N. Gershenfeld, and M. Kubinec, *"Experimental implementation of fast quantum searching"*, Phys. Rev. Lett., vol. 80, no. 15, pp. 3408–3411, 1998.

[7] J. Ku, X. Xu, M. Brink, D. C. McKay, J. B. Hertzberg, M. H. Ansari, and B. L. T. Plourde, *"Suppression of unwanted zz interactions in a hybrid two-qubit system"*, Physical review letters, vol. 125, no. 20, p. 200504, 2020.

[8] P. W. Shor, *"Fault-tolerant quantum computation"*, in Proceedings of 37th Conference on Foundations of Computer Science, IEEE Comput. Soc. Press, 2002.

[9] A. L. Samuel, *"Some studies in machine learning using the game of checkers"*, IBM journal of research and development, vol. 3, no. 3, pp. 210–229, 1959.

[10] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, *"Gradient-based learning applied to document recognition"*, Proc. IEEE Inst. Electr. Electron. Eng., vol. 86, no. 11, pp. 2278–2324, 1998.

[11] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, *"Generative adversarial networks"*, in Advances in Neural Information Processing Systems, vol. 27, 2014.

[12] D. Flam-Shepherd, T. Wu, X. Gu, A. Cervera-Lierta, M. Krenn, and A. Aspuru-Guzik, *"Learning interpretable representations of entanglement in quantum optics experiments using deep generative models."*, arXiv 2109.02490, 2021.

[13] M. C. Caro, H.-Y. Huang, M. Cerezo, K. Sharma, A. Sornborger, L. Cincio, and P. J. Coles, *"Generalization in quantum machine learning from few training data."*, arXiv 2111.05292, 2021.

[14] M. Schuld and F. Petruccione, *Machine learning with quantum computers*. Springer Nature, 2 ed., 2021.

[15] K. Hornik, M. Stinchcombe, and H. White, *"Multilayer feedforward networks are universal approximators"*, Neural Netw., vol. 2, no. 5, pp. 359–366, 1989.

[16] M. Nielsen, *Neural Networks and Deep Learning*. Determination Press, 2015.

[17] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. London, England: MIT Press, 2016.

[18] S. Ruder, *"An overview of gradient descent optimization algorithms."*, arXiv 1609.04747, 2016.

[19] T. Dozat, *"Incorporating Nesterov Momentum into ADAM."*, ICLRWorkshop, 2016.

[20] B. T. Polyak, *"Some methods of speeding up the convergence of iteration methods"*, U.S.S.R. comput. math. math. phys., vol. 4, no. 5, pp. 1–17, 1964.

[21] D. P. Kingma and J. Ba, *"Adam: A method for stochastic optimization"*, International Conference on Learning Representations, 2015.

[22] Y. Nesterov, *"A method of solving a convex programming problem with convergence rate o(1/k2)"*, Proceedings of the USSR Academy of Sciences, vol. 269, pp. 543–547, 1983.

[23] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, *"Learning representations by back-propagating errors"*, Nature, vol. 323, no. 6088, pp. 533–536, 1986.

[24] C. M. Bishop, *Pattern Recognition and Machine learning*. New York, NY: Springer, 2006.

[25] X.-M. Zhang, Z. Wei, R.Asad, X.-C Yang, X. Wang, *When does reinforcement learning stand out in quantum control? A comparative study on state preparation"*, Npj Quantum Inf., vol. 5, no. 1, 2019.

[26] F. Marquardt, *Machine learning and quantum devices"*, SciPost Phys. Lect. Notes, no. 29, 2021.

[27] M. Schuld, I. Sinayskiy, and F. Petruccione, *"An introduction to quantum machine learning"*, Contemp. Phys., vol. 56, no. 2, pp. 172–185, 2015.

[28] P. W. Shor, *"Algorithms for quantum computation: discrete logarithms and factoring"*, in Proceedings 35th Annual Symposium on Foundations of Computer Science, IEEE Comput. Soc. Press, 2002.

[29] V. Lopez-Pastor and F. Marquardt, *"Self-learning machines based on hamiltonian echo backpropagation."*, arXiv 2103.04992, 2021.

[30] P. Rebentrost, M. Mohseni, and S. Lloyd, *"Quantum support vector machine for big data classification"*, Physical review letters, vol. 113, no. 13, p. 130503, 2014.

[31] S. Lloyd, M. Mohseni, and P. Rebentrost, *"Quantum principal component analysis"*, Nature physics, vol. 10, no. 9, pp. 631–633, 2014.

[32] E. Farhi and H. Neven, *"Classification with quantum neural networks on near term processors."*, arXiv 1802.06002, 2018.

[33] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, and P. J. Coles, *"Variational quantum algorithms"*, Nature Reviews Physics, vol. 3, pp. 625–644, 2021.

[34] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'Brien, *"A variational eigenvalue solver on a quantum processor"*, Nature, vol. Communications, pp. 5:4213,, 2014.

[35] P. Döring, *"Aspects in quantum machine learning for chemistry and neural networks"*, Master's thesis, RWTH Aachen, 2020.

[36] E. Farhi, J. Goldstone, and S. Gutmann, *"A quantum approximate optimization algorithm."*, arXiv 1411.4028, 2014.

[37] Z. Holmes, K. Sharma, M. Cerezo, and P. J. Coles, *"Connecting ansatz expressibility to gradient magnitudes and barren plateaus"*, PRX Quantum, vol. 3, no. 1, 2022.

[38] M. Larocca, P. Czarnik, K. Sharma, G. Muraleedharan, P. J. Coles, and M. Cerezo, *"Diagnosing barren plateaus with tools from quantum optimal control."*, arXiv 2105.14377, 2021.

[39] K. Beer, D. Bondarenko, T. Farrelly, T. J. Osborne, R. Salzmann, D. Scheiermann, and R. Wolf, *"Training deep quantum neural networks"*, Nat. Commun., vol. 11, no. 1, 2020.

[40] K. Sharma, M. Cerezo, L. Cincio, and P. J. Coles, *"Trainability of dissipative perceptron-based quantum neural networks."*, arXiv 2005.12458, 2020.

[41] M. Cerezo, A. Sone, T. Volkoff, L. Cincio, and P. J. Coles, *"Cost function dependent barren plateaus in shallow parametrized quantum circuits"*, Nature communications, vol. 12, no. 1, p. 1791, 2021.

[42] D. Bondarenko and P. Feldmann, *"Quantum autoencoders to denoise quantum data"*, Phys. Rev. Lett., vol. 124, no. 13, p. 130502, 2020.

[43] S. Lloyd, *"Quantum machine learning."*, Lecture at Keio University, June 2016.

[44] T. F. Rønnow, Z. Wang, J. Job, S. Boixo, S. V. Isakov, D. Wecker, J. M. Martinis, D. A. Lidar, and M. Troyer, *"Defining and detecting quantum speedup"*, Science, vol. 345, no. 6195, pp. 420–424, 2014.

[45] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, *"Quantum machine learning"*, Nature, vol. 549, no. 7671, pp. 195–202, 2017.

[46] A. W. Harrow, A. Hassidim, and S. Lloyd, *"Quantum algorithm for linear systems of equations"*, Physical Review Letters, vol. 103, no. 15, p. 150502, 2009.

[47] K. Gili, M. Mauri, and A. Perdomo-Ortiz, *"Evaluating generalization in classical and quantum generative models."*, arXiv 2201.08770, 2022.

[48] P. Bashivan, R. Bayat, A. Ibrahim, K. Ahuja, M. Faramarzi, T. Laleh, B. A. Richards, and I. Rish, *"Adversarial feature desensitization."*, arXiv 2006.04621, 2020.

[49] S. Lu, L.-M. Duan, and D.-L. Deng, *"Quantum adversarial machine learning"*, Physical Review Research, vol. 2, no. 3, 2020.

[50] A. Kurakin, I. Goodfellow, and S. Bengio, *"Adversarial machine learning at scale."*, arXiv 1611.01236, 2016.

[51] M. Weber, N. Liu, B. Li, C. Zhang, and Z. Zhao, *"Optimal provable robustness of quantum classification via quantum hypothesis testing"*, npj Quantum Information, vol. 7, no. 1, 2021.

[52] M. Schuld, R. Sweke, and J. J. Meyer, *"Effect of data encoding on the expressive power of variational quantum-machine-learning models"*, Physical Review. A, vol. 103, no. 3, 2021.

[53] S. Aaronson, *"Read the fine print"*, Nature physics, vol. 11, no. 4, pp. 291–293, 2015.

[54] X. Lu, S. Matsuda, C. Hori, and H. Kashioka, *"Speech restoration based on deep learning autoencoder with layer-wised pretraining"*, in INTERSPEECH, 2012.

[55] M. Kim, *"Collaborative deep learning for speech enhancement: A run-time model selection method using autoencoders"*, in 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2017.

[56] R. Sinha and P. Rajan, *"A deep autoencoder approach to bird call enhancement"*, in 2018 IEEE 13th International Conference on Industrial and Information Systems (ICIIS), pp. 22–26, 2018.

[57] L. Gondara, *"Medical image denoising using convolutional denoising autoencoders"*, in 2016 IEEE 16th international conference on data mining workshops (ICDMW), pp. 241–246, IEEE, 2016.

[58] Y. Sakai, Y. Itoh, P. Jung, K. Kokeyama, C. Kozakai, K. T. Nakahira, S. Oshino, Y. Shikano, H. Takahashi, T. Uchiyama, G. Ueshima, T. Washimi, T. Yamamoto, and T. Yokozawa, *"Unsupervised learning architecture for classifying the transient noise of interferometric gravitational-wave detectors."*, arXiv 2111.10053, 2021.

[59] K. Hartnett, *"Quantum supremacy is coming: Here's what you should know."*, www.quantamagazine.org, July 2019.

[60] A. Peres, *"Reversible logic and quantum computers"*, Physical Review A: General physics, vol. 32, no. 6, pp. 3266–3276, 1985.

[61] P. W. Shor, *"Scheme for reducing decoherence in quantum computer memory"*, Physical Review. A, vol. 52, no. 4, pp. R2493–R2496, 1995.

[62] A. Steane, *"Multiple-particle interference and quantum error correction"*, Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences, vol. 452, no. 1954, pp. 2551–2577, 1996.

[63] D. Gottesman, *Stabilizer Codes and Quantum Error Correction*, PhD thesis, California Institute of Technology, 1997.

[64] A. Kitaev, *"Fault-tolerant quantum computation by anyons"*, Annals of Physics, vol. 303, no. 1, pp. 2–30, 2003.

[65] C. Ryan-Anderson, J. G. Bohnet, K. Lee, D. Gresh, A. Hankin, J. P. Gaebler, D. Francois, A. Chernoguzov, D. Lucchetti, N. C. Brown, T. M. Gatterman, S. K. Halit, K. Gilmore, J. Gerber, B. Neyenhuis, D. Hayes, and R. P. Stutz, *"Realization of real-time fault-tolerant quantum error correction"*, Physical Review. X, vol. 11, no. 4, 2021.

[66] W. K. Wootters and W. H. Zurek, *"A single quantum cannot be cloned"*, Nature, vol. 299, no. 5886, pp. 802–803, 1982.

[67] A. Ekert and C. Macchiavello, *"Quantum error correction for communication"*, Physical Review Letters, vol. 77, no. 12, pp. 2585–2588, 1996.

[68] R. Uzdin, *"Methods for measuring noise, purity changes, and entanglement entropy in quantum devices and systems."*, arXiv 2112.00546, 2021.

[69] T. Lubinski, S. Johri, P. Varosy, J. Coleman, L. Zhao, J. Necaise, C. H. Baldwin, K. Mayer, and T. Proctor, *"Application-oriented performance benchmarks for quantum computing."*, arXiv 2110.03137, 2021.

[70] D. Gottesman, *"The Heisenberg representation of quantum computers."*, arXiv 9807006, 1998.

[71] E. Magesan, J. M. Gambetta, and J. Emerson, *"Robust randomized benchmarking of quantum processes."*, arXiv 1009.3639, 2018.

[72] *"Qiskit textbook: Randomized benchmarking."* https://qiskit.org/textbook/ch-quantum-hardware/randomized-benchmarking.html, April 2022.

[73] P. Krantz, M. Kjaergaard, F. Yan, T. P. Orlando, S. Gustavsson, and W. D. Oliver, *"A quantum engineer's guide to superconducting qubits"*, Applied Physics Reviews, vol. 6, no. 2, p. 021318, 2019.

[74] A. W. Cross, L. S. Bishop, S. Sheldon, P. D. Nation, and J. M. Gambetta, *"Validating quantum computers using randomized model circuits"*, Physical Review. A, vol. 100, no. 3, 2019.

[75] J.-S. Kim, L. S. Bishop, A. D. Corcoles, S. Merkel, J. A. Smolin, and S. Sheldon, *"Hardware-efficient random circuits to classify noise in a multi-qubit system"*, Phys. Rev. A, vol. 104, p. 022609, 2021.

[76] L. Cincio, Y. Subaşı, A. T. Sornborger, and P. J. Coles, *"Learning the quantum algorithm for state overlap"*, New J. Phys., vol. 20, p. 113022, 2018.

[77] L. Cincio, K. Rudinger, M. Sarovar, and P. J. Coles, *"Machine learning of noise-resilient quantum circuits"*, PRX Quantum, vol. 2, no. 1, 2021.

[78] K. Temme, S. Bravyi, and J. M. Gambetta, *"Error mitigation for short-depth quantum circuits"*, Physical Review Letters, vol. 119, no. 18, 2017.

[79] Y. Kim, C. J. Wood, T. J. Yoder, S. T. Merkel, J. M. Gambetta, K. Temme, and A. Kandala, *"Scalable error mitigation for noisy quantum circuits produces competitive expectation values."*, arXiv 2108.09197, 2021.

[80] P. Czarnik, A. Arrasmith, P. J. Coles, and L. Cincio, *"Error mitigation with clifford quantum-circuit data"*, Quantum, vol. 5, no. 592, p. 592, 2021.

[81] A. Lowe, M. H. Gordon, P. Czarnik, A. Arrasmith, P. J. Coles, and L. Cincio, *"Unified approach to data-driven quantum error mitigation"*, Physical Review Research, vol. 3, no. 3, 2021.

[82] Y. Lecun, *Modeles connexionnistes de l'apprentissage (connectionist learning models)*, PhD thesis, Universite Pierre et Marie Curie (Paris VI), June 1987.

[83] H. Bourlard and Y. Kamp, *"Auto-association by multilayer perceptrons and singular value decomposition"*, Biological cybernetics, vol. 59, no. 4–5, pp. 291–294, 1988.

[84] G. E. Hinton and R. S. Zemel, *"Autoencoders, minimum description length and Helmholtz free energy"*, in Proceedings of the 6th International Conference on Neural Information Processing Systems, NIPS'93, (San Francisco, CA, USA), pp. 3–10, Morgan Kaufmann Publishers Inc., 1993.

[85] J. Gehring, Y. Miao, F. Metze, and A. Waibel, *"Extracting deep bottleneck features using stacked auto-encoders"*, in 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, IEEE, 2013.

[86] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, *"Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion"*, Journal of Machine Learning Research, vol. 11, no. 110, pp. 3371–3408, 2010.

[87] P. Baldi and K. Hornik, *"Neural networks and principal component analysis: Learning from examples without local minima"*, Neural networks: the official journal of the International Neural Network Society, vol. 2, no. 1, pp. 53–58, 1989.

[88] R. Salakhutdinov and G. Hinton, *"Semantic hashing"*, International journal of approximate reasoning: official publication of the North American Fuzzy Information Processing Society, vol. 50, no. 7, pp. 969–978, 2009.

[89] G. E. Hinton and R. R. Salakhutdinov, *"Reducing the dimensionality of data with neural networks"*, Science (New York, N.Y.), vol. 313, no. 5786, pp. 504–507, 2006.

[90] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, *"Extracting and composing robust features with denoising autoencoders"*, in Proceedings of the 25th international conference on Machine learning - ICML '08, (New York, New York, USA), ACM Press, 2008.

[91] K. H. Wan, O. Dahlsten, H. Kristjánsson, R. Gardner, and M. S. Kim, *"Quantum generalisation of feedforward neural networks"*, npj Quantum Information, vol. 3, no. 1, 2017.

[92] C. Cao and X. Wang, *"Noise-assisted quantum autoencoder"*, Phys. Rev. Applied, vol. 15, no. 5, p. 054012, 2020.

[93] J. Romero, J. P. Olson, and A. Aspuru-Guzik, *"Quantum autoencoders for efficient compression of quantum data"*, Quantum Sci. Technol., vol. 2, no. 4, p. 045001, 2017.

[94] C. Bravo-Prieto, *"Quantum autoencoders with enhanced data encoding"*, Machine Learning: Science and Technology, vol. 2, no. 3, p. 035028, 2021.

[95] Y. Du and D. Tao, *"On exploring practical potentials of quantum auto-encoder with advantages."*, arXiv 2106.15432, 2021.

[96] D. M. Greenberger, M. A. Horne, and A. Zeilinger, *Going beyond bell's theorem*, pp. 69–72. Springer Netherlands, 1989.

[97] N. D. Mermin, *"Extreme quantum entanglement in a superposition of macroscopically distinct states"*, Physical Review Letters, vol. 65, pp. 1838–1840, October 1990.

[98] N. D. Mermin, *"Quantum mysteries revisited"*, American journal of physics, vol. 58, no. 8, pp. 731–734, 1990.

[99] D. Alsina and J. I. Latorre, *"Experimental test of Mermin inequalities on a five-qubit quantum computer"*, Physical Review. A, vol. 94, no. 1, 2016.

[100] W.-J. Huang, W.-C. Chien, C.-H. Cho, C.-C. Huang, T.-W. Huang, and C.-R. Chang, *"Mermin's inequalities of multiple qubits with orthogonal measurements on ibm q 53-qubit system"*, Quantum Engineering, vol. 2, no. 2, p. e45, 2020.

[101] C. H. Bennett, G. Brassard, C. Crépeau, R. Jozsa, A. Peres, and W. K. Wootters, *"Teleporting an unknown quantum state via dual classical and einstein-podolsky-rosen channels"*, Physical Review Letters, vol. 70, no. 13, p. 1895, 1993.

[102] R. Raussendorf, D. E. Browne, and H. J. Briegel, *"Measurement-based quantum computation on cluster states"*, Physical Review. A, vol. 68, no. 2, 2003.

[103] L. Pezzè, A. Smerzi, M. K. Oberthaler, R. Schmied, and P. Treutlein, *"Quantum metrology with nonclassical states of atomic ensembles"*, Reviews of Modern Physics, vol. 90, sep 2018.

[104] W. Dür, G. Vidal, and J. I. Cirac, *"Three qubits can be entangled in two inequivalent ways"*, Phys. Rev. A, vol. 62, p. 062314, 2000.

[105] T. Achache, L. Horesh, and J. Smolin, *"Denoising quantum states with quantum autoencoders – theory and applications"*, arXiv 2012.14714, 2020.

[106] C. H. Bennett, D. P. DiVincenzo, and J. A. Smolin, *"Capacities of quantum erasure channels"*, Phys. Rev. Lett., vol. 78, no. 16, pp. 3217–3220, 1997.

[107] M. H. Ansari, A. van Steensel, and Y. V. Nazarov, *"Entropy production in quantum is different"*, Entropy, vol. 21, p. 854, aug 2019.

[108] M. H. Ansari and Y. V. Nazarov, *"Rényi entropy flows from quantum heat engines"*, Phys. Rev. B, vol. 91, p. 104303, Mar 2015.

[109] M. H. Ansari and Y. V. Nazarov, *"Exact correspondence between Renyi entropy flows and physical flows"*, Phys. Rev. B, vol. 91, p. 174307, May 2015.

[110] T. Brydges, A. Elben, P. Jurcevic, B. Vermersch, C. Maier, B. P. Lanyon, P. Zoller, R. Blatt, and C. F. Roos, *"Probing Rényi entanglement entropy via randomized measurements"*, Science, vol. 364, no. 6437, pp. 260–263, 2019.

[111] A. Gilchrist, N. K. Langford, and M. A. Nielsen, *"Distance measures to compare real and ideal quantum processes"*, Physical Review. A, vol. 71, no. 6, 2005.

[112] R. L. Frank and E. H. Lieb, *"Monotonicity of a relative Rényi entropy"*, Journal of mathematical physics, vol. 54, no. 12, p. 122201, 2013.

[113] J. Rogers, G. Bhattacharyya, M. S. Frank, T. Jiang, O. Christiansen, Y.-X. Yao, and N. Lanatà, *"Error mitigation in variational quantum eigensolvers using probabilistic machine learning."*, arXiv 2111.08814, 2021.

[114] J. W. Rocks and P. Mehta, *"Bias-variance decomposition of overparameterized regression with random linear features."*, arXiv 2203.05443, 2022.

[115] M. Belkin, D. Hsu, S. Ma, and S. Mandal, *"Reconciling modern machine learning practice and the bias-variance trade-off."*, arXiv 1812.11118, 2018.

[116] M. Larocca, N. Ju, D. García-Martín, P. J. Coles, and M. Cerezo, *"Theory of overparametrization in quantum neural networks"*, arXiv 2109.11676, 2021.

[117] M. Y. Niu, A. Zlokapa, M. Broughton, S. Boixo, M. Mohseni, V. Smelyanskyi, and H. Neven, *"Entangling quantum generative adversarial networks"*, arXiv 2105.00080, 2021.

[118] S. Lloyd and C. Weedbrook, *"Quantum generative adversarial learning"*, Physical Review Letters, vol. 121, no. 4, 2018.

[119] P.-L. Dallaire-Demers and N. Killoran, *"Quantum generative adversarial networks"*, Physical review. A, vol. 98, no. 1, 2018.

[120] K. Beer and G. Müller, *"Dissipative quantum generative adversarial networks"*, arXiv 2112.06088, 12 2021.

[121] N. Tishby, F. C. Pereira, and W. Bialek, *"The information bottleneck method."*, arXiv 0004057, 2000.

[122] M. Grassl, T. Beth, and T. Pellizzari, *"Codes for the quantum erasure channel"*, Physical Review. A, vol. 56, no. 1, pp. 33–38, 1997.

[123] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10Th Anniversary Edition*. Cambridge, England: Cambridge University Press, 2012.

Appendix A

Adjoint channel

In section 1.2.2.2, we explained that the training rule relies on measuring the difference between an input state that has been propagated forwards through the network until some layer l of interest, and the corresponding desired output state propagated backwards until the same layer. Therefore, we derive a formula for the backward propagation in terms of the quantum map of the forward pass. This derivation is taken from [39] and is provided for completeness. The map $\mathcal{F}^l(\rho_x^l)$ is the adjoint of $\mathcal{E}^l(\rho_x^{l-1})$. A completely positive quantum map can be written in the Kraus formalism [123] with:

$$\begin{aligned} \mathcal{E}^l(\rho_x^{l-1}) &= \textstyle\sum_\alpha A_\alpha^l \rho_x^{l-1} A_\alpha^{l\,\dagger} \\ \text{where } \textstyle\sum_\alpha A_\alpha^{l\,\dagger} A_\alpha^l &= \mathbb{I}. \end{aligned} \tag{A.1}$$

Taking the adjoint channel, a general form for $\mathcal{F}^l(\rho_x^l)$ is then simply:

$$\mathcal{F}^l(\rho_x^l) = \sum_\alpha A_\alpha^{l\,\dagger} \rho_x^l A_\alpha^l. \tag{A.2}$$

To obtain an explicit form of the channel, we must precise the Kraus operators. For this purpose, let $\{|m\rangle\}, \{|n\rangle\}$ be two sets of orthogonal states for layer $l-1$, and $\{|i\rangle\}, \{|j\rangle\}$ for layer l. In addition, $\{|\alpha\rangle\}$ is an orthonormal basis for the Hilbert space of layer $l-1$. With these bases, we can find each element of the Kraus operators:

$$\begin{aligned} \langle i|\mathcal{E}^l(\rho_x^{l-1})|j\rangle &= \langle i| \operatorname*{Tr}_{l-1} \left\{ \mathcal{U}^l \left(\rho^{l-1} \otimes |0\rangle\langle 0|^{\otimes N_l} \right) \mathcal{U}^{l\dagger} \right\} |j\rangle \\ &= \sum_\alpha \langle \alpha, i| \mathcal{U}^l \left(\rho^{l-1} \otimes |0\rangle\langle 0|^{\otimes N_l} \right) \mathcal{U}^{l\dagger} |\alpha, j\rangle \end{aligned} \tag{A.3}$$

Looking at the results of the channel for a component of the density matrix on layer l, $|m\rangle\langle n|$:

$$\begin{aligned}\langle i|\mathcal{E}^l(|m\rangle\langle n|)|j\rangle &= \sum_\alpha \langle \alpha, i|\,\mathcal{U}^l\left(|m\rangle\langle n|\otimes|0\rangle\langle 0|^{\otimes N_l}\right)\mathcal{U}^{l\dagger}\,|\alpha,j\rangle \\ &= \sum_\alpha \langle \alpha, i|\,\mathcal{U}^l\left(|m,0\cdots 0\rangle\langle n,0\cdots 0|\right)\mathcal{U}^{l\dagger}\,|\alpha,j\rangle \\ &= \sum_\alpha \Big(\langle \alpha, i|\,\mathcal{U}^l|m,0\cdots 0\rangle\Big)\ \Big(\langle n,0\cdots 0|\mathcal{U}^{l\dagger}\,|\alpha,j\rangle\Big) \end{aligned} \tag{A.4}$$

Finally, the Kraus operators are defined by their matrix components:

$$\langle i|A^l_\alpha|m\rangle = \langle \alpha, i|\,\mathcal{U}^l|m,0\cdots 0\rangle \tag{A.5}$$

This equation enables us to find the explicit formula for the adjoint channel by using the Kraus operators above, and applying them on a component $|i\rangle\langle j|$ of the density matrix on layer l:

$$\begin{aligned}\langle m|\mathcal{F}^l(|i\rangle\langle j|)|n\rangle &= \langle m|\sum_\alpha {A^l_\alpha}^\dagger|i\rangle\langle j|A^l_\alpha|n\rangle \\ &= \sum_\alpha \langle m,0\cdots 0|\mathcal{U}^{l\dagger}\,|\alpha,i\rangle\ \langle \alpha,j|\,\mathcal{U}^l|n,0\cdots 0\rangle \\ &= \langle m,0\cdots 0|\mathcal{U}^{l\dagger}\left(\mathbb{I}^{l-1}\otimes|i\rangle\langle j|\right)\mathcal{U}^l|n,0\cdots 0\rangle \\ &= \langle m|\operatorname*{Tr}_l\left\{\left(\mathbb{I}^{l-1}\otimes|0\rangle\langle 0|^{\otimes N_l}\right)\mathcal{U}^{l\dagger}\ \left(\mathbb{I}^{l-1}\otimes|i\rangle\langle j|\right)\ \mathcal{U}^l\right\}|n\rangle. \end{aligned} \tag{A.6}$$

Finally, a general state on layer l is propagated backwards to layer $l-1$ by the application of the adjoint channel:

$$\mathcal{F}^l(\rho^l_x) = \operatorname*{Tr}_l\left\{\left(\mathbb{I}^{l-1}\otimes|0\rangle\langle 0|^{\otimes N_l}\right)\mathcal{U}^{l\dagger}\ \left(\mathbb{I}^{l-1}\otimes\rho^l_x\right)\ \mathcal{U}^l\right\} \tag{A.7}$$

Appendix B

Derivation of the learning rule for DQNN

For completeness and to show more explicitly that the update rule for DQNN is derived completely from the cost function in a way that is compatible with quantum mechanics, we report here its derivation, as shown by [39]. It can also be useful if one aims at using a distance measure that is different from the fidelity: the Rényi divergence introduced in 2.3.3.2 could be a more complete cost function. It would, however, require additional terms, such as a regularizer, to introduce a bias towards the solution due to its symmetry in the probabilities.

To derive an explicit formula for the gradient of the cost function dC/dt where C is the cost function and t is a training round, we impose the constraint:

$$U_j^l(t+\varepsilon) = e^{i\varepsilon K_j^l(t)} U_j^l(t) \tag{B.1}$$

to define the form of the updates in a quantum mechanical way. The matrix $K_j^l(t)$ is the parameter matrix. To make the training efficient, $K_j^l(t)$ must be the fastest update possible.

We take the cost function to be the fidelity of the output state ρ_x^{out} with a pure target state $\rho_x^* = |\phi^*\rangle\langle\phi^*|$:

$$C = \frac{1}{N}\sum_x \text{Tr}\left\{\rho_x^* \rho_x^{\text{out}}\right\} = \frac{1}{N}\sum_x \langle\phi^*|\rho_x^{\text{out}}|\phi^*\rangle. \tag{B.2}$$

Since it is a smooth function of the variable t, its derivative can be written as:

$$\frac{dC}{dt} = \lim_{\varepsilon\to 0} \frac{C(t+\varepsilon) - C(t)}{\varepsilon}, \tag{B.3}$$

for an infinitesimal update.

By equations B.1 and B.2, the first term of the difference can be calculated. In particular, it requires the state at step $t+\varepsilon$:

$$\rho_x^{\text{out}}(t+\varepsilon) = \underset{\text{in, hid}}{\text{Tr}} \left\{ \left(\prod_{l=L}^{1} \prod_{j=N_L}^{1} e^{i\varepsilon K_j^l(t)} U_j^l(t) \right) \left(\rho_x^{\text{in}} \otimes |0\rangle\langle 0|^{\otimes \text{hid, out}} \right) \left(\prod_{l=1}^{L} \prod_{j=1}^{N_L} U_j^l(t)^\dagger e^{-i\varepsilon K_j^l(t)} \right) \right\}. \tag{B.4}$$

We perform the Taylor expansion to first order in ε of the update $e^{i\varepsilon K_j^l(t)} \approx 1 + i\varepsilon K_j^l(t) + \mathcal{O}(\varepsilon^2)$:

$$\begin{aligned}
&\rho_x^{\text{out}}(t+\varepsilon) \\
&= \underset{\text{in, hid}}{\text{Tr}} \left\{ \left(\prod_{l=L}^{1} \prod_{j=N_L}^{1} (1 + i\varepsilon K_j^l(t)) U_j^l(t) \right) \left(\rho_x^{\text{in}} \otimes |0\rangle\langle 0|^{\otimes \text{hid, out}} \right) \left(\prod_{l=1}^{L} \prod_{j=1}^{N_L} U_j^l(t)^\dagger (1 - i\varepsilon K_j^l(t)) \right) \right\} + \mathcal{O}(\varepsilon^2) \\
&= \underset{\text{in, hid}}{\text{Tr}} \left\{ \prod_{l=L}^{1} \prod_{j=N_L}^{1} U_j^l(t) \left(\rho_x^{\text{in}} \otimes |0\rangle\langle 0|^{\otimes \text{hid, out}} \right) \prod_{l=1}^{L} \prod_{j=1}^{N_L} U_j^l(t)^\dagger \right\} \\
&+ i\varepsilon \underset{\text{in, hid}}{\text{Tr}} \left\{ K_{N_L}^L U_{N_L}^L (U_{N_L-1}^L \cdots U_1^1) \left(\rho_x^{\text{in}} \otimes |0\rangle\langle 0|^{\otimes \text{hid, out}} \right) (U_1^{1\dagger} \cdots U_{N_L}^{L\ \dagger}) \right\} \\
&- i\varepsilon \underset{\text{in, hid}}{\text{Tr}} \left\{ (U_{N_L}^L \cdots U_1^1) \left(\rho_x^{\text{in}} \otimes |0\rangle\langle 0|^{\otimes \text{hid, out}} \right) (U_1^{1\dagger} \cdots U_{N_L-1}^{L\ \dagger}) U_{N_L}^{L\ \dagger} K_{N_L}^L \right\} \\
&+ \cdots - \cdots \\
&+ i\varepsilon \underset{\text{in, hid}}{\text{Tr}} \left\{ (U_{N_L}^L \cdots K_j^l U_j^l \cdots U_1^1) \left(\rho_x^{\text{in}} \otimes |0\rangle\langle 0|^{\otimes \text{hid, out}} \right) (U_1^{1\dagger} \cdots U_{N_L}^{L\ \dagger}) \right\} \\
&- i\varepsilon \underset{\text{in, hid}}{\text{Tr}} \left\{ (U_{N_L}^L \cdots U_1^1) \left(\rho_x^{\text{in}} \otimes |0\rangle\langle 0|^{\otimes \text{in, hid}} \right) (U_1^{1\dagger} \cdots U_j^{l\dagger} K_j^l \cdots U_{N_L}^{L\ \dagger}) \right\} \\
&+ \cdots - \cdots + \mathcal{O}(\varepsilon^2)
\end{aligned}$$

$$\begin{aligned}
&= \rho_x^{\text{out}}(t) + i\varepsilon \underset{\text{in, hid}}{\text{Tr}} \left\{ \left[K_{N_L}^L, \left(\prod_{l=L}^{1} \prod_{j=N_L}^{1} U_j^l(t) \right) \left(\rho_x^{\text{in}} \otimes |0\rangle\langle 0|^{\otimes \text{ hid, out}} \right) \left(\prod_{l=1}^{L} \prod_{j=1}^{N_L} U_j^l(t)^\dagger \right) \right] \right\} \\
&+ \cdots \\
&+ i\varepsilon \underset{\text{in, hid}}{\text{Tr}} \left\{ \left(\prod_{l=L}^{k} \prod_{j=N_l}^{\overleftarrow{q_l}} U_j^l(t) \right) \left[K_q^k, \left(\prod_{l=k}^{1} \prod_{j=\overrightarrow{q_l}}^{1} U_j^l(t) \right) \left(\rho_x^{\text{in}} \otimes |0\rangle\langle 0|^{\otimes \text{hid, out}} \right) \left(\prod_{l=1}^{k} \prod_{j=1}^{\overrightarrow{q_l}} U_j^l(t)^\dagger \right) \right] \left(\prod_{l=k}^{L} \prod_{j=\overleftarrow{q_l}}^{N_l} U_j^l(t)^\dagger \right) \right\} \\
&+ \cdots \\
&+ \mathcal{O}(\varepsilon^2)
\end{aligned}$$

$$= \rho_x^{\text{out}}(t) + i\varepsilon \Delta \rho_x^{\text{out}}(t) \tag{B.5}$$

where we simplify the notation with $\overleftarrow{q_l} = 1$ when $l \neq k$ and $\overleftarrow{q_l} = q + 1$ when $l = k$ and similarly, $\overrightarrow{q_l} = N_l$ when $l \neq k$ and $\overrightarrow{q_l} = q$ when $l = k$. We can now turn back to the cost function, and use equation B.5 in the first term of the difference.

$$
\begin{aligned}
\frac{dC}{dt} &= \lim_{\varepsilon\to 0}\frac{C(t+\varepsilon)-C(t)}{\varepsilon} \\
&= \lim_{\varepsilon\to 0}\frac{1}{N\varepsilon}\sum_x(\langle\phi_x^*|\rho_x^{\text{out}}(t+\varepsilon)|\phi_x^*\rangle - \langle\phi_x^*|\rho_x^{\text{out}}(t)|\phi_x^*\rangle) \\
&= \lim_{\varepsilon\to 0}\frac{i\varepsilon}{N\varepsilon}\sum_x\langle\phi_x^*|\Delta\rho_x^{\text{out}}(t)|\phi_x^*\rangle \\
&= \frac{i}{N}\sum_x \operatorname*{Tr}_{\text{out}}\left\{|\phi_x^*\rangle\langle\phi_x^*|\Delta\rho_x^{\text{out}}(t)\right\} \\
&= \frac{i}{N}\sum_x \operatorname*{Tr}_{\text{out}}\{|\phi_x^*\rangle\langle\phi_x^*|\sum_{k=1}^{L}\sum_{q=1}^{N_l}\operatorname*{Tr}_{\text{in, hid}}\{\left(\prod_{l=L}^{k}\prod_{j=N_l}^{\overleftarrow{ql}}U_j^l(t)\right) \\
&\left[K_q^k,\left(\prod_{l=k}^{1}\prod_{j=\overrightarrow{ql}}^{1}U_j^l(t)\right)\left(\rho_x^{\text{in}}\otimes|0\rangle\langle 0|^{\otimes\text{hid, out}}\right)\left(\prod_{l=1}^{k}\prod_{j=1}^{\overrightarrow{ql}}U_j^l(t)^\dagger\right)\right]\left(\prod_{l=k}^{L}\prod_{j=\overleftarrow{q_l}}^{N_l}U_j^l(t)^\dagger\right)\}\} \\
&= \frac{1}{N}\operatorname*{Tr}_{\text{all}}\{\sum_x\sum_k\sum_q\left(\mathbb{I}^{\text{in, hid}}\otimes|\phi_x^*\rangle\langle\phi_x^*|\right)\left(\prod_{l=L}^{k}\prod_{j=N_l}^{\overleftarrow{ql}}U_j^l(t)\right) \\
&\left[iK_q^k,\left(\prod_{l=k}^{1}\prod_{j=\overrightarrow{ql}}^{1}U_j^l(t)\right)\left(\rho_x^{\text{in}}\otimes|0\rangle\langle 0|^{\otimes\text{hid, out}}\right)\left(\prod_{l=1}^{k}\prod_{j=1}^{\overrightarrow{ql}}U_j^l(t)^\dagger\right)\right]\left(\prod_{l=k}^{L}\prod_{j=\overleftarrow{q_l}}^{N_l}U_j^l(t)^\dagger\right)\} \quad \text{(B.6)}
\end{aligned}
$$

Two simple identities can help rewriting equation B.6. They result from the invariance of the trace under cyclic permutations, for three square matrices A,B,C, and a unitary matrix U with the same dimensions:

$$\operatorname{Tr}\{A[B,C]\} = \operatorname{Tr}\{[C,A]B\} = \operatorname{Tr}\{[A,B]C\} \quad \text{(B.7)}$$

$$\operatorname{Tr}\{A\,(U[B,C]U^\dagger)\} = \operatorname{Tr}\{(U^\dagger AU)[B,C]\} = \operatorname{Tr}\{[C,(U^\dagger AU)]B\}. \quad \text{(B.8)}$$

Applying B.8 to the cost derivative, we can isolate the parameter matrix K_j^l out of the commutator:

$$
\begin{aligned}
\frac{dC}{dt} = \frac{1}{N}\operatorname*{Tr}_{\text{all}}\{\sum_x\sum_k\sum_q[&\left(\prod_{l=k}^{1}\prod_{j=\overrightarrow{ql}}^{1}U_j^l(t)\right)\left(\rho_x^{\text{in}}\otimes|0\rangle\langle 0|^{\otimes\text{hid, out}}\right)\left(\prod_{l=1}^{k}\prod_{j=1}^{\overrightarrow{ql}}U_j^l(t)^\dagger\right), \\
&\left(\prod_{l=k}^{L}\prod_{j=\overleftarrow{q_l}}^{N_l}U_j^l(t)^\dagger\right)\left(\mathbb{I}^{\text{in, hid}}\otimes|\phi_x^*\rangle\langle\phi_x^*|\right)\left(\prod_{l=L}^{k}\prod_{j=N_l}^{\overleftarrow{q_l}}U_j^l(t)\right)]\,.iK_q^k(t)\}
\end{aligned}
$$

$$= \frac{i}{N}\operatorname*{Tr}_{\text{all}}\left\{\sum_x\sum_k\sum_q M_q^k(t).K_q^k(t)\right\} \quad \text{(B.9)}$$

Equation B.9 shows that the gradients of the cost function can be implemented in the form of a quantum unitary evolution, as the exponential of a Hermitian matrix. Therefore, the fidelity

as a cost function facilitates the elaboration of a quantum learning rule. To apply this learning rule in an algorithm, an explicit form must be derived for the parameter matrices $K_j^l(t)$. We drop the iteration indices t for simplicity. Each parameter matrix can be decomposed in a basis $\{\sigma^{\alpha_1} \otimes \cdots \otimes \sigma^{\alpha_{N_l}} \otimes \sigma^{\beta}\}$, where σ^{α_i} and σ^{β} act on qubits in layer $l-1$ and l respectively:

$$K_j^l = \sum_{\alpha_i,\beta} K_{j,\alpha_i,\beta}^l (\sigma^{\alpha_1} \otimes \cdots \otimes \sigma^{\alpha_{N_l}} \otimes \sigma^{\beta}). \tag{B.10}$$

For example, the Pauli matrices complemented with the identity are a valid basis. We choose K_j^l such that it maximizes the gradient $\frac{dC}{dt}$ in equation B.9. Since the cost derivative is linear in K_j^l, its extrema are realized for infinite values. To avoid this issue, a Lagrangian multiplier λ is introduced as a regularizer to ensure finite updates. We optimize the resulting gradient:

$$\begin{aligned}
&\max_{K_{j,\alpha_i,\beta}^l} \left\{ \frac{dC}{dt} - \lambda \sum_{\alpha_i,\beta} {K_{j,\alpha_i,\beta}^l}^2 \right\} \\
&= \max_{K_{j,\alpha_i,\beta}^l} \left\{ \frac{i}{N} \operatorname*{Tr}_{\text{all}} \left\{ \sum_x \sum_k \sum_q M_q^k K_q^k \right\} - \lambda \sum_{\alpha_i,\beta} {K_{j,\alpha_i,\beta}^l}^2 \right\} \\
&= \max_{K_{j,\alpha_i,\beta}^l} \frac{i}{N} \operatorname*{Tr}_{\gamma \in K_j^l} \left\{ \sum_x \sum_k \sum_q \operatorname*{Tr}_{\bar{\gamma} \notin K_j^l} \left\{ M_q^k K_q^k \right\} \right\} - \lambda \sum_{\alpha_i,\beta} {K_{j,\alpha_i,\beta}^l}^2
\end{aligned} \tag{B.11}$$

where γ stands for all qubits U_j^l acts on, and $\bar{\gamma}$ all the remaining qubits in the network. As usual, to maximize the update, the derivative of equation B.11 with respect to $K_{j,\alpha_i,\beta}^l$ is calculated and set to 0:

$$\frac{i}{N} \operatorname*{Tr}_{\gamma \in K_j^l} \left\{ \sum_x \operatorname*{Tr}_{\bar{\gamma} \notin K_j^l} \left\{ M_j^l \right\} \left(\sigma^{\alpha_1} \otimes \cdots \otimes \sigma^{\alpha_{N_l}} \otimes \sigma^{\beta} \right) \right\} - 2\lambda K_{j,\alpha_i,\beta}^l = 0, \tag{B.12}$$

from which we find the coefficients for the parameter matrix:

$$K_{j,\alpha_i,\beta}^l = \frac{i}{2\lambda N} \sum_x \operatorname*{Tr}_{\gamma \in K_j^l} \left\{ \operatorname*{Tr}_{\bar{\gamma} \notin K_j^l} \left\{ M_k^l \right\} . \left(\sigma^{\alpha_1} \otimes \cdots \otimes \sigma^{\alpha_{N_l}} \otimes \sigma^{\beta} \right) \right\}. \tag{B.13}$$

Coming back to the expression for the parameter matrix in equation B.10 and writing the explicit formula for its coefficients, the explicit form of the parameter matrix results in:

$$\begin{aligned}
K_j^l &= \sum_{\alpha_i,\beta} K_{j,\alpha_i,\beta}^l (\sigma^{\alpha_1} \otimes \cdots \otimes \sigma^{\beta}) \\
&= \frac{i}{2\lambda N} \sum_x \sum_{\alpha_i,\beta} \operatorname*{Tr}_{\gamma \in K_j^l} \left\{ \operatorname*{Tr}_{\bar{\gamma} \notin K_j^l} \left\{ M_k^l \right\} . \left(\sigma^{\alpha_1} \otimes \cdots \otimes \sigma^{\alpha_{N_l}} \otimes \sigma^{\beta} \right) \right\} \left(\sigma^{\alpha_1} \otimes \cdots \otimes \sigma^{\beta} \right) \\
&= \frac{i}{2\lambda N} \sum_x 2^{N_{l-1}+1} \operatorname*{Tr}_{\gamma \in K_j^l} \left\{ M_k^l \right\} \\
&= \frac{i 2^{N_{l-1}+1}}{2\lambda N} \sum_x \operatorname*{Tr}_{\gamma \in K_j^l} \left\{ M_k^l \right\}
\end{aligned} \tag{B.14}$$

where

$$
\begin{aligned}
M_j^l(t) = [& \left(\prod_{l=k}^{1} \prod_{j=\overrightarrow{q_l}}^{1} U_j^l(t) \right) \left(\rho_x^{\text{in}} \otimes |0\rangle\langle 0|^{\otimes \text{hid, out}} \right) \left(\prod_{l=1}^{k} \prod_{j=1}^{\overrightarrow{q_l}} U_j^l(t)^\dagger \right), \\
& \left(\prod_{l=k}^{L} \prod_{j=\overleftarrow{q_l}}^{N_l} U_j^l(t)^\dagger \right) \left(\mathbb{I}^{\text{in, hid}} \otimes |\phi_x^*\rangle\langle\phi_x^*| \right) \left(\prod_{l=L}^{k} \prod_{j=N_l}^{\overleftarrow{q_l}} U_j^l(t) \right)].
\end{aligned}
\tag{B.15}
$$

Appendix C

Entropy correlations

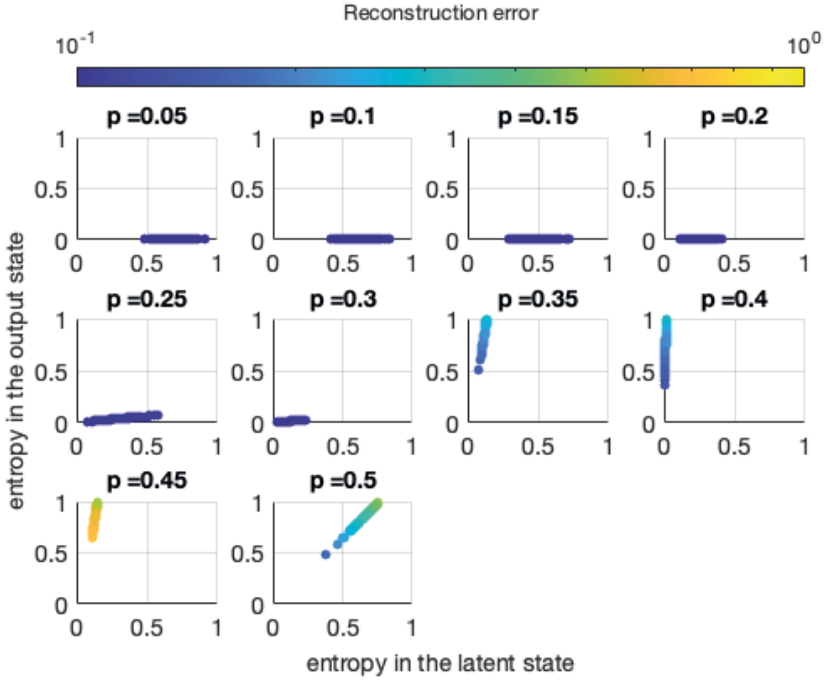
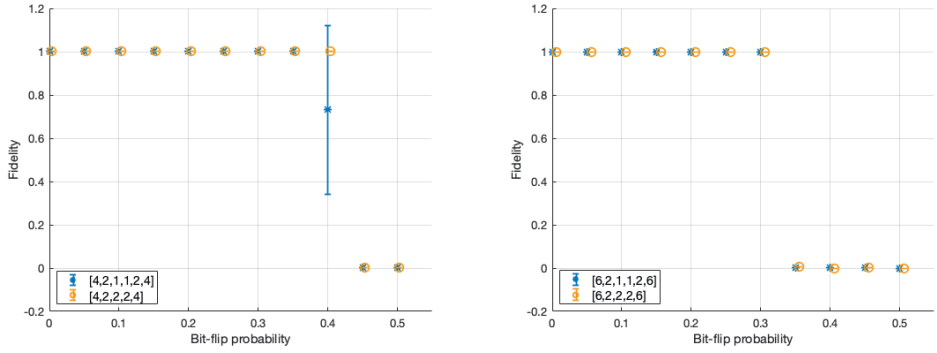


Figure C.1: View of the correlation between the output and latent state's entropies, when the network is trained with the bit-flip channel with probability p and tested with random pure states. These two quantities are linked by linear relations, and an abrupt change of behavior occurs past the tolerance threshold of $p = 0.3$, where the proportionality coefficient becomes suddenly finite. The linear fit for the proportionality factors is shown in figure (3.3 c)

Appendix D

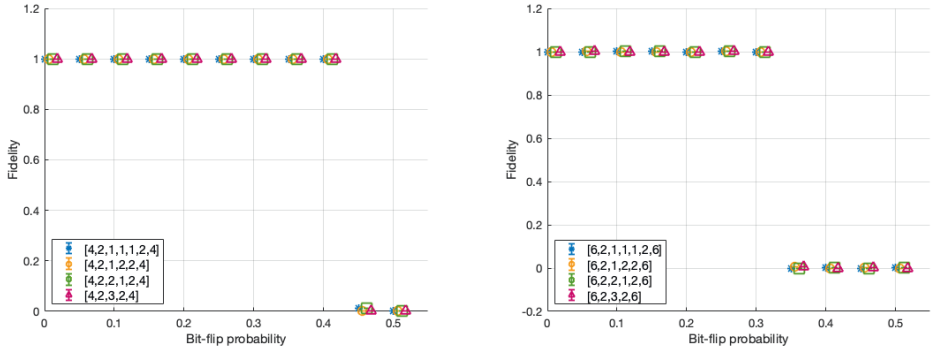
Test results for the brain boxes

For completeness, we provide the results for the testings for the networks completed with the brain boxes in section 3.2.2, for 2, 3, and 4 qubits in the brain box. For 2-qubit brain boxes, a difference is notable between the [1,1] and [2] qubit-brain boxes: the single-layer box has improved tolerance to noise, and the variations between the different states are canceled, as shown by the vanishing standard deviations. For the other brain boxes, the testing results do not vary: they all saturate at the same tolerance threshold and can complete the denoising perfectly for all states in the data set below the threshold. In addition, the fidelity in the strong noise regime vanishes completely: even the noiseless GHZ-state cannot be recovered.



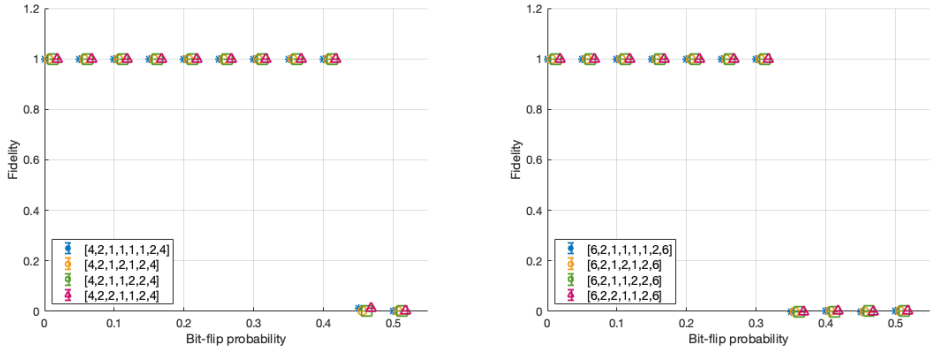
(a) (b)

Figure D.1: Testing results for brain box-enhanced QAE. The 4- and 6-qubits inputs are considered in (a) and (b) respectively. The brain boxes have no more than 2 qubits.



(a) (b)

Figure D.2: Testing results for brain box-enhanced QAE. The 4- and 6-qubits inputs are considered in (a) and (b) respectively. The brain boxes have no more than 3 qubits.



(a) (b)

Figure D.3: Testing results for brain box-enhanced QAE. The 4- and 6-qubits inputs are considered in (a) and (b) respectively. The brain boxes have no more than 4 qubits.

Band / Volume 69
Disentangling parallel conduction channels by charge transport measurements on surfaces with a multi-tip scanning tunneling microscope
S. Just (2021), xii, 225 pp
ISBN: 978-3-95806-574-1

Band / Volume 70
Nanoscale four-point charge transport measurements in topological insulator thin films
A. Leis (2021), ix, 153 pp
ISBN: 978-3-95806-580-2

Band / Volume 71
Investigating the Interaction between π-Conjugated Organic Molecules and Metal Surfaces with Photoemission Tomography
X. Yang (2021), xviii, 173 pp
ISBN: 978-3-95806-584-0

Band / Volume 72
Three-Dimensional Polymeric Topographies for Neural Interfaces
F. Milos (2021), 133 pp
ISBN: 978-3-95806-586-4

Band / Volume 73
Development, characterization, and application of compliantintracortical implants
K. Srikantharajah (2021), xiv, 155, xv-xvii pp
ISBN: 978-3-95806-587-1

Band / Volume 74
Modelling, implementation and characterization of a Bias-DAC in CMOS as a building block for scalable cryogenic control electronics for future quantum computers
P. N. Vliex (2021), xiv, 107, xv-xxviii pp
ISBN: 978-3-95806-588-8

Band / Volume 75
Development of Electrochemical Aptasensors for the Highly Sensitive, Selective, and Discriminatory Detection of Malaria Biomarkers
G. Figueroa Miranda (2021), XI, 135 pp
ISBN: 978-3-95806-589-5

Band / Volume 76
Nanostraw- Nanocavity MEAs as a new tool for long-term and high sensitive recording of neuronal signals
P. Shokoohimehr (2021), xi, 136 pp
ISBN: 978-3-95806-593-2

Band / Volume 77
Surface plasmon-enhanced molecular switching for optoelectronic applications
B. Lenyk (2021), x, 129 pp
ISBN: 978-3-95806-595-6

Band / Volume 78
Engineering neuronal networks in vitro: From single cells to population connectivity
I. Tihaa (2021), viii, 242 pp
ISBN: 978-3-95806-597-0

Band / Volume 79
Spectromicroscopic investigation of local redox processes in resistive switching transition metal oxides
T. Heisig (2022), vi, 186 pp
ISBN: 978-3-95806-609-0

Band / Volume 80
Integrated Control Electronics for Qubits at Ultra Low Temperature
D. Nielinger (2022), xviii, 94, xix-xxvi
ISBN: 978-3-95806-631-1

Band / Volume 81
Higher-order correlation analysis in massively parallel recordings in behaving monkey
A. Stella (2022), xiv, 184 pp
ISBN: 978-3-95806-640-3

Band / Volume 82
Denoising with Quantum Machine Learning
J. Pazem (2022), 106 pp
ISBN: 978-3-95806-641-0

Information
Band / Volume 82
ISBN 978-3-95806-641-0

Mitglied der Helmholtz-Gemeinschaft

JÜLICH
Forschungszentrum