UNIVERSITÀ
DEGLI STUDI
FIRENZE

Mathematics, Physics
and Natural Sciences School
PhD course in
Physics and Astronomy

Design and Training of Quantum Machine Learning Models for Noise Sensing and Phases of Matter Classification

Supervisors:

Prof. Leonardo Banchi
Prof. Filippo Caruso

PhD candidate:

Paolo Braccia

Academic year: 2022/2023

Acknowledgements

Ringraziamenti

Questi tre anni avrebbero avuto un sapore diverso senza le persone che mi hanno accompagnato in questo viaggio. È difficile mettere per iscritto la gratitudine senza scadere in una cringiosa banalità, ma farò del mio meglio (spoiler: fallirò).

In primis, ci tengo a ringraziare i miei relatori, per essere riusciti a guidarmi attraverso questo percorso nonostante la pandemia e le conseguenti complicazioni. Insieme a loro, tutte le persone che ho potuto conoscere grazie a questo lavoro e che hanno contribuito a renderlo leggero.
Alla mia famiglia, oltre che grazie, voglio dire "continuiamo così" perché questi anni, con le loro difficoltà, ci hanno reso più forti ed uniti.
I miei bromi, chi sarei senza di loro? Grazie per la pazza compagnia e per essere sempre pronti a condividere gioie e, soprattutto, dolori. No sul serio, senza di voi sarebbe un bel casino.
Ringrazio la mia squadra, e in particolare i gorgoni, fonte inesauribile di risate e leggerezza.
Un grazie al mitonniere, che sebbene dall'altra parte del mondo è sempre stata presente.
Grazie Pranzo! Così, per metterti in difficoltà.
Infine, grazie alla persona che più di tutte associo a questo dottorato, non tanto perché dea quantistica (stacce) quanto perché portatrice di psicopatie perfettamente compatibile alle mie. Non basterebbe un praticissimo caschetto porta acqua per sdebitarmi del tuo supporto.
In coda a tutti questi ringraziamenti, un'unica fondamentale offesa: nespole, fate schifo!

Contents

Overview

Since ancient times, humankind has inherently sought to simplify its life by automating disparate tasks. Ever since, advances in this field have corresponded to epochal progressions of our species, think of the invention of the wheel for transportation, all the way to the industrial revolution. While these efforts were initially expended toward automating mechanical tasks, with the advent of computers (DATA) and the onset of the information age, much energy has begun to be invested in creating increasingly high-performance computational models to automate the way we process information. Then with the advent of the Internet, every day we produce an enormous amount of data that we have discovered can be used to improve (but alas, also make worse in some cases) our daily lives. In recent years, one area of computational science has taken the limelight, so much so that it is now one of the first applications that come to mind when we talk about technology, although it is often misunderstood and associated with sci-fi scenarios. This area is the famous Machine Learning, also often known as Artificial Intelligence. In fact, thanks to technological advancement and the creation of increasingly high-performance computers, results that previously remained only theoretical in the field of machine learning are now employable on a large scale to make the most of the immense amount of data we were talking about earlier. At the same time, these advances present us with an even greater challenge, the implementation of these solutions in an efficient and high-performance manner. Moore's law inevitably brings us to the point where the processors we will need in the future will be of such a size that they must be described by the laws that govern the microscopic world, quantum mechanics. The inevitability of this fact is not a condemnation, however, so much as a gateway to a new era of computation. We are talking about quantum computation. This computation paradigm, which unfortunately is still far from being fully implemented on an industrial scale, promises to revolutionize the way we process information by appealing to the laws of quantum mechanics. These will allow us to build algorithms and computational procedures that can beat their classical counterparts dramatically, being able to go so far as to require exponentially fewer resources. As much as hardware technology is not yet up to speed with the theoretical framework that quantum computing has established since the 1980s, the first experimental quantum processors are now beginning to become available, albeit they are still small, unreliable, and noisy. These detrimental

properties led the quantum computing community to refer to them as NISQ processors, abbreviation for Noisy Intermediate-Scale Quantum. Interestingly, learning algorithms seem to be best suited for this early era of quantum computing. In fact, such computational processes can and have to be supported by classical processors, resulting in a hybrid computational scheme that is proving more robust to noise and promises to be one of the first commercial implementations of the quantum world. This has meant that in the past 5 years, a new branch of quantum computational physics, quantum machine learning, has rapidly taken hold and led to the generalization to the quantum world of the most popular machine learning models that are characterizing this era. A great deal of effort is currently being expended to "quantize" classical learning techniques and to propose new learning algorithms that take full advantage of the quantum properties of the processors on which they will run. Possibly over-hyped by misconceptions about quantum “magic", this new rising field of Quantum Machine Learning (QML) is nonetheless a revolutionary one that combines the power of quantum computers with the insights of machine learning. By leveraging the unique properties of quantum systems, this approach has the potential to solve complex problems and make predictions with unprecedented accuracy. With its potential to transform industries and drive scientific discovery, quantum machine learning is set to be a major driving force in the 21st century.

The work presented below, the result of the research carried out during these three years of doctoral studies, is set precisely in the context of quantum machine learning and addresses the problem of finding good design and training strategies for quantum learning models. When we talk about quantum models we refer, in a somewhat pop sense, to a quantum generalization of the famous neural networks that revolutionized classical machine learning as soon as hardware computational capabilities were able to handle their large resource consumption. In the quantum world when we talk about Quantum Neural Networks, we are referring to the possibility of parameterizing and training the physical evolution of a quantum system in which we have encoded the information we want to process, by literally modifying its interactions with a control environment. Therefore, the problem of choosing the architecture, i.e., how to go about manipulating the evolution of the quantum system under consideration, and finding effective strategies for finding the ideal form of this evolution is of paramount importance. Just as in the classical world it has been realized that neural networks, although they can contain the solution to any problem if deep enough, are in general impossible to train successfully unless their structure is appropriately tailored to the problem under consideration. Examples of this are the success of convolutional networks in dealing with classification problems, recurrent networks for generating temporal predictions, etc.

The first part of this research addresses this problem on the particular learning framework that is generative adversarial learning. This paradigm, based on Nash game theory, has achieved tremendous success in recent years in classical machine learning and has therefore naturally been considered for generalization to the quantum world. Nevertheless, in trying to train these models, problems can be encountered when information is encoded in noisy (i.e., mixed) states. The first result we achieved was to find a training strategy for QGANs that makes it possible to achieve the optimal equilibrium configuration even in the presence of noise.
This result then led us to design an architecture based on adversarial generative learning that, instead of learning the information encoded within an unknown quantum state, is able to reproduce a quantum process. Specifically, through the SuperQGANs we have introduced, it is possible to learn an approximation of the very noise that plagues a quantum processor and disrupts its operation. Arguably, that of characterizing noise is one of the most pressing and important problems of the NISQ era in which we live, because knowing the characteristics of it can allow us to find optimal strategies to make the best use of its properties.
In the last part of this research, we instead approached the problem of NISQ design from a more general (and foundational) point of view. Indeed, it is a recent achievement of classical machine learning to be able to build neural network architectures that possess cognitive biases. As mentioned, neural networks can guarantee that they contain within them the solution to the problem they face, but in general they have no idea about the specific properties of the latter. As a result, relying on networks that are too deep and then searching for the solution amidst a huge space of incorrect solutions can lead to the failure of the learning process.
In this sense, cognitive biases become necessary. By this term we refer to the prior intuition that, as humans, we have with respect to a problem. Often many features of the data we want to analyze are irrelevant. For example, a child who sees a picture of a kitten is immediately able to tell that the mirrored version of the picture contains the same cat. He does not need to "reprocess" the picture as new. A generic learning model is unable to make this association a priori, forcing us to train it on mirrored, translated, etc. variants of the same image. Recently, however, strategies have been discovered to embed these cognitive biases directly into the architecture of neural networks, without having to impose them in the training phase. The last part of this research has been devoted to the implementation of cognitive biases in quantum learning models, leading to the formulation of Geometric QML, a framework containing the recipes needed to cook QNNs capable of ignoring the symmetries of the problem they address, thus going for the solution in a reduced and more "benign" space. We believe that this can be a major step forward to arrive at building quantum learning models that are as easily implementable as possible on NISQ processors.

This PhD thesis is organized as follows.
In Chapter 1 we introduce some basic concepts of the machine learning field, making the reader familiar with the basing ingredients that any learning recipe is cooked up with. Particularly, we will discuss the three main paradigms of machine learning: supervised, unsupervised and reinforcement learning. Then we will comment about the role of data and move on to talk about Neural Networks, the main learning model that is used in any learning task. We will then show how to set up their training procedure. Most importantly, the last section of this chapter will introduce the generative adversarial learning framework that will be extensively used in later chapters.
The next two Chapters, give the reader a crash course on quantum computing first and then set the stage for quantum machine learning routines. The main focus of Chapter 2 is to introduce all the quantum notions and notations that will constantly be used in the rest of the manuscript. We will start by stating the postulates quantum mechanics rests on, using the circuital description of a prototypical quantum computation to make the reader familiar with it. Then we will also discuss what happens when reality, *i.e.* noise, is is taken into account. In fact, in real world quantum computation the ideal description of quantum circuits as being isolated quantum systems breaks, and one needs to adopt a more general formalism that takes unwanted interaction with an unpredictable environment into account. Chapter 3 builds up the quantum machine learning framework by drawing a parallelism with the machine learning introduction presented in Chapter 1. The main takeaways from this chapter are the definition of quantum learning models as parameterized quantum circuits, also called quantum neural networks, namely sequences of logical quantum operations that can be tuned by changing some parameters, and how we can gather information on how to update their values through the computation of quantum gradients.
This concludes the introductory part of this thesis, and from Chapter 4 the original part of it begins.

Chapter 4 introduces quantum generative adversarial learning, a promising strategy to use quantum devices for quantum estimation or generative tasks. After reviewing its ability to properly learn data stored in non-noisy quantum states, already known to the literature, the convergence behaviours of its training process when those states are noisy instead, which is crucial for its practical implementation on quantum processors, is investigated. We show how different training problems may occur during the optimization process, such as the emergence of a phenomenon known as limit cycles. The latter may remarkably extend the convergence time in the scenario of mixed quantum states playing a crucial role in the already available noisy intermediate scale quantum devices. We propose new strategies to achieve a faster convergence in any operating regime and test their effectiveness with a numerical analysis.

Then, in Chapter 5 we leverage this improved convergence ability of quantum generative adversarial learning to design an architecture that is able to reconstruct and characterize the noise affecting a real quantum processor. Super quantum generative adversarial networks (SuperQGANs), as we decided to call this newly introduced architecture, generalize the previously studied learning paradigm from quantum states to quantum maps, or superoperators. SuperQGANs are not only able to reproduce quantum noise in the form of a particular class of quantum maps, but also to characterize the correlations, be those temporal or spatial, that emerge when the processor is used multiple times in series or parallel. After describing their architecture, we end the chapter by testing their performance by numerical experiments, and we also show how to employ them for quantum metrology applications.
Lastly, in Chapter 6 we introduce the geometric quantum machine learning framework. After motivating the need for inductive biases in quantum learning models, we proceed to review some basic notions of group and representation theory that are needed in order to derive the building blocks of informed quantum neural networks, *i.e.* equivariant quantum maps. The concept and uses of equivariance will be discussed in detail, and then methods for building and parameterizing such quantum operations will be shown. We will conclude the chapter by showing how equivariant quantum learning models perform better than problem agnostic ones on phases of matter classification tasks.
Conclusions and outlooks are drawn in the final Chapter.

Chapter 1

Machine Learning

This chapter aims at making the reader familiar with the basic concepts of machine learning. Particularly, after a general introduction on the topic of artificial intelligence (Sec. 1.1), the three main learning paradigms are discussed and the universal recipe for learning tasks is laid down (Secs. 1.2,1.3). Lastly, in Sec. 1.4 we introduce in more detail a particular learning framework: generative adversarial networks, which will lie at the core of the first part of this work.
The main references for this chapter are [1–3].

1.1 Artificial intelligence

Over the last decade terms like *Artificial Intelligence* (AI) and *Machine Learning* (ML) have become more and more popular, to the point that nowadays they are the first ones to come to our minds when we think about technology. But what do they actually mean?
Under the AI framework falls any craft, be it hardware or software, that tries to mimic the innate human ability to learn from their environment and use the acquired knowledge to make predictions and benefits. Thus, achieving a general artificial intelligence, as in the Asimov fictional world, is the final goal of AI research. Now that the goal is clear, how could a machine ever behave as we do? A focal point of our way of learning is that we develop our knowledge by acquiring data from the world and inferring, with the help of teachers, the correlation laws between them. In the very same way, Machine Learning (ML) is the use of logical algorithms, guided by the statistical theory of computation, to make a computer learn from data, labelled and raw, without being explicitly programmed.
An interesting fact about ML is that, as recent as it may appear due to the pop interest that it has attracted, the term machine learning debuted back in the late '50s, and most of the theoretical foundations of the field were laid during the '70s and '80s. The reason why ML has risen to prominence is the advance in hardware technology, particularly the capability of nowadays com-

puters to store and process the huge amounts of data that are needed to carry on learning tasks. We daily interact with machine learning and AI, even if we do not notice this interaction most of the times. The most frequent of these interactions is called profiling. Basically, every time we search for something on the web, or even pause to look at an insertion on our favourite social network, this action is saved by an algorithm and the thing we were searching for is labeled as something we may like. Then this information is used by an AI to present us with fine-tuned advertisements. Thus, the more we browse the more personalized our browsing experience becomes. Another common example are self-driving cars, but AI applications are nearly infinite, from the discovery of new drugs to cancer detection.
The standard ML introduction would now go on and define in detail the various approaches one can take to get a computer to learn from data, making the reader familiar with the three main classes of machine learning: *supervised*, *unsupervised* and *reinforcement* learning. However, since this is not the scope of this thesis, we will just give a bird's-eye view of these frameworks. Before doing this, let us stress that each of these frameworks have their benefits and drawbacks, and that there is no god-like algorithm that is able to accomplish learning for every possible task. This is the content of the so called *No Free Lunch Theorem* [4].

1.1.1 Supervised Learning

In supervised learning tasks, a machine has to infer a function given a set of known input-output pairs. One usually refers to inputs as *feature vectors*, while outputs are commonly named *labels*. A nice recent result of this kind of ML approach is *Google Lens* [5], an AI driven tool that is able to recognize, *i.e.* label, the semantic content of a picture, informing the user about what kind of tree, bug, etc. they are seeing. In general, the function we want to learn may describe any kind of relation present in the data, we can even use it to make future predictions. Regardless of this, what defines a supervised learning task is the human supervision to the machine, namely the preparation of a labeled training dataset that the computer can exploit to understand the relation that we want it to then find in new, unseen, data. We can formally state a supervised learning problem as follows: given a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$ of labeled examples $x \in \mathcal{X}$ belonging to some data space $\mathcal{X}$ with $\mathcal{Y}$ being the space of labels, learn the relation $f : \mathcal{X} \to \mathcal{Y}$, so that upon feeding it a new data point $\tilde{x} \notin \mathcal{D}$ it predicts the correct output $\tilde{y} = f(\tilde{x})$.

Even though this seems a very simple, maybe even childish, learning framework, things can go wrong in lots of ways. Above all, the possibility of *overfitting* and thus not being able to generalize well. Overfitting means that our model gets very biased towards the set of examples we used during training, eventually learning these *by heart* rather than understanding the underlying

fundamental rules. This causes the model, when presented with previously unseen data that we want to analyze to make predictions, to do poorly, like a bad scholar being caught unprepared. Being able to avoid this behaviour is the ultimate quality we look for in our learner, and this property is referred to as *generalization*. When training, we aim at achieving maximum generalization power, so that the model gets to learn the true relationship underlying the data at hand. Training for too long, or using over-complex models may lead to overtraining and ultimately to overfitting. Another pitfall to be avoided in supervised learning is inducing biases during learning. Supervision is surely a strong aid to the learning process, but since the model can only rely on what we choose to show him, it is paramount to construct a training dataset that is not skewed and biased. Often, finding enough training data, and preprocessing it into a good training dataset turns out to be the true challenge of supervised learning.

Finally let us recall the most common applications for supervised learning:

- **Classification**. classification is the process of predicting a categorical label for a given input data sample. Some common examples of classification tasks include: email spam detection, *i.e.* given a list of emails, predict whether each email is spam or not spam; sentiment analysis, namely predicting whether a given a piece of text expresses a positive, negative, or neutral sentiment; fraud detection: given a set of financial transactions, predict which transactions are fraudulent and which are legitimate. Image classification, *i.e.* predicting what object or objects are present in an image. The models that perform classification are, unsurprisingly, called *classifiers*.

- **Regression**. Regression is a type of machine learning task that involves predicting a continuous numerical value for a given input data sample. Some examples of regression tasks include: predicting the price of a house based on its characteristics (e.g., size, location, age, etc.); predicting the demand for a product based on various factors (e.g., price, marketing efforts, seasonality, etc.); predicting the effectiveness of a medical treatment based on patient characteristics (e.g., age, gender, medical history, etc.); predicting the likelihood of a customer churning (leaving a company) based on their behavior and other factors. In a regression task, the input data is usually represented as a set of feature vectors, and the goal is to learn a function that maps these feature vectors to a continuous numerical output value. This function is called a *regressor*.

1.1.2 Unsupervised Learning

As the name suggests, unsupervised learning does without the human help. Only unlabeled data are available in training. Nonetheless, one does not always need supervision to learn patterns in data, afterall it is a no-brainer to

group marbles according to their colours, even if nobody told us the colours' names, or even what a colour is. An unsupervised learning task can be formally stated as: given a space $\mathcal{X}$ where to draw samples from, and given a dataset $\mathcal{D} = \{x_i\}_{i=1}^N$ of points drawn according to some, unknown, probability distribution $p(x)$, learn the latter. A nice property of learning without supervision, is that the model gets to have less bias and may even think outside the box, reaching a better solution than a supervisor may think of. Therefore, unsupervised learning is also referred to as knowledge discovery, and it is often used with profit to carry on exploratory data analysis. However, one of the main challenges of unsupervised learning is evaluating the quality of the model's output. Since the model is not provided with labeled data, it can be difficult to determine whether the discovered patterns and relationships are meaningful or simply artifacts of the data. As a result, unsupervised learning often requires domain expertise and human interpretation to be useful. Some examples of tasks that are tackled with this learning approach are the following

- **Clustering**. Clustering means grouping together subsets of data that share the same properties, as in the coloured marbles example. Now, not having access to any external help, there are no restrictions on the number of clusters we can look for, and while this freedom may sound nice, this actually means that finding the correct complexity of the model becomes an empirical trial and error process.

- **Dimensionality Reduction**. Sometimes the data that we want to use to infer some property might contain redundant features that would ultimately just confuse the learner. If the human setting up the learning process cannot spot them on their own, or if the problem at hand is very complex, say we want to predict the stock price of some product from personal data collected from social networks, we can resort to dimensionality reduction. The idea is to compress the feature space, the space where datapoints live, to have a smaller dimension in such a way that only the most important features are represented whereas irrelevant ones are suppressed. Principal Component Analysis (PCA) is the go-to routine for this kind of tasks.

1.1.3 Reinforcement Learning

Lastly, there is the reinforcement learning framework, arguably the one that is closest to how animals learn. In this approach, no examples or labels are given. Rather, the learner is let free to interact with an environment, and depending on the actions they choose to take they get either rewarded or punished. Positive rewards reinforce good strategies, while negative ones make the learner refrain from those that are detrimental. This is just what we do to train our pets, they learn how to well behave by associating it to treats reward. One of the most peculiar characteristics of Reinforcement Learning is that it

does not make use of static datasets, but rather a dynamic environment where to exploit trial and error strategies, and that is why it is the best approach to tackle automation in fields like automotive or gaming. Gaming is also the best framework to explain a typical reinforcement learning routine. Basically, the learner plays the game over and over again, collecting rewards that depend on the strategy they use. At every iteration, they update their policy to maximize the expected reward at the end of the game, until they eventually master it.

One of the main challenges of reinforcement learning is balancing exploration (trying out new actions to see what happens) with exploitation (using the actions that have proven most effective so far). If the agent focuses too much on exploration, it may take a long time to learn a good policy. If it focuses too much on exploitation, it may get stuck in a suboptimal policy. Finding the right balance is important for efficient reinforcement learning.

Some examples reinforcement learning tasks are

- **Robotics**. Given a robot and a reward signal based on its performance, learn a policy that allows the robot to perform a task effectively.
- **Resource allocation**. Given a system with limited resources and basing the reward given to the agent on the efficiency of the system, have the learner learn a policy that maximizes the reward.
- **Games**. Given a set of possible moves in the game and a reward function based on those and on the game's outcome, learn a policy that maximizes the reward and ultimately makes you win as much as possible.

Reinforcement learning algorithms include Q-learning [6], SARSA [7], and deep Q-networks (DQN) [8]. The choice of algorithm will depend on the nature of the environment and the specific task at hand.

1.2 Ingredients of ML

When dealing with a machine learning task, be it supervised or unsupervised, it eventually all boils down to cook with three ingredients: the data available, the learning models we can choose from, and the training objective function, often called *loss function*. Reinforcement learning is a different kind of recipe, and we will refrain to go into its details here.

1.2.1 Data

With the explosion of the ML field, thanks to computer finally being able to handle loads of data in reasonable time, gathering and controlling data has become a task of paramount importance, so much that people have started to

refer to it as the *new oil.*[1]
Just collecting, and eventually labeling, the data is not enough though. There are a few assumptions that are needed in order to fall into the mathematical framework of a working ML algorithm. First of all, the data must be *i.i.d.*, that is independently and identically drawn from the distribution they come from. This is a somewhat unrealistic assumption when it comes to real-world data, but is nonetheless needed to make use of the underlying statistical theory supporting ML convergence guarantees. Practically, this means that we need to put all the effort possible into making sure that the dataset we are going to work with is as little biased as possible. Then, raw data sometimes is not enough. Most learning algorithms are picky, and they require *data preprocessing* before they actually try to learn. With preprocessing one refers to all the actions that come after the collection of the data and before beginning the training of the model. A few examples of this are the *rescaling* of data, for example to have all data-points bounded in norm, or to change mean and variance of their distribution and *feature selection*, through which some components of the feature vectors are discarded or merged together [9]

1.2.2 Model

Generally speaking, in ML we call *model* the, usually parametrized, family of learners that we want to train to solve our task. Thinking about the supervised learning scenario, a model would be a parametrized function $f_{\boldsymbol{\theta}} : \mathcal{X} \to \mathcal{Y}$ for which we want to find the optimal values of parameters $\boldsymbol{\theta}^*$ that best reproduces the relation between feature space $\mathcal{X}$ and label space $\mathcal{Y}$, as hiddenly described by the dataset $\mathcal{D}$. Notice that we used a single $\boldsymbol{\theta}$ symbol to identify all of the model's parameters, but this does not mean that they have to be continuous or smooth. Most often than not, *hyperparameters* are thrown in the mix. These are usually discrete, or even non-numerical parameters that give additional freedom to the model's family. As a dummy example, consider a linear regression problem where we want to fit $y = f_{\boldsymbol{\theta}}(x) = \boldsymbol{\theta}_0 + \boldsymbol{\theta}_1 x$ to some given data $\mathcal{D} = \{(x, y)\}$. Here intercept $\boldsymbol{\theta}_0$ and slope $\boldsymbol{\theta}_1$ are the smooth parameters of the model, whereas there are no hyperparameters to choose from. When introducing more complex models in the next section we will show an example of hyperparameters too. Let us use this example to briefly distinguish between *deterministic* and *probabilistic* models. The former, akin to the regression example, are families of functions that map the input data space $\mathcal{X}$ to the output space $\mathcal{Y}$, which can be either a numerical field, as for predictive models for, say, market prices, or a discrete space of labels for classification tasks. A probabilistic model instead outputs a probability, be it full or conditional. Supervised probabilistic models would then parametrize functions such as $f_{\boldsymbol{\theta}} : \mathcal{X} \times \mathcal{Y} \to [0, 1]$ s.t. $f_{\boldsymbol{\theta}}(x, y) = p_{\boldsymbol{\theta}}(x|y)$, namely the probability of, say, as-

[1] This expression was coined in 2006 by the British mathematician Clive Humby.

signing the label y upon being given the input x, or $f_{\boldsymbol{\theta}}(x, y) = p_{\boldsymbol{\theta}}(x, y)$ the full probability of the couple (x, y). An unsupervised model would simply output the distribution $p(x)$. Notice that deterministic models may have an underlying probabilistic interpretation. An example of this, that will be treated thoroughly in section 1.4, are generative models that are trained to output new samples from some underlying data distribution. The model is deterministic, because same inputs will yield identical outputs, but their aim is rather to learn a difficult distribution. In turn, probabilistic models can turn into deterministic ones once we decide a routine for drawing from the distribution they describe. For example, most classifiers work by outputting the probability distribution of the possible classes y_i associated to an input x, namely $p(y_i|x)$ and then picking the most probable as prediction $y = \underset{y_i}{\operatorname{argmax}}\, p(y_i|x)$.

Let us now briefly talk about the most known, and hyped, ML models: Neural Networks (NN). We will not delve any deep into the topic, and refer the reader to the vast literature about them, but the analogy with them will come in handy when introducing quantum machine learning models in section 3.3.

Neural Networks

Modeled after the human neural structure and functioning, *Neural Networks* (NNs) are computational models that make of their flexible and modular architecture their strength. Theorized roughly 80 years ago [10], they have achieved global popularity over the last decade thanks to computing hardware finally reaching up to their need of resources. Indeed, in an exponential growing fashion [11], deeper and heavier NNs have become practical, and this led to the rise of *Deep Learning*. The latter is just ML where deep neural networks are used as learning models [1, 2].

The ancestor, and building block, of any NN is the *perceptron* [12], which mimics the behaviour of human neuron by reproducing their *integrate and fire* mechanism. Basically, input signals are collected from the terminations that are afferent to the neuron, weighted and summed up together, this is the *integration* step, and the resulting signal determines the activation, or *firing*, of the neuron. Mathematically, if $\boldsymbol{x}$ is the collection of input signals, the output $\phi(\boldsymbol{x})$ of a perceptron reads

$$\phi(\boldsymbol{x}) = \sigma(\boldsymbol{w} \cdot \boldsymbol{x} + \boldsymbol{b})\,, \tag{1.1}$$

where $\boldsymbol{w}$ is the weights vector, $\boldsymbol{b}$ an eventual set of biases, and σ is the so called activation function. In real neurons this is basically a step function, being active when its input is above a certain threshold, and staying off otherwise. In artificial perceptrons, many different activation functions have been proposed and used throughout the vast ML community, but we will refrain from listing even only a fraction of them here and address the interested reader to any of

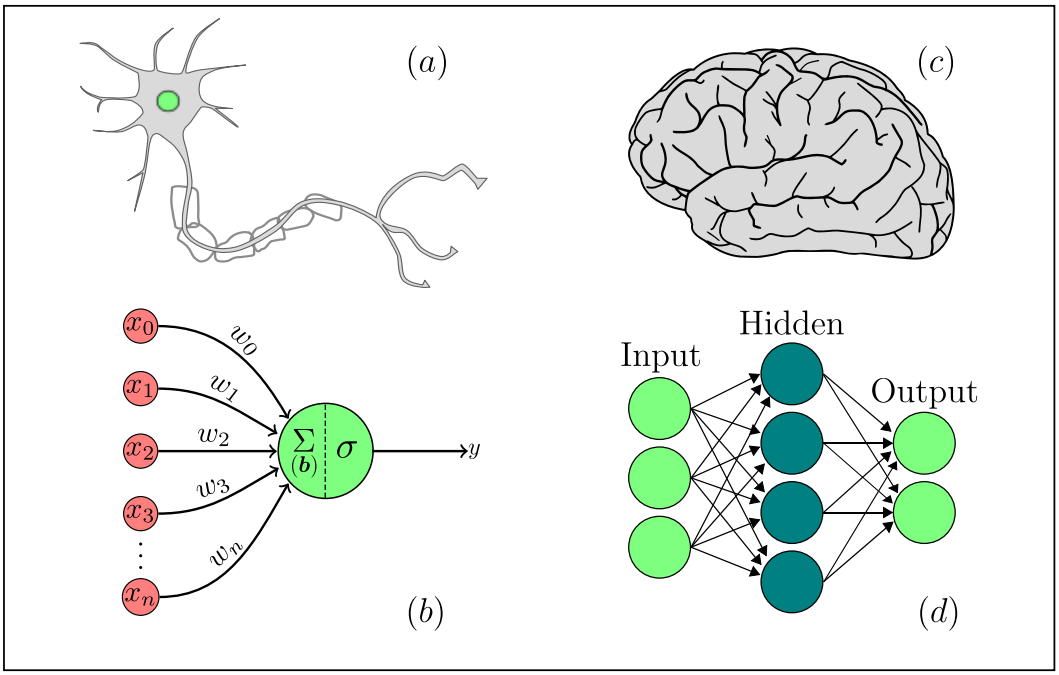


Figure 1.1: **Biological intuition behind Neural Networks.** Neural networks (d) are a computational model that mimics the functioning of our brain (c). Their constituents are the perceptrons (b), a mathematical transposition of the behaviour of the neuron cells (a). A single perceptron is able to perform linear algebra on its input signals $\boldsymbol{x} \to \boldsymbol{w} \cdot \boldsymbol{x} + \boldsymbol{b}$ and then apply a non-linear activation function σ to the result. The output signal of Eq. (1.1) is then fed into the next perceptrons, giving rise to a neural network. With enough intermediate (hidden) layers, NNs can in principle encode any function imaginable.

these great textbooks [2, 13, 14]. The important point about activation functions is that they have to be non-linear. Indeed, if this is true, stacking enough perceptrons onto each other grants us access to the *universal approximation theorem* [15]. This is exactly what neural networks were born for. Even if we do not reach the level of universal expressibility, combining many perceptrons in one network enables us to explore a huge space of input-output relations just by tuning the perceptrons weights and biases. Not to mention, the way we arrange perceptrons together in a NN determines its properties more than the sheer number of layers the network has. There are several different types of neural network architectures that are commonly used, each with its own characteristics and applications. One type is the feed-forward neural network, which consists of layers of interconnected "neurons" that process and transmit information. The input layer receives data, and each subsequent layer processes and transforms the data, until it reaches the output layer, which produces the final result. This is the standard architecture that we have considered in the previous description and that is depicted in Fig. 1.1. Another type is the convolutional neural network (CNN), which is commonly used for image and video recognition tasks. It includes features such as convolutional layers, which apply filters to the input data to identify patterns, and pooling layers, which down-sample the data to reduce the number of parameters and computational requirements. Recurrent neural networks (RNNs) are another type of neural network that are well-suited for tasks involving sequential data, such as natural language processing. RNNs include "memory" in the form of hidden states that can retain information from previous time steps, allowing them to process data with temporal dependencies. There are many other types of neural network architectures, such as autoencoders, and long short-term memory (LSTM) networks, each with their own unique characteristics and applications. All of these architectures are analyzed in depth in [2].

1.2.3 Loss

Lastly, the third main ingredient of any ML routine is the *loss function.* Known also as objective function, error, score, etc., a loss function is a measure of how good the model family we chose for the task at hand is performing. When dealing with models depending on continuous parameters $\boldsymbol{\theta}$, we always look for a continuous and differentiable function of those. This way, we can eventually rely on gradient based methods to move across the model space in order to find the best one. As a simple example of that, consider a classification task where we want to associate new data instances $\tilde{x}$ with the class they belong to, choosing from C different possibilities $\{y_i\}_{i=1}^{C}$. given a training dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$, we can define a model $f_{\boldsymbol{\theta}}$ and evaluate how bad it is doing its job via the mean squared euclidean distance between its predictions

and the correct labels

$$L(f_{\boldsymbol{\theta}}, \mathcal{D}) = \frac{1}{N} \sum_{\mathcal{D}} (f_{\boldsymbol{\theta}}(x_i) - y_i)^2 \,. \tag{1.2}$$

This is arguably the most common loss function used in ML, but ultimately the loss function is tied to the task at hand, and there are endless variation for each kind of them. For probabilistic models, usual choices are the Kullback-Leibler divergence, or the cross entropy

$$L(f_{\boldsymbol{\theta}}, \mathcal{D}) = -\sum_{i=1}^{N} y_i \ln f_{\boldsymbol{\theta}}(x_i) \,, \tag{1.3}$$

where now the y_i are the components of the true probability distribution underlying the data, and $f_{\boldsymbol{\theta}}(x_i)$ are the model's predicted probabilities.
Defining a metric for the performance of an unsupervised learning model is clearly harder, as we cannot make use of the labeled examples to build a distance measure between the target distribution and the model one. In an unsupervised scenario we are only given samples from the real distribution, and our model can either generate samples itself, or directly output its parametric distribution $p_{\boldsymbol{\theta}}(x)$, thus learning becomes a matter of comparing distributions via samples. Most of the approaches to this problem rely on Bayesian learning [16], and the standard tool used is *maximum likelihood estimation* [17]. However, we will not cover the details of this framework, because at the core of this thesis lies an alternative to it, the *generative adversarial learning* paradigm that will be explained in detail in section 1.4.

1.3 Training a ML model

Now that we have gathered all the ingredients, it is time to cook up our learning procedure. Now, every different ML task has its own preferred setup, mainly which kind of loss function to use and what class of models to train. Nonetheless, after those choices are made we are usually left with an extremization process, the most common scenario being that we just need to minimize the selected loss function. Ideally, we would want to formulate the optimization problem at hand in such a way that it is convex. This would allow us to use tools from convex optimization theory, that many times help define a closed-form solution [18]. However, most ML problems do not grant us this luxury, and all we can do is try to iteratively look for better parameters values that improve the model's performance. In this regards, the most common approach is *gradient descent*.

The simple idea behind gradient based methods is that the gradient of the objective function $L(\boldsymbol{\theta})$ that we want to, say, minimize with respect to the model's parameters $\boldsymbol{\theta}$, points towards the direction of maximum ascent and

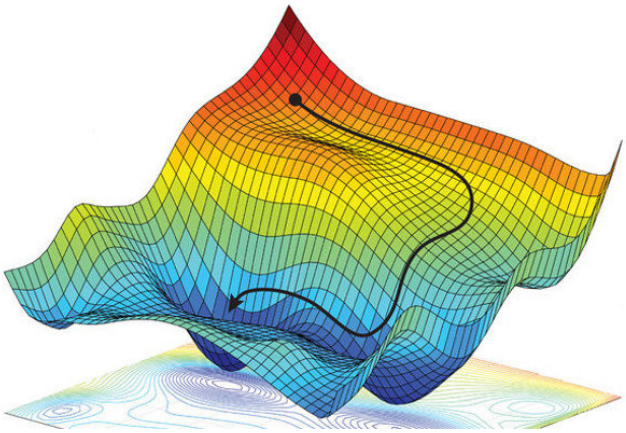


Figure 1.2: **Gradient descent trajectory through loss function landscape.** Gradient-based optimization techniques such as *gradient descent*, use the gradient $\nabla_{\boldsymbol{\theta}} L$ of the objective function L as a compass to move through the latter's landscape. The goal, in this case, is to reach L's global minimum.

thus we can follow it backwards to head to the objective function minima, as outlined in Fig. 1.2. Gradient descent update rule reads

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \eta \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}^{(t)}) \,, \tag{1.4}$$

where η is a hyperparameter known as the *learning rate* while t is an integer that keeps track of the current iteration. The gradient, $\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}^{(t)})$, indicates the direction of ascent in the landscape of the cost function, so following its negative means moving down into valleys. Tuning the learning rate η we can make the parameters update steps smaller or bigger. Finding a good value for η is of great importance, as too small values would lead to very slow training, while too large ones can potentially make the update jump over minima. It's important to note that the cost function and its gradient at each step depend on the training data, an example being Eq. (1.2). Indeed, we have been a little sloppy in the definition of the objective functions in the previous section. Since the datasets one can count on when training a ML model are finite, they will never be completely representative of the true semantic relations underlying the data. Thus the loss functions we have shown in eqs. (1.2,1.3) are only proxies for their exact versions. These data dependent proxies are called *risks* in the ML literature [2], but for us nothing changes if we omit this detail, so we will continue using the term objective (loss) function. Let us get back to gradient descent. The fact that also the gradient depends on the training data allows us to choose between different ways on how to compute it, for example *stochastic gradient descent* is a variation of gradient descent that uses only a subset of the training data to compute the gradient of the cost function at each step. Originally, this involved using a single randomly chosen training input per iteration, but more commonly it involves using mini-batches of randomly selected data, with the batch size being a hyperparameter

in the training process. While gradient descent can get stuck in local minima or at saddle points where the gradient is zero, the stochastic nature of the gradient direction in stochastic gradient descent can help avoid these issues. While standard gradient descent is guaranteed to decrease the cost function in each iteration (unless the learning rate is too high), stochastic gradient descent can fluctuate more with smaller batch sizes. There are several variations of stochastic gradient descent, including methods that dynamically change the learning rate, make the change in parameters at each step depend on the change made in previous steps, or take into account the curvature of the optimization landscape [19].

1.4 Generative Adversarial Learning

Generative adversarial learning is a machine learning framework for training generative models, specifically designed to address the problem of generating new, synthetic data samples whose probability distribution closely resembles that of the training data. It was first introduced in the breakthrough paper [20] where the term *Generative Adversarial Networks* (GANs) was coined. As in the other generative ML tasks, the final goal is to train a *generator* to be able to output new data, but GANs tackle this problem by exploiting Nash's game theory [21] (whose basic concepts are presented in Appendix A) and using a second learning agent, a *discriminator* to challenge the generator and give training feedback to it. In this section we are going to set the stage for the body of work that we are going to present later in chapter 4.

In the context of generative adversarial networks, the generator can be thought of as a counterfeiter attempting to create forgery that is indistinguishable from the real thing, while the discriminator plays the role of a detective trying to identify and distinguish the counterfeit money from genuine currency. As the generator and discriminator both learn and improve their respective skills, the generator will occasionally produce high-quality fake money that the discriminator has difficulty detecting. However, the more fake samples the discriminator analyzes, the better it becomes at recognizing genuine currency, leading to an ongoing adversarial game between the two.

A generative model is a type of parametrized agent, often implemented as a deep neural network, that is capable of synthesizing new data instances. On the other hand, a discriminative model is a classifier that is able to learn strategies for distinguishing between data coming from two different distributions. For example, a GAN may have a generator that creates new images of cats and a discriminator that classifies images as either real-world cats or synthetic ones.

Let us now add some mathematical details. First of all, let $\mathcal{X}$ denote the space where the real data samples x come from. The characteristic that we want to learn to reproduce its their distribution $P(x)$, which effectively defines the data at hand. For example, $\mathcal{X}$ may be the space of pictures and $P(x)$ the

distribution therein describing cats. Now, the generator G must be able to generate new data samples $\tilde{x} \in \mathcal{X}$ and to control the distribution of these fake data $Q(\tilde{x})$. The way we build such a generator is by first choosing some prior distribution $Q_0(z)$ over a so called *latent space* $\mathcal{Z}$, usually a normal, and then defining G as the map

$$\begin{aligned} G:\ \mathcal{Z} &\to \mathcal{X}\,, \\ z &\to \tilde{x} = G(z)\,. \end{aligned} \tag{1.5}$$

The optimal generator, *i.e.* the one exactly reproducing the target distribution P, will be dubbed G^*, and will be the optimal point of a family of parametrized generators $G_{\boldsymbol{\theta}}$. Here $\boldsymbol{\theta}$ generally refers to any trainable set of parameters that we endow our model with. Usually, this parametrization is made by resorting to neural networks, and in that case $\boldsymbol{\theta}$ would embody both smooth parameters as the networks ones, and the discrete ones as the number and type of layers etc.
Secondly, we need a discriminator D. This ultimately has to be a binary classifier

$$D : X \to [0, 1]\,, \tag{1.6}$$

that associates with samples coming either from P or Q the probability of them belonging to the real distribution, that is $D(x) = p(R|x)$ is the conditional probability of labeling the input data x as *Real*. Again, we will rather work with a parametrized family of such probability measures, and we will refer to it as $D_{\boldsymbol{\lambda}}$, with $\boldsymbol{\lambda}$ symbolizing the set of parameters needed by the discriminator, which is most often realized as a neural network just as G.

Lastly, we need to define the adversarial game in such a way that D and G will self-supervise each other and accomplish training. The idea [20] is to make G and D play a *zero-sum game*, namely a game where each player's progress equals their foe's regress. In this way, we can appeal to Nash's game theory to be certain that the game will have a single fixed point, where $Q = P$ and G is able to perfectly fool the discriminator, which in turn will no longer distinguish false data from real data, merely constantly outputting $1/2$ in surrender.
There are many ways to set up the game, and we will only cover the most common one. The min-max game we are looking for can be stated as

$$\min_G \max_D S(G, D) = \min_G \max_D \left[E_{x\sim P}[\ln(D(x))] + E_{z\sim Q_0}[\ln(1 - D(G(z)))]\right]\,. \tag{1.7}$$

Here, $S(G, D)$ goes by the name of *score function* and is closely related to the binary cross entropy in Eq. (1.3). The game in Eq. (1.7) falls into Nash's game theory theorems hypothesis [21], thus, given that G and D are expressive enough, theoretically the generator can end up fooling the discriminator.

Once the problem has been laid down, it remains to find a fruitful way to solve it, namely to train our agents D and G. We tackle the min-max game as a turn based one. We begin by randomly initializing the parameters for both

agents, *i.e.* we pick a random starting point in the space of generators and discriminators. Then, we keep G fixed as we present D with a batch of samples coming from P and Q and evaluate $S(G, D)$. It is easy to understand why G has to be held fixed. Indeed, discriminator training tries to come up with the best way to distinguish real data from fake, put in another way, we want D to learn how to recognize the generator's flaws.If G were to change during this process, D could not ever focus on a particular flaw and would eventually have a hard time improving its strategy. Thus, we update D to maximize S, for example we could take some gradient ascent steps, but we do not look for full maximization, just for a slight improvement of D's performance. After D has improved its strategy, G does the same using the updated D as competitor. Again, we do not fully minimize S, and we keep D constant during its training, as we do not want G to aim at a moving target. This concludes one turn of the game, and we keep playing until convergence. This juggling back and forth between two competing agents is what makes GANs capable to solve very complex generative tasks [22, 23], and a scheme of this training procedure is found in Fig. 1.3.
As a closing remark, let us point out that GAN convergence is challenging to monitor and comprehend. Since G is randomly initialized, while D initially has an easy time distinguishing between fakes and actual objects, as the generator improves with practice, the discriminator's performance degrades. The discriminator has a 50% accuracy when the generator runs flawlessly. To make its forecast, the discriminator essentially flips a coin. The discriminator feedback diminishes over time, which complicates the GAN's overall ability to converge. The generator starts to train on garbage feedback when the discriminator has reached the point where its output is entirely random, which could cause the GAN's quality to collapse. For a GAN, convergence is often a fleeting, rather than stable, state.

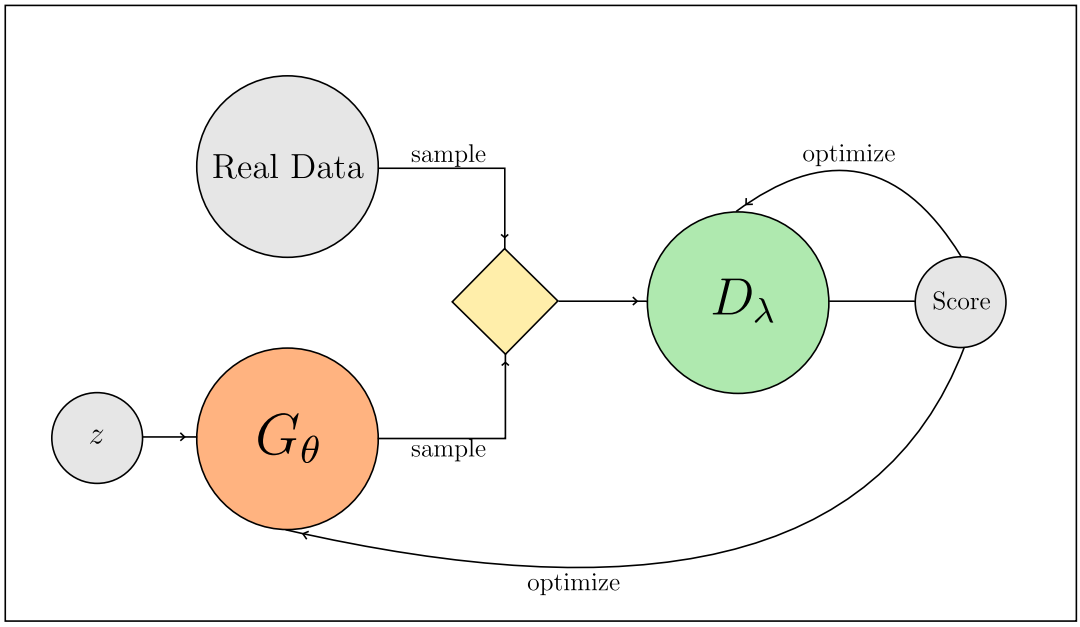


Figure 1.3: **The training pipeline of a classical GAN.** An input data point, or a batch of them, is sampled either from the real distribution or from the fake, synthetic, one. The discriminator processes the data and outputs its verdict, telling us with which probability it came from the real distribution. This probability is used as a feedback from either G and D to improve on their strategies.

Chapter 2

Quantum Computing

Having introduced the basics of the machine learning framework, we are one step away from getting to the core of this work, namely *quantum machine learning*. The step we need to take is arguably the hardest one: *quantum mechanics*. Using the properties of the microscopic world to design new computational routines, or to enhance classical existing ones, goes by the name of *quantum computation*, and giving the reader a crash-course on this field is the aim of this chapter.

In section 2.1 we will use a typical quantum computational circuit as an excuse to go through the defining postulates of Quantum Mechanics. Later, in section 2.2 we will see how the ideal description of quantum computing has to be adapted to describe real-world quantum processors. Section 2.3 closes the chapter by listing some important tools that will be used later in this work. The main references for this chapter are [3, 24, 25].

2.1 Ideal Quantum Computing Basic Elements

Once well understood, circuital quantum computing, *i.e.* the quantum computing paradigm where every computation is laid down as a sequence of simple logical operations arranged in a circuit, turns into a very diagrammatic and graphical field. We will take the backwards journey, starting from one such diagram, the *quantum circuit* in Fig. 2.1, and explaining its components one by one. In doing so, we will also introduce the postulates upon which quantum mechanics rests. As the name of the section suggests, all of the following applies on paper, while making actual quantum computations in the real world is much more of a mess, as we will discuss later in section 2.2.

2.1.1 Qubits

The first circuital element we need to address is the unit of quantum information, the *qubit*, one of them is singled out in component (a) of Fig. 2.1.

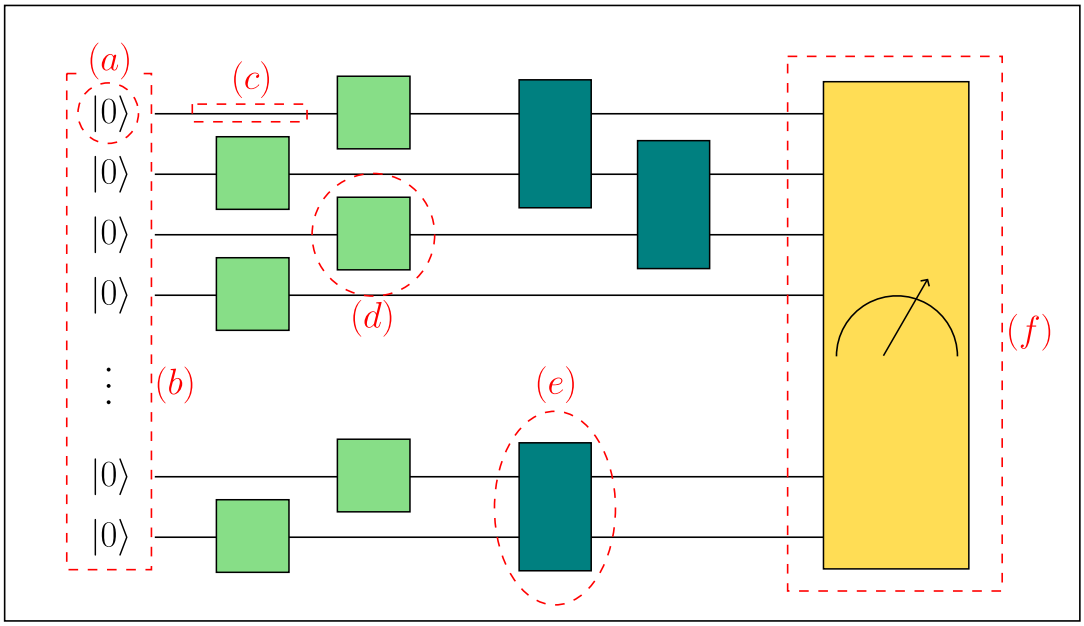


Figure 2.1: **The prototypical quantum circuit.** (a): a qubit; (b): the qubits register; (c) transmission wire (no evolution); (d): a single qubit gate; (e): entangling two qubits gate; (f): measurement apparatus.

Just like classical computation is built upon *bits*, units of information that can be in either one of two states usually labelled 0 and 1, quantum computing uses qubits for that scope. Qubits are simple quantum physical systems, but how do we describe the state they are in? The first postulate of Quantum Mechanics (QM) addresses this question: *the state of an isolated quantum mechanical system ψ lives in a Hilbert space $\mathcal{H}_\psi$*. Hilbert spaces are vector spaces endowed with a scalar product and a notion of distance following from it, and they can be defined over the real or complex field, the latter being the case for QM. Quantum computation always uses finite dimensional systems, so we can represent a qubit's quantum state as a vector $|\psi\rangle \in \mathcal{H}_\psi$. The symbol $|\cdot\rangle$ goes by the name of *ket*, and together with the *bra* $\langle\cdot|$, which is used for vectors in the dual space[1] $\mathcal{H}_\psi^*$ it defines Dirac's *braket* notation. Here a braket is the inner product of two quantum state vectors $|\psi\rangle, |\phi\rangle \to \langle\psi|\phi\rangle$. The braket inner product defines the states' norms as $\||\psi\rangle\|^2 = \langle\psi|\psi\rangle$. As in any vector space, we are free to choose a complete orthonormal basis $\{|e_k\rangle\}_{k=1}^{\dim(\mathcal{H})}$ and expand the quantum state $|\psi\rangle$ at hand over it

$$|\psi\rangle = \sum_{k=1}^{\dim(\mathcal{H})} \langle e_k|\psi\rangle\, |e_k\rangle \ . \tag{2.1}$$

The coefficients $\langle e_k|\psi\rangle$ are called the *amplitudes* of the quantum state $|\psi\rangle$. Notice that we have dropped the subscript $|\psi\rangle$, and that from now on we will

[1] The space of one-forms acting over $\mathcal{H}_\psi$

always refer to the Hilbert space of a quantum system simply as $\mathcal{H}$. As we are going to see better in a few steps, the probabilistic nature of quantum mechanics leads to the need of adding a constraint to the accessible region of $\mathcal{H}$, indeed physical states are those with unit norm $\langle\psi|\psi\rangle = 1$. Notice that this requirement also gives rise to a gauge freedom for quantum states, namely the two states $|\psi\rangle$ and $|\phi\rangle = e^{i\theta}|\psi\rangle$ contain the very same information about the quantum system they are describing.

With this in mind, a qubit can be defined as any quantum *two-level* system, that is, the simplest possible non-trivial quantum object. Two-level means that $\dim(\mathcal{H}) = 2$, and in analogy with their classical counterpart, the two orthonormal levels are dubbed $|0\rangle$ and $|1\rangle$. Thus, any single qubit state reads:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \qquad \alpha, \beta \in \mathbb{C}, \quad |\alpha|^2 + |\beta|^2 = 1\,. \tag{2.2}$$

The basis $\{|0\rangle, |1\rangle\}$ is usually called *computational basis*.

2.1.2 Computational Register

The set of qubits labeled by the index (b) of Fig. 2.1 represents the workspace of our quantum computer, and goes by the name of *computational register* or simply *qubit register*. The way quantum systems assemble in a composite one is described by yet another postulate. This states that given two systems Q_1, Q_2 with associated Hilbert spaces $\mathcal{H}_{Q_1}, \mathcal{H}_{Q_2}$, the global quantum state of the system $Q_1 \cup Q_2$ lives in the tensor product Hilbert space $\mathcal{H} = \mathcal{H}_{Q_1} \otimes \mathcal{H}_{Q_2}$. In general, given N quantum systems $\{Q_i\}_{i=1}^N$, the composite Hilbert space reads

$$\mathcal{H}_{\text{tot}} = \bigotimes_{i=1}^{N} \mathcal{H}_{Q_i}\ , \tag{2.3}$$

The dimension of a composite quantum system thus scales multiplicatively: $\dim(\mathcal{H}_{\text{tot}}) = \prod_{i=1}^N \dim(\mathcal{H}_{Q_i})$. As all that matters to us are qubits, and since a single qubit has an associated Hilbert space of size two, we get that a computational quantum register QR of size N accommodates an exponential number of basis states, namely $\dim(\mathcal{H}_{\text{QR}}) = 2^N$.
Consider now the simplest case, where just $N = 2$ qubits compose the register, Q_1 and Q_2. Calling $\{|0\rangle_1, |1\rangle_1\}$ and $\{|0\rangle_2, |1\rangle_2\}$, the computational bases of the two qubits, the resulting computational basis of $\mathcal{H}_{\text{QR}}$ is:

$$\{|00\rangle, |10\rangle, |01\rangle, |11\rangle\}\,, \tag{2.4}$$

Notice that, for the sake of simplicity, we dropped both the tensor product sign $|00\rangle = |0\rangle \otimes |0\rangle$, and the subscripts indexing the single qubits. This is usual in quantum computation, rather, the ordering of the binary digits in the register's ket is what keeps track of it, the first qubit being on the far right.

With two qubits we can start to study fancy states like:

$$|B\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}} \,. \tag{2.5}$$

This state is famously known as the *Bell state*, and what makes it relevant is that it is an *entangled* state. The state of a composite quantum system is said to be entangled when it cannot be described as a collection of ket states of its constituents $|\psi\rangle \neq |\psi_A\rangle \otimes |\psi_B\rangle$. On the other hand, when this is possible, and the system state can be written as a tensor product of single subsystems kets, we call the state *separable*.

2.1.3 Gates and Wires

Reading the quantum circuit in Fig. 2.1 from left to right we are reading it forward in time. That is, the circuit diagram represents a time sequence of operations acting on the qubit, or more of them as we will discuss in the following sections. That means that the piece of wire, and the boxes that appear respectively as marks (c), (d), (e) in the figure represent temporal evolution of the qubit's state $|\psi\rangle$. The way *isolated* quantum systems evolve in time is stated in the second postulate of quantum mechanics: given the state $|\psi_0\rangle$ of a quantum system at time $t = 0$, its evolved state at time t will be related to the initial one by the action of a unitary operator U_t as $|\psi_t\rangle = U_t |\psi_0\rangle$. The unitary operator is the solution of the famous *Schrödinger* equation

$$i\hbar \frac{d}{dt} |\psi\rangle = H |\psi\rangle \,, \tag{2.6}$$

and we recall that an operator U is unitary if and only if $UU^\dagger = U^\dagger U = I$. The operator H appearing in (2.6) is called the *Hamiltonian* of the system, and is nothing but the quantum analogue of the usual energy functional of classical mechanics. Indeed, states that have a well defined energy are eigenvectors of H, and the associated eigenvalues is their energy. However, let us stop for now, we will discuss about physical observables later when introducing the last component of any quantum circuit. When the Hamiltonian H is time-independent, Eq. (2.6) has a simple solution in terms of it

$$|\psi_t\rangle = U_t |\psi_0\rangle = e^{-itH} |\psi_0\rangle \,, \tag{2.7}$$

where we adopted the convention $\hbar = 1$, as is usual in theoretical quantum physics. In the following, we will not need the general solution of the Schrödinger equation for time-dependent Hamiltonians, thus we refer the interested reader to [24, 25] for the generalization of Eq. (2.7) to that case.

So, wires and boxes describe the evolution of the state of our information carrier, and in practice they allow us to control the information flow through the circuit. But what kind of evolution are we talking about? Wires are associated to null evolution. That is, their role is just to preserve the state

of the qubit they are linked to, and we can interpret them as imposing a null dynamics $H \sim 0$ onto the qubit.

Boxes instead implement actual operations on the qubit, and their role is to control the logical workflow of the algorithm at hand. They go by the name of *gates*, just as in classical computer circuits. The action of every quantum gate V can be expressed as in Eq. (2.7), more explicitly:

$$V = e^{-i\tau G}, \tag{2.8}$$

where G is called the gate's *generator*, and τ is the $gate - time$. We will give a list of common generators and gates at the end of this chapter. The gate time is usually what allows us to devise *parametric gates*. Indeed, by letting the qubit evolve under the dynamics imposed by the generator G for a tunable amount of time τ we can control the way the qubit's state steers in its Hilbert space. This lies at the heart of the most prominent class of quantum machine learning models, as we will see later in section 3.3.

The box (e) that we find along the circuit has two input wires and two output ones, it is the prototypical *two-qubit gate*. These gates, whose action has the very same structure as the single qubit ones that we just described, have the additional, and fundamental, effect of generating entanglement. That is, their generator corresponds to an *interaction* between the two physical systems realizing the qubits. Interaction here means that the generator hamiltonian G cannot be split into two separate hamiltonians $G = G_1 \otimes G_2$ acting separately on the two systems. This is always assumed, as otherwise two parallel single boxes would have been used in the circuit diagram of Fig. 2.1. It is easy to see how single qubit gates cannot introduce entanglement in the qubit register. Indeed, whenever we perform separate evolutions of two qubits via gates V_1, V_2, the global evolution of the two-qubit register reads $V = V_1 \otimes V_2$. Thus, if the register was in a separate state $|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle$, after the evolution entanglement will still be absent as $V|\psi\rangle = (V_1|\psi_1\rangle) \otimes (V_2|\psi_2\rangle)$. In section 2.3 we will show a simple circuit that generates entanglement, synthesizing the Bell state in Eq. (2.5).

2.1.4 Measurement

At the end of any quantum computational routine a *measuring apparatus*, labeled as (f) in Fig. 2.1, awaits. Indeed, without it no information could ever be retrieved from a quantum system. While all of the previous postulates implied a deterministic description and dynamics of the ket states describing the state of a quantum system, the measurement postulate, the one describing how an observer can actually gather the information contained in such kets, introduces the stochasticity that quantum mechanics is known for.

A quantum mechanical measurement can be described as follows [24]. Consider a quantum system Q in a ket state $|\psi\rangle$ that we want to probe for a physical observable A. In quantum mechanics, every observable is described

by an hermitian operator $\hat{A} = \hat{A}^\dagger$ acting on the Hilbert space of the system at hand, $\mathcal{H}_Q$. Then, the possible outcomes of a measurement of A are all and only the eigenvalues $\{a_k\}$ (again, we are assuming a finite-size Hilbert space $d = \dim(\mathcal{H})$) and they can occur with probability

$$p_k = |\langle a_k|\psi\rangle|^2 \ . \tag{2.9}$$

where $|a_k\rangle$ is the eigenvector of $\hat{A}$ associated with the corresponding eigenvalue a_k. Eq. (2.9) is known as the *Born rule*. This type of measurement is called *projective measurement*, as it revolves around the projection of $|\psi\rangle$ on the eigenspace of the measurement outcome via the projector $P_k = |a_k\rangle\langle a_k|$. After measuring a_k, the state of the system drastically changes into

$$|\psi\rangle \to \frac{P_k\,|\psi\rangle}{\sqrt{\langle\psi|\,P_k\,|\psi\rangle}}\,, \tag{2.10}$$

and this phenomenon is known as *collapse*. The average value of the observable A over the state $|\psi\rangle$ can be statistically defined in the usual way, and this leads to a nice and compact quantum mechanical expression

$$\langle A\rangle_\psi = \sum_k p_k a_k = \sum_k a_k \langle\psi|a_k\rangle \langle a_k|\psi\rangle = \langle\psi| \sum_k a_k P_k |\psi\rangle = \langle\psi|\,\hat{A}\,|\psi\rangle \ . \tag{2.11}$$

where we used the spectral decomposition of $\hat{A}$.

The stochastic nature of the measurement outcomes is what really defines quantum mechanics, and it also explains why ket vector states must have unit norm. Indeed, recalling that the set of eigenvectors of an operator satisfies the completeness property $\sum_{k=1}^{d} |a_k\rangle\langle a_k| = \mathbb{1}$, and that by definition of probability $\sum_k p_k = 1$, we have

$$1 = \sum_{k=1}^{M} p_k = \sum_{k=1}^{d} |\langle a_k|\psi\rangle|^2 = \sum_{k=1}^{d} \langle\psi|a_k\rangle\langle a_k|\psi\rangle = \langle\psi|\psi\rangle \ , \tag{2.12}$$

Thus, after having steered the ket state of the quantum register $|QR\rangle$ through its Hilbert space by the means of the quantum gates, at the end of the circuit we perform measurements, reconstructing the outcomes probabilities from their frequencies, and constructing expectation values with those. Last but not least, let us comment that the expectation values in Eq. (2.11) can only be computed by using the average of each single measurement outcome as their proxy. In the quantum computing community, each of the single measurements is called a *shot*, and the number of shots one uses to compute expectation values is decided by the precision one needs to achieve.

2.2 How to deal with Noise

When a quantum system is not isolated, but rather is a subsystem of some larger one, the description of its state via ket vectors may break down. As we

have seen with the example of the Bell state in Eq. (2.5), whenever entanglement is present in the system as a whole its constituents cannot be associated with a well defined ket state. This property, which has no classical analogue, is a fundamental property of the microscopic world, and not a flaw of our description of it. Still, we might be interested in the description of a subsystem alone, as it might be the only thing we have access to. This is what always happens in real quantum computing, since we still do not have the technology to completely isolate the qubit register from its surroundings and we do not even know how these two interact. Sure, the universe as a whole is an isolated quantum system, and we could try to start from its ket and derive what happens to the qubit register...
Fortunately, quantum mechanics has developed an elegant way to deal with this: the *density matrix* formalism, and the general *open quantum system* framework.

2.2.1 Density matrix formalism

Let us consider an isolated quantum system described by a ket $|\psi\rangle$. We call such a state a *pure state*. Now, from $|\psi\rangle$ we can build the projector onto it

$$\rho_\psi = |\psi\rangle \langle\psi| \,, \tag{2.13}$$

and call it the *density matrix* associated to the pure state $|\psi\rangle$. In fact, we can completely reformulate the description of isolated quantum systems in terms of ρ_ψ, that from now on for the sake of brevity will be simply dubbed as ρ. The evolution and measurement postulates can be reformulated as

- Given the unitary evolution operator defined in (2.7), the density operator evolves with:
$$\rho(t) = \hat{U}(t)\rho(0)\hat{U}(t)^\dagger \,. \tag{2.14}$$

- Referring to the observable A defined around Eq. (2.9), the probability p_k of obtaining the outcome a_k, associated with the projective measurement $P_k = |a_k\rangle \langle a_k|$ is:
$$p_k = \mathrm{Tr}(\rho P_k) \,. \tag{2.15}$$
and the expectation value of A thus reads
$$\langle A\rangle = \mathrm{Tr}(\rho \hat{A}) \,. \tag{2.16}$$
Moreover, the collapse of the state after the outcome a_k has been observed implies
$$\rho_\psi \xrightarrow{a_k} \frac{P_k \rho P_k}{\mathrm{Tr}(P_k \rho P_k)} \,. \tag{2.17}$$

More than this, we can avoid using kets altogether and define as viable quantum states the operators $\rho \in \mathcal{B}(\mathcal{H})$, belonging to the space of bounded linear operators acting over the Hilbert space $\mathcal{H}$ associated with the quantum system that satisfy

- Hermiticy, $\rho^\dagger = \rho$;
- Positivity $\langle\psi|\,\rho\,|\psi\rangle \geqslant 0 \quad \forall\ |\psi\rangle \in \mathcal{H}$;
- $\mathrm{Tr}(\rho) = 1$.

These properties are indeed satisfied by pure states $\rho = |\psi\rangle\langle\psi|$, but those also have an additional property that we can relax, allowing for the description of more general states, namely that of being projectors: $\rho^2 = \rho \to \mathrm{Tr}[\rho^2] = 1$. The states that are not described by projectors are not pure, and rather go by the name of *mixed states*. The quantity $P(\rho) = \mathrm{Tr}[\rho^2] \in [1/d, 1]$, where $d = \dim(\mathcal{H})$, is called the *purity* of the quantum state, and it is the figure of merit of states being pure ($P(\rho) = 1$) or mixed ($P(\rho) < 1$).

Mixed states arise either from entanglement, as hinted before and as we shall see in a moment, or from the absence of complete knowledge of the state of an isolated system. Indeed, all we might know about an isolated quantum system is that it could be in either one of an ensemble of states $\{|\psi_k\rangle\}$ with probabilities p_k. This is the case when we are told that the system $|\psi\rangle$ has been measured along some observable which we know the eigenvectors $\{|\psi_k\rangle\}$ of, but we forgot to write down the measurement outcome and so all we know is that now the system has collapsed in one of the $|\psi_k\rangle$ with probability $p_k = \|\langle\psi_k|\psi\rangle\|^2$. We can then include this classical uncertainty into the description of the quantum system by associating to the ensemble $\{(p_k, |\psi_k\rangle)\}$ the density matrix

$$\rho_{\text{ensemble}} = \sum_k p_k\,|\psi_k\rangle\langle\psi_k|\ . \tag{2.18}$$

However, there is actually no need to carry on with the distinction between entanglement-born mixed states and ensemble ones, as they are treated identically by quantum mechanics.

Let us finally come to the characterization of entangled quantum subsystems. Consider an isolated bipartite quantum system Ψ consisting of two subsystems Ψ_A and Ψ_B. Now, say that we can only access Ψ_A and that the system as a whole is in the pure state $|\psi\rangle \in \mathcal{H}_{\Psi_A} \otimes \mathcal{H}_{\Psi_B}$. The state of the subsystem Ψ_A will then be described by

$$\rho_A = \mathrm{Tr}_B(\rho)\,, \tag{2.19}$$

where $\rho = |\psi\rangle\langle\psi|$ is the pure density matrix associated to $|\psi\rangle$, and we used A, B in place of Ψ_A, Ψ_B to shorten up the notation. The state ρ_A is called the

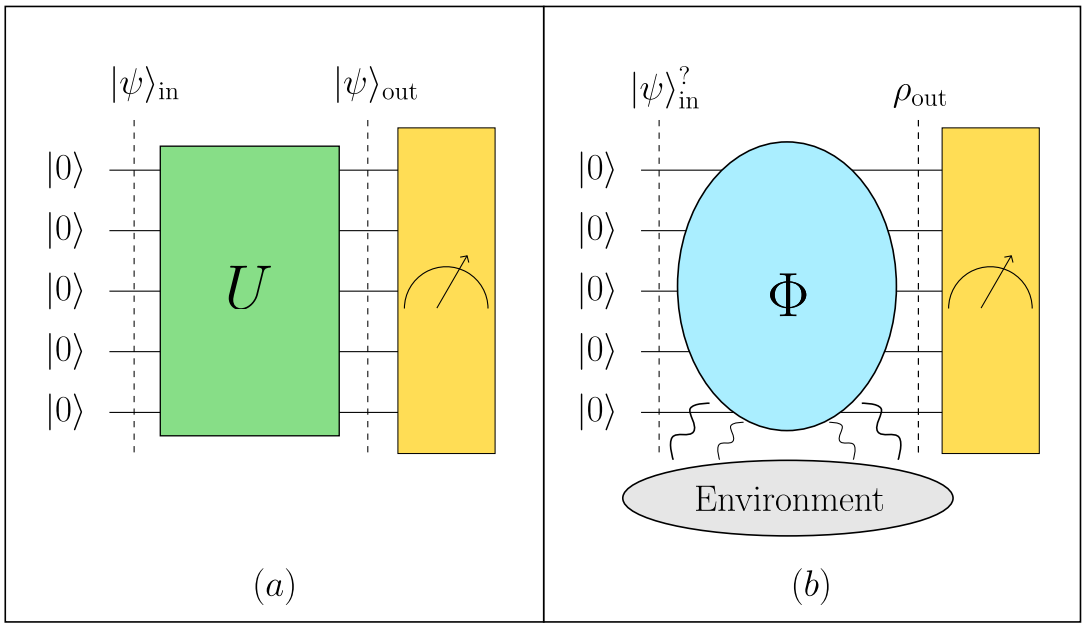


Figure 2.2: **Ideal vs Real quantum computation.** (a): the ideal circuital scheme of quantum computation. (b): what happens in a real-world quantum computation where noise is unavoidable.

reduced density matrix of subsystem Ψ_A, and the operator Tr_B is the *partial trace* operator, defined as:

$$\text{Tr}_B(\rho) = \sum_b (\mathbb{1}_A \otimes \langle b|)\rho(\mathbb{1}_A \otimes |b\rangle)\,. \tag{2.20}$$

with $\mathbb{1}_A$ the identity operator on the Hilbert space $\mathcal{H}_{\Psi_A}$. Basically, we are *tracing out* all of the degrees of freedom associated with the inaccessible subsystem Ψ_B and encoded in the orthonormal basis $\{|b\rangle\}$. The reduced density matrix ρ_A contains all the information we need to compute expectation values of observables with support on Ψ_A only, indeed any such observable O_A can be extended to the whole Hilbert space $\mathcal{H}$ as $O_A \to O_A \otimes I_B$, and plugging this in the expectation value (2.16) we get

$$\langle O_A \rangle = \text{Tr}[\rho(O_A \otimes I_B)] = \text{Tr}_A[\rho_A O_A]\,. \tag{2.21}$$

2.2.2 Quantum channels

We have seen that the unitary pure states evolution can be easily reformulated in terms of their density matrix (Eq. (2.14)), and this simple extension still works for ensemble mixed states, as long as the system they describe is isolated, and their mixedness arises only from our classical uncertainty about it. The evolution of entangled subsystem, whose mixedness emerges from the impossibility of treating them separately, is a different story. Let us keep analysing the simple bipartite system $\Psi_A \cup \Psi_B$. As it evolves unitarily for a

time t with U_t, the global (say pure) state ρ of Ψ changes into $\rho_t = U_t \rho U_t^\dagger$. Consequently, the reduced density matrix ρ_A of Ψ_A evolves into

$$\rho_A^{(t)} = \mathrm{Tr}_B\left[U_t \rho U_t^\dagger\right]. \tag{2.22}$$

It is easy to guess that the evolution of ρ_A cannot be recast as $\rho_A^{(t)} = \tilde{U}\rho_A^{(0)}\tilde{U}^\dagger$ for some unitary $\tilde{U}$, and this leads to the question: how can we characterize the evolution of a system when we do not have access to the full dynamics of the universe? In order to describe the most general processes that can evolve a quantum system, the formalism of *quantum maps*, or quantum operations, has been developed [26], see Fig. 2.2. Quantum maps thus are functions Φ from the space of density matrices $\mathcal{B}(\mathcal{H}_{\mathrm{in}})$ on an input Hilbert space $\mathcal{H}_{\mathrm{in}}$ to that on an output one $\mathcal{H}_{\mathrm{out}}$.

$$\rho_{\mathrm{in}} \in \mathcal{B}(\mathcal{H}_{\mathrm{in}}) \to \rho_{\mathrm{out}} = \Phi(\rho_{\mathrm{in}}) \in \mathcal{B}(\mathcal{H}_{\mathrm{out}}). \tag{2.23}$$

Notice that now we allow the output Hilbert space to be different from the starting one. This can happen, for example, when part of the system we are dealing with is discarded, or when adding an external auxiliary system is needed. What kind of properties do these maps need to have? The only requisite we need to impose onto them is that they map valid quantum states into valid quantum states. Since density matrices have to comply with the properties listed below Eq. (2.17), we get that quantum maps Φ must be:

1. *Linear*: they must transform any superposition of input states into the superposition of the associated output ones, namely:

$$\rho_{\mathrm{in}} = \sum_k p_k \rho_k \to \rho_{\mathrm{out}} = \sum_k p_k \Phi(\rho_k) \tag{2.24}$$

   As a side note, notice how a superposition of density matrices must be *convex*, that is the coefficients p_k must be a probability distribution ($p_k \geqslant 0$, $\sum p_k = 1$), or else the superposition's trace would not be one.

2. *Trace-Preserving*: the normalization of the input density matrix, *i.e.* $\mathrm{Tr}[\rho_{\mathrm{in}}] = 1$ must be preserved $\mathrm{Tr}[\Phi(\rho_{\mathrm{in}})] = 1$

3. *Completely-Positive*: since a valid density matrix is a positive operator, the same must hold for the output of the quantum map. Particularly, a completely positive map Φ is such that $\Phi(A) \geqslant 0 \,\forall\, A \geqslant 0$ and $(\mathbb{1}_B \otimes \Phi(A)) \geqslant 0$ whenever A is embedded in an enlarged space, B being the extra system.

From these properties stems the usual way that quantum maps are called: *CPTP* (Completely Positive and Trace Preserving) maps.[2] Moreover, when

[2] CPTP maps are not the only possible quantum operations. The trace condition can be relaxed to be *non-increasing* $\mathrm{Tr}[\Phi(\rho_{\mathrm{in}})] \leqslant 1$, and correlations between a system and its environment may lead to a non-CP evolution [27, 28]. We will never need to consider these cases.

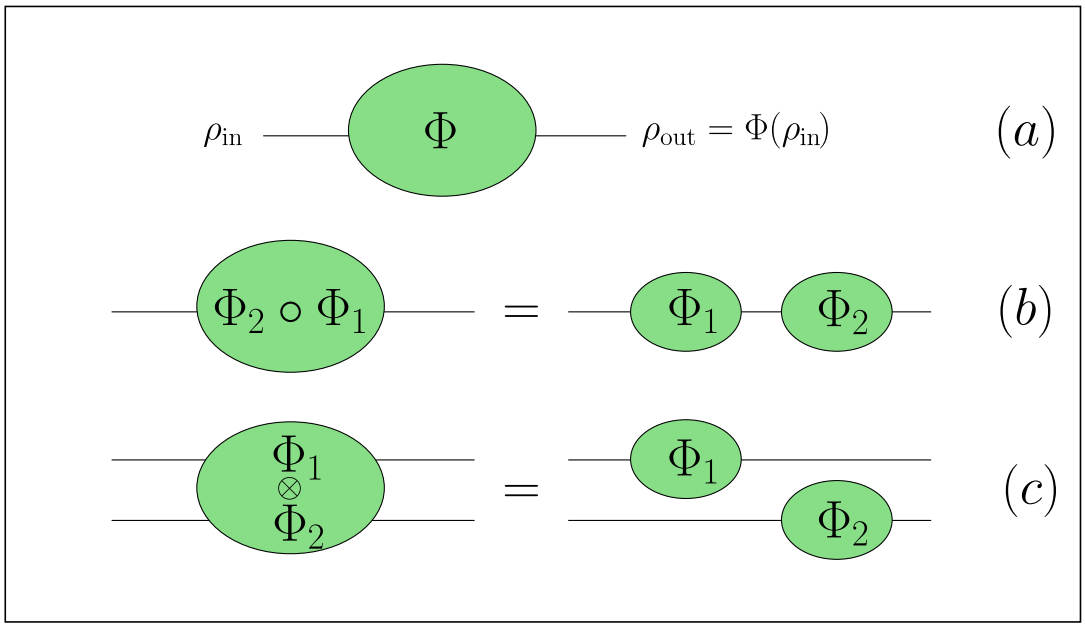


Figure 2.3: **Diagrammatic representation of quantum channels and their compositions.** (a): a quantum channel Φ mapping an input state $\rho_{\text{in}} \to \rho_{\text{out}}$; (b): two channels $\Phi_{1,2}$ being composed in series; (c): the same channels composed in parallel.

the output Hilbert space is the same as the input one, they are usually called quantum *channels*. We will use channel and map interchangeably, and we will refrain from specifying every time input and output Hilbert spaces, as it will be clear from the context. Notice that the unitary evolution $\rho_t = U_t \rho U_t^\dagger$ is just a particular kind of channel $\rho_t = \mathcal{U}_t(\rho)$, namely a *unitary channel*.

Given two channels Φ and Λ, it is straightforward to check that any convex combination of them $\epsilon = p\Phi + (1-p)\Lambda$, $p \in [0,1]$, is again a CPTP map. This is the *convexity* property of quantum channels. Aside from convex combinations, channels might be composed in series (*concatenation*) or in parallel (*tensor product*), where we borrowed terminology from classical electrical circuits. Concatenation is the application of the two channels one after the other $\epsilon = \Lambda \circ \Phi$. Two channels are said to be unitarily equivalent if their action is related by changes of basis in the input and output spaces, namely if there exist two unitary channels $\mathcal{U}, \mathcal{U}'$ s.t. $\Lambda = \mathcal{U}' \circ \Phi \circ \mathcal{U}$. Composition in parallel is done by letting the two channels act on different subsystems at the same time, *i.e.* $\epsilon = \Phi \otimes \Lambda$, an example of this being the *extension* of a channel onto a bigger space $\Phi_{\text{ext}} = \Phi \otimes I$, but also the simultaneous evolution under different dynamics of two components of a quantum systems. Both of these composition rules are sketched in Fig. 2.3.

CPTP requirements can be recast in an alternative condition for quantum maps, namely that they can be expressed as sum of *conjugation* by linear

operators:

$$\Phi(\rho) = \sum_{k=1}^{R} K_k \rho K_k^\dagger \,, \qquad \text{with: } \sum_k K_k^\dagger K_k = I \,. \tag{2.25}$$

The operators $\{K_k\}$ are called the *Kraus operators*, and the representation (2.25) is called the *Kraus representation* of the channel Φ. The number R of Kraus operators is called the *Kraus rank*. Notice that this decomposition is not unique, as two sets of Kraus operators $\{K_k\}, \{K'_k\}$ yield the same map if they are related by a unitary rotation $K_k = U_{kj} K'_j$, indeed

$$\sum_k K_k \rho K_k^\dagger = \sum_k U_{kj} K'_j \rho K_i'^\dagger U_{ik}^\dagger = \sum_k U_{ik}^\dagger U_{kj} K'_j \rho K_i'^\dagger = \sum_i K'_i \rho K_i'^\dagger \,. \tag{2.26}$$

Nonetheless, it can be shown that there is a upper bound to the Kraus rank of a quantum channel $R \leqslant d^2$, where $d = \dim(\mathcal{H})$ is the dimension of the input Hilbert space.

Another, equivalent, representation for quantum maps is the *Stinespring representation* [26]. This states that it is always possible to find a suitable, albeit fictional, *environment* E, in a pure state $|\omega\rangle$ non-entangled with the system state ρ such that the global state of the system Ψ plus the environment E reads $\rho \otimes |\omega\rangle\langle\omega|$, so that:

$$\Phi(\rho) = \mathrm{Tr}_E \left[U_{\Psi E} (\rho \otimes |\omega\rangle\langle\omega|) U_{\Psi E}^\dagger \right] \,, \tag{2.27}$$

for some unitary coupling interaction $U_{\Psi E}$. As for the Kraus representation (2.25), Stinespring's representation (2.27) is not unique, its freedom lying in the choice of the initial environment state $|\omega\rangle$. The equivalence between Stinespring and Kraus representation can be readily understood performing the partial trace in (2.27), indeed, choosing $\{|e_k\rangle\}$ as an orthonormal basis for the environment E we can identify the Kraus operators K_k as $K_k = \langle e_k| U_{\Psi E} |\omega\rangle$.

Lastly, there exists yet another way to characterize a quantum map: the *Choi-Jamiolkowski* (CJ) isomorphism [26]. This allows to associate with every map Φ acting on a system Ψ a unique density matrix J^Φ of an extended system ΨA, where A is an auxiliary quantum system with the same size d as Ψ if Φ maps Ψ onto itself, or with size equal to that of the output system. In particular, the CJ state is defined as

$$J^\Phi = (\Phi \otimes \mathbb{1}_A)(|\Omega\rangle\langle\Omega|) \,, \tag{2.28}$$

where $|\Omega\rangle = (1/\sqrt{d}) \sum_i |e_i\rangle_\Psi \otimes |e_i\rangle_A$ is a maximally entangled state of the composite system ΨA, with $\{|e_i\rangle_{\Psi/A}\}$ being orthonormal bases of Ψ and A. The CJ state J^Φ holds all of the information needed to specify the action of the map Φ on any state, as one can see that in it are encoded the results of applying Φ to all the elements of the canonical basis of density matrices. Particularly, we can express the action of Φ on any quantum state ρ via the CJ state as [29]

$$\Phi(\rho) = \mathrm{Tr}_\Psi [J^\Phi (\rho^\top \otimes \mathbb{1}_{\dim(\mathcal{H}^{\mathrm{out}})})] \,. \tag{2.29}$$

2.2.3 Generalized Measurements

We have seen in section 2.1.4 that quantum measurements can be described in terms of sets of projectors $\{P_a\}$ onto the eigenspaces associated with the possible outcomes $\{a\}$ of an observable A. The open quantum systems framework admits a generalization of this projective measurement scheme. Indeed, consider yet again a bipartite quantum system $\Psi_A \otimes \Psi_B$ whose second component Ψ_B we do not have access to, or we do not care about. If a global projective measurement $\{P_k\}$ is performed on the whole system, assuming its state to be the unentangled state $\rho = \rho_A \otimes \rho_B$, the probability of observing the outcome k reads

$$p(k) = \mathrm{Tr}\left[P_k \rho\right] = \mathrm{Tr}_A\left[\left[\mathrm{Tr}_B\, P_k \rho_B\right] \rho_A\right] = \mathrm{Tr}_A\left[E_k \rho_A\right] , \tag{2.30}$$

where the operators $\{E_k\}$ define a *Positive Operator valued Measurement* (POVM). The elements $\{E_k\}$ of a POVM satisfy $\sum_k E_k = I$, as follows from the probabilities $p(k)$ summing up to one in (2.30). In general a POVM is defined by a set of positive semi-definite hermitian operators $\{E_k\}$, called *effects*, summing up to I. Each of them is associated with an outcome, and the probability of said outcome is given by $\mathrm{Tr}[E_k \rho]$, where ρ is the state of the system being measured. In general, the number of effects in a POVM might be larger than the size of the Hilbert space they act on, as opposed to what happens with a projective measurement. Interestingly, the outcome associated with an effect might actually be undefined. This is at the core of the so called *unambiguous* state discrimination protocol [30]. In quantum computing, implementing a POVM directly in a quantum circuit is a hard task. Most architectures only allow for rather simple projective measurements along the computational basis defined around Eq. (2.2). However, thanks to *Naimark's dilation theorem* [24], we can physically realize a POVM by performing a projective measurement on an extended Hilbert space. Naimark's theorem, in its simplest form, *i.e.* that of finite dimensional quantum systems, states that a POVM $\{E_k\}$ acting on a Hilbert space $\mathcal{H}$ of dimension d, can be realized through a projective measurement $\{P_k\}$ on an Hilbert space $\mathcal{H}'$ of size d' related to the original one via an isometry $V : \mathcal{H} \to \mathcal{H}'$. In practice, one realises a POVM by adding an auxiliary system to the quantum register, entangling the two together via some unitary circuit U and then performing a projective measurement P_k on the auxiliary system alone. This yields the POVM:

$$E_k = \mathrm{Tr}_a\left[(\mathbb{1} \otimes |0\rangle_a \langle 0|_a)(U(\mathbb{1} \otimes P_k^a)U^\dagger)\right] , \tag{2.31}$$

where we labeled the auxiliary system as a.

2.3 Compendium of useful notions

In this final section we will list the most used quantum gates, with their matrix representation and action. Before doing this however, we are going to

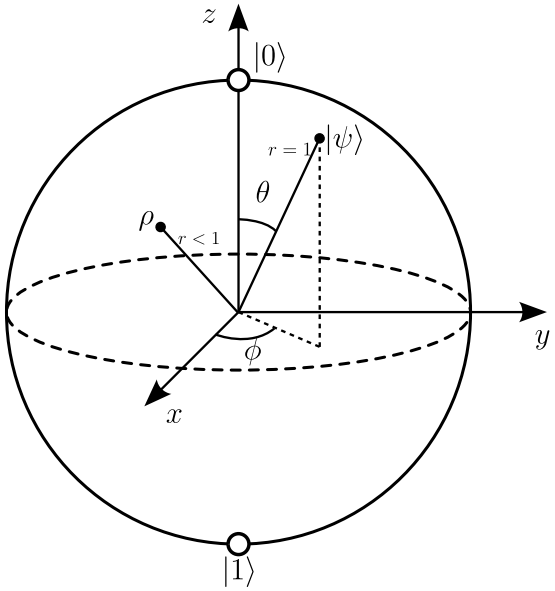


Figure 2.4: **The Bloch sphere representation of the Hilbert space of a single qubit.** A pure state $|\psi\rangle$ lives on the surface of the sphere ($r = 1$), whereas a mixed one ρ is found inside the ball ($r < 1$).

cover a helpful representation of the Hilbert space $\mathbb{C}^2$ where single qubit states live.

2.3.1 The Bloch Sphere

The Bloch sphere [31, 32] is a geometric representation of the state space of a single qubit, the basic unit of quantum information. It is a convenient way to visualize and understand the behavior of quantum states and operations in quantum computing and quantum information theory. The Bloch sphere is a unit sphere in three-dimensional space, with pure qubit states being represented as points on the *surface* of the sphere. The state of a qubit can be represented by a two-dimensional complex vector, as we have seen in Eq. (2.2), with the Bloch sphere providing a convenient way to visualize this vector in three-dimensional space. Indeed, we can express any qubit state as

$$|\psi\rangle = \cos(\theta/2)\,|0\rangle + e^{i\phi}\sin(\theta/2)\,|1\rangle\ , \tag{2.32}$$

where $\{|0\rangle, |1\rangle\}$ is the computational basis we introduced at the end of section 2.1.1, and its elements correspond to the north and south poles of the sphere, respectively. The coordinates (θ, ϕ) are just the usual spherical coordinates of $\mathbb{R}^3$, while the radial one is fixed to $r = 1$ by the quantum mechanical constraint $\||\psi\rangle\| = 1$. In figure 2.4 we show the Bloch sphere and how the gates that we are going to review steer the state $|\psi\rangle$ on it.

Mixed single qubit states populate the interior of the sphere instead. Indeed

any mixed single qubit state ρ can be decomposed as

$$\rho = \frac{1}{2}(\mathbb{1} + \boldsymbol{r} \cdot \boldsymbol{\sigma}) \,, \quad s.t. \; \|\boldsymbol{r}\| < 1 \,, \tag{2.33}$$

where $\boldsymbol{\sigma} = (X, Y, Z)$ are the three Pauli operators and $\boldsymbol{r}$ is the coordinate of ρ inside the Bloch sphere. It is easy to check that pure states (2.32) can be expressed in the same way with the additional constraint $\|\boldsymbol{r}\| = 1$.

2.3.2 Common gates

In what follows we are going to list common single and two-qubit gates. Let us start form the single qubit ones.

- **Pauli Operations**: the three Pauli matrices, usually dubbed X, Y, Z or $\sigma_x, \sigma_y, \sigma_z$, being unitaries can function as single qubit quantum gates. Indeed, they are constantly used in any quantum computational routine. Their matrix representation over the single qubit Hilbert space $\mathbb{C}^2$, w.r.t. the computational basis reads

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{2.34}$$

- **Hadamard Gate**: The Hadamard gate acts via

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}. \tag{2.35}$$

  The action of the Hadamard gate on a qubit is to rotate it by an angle of $\pi/2$ around the x-axis of the Bloch sphere. What makes the Hadamard gate one of the most useful quantum gates is the fact that, when applied to the computational basis states, it outputs an homogeneous superposition of those, *i.e.* $H|0\rangle = 1/\sqrt{2}(|0\rangle + |1\rangle)$ and $H|0\rangle = 1/\sqrt{2}(|0\rangle - |1\rangle)$. These states are the eigenstates of X, usually they are dubbed respectively as $|+\rangle$ and $|-\rangle$, and they are a common choice for the initialization of the quantum register.

- **Rotation Gate**: the last single qubit gate that we will review is the most general one. It is a parametric operation, and it will have an important role in the rest of this manuscript. The rotation gate $R(\alpha, \beta, \gamma)$ reads

$$R(\alpha, \beta, \gamma) = \begin{pmatrix} \cos(\alpha/2) & -e^{i\gamma} \sin(\alpha/2) \\ e^{i\beta} \sin(\alpha/2) & e^{i(\beta+\gamma)} \cos(\alpha/2) \end{pmatrix}. \tag{2.36}$$

  This gate implements the most general unitary evolution of the state of a qubit, thus it can map between any two points of the Bloch Sphere. It can be decomposed via the *Euler angles* in a sequence of rotations around two axes of the sphere, for example as $R(\alpha, \beta, \gamma) = R_z(\gamma) R_y(\alpha) R_z(\beta)$. Here, rotations around a fixed axis with versor $\boldsymbol{n}$ read $R_{\boldsymbol{n}}(\theta) = \cos(\theta/2)\,\mathbb{1} + i \sin(\theta/2)\,\boldsymbol{n} \cdot \boldsymbol{\sigma}$.

Let us move on to two qubits entangling gates

- **CNOT**: The Controlled NOT operation (CNOT) is probably the most used entangling operation. It flips the computational basis state of the target qubit whenever the control one is found in state $|0\rangle$. Its matrix representation over such basis is

$$CNOT = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \tag{2.37}$$

  We can understand its utility by realizing that by only applying H and it we can synthesize the maximally entangled state in eq (2.5). Indeed, starting from the $|00\rangle$ computational basis state of a two-qubits register, if we first apply H on the first (rightmost) one we get $H|00\rangle = (1/\sqrt{2})(|00\rangle + |01\rangle)$. Then, if we apply a $CNOT$ with the control qubit being the one we acted on with H we arrive at the bell state $CNOT\, H|00\rangle = (1/\sqrt{2})(|00\rangle + |11\rangle)$.

- **SWAP**: the SWAP operator acts by swapping the states of the two targeted qubits, namely $\text{SWAP}|\psi\rangle \otimes |\phi\rangle = |\phi\rangle \otimes |\psi\rangle$. Its matrix representation is

$$\text{SWAP} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{2.38}$$

  It is needed in the NISQ era because of the limited connectivity of current quantum processors. Indeed, if one needs to entangle two physically distant qubits in the register, one first has to move their states close with a ladder of SWAPs, and then apply the entangling operation.

Chapter 3

Quantum Machine Learning

This chapter introduces the framework of this thesis: *Quantum Machine Learning* (QML). By incorporating the concepts and results of machine learning that we quickly covered in chapter 1 into the field of quantum computation that we just finished overviewing in the last chapter, QML tries to get the best of both worlds. The power of quantum computing can lead to exponential increase in performance for classical ML tasks, and at the same time the advanced learning algorithms can help experimental quantum computing to deal with nowadays NISQ limitations. Here we will set the stage of quantum machine learning, trying to build upon our introduction of classical ML with the aim of introducing the basic tools that we will use in the following chapters. The main source of this chapter are [3] and references therein.
After quickly discussing strategies to encode classical data into quantum computers in Section 3.2, the main body of this chapter is devoted to introducing quantum learning models (Sec. 3.3) and how to train them (Sec. 3.4).

3.1 Going Quantum

Why going quantum? How could resorting to quantum computational techniques benefit the ML community? How could quantum physics take advantage of machine learning tools? In brief, what is the *Quantum Machine Learning* program? On one hand, the idea that quantum computing could level up classical machine learning is almost natural. There is a direct relation between what machine learning models can efficiently generate and process, namely ML is very good at finding patterns in data that can be as easily synthesized. At the same time, it is well known [33] that classical devices struggle to simulate even small quantum systems, and that quantum data distribution can hardly be tackled with said machines. In turn, quantum devices come with the innate ability to generate and handle such complex data patterns. This leads to the belief that quantum computers will overcome their classical counterparts for the most difficult tasks. This belief is not just faith, obviously, technically speaking quantum algorithms come with a list of provable *speedups*, meaning

that they can be proved to outperform the best known classical algorithms. This outperforming is usually weighted in terms of computational time, but other resources might be involved in the process [3]. Nonetheless, when a quantum algorithm is *exponentially* better than its classical competitor we talk about *quantum supremacy*, as opposed to the term *quantum advantage*, that is instead used for power law separation. Notice that it may be that the best possible classical algorithm for a given task is simply unknown, so from a theoretical computer science perspective the whole *supremacy* discussion might be futile. An example of this is the famous algorithm for integer factorization devised by Shor [34]. It promises to exponentially outperform the classical approach to the problem, but it is still possible that an even better classical algorithm will be found. Despite this, with quantum processors finally becoming available for experiments, solid statistical evidence of scaling advantage, even if only over a finite range of problem sizes, can be (and is being [35, 36]) acquired. On the other hand, ML techniques have taken over a whole bunch of physics fields, particularly where a lot of experimental data must be analyzed, as those collected at the L.H.C. experiment in Geneva [37].

Thus, merging the fields of machine learning and quantum computing just feels natural, and it is no surprise that attempts at combining them have been around since the beginning of quantum computing in the 1980s. The term *quantum machine learning* started to be used around 2013 [38], and interest in the topic began to increase significantly in 2014 when the first book entirely dedicated to it came out [39], leading to a growing body of literature and the formation of a community around the field. In recent years, quantum machine learning has become a well-established sub-discipline of quantum computing research, with a strong presence in industry, with big names such as IBM and Google leading the private efforts, and various open-source software frameworks [40–42] available.

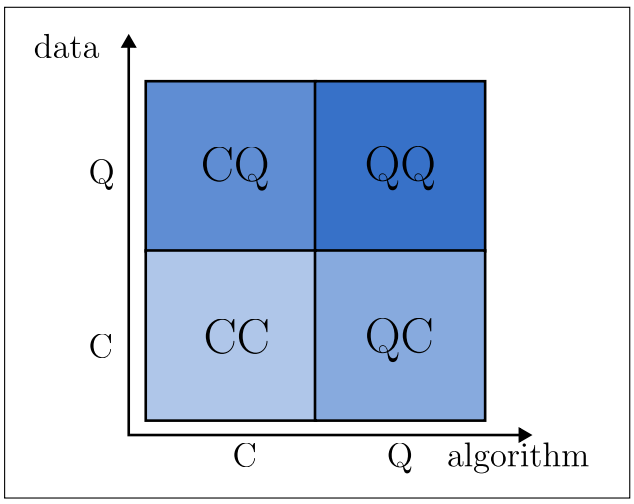


Figure 3.1: **The four sectors of QML.**

As we have seen in the introductory chapter on classical machine learning, learning frameworks have three main ingredients: data, models and optimization strategies. Depending on which of those we turn quantum, we can define the QML diagram shown in Fig. 3.1. The division usually only looks at data and processing device, both of which can be either classical (C) or quantum (Q). One would say that the CC sector does not belong to quantum machine learning, but in the QML community, this sector studies all of the classical machine learning improvements and new methods that are inspired or borrowed from quantum physics. The problem of simulability of large quantum systems [33] has indeed fueled a lot of computational research on the quantum side, leading to the establishment of powerful classical methods such as *tensor networks* [43], that can, and are, be adapted to tackle classical ML tasks [44]. QC refers to the collection of classical machine learning techniques that are used to benefit quantum computing. Among these, discrimination strategies for quantum states [45], error correction [46], prediction of observables expectation values based on a limited number of measurements [47], or even *neural-network quantum states* [48]. We then come to the CQ corner of QML. This arguably is the real QML domain, or at least the one that has the most commercial power. Indeed, as commented before, using quantum processor to treat classical data, and thus for classical applications such as those governing our daily life, has the potential to revolutionize AI. The main challenge in CQ QML is *data-encoding*, as to use a quantum device to, say, image classification we need to turn the images into quantum states that can be loaded on a quantum register. Many encoding strategies have been developed [3] and we will briefly look at some of those in the next section. The sector this manuscript fits in is, however, the last one. The QQ intersection of QML deals with coherent quantum information, already stored or directly produced, on a quantum device. Of course, once the data encoding process has been carried out, CQ and QQ come together as one, but QQ mostly deals with completely quantum problems, such as efficient quantum state generation [49], simulation and classification of the phases of some condensed matter model [50] and many others, see [3] and references therein.

3.2 Quantum Data

As in classical machine learning, data stays the fuel of any QML routine. Arguably, loading data into a quantum device is the greatest challenge for CQ QML. On top of being a delicate process, it also constitutes a *bottleneck* for any QML algorithm. In this work, we will not use any encoding strategy, since we will always assume to work in the QQ sector and that our quantum dataset are already available at the quantum processor level. Nonetheless, here we list two common, but alas not much useful, encoding strategies.

- *Basis Encoding*: Basis encoding strategy associates a quantum state in

the computational basis $\{|0\rangle, |1\rangle\}^{\otimes n}$, where n is the size of the qubits register, with a classical n-bit string. For example, the quantum state $|010\rangle$ would be associated with the classical bit-string 010. This is a straightforward method of encoding data, as each bit is directly replaced by a qubit and computations can be performed in parallel on all possible bit sequences in a superposition. In quantum algorithms, the output is often also encoded in the computational basis. The amplitudes of the basis states can then be used to indicate the likelihood of a particular measurement result, with the goal of increasing the probability or absolute square of the amplitude corresponding to the solution. It is evident that this strategy cannot be NISQ-friendly, as it requires as many qubits as the classical bits.

- *Amplitude Encoding*: This strategy corresponds to associating each data point with an amplitude of the qubits register quantum state. Thus now, instead of binary-representing a classical floating point number x and then associating it to the equivalent quantum basis state, one wants x as the coefficient of said basis states. Clearly, first of all one needs to normalize the classical data such that it has norm one, or else it would be impossible to load it into a valid quantum state. Then, we map $\boldsymbol{x} = (x_1, \ldots, x_n)$ into our register as $|\psi_x\rangle = \sum_i x_i |i\rangle$, where $|i\rangle$ are the computational basis states. While amplitude encoding requires fewer qubits than basis encoding, the methods used to prepare the dense amplitude vectors are computationally expensive, making it less suitable for use on near-term quantum devices.

Data encoding on quantum computers is a great challenge for full-purpose QML, and people are also starting to realize that it might not only be a necessary preliminary step to later perform a QML task, but rather that finding a clever encoding map can be formulated as part of the learning algorithm as well. For further details and references we address the reader to [3].

3.3 Quantum Learning Models

We have seen in section 1.2.2 that learning models are basically parametric functions $f_\theta : X \to Y$ from the dataset space to the labels' one, if the model is deterministic, or to $[0, 1]$ if it is probabilistic. Conceptually, a quantum learning model has the very same structure, with the only difference being that the dataset space X is now some Hilbert space $\mathcal{H}_X$. The way the quantum model f_θ is implemented in practice is via *parametric quantum circuits* (PQC). Examples of learning algorithms realized via PQCs are the quantum approximation optimization algorithm [51], and the variational autoencoders [52] and eigensolvers [53]. A detailed review on PQCs and their features, as well as applications, can be found in Ref. [54].

PQCs are quantum circuits whose gates depend on tunable external parameters. Recalling the gate structure $U = \exp\{-itG\}$ discussed in section 2.1, we can understand how the gate time t is the best candidate to play the role of the controllable parameter θ. Indeed, most of the parametric gates are realized by choosing a fixed gate generator G and letting it act on the qubits register for a variable time θ. As an example, from the Pauli operators $\{\sigma_x, \sigma_y, \sigma_z\}$ that we reviewed in section 2.3 we can build three of the most used parametric gates [55]:

$$R_x(\theta) = e^{-i\theta\sigma_x/2} = \begin{pmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2}\,, \end{pmatrix} \tag{3.1}$$

$$R_y(\theta) = e^{-i\theta\sigma_y/2} = \begin{pmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ -\sin\frac{\theta}{2} & \cos\frac{\theta}{2}\,, \end{pmatrix} \tag{3.2}$$

$$R_z(\theta) = e^{-i\theta\sigma_z/2} = \begin{pmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2}\,. \end{pmatrix} \tag{3.3}$$

These implement rotations around the x, y and z axis of the Bloch sphere discussed in the compendium section of the previous chapter, and are nothing but the special case of the rotation gate from Eq. (2.36). Although two-qubit entangling gates can also be parameterized, a common choice being $G = \sigma_z \otimes \sigma_z$, most of the PQCs used by the QML community defer the need for entangling operations to fixed gates, most often than not to the sole CNOT gate defined in eq (2.37). Adopting this strategy, a common layout for a parametrized quantum circuit is the following

$$U(\boldsymbol{\theta}) = V_{D+1} \prod_{j=1}^{D} W_j(\boldsymbol{\theta}_j) V_j\,, \tag{3.4}$$

where we denoted as $W(\boldsymbol{\theta})$ the parametric gates, usually single qubit ones, and with V the non parametric, entangling, operations. Here D is the number of *layers*, sometimes also called the *depth* of the quantum circuit.

One can even consider more general operations, *i.e.* parametrized quantum channels that need not implement unitary evolution on the qubit register, but can leverage the larger family of CPTP maps. Sometimes, this extension of parameterized quantum circuits is called *quantum neural network* (QNN) [56]. In this case, a D-layered QNN would read as a concatenation of CPTP channels $\Phi_{\boldsymbol{\theta}} = \mathcal{N}^D_{\boldsymbol{\theta}_D} \circ \cdots \circ \mathcal{N}^1_{\boldsymbol{\theta}_1}$, where the $\mathcal{N}^l_{\boldsymbol{\theta}_l}$ (with $l = 1, \ldots, D$) are parametrized CPTP channels such that $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_D)$. From the previous, the l-th layer maps between operators (density matrices) acting on some Hilbert space $\mathcal{H}^{l-1}$ to operators acting on some (possibly different) Hilbert space $\mathcal{H}^l$. That is, $\mathcal{N}^l_{\boldsymbol{\theta}_l} : \mathcal{B}^{l-1} \to \mathcal{B}^l$, where we have defined for simplicity of notation $\mathcal{B}^l := \mathcal{B}(\mathcal{H}^l)$.

Let us go back to unitary PQCs for the moment being, as their QNNs cousins are straightforwardly recovered. Upon measuring an observable O, the PQC in Eq. (3.4) leads to the model's output

$$f_{\boldsymbol{\theta}}(x) = \mathrm{Tr}\left[OU(\boldsymbol{\theta})\rho_x U(\boldsymbol{\theta})^\dagger\right] = \langle U(\boldsymbol{\theta})^\dagger O U(\boldsymbol{\theta})\rangle_{\rho_x}\,, \tag{3.5}$$

where we denoted the input data point as ρ_x, and we used the density matrix formalism for the sake of generalization. Notice that we kept things easy here, one is always free to use more observables $\{O_j\}$ in order to build a set of outputs $\{\langle U(\boldsymbol{\theta})^\dagger O_j U(\boldsymbol{\theta})\rangle_{\rho_x}\} = \{f^j_{\boldsymbol{\theta}}(x)\}$, and apply some classical post processing function $\mathcal{C}$ to make the model's output more complex $\tilde{f}_{\boldsymbol{\theta}}(x) = \mathcal{C}(\{f^j_{\boldsymbol{\theta}}(x)\})$. The second equality in (3.5) shows how we can interpret the PQC output both as the expectation value of a fixed, predetermined, observable O over the parametric state that we obtain by steering the input state ρ_x in its Hilbert space with the PQC $U(\boldsymbol{\theta})$, or equivalently as the expectation value over ρ_x of a parametric observable $U(\boldsymbol{\theta})^\dagger O U(\boldsymbol{\theta})$. Recall from section 2.1 that the theoretical expectation value in Eq. (3.5) has to be statistically evaluated when performing real quantum computation. This means that, given an available number of shots S, we sample O S times and average over the observed eigenvalues. Since $\mathcal{O}(1/\epsilon^2)$ shots are needed to achieve error ϵ, this might sound as a great obstacle for reaching a low runtime, as each single measurement has to be repeated by re-running the circuit as a whole, because of the measurement collapsing the state. Perhaps surprisingly though, running a fixed circuit many times does not constitute that big of an overhead, the more costly operation being the change of the circuit's parameter or architecture instead. Analogously, one can build a probabilistic quantum model by simply identifying the generated distribution as that of the measurement outcomes.

We are now left with the question: how would one choose a PQC architecture? An important figure of merit of parametrized quantum circuit is *expressibility*. In brief, expressibility measures how well the chosen PQC architecture is able to probe the Hilbert space of the quantum register. The more it explores it, the more it is expressive. Sometimes just making the circuit deeper by adding more layers is enough to reach the desired expressibility, but the real game-changer is always the *ansatz* that is being used. Think of how a PQC made of single qubit gates only will never change the entanglement content of the input state, no matter how deep we make it, while a single CNOT can do it, as we have shown under eq (2.37).

More formally, given a N qubit PQC $U(\boldsymbol{\theta}) \in SU(2^N)$, its expressibility is defined by comparing the distribution of the circuit output states $|\psi(\boldsymbol{\theta})\rangle = U(\boldsymbol{\theta})\,|0\ldots0\rangle$ for randomly chosen $\boldsymbol{\theta}$ [57] with the Haar distribution, the uniform distribution over the special unitary group $SU(2^N)$. The comparison is carried out by evaluating the difference between the t-moments of the distributions

$$A^t = \left\| \int_{\text{Haar}} (|\psi\rangle\langle\psi|)^{\otimes t} d\psi - \int_{\boldsymbol{\theta}} (|\psi(\boldsymbol{\theta})\rangle\langle\psi(\boldsymbol{\theta})|)^{\otimes t} d\psi(\boldsymbol{\theta}) \right\|^2_{\text{HS}} . \tag{3.6}$$

Here $\int_{\text{Haar}} d\psi$ is the integration over ket states distributed according to the Haar measure, i.e. over unitaries V with $|\psi\rangle = V\,|0\ldots0\rangle$, whereas $\|\cdot\|_{\text{HS}}$ is the usual Hilbert-Schmidt norm:

$$\|A\|_{\text{HS}} = \text{Tr}(A^\dagger A)\,. \tag{3.7}$$

Full expressibility would imply $A^t = 0 \forall t$, since in that case the PQC could explore the whole Hilbert space. This would come at the cost of having high model *complexity*, roughly speaking a large number of parameters, and thus more difficult training. As usual in learning scenarios, one has to find the sweet spot between expressibility and complexity [58], more often than not relying on heuristic approaches. We address the interested reader to [59–61] for a refined analysis of the expressibility, and other interesting figures of merit like *entangling capability* or *parameter dimension*, of some of the most used ansätze.

Unfortunately, *one does not simply pick the most expressive anstatz*. Indeed, one has to add the practical limitations of current NISQ devices to the discussion. Particularly, their low-depth requirement and the limited connectivity between the qubits they support.

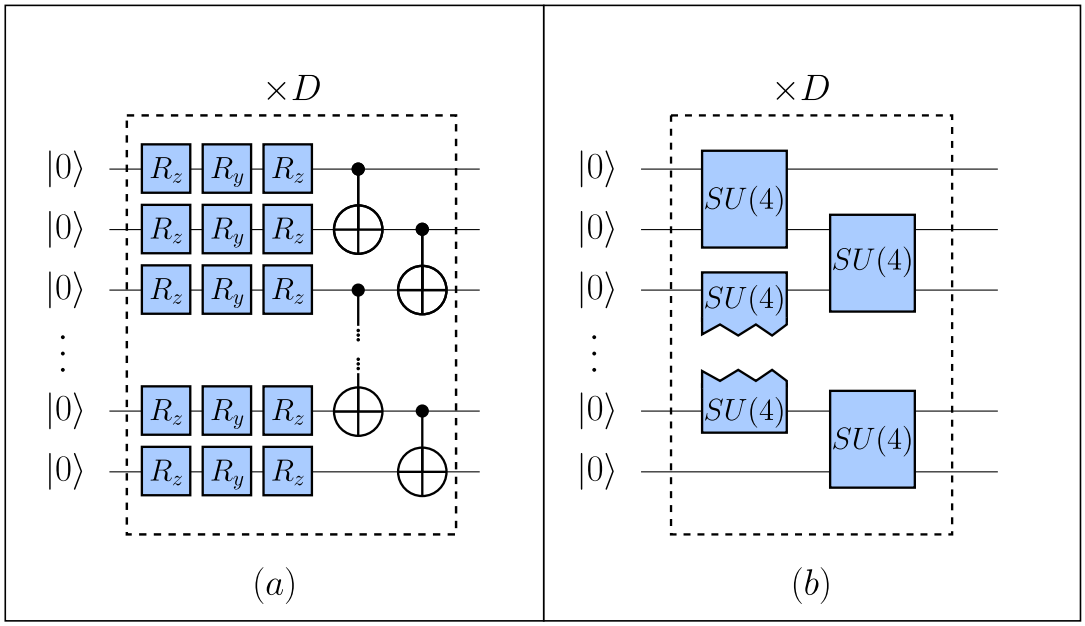


Figure 3.2: **Two ansätze for PQCs.** (a): Hardware Efficient Ansatz (HEA); (b): $SU(4)$ brick-layered ansatz. The HEA architecture features general single qubit unitary evolutions $U \in SU(1)$, usually decomposed via the Euler angles, as discussed under Eq. (2.36), through rotations around the z and y axes, followed by alternated CNOTs. The $SU(4)$ brick-layered ansatz, is pretty much self explanatory, general $SU(4)$ operations, parameterized by 15 parameters are applied in a brick-layered fashion.

Most of the available quantum processors have their qubits arranged in simple one-dimensional geometries, such as lines, rings, or t-shapes. at the same time, current technology allows good control over operations that are performed over physically adjacent qubits. Thus, to perform entangling operations between two distant qubits one needs to first perform ladders of SWAP gates in order to bring the targeted states to adjacent spots. This obviously leads to additional circuit depth and noise accumulation. All of these limitations have led the QML practitioners to almost only rely on *harware-efficient*

ansätze (HEA). With this term, the QML community refers to a class of circuit architectures that aim at being device-friendly, by using only gates that are native to the physical realization of the quantum processor at hand, and arranging them to get the most out of the given connectivity. The prototypical HEA layout, for a quantum processor that has line connectivity, is shown on the left of Fig. 3.2. As we can see, the idea is to perform simple one-qubit operations on each qubit composing the quantum register, and then follow those with a ladder of entangling gates.

When performing numerical simulations of QML routines it is common to adopt stronger ansätze than the HEA class. On one hand, in a simulation scenario we do not need to worry about noise and decoherence, and on the other hand at that stage we are mostly concerned with testing the theoretical feasibility of the learning protocol. Once the method has been tested, one can scale it down to be hardware-friendly, and hopefully perform experiments on real quantum devices. A typical choice for the PQC architecture at the simulation stage is the $SU(4)$ *brick-layered ansatz* depicted on the right of Fig. 3.2.

Finally, there exists a third avenue for the choice of PQC architecture: *problem-inspired ansätze*. This class of PQCs are directly inspired by the physical properties of the problem at hand, a famous example being the case of the *Quabtum Alternating Operator Ansatz* (QAOA) [51] for combinatorial problems. Here, one cannot use a generic ansatz, but rather has to implement unitary evolution of the qubit register according to a given hamiltonian that is defined by the problem itself Belonging to this class is also a whole new family of architectures that is taking over the QML community: *geometric ansätze*. Their name is derived from the geometric deep learning architectures that are under the spotlight in classical machine learning, and their guiding principles are the symmetries of the problem or dataset we need to tackle. We will not delve into their detail here, as this topic will be extensively covered in chapter 6.

3.4 Training a variational quantum circuit

As for its classical counterpart, training a quantum machine learning model $f_{\boldsymbol{\theta}}$ is the act of extremising an objective function $C(\boldsymbol{\theta})$ in terms of the model's parameters $\boldsymbol{\theta}$. To optimize the model one can use gradient-based techniques such as the gradient descent method described in section 1.3. Now, the cost function may depend non-trivially on the model's output $f_{\boldsymbol{\theta}}(x)$, where x is some input data point and we are using a deterministic model $y = f_{\boldsymbol{\theta}}(x)$ as an example. No matter the kind of functional dependence of the cost function on the variational circuit model, it is still a classical dependence, this meaning that after the circuit $f_{\boldsymbol{\theta}}$ has been computed, all the following operations can, and are, be performed on a classical computer. Thus, the derivative $\frac{\partial C}{\partial f_\theta}$ can easily be computed classically and embedded in *automatic differentiation* [62]

frameworks such as those used by the strongest ML libraries. The tricky part comes when realizing that, in order to compute the full derivative of the cost function w.r.t. the model's parameters, one also needs to evaluate the derivative of the circuit's output. Namely, one needs

$$\partial_{\theta_k} C = \frac{\partial C}{\partial f_{\boldsymbol{\theta}}} \frac{\partial f_{\boldsymbol{\theta}}}{\partial \theta_k} \,, \tag{3.8}$$

and while the first factor is easy to compute, the second is non-trivial. To keep things easy, we will consider a single output quantum model, but all of the following can be extended to multi-valued ones. Thus, keep in mind we will use the term gradient somewhat recklessly. In principle, one can always resort to *finite difference* methods to compute derivatives numerically, e.g. by the central derivative rule

$$\frac{\partial f_{\boldsymbol{\theta}}}{\theta_k} \sim \frac{f_{\boldsymbol{\theta}-\epsilon_k} - f_{\boldsymbol{\theta}+\epsilon_k}}{2\epsilon} \,, \tag{3.9}$$

that approximates the partial derivative given two function evaluations at slightly diminished and augmented values, by a shift of magnitude ϵ, of the parameter θ_k with respect to which we want to differentiate $f_{\boldsymbol{\theta}}$. However, for a quantum computation relying on approximate methods such as finite difference ones is risky. In fact, since any practical, *i.e.* on a real quantum device, evaluation of the model $f_{\boldsymbol{\theta}}$ comes with an innate error stemming from the finite number of shots we make to compute expectation values of quantum observables, if the gradient is small we may not trust its approximate value, and we might be forced to use a lot more circuit repetition in order to carry on the optimization. Is there a way to compute the exact gradient of a quantum function? Fortunately, the answer is yes. Consider a simple quantum model

$$f_{\boldsymbol{\theta}} = \langle 0 | \, U(\boldsymbol{\theta})^\dagger O U(\boldsymbol{\theta}) \, | 0 \rangle \,, \tag{3.10}$$

given by the expectation value of some observable O over the parametrized state $|\psi_{\boldsymbol{\theta}}\rangle = U(\boldsymbol{\theta}) |0\rangle$ prepared by a PQC $U(\boldsymbol{\theta})$. Computing its derivative with respect to θ_k we get

$$\partial_{\theta_k} f_{\boldsymbol{\theta}} = \langle 0 | \, (\partial_{\theta_k} U(\boldsymbol{\theta})^\dagger) O U(\boldsymbol{\theta}) \, | 0 \rangle + \langle 0 | \, U(\boldsymbol{\theta})^\dagger O (\partial_{\theta_k} U(\boldsymbol{\theta})) \, | 0 \rangle \,, \tag{3.11}$$

where we used the linearity of the expectation value. This looks like a nice expression to be evaluated on a quantum device, except that the two terms appearing in (3.11) are not expectation values themselves, since bra and ket states on the left and right of O are different, and the derivative of a PQC might not be unitary itself. Nonetheless, there are choices of gate generators for the PQC that make so that the gradient in (3.11) can indeed be computed [63]. Consider, for the sake of simplicity, that the PQC $U(\boldsymbol{\theta})$ consists of a single gate $U(\theta) = \exp\{-i\theta/2G\}$. Assume further that the generator G satisfies $G^2 = I$,

then we can expand the gate as $U(\theta) = \cos\theta/2 I - i \sin\theta/2 G$, and substituting in Eq. (3.10) and using duplication trigonometric identities we find

$$f_\theta = \langle 0| A + B\cos\theta + C\sin\theta |0\rangle \ , \tag{3.12}$$

for hermitian operators A, B, C depending only on O and G and not on θ. Then, using

$$\begin{aligned} \frac{d\cos x}{dx} &= -\sin x = \frac{\cos x + s - \cos x - s}{s \sin s} \\ \frac{d\sin x}{dx} &= \cos x = \frac{\sin x + s - \sin x - s}{s \sin s} \ , \end{aligned} \tag{3.13}$$

which come from the trigonometric addition and subtraction relations, and are valid as long as $s \neq k\pi, k \in \mathbb{Z}$, we arrive at

$$\frac{\partial f_\theta}{\partial \theta} = \frac{f(\theta + s) - f(\theta - s)}{2 \sin s} \ , \tag{3.14}$$

which is the celebrated *parameter shift rule* (PSR). Notice that the condition $G^2 = I$ which we started from holds for any Pauli operator, and those are the most common generators for parametric gates. Not only that, but in principle one can always expand any generic gate into a product of Pauli generated gates [64], and then use (3.14). Notice that, while any value of s that is not an integer multiple of π can do, there is a preferred choice that not only makes Eq. (4.11) nice but most importantly minimizes its variance when experimentally computed via a finite number of shots [63], namely $s = \pi/2$. In the following we will always assume that this is the choice when computing quantum gradients. Thus, for us the parameter shift rule reads:

$$\frac{\partial f_\theta}{\partial \theta} = \frac{f(\theta + \frac{\pi}{2}) - f(\theta - \frac{\pi}{2})}{2} \ . \tag{3.15}$$

Even though we derived the parameter shift rule assuming that the PQC consisted of a single rotation-like gate, it is clear that if the target parameter θ appears in a single gate all of the above still holds. If the parameter which we want to derive controls more gates along the PQC instead, we can simply use the chain derivative rule, compute the parameter shift for each appearance of θ and then sum up each contribution. Moreover, let us mention that the PSR can be generalized to compute second-order derivatives such as the Hessian, that are at the core of more advanced gradient-based optimization approaches [65]. Thus, wrapping up, quantum machine learning models allow for efficient gradient estimation, which in turn enables the use of standard optimization strategies based on gradient descent. For a N parameters PQC, assuming we use S shots to estimate expectation values, computing a gradient requires $2NS$ calls to the quantum model. While S can be chosen to be small [3], having to iterate over all of the N parameters is a huge step back with respect to classical ML, where backpropagation algorithms allow to compute the gradient of a model in a single evaluation. This kind of power is not in reach for quantum

devices, as quantum mechanics forbids sharing of information between partial derivatives [24], but since N has to be low anyways, using the PSR is safe and useful.

Once the ability of computing quantum gradients has been acquired, we can embed any QML routine in a *variational hybrid computation* pipeline, as depicted in Fig. 3.3. Basically, a quantum device is queried any time we need to evaluate the quantum model to get its predictions, or to compute its gradients, and all the information retrieved from the measurements is fed into a classical optimizer which, according to the adopted optimization strategy, suggests an update of the model's parameters. The synergistic use of both classical and quantum devices, allowing for a minimal use of the noisy quantum processor, while all of the auxiliary computation are run on the classical one, makes the *hybrid* protocol NISQ-friendly, as opposed to fully quantum routines as the famous Shor algorithm [34].

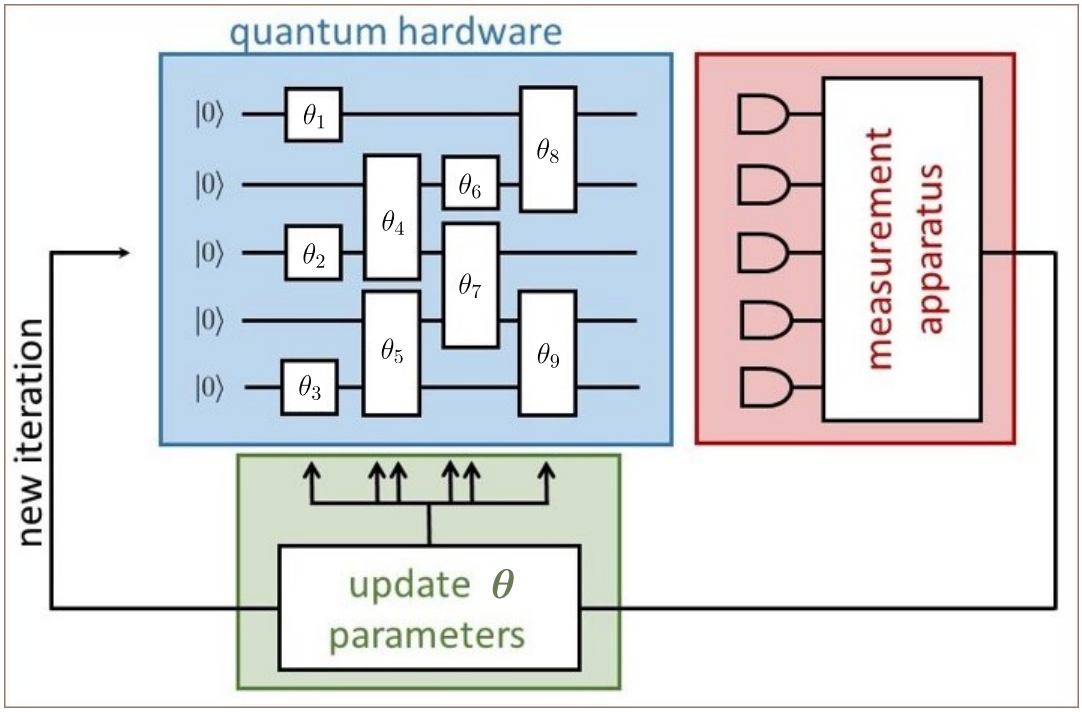


Figure 3.3: **The Hybrid Quantum-Classical Computation Scheme [66].** Quantum and classical processor work together in the variational hybrid computation paradigm. Quantum hardware (blue box) is queried to compute losses and gradients, that are reconstructed via the measurement apparatus (red box). The results are fed into a classical machine (green box) that suggests updated values for the parameters controlling the quantum circuit.

Chapter 4

Quantum Generative Adversarial Learning of Noisy Information

Now that we have introduced all the needed preliminaries and the quantum machine learning framework, we are ready to delve deeper in one of its applications. In this chapter we will describe a particular branch of quantum machine learning: *quantum generative adversarial networks*, and we will show how to solve some issues that arise when training them on noisy devices. The following is the result of our published work [67].

4.1 Quantum adversarial game

We introduced in section 1.4 the classical framework of generative adversarial networks (GANs) [20], which is arguably one of the most outstanding and discussed ML applications. Let us here quickly recall that GANs are learning models that, by exploiting Nash's game-theory results, particularly Kakutani's fixed point theorem [68], can learn to, in principle exactly, reproduce some target data distribution. Practically, two agents, usually dubbed the *generator* (G) and the *discriminator* (D), compete against each other in a zero-sum game, playing in turns, each turn trying to improve their own strategy. The generator wants to fool the discriminator, making it label its generated data as coming from the target distribution, whereas the discriminator wants to correctly tell fake samples apart from real ones. Under some reasonable assumptions[1] the game admits a unique equilibrium state, where G can exactly generate samples from the target distribution and D can only helplessly assign labels at random.

Nash's game theory and the adversarial game framework find a place in quantum machine learning as well. Indeed, it is a recent result of QML that

[1]Namely that the *strategy spaces* of the agents are compact and convex [68]

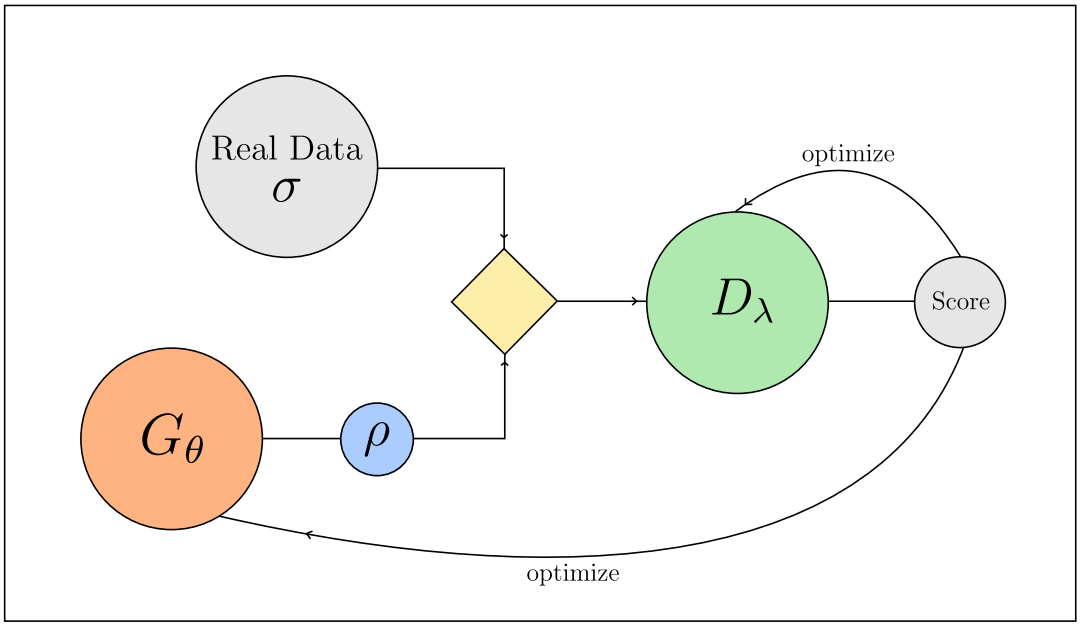


Figure 4.1: **The pipeline of a QGAN.** Here σ denotes the real data, ρ is the fake sample generated by G, whereas D implements the discriminative POVM and decides if its input was real or fake.

GANs can be promoted to the quantum domain [69, 70], and their quantum cousins, perhaps not surprisingly, go by the name of *Quantum Generative Adversarial Networks* (QGANs). A schematic pipeline of their functioning is sketched in Fig. 4.1.

As a classical GAN aims at learning to copy some data distribution, QGANs goal is to learn reproducing an unknown state of a quantum system. Indeed, now the target data distribution is encoded in some quantum state described by the density operator ρ_R, where the subscript makes clear that it is the *real* distribution. Analogously, the generator will output its *fake* data as a quantum density matrix ρ_G. Thus, the discriminator task is that of distinguishing between two quantum states, and as such its action can be implemented as a two-outcome positive operator-valued measure (POVM) Π_i^D whose outcome $i \in \{R, G\}$ judges whether the state is real or fake. Again, D and G play against each other. At each turn G will update ρ_G, using the discriminator's feedback to get closer and closer to ρ_R. In turn, D will improve its strategy for the binary quantum state discrimination task it has to carry out. The error in such discrimination process is given by the conditional probability of judging real a generated state, i.e.

$$p(R|G) = \mathrm{Tr}[\Pi_R^D \rho_G]\,, \tag{4.1}$$

and by that of judging fake a real state

$$p(G|R) = \mathrm{Tr}[\Pi_G^D \rho_R] = 1 - \mathrm{Tr}[\Pi_R^D \rho_R]\,, \tag{4.2}$$

where we used the condition satisfied by the POVM effects $\sum_{i=G,R} \Pi_i = I$, and the defining property of density matrices $\mathrm{Tr}[\rho] = 1$. Assuming that real and fake states are presented to the discriminator with equal *a priori* probabilities, we can define the *discrimination error* as

$$S_D = \frac{p(R|G) + p(G|R)}{2} \,. \tag{4.3}$$

The discriminator strategy during their turn can then be formalized as a minimization of the discrimination error that, without loss of generality, can be written as

$$\text{Discriminator :} \qquad \max_{\Pi_D} \mathrm{Tr}[\Pi_D(\rho_R - \rho_G)] \,, \qquad \text{with } \rho_R, \rho_G \text{ fixed} \tag{4.4}$$

where we set $\Pi_D \equiv \Pi_R^D$ to simplify the notation. An analytic solution to the above optimization is provided by Helstrom's theorem [71, 72], which states that the optimum POVM $\{\Pi_i^D\}$ is formed by projectors onto the positive (Π_R^D) and negative (Π_G^D) subspaces of the operator $\rho_R - \rho_G$. On the other hand, the generator's strategy is to fool the discriminator as much as possible by reducing their ability to distinguish the real and generated states. This can be restated as making the probability of D labeling the fake state as real as big as possible. Thus, the generator's objective function can be chosen as

$$S_G = p(R|G) \,. \tag{4.5}$$

Since the generator can only act on ρ_G, we can also recast its optimal strategy as

$$\text{Generator :} \qquad \min_{\rho_G} \mathrm{Tr}[\Pi_D(\rho_R - \rho_G)] \,, \qquad \text{with } \rho_R, \Pi_D \text{ fixed} \,. \tag{4.6}$$

This strategy has a formal analytic solution as $\rho_G = |\pi_{\max}\rangle\langle\pi_{\max}|$, where $|\pi_{\max}\rangle$ is the eigenvector of Π_D with maximum eigenvalue. If D is always playing with the optimal Helstrom measurement, then ρ_G is a projection onto an eigenstate of $\rho_R - \rho_G$ with positive eigenvalue. As in the classical case, without any restriction on the operations performed by both agents, the game *should* end when G is able to perfectly reproduce the real data and, accordingly, D is unable to correctly discriminate fake data from real ones. Even in the quantum domain, this corresponds to the unique Nash equilibrium of the underlying game [69].

However, it is simple to show that D and G cannot and *should not* solve the optimization problems (4.4) and (4.6) at each iteration. Firstly, they *cannot* find the optimal solution without perfectly knowing, at each iteration, ρ_R and the other player's strategy, which contradicts the original scope of the game. Secondly, they *should not* perform such difficult optimization at each round: if D and G iteratively play using the solution of Eqs. (4.4)-(4.6), then they never reach the equilibrium for mixed states ρ_R. This is summarized by the following theorem, whose proof is straightforward: as discussed above, the solution of (4.6) is always a pure state $\rho_G = |\pi_{\max}\rangle\langle\pi_{\max}|$, and as such $\rho_G \neq \rho_R$ in general.

Theorem 1. *For mixed states ρ_R, the adversarial game fails to converge when D and G iteratively solve the strategies* (4.4)-(4.6).

To achieve convergence, each player must only *slightly* update their strategy at each operation [69], rather than using Eqs. (4.4)-(4.6). Moreover, in the language of Nash equilibria, each player is unaware of the adversary's move, and the best they can do is to assume that the opponent is playing with the optimal strategy and fight against it. We can then define the *score function* of a QGAN as the bilinear function S_D we defined in (4.3)

$$S(\rho_G, \Pi_D) = \mathrm{Tr}[\Pi_D(\rho_R - \rho_G)] \ , \tag{4.7}$$

where we dropped the D subscript since the same score can be used to train the generator, and use this as the objective function of a standard GAN game, as discussed in section 1.4. Let us check one last time that: G increases its score whenever D loses the same amount, making this a zero-sum game; both the states ρ_G and the measurement operators Π_D form a convex set in their respective spaces.[2] Therefore, the Nash equilibrium is the result of the minimax problem $\min_{\rho_G} \max_{\Pi_D} S(\rho_G, \Pi_D) = \frac{1}{2} \min_{\rho_G} \|\rho_R - \rho_G\|_1 = 0$ where the first equality follows from the Helstrom theorem and the definition of the 1-norm [73]. As a result, the Nash equilibrium is when the generator is able to perfectly reproduce ρ_R, as originally shown in [69].
However, how to achieve in practice this equilibrium configuration is far from being trivial.

Inspired by the success of gradient-based training of generative adversarial networks [74], the most natural approach to play the quantum adversarial game described by the score function (4.7) is to use a suitable parametrization of ρ_G and Π_D, see e.g. the one with PQCs described in Fig. 4.2, and then iteratively update these parameters, *e.g.* via the gradient descent optimizer in Eq. (1.4) [69]. Using these methods, convergence with pure target states $\rho_R = |\psi_R\rangle \langle\psi_R|$ has been obtained in several scenarios [70, 75]. Moreover, QGANs have been exploited to learn classical distributions of data [76, 77], thus they have immediately found applications in both CQ and QQ quantum machine learning. All of the early works on QGANs focused on learning pure quantum states, whereas mixed states (i.e. noisy information) have been addressed only as ensembles of pure data [78]. However, the latter play a crucial role in the coming NISQ technologies since the environmental noise is unavoidable and usually partially destroys the quantum features as entanglement that do not have a classical counterpart and that are instead mainly responsible for the predicted quantum speedups. These reasons led us to strongly believe that, in order to get a deeper understanding of the performance of QGANs on real noisy quantum device, it is remarkably relevant to investigate the scenario of learning mixed quantum states. This has been the main focus of this first work of us, and in the next sections we will show a thorough numerical investigation

[2] Since Π_D is part of a POVM it is a positive operator with $\|\Pi_D\|_\infty < 1$.

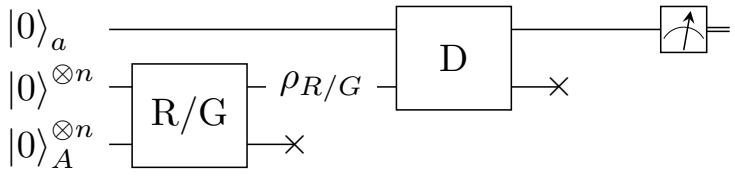


Figure 4.2: **QGAN circuit structure for generic n-qubit mixed states.** The R/G/D blocks are PQCs that are exploited to create real/generated data, and to implement the discrimination process, respectively. The discriminator has access to an ancilla qubit a, while the auxiliary qubits A are used by R/G to create mixed states. The $\times$ symbol over a wire means tracing the degrees of freedom associated to it.

on the problematics of naive gradient-descent based training of QGANs when ρ_R, and consequently ρ_G, are mixed.

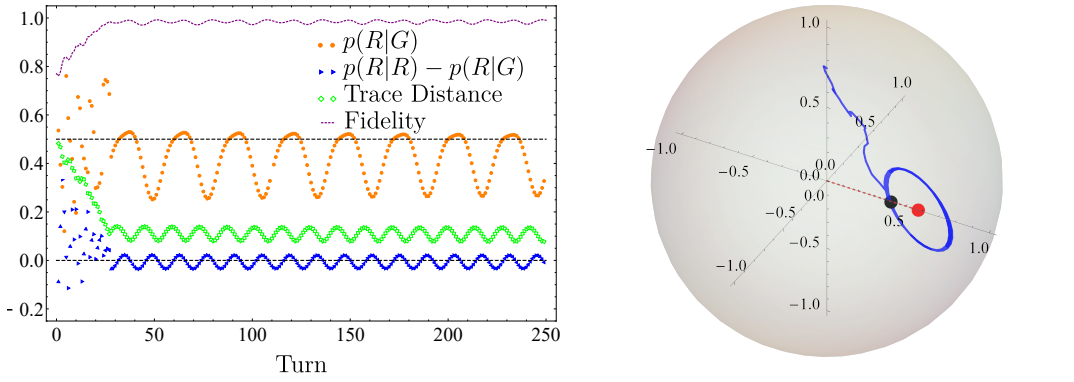
This is due to the bilinear nature of the score function (4.7). Indeed it has been shown that the adversarial optimization of bilinear score functions may display limit cycles when trained with standard gradient descent rules, or even a "chaotic" behaviour, see *e.g.* [79, 80] and references therein.

Recall that, as we have seen in section 2.2, to generate mixed quantum states, one can create a generic pure state (living in a larger Hilbert space) as a quantum circuit by applying quantum gates to a given pure (ground) state, and then tracing out half of the qubit register. The same procedure can be exploited to generate the fake data, but in terms of a PQC where the gate parameters can be tuned. Besides, D applies another PQC to the real or fake state at hand, and entangles it to an additional (ancilla) qubit that later is measured in order to perform the discrimination, this way applying Naimark's theorem and allowing for a two outcomes POVM to be implemented – see Fig. 4.2.

More precisely, let us consider the simplest case where ρ_R is a single-qubit mixed state. The most natural parametrization of ρ_R is via the Bloch vector $\boldsymbol{r}$, namely $\rho_R = [\mathbb{1} + \boldsymbol{r} \cdot \boldsymbol{\sigma}]/2$ where we recall from compendium section 2.3 that $\boldsymbol{\sigma}$ is the vector of Pauli matrices and $|\boldsymbol{r}| \leqslant 1$. Similarly we parametrize ρ_G with the Bloch vector $\boldsymbol{g}$ and $\Pi_D = [d^0\mathbb{1} + \boldsymbol{d} \cdot \boldsymbol{\sigma}]/2$, where $d^0 = \mathrm{Tr}\Pi_D$ and $d^0 \geqslant |\boldsymbol{d}|$ (because $\Pi_D \geqslant 0$). With this simple parametrization Eq. (4.7) becomes a bilinear form in the Bloch vectors

$$S(\Pi_D, \rho_G) = \frac{\boldsymbol{d} \cdot (\boldsymbol{r} - \boldsymbol{g})}{2} \ . \tag{4.8}$$

The above score function has been extensively investigated in Refs. [81, 82] where the emergence of limit cycles in classical GANs training was shown. Limit cycles are a detrimental behaviour that can spoil GANs training. Roughly speaking, when training gets stuck in a limit cycle the two agents start modifying their strategies in a periodic way, endlessly chasing each other without ever attaining convergence. Nonetheless, Refs. [81, 82] focus on bilinear prob-



a) Figures of merit.

b) Trajectory of ρ_G in the Bloch ball.

Figure 4.3: **Emergence of a limit cycle when the score** (4.8) **is optimized via gradient descent/ascent**, as described in the main text. Here the learning rate of both agents is $\eta = 0.1$, and one training turn consists of 5 discriminator's steps followed by a single generator's one.

lems with linear constraints, while Bloch vectors satisfy a non-linear constraint since they live in the Bloch ball. This difference may be the reason behind the good performance of quantum adversarial learning for pure states [70, 75], as pure states lie at the boundary of the Bloch sphere where such non-linear constraints are important. However, when dealing with the optimization of highly mixed states, which lie well inside the Bloch ball, the presence of the boundary may not affect the optimization, and limit cycles may emerge. We summarise this aspect in the following theorem, whose proof, adapted from [81], can be found in Appendix B:

Theorem 2. *(informal statement): Gradient descent/ascent applied to the problem* $\min_{\rho_G} \max_{\Pi_D} S(\Pi_D, \rho_G)$ *diverges for states far from the surface of the Bloch sphere.*

We bring evidence to the previous statement by running a QGAN game in a single qubit scenario where both D and G are parametrized via their Bloch vectors. We employ gradient descent/ascent (GDA) – namely gradient descent for $\boldsymbol{g}$ and gradient ascent for $\boldsymbol{d}$ –on the score function (4.8) with an added penalty term to enforce the constraints on the Bloch vectors, i.e. $\|\boldsymbol{g}\| \leqslant 1$ and $\|\boldsymbol{d}\| \leqslant d^0 \leqslant 2 - \|\boldsymbol{d}\|$. Results are shown in Fig. 4.3, where the limit cycle behaviour in the trajectory of $\boldsymbol{g}$ is evident.

An algorithm dubbed "optimistic mirror descent" (OMD) has been proposed in Ref. [81] to escape from the limit cycles that emerge in the minimax optimization of bilinear cost functions (4.8). In the next section we show that, although perfect limit cycles may not exist for more complex parametrizations of ρ_G and Π_D, a simple gradient descent/ascent update may produce a "chaotic" behaviour, where convergence is not observed. We find instead that convergence is obtained via OMD.

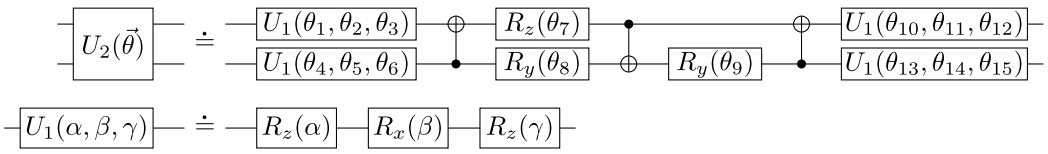


Figure 4.4: **The building block of G and D circuits** with 3 CNOTs and 15 single-qubit rotations. U_1 and U_2 implement elements of SU(2) and SU(4), respectively, where R_i are rotations around the i-th axis.

4.2 Training with parametric quantum circuits

Here, we will ultimately be concerned with the problem of learning a mixed state via a QGAN game. Since every mixed state can be written as a pure state in a larger Hilbert space (Fig. 4.2), we build the generator via the following PQC with classical parameters $\boldsymbol{\theta}_G$

$$\rho_G \;= \mathrm{Tr}_A\left[|\psi_{GA}(\boldsymbol{\theta}_G)\rangle\langle\psi_{GA}(\boldsymbol{\theta}_G)|\right] \;, \quad |\psi_{GA}(\boldsymbol{\theta}_G)\rangle \;= U(\boldsymbol{\theta}_G)\,|0\rangle^{\otimes 2n} \;, \tag{4.9}$$

where both A and ρ_G contain n qubits, and $U(\boldsymbol{\theta}_G)$ is the unitary operator corresponding to the PQC. Similarly, since every measurement operator can be written as a projective measurement onto a larger Hilbert space (Fig. 4.2), we define the discriminator's POVM with classical parameters $\boldsymbol{\theta}_D$ as

$$\Pi_D = \mathrm{Tr}_a\left[U(\boldsymbol{\theta}_D)^\dagger(\mathbb{1}_D \otimes |0\rangle_a\langle 0|)U(\boldsymbol{\theta}_D)\right] \;. \tag{4.10}$$

where a is a single auxiliary qubit. This measurement can be interpreted as follows: first apply a PQC $U(\boldsymbol{\theta}_D)$ entangling the system with an auxiliary qubit a, then measure the qubit a in the computational basis. If the outcome 0 is detected, then we guess that the state is the real state, otherwise (outcome 1) the state is judged as fake.

4.2.1 Circuits Ansätze

Following Refs. [75, 83], G and D circuits are built by repeating a two-qubit block which implements a generic unitary $U \in \mathrm{SU}(4)$. As discussed in section 3.3, this ansatz is not necessarily device-friendly, as controlling a generic element of SU(4) and doing so many times requires many logical operations. Nonetheless, since we are interested in exact numerics to get a grasp on the feasibility of the QGAN method for mixed states, we do not have to worry about that. As shown in Fig. 4.4, the building block is composed of 15 single-qubit rotations as those described in eqs. (3.1,3.1,3.3) and 3 CNOT gates. One block allows to generate every two-qubit pure state. For larger registers, we apply this block to each pair of consecutive qubits, thus obtaining a layer. Layers are then concatenated in a staggered pattern.

We have discussed in section 4.2 that gradients of quantum models can quite conveniently be analytically evaluated directly via the quantum device by querying the model itself twice for every parameter it has. This is the parameter shift rule [84–86] that we introduced in Eq. (3.15), and that we restate here

$$\frac{\partial f}{\partial \theta_i} = \frac{1}{2}\left[f\left(\boldsymbol{\theta} + \frac{\pi}{2}\boldsymbol{e}_i\right) - f\left(\boldsymbol{\theta} - \frac{\pi}{2}\boldsymbol{e}_i\right)\right], \tag{4.11}$$

where $\boldsymbol{e}_i$ is the unit vector in the i-th direction, and the rule holds thanks to our ansatz being made of repeated Pauli rotations and fixed entangling gates.

First of all, before tackling mixed states learning, let us confirm the feasibility of the QGAN protocol for pure states. In Fig. 4.5 we show the results of our numerical investigation, where the training was evaluated in terms of some relevant figures of merit:

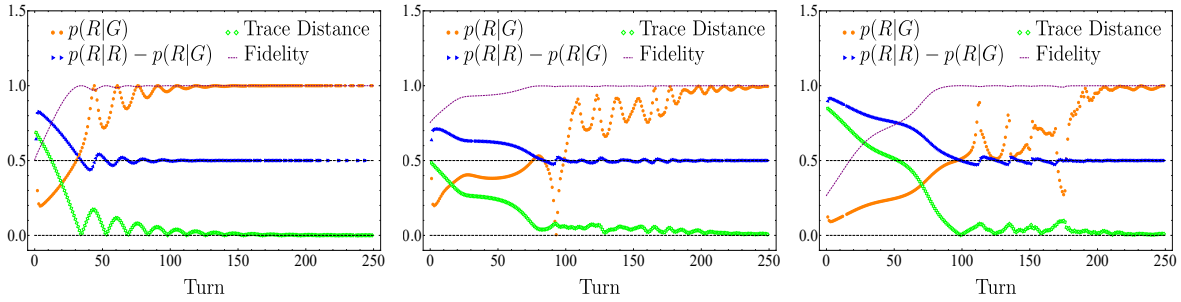
- the score function (4.7) value S. Recall that, while in other learning task such as regression or classification the objective function has a definite meaning, usually the error the model is making, and as such we know that small values of it are favourable, in the adversarial learning framework the score function cannot be used to understand how the agents are performing. Still, we expect it to be stationary at the equilibrium point.

- the generator's score function S_G (4.5), *i.e.* the probability $p(R|G)$ of D labelling fake data as true. Again, we expect it to be stationary when the game converges.

- the trace distance $d = \frac{1}{2}\|\rho_G - \rho_R\|_1$ between the generated state and the target one, unequivocally telling us whether the generated fake state is approaching and has, at the end, reached the target one.

- the fidelity $F = \left[\mathrm{Tr}\sqrt{\sqrt{\rho_G}\rho_R\sqrt{\rho_G}}\right]^2$ between target and fake state, which is another measure of distance between quantum states. For pure states fidelity and trace distance are equivalent, but for mixed states they capture different features [24].

In our simulations the target *real* data are random pure states of n qubits, with $n = 1, 2, 3$, obtained via a PQC with the same structure of the one used for G, but with random fixed parameters. Here, training is carried out via alternately updating D and G via a single gradient descent/ascent step. We have tried different optimizers, always observing a qualitatively similar convergence behaviour. Particularly, the task resulted easy already for the vanilla gradient descent of Eq. (1.4), and while we could arguably improve convergence time by performing some *hyperparameter optimization* to look for e.g. the best learning rate this is beyond the scope of this analysis.



a) $n = 1$. **b)** $n = 2$. **c)** $n = 3$.

Figure 4.5: **Convergence of QGANs for learning pure states of n qubits.** Both agents rely on standard gradient descent/ascent optimization. The target and the initial fake states are randomly chosen on the Bloch sphere.

4.2.2 Emergence of limit cycles

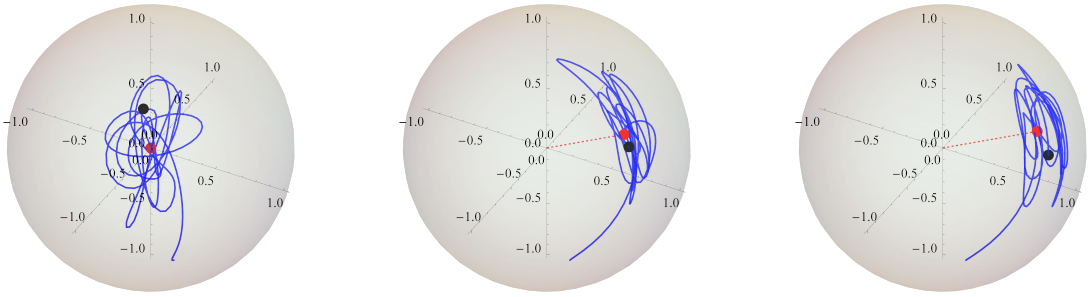
We now turn to our real interest, the more appealing case of learning *mixed states*. They have been so far addressed as an ensemble of orthogonal pure states [78], while here they are created by tracing out half of the qubits register. Our results are summarized in Fig. 4.6, where we show the learning process for mixed states of the form $\rho_R = \frac{I}{2} + \frac{a}{2\sqrt{2}}(\sigma_x + \sigma_y)$ with purity $P = \mathrm{Tr}\rho^2 = \frac{1+a^2}{2}$, ranging from the completely mixed one $P = 1/2$ to $P = 3/4$. The selected optimizers are the previously defined GDA and *Adam*, i.e. one of the best performing optimization algorithm for ML [87]. The messy behaviour that we can observe in Fig. 4.6 shows how none of the chosen optimizers allows to reach convergence. This does not change even if we tweak the values of the optimization hyperparameters. However, comparing these results with the ones in Fig. 4.3, we can see that for PQCs the exact limit-cycle behaviour disappears because the score function is no longer bilinear. Let us point out that in Fig. 4.6 we have an overparametrization because D and G use 15 parameters each, whereas a general single-qubit POVM has 4 real degrees of freedom only, and a single qubit mixed state has 3. For this reason we devise two tailored circuits in order to achieve a minimal parametrization for both D and G (see Fig. 4.7), as in the following:

$$\rho_G(\boldsymbol{\theta}) = \frac{1}{2}\begin{pmatrix} 1 + c(\theta_1)c(\theta_2) & c(\theta_1)s(\theta_2)\left(s(\theta_3) + ic(\theta_3)\right) \\ c(\theta_1)s(\theta_2)\left(s(\theta_3) - ic(\theta_3)\right) & 1 - c(\theta_1)c(\theta_2) \end{pmatrix}, \quad (4.12)$$

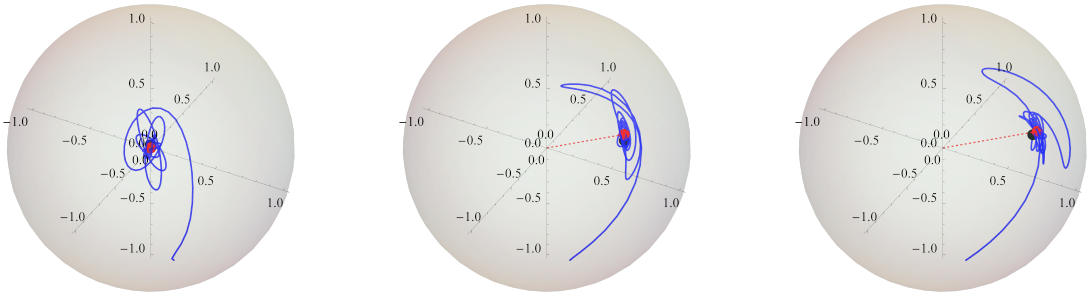
and

$$\Pi_D(\boldsymbol{\theta}) = \frac{1}{2}\begin{pmatrix} 1 + c\left(\theta_1 + \theta_2\right)c\left(\theta_4\right) & s(\theta_4)\left(c(\theta_1)s(\theta_3) - ic(\theta_2)c(\theta_3)\right) \\ s(\theta_4)\left(c(\theta_1)s(\theta_3) + ic(\theta_2)c(\theta_3)\right) & 1 - c\left(\theta_1 - \theta_2\right)c\left(\theta_4\right) \end{pmatrix}, \quad (4.13)$$

with $\cos(\theta) \equiv c(\theta)$ and $\sin(\theta) \equiv s(\theta)$. Even with these tailored circuits, convergence is not achieved as numerically reported in Fig. 4.8. Moreover, by



a.1) GDA: $P = 1/2$. **a.2)** GDA: $P = 2/3$. **a.3**) GDA: $P = 3/4$.



b.1) Adam: $P = 1/2$. **b.2)** Adam: $P = 2/3$. **b.3**) Adam: $P = 3/4$.

Figure 4.6: **Limit-cycle-like behaviour of QGANs when learning mixed states for different values of the purity P.** As optimizers, we use GDA (top row) and Adam (bottom row). None of them display convergence, although the latter has a less pronounced oscillating behaviour. In all these cases the initial configurations of G and D correspond to the same random parameters. These trajectories have been obtained by running the QGAN for 250 total turns, where each turn comprises 10 optimization steps for D followed by 1 for G. Lastly, the learning rate of GDA is set to 0.1, whereas that of Adam is 0.05.

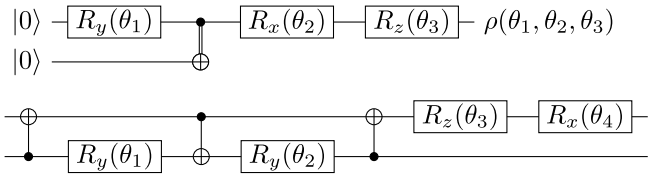
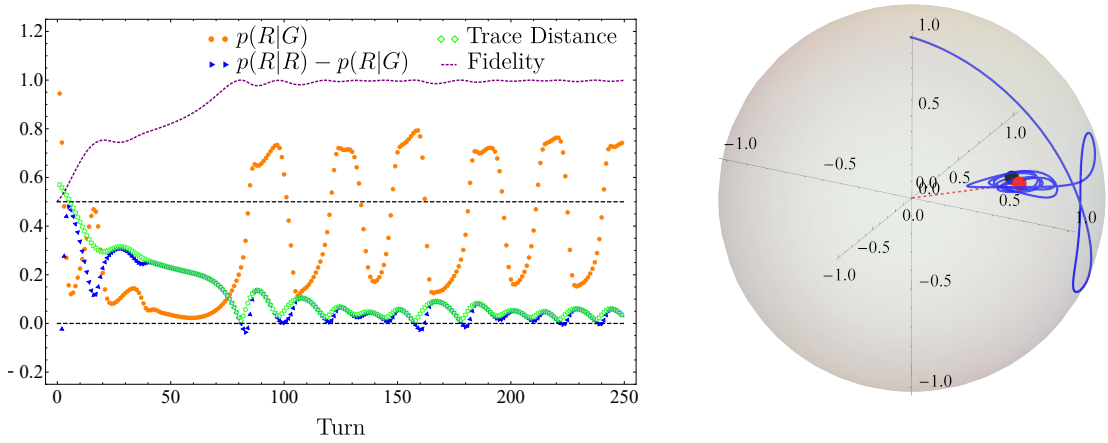


Figure 4.7: **The minimal circuits for G (top), and D (bottom).**

using simple gradient descent method, we still observe limit cycles (not shown in figure).

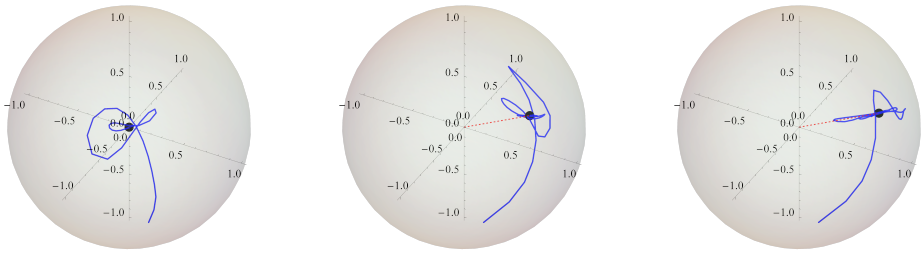
4.2.3 Training with optimism

In standard GANs competing players are usually unaware of the opponent's strategy. However, each player may try to guess the opponent's move in order to improve its strategy. This is the building concept of the *Optimistic*



a) Figures of merit. **b)** Bloch ball trajectory.

Figure 4.8: **Training behaviour for learning a mixed state with** $P = 3/4$ under the same conditions of 4.6 with Adam and the tailored agents, but a different initial configuration.



a) OMD: $P = 1/2$. **b)** OMD: $P = 2/3$. **c)** OMD: $P = 3/4$.

Figure 4.9: **Convergence of QGANs for learning mixed states via OMD** under the same conditions used in Fig. 4.6.

Mirror Descent (OMD) optimization algorithm [88], which was shown to fix convergence issues, namely limit cycles, of classical GANs with bilinear score functions – see Ref. [81]. However, there it has been used to enhance convergence also in the case of non-bilinear score functions. Motivated by these results, we now show that OMD works successfully also for our QGANs – see Fig. 4.9. More specifically, the OMD-based update rule for the score function of Eq. (4.7), $S(\boldsymbol{\theta}_D, \boldsymbol{\theta}_G) := S(\Pi_D(\boldsymbol{\theta}_D), \rho_G(\boldsymbol{\theta}_G))$, reads

$$\boldsymbol{\theta}_D^{t+1} = \boldsymbol{\theta}_D^t + 2\eta_D \nabla_{\boldsymbol{\theta}_D} S(\boldsymbol{\theta}_t^D, \boldsymbol{\theta}_t^G) - \eta_D \nabla_{\boldsymbol{\theta}_D} S(\boldsymbol{\theta}_{t-1}^D, \boldsymbol{\theta}_{t-1}^G)\,, \tag{4.14}$$

$$\boldsymbol{\theta}_G^{t+1} = \boldsymbol{\theta}_G^t - 2\eta_G \nabla_{\boldsymbol{\theta}_G} S(\boldsymbol{\theta}_{t+1}^D, \boldsymbol{\theta}_t^G) + \eta_G \nabla_{\boldsymbol{\theta}_G} S(\boldsymbol{\theta}_t^D, \boldsymbol{\theta}_{t-1}^G)\,, \tag{4.15}$$

where $\eta_{D/G}$ are the learning rates for D and G. Notice that this rule corresponds to the scenario where D is optimized first, which is the one we adopted.

4.3 Convex optimization

Since PQCs are not the only way of modelling quantum states, here we present a non-parametric method, hereafter dubbed ConvexQGAN, to solve the minimax problem $\min_{\rho_G}\max_{\Pi_D} S(\rho_G,\Pi_D)$ using the formalism of convex optimization presented in Ref. [89]. Both $\{\rho_G\}$ and $\{\Pi_D\}$ are convex sets, while $S(\rho_G,\Pi_D)$ is bilinear, thus when we iteratively fix either ρ_G or Π_D we always obtain a convex function over a convex set. Therefore, by adapting the Frank-Wolfe algorithm from Ref. [89], we may write the following update rules at the k-th step

$$\Pi_D^{k+1} = (1-\alpha_k)\Pi_D^k + \alpha_k \left|D_k\right\rangle\left\langle D_k\right| \ , \tag{4.16}$$

$$\rho_G^{k+1} = (1-\beta_k)\rho_G^k + \beta_k \left|G_k\right\rangle\left\langle G_k\right| \ , \tag{4.17}$$

where α_k and β_k are decaying learning rates, e.g. typically $\alpha_k=\beta_k=\frac{2}{k+2}$, the state $|G_k\rangle$ is the eigenvector with smallest eigenvalue of $\nabla_{\rho_G} S(\rho_G^k,\Pi_D^k) = -\Pi_D^k$, while $|D_k\rangle$ is the eigenvector with smallest eigenvalue of $-\nabla_{\Pi_D} S(\rho_G^k,\Pi_D^k) = -(\rho_R-\rho_G^k)$. Although the update rules directly follow from the Frank-Wolfe algorithm, we highlight here an interesting result from the physics points of view. The states $|D_k\rangle$ are elements of Helstrom measurement to optimally distinguish the real state ρ_R from the current fake state ρ_G^k. As such, it is tempting to consider a different strategy with $\alpha_k=1$ at each iteration step. The downside of the latter approach is that the measurement operator highly fluctuates between different steps. However, for $\alpha_k=1$ we get $|D_k\rangle=|G_k\rangle$ so Eq. (4.17) gets a clear operational meaning. The generator's state is iteratively updated with one of the states entering in the Helstrom optimal measurement. This reminds us the original optimization from Eq. (4.6), but without its convergence issues for mixed states. Indeed, the update rule of Eq. (4.17) allows the generation of mixed states, unlike in Eq. (4.6).

Finally we propose a physics inspired alternative by observing that, for small β_k, Eq. (4.17) can be interpreted as an imaginary time evolution

$$\rho_G^{k+1} \propto e^{\beta_k H_k}\rho_G^k e^{\beta_k H_k} \ , \qquad H_k = \left|G_k\right\rangle\left\langle G_k\right| \ , \tag{4.18}$$

where after the imaginary evolution we need to normalize the state such as $\mathrm{Tr}[\rho_G^{k+1}]=1$. The gradient-based Frank-Wolfe algorithm (4.16)-(4.17) and the imaginary time iteration (4.18) are numerically studied in Fig. 4.10 for random 5-qubit states with full-rank. For the imaginary time iteration we use $\alpha_k=1$, so $|G_k\rangle=|D_k\rangle$ in (4.18). We observe in Fig. 4.10 that the imaginary time evolution, together with the optimal Helstrom measurement at each step, significantly outperforms the Frank-Wolfe iteration, both in terms of speed and accuracy.

ConvexQGAN methods show fast convergence towards the equilibrium configuration, but they require eigendecompositions of the state at each step. This operation is simple for classical computers as long as the Hilbert space is sufficiently small. To extend this operation to larger systems, we now discuss

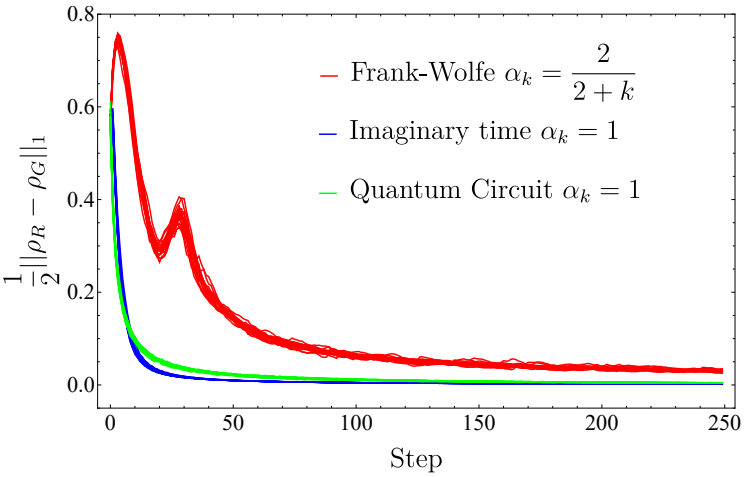


Figure 4.10: **ConvexQGAN:** Learning different random mixed states using either the Frank-Wolfe iteration (4.16)-(4.17), the imaginary time evolution (4.18) or the quantum circuit update from (4.21). The cases with $\alpha_k = 1$ in (4.17) correspond to Helstrom measurements. For each algorithm, 20 lines are plotted for different random initial states. All simulations are for 5-qubit states.

how to write an update like in Eq. (4.18), but using a quantum circuit. For this purpose, we use the following quantum map, which is at the heart of the quantum density matrix exponentiation algorithm [90],

$$\mathcal{E}_\sigma^{\pm t}[\rho] = \mathrm{Tr}_2[e^{\pm it\,\mathrm{SWAP}}\rho\otimes\sigma e^{\mp it\,\mathrm{SWAP}}] = \cos(t)^2\rho + \sin(t)^2\sigma \pm \frac{i}{2}\sin(2t)[\rho,\sigma]\ , \tag{4.19}$$

where SWAP is the swap operator (see Eq. (2.38)). Applying this map twice with different signs, one has

$$\mathcal{E}_\sigma^{-t}\circ\mathcal{E}_\sigma^{+t}[\rho] = \cos(t)^4\rho + \sin(t)^2(1+\cos^2 t)\sigma + \frac{1}{4}\sin(2t)^2[[\rho,\sigma],\sigma]\ . \tag{4.20}$$

Therefore, setting $I_\sigma^t[\rho] = \mathcal{E}_\sigma^{-t}\circ\mathcal{E}_\sigma^{+t}[\rho]$ and t_k such that $\cos^4 t_k = 1-\beta_k$, namely $\beta_k \approx 2t_k^2$, we get

$$\rho_G^{k+1} = I_{H_k}^{t_k}[\rho_G^k] = (1-\beta_k)\rho_G^k + \beta_k(H_k + H_k\rho_G^k + \rho_G^k H_k - 2H_k\rho H_k) + \mathcal{O}(\beta_k^2)\ , \tag{4.21}$$

where H_k was defined in (4.18). The latter update rule is akin to a mixture of (4.17) and (4.18), but it has the advantage that it can be explicitly evaluated as a quantum circuit applied to ρ_k and two copies of the state $|G_k\rangle$. As shown in Fig. 4.10, the performance obtained with the update rule (4.21) is similar to that of imaginary time evolution. Therefore, if the states $|G_k\rangle$ can be efficiently prepared, for instance via strategies like the Helstrom classifier circuit from [91], then the above update rule can be used for QGAN training in a quantum computer.

Chapter 5

Quantum Generative Adversarial Learning of Noisy Maps

In this chapter we show how to generalize the QGAN architecture we just finished analysing in Chapter 4, from the context of quantum states to the context of quantum maps (or superoperators). In other words, the real data we want to target is a noisy quantum map while the generator's duty will now be that of synthesising fake quantum maps that mimic the real (unknown) one's action as best as possible. Since this new learning scheme targets *super*operators and is still based on the QGAN paradigm, we decided to dub it: SuperQGAN.

In what follows we are going to present the reader with the introduction, and preliminary study, of SuperQGANs that we published in [92].

5.1 Motivation

We ended the previous chapter having endowed quantum generative adversarial networks with the ability of fruitfully tackling mixed states by the means of *optimism*. As we have discussed in section 2.2, mixed states arise naturally when the quantum systems we want to describe is not isolated, but is part of a larger one. Contextually, the transformations these states are subject to, *i.e.* their evolution in time, has to be described by the quantum maps [24, 26] we introduced in section 2.2.2. These maps are thus the only way we can characterize what happens in a real quantum computing device instead of the ideal unitary circuit we designed.

Indeed, in this NISQ era, the circuital paradigm of perfect quantum computation remains only an ideal abstraction. The simple logic gates one would like to compose in order to build the desired algorithm are not perfect unitary evolutions of the targeted systems. Rather, they also induce unwanted, but also unavoidable, couplings with the environment leading, for example, to de-

coherence and loss of quantumness. It is thus more appropriate to address the physical processes occurring in a NISQ processor by describing the action of a quantum circuit, or of any of the gates constituting it, rather than with a unitary U mapping the input state as $|\psi\rangle \to U|\psi\rangle$, with a general CPTP map Φ. Recall that those are completely positive (CP) and trace-preserving (TP) linear super-operators acting on the space of density operators of the input system $\Phi : \rho \to \Phi(\rho)$. Sometimes they can go by the name of quantum channels, namely when their input and output spaces are the same, but we will continue using the terms map and channel equivalently. Understanding the properties of the quantum maps, and thus of the hidden couplings with the environment, that are really happening in a quantum computer is of paramount importance in order to move on to an era of computation where quantum protocols are dominant. These unwanted couplings, on top of limiting the depth of the quantum circuits that can be reliably devised, may also induce back-flows of information from the environment to the computing system, leading to the observation of *memory effects* when repeatedly using a given quantum gate. Having ways to characterize the noise occurring on NISQ processors can lead to devise tailored circuital schemes that can minimize error rates, as in error mitigation protocols such as those of [93, 94], or even exploit noisy processes to achieve the desired goal – see for instance Refs. [95–97].

5.2 Definition of SuperQGANs for quantum maps

Let us recall from the discussion around figure 2.3 that when a single qubit map is independently applied (in parallel) to n qubits, then the global quantum map reads as $\Phi^{\otimes n}$. When this map is applied n times (in series) to the same qubit, we will write it as $\Phi^n = \Phi \circ \cdots \circ \Phi$. In both cases, it is assumed that the noisy operations are uncorrelated: there are no spatial or temporal noise correlations. However, in a NISQ device neighboring qubits typically experience spatially correlated noise, and the later-time evolution may display (non-Markovian) memory effects, hence leading to temporal noise correlations. As depicted in Fig. 5.2, both these cases can be represented by the action of a map $\Phi^{(n)}$, more general than either $\Phi^{\otimes n}$ or Φ^n. Particularly, this generalized map $\Phi^{(n)}$ is defined so that for spatially correlated noise, it maps n-qubit states to n-qubit states, while for temporally correlated noise $\Phi^{(n)}$ maps a single qubit to a "history" of single qubit states ρ_t, with $t = 1, \ldots, n$, each one representing the state of the system after the t-th use of the memory map Φ – see Fig. 5.2.

Before stating our definition of SuperQGANs, let us quickly recall the basic structure of a "standard" QGAN that we covered in detail in the previous Chapter. QGANs are generative quantum models that, by exploiting an adversarial game where a Generator (G) agent, able to produce tunable fake instances ρ of some target (real) distribution of data, encoded in a quantum

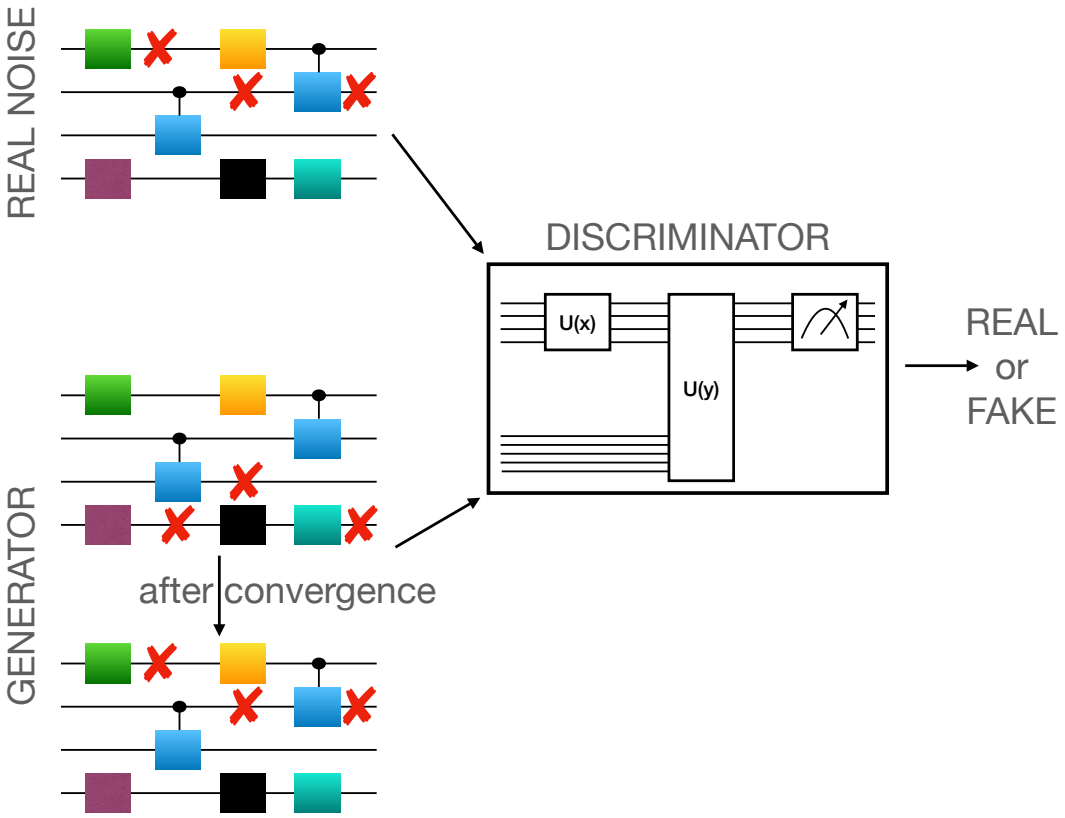


Figure 5.1: **Pictorial representation of a** SuperQGAN**,** where the Discriminator needs to distinguish a real noisy quantum circuit from a fake one created by the Generator. These two agents play against each other, in particular the Generator needs to generate better and better data such that the task of the Discriminator becomes more and more complicated. The game ends (convergence) when the generator learns to create the real noisy quantum circuit (i.e., fake=real), hence identifying the errors occurring in the real circuit (crosses) running on a NISQ device.

state σ, is opposed to a Discriminator (D) that is in turn able to find good strategies, *i.e.* good POVMs, to tell real and fake data apart. Played in turns, this game can be framed in Nash's game-theory and, under reasonable assumption of convexity, possesses a unique equilibrium point [68], where G is able to completely fool D and achieve a perfect data copying strategy.

Since ultimately the QGAN scheme is an adaptive state-tomography protocol, the extension from states to superoperators is conceptually simple: we need to promote QGANs to a *process*-tomography [98] scheme. With this in mind, we define a SuperQGAN as yet another two-player game, where now the generator tries to reproduce a general CPTP map $\Phi^{(n)}$ and D has to carry out an optimization over process-tomography strategies.

The general maps $\Phi^{(n)}$, portrayed in Fig. 5.2 describe, either spatially correlated or temporally correlated noise. For temporal correlations, the map can be expressed as a "quantum comb" [99], where the name comes from their

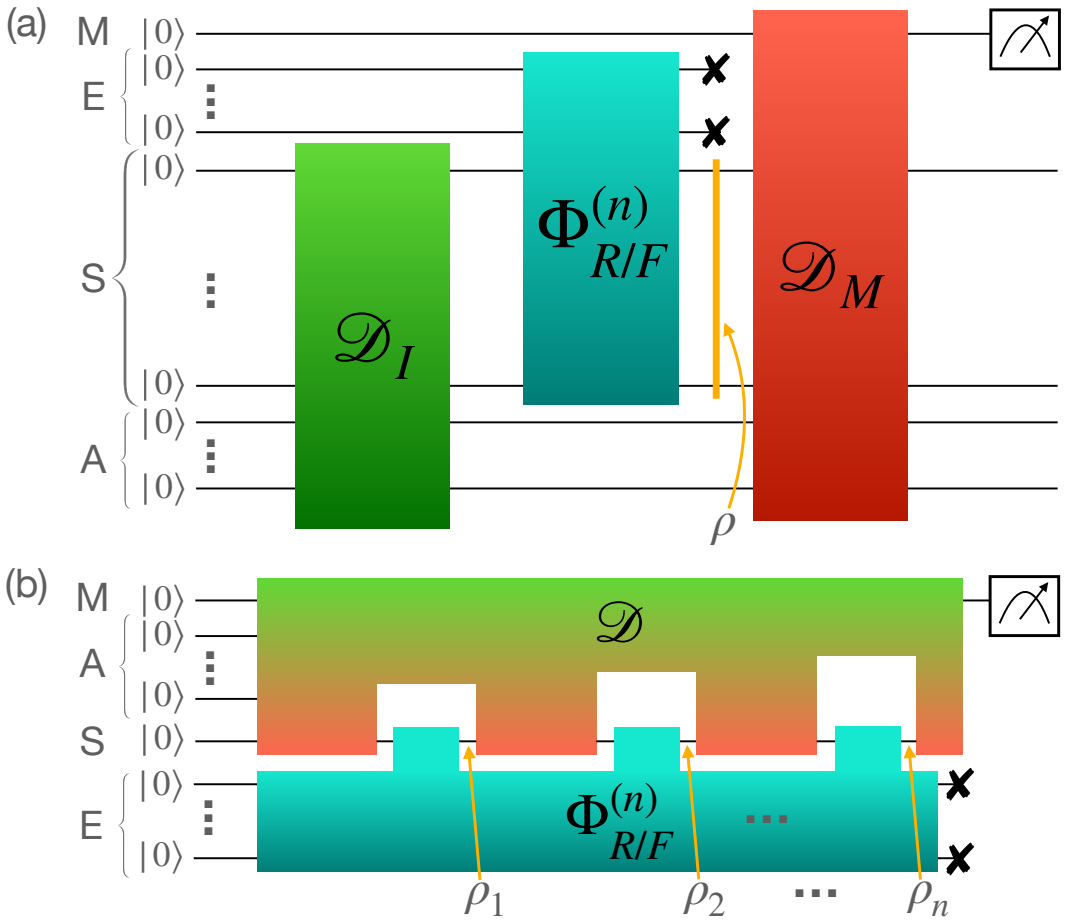


Figure 5.2: **General detection scheme for spatially correlated (a) and temporally correlated (b) noise.** Noise couples the system qubits S with the environmental qubits E. We use the same diagram to display both the real noise $\Phi_R^{(n)}$ and the generated fake noise $\Phi_F^{(n)}$, though these may physically correspond to different evolutions – e.g. real interaction with an environment vs. a quantum circuit. The discriminator has access to auxiliary qubits A and a measurement qubit M. Based on the measurement outcome on M, the map $\Phi^{(n)}$ is judged either real or fake. For spatially correlated noise (a), the generator applies an initialization map $\mathcal{D}_I$ on S+A and a measurement map $\mathcal{D}_M$ on S+A+M, finally measuring M. For temporally correlated noise (b), the discriminator applies the general map $\mathcal{D}$ that probes the system S at intermediate times, finally measuring M. In both cases, the discriminator has no access to the environmental qubits E.

schematic resemblance with an actual comb. Quantum combs are graphical representations of quantum circuits that repeatedly act on a system and on environmental ancillary qubits that sort of play the role of an entanglement reservoir, as in Fig. 5.2(b). Each "comb tooth" implements a quantum operation on the system at a certain time. The system state ρ_t^{in} goes into the

tooth from the left and the output state ρ_t^{out} exits from the right. All teeth are linked via the comb shaft, which represents environmental memory effects due to entanglement or other correlations between system and environment. Without external perturbations $\rho_t^{\text{in}} = \rho_{t-1}^{\text{out}}$, while in general the input and output states will be different if the system is probed, as in Fig. 5.2(b). For both spatial and temporal correlations, the discriminator can use all the resources offered by quantum mechanics to optimally discriminate the two processes [73]: these include entangling the system with a suitable number of ancillary qubits and performing generalized measurements (POVM) on the extended space. Nonetheless, the discriminator has no access to the environment responsible for the noisy evolution (see Fig. 5.2).

Our SuperQGAN can be mathematically described as the following min-max game

$$\min_{\Phi_F^{(n)}} \max_{\mathcal{D}} \text{Tr}\left[\mathcal{D} \star \left(\Phi_R^{(n)} - \Phi_F^{(n)}\right)\left(|0\rangle\langle 0|^{\otimes N_D}\right)\right] , \tag{5.1}$$

where N_D is the total number of qubits used by the discriminator, namely the sum of system qubits S, ancillary qubits A and measurement qubits M in Fig. 5.2, $\Phi_{R/F}^{(n)}$ are, respectively, the real (R) and fake (F) process maps, while $\mathcal{D}$ describes the set of operations performed by the discriminator. When the task is to discriminate between two processes as in Fig. 5.2(a), then $\mathcal{D} = (\mathcal{D}_I, \mathcal{D}_M)$ is a pair of CPTP maps, the initialization map $\mathcal{D}_I$ and the measurement map $\mathcal{D}_M$, and the *star-operation* in Eq. (5.1) refers to the composition map $\mathcal{D} \star \Phi = \mathcal{D}_M \circ \Phi \circ \mathcal{D}_I$, as in Fig. 5.2(a), with Φ being a CPTP map. When the task is to discriminate between two quantum "combs", as in Fig. 5.2(b), the discriminator's strategy can be entirely different: the discriminator can probe the system at all times $t = 1, \ldots, n$ and, by doing this, alter the state in S. In other terms, the input ρ_t^{in} in the t-th comb tooth will be different from the output ρ_{t-1}^{out} from the previous tooth. As a consequence, all the outputs in S at later times will be altered. The probe can be effectively implemented via measurements or via operations that couple the system S with the ancillary qubits A, owned by the discriminator, and the latter is the path we decided to follow since mid-circuit measurements are difficult to carry out on NISQ devices. All these operations are grouped into a process map $\mathcal{D}$, which is pictorially written via the "upside-down comb" in Fig. 5.2(b), while the combined action of $\mathcal{D}$ and $\Phi^{(n)}$ is represented by the *star-operation* $\star$ in Eq. (5.1).

Let us take a quick break and discuss an alternative approach to adversarial quantum noise sensing. We have seen in section 2.2.2 that there exists an isomorphism between CPTP maps Φ acting over d-dimensional quantum systems and the bipartite states $J^\Phi = (\mathcal{I} \otimes \Phi)(|\Omega\rangle\langle\Omega|)$ living in d^2-dimensional quantum systems, where $|\Omega\rangle = \sum_{i=1}^{d} |i,i\rangle / \sqrt{d}$. The bipartite states J^Φ are called Choi-Jamiołkowski (CJ) states [73], and completely characterize the channel Φ. Thus, if we can prepare the maximally entangled state $|\Omega\rangle$ and manage to apply the channel to half of it, we could in practice prepare the state J^Φ and try to learn it to investigate the map Φ. Since the CJ states are mixed,

resorting to the optimism-improved QGAN we have studied in the previous chapter would analogously lead to learning the noise affecting our computations. However, this approach can still be framed in the SuperQGAN game of Eq. (5.1). It just corresponds to a particular discrimination strategy: we need as many ancillary qubits as the system qubits in Fig. 5.2(a), and we need to fix the initialization circuit $\mathcal{D}_I$ such that the input for CPTP map $\Phi^{(n)}$ is $|\Omega\rangle$, namely $\mathcal{D}_I(|0\rangle\langle 0|^{\otimes N_D}) = |\Omega\rangle\langle\Omega|$. Most process tomography schemes [98] tackle the CJ state to carry out the characterization of the map at hand. However, we decided to directly model the channel Φ and avoid the preparation of $|\Omega\rangle$, to try to keep the need of quantum resources as low as possible.

Indeed, in the following applications we will approximate the general maps $\Phi_F^{(n)}$ and $\mathcal{D}$ via quantum circuits with a certain depth and with a certain amount of ancillary qubits. For spatially correlated noise, without any restriction on the possible operations, the maximization over $\mathcal{D}$ in Eq. (5.1) results in the diamond distance [73] between two channels $\left\|\Phi_R^{(n)} - \Phi_F^{(n)}\right\|_\diamond$, whose minimum is always zero with $\Phi_R^{(n)} = \Phi_F^{(n)}$. On the other hand, when either D or G have access to non-universal resources, the final value in (5.1) may be greater than zero and in general $\Phi_R^{(n)} \neq \Phi_F^{(n)}$. For instance, a restricted discrimination strategy without ancillary qubits will be computationally simpler, yet not general enough. On the other hand, deep quantum circuits with many ancillary qubits may be universal, yet numerically hard to *train*.

5.3 Applications

5.3.1 Random Unitary Channels

A random unitary operation describes a physical process that can be decomposed into the probabilistic application of one of a finite set of unitary operations [97]. It has been demonstrated that in this case if one has access to the environment introducing noise and can measure it obtaining classical information, then the corresponding noise process can be *corrected* [100]. In a real quantum computer this might be also the very likely case when one is dealing with a quantum circuit that is ideally a unitary transformation on some initial qubit states but in practice each gate of the circuit with some probability can correspond to a slightly different gate. Since the user has no access to such information, this introduce noise in the quantum computation that can be described by a random unitary map. Random operations can be also exploited to create quantum information scrambling as it was experimentally demonstrated in a 10-qubit trapped-ion quantum simulator [101]. Moreover, random operations can also allow to tailor the noise for scalable quantum computation via randomized compiling [102]. In Ref. [103] they exploit ML to classify single-qubit stochastic errors that can be written as a convex combination of unitary operations [103]. From the mathematical point of view, these transformations

are described by CPTP maps whose action on an input state ρ can be put in the form

$$\Phi_R(\rho) = \int U(s)\rho U(s)^\dagger p(s)ds, \text{ or } \Phi_R(\rho) \quad = \sum_k p_k U_k \rho U_k^\dagger \tag{5.2}$$

where in the continuous case $p(s)$ is a probability density and $U(s)$ is some trajectory in the space of unitaries $U(2^N)$, assuming ρ to be a N qubit state as usual in quantum computation, while in the discrete case $\{U_k\}$ and $\{p_k\}$ are, respectively, a set of unitary operators and the frequencies with which they occur in the system. That is, a random unitary map implements a particular non-unitary evolution of the system, where different unitary evolutions happen in a probabilistic manner. Let us remind here that, as we have already discussed in section 2.2.2 any convex linear combination of CPTP maps is again a valid quantum channel, and thus so are the random unitary channels in Eq. (5.2).

We focus on the simple case where the operators $U(s)$ are known and the task is to reconstruct the probability density $p(s)$. This task can be naturally expressed as SuperQGAN where the cost function (5.1) depends linearly on $p(s)$. The generator G can use a trial CPTP map $\Phi_F(\rho) = \int U(s)\rho U(s)^\dagger q(s)ds$ where the unitary operators $U(s)$ are those entering in (5.2), assumed to be known, while $q(s)$ must be learnt during the game. Even if the discriminator D can apply all possible detection schemes, this game *can* end with $p(s) \neq q(s)$. Mathematically speaking, the mapping $p(s) \mapsto \Phi_R$ is not injective and we may get $\Phi_F = \Phi_R$ even with $p(s) \neq q(s)$. This possibility can be formally checked by studying the CJ state of the random unitary map Φ_R, which is given by the convex combination $J^{\Phi_R} = \int J^{U(s)} p(s)ds$ of the CJ states $J^{U(s)}$ of the unitary channels $\rho \mapsto U(s)\rho U(s)^\dagger$, with the same probability density $p(s)$. In general the states $J^{U(s)}$ are not linearly independent and the perfect reconstruction of the random unitary map is not enough to learn $p(s)$. Notice that this is due to the unitary gauge freedom of the Kraus representation (Eq. (2.26)) of any channel. Indeed, considering a discrete random unitary channel for the sake of simplicity, we can rearrange it as

$$\begin{aligned} \Phi(\rho) &= \sum_k p_k U_k \rho U_k^\dagger \\ &= \sum_k (\sqrt{p_k} U_k)\rho(\sqrt{p_k} U_k)^\dagger \quad = \sum_k K_k \rho K_k^\dagger \,, \end{aligned} \tag{5.3}$$

where we simply reabsorbed the positive coefficient $\sqrt{p_k}$ in the unitaries U_k to reveal the Kraus decomposition of Φ.

In what follows, we will show the study carried on in [92] where, tackling a particular kind of random unitary channels we have been able to completely characterize a noisy map in terms of the probability distribution $\{p_k\}$. This special family of random unitary maps is that of *Pauli Channels*.

5.3.2 Pauli channels: spatial correlations

As we are going to see shortly, Pauli channels posses the desirable property of being in one-to-one correspondence with the probability distribution $\{p_k\}$ that appears in the second line of (5.2). This makes them the perfect choice for our generative agent, and this had an heavy weight in us deciding to study them. However, there are other favourable properties of Pauli channels that make them worth to investigate. Indeed, it is reasonable to assume that the average noise affecting a quantum circuit is a Pauli channel [104], which represents a very large family of random unitary maps. Although this class is not the more general one, one can show that a Pauli map can exceptionally well approximate any realistic noise without introducing new errors [105, 106].
Learning schemes for Pauli channels have been previously discussed in Refs. [107, 108]. Their methods rely on acquiring a large dataset of n-qubits measurement results that later get analysed to efficiently infer the Pauli error rates [108], or an averaged version of those [107]. In contrast, our procedure needs only single qubit measurements and uses machine learning techniques to produce error rates that get closer to the real ones after each measurement.

More specifically, Pauli channels belong to the family of random unitary channels described before in Eq. (5.2), where the unitary operators $U(s)$ belong to the *discrete* set of Pauli matrices. In the single qubit case, these are obtained by choosing $\{U(k)\} = \{\sigma_k\}$ with $\sigma_0 = \mathrm{I}$ and $\sigma_{1:3} = \{X, Y, Z\}$, *i.e.* they are convex combinations of Pauli evolutions. This single-qubit Pauli channel can be readily extended to the n-uses case, both in series (e.g., when the channel is applied n times to the same qubit) and in parallel (e.g., when multiple copies of the channel are used to process a string of input qubit states at the same time). We have studied the time and space correlated Pauli channels separately.

For spatial correlations, the channel can be represented as in Fig. 5.2(a), and maps n-qubit states $\rho^{(n)}$ to n-qubit states as follows

$$\Phi_p^{(n)}(\rho^{(n)}) = \sum_{\boldsymbol{k}} p_{\boldsymbol{k}}^{(n)} \sigma_{\boldsymbol{k}}^{(n)} \rho^{(n)} \sigma_{\boldsymbol{k}}^{(n)} \,, \tag{5.4}$$

where $\sigma_{\boldsymbol{k}}^{(n)} = \sigma_{k_1} \otimes \ldots \otimes \sigma_{k_n}$ are Pauli strings, and $\boldsymbol{k}$ is a multi-index. It is simple to check that the CJ states of different channels $J^{\sigma_{\boldsymbol{k}}}$ are linearly independent, so the mapping $p_{\boldsymbol{k}}^{(n)} \mapsto \Phi_p^{(n)}$ is bijective. Indeed by linearity of quantum maps, the Choi state J^{Φ} defined in Eq. (2.28) of a random unitary channel is the convex combination of those of the single unitaries. Thus, if they are linearly independent any of their convex combinations will yield a different channel. One can readily check that

$$\mathrm{Tr}\left[J^{\sigma_i}(J^{\sigma_j})^\dagger\right] \propto \left|\mathrm{Tr}\left[\sigma_i \sigma_j^\dagger\right]\right|^2 \propto \delta_{i,j} \,, \tag{5.5}$$

where $\delta_{i,j}$ is the usual Kronecker delta, and we used single qubit Pauli matrices, but the result is readily extendable to the case of Pauli strings. Orthogonality with respect to the Hilbert-Schmidt inner product is enough to ensure that we

can uniquely associate a Pauli channel to its probability distribution.

Notice that, when the probability can be factorized as $p_{\boldsymbol{k}}^{(n)} = \prod_{j=1}^{n} p(k_j)$, the channel has no spatial correlations, and $\Phi_p^{(n)}$ is a tensor product of independent channels on each qubit, whereas the above factorization property does not hold anymore when the noise is correlated. We can exploit the above relation to check for spatial correlations, by first learning $p(k)$ and then using it to check if $p_{\boldsymbol{k}}^{(2)}$ is factorized or not. Indeed, it suffices to show $\Phi^{(2)} \neq \Phi^{(1)} \otimes \Phi^{(1)}$ to rule out the absence of correlations.

Let us then present the SuperQGAN protocol to learn the $p_k^{(n)}$ of a general, parallel n-uses Pauli channel. The generative agent (G) tunes a fake distribution $q_{\boldsymbol{k}}^{(n)}$ to generate its copy $\Phi_q^{(n)}$ of the channel in Eq. (5.4). Particularly, G will simulate the Pauli channel $\Phi_q^{(n)}$ by acting separately on the probes register with all the Pauli words appearing in Eq. (5.4) and then weighting the results with the probabilities $q_{\boldsymbol{k}}^{(n)}$. For the sake of numerical simulations, the fake distribution will be parameterized with unbounded real parameters $\beta_{k_i^{(n)}}$ as

$$q_{k_i}^{(n)}(\boldsymbol{\beta}) = \frac{e^{-\beta_{k_i^{(n)}}}}{Z}, \qquad \text{with} \quad Z = \sum_{\boldsymbol{k}^{(n)}} e^{-\beta_{\boldsymbol{k}^{(n)}}} . \tag{5.6}$$

This obviously introduces a redundant degree of freedom, but also allows us to discard any constraint issue on their domain. The other agent, the Discriminator (D), will control both the measurement operator M and the initialization circuit I. The measurement operator is modelled as a parameterized quantum circuit (PQC) [54] with parameters $\boldsymbol{\theta_M}$, followed as in standard QGANs by a single-qubit measurement on the ancillary qubit M, see Fig. 5.2(a). The initialization circuit is also modelled as a PQC with parameters $\boldsymbol{\theta_I}$, entangling the system with ancillary qubits A. The resulting score function reads

$$\begin{aligned} S(\boldsymbol{\theta_I}, \boldsymbol{\theta_M}, \boldsymbol{\beta}) &= \mathrm{Tr}\left[M(\boldsymbol{\theta_M})\left(\Phi_p^{(n)}(\rho(\boldsymbol{\theta_I})) - \Phi_{q(\boldsymbol{\beta})}^{(n)}(\rho(\boldsymbol{\theta_I}))\right)\right], \\ &\hat{=}\ S_p - S_q \end{aligned} \tag{5.7}$$

where $M(\boldsymbol{\theta_M})$ is the POVM element and $\rho(\boldsymbol{\theta_I})$ is the global input state of the channels, *both* controlled by D. The linearity of the trace allows us to rewrite each of the two terms S_p and S_q in (5.7) as a sum of terms weighted by the respective distributions. Suppressing parameters dependencies and indexes we can write $S_p = \sum_{\boldsymbol{k}} p_{\boldsymbol{k}}^{(n)} S(\sigma_{\boldsymbol{k}})$ and analogously for S_q, with $S(\sigma_{\boldsymbol{k}})$ being the scores associated to the Pauli string $\sigma_{\boldsymbol{k}}$. We address the reader to the Methods Appendix C.1 for a detailed description of the circuits architectures. All the simulations in this section are based on the Yao.jl quantum computation package for Julia [42].

In Fig. 5.3 we show the success of our protocol in assessing a single-use ($n = 1$) Pauli channel. Among different figures of merit used to track the learning process, we stress the role of

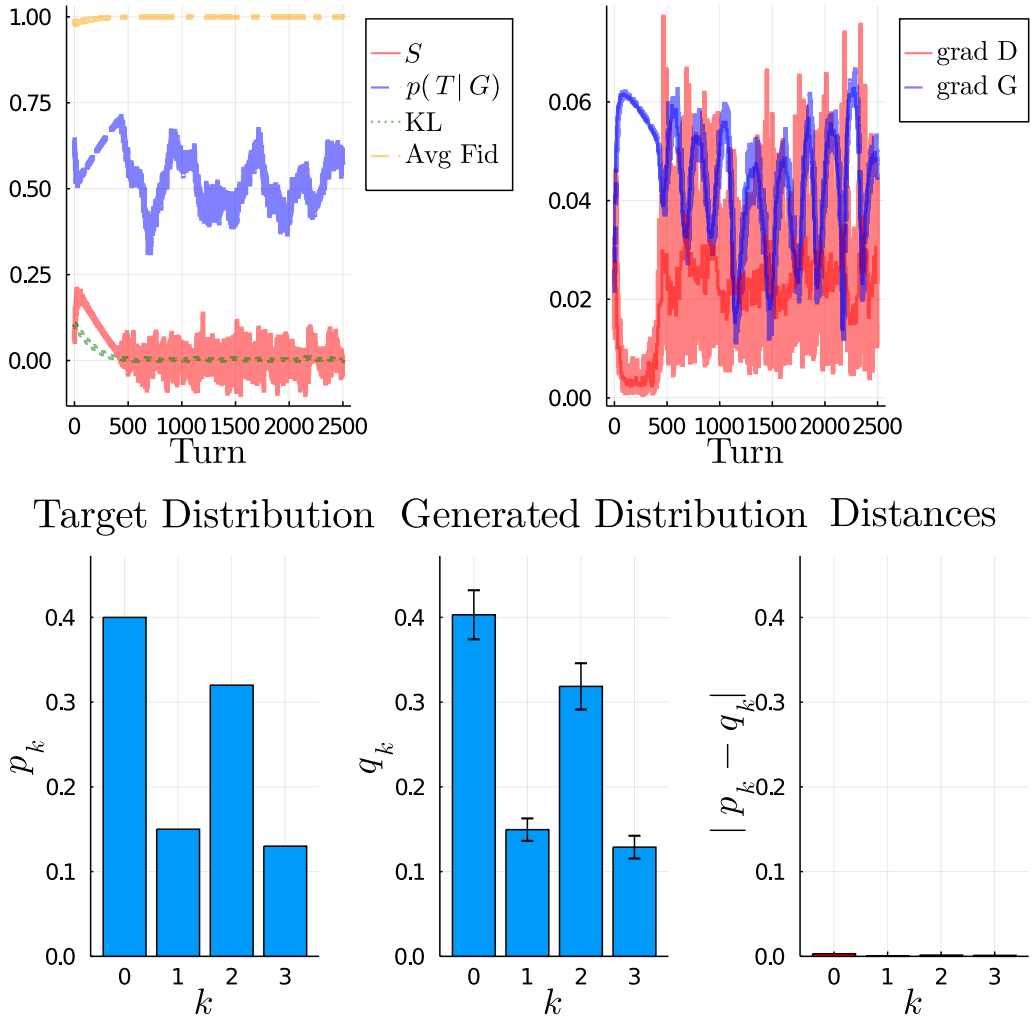


Figure 5.3: SuperQGAN **learning a single-use Pauli channel.** Top panel shows the training figures of merit (left), and gradients norms (right). Bottom one compares target and learnt distributions. The figures of merit that were tracked during training are: S, the score function (5.7); $p(T|G)$, G's objective function; KL, the Kullback-Leibler divergence between target and generated distributions; Avg Fid, the averaged fidelity between target and generated channels. The latter two quantities are defined in the main text above Eq. (5.10). The simulation was run mimicking real measurements taken with one hundred shots, and the error bars showed over the final generated distributions are obtained as the standard deviation over one hundred runs with different random initialization.

- the Kullback-Leibler divergence

$$KL(p,q) = \sum_k p_k \log(p_k/q_k)\,, \tag{5.8}$$

which is a measure of similarity between two probability distributions.

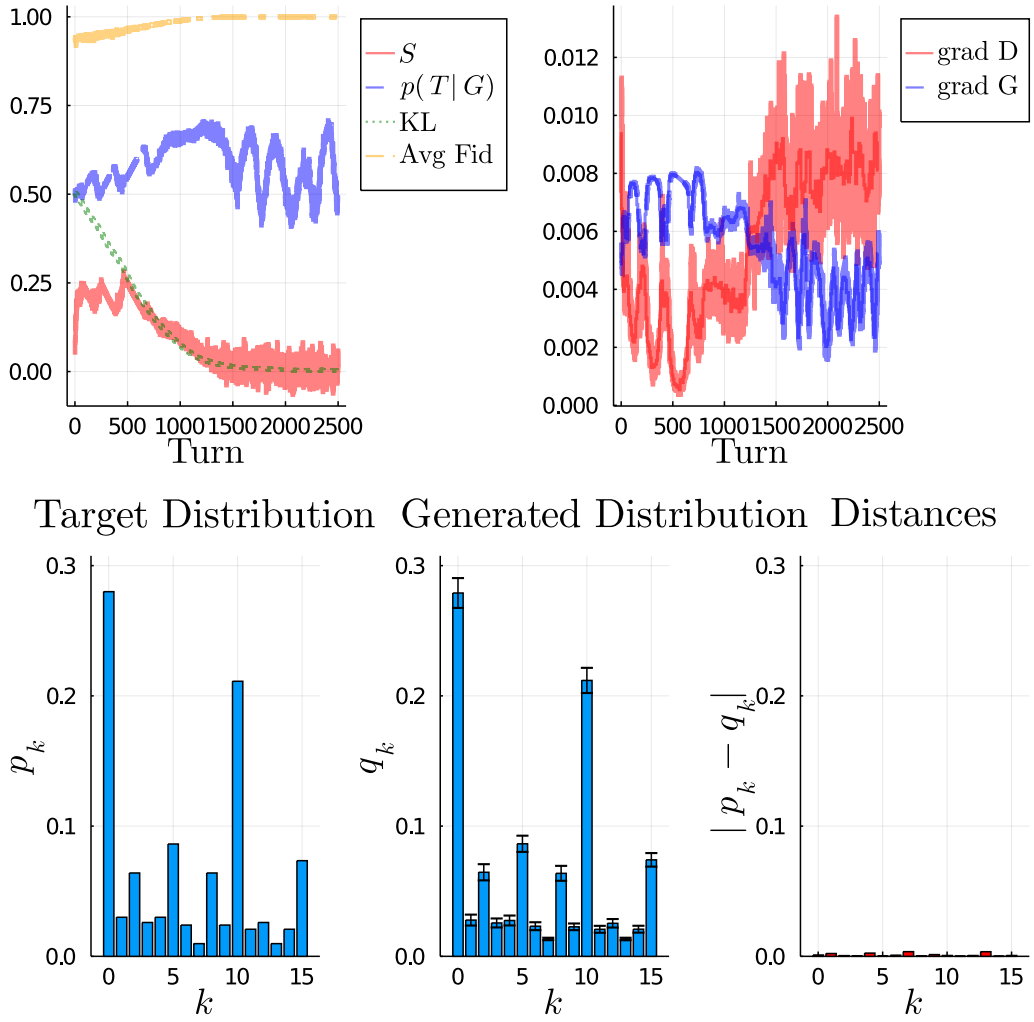


Figure 5.4: SuperQGAN **learning a two-uses spatially correlated Pauli channel.** Top panel shows the training figures of merit (left), and gradients (right). Bottom one compares target and learnt distributions, as described in Fig. 5.3. The target distribution is generated using the single use distribution of Fig. 5.3, with the correlation law (5.10) with $\mu = 0.5$. Measurement outcomes are simulated as the result of one hundred shots, whereas the uncertainty on the final generated distribution is due to averaging over one hundred different runs, as described in Fig. 5.3.

Notice that it is not a proper metric, it is not symmetric in the two distributions, and does not satisfy the triangle inequality. Nonetheless, it is a type of statistical distance, that measures the surprise stemming from using q as a model when the real distribution underlying the data is p.

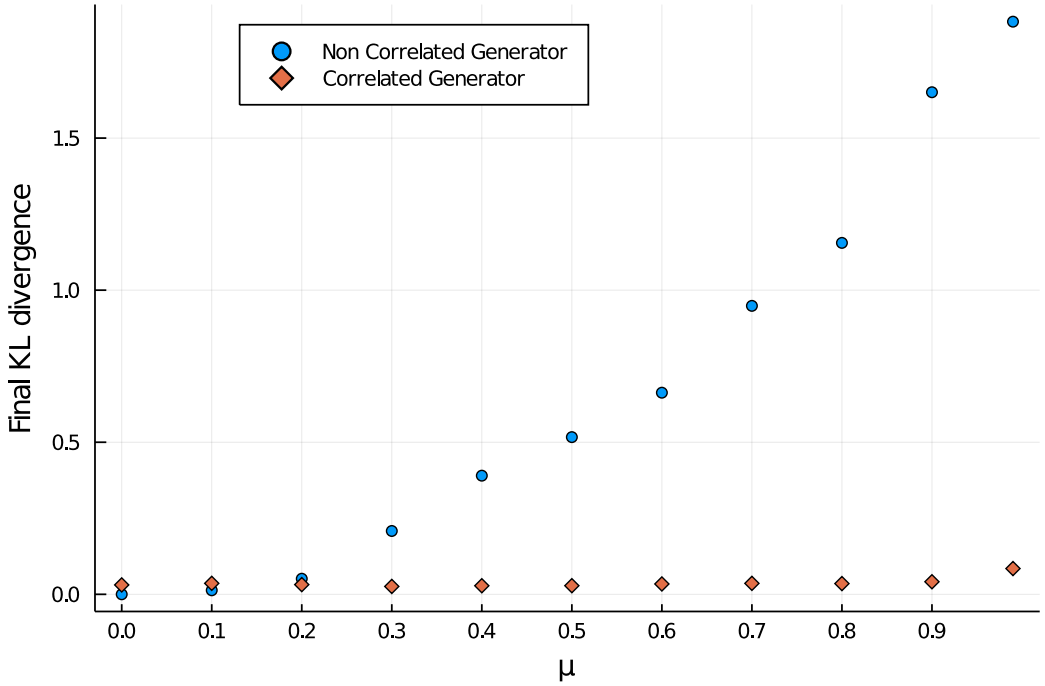


Figure 5.5: SuperQGAN **learning a correlated two-uses Pauli channel when G is not allowed to generate correlated distributions (blue dots) as opposed to the case when it can (red diamonds).** Each point corresponds to a full learning procedure, where the target distribution is obtained from the prior of Fig. 5.3 with the correlation of Eq. (5.10).

- the (discretized) averaged fidelity [24]

$$\bar{F}(\Phi,\Lambda) = \lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} F\left(\Phi(\rho_i), \Lambda(\rho_i)\right) , \tag{5.9}$$

  where $F(\rho,\sigma) = \left(tr\sqrt{\sqrt{\rho}\sigma\sqrt{\rho}}\right)^2$ is the usual fidelity between quantum states and the average is carried out over N output states of the given maps when the input states are as many Haar random input ones.

In order to test the performance of our setup when correlations may be present, we considered the multi-use scenario and we resorted to a particular form of spatial correlations described by

$$p_{ij} = (1-\mu)p_i p_j + \mu p_i \delta_{ij} , \tag{5.10}$$

which has been introduced in [109], and interpolates between non-correlated channel for $\mu = 0$ and a maximally correlated one for $\mu = 1$. Indeed, $\mu = 0$ corresponds to a factorized probability distribution, meaning that each use of the channel happens independently from the others, while $\mu = 1$ implies that whichever action (Pauli operator) is drawn on one use, gets repeated on the

other information carrier.
When the generator is allowed to tune a generic, *i.e.* correlated, distribution, the SuperQGAN is able to learn the target distribution no matter the amount of correlations. In Fig. 5.4 we show it with a two-uses example.

On the other hand, if G is constrained to generate non-correlated distributions only, *i.e.*is it is allowed to only tune a prior $q_k^{(1)}$ and use it to output $q_{\boldsymbol{k}}^{(n)} = \prod_{j=1}^{n} q_{k_j}^{(1)}$, the process fails. Particularly, as one would expect, in this case the final Kullback-Leibler divergence grows with μ, as shown in Fig. 5.5.

5.3.3 Pauli channels: temporal correlations

The reiterate interaction between a system and its surrounding environment typically gives rise to a non-Markovian evolution of the system [110]. If the system is probed at discrete times t_n, such evolution can be expressed as a quantum comb [111], as in Fig. 5.2b.

In our analysis we considered a simplified non-Markovian noise model based on Pauli channels with probability vectors $p_{\boldsymbol{k}}^{(n)}$, as in Eq. (5.4). While in Eq. (5.4) the probabilities $p_{\boldsymbol{k}}^{(n)}$ describe the (possibly spatially correlated) noisy operations on different qubits, here $p_{\boldsymbol{k}}^{(n)}$ model the noisy operations on a *single* qubit but at different times, *i.e.*

$$\Phi^n(\rho) = \sum_{\boldsymbol{k}} p_{\boldsymbol{k}}^{(n)} \sigma_{\boldsymbol{k}} \rho \sigma_{\boldsymbol{k}} \,, \tag{5.11}$$

where now $\sigma_{\boldsymbol{k}} = \sigma_{k_1}\sigma_{k_2}\cdots\sigma_{k_n}$. In other terms, in Eq. (5.4) k_s refers to the Pauli operation applied to the s-th qubit, while in Eq. (5.11) k_t refers to the Pauli operation applied to a single qubit at the t-th discrete iteration. One way to express the above circuit as the comb of Fig. 5.2b is to assume that during the t-th iteration, and for all iterations $t = 1, \ldots, n$, the environment is measured with a four-outcome POVM, and depending on the measurement outcome k_t a Pauli operation σ_{k_t} is applied onto the system. The probability vector then models the joint probability of all possible POVM outcomes. If noise is temporally uncorrelated, then the probability is factorized $p_{\boldsymbol{k}}^{(n)} = \prod_{j=1}^{n} p(k_j)$. If noise is Markovian, then $p_{\boldsymbol{k}}^{(n)} = p(k_1)\prod_{j=2}^{n} p(k_j|k_{j-1})$. For more general probabilities, this model allows to describe non-Markovian noise.

In a similar way to what we have done for spatial correlations, we can define a SuperQGAN to learn $p_{\boldsymbol{k}}^{(n)}$: as usual, G tries to reproduce the distribution and D tries to discriminate between the real noise and the generated one. Although spatial and temporal Pauli correlations can both be modelled via $p_{\boldsymbol{k}}^{(n)}$, the discrimination strategy can be entirely different. Indeed, the most general discrimination strategy for temporal correlations is the one depicted in Fig. 5.2, where D inserts some probing operations at the intermediate times $t = 1, \ldots, n$ and, depending on the outcomes, decides whether the noisy channel was real or generated. In general, the probe alters the state, due to the wavefunction collapse (Eq. (2.10)), thus influencing all the future evolution.

We now proceed to show the results of our numerical test of SuperQGAN's performance in the temporal correlations case. Again, the interested reader may find a detailed description of the SuperQGAN architecture, as well as of the training procedure, in the Methods Appendix C.1. In the single-use case the setup coincides with the one exploited for spatial correlations, as do the results. Hence, we show a correlated example given again in terms of the correlation law (5.10), which can be readily interpreted as a time correlation once the multi-indices are treated as time labels. Particularly, in Fig. 5.6 we show the success of our protocol in a two-uses Pauli channel. Assuming to know in advance the model of correlation occurring in the quantum map, we can devise G in such a way that it only has to tune the correlations parameters, rather than the whole distribution. Then, we have used the learnt $n = 1$ error rates p to tackle the $n = (2, 3, 4, 5, 6)$ temporally correlated Pauli channels with a generator that only controls μ. The number of turns needed to achieve convergence is found to be independent from n, as shown in Fig. 5.7.

5.3.4 Quantum metrology

Not only noise sensing, but also quantum metrology [112] can be rephrased as a SuperQGAN with δ-like probability distribution in Eq. (5.2), *i.e.* $p(s) = \delta(s - \bar{s})$. In other terms, we have a mapping implementing a unitary evolution $\rho \to U(\bar{s})\rho U(\bar{s})^\dagger$ and the metrology task is to estimate $\bar{s}$. Efficient quantum algorithms that fully exploit quantum effects to maximize the estimation precision typically employ either adaptive strategies or parallel applications of the unitary channel $U(\bar{s})^{\otimes n}$ on an entangled state. Similar strategies are also needed when the parameter s to be estimated is not fixed, but rather distributed according to some probability $p(s)$. In particular, we studied a paradigmatic model of quantum metrology, namely the Mach-Zehnder-type interferometer [113, 114], whose unitary evolution can be written as

$$U(s) = \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i s} \end{pmatrix} . \tag{5.12}$$

We assume that we can exactly express the parameter s by using m-bits as $s = \sum_{j=1}^{m} s_j/2^j$, where $0 \leqslant s < 1$ and $s_j \in \{0, 1\}$, i.e. $s \equiv s_b = b/2^m$ for an integer $b < 2^m$. When this assumption is not satisfied, we may get a reconstruction error. For instance, let us suppose to run the phase estimation algorithm for general s using an $m + 1$ qubit register. If s_b is the best m-bit approximation of s, then the algorithm will output $b' \neq b$ with probability $p_r(b'|b) = |2^{-m}(1 - e^{2^m i\delta})/(1 - e^{i\delta})|^2$, where $\delta = 2\pi(s - s_b - s_{b'})$ [24]. The distribution $p_r(b'|b)$ is peaked around $b' = b$ or around $b' = b \pm 1$ when $2^m s$ is close to two different integers, so the reconstruction error is small and mostly limited to nearby values. In our analysis, we fixed m and consider the error due to the finite m as an imperfect reconstruction of $p(s)$.

The number n of independent applications of $U(s)$ needed to reconstruct s with m-bit precision increases with m [115]. To simplify our treatment, here

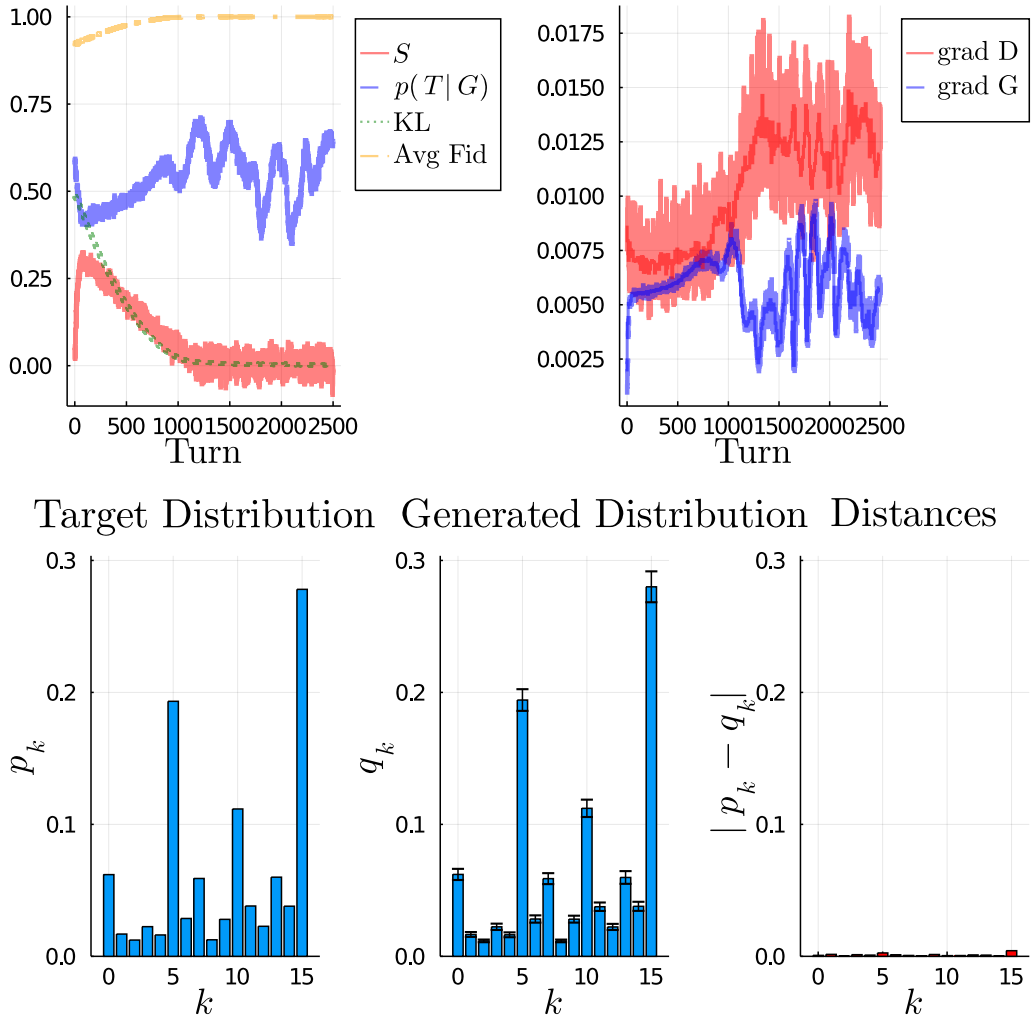


Figure 5.6: SuperQGAN **learning a two-uses temporally correlated Pauli channel.** Top panel shows the training figures of merit (left), and gradients (right). Bottom one compares target and learnt distributions, as described in Fig. 5.3. The target distribution is generated using a random single-use prior, using the correlation law (5.10) with $\mu = 0.5$. As in Fig. 5.3, one hundred shots were used to simulate real measurement outcomes, and error bars over the learnt distribution correspond to the standard deviation of one hundred randomly initialized runs.

we assumed that m is *fixed*, so $p(s)$ becomes a discrete distribution with 2^m entries, and we consider n parallel applications of $U(s)$. As a result, we get the following random unitary channel

$$\Phi_R^{(n)}(\rho) = \sum_{b=0}^{2^m-1} p(s_b) U(s_b)^{\otimes n} \rho\, U(s_b)^{\otimes n\dagger}\,, \tag{5.13}$$

where $s_b = b/2^m$ as above and b is an integer. The CJ state of each unitary

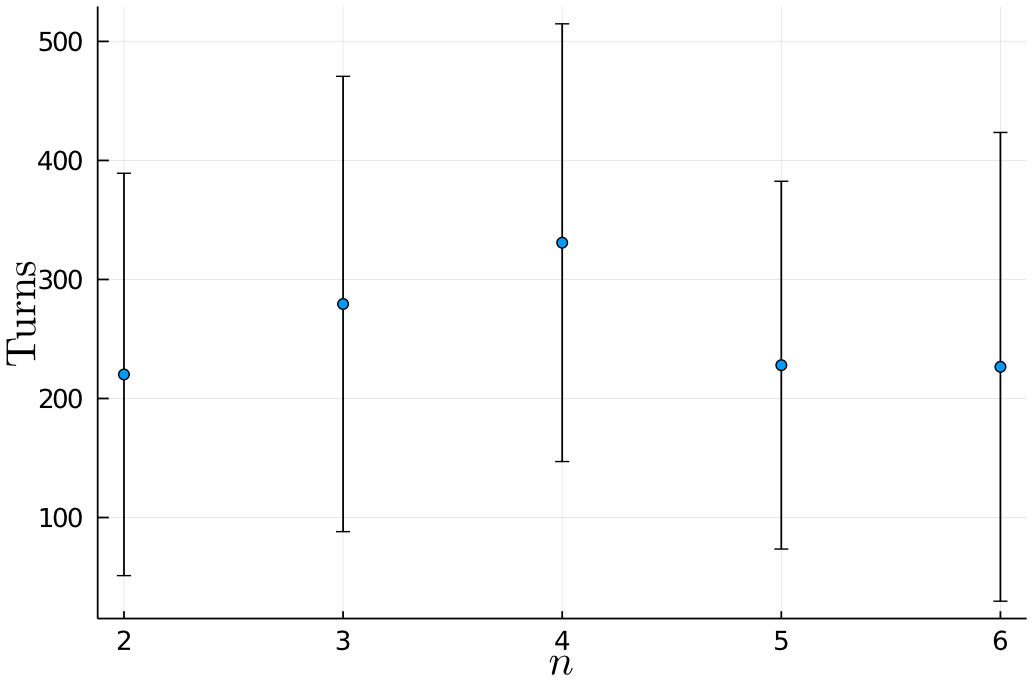


Figure 5.7: **Total number of turns needed to achieve averaged fidelity greater than threshold value of 0.999 between target and generated channels.** Each dot corresponds to the mean over 10 runs of the modified SuperQGAN whose generator knows the correlation model of Eq. (5.10) and the $n = 1$ probabilities p. Although the sample size is small, we observe that the number of iterations to achieve convergence does not increase with the number n of channel uses, hence supporting the successful feasibility of our protocol for larger n.

channel $U(s)$ is a tensor product of a maximally entangled pure state $|\chi_s\rangle^{\otimes n}$, with $|\chi_s\rangle = (|00\rangle + 2^{2\pi i s}|11\rangle)/\sqrt{2}$. To check for their linear independence, we may focus on the Gram matrix[1] with the Hilbert-Schmidt product, $G_{st} = \mathrm{Tr}[\chi_s^{\otimes n}\chi_t^{\otimes n}] = |\tilde{G}_{st}|^2$, where $\chi_s = |\chi_s\rangle\langle\chi_s|$ and $\tilde{G}_{st} = \langle\chi_s|\,\chi_t\rangle^n$. The Gram matrix has zero determinant, and hence at least a zero eigenvalue, when the matrices $\chi_s^{\otimes n}$ are linearly dependent. The matrix $\tilde{G}$ can be diagonalized via a discrete Fourier transform, obtaining the eigenvalues $\tilde{g}_k = 2^{m-n}\sum_{\ell=0}^{n}\binom{n}{\ell}\delta_{\ell,k}^{(2^m)}$ where $\delta_{ab}^{(c)}$ is 1 if $a = b \pmod c$ and 0 otherwise, and $k = 0, \ldots, 2^m - 1$. The eigenvalues of G are then obtained via convolution $g_k = 2^{-m}\sum_u \tilde{g}_u\tilde{g}_{k\oplus u} = 2^{m-2n}\sum_\ell \binom{n}{\ell}\binom{n}{\ell\oplus k}$, where $\oplus$ is the addition modulo 2^m and $\binom{n}{k} = 0$ for $k > n$. Therefore, when $n < 2^m/2$ at least one eigenvalue g_k is zero and, accordingly,

[1]The Gram matrix G is a square matrix that represents the inner products between a set of vectors $\{\boldsymbol{v}_i\}$. Its elements read $G_{ij} = \langle \boldsymbol{v}_i, \boldsymbol{v}_j\rangle$, where $\langle\cdot,\cdot\rangle$ is the appropriate inner product of the space the vectors belong to. One important use of the Gram matrix is to determine whether the set of vectors is linearly independent, which can be determined by checking if the determinant of the Gram matrix is non-zero.

the mapping (5.13) is not injective, namely two channels $\Phi_R^{(n)}$ may be equal even with different distributions $p(s)$. According to our analysis, we need a number of probes satisfying

$$n \geqslant 2^{m-1}\,, \tag{5.14}$$

to be sure that the reconstruction of $\Phi_R^{(n)}$ allows a unique reconstruction of $p(s)$.

We then performed a numerical study using a SuperQGAN, where G tries to generate a fake channel with the same mathematical form of Eq. (5.13), but different probability $q(s)$ instead of $p(s)$. To simplify the numerical treatment, G parametrizes its distribution again as in Eq. (5.6),

$$q(s) = e^{-\beta_s}/Z, \qquad\qquad Z = \sum_s e^{-\beta_s}\,, \tag{5.15}$$

with real parameters β_s. The SuperQGAN setup is analogue to that of spatial correlation learning outlined in Sec. 5.3.2 although we do not need to test all possible combination of unitaries since only tensor products appear. No differences in training performance are expected, and indeed, when one has enough resources, namely when (5.14) is satisfied, G is always able to learn the correct distribution $p(s)$. In table 5.1 we show the final Kullback-Leibler divergence between $p(s)$ and $q(s)$ after the averaged fidelity between real and generated channels has reached the threshold value $f_{\text{tr}} = 0.99999$. As one can see, sub-optimal values of n lead to a learnt distribution $q(s)$ that is not converging to the target one.

m \ n	2^{m-1}	$2^{m-1}-1$
2	0.000065(4)	0.088(2)
3	0.00016(1)	0.038(1)
4	0.00012(8)	0.0015(1)

Table 5.1: **Final values of Kullback-Leibler divergence KL(p, q)** between target distribution $p(s)$ appearing in Eq. (5.13) and G's generated one $q(s)$. The SuperQGAN is stopped as soon as the averaged fidelity between target and fake channels gets larger than 0.99999. Values reported here refer to an average over 10 runs with fixed targets and different (random) parameters' initializations.

Chapter 6

Inductive Biases in QML: the Power of Equivariance

While in Chapter 4 we have dealt with finding a training strategy for learning mixed states with QGANs, which we then exploited to design a noise sensing architecture that resorts to generative-adversarial learning, the SuperQGANs described in Chapter 5, in this chapter we are going to leave the particular paradigm of quantum GANs and tackle the design of learning models from a general perspective.
In the first part of this chapter, Sections 6.1-6.3, we are going to introduce a general framework for designing quantum learning models that are not completely agnostic to the problem they will try to solve, but rather are endowed with *inductive biases* regarding the symmetries of the data they need to process. This framework, called *geometric quantum machine learning* (GQML) is the result of the work carried out during the 2022 Quantum Computing Summer School at the Los Alamos National Laboratories, which has been drafted in [116].
In the last part, Section 6.4, we are going to apply the recipe for informed quantum learning model to build a classifier to distinguish different phases of matter, and test it against a problem agnostic one to showcase the benefits of GQML.

6.1 The role of inductive biases

When we discussed learning models architectures, particularly the quantum ones of section 3.3, we said that one of the main avenues that is currently being followed by the QML community for choosing the parameterized quantum circuit structures that are eventually trained is that of *hardware efficiency*. We commented on how current quantum devices, belonging to the so-called NISQ era, are noisy and small, and how these limitations make it so that circuital ansätze over a certain depth do not allow for reliable quantum computation. However, besides knowing that we need to keep the depth, and the connec-

tivity, of our quantum learning models low, *hardware efficient ansätze* do not come with suggestions on which gates to use, or on how to arrange them. Nonetheless, choosing a PQC architecture is not simply a matter of taste, and there are indeed other criteria than just NISQ-friendliness. Among these, a paramount one are dataset *symmetries*.

In classical machine learning, recognizing the symmetries that underlie the targeted dataset has been crucial, and has led to great advances. For example, knowing that an image of a cat will still be an image of a cat even if we move the pixels around, e.g. by translating, rotating or flipping the whole image, helps explain why convolutional neural networks [117] are so effective at image classification: they process images in a way that is symmetrical with respect to translation [118].

In recent years, the role of symmetries in machine learning has been studied in problems that involve more general symmetry groups than just translations. This has led to the growth of a field known as *geometric deep learning* [118]. At the core of this field lies the idea that incorporating prior knowledge of symmetry into a model can effectively constrain the search space and make the learning task easier. In fact, symmetry-aware models have been shown to perform and generalize better than models that are agnostic to the problem in a wide range of tasks [118–125]. As a result, there has been a lot of work on developing a mathematically rigorous framework for designing symmetry-embedded models using the tools of representation theory. This has led to the development of so-called *equivariant neural networks* (ENNs) [126–130], which have the key property that their action commutes with that of the symmetry group. In other words, applying a symmetry transformation to the input and then passing it through the ENN produces the same result as passing the raw input through the ENN and then applying the transformation.

Equivariance[1] is the mathematical property of preserving the symmetries of the feature vectors belonging to the dataset we want to process through a deep (equivariant) neural network. The toolbox that enables geometric deep learning is representation theory [118], whose machinery allows to find ways to craft equivariant layers that, stacked up, eventually build an ENN. The convolutional neural network (CNN) [117] is a well-known example of an equivariant architecture, commonly used in image and signal processing. In CNNs, the relevant symmetry group is the translation group in $\mathbb{R}^2$, and it has been shown that the convolution and pooling layers of a CNN are equivariant to this group [127]. There have been attempts to generalize CNNs to other groups and data [126], including homogeneous ENNs for spherical images and molecular data [131–134], and non-homogeneous architectures such as graph neural networks [120, 135, 136]. In the first case, the underlying symmetry groups are the one of rotations $\mathbb{SO}(3)$ and the Euclidean one $\mathbb{E}(n) = \mathbb{R}^n \rtimes \mathbb{O}(n)$, in the second case instead the relevant group is that of permutations S_n. Addi-

[1] Sometimes referred to as *covariance*. Here we will stick to the term that has been adopted by the ML community.

tionally, more advanced representation-theoretic techniques have been used to develop steerable and gauge-equivariant CNNs on general manifolds [129, 137]. It has also been demonstrated that equivariant layers can be constructed using either real or Fourier space [121].

The geometric deep learning field is growing rapidly, and over the last two years a lot of effort has been devoted to finding designing strategies for equivariant network layers [124, 138, 139]. A theoretical analysis of the benefits of using ENNs in terms of improved training and generalization error can be found in [125, 140–142], whereas expressibility and universality of ENNs have been investigated in [143–147].

6.2 Geometric Quantum Machine learning

It just feels natural to import the fundamental ideas of geometric deep learning into quantum machine learning (QML) to exploit the inductive biases coming from the symmetries of the quantum problem at hand in order to craft better models ansätze than the hardware efficient ones, and this process is in full swing. Some proposals have already been put forward. Consider for example the problem of classifying states that present either a large or a low amount of multipartite entanglement [148–150]. Since the entanglement spectrum cannot be altered by local unitaries, we can employ models whose outputs are invariant under the action of any such transformation [151]. In this rush for the development of *Geometric Quantum Machine Learning* (GQML) [151–156] we contributed by establishing a theoretical framework to design equivariant ansätze for parametric quantum circuits (see Fig. 6.1), and by numerically studying the performance of equivariant quantum models against problem-agnostic ones.

In order to be able to explain how to build equivariant quantum models, we first need to introduce some basic definitions and concepts from group and representation theory. The interested reader can find more details in the established textbooks [157–159], or in [160], where representation theory is presented from a QML practitioner point of view.

6.2.1 Basic concepts from representation theory

It should be clear by now that the aim of GQML is to exploit the symmetries underlying the (quantum) dataset at hand to build learning models that preserve them. We will label general symmetry groups as G, and we here briefly recall their fundamental properties. A group G is a set of elements equipped with an internal operation, usually dubbed product that satisfy

- *Closure*: the group must be closed under the operation that defines it, *i.e.* $g_1 \circ g_2 \in G \ \forall g_1, g_2 \in G$ where we denoted the group product as $\circ$. In the following we will always omit that and simply write $g_1 \circ g_2 = g_1 g_2$.

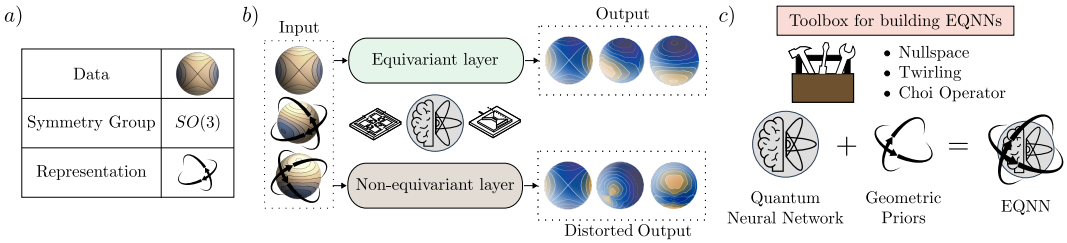


Figure 6.1: **Schematic workflow of GQML [116].** a) In GQML, we begin by identifying the symmetry group(s) that do not change the labels of the data. For example, if the data can be depicted on a 3D sphere and the labels remain unchanged when the action of $SO(3)$ is applied, then $SO(3)$ is the symmetry group. b) It has been observed in both classical and quantum machine learning that models with equivariant layers often perform better than those with non-equivariant architectures. Equivariance means that applying a rotation to the input data and sending it through the layer is the same as first sending the data through the layer and then rotating the output. On the other hand, using either raw or rotated data as input in a non-equivariant layer often results in distorted, non-rotationally related outputs. c) In this research, we present a set of methods for creating equivariant quantum neural networks (EQNNs) that can be used to easily build quantum models with strong geometric assumptions.

- *Identity Element*: there must exist in G an unique element e, called the *identity* element, s.t. $eg = ge = g \ \forall g \in G$.
- *Inverse Element*: for any element g of the group, there must exist a unique element, called the *inverse* element g^{-1}, such that $gg^{-1} = g^{-1}g = e$.
- *Associativity*: the operation that defines the group must be associative, which means that the order in which the elements are combined does not affect the result, namely $(g_1 g_2)g_3 = g_1(g_2 g_3)$.

A *symmetry group* is then a group G whose elements are associated with symmetry transformations of the system under consideration.

The way we describe the actual action of symmetry group elements onto a system is by their *representation*. Representation theory lies at the core of geometric deep learning and, as such, of geometric quantum machine learning, thus here we recall some of its basic concepts.
Assume that the system state is described by vectors living in a Hilbert space $\mathcal{H}$, then:

Definition 1 (Representation)**.** *A representation, dubbed* $(R, \mathcal{H})$ *or just* R *for short, of a group* G *on a vector space* $\mathcal{H}$ *is a homomorphism* $R : G \to GL(\mathcal{H})$ *from the group* G *to the space of invertible linear operators on* $\mathcal{H}$*, that preserves the group structure of* G*.*

Specifically, a group homomorphism R satisfies

$$R(g_1)R(g_2) = R(g_1 g_2) \quad \forall g_1, g_2 \in G\,. \tag{6.1}$$

This implies that, for all $g \in G$, the representation of its inverse is the inverse of its representation, $R(g^{-1}) = R(g)^{-1}$, and the representation of the identity element e is the identity operator on $\mathcal{H}$, $R(e) = \mathbb{1}_{\dim(\mathcal{H})}$. Given a representation, it is relevant to define its commutant.

Definition 2 (Commutant)**.** *Given a representation R of G, we define the commutant of R as the set of bounded linear operators on $\mathcal{H}$ that commute with every element in R, i.e.,*

$$\mathfrak{comm}(R) = \{H \in \mathcal{B}(\mathcal{H}) \,|\, [H, R(g)] = 0 \;\forall g \in G\}\,. \tag{6.2}$$

Let us furthermore introduce the following definitions about representations:

- If a representation R is such that $R(g) = \mathbb{1}\;\forall g \in G$, the representation is called *trivial*.
- A representation is *faithful* if it maps distinct group elements to distinct elements in $\mathcal{H}$. As an example of unfaithfulness, the *trivial representation* maps all group elements to the identity in $\mathcal{H}$.
- Two representations R_1 and R_2 are *equivalent* if there exists a change of basis W such that $WR_1(g)W^\dagger = R_2(g)$ for all $g \in G$, in which case we denote $R_1 \cong R_2$.
- A *subrepresentation* is a subspace $\mathcal{K} \subset \mathcal{H}$ that is invariant under the action of the representation, i.e., $R(g)\,|w\rangle \in \mathcal{K}$ for all $g \in G$ and $|w\rangle \in \mathcal{K}$. The group can then be represented through $R|_{\mathcal{K}}$, the restriction of R to the vector subspace $\mathcal{K}$. A subrepresentation $\mathcal{K}$ is non-trivial if $\mathcal{K} \neq \{0\}$ (the zero vector) and $\mathcal{K} \neq \mathcal{H}$.

Most of the interesting symmetry group in physics are non finite. This means that G is not a finite set of elements, but rather a continuous space. We will assume any non finite group G to be a (compact) *Lie group*, with its associated *Lie algebra* $\mathfrak{g}$. For us, a Lie group will be defined by the relation $e^{\mathfrak{g}} = G$, *i.e.* $\mathfrak{g} = \{a | e^a \in G\}$. In particular, if G has a representation R then $\mathfrak{g}$ has a representation r given by the differential of R, that is, given $a \in \mathfrak{g}$, $R(e^a) = e^{r(a)}$. Moreover, a very important fact about representations for QML practitioners is that if a group G is finite or compact, its representations can be chosen to be unitary [161], and we will always assume that this choice has been made in the following.

The representation we will use and mention the most in the following is the so-called *adjoint representation*, denoted Ad_R, which describes how the group acts on density matrices (and other bounded operators). A unitary representation R on $\mathcal{H}$ induces an action on $\mathcal{B}(\mathcal{H})$, given by

$$\mathrm{Ad}_{R(g)}(\rho) = R(g)\rho R(g)^\dagger, \ \forall g \in G, \rho \in \mathcal{B}(\mathcal{H})\,. \tag{6.3}$$

Note that for the case of Lie groups, the adjoint representation also exists at the Lie algebra level and is given by $\mathrm{ad}_{r(a)}(\cdot) = [r(a), \cdot]$.

Lastly, let us introduce the following distinction between symmetry groups.

Definition 3 (Inner and outer symmetries)**.** *Given a composite Hilbert space, such as the one describing a qubit register, we call a representation of a group an* ***inner*** *symmetry if it acts locally on each subsystem, and an* ***outer*** *symmetry if it permutes the subsystems.*

Consider for example an n-qubit register, for the group $\mathbb{SU}(2)$, whom the single qubit gates belong to, the tensor product representation $R(g \in \mathbb{SU}(2)) = g^{\otimes n}$ is an inner symmetry, locally rotating each qubit by the same amount. An example of an outer symmetry is instead the qubit-permuting representation of S_n, given by $R(g)\bigotimes_{j=1}^n |\psi_j\rangle = \bigotimes_{j=1}^n \left|\psi_{g^{-1}(j)}\right\rangle$.

Given a quantum machine learning problem, that we will consider to be a supervised one for the sake of exposure simplicity, we want to first identify the underlying symmetries and then devise a model that preserves them. Let us well define the problem first.

6.2.2 Quantum Model for Classification Tasks

We are going to study a simple classification problem where we are given a dataset $\mathcal{D} = \{\rho_i, y_i\}_{i=1}^M$ of quantum data points ρ_i with their associated scalar labels y_i and we want to infer the underlying relation $F : \mathcal{X} \to \mathcal{Y}$ that assigns the latter to the former. We denoted $\mathcal{X}$ the data points space, which is some subset of the space of density operators acting on the Hilbert space describing the quantum system at hand, while we used $\mathcal{Y}$ for the labels space. Notice that this framework holds regardless of the data being classical and embedded into a quantum computer (CQ) [162] or straight up quantum (QQ) [150] As thoroughly described in section 3.3, we now need devise a quantum learning model $f_{\boldsymbol{\theta}} : \mathcal{X} \to \mathcal{Y}$ to approximate the true relation F. As described there, our quantum learning model is a so-called quantum neural network (QNN), composed of a sequence of layers of quantum CPTP maps. We resort to general QNNs rather than PQCs because in what follows we will need to consider operations that do not preserve the number of qubits they act on. Thus, let us recall the structure of a QNN

$$\Phi_{\boldsymbol{\theta}} = \mathcal{N}^D_{\boldsymbol{\theta}_D} \circ \cdots \circ \mathcal{N}^1_{\boldsymbol{\theta}_1}\,, \tag{6.4}$$

with each $\mathcal{N}^l_{\boldsymbol{\theta}_l}$ being the l-th layer CPTP map. Sticking to our supervised classification example, the QNN will output its predicted labels as

$$\ell_{\boldsymbol{\theta}}(\rho) = \mathcal{C}(\{\mathrm{Tr}[\Phi_{\boldsymbol{\theta}}(\rho)O_j]\}_j)\,. \tag{6.5}$$

Where $\rho \in \mathcal{X}$ is the input state, the set of observables $\{O_j\}_j$ are measured on the QNN output $\Phi_{\boldsymbol{\theta}}(\rho)$ and their expectation values are fed to some classical post-processing function $\mathcal{C}$. To train the model, we minimize the empirical loss

$$\widehat{\mathcal{L}}_{\boldsymbol{\theta}}(\{\rho_i, y_i\}_{i=1}^M) = \frac{1}{M}\sum_{i=1}^{M} \mathcal{F}(\ell_{\boldsymbol{\theta}}(\rho_i), y_i)\,, \tag{6.6}$$

defined in terms of some problem-dependent function $\mathcal{F}$, that in the case of classification might for example be the squared error $\mathcal{F}(\ell_{\boldsymbol{\theta}}(\rho_i), y_i) = (\ell_{\boldsymbol{\theta}}(\rho_i) - y_i)^2$. Once the hybrid quantum-classical training has been carried out, we will use the optimal parameters $\boldsymbol{\theta}^*$ to classify unseen data.

Now that we have a solid setting, we are ready to introduce equivariance into QML to promote it to GQML.

6.2.3 Equivariant QNNs

First of all, what is a *symmetry* of a quantum machine learning problem?

Definition 4 (G-invariance)**.** *Given a group G acting via some representation R on the data points $\rho_i \in \mathcal{X}$, we say that it is a symmetry group of the QML task at hand if its action leaves the true labels $y_i = F(\rho_i)$ unchanged, i.e. if*

$$F(R(g)\rho R(g)^\dagger) = F(\rho),\ \forall g \in G\, \forall \rho \in \mathcal{X}\,. \tag{6.7}$$

We call this property G-invariance.

Once one such G-invariance has been identified in the QML task under consideration, it becomes natural to try to embed this prior information (inductive biases) about the problem into the learning model. In the case of classifying QNNs, this translates into finding appropriate parameterized quantum maps $\mathcal{N}^l_{\boldsymbol{\theta}_l}$ and measurement operators $\{O_j\}$. The injection of inductive biases in the model make it so that the latter will only explore a *relevant* subset of all possible functions $f : \mathcal{X} \to \mathcal{Y}$. Models with strong inductive biases often perform better than agnostic ones [163–165]. GQML scope is to provide a framework for incorporating prior geometric knowledge into the learning model in order to improve its trainability, data requirements, generalization, and overall performance.

Embedding the spotted G-invariance into our QNN means making it so that $f_{\boldsymbol{\theta}}(\rho) = f_{\boldsymbol{\theta}}(R(g)\rho R(g)^\dagger)$, for any $g \in G$ and $\rho \in \mathcal{X}$, and for all values of $\boldsymbol{\theta}$. That is, our model must be G-invariant too. *Equivariance* is the tool we use to achieve that.

Definition 5 (Equivariant map)**.** *Given a group G and its representations $(R^{in}, \mathcal{H}^{in})$ and $(R^{out}, \mathcal{H}^{out})$. A linear map $\phi : \mathcal{B}^{in} \to \mathcal{B}^{out}$ is (G, R^{in}, R^{out})-equivariant if and only if*

$$\phi \circ \mathrm{Ad}_{R^{in}(g)} = \mathrm{Ad}_{R^{out}(g)} \circ \phi, \ \forall g \in G\,. \tag{6.8}$$

Definition 5 can be graphically understood noting that it implies the closure of the following diagram:

$$\begin{array}{ccc} \mathcal{B}^{\mathrm{in}} & \xrightarrow{\mathrm{Ad}_{R^{\mathrm{in}}(g)}} & \mathcal{B}^{\mathrm{in}} \\ \Big\downarrow{\scriptstyle \phi} & & \Big\downarrow{\scriptstyle \phi} \\ \mathcal{B}^{\mathrm{out}} & \xrightarrow{\mathrm{Ad}_{R^{\mathrm{out}}(g)}} & \mathcal{B}^{\mathrm{out}}\,. \end{array}$$

Equivariant maps ϕ are then those who commute with the symmetry transformations, belonging to some group G, of the states they act on. This means that transforming the input state ρ via the input representation R^{in} as $\rho' = \mathrm{Ad}_{R^{\mathrm{in}}(g)}(\rho)$ and then applying the map to get $\rho_{\mathrm{out}} = \phi(\rho')$ yields the same result as first applying ϕ and then transforming the output state $\rho_{\mathrm{out}} = \mathrm{Ad}_{R^{\mathrm{out}}(g)}(\phi(\rho))$. The definition of equivariance given in Def. 5 also allows to characterize *invariant maps*, as it suffices to choose the trivial representation as the output one to have $\phi \circ \mathrm{Ad}_{R^{\mathrm{in}}(g)} = \phi \, \forall g \in G$.

We can define equivariance for operators as well:

Definition 6 (Equivariant operator)**.** *Given a group G and its representation $(R, \mathcal{H})$, an operator $O \in \mathcal{B}(\mathcal{H})$ is (G, R)-equivariant if and only if*

$$[O, R(g)] = 0, \ \forall g \in G\,. \tag{6.9}$$

This basically corresponds to saying that $O \in \mathfrak{comm}(R)$, *i.e.* that $\mathfrak{comm}(R)$ only contains equivariant operators.

Equivariant quantum maps and operators now open up the possibility of defining a strategy to come up with quantum learning models that are endowed with the inductive bias of the symmetry of the problem. Indeed, using Defs. 5 and 6 we can make so that the predictions of Eq. (6.5) are G invariant.

Proposition 1 (Invariance from equivariance)**.** *A model consisting of an (G, R^{in}, R^{out})-equivariant QNN and a (G, R^{out})-equivariant set of measurements is G-invariant.*

Proof. For every $g \in G$, $\rho \in \mathcal{B}^{\mathrm{in}}$ and $\boldsymbol{\theta}$ we have

$$\begin{aligned} f_{\boldsymbol{\theta}}(\mathrm{Ad}_{R^{\mathrm{in}}(g)}(\rho)) &= \mathcal{C}(\{\mathrm{Tr}\Big[\Phi_{\boldsymbol{\theta}}(\mathrm{Ad}_{R^{\mathrm{in}}(g)}(\rho))O_j\Big]\}_j) \\ &= \mathcal{C}(\{\mathrm{Tr}\Big[\mathrm{Ad}_{R^{\mathrm{out}}(g)}(\Phi_{\boldsymbol{\theta}}(\rho))O_j\Big]\}_j) \\ &= \mathcal{C}(\{\mathrm{Tr}\Big[\Phi_{\boldsymbol{\theta}}(\rho)R^{\mathrm{out}}(g)^\dagger O_j R^{\mathrm{out}}(g)\Big]\}_j) \\ &= \mathcal{C}(\{\mathrm{Tr}[\Phi_{\boldsymbol{\theta}}(\rho)O_j]\}_j) = h_{\boldsymbol{\theta}}(\rho)\,. \end{aligned}$$

□

The basic principle of this recipe is that we want the processing map Φ of the learning models to be agnostic to the symmetry, which is reflected in the fact that the representations of the group at hand commute with it, so that when we take an equivariant measurement at the end the group action cancels out leaving an invariant prediction.

We are now ready to formalize the basic framework for EQNNs. However, let us first make some remarks. First of all, even if we decided to only consider a supervised classification setting, the geometric quantum machine learning machinery can be applied also in unsupervised learning scenarios [166, 167], generative modeling [70, 168–170] or reinforcement learning [171, 172]. Secondly, equivariance is not new to quantum information theory [173–196] but its appearance and use in the QML community is only a couple years old.

6.3 How to build an Equivariant Quantum Neural Network

In [116], come to this point, we delve deeper in the theory of equivariant quantum maps, using representation theory tools to characterize their degrees of freedom, and to understand how to exploit intermediate representation changes to control the EQNN expressibility. In this thesis however, we will adopt a more pragmatical approach and jump straight into the techniques we have developed to build the $(G, R^{\text{in}}, R^{\text{in}})$-equivariant layers of an EQNN. With those, we will be able to conclude with a working example where we show the advantages of using quantum models endowed with inductive biases as opposed to problem-agnostic ones.

Thus, from now on we will assume that every Hilbert space $\mathcal{H}$ being considered is that of a quantum register composed of n qubits. Furthermore, let us make the structure of a general equivariant quantum neural network explicit:

Definition 7 (Layered EQNN)**.** *An L-layered G-equivariant QNN is defined by a sequence of $L+1$ representations of G, $(R^{in}, R^1, \ldots, R^{out})$, and a sequence of (G, R^l, R^{l+1})-equivariant layers.*

The way the data is G-equivariantly processed in such a quantum model

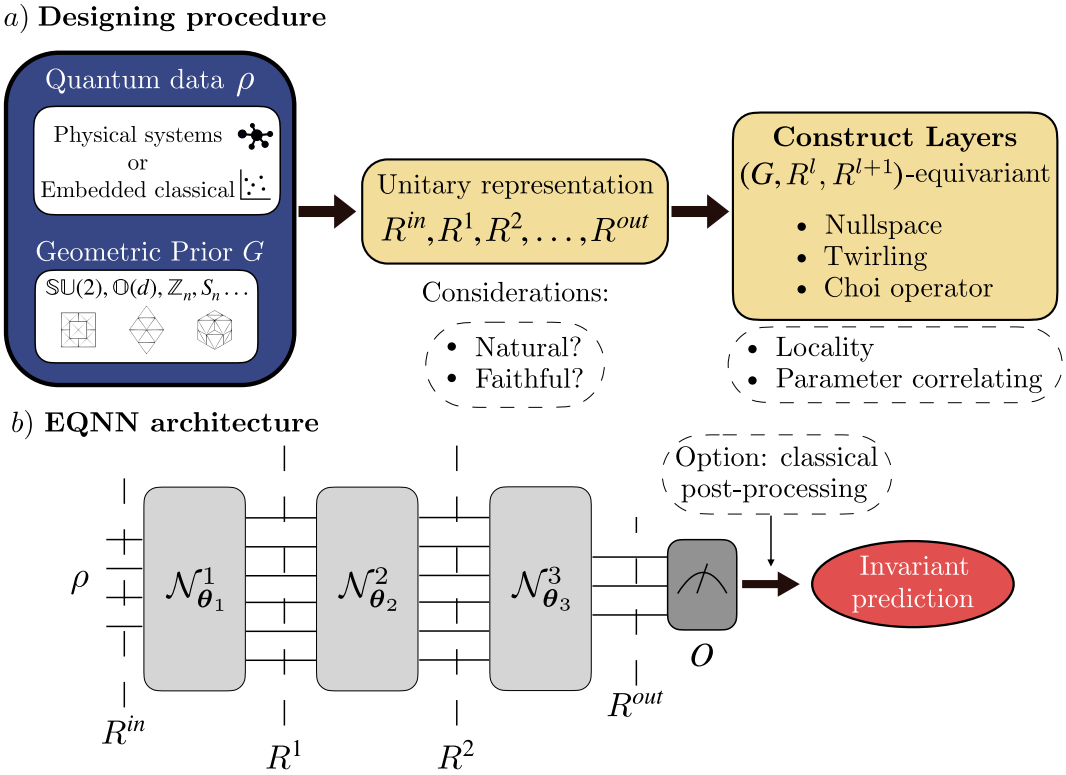
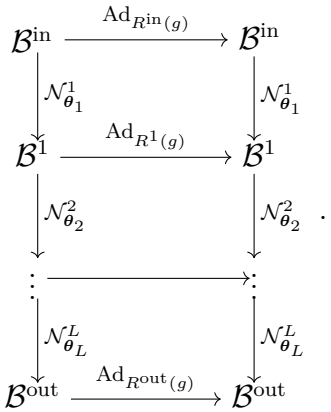


Figure 6.2: **Equivariant quantum neural network [116].** a) In a QML problem, we have a dataset (which can either be a quantum mechanical dataset or classical data encoded in quantum states) and a label symmetry group, denoted as G. The first step is to define the input and output representation of G at each layer, which can be natural, faithful, non-faithful, etc. We will then provide various methods for constructing EQNN layers and controlling features such as gate locality. b) Dashed lines in the architecture separate the representation of the symmetry group G at specific stages in the EQNN, which may change between layers. At the beginning, the input state ρ_{in} is transformed by the representation R^{in}. The l^{th} layer of the EQNN, $\mathcal{N}^l_{\boldsymbol{\theta}_l}$, must be (G, R^l, R^{l+1})-equivariant. Overall, the full architecture, $\phi = \mathcal{N}^L_{\boldsymbol{\theta}_L} \circ \cdots \circ \mathcal{N}^1_{\boldsymbol{\theta}_1}$, is $(G, R^{\text{in}}, R^{\text{out}})$-equivariant. The $(G, , R^{\text{out}})$-equivariant measurement operator O is in the commutant of the output representation R^{out}. If we only want the EQNN to produce an output state equivariantly or invariantly (e.g. in generative models), we can omit the measurements.

can be visually rendered via the following commutative diagram



It is straightforward to conclude from this graph that the sequence $\mathcal{N} = \mathcal{N}^L_{\theta_L} \circ \cdots \circ \mathcal{N}^1_{\theta_1}$ ends up being a $(G, R^{\text{in}}, R^{\text{out}})$-equivariant QNN. Thus, if we follow such EQNN with (G, R^{out})-equivariant measurements, we achieve a G-invariant model.

Notice how in practice, while the input representation R^{in} is fixed by the physical action of the symmetry group on the input data, everything that happens next, *i.e.* the intermediate and final (output) representations acting on the spaces $\mathcal{B}^l$ are not. This means there exists freedom in choosing a sequence of representations $(R^{\text{in}}, R^1, \ldots, R^{\text{out}})$ under which the layers are equivariant. While we address the reader to Appendix D for an analysis of the effect the choice of intermediate representations, we now introduce a distinction between (G, R^l, R^{l+1})-equivariant quantum layers based on the relative properties of R^l and R^{l+1}.

Definition 8 (Equivariant layers: standard, embedding and pooling)**.** *Let $\Phi^l_{\boldsymbol{\theta}_l} : \mathcal{B}^{l-1} \to \mathcal{B}^l$ be an (G, R^{l-1}, R^l)-equivariant layer. We say that $\Phi^l_{\boldsymbol{\theta}_l}$ is a pooling layer if $\dim(\mathcal{B}^l) < \dim(\mathcal{B}^{l-1})$, an embedding layer if $\dim(\mathcal{B}^l) > \dim(\mathcal{B}^{l-1})$, and a standard layer if $\dim(\mathcal{B}^l) = \dim(\mathcal{B}^{l-1})$.*

Since we are considering EQNNs working on a quantum computer, if the input register has n qubits and the output one has m, we can restate definition 8 as

- Pooling layers: they decrease the number of active qubits, $m < n$.
- Standard layers: they keep the size of the quantum register unaltered, $m = n$.
- Embedding layers: they enlarge the qubit register, $m > n$.

with these prototypical layers, whose names and roles are inspired by their classical counterparts [119, 130], a general EQNN architecture can be sketched as we present in Fig. 6.3.

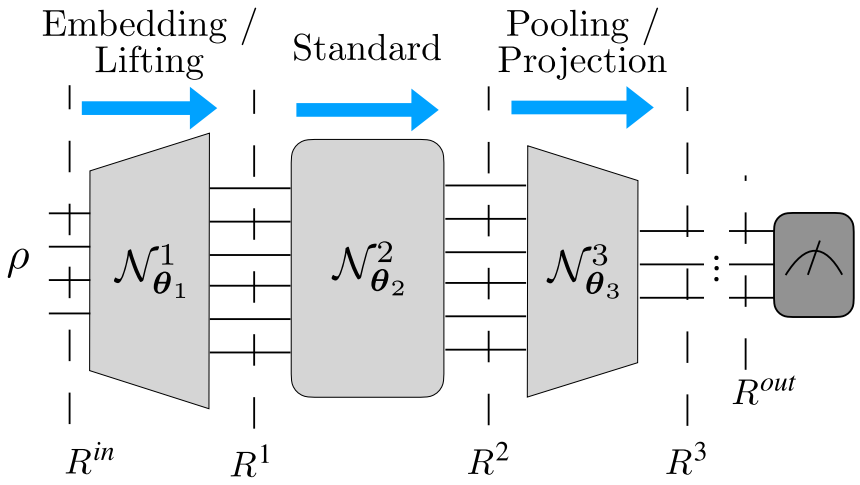


Figure 6.3: **Different types of equivariant layers in a general architecture of EQNNs [116].** A standard layer maps data between spaces of the same dimension. An embedding (pooling) layer maps the data to a higher-dimensional (smaller-dimensional) space.

We can finally answer the question that the reader is probably asking: *how do I explicitly build an equivariant quantum layer?* One of the main achievement of our work [116] is the formalization of three different avenues to craft the quantum gates and channels that can be stacked up to build EQNNs. Here, we will discuss the first two of these methods, leaving the last one for Appendix E, and then show how to parameterize the found quantum maps to allow for their training.

6.3.1 Thinking in terms of superoperators

From Definition 5 it follows that any linear map ϕ that wants to be G-equivariant must satisfy the superoperator equation

$$\phi \circ \mathrm{Ad}_{R^{\mathrm{in}}(g)} - \mathrm{Ad}_{R^{\mathrm{out}}(g)} \circ \phi = 0, \ \forall g \in G\,. \tag{6.10}$$

This condition, by linearity, defines a vector space. Thus, if we can find a basis of such space we can characterize all of the equivariant maps, and although it may seem that one needs to solve Eq. (6.10) for every element g of the group G, we will demonstrate that it is often sufficient to solve this equation only for a selected subset of elements within the group or its Lie algebra for Lie groups of symmetries.

Finite groups

In the case when G is a finite group, we can use a generating set to efficiently represent and work with the group. A generating set is a subset $S = g_1, \ldots, g_{|S|}$ of G such that every element in G can be expressed as a product of elements

in S. The closure of S, denoted as $\langle S \rangle$, is the set of all possible products of elements in S. A generating set S generates G if $\langle S \rangle = G$. For instance, the symmetric group S_n can be generated by the set of transpositions. It has been shown that a finite group can be generated using a subset S of size at most $\log_2(|G|)$ [158]. This means that even large groups can be efficiently represented and manipulated through their generating set. In particular, finding equivariant maps can be simplified using the following theorem.

Theorem 1 (Finite group equivariance)**.** *Given a finite group G with generating set S, a linear map ϕ is (G, R^{in}, R^{out})-equivariant if and only if*

$$\phi \circ \mathrm{Ad}_{R^{in}(g)} - \mathrm{Ad}_{R^{out}(g)} \circ \phi = 0, \quad \forall g \in S\,. \tag{6.11}$$

Lie groups

Theorem 1 turns out to be useful for groups G that are finitely generated, but many important groups, such as the Lie group $\mathbb{U}(d)$, are not. However, we can still use generating sets, but now at the Lie algebra level. Ref. [138] presents a method for imposing equivariance under Lie groups, in which the equivariance constraint is applied to a basis of the Lie algebra. However, this becomes impractical for large Lie groups because the method scales linearly with their dimension. Instead, we prove that it is sufficient to impose the constraint only over a generating set. This means we can consider a subset $s = a_1, \ldots, a_{|s|}$ of the Lie algebra $\mathfrak{g}$ to be a generating set if its Lie closure $\langle s \rangle_{\mathrm{Lie}}$, which is the set of all possible nested commutators of elements in s, spans the entire Lie algebra. With these concepts in mind, we can now impose equivariance at the algebra level.

Theorem 2 (Lie group equivariance)**.** *Given a compact Lie group G with a Lie algebra $\mathfrak{g}$ generated by s such that exponentiation is surjective, a linear map ϕ is (G, R^{in}, R^{out})-equivariant if and only if*

$$\mathrm{ad}_{r^{out}(a)} \circ \phi - \phi \circ \mathrm{ad}_{r^{in}(a)} = 0, \quad \forall a \in s\,, \tag{6.12}$$

where r^{in}, r^{out} are the representations of G induced by R^{in}, R^{out}.

6.3.2 Nullspace and twirling

Having simplified the task of finding equivariant maps, since we can now think in terms of linear superoperators and only need to worry about finding a basis of equivariant operations, we are ready to illustrate the two methods to solve this task.

Nullspace method

The *nullspace* method involves expressing the equivariance constraints of Eq. (6.10) as a system of matrix equations and finding a basis for the vector space of the solutions. Notice that the basis elements that we may find need not to be acceptable CPTP maps, we will later have to impose this constraint. For a finite group, that we will assume to be the case in the rest of this section, we can solve the equations with respect to a set of generators. A similar approach can be used for Lie groups by working at the Lie algebra level.

Our method generalizes those in [138, 197] and is composed by the following steps.

- Express the superoperators appearing in Eq. (6.11) as matrices by using the following map $\phi \mapsto \overline{\phi} = \sum_{i,j} \phi_{i,j} |P_i\rangle\rangle\langle\langle P_j|$, where P_j and P_i are Pauli operators acting over the input and output Hilbert spaces, respectively [198], and $|\cdot\rangle\rangle$ denotes the vectorized version of an operator. Namely, once a basis of the operators space has been chosen, $|A\rangle\rangle$ is just the vector of components of any operator A w.r.t. it. Here, $\overline{\phi}$ is a $\dim(\mathcal{B}^{\text{out}}) \times \dim(\mathcal{B}^{\text{in}})$ matrix. Substituting this expression in Eq. (6.11), turns the latter into a matrix multiplication equation of the form

$$\overline{\phi} \cdot \overline{\text{Ad}}_{R^{\text{in}}(g)} - \overline{\text{Ad}}_{R^{\text{out}}(g)} \cdot \overline{\phi} = 0\,, \quad \forall g \in S\,. \tag{6.13}$$

- Perform a vectorization [199], mapping a matrices to column vectors and allowing us to write Eq. (6.13) as

$$M_g \cdot \text{vec}(\overline{\phi}) = 0\,. \tag{6.14}$$

  Here, $\text{vec}(\overline{\phi})$ is a $\dim(\mathcal{B}^{\text{in}})\dim(\mathcal{B}^{\text{out}})$-dimensional column vector and

$$M_g = (\overline{\text{Ad}}_{R^{\text{in}}(g)})^\top \otimes \mathbb{1}_{\dim(\mathcal{B}^{\text{out}})} - \mathbb{1}_{\dim(\mathcal{B}^{\text{in}})} \otimes \overline{\text{Ad}}_{R^{\text{out}}(g)}\,, \tag{6.15}$$

  is a $\dim(\mathcal{B}^{\text{in}})\dim(\mathcal{B}^{\text{out}}) \times \dim(\mathcal{B}^{\text{in}})\dim(\mathcal{B}^{\text{out}})$ matrix.

- Obtain the sought equivariant maps by computing the intersection of the *nullspaces* of each M_g, i.e.,

$$\text{vec}(\overline{\phi}) \in \bigcap_{g \in S} \text{Null}(M_g)\,. \tag{6.16}$$

  for example by Gaussian elimination [200].

In Fig. 6.4 a prototypical application of the nullspace method is shown.

Before moving on to the next methods, let us point out some important remarks about the nullspace method we just outlined. First, this procedure is bound to quickly become computationally expensive. Indeed, looking for equivariant quantum maps from n to m qubits by solving the nullspaces

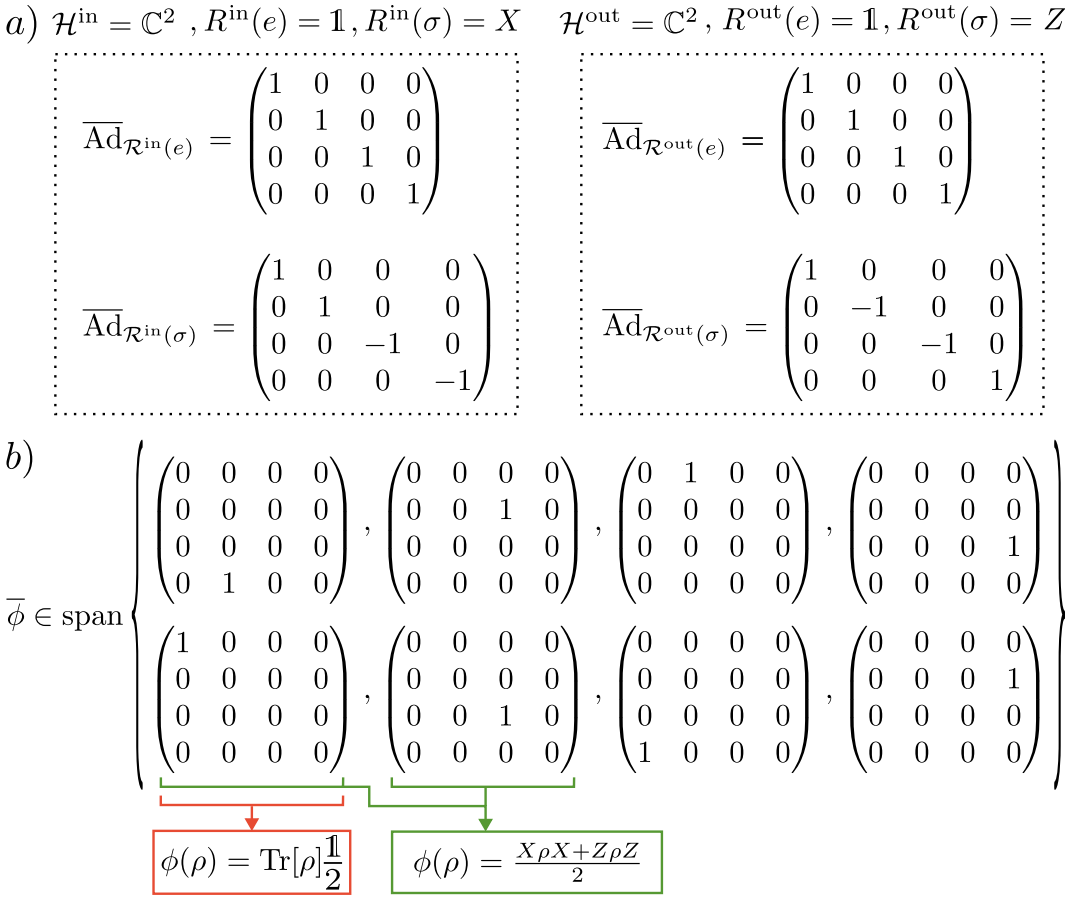


Figure 6.4: **Example of the nullspace method [116].** The nullspace method is used to identify the set of 1-to-1-qubit quantum channels that are $(G,R^{\text{in}},R^{\text{out}})$-equivariant. In this case, G is the group $\mathbb{Z}_2 = e, \sigma$, R^{in} is the group of transformations represented by id, X, and R^{out} is the group of transformations represented by id, Z. a) The matrix representation of the adjoint representation of the symmetry group for both input and output. b) A basis for the 8-dimensional solution space, as well as two specific quantum channels that are equivariant with respect to this symmetry group: one obtained from the solution in red, and one obtained by combining the solutions in green.

through Gaussian elimination [201] requires an exponential amount of resources, as its complexity scales as $\mathcal{O}(2^{6(m+n)})$. Second, let us stress again that solving Eq. (6.16) only leads to a basis for the space of all possible equivariant linear maps, without any physical constraint to ensure that they are also valid quantum maps. Thus, additional steps must be taken to extract the realizable operations (see Sec. 6.3.3). For instance, to single out the trace-preserving maps (TP) ϕ, one needs to make sure that $\overline{\phi}$ contains the term $\frac{\dim(\mathcal{H}^{\text{in}})}{\dim(\mathcal{H}^{\text{out}})} \left|\mathbb{1}_{\dim(\mathcal{H}^{\text{out}})}\right\rangle\!\rangle\langle\!\langle \mathbb{1}_{\dim(\mathcal{H}^{\text{in}})}\right|$ and no other terms mapping to $\left|\mathbb{1}_{\dim(\mathcal{H}^{\text{out}})}\right\rangle\!\rangle$.

In practice however, if one can restrict the set of Pauli operators that appear in the input and output spaces decompositions, the computational complexity of the nullspace method can be significantly lowered. This is particularly useful for inner symmetries 3, where the action of the group can be locally studied. For example, consider the following lemma:

Lemma 1 (Global equivariance via local equivariance)**.** *Let $\mathcal{B}^{in\ (out)}$ be composite input (output) spaces of the form $\mathcal{B}^{in\ (out)} = \bigotimes_j \mathcal{B}_j^{in\ (out)}$. Then, assume that the representations acting on each of these space takes a tensor product structure over subsystems as $R^{in\ (out)}(g) = \bigotimes_j R_j^{in\ (out)}(g)$. For local equivariant channels mapping between each pair of in-and-out subsystems $\phi_j : \mathcal{B}_j^{in} \to \mathcal{B}_j^{out}$ that are (G, R_j^{in}, R_j^{out})-equivariant, we have that $\bigotimes_j \phi_j$ is (G, R^{in}, R^{out})-equivariant.*

By constructing equivariant maps locally and taking their tensor product, it is possible to create a global equivariant layer in a computationally efficient way (e.g., solving for 2-to-1 qubit maps only requires working with 64×64 matrices). However, this approach may not be as expressive as a general equivariant global channel, as the composition of local equivariant channels may have limited action [190].

This holds also for outer symmetry groups 3 such as the permutation group S_n, as one can express all the possible permutations in terms of local ones, which involve only two-qubit operations.

Twirling method

The second method that can be used to look for equivariant quantum layers is based on *twirling* and was first proposed in [152] where it was used to find equivariant unitary channels. We extended this framework to general non-unitary quantum maps, allowing also for the change of representations from input to output spaces.

The *twirl* over a finite symmetry group G of a channel $\phi : \mathcal{B}^{\text{in}} \to \mathcal{B}^{\text{out}}$ is defined as

$$\mathcal{T}_G[\phi] = \frac{1}{|G|} \sum_{g \in G} \text{Ad}_{R^{\text{out}}(g)} \circ \phi \circ \text{Ad}^{\dagger}_{R^{\text{in}}(g)} \,. \tag{6.17}$$

If G is a Lie group instead, it suffices to replace the summation with an integral over the Haar measure

$$\mathcal{T}_G[\phi] = \int_G d\mu(g) \mathrm{Ad}_{R^{\mathrm{out}}(g)} \circ \phi \circ \mathrm{Ad}^{\dagger}_{R^{\mathrm{in}}(g)} \,. \tag{6.18}$$

$\mathcal{T}_G[\phi]$ is no other than the projection of the map ϕ onto the equivariant subspace. Indeed, applying any group element $\mathrm{Ad}^{\dagger}_{R^{\mathrm{in}}(\tilde{g})}$ on the right corresponds to the rescaling $g \to g\tilde{g}$. Then, by the properties of the Haar measure, we can change the integration variable $g \to g' = g\tilde{g}$. This makes so that in the left operator $\mathrm{Ad}_{R^{\mathrm{out}}(g)}$ in Eq. (6.18) the group element changes into $g \to \tilde{g}^{-1}g'$. Now, we can use the fact that $\mathrm{Ad}_{R(g^{-1})} = \mathrm{Ad}^{\dagger}_{R(g)}$ to extract $\mathrm{Ad}^{\dagger}_{R^{\mathrm{out}}(\tilde{g})}$ on the left of the twirling integral, thus proving that the twirled channel is indeed equivariant. Moreover, if ϕ is $(G, R^{\mathrm{in}}, R^{\mathrm{out}})$-equivariant $\mathcal{T}_G[\phi] = \phi$ since the action of the group elements commute with it and annihilate each other in the integral, leaving ϕ unchanged. The same clearly holds for the finite group case too. This means that any channel ϕ admits a decomposition

$$\phi = \mathcal{T}_G[\phi] + \phi_A \,, \tag{6.19}$$

where ϕ_A is the "anti-symmetric" part of ϕ, i.e., the part satisfying $\mathcal{T}_G[\phi_A] = 0$. As such, any measure of the form $\|\phi_A\|$ can be used to quantify the degree of equivariance of ϕ. As for the nullspace approach, we sketched an example of the application of the twirling method in figure 6.5).

Twirling is relatively simple to implement for small groups, as it is possible to efficiently compute the summation in Eq. (6.17). However, if the group is large or even worse if it is a Lie group, directly implementing twirling becomes more complex and may require more advanced techniques. In [116] we tackled this problem in several ways, one can for example resort to Weingarten calculus [202, 203] for analytically computing twirled operators, use experimental methods such as approximate twirling[204] or in-circuit twirling via the methods b) and c) shown in Fig. 6.5).

For comparison, it is worth noting that one of the main benefits of twirling is that it *guarantees* that, if the starting map ϕ is a valid quantum channel, the resulting map will in turn be completely positive and trace preserving, unlike the results of the nullspace method. However, the latter can identify *all* equivariant maps, while twirling is applied to each map individually, which could make it more difficult to find a complete basis for the space of equivariant maps. It may even happen that upon an unfortunate choice of gates and maps, after twirling via Eqs. (6.17,6.18) the result is the null map [152]. Therefore, if one is looking for a single equivariant channel, twirling is a good choice, but if one wants to find the complete set of equivariant maps, the nullspace method may be more suitable.

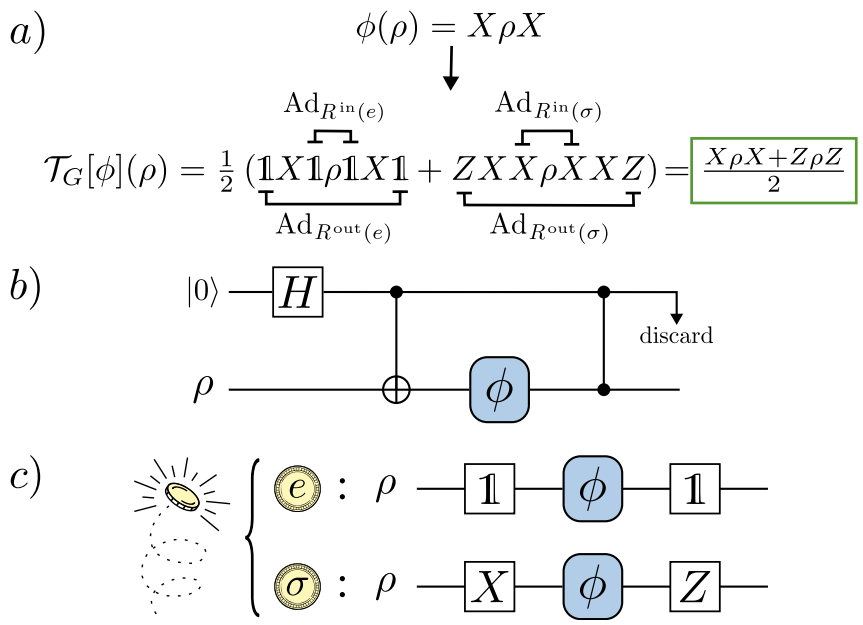


Figure 6.5: **Example of the twirling method [116].** We demonstrate how to use the twirling method to determine the space of 1-to-1 qubit (G,R^{in},R^{out})-equivariant quantum channels, with $G = \mathbb{Z}_2 = \{e, \sigma\}$, $R^{\mathrm{in}} = \{\mathbb{1}, X\}$ and $R^{\mathrm{out}} = \{\mathbb{1}, Z\}$. a) Explicit calculation using the twirling formula of Eq. (6.17). b) Ancilla-based scheme for in-circuit twirling. c) Classical-randomness scheme for in-circuit twirling.

6.3.3 Parametrizing the layers of an EQNN

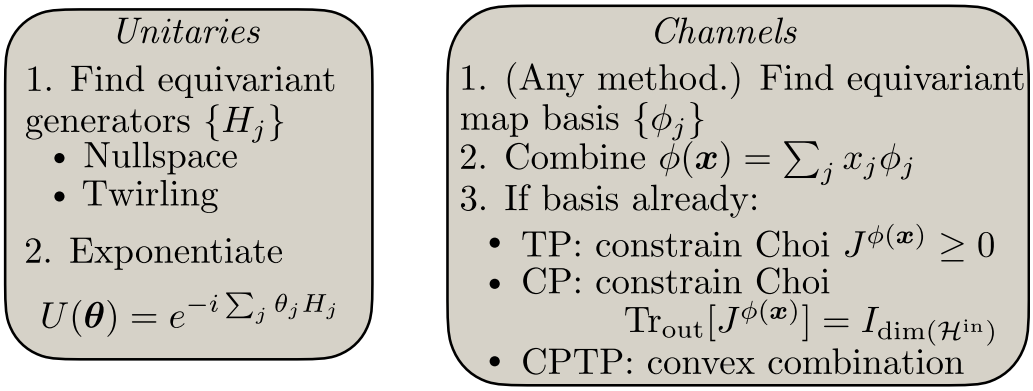
Since we ultimately want to use equivariant maps to build quantum learning models, we are interested in not only identifying such channels but also in parameterizing and optimizing them. In this section, we will demonstrate how to parameterize the layers of an EQNN. We will begin by considering the case of unitary channels and then extend the discussion to general maps. A summary of the methods that we are now going to discuss is given in Fig. 6.6.

Parametrizing equivariant unitaries

Let us start by considering the case of a *unitary* EQNN layer with the same input and output representations. That is, $\mathcal{H}^{\mathrm{in}} = \mathcal{H}^{\mathrm{out}}$, $R^{\mathrm{in}} = R^{\mathrm{out}} = R$ and $\mathcal{N}^l_{\boldsymbol{\theta}_l}(\rho) = U_l(\theta_l)\rho U_l(\theta_l)^\dagger$. Notice that this task has already been considered in [151, 152, 205].

As discussed in section 3.3, a common way to parameterize a unitary is to express it as the exponential of a Hermitian operator, called its generator, i.e. $U_l(\theta_l) = e^{-i\theta_l H_l}$, where θ_l is a trainable parameter. To obtain (G, R)-equivariant parametric unitaries, we can then use equivariant generators, i.e. $H_l \in \mathfrak{comm}(R)$, which can be found using the nullspace or twirling methods previously discussed. These techniques were originally presented for superoperators, but they can also be applied to operators.

a) **Parametrizing equivariant channels**



b) **Optimizing EQNNs**

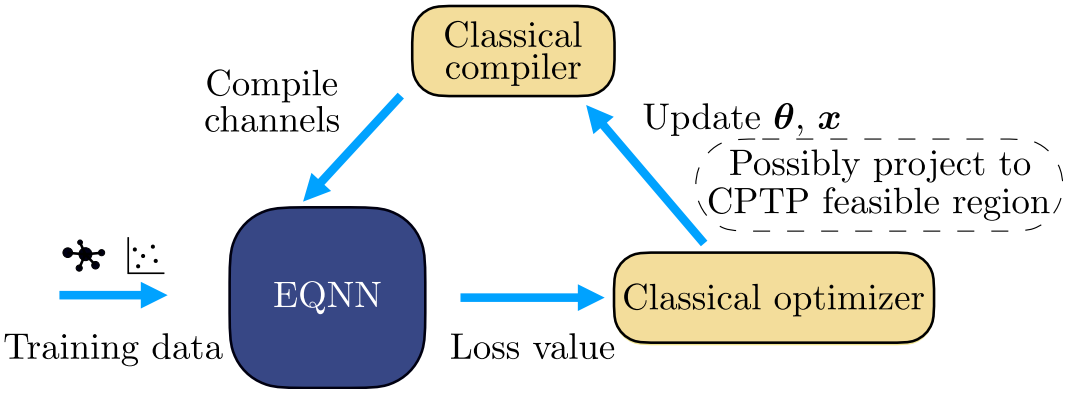


Figure 6.6: **Procedure to parametrize and optimize equivariant quantum neural networks [116].** a) We present methods for parametrizing equivariant QNN layers, be those unitaries or channels. b) Once we parametrize equivariant quantum neural network (EQNN), we can train it by feeding in training data and using the outputs to calculate the loss function. We can then use a classical optimizer to find updates for the EQNN parameters, possibly projecting the updated map onto the feasible completely positive trace-preserving (CPTP) region. This process is repeated until convergence is achieved.

Parametrizing equivariant channels

Lastly, let us now describe how to parameterize and optimize over equivariant *channels*. We will assume that a basis of equivariant maps (or a subset of this basis) has already been identified using the nullspace method. Although it is relatively easy to find equivariant maps, remember that they may not necessarily be physical channels because they may not be trace preserving, completely positive, or both. However, it is still possible to parameterize a set of non-CPTP equivariant maps and optimize over them by appropriately constraining the parameters to ensure that the final map is CPTP.

Say that, after having used the nullspace approach, one has managed to identify a basis set $\{\phi_j\}$ of equivariant maps. Any equivariant layer can then be expressed as a linear combination of those $\mathcal{N} = \sum_j x_j \phi_j$. Now, to impose the conditions that we want on $\mathcal{N}$ we just need to derive how those translate to the parameter space. First of all, we can check if among the basis elements ϕ_j there are any prohibited operations, such as maps that non-trivially change the trace of the operators they act on. We can then prune the basis by dropping them, remaining with only either trace-preserving or trace-annihilating maps. Then, to make the composite map $\mathcal{N}$ trace-preserving, it suffices to make sure that the parameters associated with trace-preserving basis elements add up to one. Coming to the CP condition, one can start noticing that it translates to the Choi operator $J^{\mathcal{N}}$ as $J^{\mathcal{N}} \geqslant 0$. Thus we can impose a further condition on the parameters x_j to make the eigenvalues of $J^{\mathcal{N}}(\boldsymbol{x})$ non-negative. This way, we will be left with a region of feasible equivariant quantum channels, see Section 6.4 for an example. During the optimization of $\boldsymbol{x}$, the update rule may take us outside of the feasible space, in which case we need to project back onto it. It is worth noting that although it may not be immediately clear how to implement the resulting channel, it can be transformed into a sequence of implementable gates acting on a potentially larger space using compilation techniques [206–209] that allow to transform general maps into a sequence of gates, acting on extended registers as per Eq. (2.27), that can be implemented on a quantum device. In some cases, particularly when the maps operate on large-dimensional spaces, it may be challenging to find the eigenvalues of $J^{\mathcal{N}}$). In these situations, we can optimize over a subset of equivariant channels (i.e., maps that are already CPTP) that can be found through twirling. We are guaranteed that any convex combination of equivariant channels will be in the feasible region because CPTP channels form a convex set [185].

An alternative approach to constructing equivariant channels is via the Stinespring dilation picture [210]. Recall from Section 2.2.2 that any channel can be written as the result of observing only a part of an unitary operation on a larger space, *i.e.*

$$\phi(\rho) = \mathrm{Tr}_E[U(\rho \otimes |\omega\rangle\langle\omega|)U^\dagger]\,, \tag{6.20}$$

where $|\omega\rangle \in \mathcal{H}^E$ is a fixed reference state on an environment Hilbert space $\mathcal{H}^E$, and where Tr_E denotes the trace over $\mathcal{H}^E$. If $U(R^{\mathrm{in}}(g) \otimes \mathbb{1}_{\dim(\mathcal{H}^E)}) =$

$(R^{\text{out}}(g) \otimes R^{(E)}(g))U,\ \forall g \in G$, then ϕ is a $(G, R^{\text{in}}, R^{\text{out}})$-equivariant channel. Here we can use any of the tools previously discussed to find and parametrize U. This approach has the advantage that by fixing the dimension of the environment, we can look for channels of small Kraus rank which are easier to implement in practice.

6.4 A case study: EQCNN for quantum phase classification

The classification of quantum phases of matter is an important topic in condensed matter physics [211], the field that studies the physical properties of matter on the macroscopic scale that emerge from its microscopic description. In this context, a *phase* of matter refers to a distinct state of the macroscopic system at hand that exhibits certain characteristic properties, such as its density, magnetization, and electrical conductivity. Note that, when discussing about phases of matter, we usually refer to the phases that the ground state $|\psi\rangle_g^{\boldsymbol{J}}$ of a family of Hamiltonians $H(\boldsymbol{J})$, depending on some parameters $\boldsymbol{J}$, belongs to. The values of the couplings $\boldsymbol{J}$, e.g. some external magnetic fields, determine the ground state properties of the Hamiltonian family. Values of the couplings at which the system changes its global properties are called *critical*, and constitute the boundaries of the different phases of the system when these are sketched down in a so-called *phase diagram*, a plot whose axes are the parameters $\boldsymbol{J}$. For simplicity, we will assume that the systems at hand are at zero temperature. Moreover, since phases are properties of macroscopic systems, they are meant to be studied in the so called thermodynamic limit where the number of microscopic constituents of the system tend to infinity. However, when analytical methods cannot (or are too hard to) be applied, and the thermodynamic limit cannot be solved, studying the behaviour of finite size systems, as we will do in the following, can still lead to grasping intuition about the different phases of matter.

An example of an approach to classifying quantum phases is looking at the *topological* properties of the system. A topological phase is a state of matter that exhibits certain properties that are robust against small perturbations and changes in the system, and can be characterized by topological invariants. Examples of topological phases include the quantum Hall effect and topological insulators [212, 213]. Topological phases are of particular interest because they can exhibit exotic properties, such as the ability to conduct electricity on their surface while remaining insulating in the bulk. These properties are thought to have potential applications in the development of new technologies, such as quantum computers and sensors [214].
Overall, the classification of quantum phases of matter is a complex and active area of research that has important implications for understanding the behav-

ior of materials and the development of new technologies.

We do not need to delve deeper into the theory underlying the study of quantum phases or of the transitions between them [212], as this is way above the scope of this work. What is important to us is that quantum phases can be thought of as labels that describe the global properties of quantum ground states, and as such one can tackle phase classification as a learning problem. Indeed, we can embed quantum phase classification in the QML framework by noticing that it boils down to learning a hidden relation between the space of ground states of the model being considered and the space of labels with which we distinguish the different phases. Thus, we aim at building a quantum classifier that, once trained on a dataset $\mathcal{D} = \{(|\psi\rangle_g^{\boldsymbol{J}_i}, y_i)\}$ of ground states with their associated phase label, is able to predict which phase a previously unseen ground state belongs to.
We chose to study the following one-dimensional model.

6.4.1 Bond-Alternating XXX Model

The 1-D bond-alternating XXX Heisenberg model is a mathematical model used to describe the behavior of a one-dimensional chain of spin-1/2 particles coupled through a Heisenberg exchange interaction. In this model, the strength of the interaction between nearest-neighbor spins alternates between two different values, known as *bond alternation*. The Heisenberg exchange interaction is a type of interaction between two spins that depends on their relative orientation. It is described by the following Hamiltonian:

$$H = \sum_{k=x,y,z} J_k S_i^k S_j^k \,, \tag{6.21}$$

where J is the exchange coupling constant, S_i and S_j are the spin operators for the i-th and j-th spins that are interacting, reading $S = (S^x, S^y, S^z) = \frac{1}{2}(\sigma_x, \sigma_y, \sigma_z)$. Standard Heisenberg models describe lattices of spins where nearest-neighbors interact through the term in Eq. (6.21). In the XXX Heisenberg model, the interaction is isotropic, meaning that it is the same in all directions, *i.e.* $J_x = J_y = J_z = J$.

The bond-alternating XXX Heisenberg model is a generalization of the regular XXX Heisenberg model, in which the exchange coupling constant J is allowed to alternate between two different values J_1 and J_2. This results in a periodic modulation of the strength of the exchange interaction along the chain, with the interaction between every other pair of nearest-neighbor spins being stronger or weaker than the interaction between the other pairs. The bond-alternating XXX Heisenberg model can thus be represented by the following Hamiltonian:

$$H = J_1 \sum_{i\,\text{even}} S_i \cdot S_{i+1} + J_2 \sum_{i\,\text{odd}} S_i \cdot S_{i+1} \,. \tag{6.22}$$

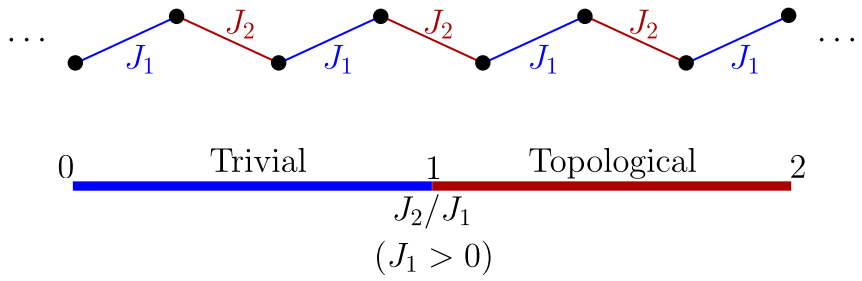


Figure 6.7: **Schematic representation of the 1-D Alternating Bond XXX Heisenberg Model (Eq.** (6.22)**).** Above: the 1-D chain with alternating coupling constants $J_{1,2}$. Each couple of spins interacts via the term in Eq. (6.21) with the corresponding J value. Below: The model's phase diagram for the configuration of interest $J_{1,2} > 0$.

The behavior of the 1-D bond-alternating XXX Heisenberg model can be studied using a variety of techniques, including exact diagonalization [215], quantum Monte Carlo simulations [216], and perturbation theory [217]. The model has been shown to exhibit a variety of interesting behaviors, including antiferromagnetic order, dimerization, and quantum phase transitions [218]. Antiferromagnetic order refers to a state in which the spins on the chain are arranged in an alternating pattern, with adjacent spins pointing in opposite directions. This can occur when the exchange coupling constants J_1 and J_2 are both antiferromagnetic (AFM), *i.e.* $J_{1,2} < 0$, meaning that they favor the alignment of spins in opposite directions. Dimerization refers to the formation of spin dimers, which are pairs of spins that are strongly coupled to each other. As opposed to antiferromagnetic ordering, dimerization needs both J_1 and J_2 to be ferromagnetic (FM), *i.e.* $J_{1,2} > 0$, since it is in this case that the alignment of spins along the same direction is favoured. We will consider the model described by Eq. (6.22) with open boundary conditions and with both the couplings in the ferromagnetic regime. This means that the spin chain does not close to form a circle, and the first and last spin only interact with one neighbor. In this case, it is possible for a quantum phase transition to occur between a *trivial* phase and a *topologically protected* phase. A trivial phase is a phase of matter that can be described by a local order parameter and exhibits no topological properties. In contrast, a topologically protected phase is a phase of matter that exhibits non-local properties and is protected against certain types of perturbations [219].

In the case of the 1-D bond-alternating XXX Heisenberg model, the topologically protected phase is characterized by a non-trivial ground state degeneracy and the presence of end states that are protected against local perturbations. These end states, also known as boundary modes, are a characteristic feature of topologically protected phases and are not present in the trivial phase [220]. The quantum phase transition between the trivial and topologically protected phases in the 1-D bond-alternating XXX Heisenberg model occurs at a critical value of the bond alternation parameter $\alpha = J_2/J_1$, which

determines the relative strength of the exchange interaction between nearest-neighbor spins. When the bond alternation parameter is below the critical value $\alpha = 1$, the system is in the trivial phase. When the bond alternation parameter is above the critical value, the system is in the topologically protected phase. The topologically protected phase in the 1-D bond-alternating XXX Heisenberg model has been studied extensively in recent years due to its potential applications in the field of quantum computing. In particular, the boundary modes of the topologically protected phase have been proposed as a platform for the realization of topological qubits, which are a type of qubit that is protected against certain types of noise [214].

Looking at the Heisenberg interaction (6.21) one can readily check that the term $S_i \cdot S_j$ possesses an $\mathbb{SU}(2)$ symmetry. Indeed, consider the total spin operator of the interacting couple of spins, which is defined as

$$S_{\text{tot}} = S_i + S_j = (S_i^x + S_j^x, S_i^y + S_j^y, S_i^z + S_j^z)\,. \tag{6.23}$$

The spin operators of S_{tot} form an $\mathfrak{su}(2)$ algebra, since they satisfy the usual spin operator commutation relation $[S_{\text{tot}}^a, S_{\text{tot}}^b] = i\epsilon_{abc}S_{\text{tot}}^c$, where Einstein summation rule is assumed. Now, we can check that

$$[H, S_{\text{tot}}] = 0\,, \tag{6.24}$$

which would imply the $\mathbb{SU}(2)$ symmetry of the interaction term. Let us show that for one component

$$\begin{aligned}
[H, S_{\text{tot}}^x] &= [(S_i^x \otimes S_j^x + S_i^y \otimes S_j^y + S_i^z \otimes S_j^z), S_i^x + S_j^x] \\
&= [S_i^y \otimes S_j^y + S_i^z \otimes S_j^z, S_i^x + S_j^x] \\
&= [S_i^y, S_i^x] \otimes S_j^y + S_i^y \otimes [S_j^y, S_j^x] + (y \to z) \\
&= -iS_i^z \otimes S_j^y - iS_i^y \otimes S_j^z + iS_i^y \otimes S_j^z + iS_i^z \otimes S_j^y \quad = 0
\end{aligned}\,, \tag{6.25}$$

where we used the commutation relations stated above and the fact that operators on different sites commute. It is immediate to check that this holds for the y and z components of S_{tot} as well.

Since all the qubits in the chain interact through that term, the symmetry extends to the whole model through the tensor product representation $R^{\text{in}}(g) = R_{\text{tens}}(g) = g^{\otimes n}$. This is of course a symmetry of the phase labels too, as quantum phases are global properties of the groundstates of the model, and symmetries of the Hamiltonian are also symmetries of the groundstates. Indeed if $[H, U_g] = 0$ and $|\psi\rangle$ is a groundstate, *i.e.* $H|\psi\rangle = E_0|\psi\rangle$ with E_0 being the minimal energy of the system, then $U_g|\psi\rangle$ is still a groundstate, as it is easy to check. Thus, our quantum phase classifier can be endowed with this inductive bias, meaning that we can search for an $\mathbb{SU}(2)$-equivariant quantum learning model.

Lastly, let us comment on another inductive bias that we can exploit. When a system is translationally invariant, meaning that an homogeneous shift of the

positions of its constituents leaves its Hamiltonian unchanged, we can further build our equivariant layers in such a way to not alter this feature. The way to do so is to carefully arrange the equivariant gates and channels that we found with respect to the other symmetries of the model such that the resulting layers act in the same way on every subsystem. Since the quantum maps considered have to be parametric in order to compose a meaningful learning model, the translational equivariance condition implies that we need to use *parameter sharing*. With this term we refer to the sharing of the same free parameter between all the identical gates that build a layer of an EQNN. For example, say that the GQML machinery has spotted two equivariant generators G_1 and G_2, respectively a single and two-qubit operators. Then we would apply $\exp[-i\theta_1 G_1]$ to every qubit, followed by $\exp[-i\theta_2 G_2]$ acting on every possible qubit pair, arranged in such a way to be translationally invariant. The result is a huge reduction in the complexity of the learning model, as the number of parameters no longer scale with the system size, but only with the number of equivariant generators, and with the number of layers we decide to stack to comprise the EQNN.
Now, the alternating model in Eq. (6.22), which we recall to have open boundary conditions, is clearly not translationally invariant. However, for large enough number of sites N, away from the boundaries any translation of the chain by two nodes leaves the model invariant, as every even (odd) node sees the same odd (even) nodes surrounding it and interacts with them through the same values of the alternating coupling constant. We can consider this as a *partial* symmetry of the alternating model, and exploit it to reduce the complexity of the resulting EQNN by adopting a weaker form of translational equivariance, especially in the large system size limits where boundary effects become less important. In our case, we decide to enforce parameter sharing, and we will discuss the details once we have laid down the $\mathbb{SU}(2)$-equivariant learning model in the next section.

6.4.2 $\mathbb{SU}(2)$-equivariant QCNN

Now that the task has been dissected, and that the symmetry underlying it has been identified, we can move to the step where we build our quantum learning model. Phase classification tasks, much like classical image classification ones, are best tackled via *quantum convolutional neural networks*. Classical convolutional neural networks (CNNs) [117, 221, 222] are a class of deep learning models specifically designed to process visual data. They are composed of multiple layers of interconnected nodes, each of which applies a combination of convolutional and pooling operations, akin to the ones we have described for EQNNs which in fact are inspired by CNNs, to the input data. Convolutional layers use learnable filters to extract relevant features from the input, such as edges, corners, and textures, while pooling layers reduce the size and complexity of the data by down-sampling the output of the convolutional lay-

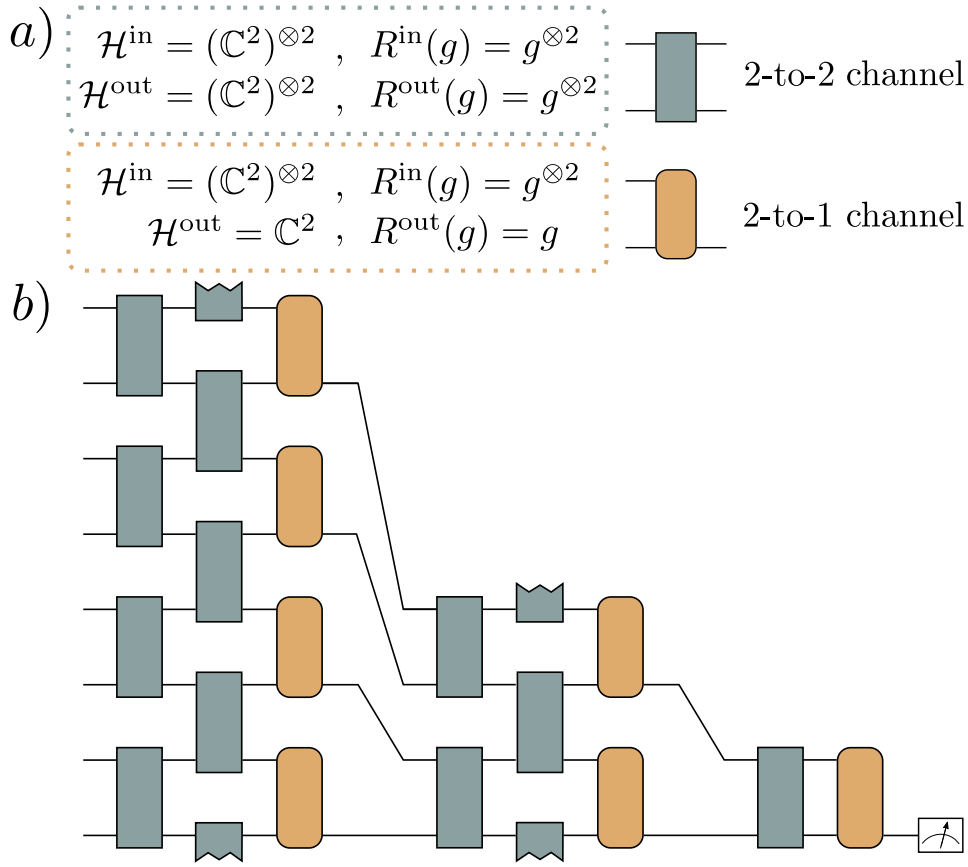


Figure 6.8: $\mathbb{SU}(2)$**-equivariant QCNN [116].** a) The task at hand is constructing 2-to-2 standard equivariant channels and 2-to-1 equivariant pooling channels. The input and output Hilbert spaces and the input and output representations for these channels are shown in the figure. b) In an $\mathbb{SU}(2)$-equivariant QCNN, the strategy is to alternate between applying 2-to-2 channels to adjacent qubits and using 2-to-1 equivariant pooling channels to reduce the dimensionality of the feature space.

ers. CNNs are trained on large datasets of labeled examples and can learn to recognize patterns and features in images, such as objects, faces, and texts. They have achieved state-of-the-art performance on many benchmarks and are widely used in applications such as image classification, object detection, and segmentation. CNNs are particularly well-suited for tasks that require learning from large amounts of unstructured data and are capable of learning from data with multiple channels, such as color images.

In QML, a quantum version of classical convolutional neural networks has been introduced in the seminal paper [223]. The architecture presented there takes inspiration from classical CNNs [117] and relies on error-correction techniques and tensor network description of quantum states [43], but equivariance, although present, is never mentioned.
QCNNs have been successfully implemented for error correction, quantum phase detection [223, 224], image recognition [225], and entanglement detection [150]. QCNNs exhibit several key features that make them promising architecture for the near-term, such as having a shallow depth or not exhibit barren plateaus [226].
Despite all the beneficial properties we just discussed, standard QCNNs need not respect the symmetries of a given task. In what follows, we will show how one can design equivariant layers for QCNNs, thus promoting them to group-equivariant QCNNs, that we will dub *EQCNNs*.

For ease of implementation on quantum hardware we restrict ourselves to channels with locality constraints (see Lemma 1). That is, as illustrated in Fig. 6.8 we want to build an equivariant QCNN that is hardware friendly, only implementing 2-to-2 standard equivariant unitary channels on adjacent qubits, that we will then stack in a brick-layered fashion as per usual, and 2-to-1 equivariant pooling maps. This choice of architecture sacrifices some expressibility in favor of locality, as there may be more general equivariant channels that operate on multiple or even on all of the qubits. However, previous research [223, 226] has shown that models with locality constraints can be successful, suggesting that this may be a worthwhile approach to explore.

2-to-2 standard layers

In the special case of 2-to-2 equivariant *unitary* layers, where $\mathcal{N}^l_{\boldsymbol{\theta}_l} : (\mathbb{C}^2)^{\otimes 2} \to (\mathbb{C}^2)^{\otimes 2}$ and $\mathcal{N}^l_{\boldsymbol{\theta}_l}(\rho) = U_l(\theta_l)\rho U_l(\theta_l)^\dagger$, we know that if $U_l(\theta_l) = e^{-i\theta_l H_l}$, it suffices to use equivariant generators, that is, such that $[H_l, g^{\otimes 2}] = 0$ for all $g \in \mathbb{SU}(2)$. Now, $\mathbb{SU}(2)$ being a continuous Lie group, we should embark in the arguably difficult quest of, say, twirling by performing an Haar integration as in Eq. (6.18). However, much more conveniently, we can resort to a famous result of representation theory, the Schur-Weyl duality [151, 227], which states that the only possible equivariant operators with respect to the tensor product representation of $\mathbb{SU}(2)$ over two qubits are $\mathbb{1}$ and the SWAP operator that we defined in Eq. (2.38), which correspond to the two elements of the

qubit-permutational representation of S_2. Thus, without loss of generality, we can set $H_l = \text{SWAP}$ so that our equivariant standard layers will consist of unitaries of the form $U_l(\theta_l) = e^{-i\theta_l \text{SWAP}}$. Following Lemma 1, we know that if we compose these two-qubit equivariant unitaries as in Fig. 6.8, the result will be an n-qubit equivariant unitary.

2-to-1 pooling layers

Next, we look for 2-to-1-qubit equivariant channels using the nullspace approach. Since $\mathbb{SU}(2)$ is a Lie group, we can work at the level of the generators of its Lie algebra, $\mathfrak{su}(2) = \text{span}\{X, Y, Z\}$. Given the representations $g^{\otimes 2}$ (the input representation) and g (the output one), the associated basis representations of the algebra are $\{\mathbb{1} \otimes X + X \otimes \mathbb{1}, \mathbb{1} \otimes Y + Y \otimes \mathbb{1}, \mathbb{1} \otimes Z + Z \otimes \mathbb{1}\}$ and $\{X, Y, Z\}$. Plugging this in the nullspace machinery that we described in sec. 6.3.2, we arrive at having to simultaneously solve for the nullspace of the following matrices

$$\begin{aligned} M_X &= \overline{\text{ad}_{\text{IX+XI}}}^{\top} \otimes \mathbb{1}_2 - \mathbb{1}_4 \otimes \overline{\text{ad}_X}\,, \\ M_Y &= \overline{\text{ad}_{\text{IY+YI}}}^{\top} \otimes \mathbb{1}_3 - \mathbb{1}_4 \otimes \overline{\text{ad}_Y}\,, \\ M_Z &= \overline{\text{ad}_{\text{IZ+ZI}}}^{\top} \otimes \mathbb{1}_3 - \mathbb{1}_4 \otimes \overline{\text{ad}_Z}\,. \end{aligned} \tag{6.26}$$

The solution comprises five superoperators that form a basis for 2-to-1 qubit $(\mathbb{SU}(2), g^{\otimes 2}, g)$-equivariant maps. These are

$$\begin{aligned} &\phi_1(\rho) = \text{Tr}[\rho]\frac{\mathbb{1}}{2}\,, \quad \phi_2(\rho) = \text{Tr}[\rho\text{SWAP}]\frac{\mathbb{1}}{2}\,, \\ &\phi_3(\rho) = \text{Tr}_A[\rho]\,, \quad \phi_4(\rho) = \text{Tr}_B[\rho]\,, \\ &\phi_5(\rho) = \sum_{ijk=1}^{3} \text{Tr}[\rho\sigma_i\sigma_j]\epsilon_{ijk}\sigma_k\,. \end{aligned} \tag{6.27}$$

Notice how the first two channels $\phi_{1,2}$ are just equivariant measurements, since as we discussed the commutant of the tensor product representation of $\mathbb{SU}(2)$ is just $\{\mathbb{1}, \text{SWAP}\}$, followed by setting the output state to the completely mixed one. The third and fourth solutions $\phi_{3,4}$ are simple to interpret as well, as they correspond to tracing out either one of the two input qubits. The last solution ϕ_5 is more interesting, and unexpected. We dubbed it the *cross-product map* as it act just like it w.r.t. the components X, Y, Z. Note that $\phi_1, \phi_3,$ and ϕ_4 are trace preserving while ϕ_5 is trace-annihilating. One can also verify that ϕ_2 may non-trivially alter trace. As the only map that may do so, we can drop it from our basis set for being non-physical. To continue and find the set of equivariant quantum channels, we first make a modification to our basis set. In the Pauli basis we have $\phi_1 \leftrightarrow 2\,|\mathbb{1}\rangle\rangle\langle\langle\mathbb{1}, \mathbb{1}|$, and it is easy to see that both ϕ_3 and ϕ_4 also contain this term. Thus, we can remove it, leaving trace-annihilating versions of partial trace, which we will denote by ϕ_3' and ϕ_4'.

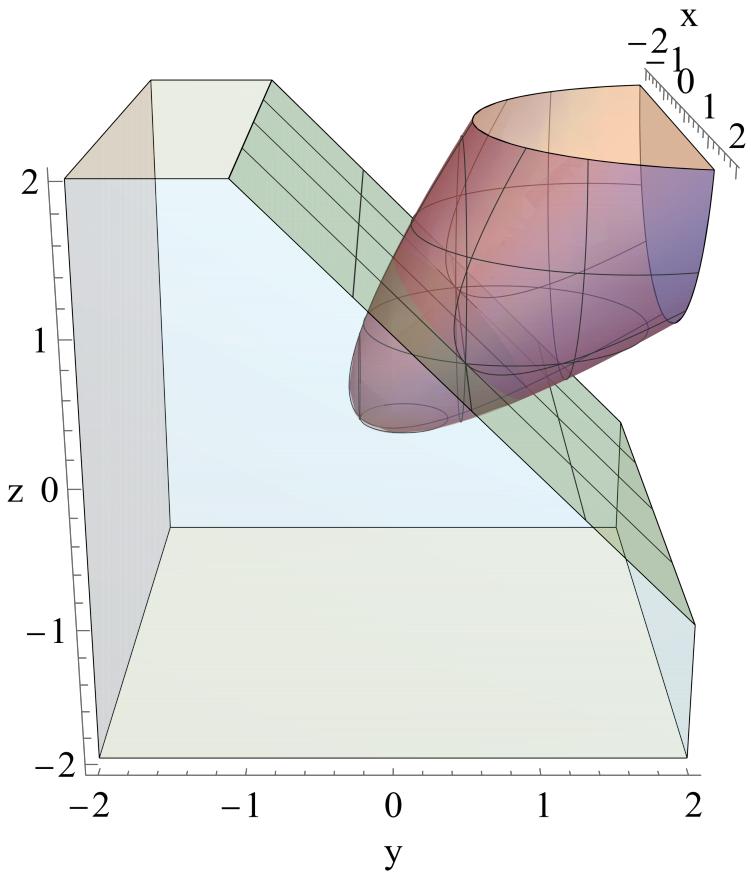


Figure 6.9: **Region of parameter space leading to CPTP channels [116].** Using the nullspace method we can find a basis for all 2-to-1 $(\mathbb{SU}(2), g^{\otimes 2}, g)$-equivariant pooling maps. These can then be linearly combined to form a general parametrized equivariant map as in Eq. (6.28), and we find in Eq. (6.30) the region in parameters space leading to CPTP channels. Here we depict said region as the volume of the hyperbole (red) below the plane (green).

Thus, any TP map must take the form

$$\phi(x, y, z) = \phi_1 + x\phi_5 + y\phi_3' + z\phi_4' \,, \tag{6.28}$$

where the coefficients are real numbers. It remains to find the region such that this channel is CP. This can be done via the Choi operators of these channels. That is, we would like to find

$$\{x, y, z \in \mathbb{R} : J^{\phi_1} + xJ^{\phi_5} + yJ^{\phi_3'} + zJ^{\phi_4'} \geqslant 0\} \,. \tag{6.29}$$

Note that, since the Choi operators in the sum are linearly independent, the coefficients here must be real numbers for the Choi operator of the sum to be positive. Requiring the eigenvalues of this linear combination to be non-negative yields the feasible region

$$\begin{aligned} x, y, z : y + z \leqslant 1 \,, \quad &\text{and} \\ y + z \geqslant \sqrt{3x^2 + 4(y^2 - yz + z^2)} - 1 \,. \end{aligned} \tag{6.30}$$

This region is illustrated in Fig. 6.9. Here we note that as the set of equivariant channels is convex, this feasible parameter region is a convex subset of $\mathbb{R}^3$.

When training a $\mathbb{SU}(2)$-equivariant quantum convolutional neural network (QCNN), it is possible to directly optimize the coefficients x, y, and z of each pooling channel $\phi(x, y, z)$ (as defined in Eq. (6.28)). This is done using methods such as gradient descent, which updates the parameters $(x^{(t+1)}, y^{(t+1)}, z^{(t+1)}) \leftarrow (x^{(t)}, y^{(t)}, z^{(t)}) - \alpha D_t((x^{(t)}, y^{(t)}, z^{(t)}))$ at each iteration t. To ensure that the resulting operations are physically allowed, we must continuously solve the projection problem at each iteration. This can be expressed as a convex optimization problem

$$\begin{aligned} &\min_{x,y,z} \|(x^{(t+1)}, y^{(t+1)}, z^{(t+1)}) - (x, y, z)\|^2 \,, \\ &\qquad \text{subject to Eq. (6.30)} \,, \end{aligned} \tag{6.31}$$

over a convex domain [116].

6.4.3 Preliminary Numerics

In this final section we are going to showcase the power of GQML, comparing the performance of the $\mathbb{SU}(2)$-equivariant QCNN we just found the ingredients of against a problem-agnostic QCNN for the phase classification task at hand. Let us start by setting up the learning problem. Looking at Fig. 6.7, we see that ground states of the Bond-Alternating Heisenberg XXX model of Eq. (6.22) can either be found in a trivial $J_2 < J_1$ or topologically protected $J_2 > J_1$ phase. Thus, we want to train a quantum classifier $y_{\boldsymbol{\theta}}$ to distinguish between these two phases. Conveniently, the alternating model, we will call it this way from now on, can be straightforwardly implemented on a quantum computer by associating a qubit to each of the spins S_i, thus, if the model is N qubits large the classifier must be a map $y_{\boldsymbol{\theta}} : \mathcal{B}((\mathbb{C}^2)^{\otimes N}) \to \{0, 1\}$, where we arbitrarily labeled 0 the trivial phase and 1 the topological one.

Defining the learning models

In the previous section we have built the equivariant 2-to-2 and 2-to-1 quantum maps that we can compose to craft our EQCNN, thus the only thing that is left to completely specify an equivariant classifier $y_{\boldsymbol{\theta}}$ is to compose those maps and pick an appropriate measurement at the end.
In order for $y_{\boldsymbol{\theta}}(|\psi\rangle_g^{J_2/J_1})$ to be an invariant prediction of the phase which $|\psi\rangle_g^{J_2/J_1}$ belongs to, we need to pick an equivariant observable operator $\hat{O}$ to measure at the end of the EQCNN. As we discussed in the previous sections, equivariant operators are those belonging to the commutant of the output representation R^{out} of the last layer of the EQCNN. We will not change the nature of the representation between layers, meaning that we will stick to the natural tensor product one acting on the physical system $R_{\text{tens}}(g) = g^{\otimes n} = R^n$, where n will start from $n = N$ and change layer after layer as we perform pooling operations to reduce the size of the register. This means that in the output layer the representation will be $R^{\text{out}}(g) = g^{\otimes m}$, with m being the number of qubits that survived the pooling layers. Now, since we need two outputs to label the two phases $y_{\text{trivial}} = 1$ and $y_{\text{topological}} = 0$, one might think that it is convenient to go all the way up to $m = 1$, since the lowest-dimensional operator with two distinct eigenvalues obviously belong to $\mathcal{B}(\mathbb{C}^2)$. However, from the discussion in Sec. 6.4.2 we know that the commutant of the natural representation of $\mathbb{SU}(2)$ over a single qubit is just $\mathfrak{comm}(R_{\text{natural}}) = \{\mathbb{1}\}$, and that would not allow to classify anything. The best we can do is then going to $m = 2$, where as we have already seen when looking for 2-to-2 equivariant unitaries, the commutant contains SWAP as the only non-trivial element. Conveniently, SWAP has two eigenvalues ± 1 so that we can bind, say, the $+1$ outcome to y_{trivial} and the -1 one to $y_{\text{topological}}$. We adopt this strategy, with a little modification to have the output of the EQCNN, that we will indicate as $f_{\boldsymbol{\theta}}(\rho)$, take values in $[0, 1]$. Namely, we define

$$f_{\boldsymbol{\theta}}(\rho) = \frac{\text{Tr}[\phi_{\boldsymbol{\theta}}(\rho) SWAP] + 1}{2} \,. \tag{6.32}$$

Where $\phi_{\boldsymbol{\theta}}$ is the equivariant quantum neural network. Then, we can assign the predicted phase label to any input state ρ as

$$y_{\boldsymbol{\theta}}(\rho) = \begin{cases} \text{trivial} & \text{if} \quad f_{\boldsymbol{\theta}}(\rho) > \tau \\ \text{topological} & \text{if} \quad f_{\boldsymbol{\theta}}(\rho) < \tau \end{cases} , \tag{6.33}$$

for some threshold value τ, that can be taken to be $\tau = 0.5$ at the beginning of the learning loop.

To test this architecture against one that has no inductive biases, we will choose a QCNN whose standard layers are PQCs inspired by the hardware efficient ansatz discussed in Sec. 3.3 and depicted on the left of Fig. 3.2, whereas the pooling layers will consist of simple alternate partial traces, *i.e.* at each

pooling operation we discard half of the qubits. The classification will then proceed as for the EQCNN, with a SWAP measurement and the assigning of the phases described in Eq. (6.33). To summarize, the general architecture of the QCNNs that we will compare is the one depicted in Fig. 6.8, the EQCNN will use standard and pooling layers described in Sec. 6.4.2, whereas the non-equivariant uses the hardware efficient ansatz circuits shown on the left of Fig. 3.2 as standard layers and simple partial traces as pooling ones. For this reasons, we dub the non-equivariant QCNN as the HEA-QCNN.
Lastly, in the spirit of wanting to test the full power of equivariance, we enable parameter sharing in the EQCNN. Given the brick-layered structure of the standard layers, composed of two qubit operations acting on alternate even-odd pairs, we can use the same parameter for all the gates applied to qubit pairs with the same parity. Looking at Fig. 6.8, this means that in each standard layer, each of the two columns of 2-to-2 gates is controlled by a single parameter. This leads to having two parameters for each standard layer. If needed, we can repeat the standard layers more than once before applying the pooling layer. Parameter sharing is easily enforced in the latter too, as it suffices to use the same parameters $\boldsymbol{x} = (x, y, z)$ in each of the 2-to-1 channels.

Training loop

To train the quantum learning models we just finished describing, we use the standard ML pipeline of supervised learning.

1. We collect a training dataset $\mathcal{D}^{N_T}_{\text{train}}$, where N_T is the size of the dataset, by choosing some representative values of the parameter J_2 while always keeping $J_1 = 1$ and then analytically computing the ground states $|\psi\rangle_g^{J_2/J_1}$ of the Hamiltonian in Eq, (6.22). Knowing the phase diagram of the alternating model, which is shown in Fig. 6.7, especially that the critical value at which the transition happens $\alpha = J_2/J_1 = 1$, we can then associate these states with their true labels $y \in \{0, 1\}$. Particularly, we try training dataset made of $N_T = (2, 4, 6, 8, 10, 12)$ ground states, always distributed homogeneously in the range $J_2/J_1 \in [0, 1]$. For example for $N_T = 2$ we use $\mathcal{D}^2_{\text{train}} = \{(|\psi\rangle_g^{0.25}, 1), (|\psi\rangle_g^{0.75}, 0)\}$.

2. We initialize the learning model at hand, equivariant or not, with random parameters $\boldsymbol{\theta}$.

3. We select an optimizer for the learning model. In our case we always use ADAM [87], the golden standard of gradient-based optimization in ML.

4. For a number of *epochs* E, we divide the training dataset $\mathcal{D}^{N_T}_{\text{train}}$ in batches of size $n_{\text{batch}} = 2$. For each batch, the training states $|\psi_i\rangle$ are processed by the model to output the predicted label $y_{\boldsymbol{\theta}}(|\psi\rangle)$ and the minimum squared error loss function of Eq. (1.2) is computed by comparing the

predictions to the real labels y_i

$$L_{\boldsymbol{\theta}} = \frac{1}{n_{\text{batch}}} \sum_{i=1}^{n_{\text{batch}}} \left(y_{\boldsymbol{\theta}}(|\psi_i\rangle) - y_i \right)^2 . \tag{6.34}$$

We then compute the gradient of $L_{\boldsymbol{\theta}}$ and use the optimizer to update the model's parameters. The goal is to minimize $L_{\boldsymbol{\theta}}$, as it would mean that the model correctly labels the training points.

5. The QCNN outputs for the training states are used to update the threshold τ. Particularly, only the two training points that are closer to the critical value $\alpha = 1$ are considered, and the the threshold value is set to the average of the corresponding outputs.

6. As an additional figure of merit for the training we keep track of the *accuracy* of the model's predictions. The accuracy of a learning model is a common metric used to rate the model's performance [1, 2]. It is simply defined as the ratio of correct predictions over the size of the dataset

$$\text{Accuracy} = \frac{\#\,\text{correct predictions}}{\#\,\text{correct predictions} + \#\,\text{wrong predictions}} . \tag{6.35}$$

If the dataset is well balanced, meaning that all of the classes are represented equally, accuracy is a good indicator of the behaviour of the model.

7. During training, we test if the model is actually learning or just overfitting to the training data by computing loss and accuracy for a set of states randomly picked from a *test dataset* $\mathcal{D}_{\text{test}}$ that the model has not access to during the optimization. The size of this set is chosen to be equal to the size of the training batches.

8. At the end of the last epoch, we let the model predict the labels of the whole test dataset, and we compute its final accuracy as a measure of the goodness of the training. Then, we also plot the predicted phase diagram to get a visual proof of the performance of the model.

Performance comparison

We are now ready to illustrate the results of our numerics. First thing first, we must state that we have still not been able to fruitfully train the EQCNN when using the general 2-to-1 pooling layers described in Sec. 6.4.2, as the required projection onto the feasible CPTP region seems to inject instability in the optimizing procedure. We leave this to the full numerical analysis that we are soon to compile into a manuscript, and here focus on the more simple tracing pooling operations. That is, the EQCNN architecture is still that depicted in Fig. 6.8, but the pooling operations are just partial traces. With

this being said, the results of training EQCNN and standard HEA-QCNN are illustrated in Figs.6.10,6.11.

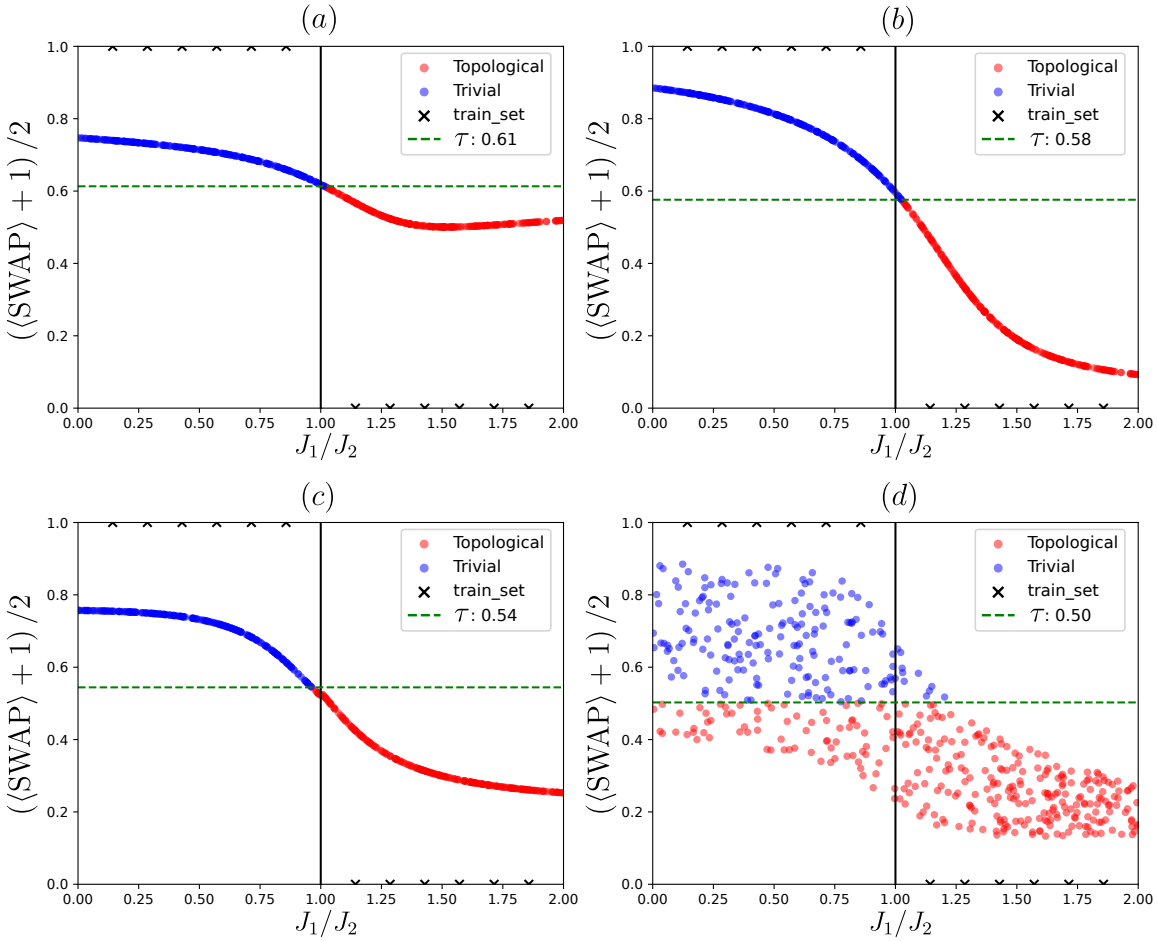


Figure 6.10: **Predicted Phase Diagrams.** The four panels show the phase diagram of the 1D Bond Alternating XXX Model for system sizes of $N = 12$ and $N = 13$ qubits as reconstructed by a trained EQCNN or HEA-QCNN. Particularly, each panel shows the QCNN output when it is tested against dataset of 500 homogeneously distributed ground states. States whose output is above (below) the optimal threshold τ (green dashed line) are colored in blue (red) and classified as belonging to the trivial (topological) phase. The training points are shown as black crosses. The vertical solid black line is the theoretical critical value $J_2/J_1 = 1$. The configurations leading to the panels are the following. (a): EQCNN, $N = 12$, 60 trainable parameters, 12 training points; (b): HEA-QCNN, $N = 12$, 63 trainable parameters, 12 training points; (c): EQCNN, $N = 13$, 66 trainable parameters, 12 training points; (d): HEA-QCNN, $N = 13$, 66 trainable parameters, 12 training points. Details about the training procedure are given in the main text.

We considered system sizes ranging from $N = 6$ to $N = 13$, and trained

both the EQCNN and the HEA-QCNN according to the training loop described in Section 6.4.3 for a fixed number of training epochs $E = 750$. Since the two architectures are very different, and the EQCNN, as opposed to the HEA-QCNN, uses parameter sharing, in order to have a fair comparison we decided to stack multiple standard layers before each pooling one in the EQCNN, in such a way to have a similar amount of training parameters for both the learning models. In Fig. 6.10 we show the predicted phase diagrams for $N = 12$ and $N = 13$. The thing that immediately stands out is the (d) panel of that figure. Indeed, while the other three basically show the same behaviour, with the QCNN at hand being able to efficiently separate the two phases of the alternating model, the last one showcases a cloudy behaviour of the HEA-QCNN predictions, as it randomly assigned phases even for really close states in the values of the parameter α. This is not an artifact of the training, and we did not handpick a poor trained HEA-QCNN, rather it is a feature of it and, when comparing this behaviour with the analogous one of the EQCNN (panel (c)), it explains the real power of equivariant quantum learning models. Indeed, the fact that for $N = 12$ (panels (a) and (b)) the equivariant QCNN does not outperform the non-informed one is due to the fact that there is actually no need for equivariance in that case. Indeed, equivariance is meant to enhance the performance of learning models that deal with invariant, under some symmetry group G, labels, but this invariance should not come from the invariance of the input states themselves. Think yet again of the classical problem of classifying images of cats and dogs, the labels, *i.e.* the semantic meaning of the images, are invariant if we translate the images, but the latter are obviously not. However, if we translate images that are completely black and white, instead of showing cats and dogs, the labels (the colors) are invariant simply because the images do not change. This is what is happening in our case. We have already discussed that if an Hamiltonian H is symmetric under a group G, any unitary representation of it U_G leaves the energy of groundstates unchanged and thus maps groundstates into groundstates. However, if the groundstate is unique, the action of U_G on it can at most modify its global phase $U_G |\psi_g\rangle = e^{i\phi} |\psi_g\rangle$. For quantum states, this is the same as not changing at all, hence phase classification when groundstates of the system are unique in every phase does not require equivariance. However, this does not hold for degenerate groundstates, *i.e.* when the Hamiltonian symmetry is broken, as in that case groundstates are not unique but rather populate a degenerate eigenspace, and the action of the symmetry group can move us through this space. Degenerate groundstates are akin to images of cats and dogs just like unique ones can be paired with black or white images, and as such equivariance can finally shine. Interestingly, the alternating model is degenerate for odd system sizes, while for even ones its groundstates are unique. This explains the different behaviours shown in Fig. 6.10.

This also motivates the analysis shown in Fig. 6.11. There, we show a

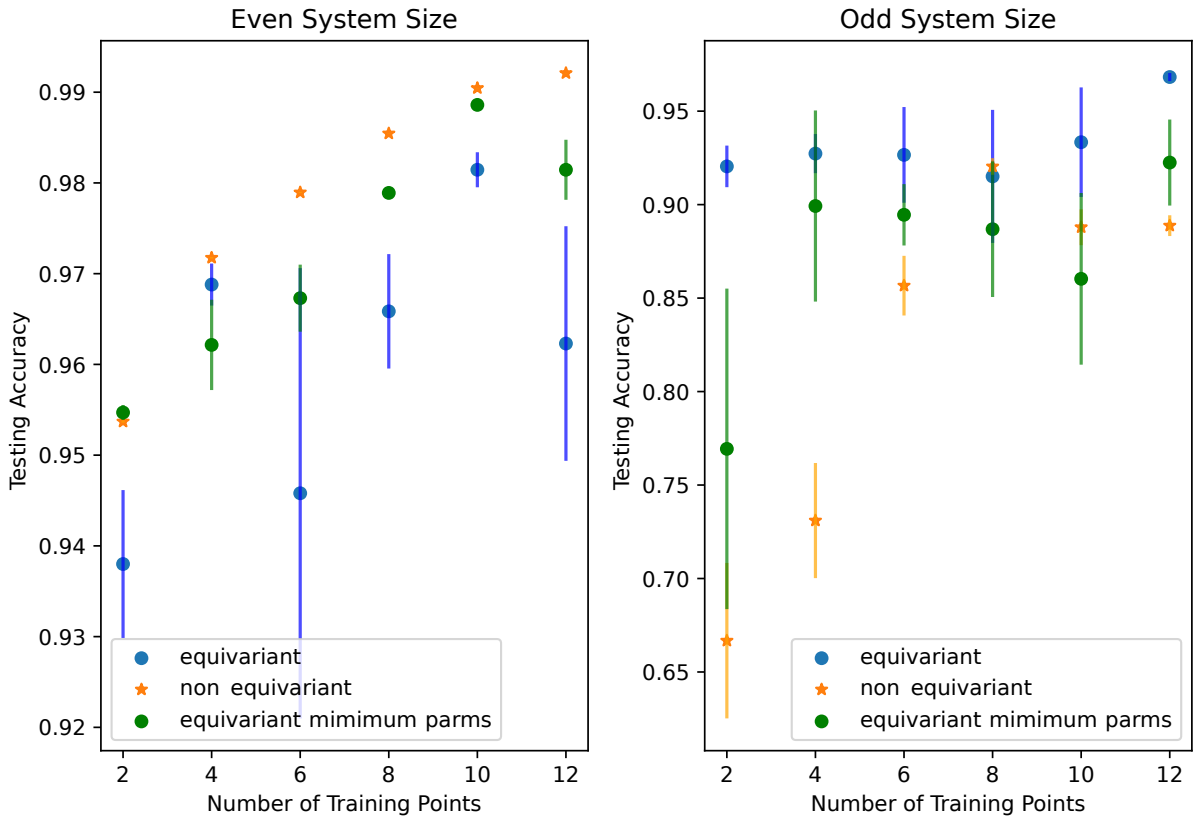


Figure 6.11: **The actual power of equivariance.** The two panels show the mean and variance of the testing accuracy reached by trained QCNNs on alternating XXX models of even (a) and odd (b) sizes. The average is conducted on both the chosen sizes, $(6, 8, 10, 12)$ for the even case and $(7, 9, 11, 13)$ for the odd one, and on 10 different, randomly initialized training runs for each problem size. The results are plotted against the number of training datapoints N_T. The blue circles refer to the EQCNN with a similar number of parameters as the HEA-QCNN (orange stars), whereas the green circles describe the EQCNN with the minimum number of parameters possible.

statistical study of the performances of EQCNN and HEA-QCNN when tackling even and odd system sizes. As is evident from the left panel, enforcing equivariance when it is not needed can be more detrimental than beneficial. Indeed, the reduced expressibility of the learning model is not compensated by any benefit and training instabilities emerge, as evidenced by the large error bars in panel (a). However, when equivariance has a reason to be used, as it is for the odd size states studied in panel (b), the advantage of using the EQCNN against a non-informed one is clear. Already with only two training points the equivariant QCNN performs greatly, while the HEA-QCNN needs more training data to generalize well. Interestingly, even with the minimum number of trainable parameters, that for the system sizes studied ranges from 4 to 6, the EQCNN seems to perform better than the non-equivariant one.

As stated in the beginning, this is only a preliminary analysis, and as such we postpone any strong conclusion until we have studied the performance of equivariant quantum learning models on more complex systems, against different non-equivariant architectures, and for different symmetry groups. Nonetheless, we think that the results shown in this section hint at confirming that injecting inductive biases into quantum neural networks boosts their performance, paving the way to the design of new, shallower and more efficiently implementable quantum circuits that could be more suitable for the current era of quantum machine learning.

Conclusions

In this thesis we have dealt with the challenge of finding efficient designs and training strategies for quantum machine learning models. Particularly, the first part of this work has been devoted to the particular framework of quantum generative adversarial learning. Quantum generative adversarial networks (QGANs) are a promising avenue for QML, but in the literature they had been studied only with pure states. Since at this stage of quantum technologies, the so-called NISQ era, quantum processors are expected to be noisy,the restriction to pure states could lead to undesirable surprises once the QGAN is trained in a real device. Thus, we decided to tackle mixed states learning in the adversarial framework (Chapter 4). In Section 4.3, we showed that alternative approaches to parameterized quantum circuit (PQC) modeled generator and discriminator, which we dubbed *ConvexQGAN*, achieve very good performance in learning mixed states, and have the benefit of having a nice physical interpretation. We also discussed a way to devise these convex methods on a quantum processor by using quantum circuits, however implementing these circuits on a real device would be costly. In contrast, the standard QML approach using PQCs with no predetermined structure revealed that basic gradient descent optimization strategies incur in limit-cycles-like behaviors (Section 4.2.2), making the QGAN game unstable and effectively making it so that mixed states cannot be learned. The convergence problems we have experienced are thought to be caused by the bilinear structure of QGAN's score function. It has indeed been noted in previous research about classical GANs that optimizing these types of score functions can lead to exact limit cycles, in which the generator gets trapped, repeatedly approaching but never reaching the desired solution. We faced this challenge in Section 4.2.3 and found an optimizing protocol, the Optimistic Gradient Descent algorithm, *i.e.* a gradient-based technique allowing provable convergence with bilinear score functions, that is able to solve the instability issues of mixed states QGAN, reopening the quest for a fully operative quantum generative adversarial learning framework.
According to our theoretical and numerical analysis, the proposed algorithms should be more effective at training QGANs, particularly when highly mixed states are involved, compared to previously used techniques. By developing effective training strategies for mixed states, we can take advantage of their higher representation power and study noisy quantum maps. Analysing the performance of QGANs in the presence of noise is mandatory if we want to ef-

ficiently devise them on NISQ processor. Encouragingly, as was demonstrated in [228], adversarial schemes exhibit the same noise robustness as other hybrid quantum-classical variational algorithms [229–232]. It is currently an open question whether adversarial strategies could potentially replace some current metrology schemes [112] by providing faster and more efficient techniques for sensing and system certification. An immediate future direction in this line of work about QGANs is assessing what kind of performance enhancement can be reached by processing entangled copies of the target state, if accessible.

Motivated by the ability of the optimistic training strategy to make the generative adversarial game converge even when mixed states are involved, in Chapter 5 we introduced a new learning scheme, that we dubbed SuperQGANs for reproducing, and hence characterizing, noisy quantum maps. Specifically, we focused on Pauli channels and showed that SuperQGANs can effectively learn and reproduce this kind of maps, even when they have different types and amounts of correlations. The method also easily extends to more general random unitary maps. SuperQGANs use parameterized quantum circuits to model generative and discriminative agents that compete until the generator learns to accurately reproduce the target quantum map and fools the discriminator. The ability to separately tackle temporal or spatial correlations allows us to better classify the unwanted couplings that unavoidably spoil nowadays quantum computations in the NISQ devices. As shown in Ref. [233], noise affecting quantum processors can be controlled in such a way to be effectively described by Pauli channels via the implementation of randomized compiling. Thus, having automatic methods such as SuperQGANs to characterize the latter could help devise optimal error mitigation protocols. Without any constraints, the SuperQGAN implements full quantum process tomography, which is bound to scale exponentially with n, be it the number of probes in a spatially correlated configuration or the number of successive uses for a temporal one. However, more interestingly, the analysis that led to Fig. 5.7 shows that when the noise model is constrained to a given form, such as when we already have some insight on the type of correlations affecting the device at hand, the SuperQGAN method is efficient and the resources it needs do not scale with n. In addition, following the analysis of the teleportation-induced correlated quantum channels in Ref. [234], once our new protocol learns the probabilities $p_{\boldsymbol{k}}^{(n)}$ of the Pauli channels, one can also analytically calculate the corresponding quantum capacities (known only in a very few cases) and the distillable entanglement of a generic bipartite quantum state being exploited to implement a teleportation protocol that can be always mapped to a correlated Pauli channel. Indeed all these quantities are very simple analytical functions of the probabilities $p_{\boldsymbol{k}}^{(n)}$ [234, 235]. Therefore, these results are expected to find applications also in other fields other than quantum computing, as quantum communication and quantum cryptography. Notice, for instance, that quantum error correction and quantum teleportation are indeed the cru-

cial building blocks towards the feasible realization of the so-called Quantum Internet [236]. Lastly, we also showed how the SuperQGAN machinery fits the problem of quantum metrology, *i.e.* the estimation of a parameter upon which a certain quantum map depends. We applied it to the estimation of the phase shift induced by a Mach-Zehnder interferometer, and found the optimal setup under which the phase can be faithfully reconstructed. We believe that quantum metrology via SuperQGANs will be particularly useful for those scenarios when the theoretically optimal entangled input state is not known, since our method has the ability to cleverly combine input preparation and final measurement to find the best discrimination policy.

In the last part of this thesis we left the generative adversarial learning paradigm to tackle the design of quantum learning models from a more general point of view. Indeed, in Chapter 6 we introduced the framework of Geometric Quantum Machine Learning (GQML), a new and exciting field that aims at encoding helpful inductive biases derived from the symmetries of the problem at hand into the architecture of the learning models we want to train. This encoding is achieved by constructing quantum circuits that "commute" with the unitary representations of the symmetry group at hand, and this property is called equivariance. The field of GQML had already begun to be tackled [151–156], but these early contribution to the literature mainly dealt with unitary PQCs which maintain the same symmetry group representation throughout the computation. We showed how to generalize these previous results and presented a theoretical framework to understand, design, and optimize over general equivariant channels, which we have referred to as Equivariant Quantum Neural Networks (EQNNs). While presented in the setting of supervised learning, the framework is readily applicable to other contexts such as unsupervised learning [166, 167], generative modeling [70, 168–170] or reinforcement learning [171, 172].
The main result of the first part of Chapter 6 is the establishment of practical methods to construct EQNN layers. The first method, referred to as the nullspace method, vectorizes the equivariance constraints turning them into a system of linear matrix equations, which is then solved for the common null space. The second method uses the process of twirling across a group, which allows projecting any given channel onto the space of equivariant maps. We have left a third method for Appendix E, since it relies on more advanced representation theory tools that did not fit the flow of this work. We then proceeded to show how the found equivariant layers can be parameterized and in the last section of Chapter 6 we presented a preliminary numerical investigation on the benefits of using equivariant learning models as opposed to problem-agnostic ones. Particularly, we tackled the QML task of classifying phases of matter in a 1-D condensed matter model known as the Alternating Bond XXX Heisenberg model. To do so, we assessed that the problem has a $\mathbb{SU}(2)$ symmetry, and applied our techniques to promote standard Quantum

Convolutional Neural Networks (QCNNs) to group-equivariant QCNNs (EQCNNs). The results we obtained, although preliminary and yet to be backed up by further analyses, show that when the symmetry is such that the labels (phases) are invariant while the input datapoints (quantum groundstates) are not, the EQCNN is able to top a more general QCNN, already when using very few training points. On the other hand, when equivariance is not justified, *i.e.* when the symmetry makes the inputs invariant and as a consequence any learning model would output invariant labels, the reduced expressibility of the EQCNN hinders its performance as compared to standard QCNN.
The field of Geometric Quantum Machine Learning is bound to experience significant growth in the near future, as it offers methods for designing architectures and inductive biases that are well-suited for a given problem, allowing for the deployment of hopefully shallower and more NISQ-friendly quantum circuits. As already discussed, an immediate avenue of future research is that of extensively studying the performance of QML ansätze that are tailored to specific symmetries relative to problem-agnostic ones and classical learning models. Additionally, it would be valuable to investigate the implications of approximate equivariance, as near-term quantum hardware is subject to noise. One could study how noise affects equivariance and the performance of equivariant models, and develop a measure of equivariance breaking. Notably, it has been observed in the classical machine learning literature that relaxing strict equivariance can result in improved performance on certain tasks [237].
Another crucial application of this framework will be the development of appropriate techniques for embedding classical data into quantum states. At present, many existing methods for dealing with classical data use embedding architectures that fail to take into account or even destroy symmetries featured by the input data, as highlighted in recent literature [162, 238, 239]. Therefore, it is essential to design embedding schemes that preserve and promote these symmetries from the classical to the quantum realm. These will be the topics of our future works.

Appendix A

Basics of game theory

Game theory [21, 240, 241] is a branch of mathematics that deals with the strategic behavior of multiple independent players who have different objectives, and try to reach the best outcome for themselves. This is opposed to cooperative game theory, where players alliances are allowed, that we will not discuss here. At its core, game theory is concerned with modeling and analyzing the decision-making of the players. John Nash's results are used everyday to analyze the strategic possibilities of chess and poker players [242, 243] and has also numerous applications in fields such as economics and risk management [244]. It can also even be useful in the field of cybersecurity [245].

A *game* can be formalized as a tuple $G = (N, S, u)$, where N is the set of players, $S = S_1 \times S_2 \times ... \times S_n$ is the set of strategies for each player, and $u = (u_1, u_2, ..., u_n)$ is the *utility function* that maps a strategy profile $s = (s_1, s_2, ..., s_n) \in S$ to a utility vector $u(s) = (u_1(s), u_2(s), ..., u_n(s)) \in \mathbb{R}^n$, representing the utility or payoff that each player receives for a given strategy profile. A *strategy* for player i is a function $s_i : N \setminus i \to S_i$ that specifies the action that player i will take for every possible strategy profile of the other players. Strategies can be either *pure*, in which case a player always chooses the same action, or *mixed*, which is a probabilistic combination of pure strategies. The *strategy profile* $s = (s_1, s_2, ..., s_n)$ of a game is a complete specification of the strategies for all players.

In this context, a *Nash equilibrium* [21] is a preferred strategy profile $s^* = (s_1^, s_2^, ..., s_n^*)$ such that, for every player i

$$u_i(s_i^*, s_{-i}^*) \geqslant u_i(s_i, s_{-i}^*) \tag{A.1}$$

for all $s_i \in S_i$, where $s_{-i} = (s_1, s_2, ..., s_{i-1}, s_{i+1}, ..., s_n)$ is the strategy profile of all players except player i. That is, a Nash equilibrium is a strategy profile in which no player has an incentive to change their strategy, given the strategies of the other players.

Examples explain more than raw theory, and a classic example of a non-cooperative game is the *prisoner's dilemma* [246]. In this scenario, two crimi-

nals are held in separate cells and offered a deal by the police. If one prisoner confesses and provides evidence against their partner, they will enter a witness protection program and avoid prison, while their partner will serve a five-year sentence. If both prisoners confess, they will both serve a slightly reduced three-year sentence. However, if neither confesses, they will only serve a one-year sentence each. The game is represented in table A.1, with the rows

P_2 \ P_1	Confess	Conceal
Confess	(3,3)	(0,5)
Conceal	(5,0)	(1,1)

Table A.1: **The *Prisoner's Dilemma* utility table.**

representing the first player's strategies, the columns representing those of the second player, and the cells containing the number of years each player will serve in prison. In this game, the optimal solution for both players would be to make a binding agreement not to confess and only serve one year in prison. However, since this is not possible in a non-cooperative game, both players must watch out for betrayal and try to minimize their prison sentence. As a result, the solution to the game is for both prisoners to confess and serve three years in prison.

In this simple example we can use brute-force logic to come to understand that there is a Nash equilibrium, and that it corresponds to the solution we have outlined. However, more complex games may be difficult to tackle with brute-force approaches. *Kakutani's fixed point theorem* [68] is a fundamental result in game theory that establishes the existence of Nash equilibria in certain types of games. The theorem, whose proof we will not give here, states that, given a compact and convex strategy set S_i for each player i and a continuous utility function $u_i : S \to \mathbb{R}$, there exists a Nash equilibrium in pure strategies. In other words, if the strategy sets for each player are compact, *i.e.* closed and bounded, and convex, and the utility function is continuous, then there exists a pure strategy Nash equilibrium in which no player has an incentive to change their strategy given the strategies of the other players. Kakutani's fixed point theorem can be used to prove the existence of Nash equilibria in a wide range of games, including two-player zero-sum games such as the one underlying GANs (Sec. 1.4) and QGANS (Chapter 4), and potential games [247].

Appendix B

Proof of Theorem 2

The following proof is adapted from [81]. We use the Bloch parametrization (4.8), which we rewrite for simplicity as $S(\boldsymbol{d}, \boldsymbol{g}) = \boldsymbol{d} \cdot (\boldsymbol{r} - \boldsymbol{g})$ ignoring the constant factors. Gradient descent/ascent applied to $\min_{\boldsymbol{g}} \max_{\boldsymbol{d}} S(\boldsymbol{d}, \boldsymbol{g})$ results in the update rule

$$\boldsymbol{d}_{t+1} = \boldsymbol{d}_t + \eta(\boldsymbol{r} - \boldsymbol{g}_t), \tag{B.1}$$

$$\boldsymbol{g}_{t+1} = \boldsymbol{g}_t - \eta(-\boldsymbol{d}_t), \tag{B.2}$$

where η is a suitably small "learning rate" and t is the iteration step. The unique fixed point is with $\boldsymbol{g} = \boldsymbol{r}$ and $\boldsymbol{d} = 0$, which physically corresponds to perfect generation $\rho_G = \rho_R$ and impossibility to distinguish real from generated data $\Pi_D \propto \mathbb{1}$. We evaluate the distance from this fixed point as $\Delta_t = \|d_t\|_2^2 + \|r - g_t\|_2^2$, where $\|\cdot\|_2$ is the ℓ_2-norm. From (B.1)-(B.2) we get

$$\|\boldsymbol{d}_{t+1}\|_2^2 = \|\boldsymbol{d}_{t+1}\|_2^2 + 2\eta\, \boldsymbol{d}_t \cdot (\boldsymbol{r} - \boldsymbol{g}_t) + \eta^2 \|\boldsymbol{r} - \boldsymbol{g}_t\|_2^2, \tag{B.3}$$

$$\|\boldsymbol{r} - \boldsymbol{g}_{t+1}\|_2^2 = \|\boldsymbol{r} - \boldsymbol{g}_t\|_2^2 - 2\eta\, \boldsymbol{d}_t \cdot (\boldsymbol{r} - \boldsymbol{g}_t) + \eta^2 \|\boldsymbol{d}_{t+1}\|_2^2, \tag{B.4}$$

and accordingly

$$\Delta_{t+1} = (1 + \eta^2)\Delta_t \ , \tag{B.5}$$

namely the distance from the equilibrium point increases at each iteration. We point out that the above proof is valid only when we neglect the physical constraints on the Bloch vectors. The latter are however important for pure states or for states near the surface of the Bloch sphere.

Appendix C

Method

C.1 SuperQGAN setup

Here we describe the technicalities of the numerical simulations described in the main text. First of all let us introduce the building blocks of the parametric quantum circuits used in both the spatial and temporal configurations. These are basically two: a two-qubit generic $SU(4)$ operation, obtained with the recipe described in [83], and the *quantum convolutional neural network* (QCNN) introduced in [223] and illustrated in Fig. C.1. Particularly, the final PQC controlled by D to implement its POVM is a QCNN, whereas intermediate operations are composed by alternating layers of $SU(4)$.

Another aspect that is common to both configurations is gradients evaluation. Indeed, we have always resorted to the *parameter shift rule* described in [85, 248].

Lastly, as far as the optimization procedure is concerned, we have used a variation of the *Optimistic Mirror Descent* (OMD) introduced in [81] and fruitfully tested in [67], namely *Optimistic ADAM*. This optimization strategy uses ADAM [87], i.e. a famous gradient descent with momentum optimizer, to evaluate the parameters increment at step t, i.e. δ_t, then implements it as in OMD $\theta \leftarrow \theta - 2\eta\delta_t + \eta\delta_{t-1}$, where η is the learning rate and θ is the parameter being updated.

Hyperparameters such as the agents' learning rates, the total number of training turns and the single agents' number of updating steps are tuned case by case. There is, however, a rule of thumb: learning rates η are chosen inside the range $0.01 < \eta < 0.25$ and D's (and I's) steps are almost always twenty times more than G's ones.

C.1.1 Spatial correlations

The circuits architecture of the SuperQGAN for spatially correlated channels is shown in Fig. 5.2(a). As discussed before, I is built of staggered layers of $SU(4)$ blocks and its depth is hand-tuned depending on the number of qubits

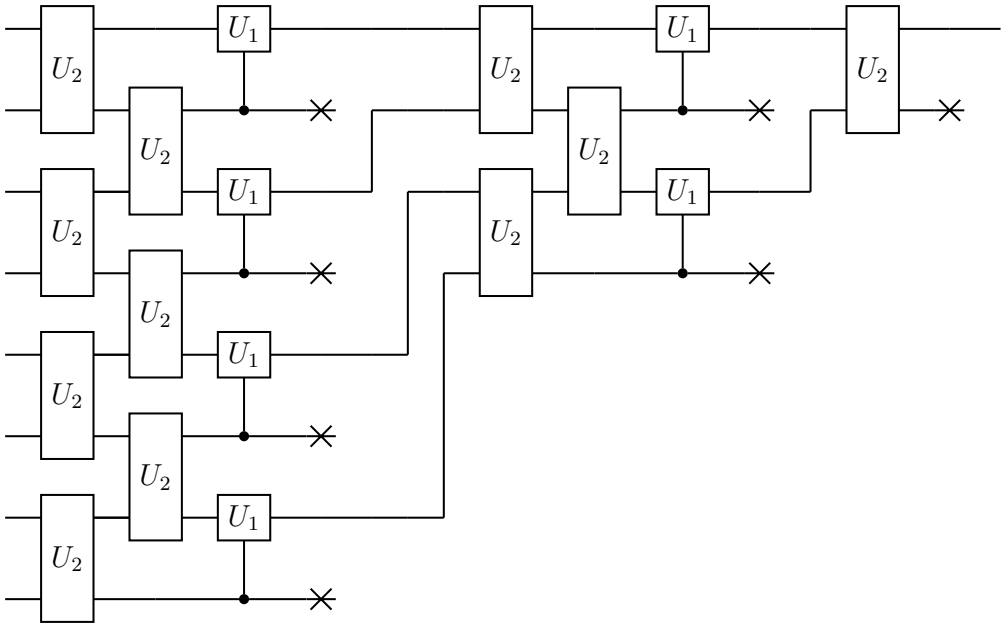


Figure C.1: **An example of QCNN.** U_1 and U_2 are elements of $SU(2)$ and $SU(4)$, respectively, and their circuital realization is explained in Fig. 3 of [67]. Crossed lines stand for *forgotten* qubits. Notice that the latter do not get measured, and are to be kept untouched until the end of the network.

it acts on. G simulates the random unitary channel at hand by separately applying the unitaries defining it and later weighting the discriminator outcomes with its parametric distribution q_k. The discriminator evaluates $p(R|\cdot)$ applying its QCNN on the output state of the channel being tested, R or G, and on its ancilla qubit. The role of the QCNN is to filter and encode the relevant information in the ancilla qubit, so that measuring it D can tell the difference between real and generated data.

C.1.2 Temporal correlations

When tackling temporal correlations, the SuperQGAN adopts a comb-like architecture, as shown in Fig. 5.2(b). Now I and D have access to an extra *workspace* qubit, which they will use to store information and process the temporal memory of the channel being tested. Now, after the initial state preparation implemented by I, whose structure is the same as the previous setup, D acts after each channel use with a similar PQC, and only after the last use implements the measurement via a QCNN.

Appendix D

A deeper representation-theoretic look at EQNNs

In the main text, in chapter 6, we showed how to put inductive biases in quantum machine learning models in order to make them better at their tasks. All the machinery we developed there is fundamentally based on representation theory, of which we gave a brief introduction in section 6.2.1. However, to keep the exposition more fluid and present the numerical analysis of section 6.4 without introducing unnecessary details, we omitted most of the representation theory based theoretical results for geometric quantum machine learning [116]. Thus, in this final appendix we are going to showcase a deeper analysis of the properties of equivariant quantum neural networks. To do so, we start by defining the key object of representation theory [157, 160], *i.e.* irreducible representations

Definition 9. *(Irreps) A representation is said to be an irreducible representation (irrep) if it contains no non-trivial subrepresentations.*

Irreps are the fundamental building blocks of representation theory. Finite-dimensional representations, as the ones we work with that are all acting on $\mathcal{H} = \mathbb{C}^N$, admit a particular decomposition of the vector space they act on. Indeed, the latter can be expressed as a direct sum of irreducible subrepresentations. Doing so, one obtains the *isotypic* decomposition of a finite-dimensional representation

$$\mathcal{H} \cong \bigoplus_{\lambda} \mathcal{H}_\lambda \otimes \mathbb{C}^{m_\lambda}, \qquad R(g) \cong \bigoplus_{\lambda} R_\lambda(g) \otimes \mathbb{1}_{m_\lambda}, \tag{D.1}$$

where $\cong$ indicates that there exists a global change of basis matrix W that simultaneously block-diagonalizes the unitaries $R(g)$ for all $g \in G$. Here, λ labels the irreps, m_λ is the multiplicity of the irrep R_λ, *i.e.* the number of times it "appears" in the decomposition, and $R_\lambda(g) \in \mathbb{C}^{d_\lambda \times d_\lambda}$ acts on its associated vector (Hilbert) space $\mathcal{H}_\lambda$. Note that $\sum_\lambda d_\lambda m_\lambda = \dim(\mathcal{H})$.

The isotypic decomposition in Eq. (D.1) allows to understand how equivariant operations transform the states they act on. Particularly, it allows to detect which components they can and cannot modify. Thus, in the following we will show how the choice of the in and out representations of a $(G, R^{\text{in}}, R^{\text{out}})$-equivariant layer can influence the structure and action of it. Especially, we will be able to count how many degrees of freedom EQNNs have and understand what is the role of the intermediate representations between their layers.

D.1 Equivariant layers as Fourier space actions

First things first, how do equivariant quantum neural networks transform the data they process? To keep things simple, we start from layers such as the standard ones defined in Def. 8, and consider their actions on pure states of the form $\rho = |\psi\rangle\langle\psi|$. Moreover, we assume the input and output representations to be the same, $R^{\text{in}} = R^{\text{out}} = R$, and the layer to be unitary, $\mathcal{N}_{\boldsymbol{\theta}}(\rho) = U(\boldsymbol{\theta})\rho U(\boldsymbol{\theta})^\dagger$. For $\mathcal{N}_{\boldsymbol{\theta}}$ to be a (G, R, R)-equivariant channel, $U(\boldsymbol{\theta})$ must be (G, R)-equivariant operator belonging to the commutant $\mathfrak{comm}(R)$ defined in Def. 2. Thus, we can inspect how $U(\boldsymbol{\theta})$ acts on a quantum state $|\psi\rangle$ by studying the structure of the commutant.

Theorem 3 (Structure of commutant, Theorem IX.11.2 in [157])**.** *Let R be a unitary representation of a finite-dimensional compact group G. Then under the same change of basis W, which block diagonalizes R as in Eq.* (D.1)*, any operator $H \in \mathfrak{comm}(R)$ takes the following block-diagonal form*

$$H \cong \bigoplus_{\lambda} \mathbb{1}_{d_\lambda} \otimes H_\lambda, \tag{D.2}$$

where each H_λ is an m_λ-dimensional operator that is repeated d_λ times.

Theorem 3 implies that any equivariant unitary can be expressed in the form $U(\boldsymbol{\theta}) = W^\dagger \left(\bigoplus_\lambda \mathbb{1}_{d_\lambda} \otimes U_\lambda(\boldsymbol{\theta})\right) W$, meaning that it can only act non-trivially on the multiplicity space when expressed in the irrep basis. This can be compared to the classical machine learning literature, where it has been shown that linear equivariant maps can only act on the group Fourier components of the data [121, 122, 131, 135, 136]. Therefore, we can interpret EQNNs action as the following sequence: (i) the data is transformed to the generalized Fourier space $W|\psi\rangle = \bigoplus_\lambda \left|\overline{\psi}_\lambda\right\rangle \otimes |\psi_\lambda\rangle$, (ii) each Fourier component $|\psi_\lambda\rangle$ is acted upon by $U_\lambda(\boldsymbol{\theta})$ whereas their complements $\left|\overline{\psi}_\lambda\right\rangle$ are left unchanged, and (iii) the transformed data is transformed back with $W^\dagger$.

$$U(\boldsymbol{\theta})|\psi\rangle \cong \bigoplus_{\lambda} \left|\overline{\psi}_\lambda\right\rangle \otimes U_\lambda(\boldsymbol{\theta})|\psi_\lambda\rangle \ . \tag{D.3}$$

This workflow can then be extended to general equivariant quantum maps.

Notably, this also means that the components of the input state that an EQNN layer has access to are predetermined by the chosen representation of G. Explicitly, the EQNN cannot manipulate information stored in the components $\left|\overline{\psi}_\lambda\right\rangle$ of the input state. Nonetheless, by using different intermediate representation, we can avoid this pitfall.

D.2 Intermediate representations as hyperparameters

Let us now discuss how to exploit intermediate representations to tune how the EQNN, defined as in Def. 7, steers the quantum state it processes. Since representations are completely characterized by their irreps R_λ and corresponding multiplicities m_λ, we can consider the latter as the hyperparameter that we can control to change which components of the input state ρ get tackled layer after layer. From a mathematical standpoint, given a group G there are no particular criteria to prefer a representation over the others. However, once physics is brought up one can motivate such choices. The main criterion that physics allows to put on the table is naturalness. Indeed, although described by a complex mathematical machinery, symmetries are not abstract entities, but are instead related to the physical properties of the system at hand. Examples of natural representations are the tensor-product representation of $\mathbb{U}(2)$ over n-qubits, *i.e.* $R(U) = U^{\otimes n}$, that acts identically on each qubit, or the representation $R(g^t)\bigotimes_{j=1}^n |\psi_j\rangle = \bigotimes_{j=1}^n |\psi_{j+t \mod n}\rangle$ of the cyclic group $\mathbb{Z}_n$, whose action is to shift qubits cyclically between themselves. The first one is natural whenever the qubits being considered interact only via Hamiltonian terms that depend on their relative orientation, such as the model in Eq. (6.22) that we studied at the end of Ch. 6. Indeed it is clear, even before setting up any group-theoretical analysis, that rotating each qubit by the same amount leaves their relative orientations unchanged. The second example is a natural representation for fully translational invariant systems instead.

Moreover, one should refrain from using equivalent representations between different layers, as they do not change the information content of the model [164, 249]. This is proved in the following proposition [116]

Proposition 2 (Insensitivity to equivalent representations)**.** *Consider an EQNN as defined in Definition 7. Then, changing an intermediate representation, R^l, to another representation equivalent to it, $VR^lV^\dagger$, where V is a unitary, does not change the expressibility of the EQNN.*

Proof. Consider two EQNNs that undergo the same representations except at one place

$$\mathcal{N}: \qquad R^{\text{in}} \longrightarrow \ldots \longrightarrow R^1 \xrightarrow{\mathcal{N}^1} R \xrightarrow{\mathcal{N}^2} R^2 \longrightarrow \ldots \longrightarrow R^{\text{out}}. \tag{D.4}$$

$$\mathcal{N}': \qquad R^{\text{in}} \longrightarrow \ldots \longrightarrow R^1 \xrightarrow{\mathcal{N}^{1'}} R' \xrightarrow{\mathcal{N}^{2'}} R^2 \longrightarrow \ldots \longrightarrow R^{\text{out}}, \tag{D.5}$$

where $R' = VRV^\dagger$ for some unitary V.

Observe that the set of (G, R_1, R)-equivariant channels is in one-to-one correspondence to the set of (G, R_1, R')-equivariant channels. Indeed, for any (G, R_1, R)-equivariant $\mathcal{N}_1$, the channel $\mathcal{N}_1' = \mathrm{Ad}_V \circ \mathcal{N}_1$ is (G, R_1, R')-equivariant as we have that

$$\mathcal{N}^1 = \mathrm{Ad}_R \circ \mathcal{N}^1 \circ \mathrm{Ad}_{R_1^\dagger} = \mathrm{Ad}_R \circ (\mathrm{Ad}_{V^\dagger} \circ (\mathcal{N}^{1'}) \circ \mathrm{Ad}_{R_1^\dagger} \Leftrightarrow \mathcal{N}^{1'} = \mathrm{Ad}_{VRV^\dagger} \circ \mathcal{N}^{1'} \circ \mathrm{Ad}_{R_1^\dagger}. \tag{D.6}$$

Similarly, we obtain $\mathcal{N}^{2'} = \mathcal{N}^2 \circ \mathrm{Ad}_{V^\dagger}$. Thus $\mathcal{N}^{2'} \circ \mathcal{N}^{1'} = \mathcal{N}^2 \circ \mathcal{N}^1$. A similar argument can be made for changing between equivalent output representations, in which case there is a bijection between the observables that commute with R^{out} and those that commute with $R^{\mathrm{out}'}$. □

Lastly, let us note that when intermediate representations in equivariant neural networks are chosen to be regular representations, namely group actions on their own group algebra, the resulting networks are known as "homogeneous ENNs" in the classical literature [130]. In this case, any equivariant map can be expressed as a group convolution, which can be implemented using quantum algorithms [250]. Combining this with quantum algorithms for polynomially transforming quantum states [251, 252] allows classical homogeneous ENNs to be implemented on a quantum processor as a special case of EQNNs.

D.3 Free parameters in EQNNs

Thanks to the Fourier-space interpretation of EQNNs, we can now understand how many degrees of freedom their layers actually possess.

D.3.1 Unitary layers

For unitary layers of equivariant quantum neural networks the following holds [116]

Theorem 4 (Free parameters in equivariant unitaries)**.** *Under the same setup as Theorem 3, the unitary operators in* $\mathfrak{comm}(R)$ *can be fully parametrized by* $\sum_\lambda m_\lambda^2$ *real scalars.*

Proof. Any unitary U in $\mathfrak{comm}(R)$ takes the block-diagonal form $U = \bigoplus_\lambda \mathbb{1}_{d_\lambda} \otimes U_\lambda$ in the Fourier basis. Observe that the operators U_λ must also be unitaries since $U^\dagger U = \bigoplus_\lambda \mathbb{1}_{d_\lambda} \otimes U_\lambda^\dagger U_\lambda$. A unitary in $\mathbb{U}(m_\lambda)$ is parametrized by m_λ^2 real scalars, hence a total number of $\sum_\lambda m_\lambda^2$ parameters suffice to parametrize U. □

Theorem 4 illustrates the impact that symmetry has on the complexity of the problem. In particular, it shows that the larger the representations of G,

Group	Representation	Free parameters	$\mathfrak{comm}(R)$
None	$R_{\text{trivial}}(g) = \mathbb{1}_{2^n}$	4^n	$\mathbb{C}[\mathbb{U}(2^n)]$
$\mathbb{U}(2^n)$	$R_{\text{def}}(g) = g$	1	$\mathbb{C}[\mathbb{1}_{2^n}]$
$\mathbb{U}(2)$	$R_{\text{tens}}(g) = g^{\otimes n}$	$\frac{1}{n+2}\binom{2n+2}{n+1} \in \Omega(2^n)$	$\mathbb{C}[R_{\text{qub}}(S_n)]$
S_n	$R_{\text{qub}}(g)\bigotimes_{i=1}^n \lvert\psi_i\rangle$ $= \bigotimes_{i=1}^n \lvert\psi_{g^{-1}(i)}\rangle$	$\binom{n+3}{3} \in \Theta(n^3)$	$\mathbb{C}[R_{\text{tens}}(\mathbb{U}(2))]$

Table D.1: **Free parameters in unitary EQNNs [116].** Different symmetry groups G and their associated number of free parameters appearing in a (G, R, R)-equivariant unitary layer. Note that we are indicating as $\mathbb{C}[S] \equiv \text{span}_{\mathbb{C}}(S)$ the span of a generating set S.

the smaller the commutant and the fewer parameters required to completely characterize equivariant unitaries. Table D.1 [116] provides examples of different symmetries that result in a unitary EQNN having either exponentially many, polynomially many, or a constant number of free parameters.

D.3.2 Equivariant channels

Let us now consider the more general $(G, R^{\text{in}}, R^{\text{out}})$-equivariant quantum maps. Recall from Eqs. (2.28,2.29) that any channel $\phi : \mathcal{B}^{\text{in}} \to \mathcal{B}^{\text{out}}$ is completely characterized by its Choi state J^ϕ, such that its action upon a state $\rho \in \mathcal{B}^{\text{in}}$ can be written as $\phi(\rho) = \text{Tr}_{\text{in}}[J^\phi(\rho^\top \otimes \mathbb{1}_{\dim(\mathcal{H}^{\text{out}})})]$.

Equivariance can be expressed in term of Choi states through the following theorem [116]

Lemma 2 (Lemma 11 in [180] paraphrased)**.** *A channel ϕ is $(G, R^{in}, R^{out})-$equivariant if and only if $J^\phi \in \mathfrak{comm}(R^{in*} \otimes R^{out})$, where the $*$ symbol denotes complex conjugate.*

Using Lemma 2 and the Theorem 3, we can now count the degrees of freedom of equivariant channels [116]

Theorem 5 (Free parameters in equivariant channels)**.** *Let the irrep decomposition of $R := R^{in*} \otimes R^{out}$ be $R(g) \cong \bigoplus_q R_q(g) \otimes \mathbb{1}_{m_q}$. Then any (G, R^{in}, R^{out})-equivariant CPTP channels can be fully parametrized via $\sum_q m_q^2 - C(R^{in}, R^{out})$ real scalars, where $C(R^{in}, R^{out})$ is a positive constant that depends on the considered representations.*

Notice that this is possible thanks to $(R^{\text{in}*} \otimes R^{\text{out}})$ being a valid representation, since it satisfies Definition 1.

Proof. Let ϕ be a $(G, R^{\text{in}}, R^{\text{out}})$-equivariant channel. By Theorem 3 and Lemma 2, the Choi operator J^ϕ is decomposed as $J^\phi \cong \bigoplus_{q=1}^Q \mathbb{1}_{d_q} \otimes J_q^\phi$, where each J_q^ϕ is an operator in an m_q-dimensional subspace corresponding to the irrep decomposition $R := R^{\text{in}*} \otimes R^{\text{out}}$. For convenience of notation, we will denote

as $\mathcal{H}_B \otimes \mathcal{H}_A$ the Hilbert space over which J^ϕ acts. Imposing $J^\phi \geqslant 0$ (CP) is equivalent to imposing $J^\phi_q \geqslant 0$ for each irrep q. An m_q-dimensional complex positive semidefinite operator is parametrized by m_q^2 real scalars, for a total of $\sum_q m_q^2$ parameters. Next, we impose TP via $\mathrm{Tr}_B[J^\phi] = \mathbb{1}_A$, where $\mathbb{1}_A$ denotes the identity over $\mathcal{H}_A$. Let the change of basis in the irrep decomposition be W, i.e., $J^\phi = W^\dagger(\bigoplus_{q=1}^Q \mathbb{1}_{d_q} \otimes J^\phi_q)W$. The TP condition reads

$$\begin{aligned}\mathrm{Tr}_B[J^\phi] = \mathbb{1}_A &= \sum_j (\langle j|_B \otimes \mathbb{1}_A) W^\dagger \left(\bigoplus_{q=1}^Q \mathbb{1}_{d_q} \otimes J^\phi_q \right) W (|j\rangle_B \otimes \mathbb{1}_A) \\ &= \sum_j T_j^\dagger \left(\bigoplus_{q=1}^Q \mathbb{1}_{d_q} \otimes J^\phi_q \right) T_j,\end{aligned} \tag{D.7}$$

where $T_j = W(|j\rangle_B \otimes \mathbb{1}_A)$. Vectorizing the above equation, we can use the property $\mathrm{vec}(M_1 M_2 M_3) = (M_3^\top \otimes M_1)\mathrm{vec}(M_2)$ to obtain

$$\begin{aligned}D \cdot \mathrm{vec}\left(\bigoplus_{q=1}^Q \mathbb{1}_{d_q} \otimes J^\phi_q \right) &= \mathrm{vec}(\mathbb{1}_A), \\ \text{where } D := \sum_{j \in \dim(\mathcal{H}_B)} T_j^\top \otimes T_j^\dagger &\in \mathbb{C}^{\dim(\mathcal{H}_A)^2 \times (\dim(\mathcal{H}_A)\dim(\mathcal{H}_B))^2}.\end{aligned} \tag{D.8}$$

Let $\widetilde{D}$ be the $\dim(\mathcal{H}_A)^2 \times \sum_q m_q^2$ matrix whose columns correspond to the nonzero entries in $\mathrm{vec}\left(\bigoplus_{q=1}^Q \mathbb{1}_{d_q} \otimes J^\phi_q\right)$. Then $\mathrm{rank}(\widetilde{D}) = C(R^{\mathrm{in}}, R^{\mathrm{out}})$.

It is readily verified that in the non-equivariant case, i.e., $W = \mathbb{1}$ and J^ϕ is fully parameterized, the matrix $\widetilde{D} = D$ is full row-rank, in which case imposing TP reduces $\dim(\mathcal{H}_A)^2$ free parameters as expected. □

A way to characterize how many parameters we are able to save using equivariant operations as opposed to agnostic ones is to resort to the *parameter utilization* metric, that is used in classical machine learning [137]

$$\mu = \frac{\dim \mathrm{Hom}^{\mathrm{CPTP}}(R^{\mathrm{in}}, R^{\mathrm{out}})}{\dim \mathrm{Hom}_G^{\mathrm{CPTP}}(R^{\mathrm{in}}, R^{\mathrm{out}})}, \tag{D.9}$$

where $\mathrm{Hom}^{\mathrm{CPTP}}(R^{\mathrm{in}}, R^{\mathrm{out}})$ denotes the set of CPTP maps between $\mathcal{B}^{\mathrm{in}}$ and $\mathcal{B}^{\mathrm{out}}$ and $\mathrm{Hom}_G^{\mathrm{CPTP}}(R^{\mathrm{in}}, R^{\mathrm{out}})$ its $(G, R^{\mathrm{in}}, R^{\mathrm{out}})$-equivariant subspace. Note that, $\dim \mathrm{Hom}^{\mathrm{CPTP}}(R^{\mathrm{in}}, R^{\mathrm{out}}) = |\mathcal{H}^{\mathrm{in}}|^2|\mathcal{H}^{\mathrm{out}}|^2 - |\mathcal{H}^{\mathrm{in}}|^2$ [210]. That is, the larger μ, the larger the benefit of using an EQNN is, in the sense that available parameters are used more effectively.

Appendix E

Choi operator method

In the main text, we referenced a third method, together with the null space approach and the twirling technique of Sec. 6.3.2, for finding equivariant quantum maps. This method requires the advanced representation theoretical tools that we showed in App. D and we are thus now ready to present it. We dubbed it the *Choi operator method*.

In Eq. (2.28) we defined the Choi state of a channel ϕ as $J^\phi = \sum_{i,j} |i\rangle\langle j| \otimes \phi(|i\rangle\langle j|)$. Then, in Lemma 2 we stated that ϕ is an equivariant map if $J^\phi \in \mathfrak{comm}(R^{\text{in}*} \otimes R^{\text{out}})$,[1] but we can rephrase such condition as

$$J^\phi(R^{\text{in}}(g)^* \otimes R^{\text{out}}(g)) - (R^{\text{in}}(g)^* \otimes R^{\text{out}}(g))J^\phi = 0 \quad \forall\, g \in G\,. \tag{E.1}$$

Now, we can decompose the representation $R = R^{\text{in}*} \otimes R^{\text{out}}$ in irreps $R(g) \cong \bigoplus_q R_q(g) \otimes \mathbb{1}_{m_q}$ to rewrite the Choi sate as

$$J^\phi \cong \bigoplus_q \mathbb{1}_{d_q} \otimes J^\phi_q\,, \tag{E.2}$$

where the Choi operators J^ϕ_q are the ones associated with each of the irreps. This enables the crafting of equivariant quantum channels by picking the action that we want, expressed by the associated Choi states, on each irrep component of the target state, as exemplified in Fig. E.1

The outputs of this procedure need not satisfy any CPTP condition, indeed, as those found by the nullspace approach, they are general equivariant maps. However, imposing such conditions on Choi states is easy. Trace preserving is enforced by putting $\text{Tr}_{\text{in}}[J^\phi] = \mathbb{1}_{\dim(\mathcal{H}^{\text{out}})}$, where the trace is carried out over the input degrees of freedom. Complete positiveness instead translates into the Choi state being positive semidefinite, *i.e.* $J^\phi \geqslant 0$. This CP condition also allows us to rewrite J^ϕ as [185]

$$J^\phi \cong \bigoplus_q \mathbb{1}_{d_q} \otimes w_q^\dagger w_q\,, \tag{E.3}$$

[1] Recall that we decided to denote with $*$ the complex conjugate operation and with R^* the *dual* representation of R

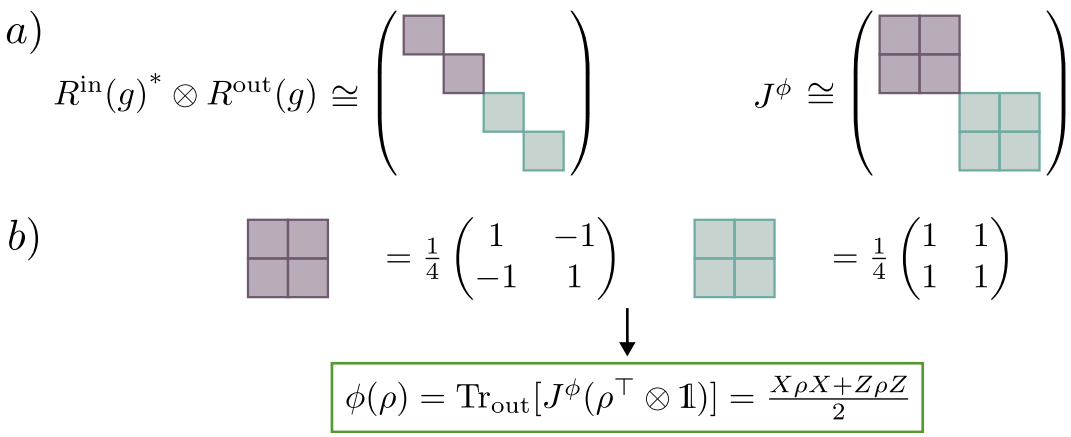


Figure E.1: **Example of the Choi operator method [116].** We demonstrate how to use the Choi operator method to determine the space of 1-to-1 qubit (G,R^{in},R^{out})-equivariant quantum channels, with $G = \mathbb{Z}_2 = \{e, \sigma\}$, $R^{\text{in}} = \{\mathbb{1}, X\}$ and $R^{\text{out}} = \{\mathbb{1}, Z\}$. a) Isotypic decomposition of the group representation . b) We show that a specific choice for the block-diagonal components of J^ϕ leads to the map $\phi(\rho) = (X\rho X + Z\rho Z)/2$.

with $w_k \in \mathbb{C}^{m_q \times m_q}$. Since expanding the TP conditions brings us to the constraint $\sum_q \text{Tr}_{\text{in}}[\mathbb{1}_{d_q} \otimes w_q^\dagger w_q] = \mathbb{1}_{\dim(\mathcal{H}^{\text{out}})}$, we can first obtain a basis of block-diagonal CP maps as in Eq. (E.3) and later impose the trace-preserving condition.

One drawback of the Choi operator technique is that it can be difficult to determine the isomorphism in the irrep decomposition of $R^{\text{in}^*} \otimes R^{\text{out}}$. While this decomposition can be implemented in software packages for common compact Lie groups [253, 254], this method is most suitable for local channels due to the size of the Choi operator scaling as $\dim(\mathcal{H}^{\text{out}})\dim(\mathcal{H}^{\text{in}})$. If the product $\dim(\mathcal{H}^{\text{out}})\dim(\mathcal{H}^{\text{in}})$ is not too large, it is possible to determine the change of basis leading to the isotypic decomposition and pick out maps with specific irrep actions.

Bibliography

[1] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[2] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[3] Maria Schuld and Francesco Petruccione. *Machine Learning with Quantum Computers*. Springer, 2021.

[4] David H Wolpert and William G Macready. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997.

[5] Google LLC. Google lens. `https://lens.google/`. Accessed: 2022-12-19.

[6] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3):279–292, 1992.

[7] Gavin A Rummery and Mahesan Niranjan. *On-line Q-learning using connectionist systems*, volume 37. University of Cambridge, Department of Engineering Cambridge, UK, 1994.

[8] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[9] Cheng Fan, Meiling Chen, Xinghua Wang, Jiayuan Wang, and Bufu Huang. A review on data preprocessing techniques toward efficient and reliable knowledge discovery from building operational data. *Frontiers in Energy Research*, 9:652801, 2021.

[10] WS McCelloch and Walter Pitts. A logical calculus of the idea immanent in neural nets. *Bulletin ofMathematical Biophysics*, 5:115–133, 1943.

[11] Gordon E Moore et al. Cramming more components onto integrated circuits, 1965.

[12] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[13] Michael A Nielsen. *Neural networks and deep learning*, volume 2018. Determination press San Francisco, CA, USA:, 2015.

[14] Josh Patterson and Adam Gibson. *Deep learning: A practitioner's approach*. " O'Reilly Media, Inc.", 2017.

[15] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[16] Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

[17] In Jae Myung. Tutorial on maximum likelihood estimation. *Journal of mathematical Psychology*, 47(1):90–100, 2003.

[18] Stephen Boyd, , and Lieven Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

[19] Muhamet Kastrati and Marenglen Biba. A state-of-the-art survey of advanced optimization methods in machine learning. In *RTA-CSIT*, pages 1–10, 2021.

[20] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, Red Hook, NY, USA, 2014. Curran Associates, Inc.

[21] John Nash Jr. Non-cooperative games. In *Essays on Game Theory*, pages 22–33. Edward Elgar Publishing, 1996.

[22] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[23] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

[24] Michael A Nielsen and Isaac L Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, Cambridge, 2010.

[25] John Watrous. Quantum statistical zero-knowledge. *arXiv preprint arXiv:quant-ph/0202111*, 2002.

[26] Filippo Caruso, Vittorio Giovannetti, Cosmo Lupo, and Stefano Mancini. *Reviews of Modern Physics*, 86(4):1203, 2014.

[27] Reinhard F Werner. Quantum states with einstein-podolsky-rosen correlations admitting a hidden-variable model. *Physical Review A*, 40(8):4277, 1989.

[28] Christopher J Wood. Non-completely positive maps: properties and applications. *arXiv preprint arXiv:0911.3199*, 2009.

[29] John Watrous. *The Theory of Quantum Information*. Cambridge University Press, 2018.

[30] Anthony Chefles. Unambiguous discrimination between linearly independent quantum states. *Physics Letters A*, 239(6):339–347, 1998.

[31] Felix Bloch. Nuclear induction. *Physical review*, 70(7-8):460, 1946.

[32] Ettore Majorana. Atomi orientati in campo magnetico variabile. *Il Nuovo Cimento (1924-1942)*, 9(2):43–50, 1932.

[33] Yiqing Zhou, E Miles Stoudenmire, and Xavier Waintal. What limits the simulation of quantum computers? *Physical Review X*, 10(4):041038, 2020.

[34] P. W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, New York, NY, USA, 1994. IEEE.

[35] Sergey Bravyi, David Gosset, and Robert König. Quantum advantage with shallow circuits. *Science*, 362(6412):308–311, 2018.

[36] Federico Centrone, Niraj Kumar, Eleni Diamanti, and Iordanis Kerenidis. Experimental demonstration of quantum advantage for np verification with limited information. *Nature communications*, 12(1):1–11, 2021.

[37] Sau Lan Wu, Jay Chan, Wen Guan, Shaojun Sun, Alex Wang, Chen Zhou, Miron Livny, Federico Carminati, Alberto Di Meglio, Andy CY Li, et al. Application of quantum machine learning using the quantum variational classifier method to high energy physics analysis at the lhc on ibm quantum computer simulator and hardware with 10 qubits. *Journal of Physics G: Nuclear and Particle Physics*, 48(12):125003, 2021.

[38] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. Quantum algorithms for supervised and unsupervised machine learning. *arXiv preprint arXiv:1307.0411*, 2013.

[39] Peter Wittek. *Quantum machine learning: what quantum computing means to data mining*. Academic Press, 2014.

[40] Ville Bergholm, Josh Izaac, Maria Schuld, Christian Gogolin, M Sohaib Alam, Shahnawaz Ahmed, Juan Miguel Arrazola, Carsten Blank, Alain Delgado, Soran Jahangiri, et al. Pennylane: Automatic differentiation of hybrid quantum-classical computations. *arXiv preprint arXiv:1811.04968*, 2018.

[41] Michael Broughton, Guillaume Verdon, Trevor McCourt, Antonio J. Martinez, Jae Hyeon Yoo, Sergei V. Isakov, Philip Massey, Ramin Halavati, Murphy Yuezhen Niu, Alexander Zlokapa, Evan Peters, Owen Lockwood, Andrea Skolik, Sofiene Jerbi, Vedran Dunjko, Martin Leib, Michael Streif, David Von Dollen, Hongxiang Chen, Shuxiang Cao, Roeland Wiersema, Hsin-Yuan Huang, Jarrod R. McClean, Ryan Babbush, Sergio Boixo, Dave Bacon, Alan K. Ho, Hartmut Neven, and Masoud Mohseni. Tensorflow quantum: A software framework for quantum machine learning. *arXiv preprint arXiv:2003.02989*, 2020.

[42] Xiu-Zhe Luo, Jin-Guo Liu, Pan Zhang, and Lei Wang. Yao. jl: Extensible, efficient framework for quantum algorithm design. *Quantum*, 4:341, 2020.

[43] Román Orús. A practical introduction to tensor networks: Matrix product states and projected entangled pair states. *Annals of Physics*, 349:117–158, 2014.

[44] Edwin Stoudenmire and David J Schwab. Supervised learning with tensor networks. *Advances in Neural Information Processing Systems*, 29, 2016.

[45] Gael Sentís, Emilio Bagan, John Calsamiglia, and Ramon Muñoz-Tapia. Multicopy programmable discrimination of general qubit states. *Physical Review A*, 82(4):042312, 2010.

[46] Hendrik Poulsen Nautrup, Nicolas Delfosse, Vedran Dunjko, Hans J Briegel, and Nicolai Friis. Optimizing quantum error correction codes with reinforcement learning. *Quantum*, 3:215, 2019.

[47] Hsin-Yuan Huang, Richard Kueng, and John Preskill. Predicting many properties of a quantum system from very few measurements. *Nature Physics*, 16(10):1050–1057, 2020.

[48] Giuseppe Carleo and Matthias Troyer. Solving the quantum many-body problem with artificial neural networks. *Science*, 355(6325):602–606, 2017.

[49] Riccardo Porotti, Antoine Essig, Benjamin Huard, and Florian Marquardt. Deep reinforcement learning for quantum state preparation with weak nonlinear measurements. *Quantum*, 6:747, 2022.

[50] Juan Carrasquilla and Roger G. Melko. Machine learning phases of matter. *Nature Physics*, 13:431, 02 2017.

[51] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028*, 2014.

[52] Alex Pepper, Nora Tischler, and Geoff J Pryde. Experimental realization of a quantum autoencoder: The compression of qutrits via machine learning. *Physical review letters*, 122(6):060501, 2019.

[53] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J Love, Alán Aspuru-Guzik, and Jeremy L O'brien. A variational eigenvalue solver on a photonic quantum processor. *Nature Communications*, 5(1):1–7, 2014.

[54] Marcello Benedetti, Erika Lloyd, Stefan Sack, and Mattia Fiorentini. Parameterized quantum circuits as machine learning models. *Quantum Science and Technology*, 4(4):043001, 2019.

[55] M.A. Nielsen and I.L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010.

[56] M. Cerezo, Andrew Arrasmith, Ryan Babbush, Simon C Benjamin, Suguru Endo, Keisuke Fujii, Jarrod R McClean, Kosuke Mitarai, Xiao Yuan, Lukasz Cincio, and Patrick J. Coles. Variational quantum algorithms. *Nature Reviews Physics*, 3(1):625–644, 2021.

[57] Kishor Bharti, Alba Cervera-Lierta, Thi Ha Kyaw, Tobias Haug, Sumner Alperin-Lea, Abhinav Anand, Matthias Degroote, Hermanni Heimonen, Jakob S. Kottmann, Tim Menke, Wai-Keong Mok, Sukin Sim, Leong-Chuan Kwek, and Alán Aspuru-Guzik. Noisy intermediate-scale quantum algorithms. *Rev. Mod. Phys.*, 94:015004, Feb 2022.

[58] Zoë Holmes, Kunal Sharma, M. Cerezo, and Patrick J. Coles. Connecting ansatz expressibility to gradient magnitudes and barren plateaus. *PRX Quantum*, 3:010313, Jan 2022.

[59] Kouhei Nakaji and Naoki Yamamoto. Expressibility of the alternating layered ansatz for quantum computation. *Quantum*, 5:434, April 2021.

[60] Tobias Haug, Kishor Bharti, and M.S. Kim. Capacity and quantum geometry of parametrized quantum circuits. *PRX Quantum*, 2:040309, Oct 2021.

[61] Sukin Sim, Peter D. Johnson, and Alán Aspuru-Guzik. Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms. *Advanced Quantum Technologies*, 2(12):1900070, 2019.

[62] Atilim Gunes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *Journal of Marchine Learning Research*, 18:1–43, 2018.

[63] Andrea Mari, Thomas R. Bromley, and Nathan Killoran. Estimating the gradient and higher-order derivatives on quantum hardware. *Physical Review A*, 103:012405, 2021.

[64] Gavin E Crooks. Gradients of parameterized quantum gates using the parameter-shift rule and gate decomposition. *arXiv preprint arXiv:1905.13311*, 2019.

[65] James Stokes, Josh Izaac, Nathan Killoran, and Giuseppe Carleo. Quantum natural gradient. *Quantum*, 4:269, 2020.

[66] Jonathan Olson, Yudong Cao, Jonathan Romero, Peter Johnson, Pierre-Luc Dallaire-Demers, Nicolas Sawaya, Prineha Narang, Ian Kivlichan, Michael Wasielewski, and Alán Aspuru-Guzik. Quantum information and computation for chemistry. *arXiv preprint arXiv:1706.05413*, 2017.

[67] Paolo Braccia, Filippo Caruso, and Leonardo Banchi. How to enhance quantum generative adversarial learning of noisy information. *New Journal of Physics*, 23(5):053024, 2021.

[68] Shizuo Kakutani et al. A generalization of brouwer's fixed point theorem. *Duke mathematical journal*, 8(3):457–459, 1941.

[69] Seth Lloyd and Christian Weedbrook. Quantum generative adversarial learning. *Physical review letters*, 121(4):040502, 2018.

[70] Pierre-Luc Dallaire-Demers and Nathan Killoran. Quantum generative adversarial networks. *Physical Review A*, 98(1):012324, 2018.

[71] Carl W Helstrom. *Quantum detection and estimation theory*, volume 3. Academic press New York, 1976.

[72] Alexander S Holevo. Statistical problems in quantum physics. In *Proceedings of the second Japan-USSR Symposium on probability theory*, pages 104–119. Springer, 1973.

[73] John Watrous. *The theory of quantum information*. Cambridge University Press, Cambridge, 2018.

[74] Ian Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016.

[75] Marcello Benedetti, Edward Grant, Leonard Wossnig, and Simone Severini. Adversarial quantum circuit learning for pure state approximation. *New Journal of Physics*, 21(4):043023, 2019.

[76] Jonathan Romero and Alán Aspuru-Guzik. Variational quantum generators: Generative adversarial quantum machine learning for continuous distributions. *Advanced Quantum Technologies*, 4(1):2000003, 2021.

[77] Christa Zoufal, Aurélien Lucchi, and Stefan Woerner. Quantum generative adversarial networks for learning and loading random distributions. *npj Quantum Information*, 5(1):1–9, 2019.

[78] Ling Hu, Shu-Hao Wu, Weizhou Cai, Yuwei Ma, Xianghao Mu, Yuan Xu, Haiyan Wang, Yipu Song, Dong-Ling Deng, Chang-Ling Zou, et al. Quantum generative adversarial learning in a superconducting quantum circuit. *Science advances*, 5(1):eaav2761, 2019.

[79] Panayotis Mertikopoulos, Christos Papadimitriou, and Georgios Piliouras. Cycles in adversarial regularized learning. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2703–2717. SIAM, 2018.

[80] Emmanouil Vasileios Vlatakis-Gkaragkounis, Lampros Flokas, and Georgios Piliouras. Poincaré recurrence, cycles and spurious equilibria in gradient-descent-ascent for non-convex non-concave zero-sum games. 2019.

[81] Constantinos Daskalakis, Andrew Ilyas, Vasilis Syrgkanis, and Haoyang Zeng. Training gans with optimism. *arXiv preprint arXiv:1711.00141*, 2017.

[82] Guojun Zhang and Yaoliang Yu. Convergence of gradient methods on bilinear zero-sum games. In *International Conference on Learning Representations*, 2019.

[83] Vivek V Shende, Igor L Markov, and Stephen S Bullock. Minimal universal two-qubit controlled-not-based circuits. *Physical Review A*, 69(6):062321, 2004.

[84] Kosuke Mitarai, Makoto Negoro, Masahiro Kitagawa, and Keisuke Fujii. Quantum circuit learning. *Physical Review A*, 98(3):032309, 2018.

[85] Maria Schuld, Ville Bergholm, Christian Gogolin, Josh Izaac, and Nathan Killoran. Evaluating analytic gradients on quantum hardware. *Physical Review A*, 99(3):032331, 2019.

[86] Leonardo Banchi and Gavin E Crooks. Measuring analytic gradients of general quantum evolution with the stochastic parameter shift rule. *arXiv preprint arXiv:2005.10299*, 2020.

[87] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[88] Alexander Rakhlin and Karthik Sridharan. Online learning with predictable sequences. 2013.

[89] Leonardo Banchi, Jason Pereira, Seth Lloyd, and Stefano Pirandola. Convex optimization of programmable quantum computers. *npj Quantum Information*, 6(1):1–10, 2020.

[90] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. Quantum principal component analysis. *Nature Physics*, 10(9):631–633, 2014.

[91] Seth Lloyd, Maria Schuld, Aroosa Ijaz, Josh Izaac, and Nathan Killoran. Quantum embeddings for machine learning. *arXiv preprint arXiv:2001.03622*, 2020.

[92] Paolo Braccia, Leonardo Banchi, and Filippo Caruso. Quantum noise sensing by generating fake noise. *Physical Review Applied*, 17(2):024002, 2022.

[93] Armands Strikis, Dayue Qin, Yanzhu Chen, Simon C Benjamin, and Ying Li. Learning-based quantum error mitigation. *PRX Quantum*, 2(4):040330, 2021.

[94] Angus Lowe, Max Hunter Gordon, Piotr Czarnik, Andrew Arrasmith, Patrick J Coles, and Lukasz Cincio. Unified approach to data-driven quantum error mitigation. *Physical Review Research*, 3(3):033098, 2021.

[95] Amikam Levy, E Torrontegui, and Ronnie Kosloff. Action-noise-assisted quantum control. *Physical Review A*, 96(3):033417, 2017.

[96] Chenfeng Cao and Xin Wang. Noise-assisted quantum autoencoder. *Physical Review Applied*, 15:054012, 2021.

[97] Filippo Caruso, Susana F. Huelga, and Martin Plenio. Noise-enhanced classical and quantum capacities in communication networks. *Physical Review Letters*, 105:190501, 2010.

[98] Masoud Mohseni, Ali T Rezakhani, and Daniel A Lidar. Quantum-process tomography: Resource analysis of different strategies. *Physical Review A*, 77(3):032322, 2008.

[99] Giulio Chiribella, G Mauro D'Ariano, and Paolo Perinotti. Quantum circuit architecture. *Physical review letters*, 101(6):060401, 2008.

[100] M. Gregoratti and R.F. Werner. Quantum lost and found. *Journal of Modern Optics*, 50:915, 2003.

[101] Manoj K. Joshi, Andreas Elben, Benoit Vermersch, Tiff Brydges, Christine Maier, Peter Zoller, Rainer Blatt, and Christian F. Roos. Quantum information scrambling in a trapped-ion quantum simulator with tunable range interactions. *Physical Review Letters*, 124:240505, 2020.

[102] J.J. Wallman and J. Emerson. Noise tailoring for scalable quantum computation via randomized compiling. *Physical Review A*, 94:052325, 2016.

[103] Travis L. Scholten, Yi-Kai Liu, Kevin Young, and Robin Blume-Kohout. Classifying single-qubit noise using machine learning. *Eprint arXiv:1908.11762*, 2019.

[104] E. Knill. Quantum computing with realistically noisy devices. *Nature*, 434:39–44, 2005.

[105] J. Wallman, J. J. Emerson. Noise tailoring for scalable quantum computation via randomized compiling. *Physical Review A*, 94(5):052325, 2016.

[106] Matthew Ware, Guilhem Ribeill, Diego Ristè, Colm A. Ryan, Blake Johnson, and Marcus P. da Silva. Experimental pauli-frame randomization on a superconducting qubit. *Phys. Rev. A*, 103:042604, Apr 2021.

[107] J. J. Flammia, S. T. Wallman. Efficient estimation of pauli channels. *CM Transactions on Quantum Computing*, 1(1):1–32, 2020.

[108] Robin Harper, Steven T Flammia, and Joel J Wallman. Efficient learning of quantum noise. *Nature Physics*, 16(12):1184–1188, 2020.

[109] Chiara Macchiavello and G Massimo Palma. Entanglement-enhanced information transmission over a quantum channel with correlated noise. *Physical Review A*, 65(5):050301, 2002.

[110] Angel Rivas and Susana F Huelga. *Open quantum systems*, volume 10. Springer, Berlin, 2012.

[111] Felix A Pollock, César Rodríguez-Rosario, Thomas Frauenheim, Mauro Paternostro, and Kavan Modi. Non-markovian quantum processes: Complete framework and efficient characterization. *Physical Review A*, 97(1):012127, 2018.

[112] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. Advances in quantum metrology. *Nat. Photonics*, 5(4):222–229, 2011.

[113] Ludwig Zehnder. Ein neuer interferenzrefraktor, 1891.

[114] Ludwig Mach. Ueber einen interferenzrefraktor. *Zeitschrift für Instrumentenkunde*, 12(3):89, 1892.

[115] Majid Hassani, Chiara Macchiavello, and Lorenzo Maccone. Digital quantum estimation. *Physical review letters*, 119(20):200502, 2017.

[116] Quynh T Nguyen, Louis Schatzki, Paolo Braccia, Michael Ragone, Patrick J Coles, Frederic Sauvage, Martin Larocca, and M Cerezo. Theory for equivariant quantum neural networks. *arXiv preprint arXiv:2210.08566*, 2022.

[117] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[118] Michael M Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

[119] Erik J Bekkers, Maxime W Lafarge, Mitko Veta, Koen AJ Eppenhof, Josien PW Pluim, and Remco Duits. Roto-translation covariant convolutional networks for medical image analysis. In *International conference on medical image computing and computer-assisted intervention*, pages 440–448. Springer, 2018.

[120] Stefanie Jegelka. Theory of graph neural networks: Representation and learning. *arXiv preprint arXiv:2204.07697*, 2022.

[121] Risi Kondor, Zhen Lin, and Shubhendu Trivedi. Clebsch–gordan nets: a fully fourier space spherical convolutional neural network. *Advances in Neural Information Processing Systems*, 31, 2018.

[122] Taco S Cohen, Mario Geiger, Jonas Köhler, and Max Welling. Spherical cnns. *arXiv preprint arXiv:1801.10130*, 2018.

[123] Alexander Bogatskiy, Brandon Anderson, Jan Offermann, Marwah Roussi, David Miller, and Risi Kondor. Lorentz group equivariant neural network for particle physics. In *International Conference on Machine Learning*, pages 992–1002. PMLR, 2020.

[124] Marc Finzi, Samuel Stanton, Pavel Izmailov, and Andrew Gordon Wilson. Generalizing convolutional neural networks for equivariance to lie groups on arbitrary continuous data. In *International Conference on Machine Learning*, pages 3165–3176. PMLR, 2020.

[125] Bryn Elesedy and Sheheryar Zaidi. Provably strict generalisation benefit for equivariant models. In *International Conference on Machine Learning*, pages 2959–2969. PMLR, 2021.

[126] Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International conference on machine learning*, pages 2990–2999. PMLR, 2016.

[127] Risi Kondor and Shubhendu Trivedi. On the generalization of equivariance and convolution in neural networks to the action of compact groups. In *International Conference on Machine Learning*, pages 2747–2755. PMLR, 2018.

[128] Taco S Cohen, Mario Geiger, and Maurice Weiler. A general theory of equivariant cnns on homogeneous spaces. *Advances in neural information processing systems*, 32, 2019.

[129] Taco Cohen, Maurice Weiler, Berkay Kicanaoglu, and Max Welling. Gauge equivariant convolutional networks and the icosahedral cnn. In *International conference on Machine learning*, pages 1321–1330. PMLR, 2019.

[130] Taco S Cohen. *Equivariant convolutional networks*. PhD thesis, University of Amsterdam, 2021.

[131] Brandon Anderson, Truong Son Hy, and Risi Kondor. Cormorant: Covariant molecular neural networks. *Advances in neural information processing systems*, 32, 2019.

[132] Mario Geiger and Tess Smidt. e3nn: Euclidean neural networks. *arXiv preprint arXiv:2207.09453*, 2022.

[133] Nathaniel Thomas, Tess Smidt, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor field networks: Rotation- and translation-equivariant neural networks for 3d point clouds. *arXiv preprint arXiv:1802.08219*, 2018.

[134] Zhantao Chen, Nina Andrejevic, Tess Smidt, Zhiwei Ding, Qian Xu, Yen-Ting Chi, Quynh T Nguyen, Ahmet Alatas, Jing Kong, and Mingda Li. Direct prediction of phonon density of states with euclidean neural networks. *Advanced Science*, 8(12):2004214, 2021.

[135] Horace Pan and Risi Kondor. Permutation equivariant layers for higher order interactions. In Gustau Camps-Valls, Francisco J. R. Ruiz, and Isabel Valera, editors, *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pages 5987–6001. PMLR, 28–30 Mar 2022.

[136] Erik Henning Thiede, Truong Son Hy, and Risi Kondor. The general theory of permutation equivarant neural networks and higher order graph variational encoders. *arXiv preprint arXiv:2004.03990*, 2020.

[137] Taco S Cohen and Max Welling. Steerable cnns. *arXiv preprint arXiv:1612.08498*, 2016.

[138] Marc Finzi, Max Welling, and Andrew Gordon Wilson. A practical method for constructing equivariant multilayer perceptrons for arbitrary matrix groups. In *International Conference on Machine Learning*, pages 3318–3328. PMLR, 2021.

[139] Johannes Brandstetter, Rianne van den Berg, Max Welling, and Jayesh K Gupta. Clifford neural layers for pde modeling. *arXiv preprint arXiv:2209.04934*, 2022.

[140] Zhiyuan Li, Yi Zhang, and Sanjeev Arora. Why are convolutional nets more sample-efficient than fully-connected nets?, 2020.

[141] Akiyoshi Sannai, Masaaki Imaizumi, and Makoto Kawano. Improved generalization bounds of group invariant/equivariant deep networks via quotient feature spaces. In *Uncertainty in Artificial Intelligence*, pages 771–780. PMLR, 2021.

[142] Jure Sokolic, Raja Giryes, Guillermo Sapiro, and Miguel Rodrigues. Generalization error of invariant classifiers. In *Artificial Intelligence and Statistics*, pages 1094–1103. PMLR, 2017.

[143] Haggai Maron, Ethan Fetaya, Nimrod Segol, and Yaron Lipman. On the universality of invariant networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 4363–4371. PMLR, 09–15 Jun 2019.

[144] Dmitry Yarotsky. Universal approximations of invariant maps by neural networks. *Constructive Approximation*, 55(1):407–474, 2022.

[145] Nicolas Keriven and Gabriel Peyré. Universal invariant and equivariant graph neural networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[146] Siamak Ravanbakhsh. Universal equivariant multilayer perceptrons. In *International Conference on Machine Learning*, pages 7996–8006. PMLR, 2020.

[147] Soledad Villar, David W Hogg, Kate Storey-Fisher, Weichi Yao, and Ben Blum-Smith. Scalars are universal: Equivariant machine learning, structured like classical physics. *Advances in Neural Information Processing Systems*, 34:28848–28863, 2021.

[148] Ryszard Horodecki, Paweł Horodecki, Michał Horodecki, and Karol Horodecki. Quantum entanglement. *Reviews of modern physics*, 81(2):865, 2009.

[149] Szilárd Szalay. Multipartite entanglement measures. *Physical Review A*, 92(4):042329, oct 2015.

[150] Louis Schatzki, Andrew Arrasmith, Patrick J. Coles, and M. Cerezo. Entangled datasets for quantum machine learning. *arXiv preprint arXiv:2109.03400*, 2021.

[151] Martín Larocca, Frédéric Sauvage, Faris M. Sbahi, Guillaume Verdon, Patrick J. Coles, and M. Cerezo. Group-invariant quantum machine learning. *PRX Quantum*, 3:030341, Sep 2022.

[152] Johannes Jakob Meyer, Marian Mularski, Elies Gil-Fuster, Antonio Anna Mele, Francesco Arzani, Alissa Wilms, and Jens Eisert. Exploiting symmetry in variational quantum machine learning. *arXiv preprint arXiv:2205.06217*, 2022.

[153] Peter Mernyei, Konstantinos Meichanetzidis, and Ismail Ilkan Ceylan. Equivariant quantum graph circuits, 2021.

[154] Andrea Skolik, Michele Cattelan, Sheir Yarkoni, Thomas Bäck, and Vedran Dunjko. Equivariant quantum circuits for learning on weighted graphs. *arXiv preprint arXiv:2205.06109*, 2022.

[155] Han Zheng, Zimu Li, Junyu Liu, Sergii Strelchuk, and Risi Kondor. Speeding up learning quantum states through group equivariant convolutional quantum ansatze. *arXiv preprint arXiv:2112.07611*, 2021.

[156] Guillaume Verdon, Trevor McCourt, Enxhell Luzhnica, Vikash Singh, Stefan Leichenauer, and Jack Hidary. Quantum graph neural networks. *arXiv preprint arXiv:1909.12264*, 2019.

[157] Barry Simon. *Representations of Finite and Compact Groups*, volume 10 of *Graduate studies in Mathematics*. American Mathematical Society, 1996.

[158] William Fulton and Joe Harris. *Representation Theory: A First Course*. Springer, 1991.

[159] Roe Goodman and Nolan R Wallach. *Representations and invariants of the classical groups*. Cambridge University Press, 2000.

[160] Michael Ragone, Quynh T. Nguyen, Louis Schatzki, Paolo Braccia, Martin Larocca, Frederic Sauvage, Patrick J. Coles, and M. Cerezo. Representation theory for geometric quantum machine learning. *Manuscript in preparation*, 2022.

[161] Brian C Hall. *Lie groups, Lie algebras, and representations*. Springer, 2013.

[162] Vojtěch Havlíček, Antonio D Córcoles, Kristan Temme, Aram W Harrow, Abhinav Kandala, Jerry M Chow, and Jay M Gambetta. Supervised learning with quantum-enhanced feature spaces. *Nature*, 567(7747):209–212, 2019.

[163] Jonas Kübler, Simon Buchholz, and Bernhard Schölkopf. The inductive bias of quantum kernels. *Advances in Neural Information Processing Systems*, 34:12661–12673, 2021.

[164] Zoë Holmes, Kunal Sharma, M. Cerezo, and Patrick J Coles. Connecting ansatz expressibility to gradient magnitudes and barren plateaus. *PRX Quantum*, 3:010313, Jan 2022.

[165] Louis Schatzki, Martin Larocca, Frederic Sauvage, and M. Cerezo. 'theoretical guarantees for permutation-equivariant quantum neural networks. *Manuscript in preparation*, 2022.

[166] J. S. Otterbach, R. Manenti, N. Alidoust, A. Bestwick, M. Block, B. Bloom, S. Caldwell, N. Didier, E. Schuyler Fried, S. Hong, P. Karalekas, C. B. Osborn, A. Papageorge, E. C. Peterson, G. Prawiroatmodjo, N. Rubin, Colm A. Ryan, D. Scarabelli, M. Scheer, E. A. Sete, P. Sivarajah, Robert S. Smith, A. Staley, N. Tezak, W. J. Zeng, A. Hudson, Blake R. Johnson, M. Reagor, M. P. da Silva, and C. Rigetti. Unsupervised machine learning on a hybrid quantum computer. *arXiv preprint arXiv:1712.05771*, 2017.

[167] Iordanis Kerenidis, Jonas Landman, Alessandro Luongo, and Anupam Prakash. q-means: A quantum algorithm for unsupervised machine learning. *Advances in Neural Information Processing Systems*, 32, 2019.

[168] Marcello Benedetti, Delfina Garcia-Pintos, Oscar Perdomo, Vicente Leyton-Ortega, Yunseong Nam, and Alejandro Perdomo-Ortiz. A generative modeling approach for benchmarking and training shallow quantum circuits. *npj Quantum Information*, 5(1):1–9, 2019.

[169] Maria Kieferova, Ortiz Marrero Carlos, and Nathan Wiebe. Quantum generative training using r\'enyi divergences. *arXiv preprint arXiv:2106.09567*, 2021.

[170] Jonathan Romero and Alán Aspuru-Guzik. Variational quantum generators: Generative adversarial quantum machine learning for continuous distributions. *Advanced Quantum Technologies*, 4(1):2000003, 2021.

[171] V. Saggio, B. E. Asenbeck, A. Hamann, T. Strömberg, P. Schiansky, V. Dunjko, N. Friis, N. C. Harris, M. Hochberg, D. Englund, S. Wölk,

H. J. Briegel, and P. Walther. Experimental quantum speed-up in reinforcement learning agents. *Nature*, 591(7849):229–233, 2021.

[172] Andrea Skolik, Sofiene Jerbi, and Vedran Dunjko. Quantum agents in the gym: a variational quantum algorithm for deep q-learning. *arXiv preprint arXiv:2103.15084*, 2021.

[173] AS Holevo. Additivity conjecture and covariant channels. *International Journal of Quantum Information*, 3(01):41–47, 2005.

[174] William Gordon Ritter. Quantum channels and representation theory. *Journal of mathematical physics*, 46(8):082103, 2005.

[175] Matthew B Hastings. Superadditivity of communication capacity using entangled inputs. *Nature Physics*, 5(4):255–257, 2009.

[176] Mark M Wilde, Marco Tomamichel, and Mario Berta. Converse bounds for private communication over quantum channels. *IEEE Transactions on Information Theory*, 63(3):1792–1817, 2017.

[177] Robert Koenig and Stephanie Wehner. A strong converse for classical channel coding using entangled inputs. *Physical Review Letters*, 103(7):070504, 2009.

[178] Nilanjana Datta, Marco Tomamichel, and Mark M Wilde. On the second-order asymptotics for entanglement-assisted communication. *Quantum Information Processing*, 15(6):2569–2591, 2016.

[179] Felix Leditzky, Eneet Kaur, Nilanjana Datta, and Mark M Wilde. Approaches for approximate additivity of the holevo information of quantum channels. *Physical Review A*, 97(1):012332, 2018.

[180] Martina Gschwendtner, Andreas Bluhm, and Andreas Winter. Programmability of covariant quantum channels. *Quantum*, 5:488, jun 2021.

[181] Dmitry Grinko and Maris Ozols. Linear programming with unitary-equivariant constraints. *arXiv preprint arXiv:2207.05713*, 2022.

[182] Linghang Kong and Zi-Wen Liu. Near-optimal covariant quantum error-correcting codes from random unitaries with symmetries. *PRX Quantum*, 3(2):020314, 2022.

[183] Philippe Faist, Sepehr Nezami, Victor V Albert, Grant Salton, Fernando Pastawski, Patrick Hayden, and John Preskill. Continuous symmetries and approximate quantum error correction. *Physical Review X*, 10(4):041018, 2020.

[184] Sisi Zhou, Zi-Wen Liu, and Liang Jiang. New perspectives on covariant quantum error correction. *Quantum*, 5:521, 2021.

[185] Giacomo Mauro D'ariano. Extremal covariant quantum operations and positive operator valued measures. *Journal of mathematical physics*, 45(9):3620–3635, 2004.

[186] Masahito Hayashi. *Group representation for quantum theory*. Springer, 2017.

[187] Hari Krovi. An efficient high dimensional quantum schur transform. *Quantum*, 3:122, 2019.

[188] Aram W Harrow. Applications of coherent classical communication and the schur transform to quantum information theory. *arXiv preprint quant-ph/0512255*, 2005.

[189] Dave Bacon, Isaac L Chuang, and Aram W Harrow. Efficient quantum circuits for schur and clebsch-gordan transforms. *Physical review letters*, 97(17):170502, 2006.

[190] Iman Marvian. Restrictions on realizable unitary operations imposed by symmetry and locality. *Nature Physics*, 18(3):283–289, 2022.

[191] Iman Marvian, Hanqing Liu, and Austin Hulse. Rotationally-invariant circuits: Universality with the exchange interaction and two ancilla qubits. *arXiv preprint arXiv:2202.01963*, 2022.

[192] Austin Hulse, Hanqing Liu, and Iman Marvian. Qudit circuits with su (d) symmetry: Locality imposes additional conservation laws. *arXiv preprint arXiv:2105.12877*, 2021.

[193] Andrew M. Childs and Wim van Dam. Quantum algorithms for algebraic problems. *Reviews of Modern Physics*, 82(1):1–52, jan 2010.

[194] Muneerah Al Nuwairan. The extreme points of su (2)-irreducibly covariant channels. *International Journal of Mathematics*, 25(06):1450048, 2014.

[195] Marek Mozrzymas, Michał Studziński, and Nilanjana Datta. Structure of irreducibly covariant quantum channels for finite groups. *Journal of Mathematical Physics*, 58(5):052204, 2017.

[196] Laleh Memarzadeh and Barry C Sanders. Group-covariant extreme and quasiextreme channels. *Physical Review Research*, 4(3):033206, 2022.

[197] Robert Zeier and Thomas Schulte-Herbrüggen. Symmetry principles in quantum systems theory. *Journal of mathematical physics*, 52(11):113510, 2011.

[198] Christopher J. Wood, Jacob D. Biamonte, and David G. Cory. Tensor networks and graphical calculus for open quantum systems. *arXiv preprint arXiv:1111.6950*, 2011.

[199] Roger A. Horn and Charles R. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, 1991.

[200] Lloyd N Trefethen and David Bau III. *Numerical linear algebra*, volume 50. Siam, 1997.

[201] Nicholas J Higham. Gaussian elimination. *Wiley Interdisciplinary Reviews: Computational Statistics*, 3(3):230–238, 2011.

[202] Benoît Collins and Piotr Śniady. Integration with respect to the haar measure on unitary, orthogonal and symplectic group. *Communications in Mathematical Physics*, 264(3):773–795, 2006.

[203] Zbigniew Puchala and Jaroslaw Adam Miszczak. Symbolic integration with respect to the haar measure on the unitary groups. *Bulletin of the Polish Academy of Sciences Technical Sciences*, 65(1):21–27, 2017.

[204] Géza Tóth and Juan José García-Ripoll. Efficient algorithm for multiqudit twirling for ensemble quantum computation. *Physical Review A*, 75(4):042311, 2007.

[205] Frederic Sauvage, Martin Larocca, Patrick J. Coles, and M. Cerezo. Building spatial symmetries into parameterized quantum circuits for faster training. *arXiv preprint arXiv:2207.14413*, 2022.

[206] Kishor Bharti and Tobias Haug. Quantum-assisted simulator. *Physical Review A*, 104(4):042418, 2021.

[207] Frederic T Chong, Diana Franklin, and Margaret Martonosi. Programming languages and compiler design for realistic quantum hardware. *Nature*, 549(7671):180–187, 2017.

[208] Thomas Häner, Damian S Steiger, Krysta Svore, and Matthias Troyer. A software methodology for compiling quantum programs. *Quantum Science and Technology*, 3(2):020501, 2018.

[209] Kunal Sharma, Sumeet Khatri, M. Cerezo, and Patrick J Coles. Noise resilience of variational quantum compiling. *New Journal of Physics*, 22(4):043006, 2020.

[210] Mark M Wilde. *Quantum information theory*. Cambridge University Press, 2013.

[211] Neil W Ashcroft and N David Mermin. *Solid state physics*. Cengage Learning, 2022.

[212] Subir Sachdev. Quantum phase transitions. *Physics world*, 12(4):33, 1999.

[213] Xiao-Gang Wen. Colloquium: Zoo of quantum-topological phases of matter. *Reviews of Modern Physics*, 89(4):041004, 2017.

[214] Zhenghan Wang. *Topological quantum computation*. Number 112. American Mathematical Soc., 2010.

[215] HQ Lin. Exact diagonalization of quantum-spin models. *Physical Review B*, 42(10):6561, 1990.

[216] WMC Foulkes, Lubos Mitas, RJ Needs, and Guna Rajagopal. Quantum monte carlo simulations of solids. *Reviews of Modern Physics*, 73(1):33, 2001.

[217] Harry J Lipkin, N Meshkov, and AJ Glick. Validity of many-body approximation methods for a solvable model:(i). exact solutions and perturbation theory. *Nuclear Physics*, 62(2):188–198, 1965.

[218] Hai Tao Wang, Bo Li, and Sam Young Cho. Topological quantum phase transition in bond-alternating spin-1 2 heisenberg chains. *Physical Review B*, 87(5):054402, 2013.

[219] Todadri Senthil. Symmetry-protected topological phases of quantum matter. *Annu. Rev. Condens. Matter Phys.*, 6(1):299–324, 2015.

[220] S Paul and AK Ghosh. Ground state properties of the bond alternating spin-$\frac{1}{2}$ anisotropic heisenberg chain. *arXiv preprint arXiv:1706.07275*, 2017.

[221] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[222] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[223] Iris Cong, Soonwon Choi, and Mikhail D Lukin. Quantum convolutional neural networks. *Nature Physics*, 15(12):1273–1278, 2019.

[224] Ian MacCormack, Conor Delaney, Alexey Galda, Nidhi Aggarwal, and Prineha Narang. Branching quantum convolutional neural networks. *arXiv preprint arXiv:2012.14439*, 2020.

[225] Lukas Franken and Bogdan Georgiev. Explorations in quantum neural networks with intermediate measurements. In *Proceedings of ESANN*, 2020.

[226] Arthur Pesah, M. Cerezo, Samson Wang, Tyler Volkoff, Andrew T Sornborger, and Patrick J Coles. Absence of barren plateaus in quantum convolutional neural networks. *Physical Review X*, 11(4):041011, 2021.

[227] Roe Goodman and Nolan R Wallach. *Symmetry, representations, and invariants*, volume 255. Springer, 2009.

[228] Abhinav Anand, Jonathan Romero, Matthias Degroote, and Alán Aspuru-Guzik. Noise robustness and experimental demonstration of a quantum generative adversarial network for continuous distributions. *Advanced Quantum Technologies*, 4(5):2000069, 2021.

[229] Abhinav Kandala, Antonio Mezzacapo, Kristan Temme, Maika Takita, Markus Brink, Jerry M. Chow, and Jay M. Gambetta. Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. *Nature*, 549(7671):242–246, Sep 2017.

[230] Yulong Dong, Xiang Meng, Lin Lin, Robert Kosut, and K Birgitta Whaley. Robust control optimization for quantum approximate optimization algorithms. *IFAC-PapersOnLine*, 53(2):242–249, 2020.

[231] Panagiotis Kl Barkoutsos, Giacomo Nannicini, Anton Robert, Ivano Tavernelli, and Stefan Woerner. Improving variational quantum optimization using cvar. *Quantum*, 4:256, 2020.

[232] Laura Gentini, Alessandro Cuccoli, Stefano Pirandola, Paola Verrucchi, and Leonardo Banchi. Noise-resilient variational hybrid quantum-classical optimization. *Phys. Rev. A (in press), arXiv:1912.06744*, 2020.

[233] Joel J Wallman and Joseph Emerson. Noise tailoring for scalable quantum computation via randomized compiling. *Physical Review A*, 94(5):052325, 2016.

[234] Filippo Caruso, Vittorio Giovannetti, and G Massimo Palma. Teleportation-induced correlated quantum channels. *Physical Review Letters*, 104(2):020503, 2010.

[235] Stefano Pirandola, Riccardo Laurenza, Carlo Ottaviani, and Leonardo Banchi. Fundamental limits of repeaterless quantum communications. *Nature communications*, 8(1):1–15, 2017.

[236] Stephanie Wehner, David Elkouss, and Ronald Hanson. Quantum internet: A vision for the road ahead. *Science*, 362(6412):eaam9288, 2018.

[237] Rui Wang, Robin Walters, and Rose Yu. Approximately equivariant networks for imperfectly symmetric dynamics. *arXiv preprint arXiv:2201.11969*, 2022.

[238] Supanut Thanasilp, Samson Wang, Nhat A Nghiem, Patrick J. Coles, and M. Cerezo. Subtleties in the trainability of quantum machine learning models. *arXiv preprint arXiv:2110.14753*, 2021.

[239] Supanut Thanasilp, Samson Wang, M. Cerezo, and Zoë Holmes. Exponential concentration and untrainability in quantum kernel methods. *arXiv preprint arXiv:2208.11060*, 2022.

[240] John Von Neumann and Oskar Morgenstern. Theory of games and economic behavior. In *Theory of games and economic behavior*. Princeton university press, 2007.

[241] Drew Fudenberg and Jean Tirole. *Game theory*. MIT press, 1991.

[242] Noam D Elkies. On numbers and endgames: combinatorial game theory in chess endgames. *Games of No Chance*, 29:135–150, 1996.

[243] Darse Billings, Neil Burch, Aaron Davidson, Robert Holte, Jonathan Schaeffer, Terence Schauenberg, and Duane Szafron. Approximating game-theoretic optimal strategies for full-scale poker. In *IJCAI*, volume 3, page 661, 2003.

[244] Robert S Gibbons. *Game theory for applied economists*. Princeton University Press, 1992.

[245] Cuong T Do, Nguyen H Tran, Choongseon Hong, Charles A Kamhoua, Kevin A Kwiat, Erik Blasch, Shaolei Ren, Niki Pissinou, and Sundaraja Sitharama Iyengar. Game theory for cyber security and privacy. *ACM Computing Surveys (CSUR)*, 50(2):1–37, 2017.

[246] Nicola Lacey. The prisoners'dilemma. *Cambridge UK*, 2008.

[247] Dov Monderer and Lloyd S Shapley. Potential games. *Games and economic behavior*, 14(1):124–143, 1996.

[248] Leonardo Banchi and Gavin E Crooks. Measuring analytic gradients of general quantum evolution with the stochastic parameter shift rule. *Quantum*, 5:386, 2021.

[249] Sukin Sim, Peter D Johnson, and Alán Aspuru-Guzik. Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms. *Advanced Quantum Technologies*, 2(12):1900070, 2019.

[250] Grecia Castelazo, Quynh T Nguyen, Giacomo De Palma, Dirk Englund, Seth Lloyd, and Bobak T Kiani. Quantum algorithms for group convolution, cross-correlation, and equivariant transformations. *arXiv preprint arXiv:2109.11330*, 2021.

[251] András Gilyén, Yuan Su, Guang Hao Low, and Nathan Wiebe. Quantum singular value transformation and beyond: exponential improvements for quantum matrix arithmetics. *arXiv preprint arXiv:1806.01838*, 2018.

[252] Zoë Holmes, Nolan Coble, Andrew T Sornborger, and Yigit Subasi. On nonlinear transformations in quantum computation. *arXiv preprint arXiv:2112.12307*, 2021.

[253] Wieb Bosma, John Cannon, and Catherine Playoust. The Magma algebra system. I. The user language. *J. Symbolic Comput.*, 24(3-4):235–265, 1997. Computational algebra and number theory (London, 1993).

[254] Marc AA Van Leeuwen, Arjeh Marcel Cohen, and Bert Lisser. Lie: A package for lie group computations, 1992.