Quantum machine learning on near term hardware with unstructured and graph structured data

Slimane Thabet

Sorbonne Université - EDITE de Paris
Laboratoire d'Informatique de Sorbonne Université (LIP6)

Quantum machine learning on near term hardware with unstructured and graph structured data

By Slimane Thabet

PhD Thesis in Computer Science

PhD defence held publicly the 12th May 2025 with the following thesis committee:

- Magniez Frédéric, Directeur de recherche, CNRS, IRIF, Université de Paris, *Président.*
- Dunjko Vedran, Professor, Leiden Universiteit, *Rapporteur.*
- Holmes Zoë, Assistant professor, EPFL, *Rapporteuse.*
- Daley Andrew, Professor, University of Oxford, *Examinateur.*
- Eisert Jens, Professor, Freie Universität Berlin, *Examinateur.*
- Kashefi Elham, Directrice de Recherche au CNRS, LIP6, Sorbonne Université and Professor at University of Edinburgh, United Kingdom, *Directrice de thèse.*
- Bredariol Grilo Alex, chercheur au CNRS, LIP6, Sorbonne Université *Co-directeur de thèse.*
- Henriet Loïc, CEO, Pasqal, *Co-directeur de thèse.*
- Henry Louis-Paul, VP of Quantum Applications, Pasqal, *Invité.*

Résumé

L'apprentissage automatique a permis de résoudre de nombreux problèmes du monde réel que d'autres méthodes informatiques traditionnelles avaient du mal à résoudre, ou qu'elles pouvaient résoudre de manière plus coûteuse. L'informatique quantique est un paradigme de calcul utilisant les états quantiques de la matière qui permet une grande vitesse de calcul pour certains problèmes. Les progrès récents dans le développement du matériel quantique ont encouragé la recherche d'applications concrètes des ordinateurs quantiques. Il est donc devenu naturel de chercher des moyens d'appliquer les ordinateurs quantiques à l'apprentissage automatique. Cette thèse est une contribution à cet objectif.

La première partie vise à comprendre les capacités générales des circuits quantiques variationnels pour les tâches d'apprentissage automatique à partir de données vectorielles, et d'éclaircir les conditions nécessaires pour obtenir un avantage quantique. Les circuits variationnels sont une famille d'algorithmes quantiques où les opérations sont paramétrées et où l'on cherche les paramètre qui minimisent une fonction de coût, à la manière des réseaux de neurones. Ce sont en réalité des modèles linéaires dans un espace à grande dimension. Je montre que bien que les circuits soient coûteux à évaluer, on peut parfois construire des approximations classiques en utilisant la technique de "random features regression". Si cette approximation est possible, l'avantage quantique est limité. Je souligne également le fait que l'apprentissage d'un modèle classique dans le même espace converge vers une solution appelée estimateur MNLS (Minimum Norm Least Square), mais que la dynamique d'apprentissage des circuits quantiques ne conduira pas nécessairement à la même solution. Cette séparation est la source de l'avantage quantique, je montre qu'il suffit que le vecteur des coefficients du modèle quantique ait une grande norme, et je donne des exemples concrets.

La deuxième partie explore l'utilisation d'ordinateurs quantiques pour effectuer des tâches d'apprentissage automatique sur des graphes. L'apprentissage automatique sur des graphes possède beaucoup d'applications, et les algorithmes pour des données vectorielles ne sont pas directement applicables. Dans cette partie, j'ai cherché à développer des algorithmes quantiques adaptés à la structure de graphes des données. L'idée principale est d'encoder le graphe dans un hamiltonien qui a une topologie similaire. On prépare ensuite un état quantique en faisant évoluer cet hamiltonien, et les mesures sont incorporées dans un algorithme classique d'apprentissage automatique. Cette approche est particulièrement adaptée aux processeurs quantiques à atomes neutres. Avec de telles plateformes, on peut en effet facilement créer un système quantique avec la connectivité souhaitée, et la géométrie peut être modifiée à chaque exécution. J'ai développé une diverse famille d'algorithmes, inspirés par les noyaux, les réseaux neurones et les transformers, avec l'intention de pouvoir les implémenter sur les machines actuelles. J'ai réalisé des expériences numériques sur des ensembles de données à grande échelle et décrit les résultats d'implémentation expérimentale sur la machine de Pasqal.

Abstract

Machine learning enabled the resolution of many real world problems that other traditional computational method struggled to solve, or could solve in more expensive ways. Quantum computing is a paradigm of computation using the quantum states of matter that enables a large computational speed up on some problems. The recent progress in the development of quantum hardware encouraged the research of concrete applications of quantum computers. It then became natural to look for ways in which quantum computers can be applied for machine learning. This thesis is a contribution towards this goal.

The first part aims at understanding the general capabilities of variational quantum circuits (VQC) for machine learning tasks given unstructured vector data inputs, and have a clearer idea on the necessary conditions in order to expect a quantum advantage. VQCs are a family of quantum algorithms where one finds gates parameters that minimizes a cost function, in the same way as neural networks. They are effectively linear models in a high dimensional feature space. I show that although VQCs are costly to evaluate, one can sometimes construct cheap classical approximators called classical surrogate using the technique of random features regression. If this approximation is possible, the quantum advantage is limited. I also highlight the fact that learning a classical model on the same feature map will lead to a solution called the Minimum Norm Least Square (MNLS) estimator, but the training dynamics of the quantum circuits will not necessarily lead to the same solution. This separation is the source of quantum advantage, I show that it is sufficient that the weight vector of quantum models has a large norm, and I give concrete examples.

The second part explores the use of quantum computers to perform machine learning tasks on graph structured data. Machine learning on graph data encompasses many real world applications, and algorithms for vector data cannot be directly applied. I aimed in this part to develop quantum algorithms adapted to the graph structure of the data. The main idea is to encode the graph into a Hamiltonian that has the same topology. One then prepares a quantum state by evolving this Hamiltonian, and the measurements are incorporated in a classical machine learning algorithm. This approach is especially suited to neutral atoms quantum computers. With such platforms, one can indeed easily create a quantum system with the desired connectivity, and the geometry can be changed at each run. I developed a large family of algorithms, inspired by kernels, graphs neural networks, and transformers with the intention to be ran on current hardware. I performed numerical experiments on large scale datasets, and described the results of an experimental implementation on the hardware of Pasqal.

Dedication and acknowledgements

As would have said my supervisor Elham speaking of ambitious projects: "It takes a village to raise a child", so I would like to thank all the people that are part of the village that made the realization of this thesis possible.

First and foremost, I would like to thank my supervisor Elham Kashefi for the dedication, for providing me support and encouragement both emotionally and scientifically in the difficult moments, managing sometimes my crazy demands (like going by train from Paris to Edinburgh).

I want to thank my advisor Alex Bredariol Grilo for giving me rigor, and help me organize with too many project.

I thank Loïc Henriet for being a role model, for showing me how one can be an great scientist and an amazing manager and CEO.

I thank Louis-Paul Henry for the lengthy passionate scientific discussions that made me more curious about statistical physics.

I thank all the people of the QGML team that make amazing people to work with, Mehdi, Igor, Sachin, Matthias, Lorenzo, Mathieu, and Shaheen who is courageous to manage everyone.

I thank the team of Pasqal Canada Raph, Hossein, Victor for developping an amazing ecosystem for quantum computing, and welcoming me to be a part of it.

I thank my co PhDs Constantin, Lucas, Elie for all the moments shared, serious and less serious in this journey.

I thank all the people at Pasqal that provided amazing company and rich ideas, Romain, Mourad, Chayma, Louis, François-Marie, Yassine, Wesley, Mario, Andrea, Casper, Anton, Pablo, Anne-Claire, Emmanuelle, Audrey, Henrique, Adrien, Lucas, Georges, Christophe.

I thank all the professors at LIP6 Fred, Marco, Eleni, Damian, Jessica for enabling such a flousrishing environment, and Gizem that does a wonderful job at managing cumbersome tasks for everyone. Such effort was amazing in organizing lab retreats.

I thank all my comrades at LIP6 Léo, Eliott, Hela, Mario, Snehal, Bo, Dominik, Cica, Paul, Léo, Armando, Uta, Naomi, Michael, Verena, Yoann, Valentina, Manon with whom I worked, and who shared countless lunches, discussions about food and politics, activities like karaoké and board games.

I especially thank Jonas for guiding me since the very begining in my research journey in quantum computing.

I thank the Quantum Software Lap in Edinburgh for frequently hosting me, and sharing scottish experience as well as scientific discussions Mina, Chirag, James, Jonathan, Ross, Ramin, Raul, Marie, Ioannis.

I thank Jean-François Hullo and the team of EDF UK Digital Innovation Nidhal, Jordan, Jack, Matt, Léo, Julien for having me as an intern in quantum computing when the field was getting barely out of academia.

I want to thank Guillaume Rabusseau for welcoming me in his group in Montréal, and providing me enriching perspectives in machine learning research. I also thank the people there for their kindness, Jun, Farzaneh, Mikael, Maude, Beheshteh.

I thank Francis Bach and Anthony Leverrier for advising me in my comité de suivi and bearing with my sometimes last minute requests.

I want to thank the examinors Zoë Holmes and Vedran Dunjko for the feedback on this manuscript and for the enriching discussions during the defense, as well as the members of the jury Frédéric Magniez, Jens Eisert and Andrew Daley.

Je remercie mes amis qui sont là depuis le début de mes études dont cette thèse est une étape, Tanguy, Gildas, Côme, Lionel, Matthieu, Hector, Mailys, Mélanie, Eloi, Xavier, Pierre-Louis, Daniel, Anthony, Samuel, Guillaume, Sylvain, Léon, Alexandre, Guillaume, Philippine, Fanny, José-Louis, Luis.

Finalement, je remercie mes parents, mon frère et ma soeur pour m'avoir donné l'envie de ne pas limiter ma curiosité intellectuelle et d'avoir été avec moi à chaque étape de la vie.

Enfin Yuchen tu apportes tellement à ma vie que cette thèse est en partie la tienne. Je nous souhaite encore plein d'aventures et de projets à partager ainsi.

Table of Contents

Introduction

Machine learning is the science of creating programs that perform tasks without a handcrafted set of rules to execute. Instead, the program adapts the parameters of a model from a set of data that would be called *training data* (BN06). An example of task for which machine learning is especially relevant is image classification. It is very easy for humans to distinguish images of cats and dogs, but it is much harder to create an algorithm to do it with deterministic logic. The pioneering work of (KSH12) that built on (Ros58; lec89; LB$^+$) showed that the most effective way to perform this task is to fit a parameterized function called *neural network* over a dataset of images, using a general set of methods called *deep learning* (LBH15). Other computational tasks that have been solved most effectively by machine learning include playing games (SSS$^+$17), compute the 3D structure of proteins (JEP$^+$21), or generating text (AI23; ADL$^+$22).

Quantum computing and quantum information processing is the use of the quantum states of matter to encode information and to perform computations (NC11). Classical computation uses classical bits and logical gates, whereas quantum computation uses quantum states, quantum operators, and measurements as building blocks of computation. Several technologies enable the building of quantum computers, including superconducting qubits (dev13), trapped ions (cir95), neutral atoms (HBS$^+$20), or photonics (O'b07).

This paradigm of information processing opened up a very rich area of research and several questions in complexity theory. Quantum computing can be indeed much more powerful than classical computing, ie some tasks can be solved in polynomial time with a quantum computer, but there is no known classical algorithms that can do the same. The most famous example of such a quantum algorithm is Shor's algorithm for prime factoring and period finding (Sho94). The best known classical algorithm to factor prime numbers takes an exponential time. Such a problem is the basis of modern cryptography, which makes the stakes even higher.

It then became natural to look for ways in which quantum computing can be used for machine learning, which gave birth to the field of *quantum machine learning* (QML) (BWP$^+$17; CVH$^+$22). The expression *quantum machine learning* is employed to designate a large variety of protocols in the literature. Such variety is due to the fact that both the data and the processing algorithm can be either classical or quantum. (SP18) proposed to create four categories in the field.

- **CC**: classical data, and classical algorithm. It refers to usual machine learning.

- **CQ**: classical data, and quantum algorithm. The goal is to create quantum algorithms to solve machine learning tasks that people used to tackle with classical machine learning algorithms.
- **QC**: quantum data, classical algorithm. It refers to the use of classical machine learning techniques to process data from quantum experiments (DAR+22).
- **QQ**: Quantum algorithms, and quantum data. One can imagine a quantum state decoded by a quantum algorithm. A classic application is the phase detection of a quantum state (CCL19).

This thesis will be exclusively focused on the **CQ** category. I will also only work on *supervised learning* where the goal is to assign a target value to a data (e.g classifying images, predicting prices). Other learning paradigms exist like unsupervised learning, reinforcement learning, or generative learning, which have also been studied in the context of quantum computing. I am interested in whether quantum computers can perform some machine learning tasks on classical data better than classical algorithms. The term "better" can be in principle be evaluated by several metrics. I will consider that a quantum algorithm is better than a classical one if it can get a better accuracy in the prediction result irrespective of the computational resources employed, or if it can match the accuracy with a lower cost.

The latter objective was the goal of an important line of work in quantum machine learning using *quantum linear algebra*. Quantum linear algebra is the use of quantum algorithms for speeding up linear algebra tasks. It started with the HHL algorithm (HHL09) for matrix inversion and linear system solving. This family of algorithms have led to the creation of quantum machine learning algorithms offering an exponential speed up over their classical counterparts known at that time. Examples of such algorithms include quantum versions of recommender systems (KP16), k-means clustering (KLLP19), or convolution neural networks (KLP19). It was later proven that the speed up is only polynomial (Tan19; Tan23), but the quantum algorithms are still valuable.

The quantum linear algebra machine learning algorithms require a fault tolerant quantum computer, which is very far from the performances of current hardware (see (ZKH+25) table 1 for a summary of state of the art error rate of current platforms). The community then proposed to use *variational quantum circuits* (VQC), which use parameterized quantum gates and tune the parameters to minimize a loss function (CAB+21; SBSW20). Contrary to previous algorithms, they can be implemented on available hardware.

VQCs are conceptually similar to deep learning and neural networks, where the goal is to fit a parameterized function with little hypothesis on the structure of the data. They are used as heuristics and there is little theoretical guarantees on how they could perform better than classical algorithms. As in modern machine learning practice, performance on real world datasets is the most important.

At the beginning of this work, it was known that VQCs suffered issues in trainability. The main problem was that for a widely used family of circuits, gradients were too small to be efficiently estimated via sampling, this phenomenon is called *barren plateaus* (MBS+18). There was also significant progress made in the characterization of the function that could be expressed with these circuits (SSM21a).

In this thesis, I wished to understand the capabilities of variational quantum circuits (VQC) for machine learning tasks, and have a clearer idea on the necessary conditions in order to expect a quantum advantage. I also wished to design algorithms that can exploit the graph structure of data. Graph machine learning problems can indeed be associated to a lot of real world tasks, and have been little investigated by the quantum machine learning community.

In Part I, I will explore the theoretical properties of VQCs with unstructured vector data input, and general architectures. I focus on the question of determining the conditions in which VQCs can provide a quantum advantage.

In Chapter 1, I introduce the concepts of variational quantum circuits (VQC), describe their use in machine learning, and give a brief overview of the existing literature.

In Chapter 2, I investigate first some limitations of these quantum models. I indeed show that under certain conditions, one can classically approximate a VQC given only the description of its architecture, with a technique from the classical machine learning community called Random Fourier features. I provide general theoretical bounds for classically approximating models built from exponentially large quantum feature space by sampling a few frequencies to build an equivalent low dimensional kernel, and I show experimentally that this approximation is efficient for several encoding strategies.

In Chapter 3, I refine the analysis made in the previous chapter. I compare the inductive bias between classical linear regression and quantum models. I characterize the separation between quantum and classical models by their respective weight vector. I show that a sufficient condition for a quantum model to avoid dequantization by its classical surrogate is to have a large weight vector norm. I suggest that this can only happen with a high dimensional feature map. Through the study of some common quantum architectures and encoding schemes, I obtain bounds on the norms of the quantum weight vector and the corresponding classical weight vector. It is possible to find instances allowing for a separation, but in these cases concentration issues become another concern. I finally prove that there exists linear models with large weight vector norm and without concentration, potentially achievable by a quantum circuit.

When I started this thesis work, there were very few work on machine learning for graph structured data. The study of machine learning on graphs and networks is an increasingly popular topic of research (Ham20) in the classical machine learning community.

In Part II, I propose methods to perform machine learning tasks on graph structured data, especially adapted to neutral atoms quantum computers. The main idea is to encode the

topology of the input graph in the Hamiltonian of a quantum system, and to make the quantum state evolve. The evolution produces measurement samples that retain key features of the data. These features are in general out of reach for a classical computer, and I prove that some of these quantum features are theoretically more expressive for certain graphs than the commonly used relative random walk probabilities or laplacian eigenvectors.

In Chapter 4, I introduce important concepts in machine learning for graph data. I define graph kernels, graph neural networks, and graph transformers. I give examples of the most recent algorithms and architectures for graph learning. I also give a brief overview on the capabilities of neutral atom quantum computers.

In Chapter 5, I detail the methods to perform graph machine learning with a quantum computer following a general blueprint. I introduce the algorithms that I developed during my thesis, and detail their inner working. I also describe the theoretical properties that could be established.

In Chapter 6, I describe all the experiments and implementations that have been performed with the algorithms. I show numerically that the performance of state-of-the-art models can be improved on standard benchmarks and large-scale datasets by computing tractable versions of quantum features. I also describe the results of the implementation of a quantum kernel algorithm on the neutral atom hardware of Pasqal. I first show that interactions in the quantum system can be used to distinguish non isomorphic graphs that are locally equivalent. I then describe the implementation of a toxicity screening experiment, consisting of a binary classification protocol on a biochemistry data set comprising 286 molecules of sizes ranging from 2 to 32 nodes, and obtain results which are comparable to the implementation of the best classical kernels on the same data set. Using techniques to compare the geometry of the feature spaces associated with kernel methods, I then show evidence that the quantum feature map perceives data in an original way, which is hard to replicate using classical graph kernels.

This thesis is based on the following published papers and preprints:

- (ADL+23) Boris Albrecht, Constantin Dalyac, Lucas Leclerc, Luis Ortiz-Gutiérrez, Slimane Thabet, Mauro D'Arcangelo, Julia RK Cline, Vincent E Elfving, Lucas Lassablière, Henrique Silvério, Bruno Ximenez, Louis-Paul Henry, Adrien Signoles, Loïc Henriet. (2022) *Quantum feature maps for graph machine learning on a neutral atom quantum processor.* Physical Review A, 107(4), 042615.

- (LTD+22) Jonas Landman, Slimane Thabet, Constantin Dalyac, Hela Mhiri, Elham Kashefi (2022). *Classically Approximating Variational Quantum Machine Learning with Random Fourier Features.* International Conference on Learning Representations 2023

- (TFH22) Slimane Thabet, Romain Fouilland, Loic Henriet (2022). *Extending Graph Transformers with Quantum Computed Correlations arXiv:2210.10610*

- (TDS+) Slimane Thabet, Mehdi Djellabi, Igor Sokolov, Sachin Kasture, Louis-Paul Henry, Loïc Henriet (2024). *Quantum Positional Encodings for Graph Neural Networks.* International Conference on Machine Learning 2024

- (TML24) Slimane Thabet, Léo Monbroussou, Eliott Z Mamon, Jonas Landman (2024). *When Quantum and Classical Models Disagree: Learning Beyond Minimum Norm Least Square arXiv:2411.04940*

I also had the opportunity to participate in the following preprint under rewiew that is not detailed in this thesis:

- (MMHG+24) Hela Mhiri, Léo Monbroussou, Mario Herrero-Gonzalez, Slimane Thabet, Elham Kashefi, Jonas Landman (2024). *Constrained and vanishing expressivity of quantum fourier models,arXiv:2403.09417*

Part I

Quantum advantages of variational quantum circuits on unstructured data

Chapter 1

Introduction to variational quantum circuits

Variational Quantum Circuits (VQC), also named Variational Quantum Algorithms (VQA), Parameterized Quantum Circuits (PQC), or Quantum Neural Networks (QNN) are a family of quantum algorithms that emerged in the recent years. (CAB+21) proposes a review of VQCs summarizing the different techniques, their potential and challenges.

This chapter will introduce the key technical elements that will be used in this part. In Section 1.1, I give a brief overview of the general structure of VQCs, and the original use cases. In Section 1.2, I give some details about linear regression and kernel ridge regression, two key machine learning algorithms that will constitute a basis for this part. In Section 1.3 and Section 1.4, I detail how VQCs are used in the context of machine learning and I introduce some key concepts that will be used throughout this part.

1.1 Overview

The general idea is to create a circuit $U(\theta)$ parameterized by the classical parameters θ, and a cost function $C(\theta)$ depending on the circuit. Such a circuit $U(\theta)$ is sometimes called an *ansatz*. The goal is therefore to find the parameters θ^* that minimize this cost function. The optimal value of the cost function, or the quantum state generated by the optimal parameters would constitute a solution to the problem one wishes to solve. Each problem will need a different circuit and cost function, therefore the construction of the parameterized circuit and the cost function is the core of a VQC.

A general formulation of a parameterized unitary $U(\theta)$ can be expressed as

$$U(\theta) = \sum_{k=1}^{K} \exp(-iH_k\theta_k)W_k \tag{1.1}$$

where $\theta = (\theta_1, \dots \theta_K)$ is the vector of parameters, H_ks are hermitian matrices, and W_ks are non parameterized unitaries. A popular way to create cost functions is to take the expectation of an observable O on the circuit, and to express the cost function as $C(\theta) = \mathrm{Tr}(U^\dagger(\theta)OU(\theta)|0\rangle\langle 0|)$.

The first example of such known algorithms is the variational quantum eigensolver (VQE) (PMS+14). The problem that motivated the method is to find the ground state energy of a given hamiltonian H. The authors of (PMS+14) proposed to create a parameterized quantum state $|\psi(\theta)\rangle$, and to minimize $\langle\psi(\theta)|H|\psi(\theta)\rangle$ with an optimization algorithm such as the Nelder Mead algorithm (NM65). The minimum of this function would be a heuristic for the ground state energy of H.

Another famous example of a variational algorithm is the Quantum Approximate Optimization Algorithm (QAOA) (FGG14). The objective of this algorithm is to solve combinatorial optimization problems. Combinatorial optimization problems can be formulated as finding the ground state of a hamiltonian (Luc14). Let H_C be the cost hamiltonian, depending on the problem, and H_M be a mixing hamiltonian. The QAOA algorithm introduces the parameterized state

$$\prod_{k=1}^{p} \exp(-i\gamma_k H_C)\exp(-i\beta_k H_M) \tag{1.2}$$

for an integer $p \geq 1$, with parameters $(\beta_1, \dots, \beta_p)$ and $(\gamma_1, \dots, \gamma_p)$. The algorithm aims to mimic the Troterrization (Suz76) of an adiabatic evolution (Kat50). If one prepares the ground state of H_M, and makes the system evolve during a time T following the time dependent hamiltonian

$$\frac{t}{T}H_C + (1 - \frac{t}{T})H_M \tag{1.3}$$

the final state obtained will be the ground state of H_C provided that T is big enough.

An important issue in the use of parameterized circuits is the barren plateaus phenomenon originally discovered by (MBS+18). It means that the gradients of the loss functions becomes too small to be reliably estimated with measurements. More quantitatively, a circuit has barren plateaus if the variance of the cost function over the distribution of the initialization of the parameters θ is exponentially small, ie $\mathbb{E}_\theta[C(\theta)] = 0$ and $\mathbb{V}_\theta[C(\theta)] = \mathcal{O}(b^{-n})$ where n is the number of qubits and $b > 0$. It is a very dynamic area of research in the community, (LTW+24) made an extensive review of the topic. This topic is mainly out of the scope of this thesis, I will only refer to it in Section 3.4.

1.2 Linear regression and kernel ridge regression

We present in this section the Linear Ridge Regression (LRR) and Kernel Ridge Regression (KRR) problem ((BN06)). The problem of regression is to predict continuous label values from

feature vectors. We are given a dataset $\{(x_i, y_i), i \in [\![1, M]\!],\ x_i \in \mathbb{R}^d,\ y_i \in \mathbb{R}\}$, and to each data point x an associated feature vector $\phi(x) \in \mathbb{R}^p$. The goal of LRR is to construct a parameterized model f such that $f(x) = y$. The model is parameterized by a weight vector $\mathbf{w}$ of size p such that $f(x; w) = \mathbf{w}^T \phi(x)$. Training the model consists of finding the vector $\mathbf{w}^*$ that minimizes the loss function

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \frac{1}{M} \sum_{i=1}^{M} |\mathbf{w}^T \phi(x_i) - y_i|^2 + \lambda ||\mathbf{w}||^2 \tag{1.4}$$

$$= \arg\min_{\mathbf{w}} \frac{1}{M} ||\mathbf{\Phi w} - \boldsymbol{y}||^2 + \lambda ||\mathbf{w}||^2 \tag{1.5}$$

$$\tag{1.6}$$

where $\mathbf{\Phi}$ is a matrix of size $M \times p$ with each row i corresponds to $\phi(x_i)^T$ and $\boldsymbol{y}$ is the vector of all the labels y_i. The first term of the loss is the Mean Square Error (MSE) and corresponds to the difference between the prediction and the ground truth. The second term is the ridge regularization, and prevents the weights from exploding. The magnitude of the regularization is controlled by the hyperparameter $\lambda > 0$.

When $p < M$, an analytic solution to this problem is given by $\mathbf{w}^* = (\mathbf{\Phi}^T \mathbf{\Phi} + M\lambda I_p)^{-1} \mathbf{\Phi}^T \boldsymbol{y}$. If $p > M$, the solution is not unique and the matrix $\mathbf{\Phi}^T \mathbf{\Phi}$ will not be invertible, one can find a solution by performing a gradient descent on the loss function $\mathcal{L}(\mathbf{w}) = \frac{1}{M} \sum_{i=1}^{M} |\mathbf{w}^T \phi(x_i) - y_i|^2 + \lambda ||\mathbf{w}||^2$.

The dual formulation of this problem is given by expressing $\mathbf{w}$ as a linear combination of the data points $\mathbf{w} = \mathbf{\Phi}^T \boldsymbol{\alpha}$. The minimization on $\mathbf{w}$ become a minimization on $\boldsymbol{\alpha}$ and can be expressed as

$$\boldsymbol{\alpha}^* = \arg\min_{\boldsymbol{\alpha}} \frac{1}{M} ||\mathbf{\Phi \Phi}^T \boldsymbol{\alpha} - \boldsymbol{y}||^2 + \lambda \boldsymbol{\alpha}^T \mathbf{\Phi \Phi}^T \boldsymbol{\alpha} \tag{1.7}$$

$$\tag{1.8}$$

The solution of this problem is $\boldsymbol{\alpha} = (\mathbf{\Phi \Phi}^T + M\lambda I_M)^{-1} \boldsymbol{y}$.

Note that the dual solution only depends on the matrix of scalar products between feature vectors $\mathbf{\Phi \Phi}^T$. One can then replace this matrix by a kernel matrix K and the obtained model is a Kernel Ridge Regression.

1.3 Variational quantum circuits as machine learning models

We consider a standard learning task where a parameterized function f, named *model*, must be optimized to map vector data points to their target values. The data used for the training is made of M points $x = (x_1, \ldots, x_d)$ in $\mathcal{X} = \mathbb{R}^d$ along with their target values y in $\mathcal{Y} = \mathbb{R}$. We define a *quantum model* as the family of parametrized functions $f : (\mathcal{X}, \Theta) \longrightarrow \mathcal{Y}$, such that

$$f(x; \theta) = \langle 0 | U(x; \theta)^\dagger O U(x; \theta) | 0 \rangle \tag{1.9}$$

where $U(x;\theta)$ is a unitary that represents the parametrized quantum circuits, θ represents the trainable parameters from a space Θ, and O is an observable. We can always describe the parametrized quantum circuit as a series of two types of gates. The first are called *encoding* gates as they only depend on input data values, whereas the *trainable* gates depend on internal parameters that are optimized during training. A typical instance of a Variational Quantum Circuit (VQC) is illustrated in Figure 1.2. In the recent literature, these gates are often grouped as *layers*, which is not mandatory, since any circuit can be sliced into alternating sequences of encoding and training blocks (even if containing a single gate).

Any quantum unitary implements the evolution of a quantum system under a Hamiltonian. Thus, we choose to write the ℓ^{th} encoding gates as $\exp(-ix_iH_\ell)$, where x_i is one of the d components of x, and H_ℓ is a Hamiltonian matrix of size 2^p if p is the number of qubits this gate acts on. We will note L the number of encoding gates for each dimension of x (the same for each dimension, for notation simplicity).

In this framework, the aim is to find the optimal mapping between data points and their target values. This is done by optimizing the parameters θ to find the best guess f^* such that

$$f^* = \arg\min_\theta \frac{1}{M}\sum_{i=1}^{M} l(f(x_i;\theta), y_i) \tag{1.10}$$

where l is a cost function adapted to the task. For a standard regression task, we can choose $l(z,y) = |z-y|^2$

1.4 Quantum models are large Fourier series

It is known since (SSM21b) that the family of quantum models defined Equation (1.9) can be rewritten as a Fourier series:

$$f(x;\theta) = \sum_{\omega\in\Omega} c_\omega e^{i\omega x} \tag{1.11}$$

where the spectrum Ω of frequencies is determined by the ensemble of eigenvalues of the encoding Hamiltonians and the coefficients c_ω depend on the parametrized ansatz, as pictured in Figure 1.2.

In order to get familiar with the structure of the spectrum, I explicitly show an example of Ω in the case of a one dimensional data input ($\mathcal{X} = \mathbb{R}$) and with a variational circuit containing only L encoding gates. The accessible frequency spectrum Ω is the ensemble of all the differences between all possible sums of the eigenvalues of the encoding gates as shown in Figure 1.1. I note λ_ℓ^k the k^{th} eigenvalue of the ℓ^{th} encoding Hamiltonian H_ℓ having d_ℓ eigenvalues. I use the multi-index $\boldsymbol{i} = (i_1, \dots, i_L)$ indicating which eigenvalue is taken from each encoding Hamiltonian. We define $\Lambda_{\boldsymbol{i}}$ as

$$\Lambda_{\boldsymbol{i}} = \lambda_1^{i_1} + \cdots + \lambda_L^{i_L} \tag{1.12}$$

Finally, I can express Ω, the set of frequencies :

$$\Omega = \left\{ \Lambda_{\boldsymbol{i}} - \Lambda_{\boldsymbol{j}}, \boldsymbol{i}, \boldsymbol{j} \in \prod_{\ell=1}^{L} [|1, d_\ell|] \right\}, \tag{1.13}$$

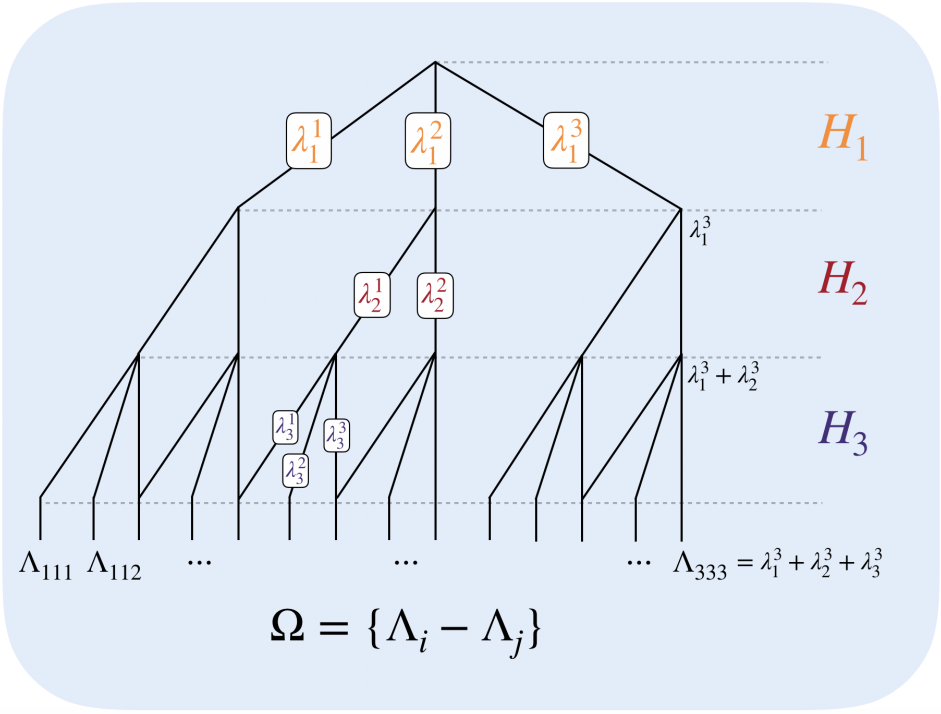


Figure 1.1: **From encoding Hamiltonians to frequencies.** The frequencies composing the VQC model (on one dimensional input) come from all the combinations of eigenvalues from each encoding Hamiltonians. This can be seen as a tree, with $L = 3$ Hamiltonians in this figure. We also see potential redundancy in the leaves.

The simplest encoding is called Pauli encoding, where all encoding Hamiltonians are Pauli matrices (e.g. encoding gates $R_Z(x) = e^{-i\frac{x}{2}\sigma_Z}$) as in (SSM21b; CGFM+21). In this case, all the eigenvalues are $\lambda = \pm 1/2$, and therefore, the $\Lambda_{\boldsymbol{i}}$ are all the integers (or half-integers, if L is odd) in $[-L/2, L/2]$. It follows that the set of distinct values in Ω is simply the set of integers in $[\![-L, L]\!]$. In this case, there are many redundant frequencies, due to the fact that all Pauli eigenvalues are the same. Namely, only $2L + 1$ distinct frequencies among the 2^{2L} possible values of $\Lambda_{\boldsymbol{i}} - \Lambda_{\boldsymbol{j}}$. As shown in Figure 1.1, more various eigenvalues would create more distinct frequencies in the end.

We can now generalize, if we now have that $\mathcal{X} = \mathbb{R}^d$, such that we encode a vector $x = (x_1, \ldots, x_d)$ in our quantum model, and we assume that each encoding gate is a function of a single coordinate x_i then Ω becomes the following $d-$dimensional Cartesian product

$$\Omega = \Omega_1 \times \Omega_2 \times \cdots \times \Omega_d \tag{1.14}$$

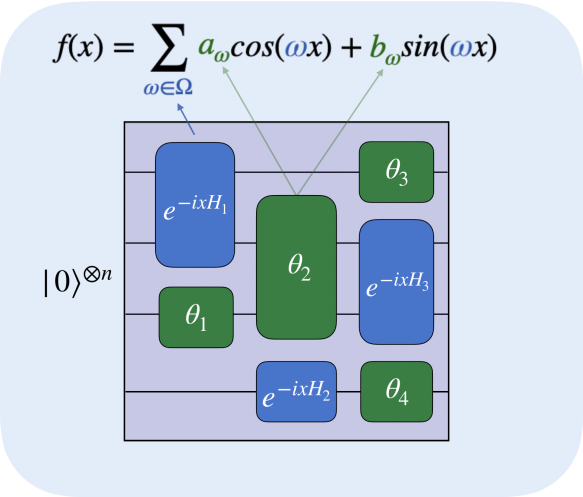


Figure 1.2: **Variational quantum circuits give rise to Fourier series.** In a quantum machine learning task, classical data is encoded in a subset of variational gates of a quantum circuit (green), while blue gates are trainable.

where each Ω_κ is defined as in Equation (1.13) on its own set of Hamiltonians. In this context, one can note that the frequencies ω are now vectors in $\mathbb{R}^d$ and there are d different trees to build Ω (see Figure 1.1). Note that for notation simplicity, we assumed that L gates were applied on each input's component, but it can be generalized to any number of gates per dimension.

We therefore see that the size of the spectrum $|\Omega|$ can potentially grow exponentially with the number of encoding gates and the dimension of the input data. For instance, if we consider a d-dimensional vector x and L Pauli-encoding gates for each dimension in such a way that there are Ld encoding gates in the VQC. According to equation (Equation (1.14)), the size of the spectrum Ω would scale as $O(L^d)$, which becomes quickly intractable as d increases. As an example, the spectrum associated to a VQC with $L = 20$ encoding gates and $d = 16$ would require more than one hundred times the world's storage data capacity available in 2007 to be stored (HL11). The ability to represent a model with such a high dimension is the intuitive reason one would think of having an advantage using quantum models.

Chapter 2

Classically approximating variational quantum models

In the previous chapter, I introduced variational quantum circuits and how one can use them for a machine learning task. Understanding their potential advantage is therefore a very dynamic research question in the community. Much of the intuition about the potential advantage of VQCs is the fact that they can represent a linear function in a very high dimensional space, and therefore would be very costly to evaluate.

However, this ability is not enough to guarantee an advantage. In this chapter, I show that high dimensional models can sometimes be approximated with smaller efficient models. I propose a classical sampling method that may closely approximate a VQC with Hamiltonian encoding, given only the description of its architecture. It would consist in building a classical approximator $\tilde{f}$ as

$$\tilde{f}(x) = \sum_{\omega \in \tilde{\Omega}} \tilde{c}_\omega e^{i\omega x} \tag{2.1}$$

such that $\tilde{\Omega}$ is of tractable size and the two solution are close, written as $\left\| f(x) - \tilde{f}(x) \right\| \leq \varepsilon$, using a given error measure (ℓ_∞ or ℓ_2 norms, for instance).

It uses the seminal proposal of Random Fourier Features (RFF) (RR07) and the fact that VQCs can be seen as large Fourier series. I provide general theoretical bounds for classically approximating models built from exponentially large quantum feature space by sampling a few frequencies to build an equivalent low dimensional kernel, and I show numerically that this approximation is efficient for several encoding strategies. In the cases where the construction of such a model is possible, it would imply that although classically simulating the encoding VQC

might not be possible, the quantum model that emerges from it can be efficiently approximated in a classical way.

This chapter is based on the following paper

- (LTD+22) Jonas Landman, Slimane Thabet, Constantin Dalyac, Hela Mhiri, Elham Kashefi (2022). *Classically Approximating Variational Quantum Machine Learning with Random Fourier Features.* International Conference on Learning Representations 2023

2.1 Quantum models are shift-invariant kernel methods

As the quantum model is a real-valued function, it follows that $\omega \in \Omega$ implies $-\omega \in \Omega$ and $c_\omega = c^*_{-\omega}$. We express the Fourier series of the quantum model as a sum of trigonometric functions by defining for every $\omega \in \Omega$:

$$a_\omega := c_\omega + c_{-\omega} \in \mathbb{R} \tag{2.2}$$

$$b_\omega := \frac{1}{i}(c_\omega - c_{-\omega}) \in \mathbb{R} \tag{2.3}$$

such that

$$\begin{aligned} f(x;\theta) &= \sum_{\omega \in \Omega_+} c_\omega e^{i\omega x} + c_{-\omega} e^{-i\omega x} \\ &= \sum_{\omega \in \Omega_+} a_\omega \cos(\omega x) + b_\omega \sin(\omega x), \end{aligned} \tag{2.4}$$

where Ω_+ contains only half of the frequencies from Ω. Considering only Pauli gates, if $d = 1$, we simply have $\Omega = [\![-L, L]\!]$ and $\Omega_+ = [\![0, L]\!]$. In dimension d, we have $\Omega = [\![-L, L]\!]^d$ and Ω_+ is built by keeping half of the frequencies (after removing those of opposite sign), plus the null vector. In the end, we have

$$|\Omega_+| = \frac{(2L+1)^d - 1}{2} + 1 \tag{2.5}$$

With a more general encoding scheme, if there is a different number of distinct positive frequencies per dimension, the formula is different but is built similarly.

In the following parts, we will focus solely on Ω_+ and conveniently drop the $+$ subscript.

We can also define the feature map of the quantum model (Sch21) as

$$f(x;\theta) = \langle \psi(x;\theta) | O | \psi(x;\theta) \rangle = \mathbf{w}(\theta)^T \phi(x) \tag{2.6}$$

where $\phi(x)$ is the *feature vector*, the mapping of the initial input into a larger *feature space*, where the new distribution of the data is supposed to make the classification (or regression) solvable with only a linear model. This linear model is in fact the inner product between $\phi(x)$ and a trainable *weight vector* $\mathbf{w}$. In the case of VQCs, we can explicitly express them as:

$$\phi(x) = \frac{1}{\sqrt{|\Omega|}} \begin{bmatrix} cos(\omega^T x) \\ sin(\omega^T x) \\ \vdots \end{bmatrix}_{\omega \in \Omega}, \quad \mathbf{w}(\theta) = \begin{bmatrix} a_\omega \\ b_\omega \\ \vdots \end{bmatrix}_{\omega \in \Omega} \tag{2.7}$$

If the spectrum Ω is known and accessible, one can fit the quantum model by retrieving the coefficients a_ω, b_ω associated to each frequency ω. This can be done by using general linear ridge regression techniques (see Section 1.2). Interestingly, there exists a dual formulation of the linear ridge regression that depends entirely on the kernel function associated to the model (BN06). In our case, the related kernel function is defined by:

$$\begin{aligned} k(x, x') &= \langle \phi(x), \phi(x') \rangle \\ &= \frac{1}{|\Omega|} \sum_{\omega \in \Omega} \cos\left(\omega^\top x\right) \cos\left(\omega^\top x'\right) + \sin\left(\omega^\top x\right) \sin\left(\omega^\top x'\right) \\ &= \frac{1}{|\Omega|} \sum_{\omega \in \Omega} \cos(\omega(x - x')) \end{aligned} \tag{2.8}$$

which is a shift-invariant kernel, meaning that $k(x, x')$ can be written $\bar{k}(x - x')$ where $\bar{k}$ is a one variable function.

It is known that quantum models from VQCs are related to kernel methods (Sch21), which means that it is equally possible to fit the quantum model by approximating the related kernel function. These kernels are high dimensional (since the Ω can be numerous) which makes it hard to simulate classically in practice. But due to their shift-invariance, we propose to study their classical approximation using Random Fourier Features (RFF), a seminal method known to be powerful approximator of high-dimensional kernels (RR07).

One should note however that classically fitting the coefficients a_ω and b_ω will not necessarily lead to the same output as fitting the quantum model. This distinction is the subject of Chapter 3. For this chapter, we will consider that it is a decent approximation.

2.2 Random Fourier features approximates high-dimensional kernels

In this section, we explain the key results of the classical method called Random Fourier Features (RFF) (RR07; LTOS19; SS15). We will use this method to create several classical sampling algorithms for approximating VQCs.

Let $\mathcal{X} \subset \mathbb{R}^d$ be a compact domain and $k : \mathcal{X} \times \mathcal{X} \longrightarrow \mathbb{R}$ be a kernel function. We assume k is shift invariant, meaning

$$k(x, y) = \overline{k}(x - y) = \overline{k}(\delta) \tag{2.9}$$

where $\overline{k} : \mathcal{X} \longrightarrow \mathbb{R}$ is a single variable function, and we will note $\overline{k} = k$ to simplify the notation.

Bochner's theorem (Rud17) ensures that the Fourier transform of k is a positive function and we can write

$$k(\delta) = \int_{\omega \in \mathcal{X}} p(\omega) e^{-i\omega^T \delta} d\omega \tag{2.10}$$

If we assume k is also normalized (ie $\int_{\omega \in \mathcal{X}} p(\omega) d\omega = 1$, then the Fourier transform $p(\omega)$ of k can be seen as a probability distribution. With a dataset of M points, fitting a Kernel Ridge Regression (KRR) model with the kernel k necessitates M^2 operations to compute the kernel matrix and $\mathcal{O}(M^3)$ to invert it. This becomes impractical when M reaches high value in modern big datasets.

The idea of the Random Fourier Feature method (RR07) is to approximate the kernel k by

$$\tilde{k}(x, y) \simeq \tilde{\phi}(y)^T \tilde{\phi}(x) \tag{2.11}$$

where $\tilde{\phi}(x) = \frac{1}{\sqrt{D}} \begin{bmatrix} \cos\left(\omega_i^T x\right) \\ \sin\left(\omega_i^T x\right) \end{bmatrix}_{i \in [\![1, D]\!]}$ and the ω_is are D frequencies sampled iid from the frequency distribution $p(\omega)$. Formally, it is a Monte-Carlo estimate of k. Note that $p(\omega)$ can be analytically found in some cases such as Gaussian or Cauchy kernel (RR07).

Then instead of fitting a KRR for k, one will solve a Linear Ridge Regression (LRR) with ϕ (see details in Section 1.2). The two problems are equivalent (BN06), and the number of operations needed for the LRR is $\mathcal{O}(MD^2 + D^3)$, $\mathcal{O}(MD^2)$ to compute the covariance matrix and $\mathcal{O}(D^3)$ to invert it. If D is much smaller than M, it is much cheaper than solving the KRR directly. Even if D is so big that the linear regression cannot be exactly solved, one can employ stochastic gradient descent or adaptive momentum optimizers such as Adam (KB14).

The output of the LRR or gradient descent is simply a weight vector $\tilde{\mathbf{w}}$ that is used to create the approximate function

$$\tilde{f} = \tilde{\mathbf{w}}^T \tilde{\phi}(x) \tag{2.12}$$

We give here two useful results about the bounds of the error of the RFF method. RFFs are supposed to approximate a certain kernel k by using fewer features. Intuitively, not enough features would lead to imprecise solutions. The following theorems ((RR07; SS15)) bound the error obtained when comparing the kernel $k(x, y)$ by the RFF approximator $\tilde{\phi}(x)^T \tilde{\phi}(y)$ using D samples.

where $p(\omega)$ is the distribution of the frequencies ω.

Theorem 2.1. *(RR07) Let $\mathcal{X}$ be a compact set of $\mathbb{R}^d$, and $\epsilon > 0$.*

$$\mathbb{P}\left(\sup_{x,y \in \mathcal{X}} |k(x-y) - \tilde{\phi}(x)^T \tilde{\phi}(y)| \geq \epsilon\right) \leq 66 \left(\frac{\sigma_p |\mathcal{X}|}{\epsilon}\right)^2 exp\left(-\frac{D\epsilon^2}{4(d+2)}\right) \tag{2.13}$$

with $\sigma_p^2 = \mathbb{E}_p(\omega^T \omega)$, the variance of the frequencies' distribution, and $|\mathcal{X}| = max_{x,x' \in \mathcal{X}}(\|x - x'\|)$ the diameter of $\mathcal{X}$.

The following theorem (SS15) bounds the actual prediction error when using RFF compared to the KRR estimate. The formula in the original reference contains a sign error and we correct it here.

Theorem 2.2. *(SS15) Let $\mathcal{X}$ be a compact set of $\mathbb{R}^d$, and $\epsilon > 0$. We consider a training set $\{(x_i, y_i)\}_{i=1}^M$. Let f be the KRR model obtained with the true kernel k and regularization $\lambda = M\lambda_0$ for $\lambda_0 > 0$, and $\tilde{f}$ the KRR model obtained with the approximate kernel and the same regularization. Then we can guarantee $|f(x) - \tilde{f}(x)| \leq \epsilon$ for all $x \in \mathcal{X}$ with probability $1 - \delta$ for a number D of samples given by:*

$$D = \Omega\left(d\left(\frac{(\lambda_0 + 1)\sigma_y}{\lambda_0^2 \epsilon}\right)^2 \left[log(\sigma_p |\mathcal{X}|) + log\frac{(\lambda_0 + 1)\sigma_y}{\lambda_0^2 \epsilon} - log\delta\right]\right) \tag{2.14}$$

with $\sigma_y^2 = \frac{1}{M}\sum_{i=1}^M y_i^2$ and σ_p, $|\mathcal{X}|$ being defined in Theorem 2.1. We recall that in Equation (2.14) the notation Ω stands for the computational complexity "Big-Ω" notation.

2.3 Random Fourier features for approximating VQCs

In this section, I present in detail our solutions to approximate a VQC using classical methods. The intuitive idea is to sample some frequencies from the VQC's frequency domain Ω, and train a classical model from them, using the RFF methods from Section 2.2. The general idea is illustrated in Figure 2.1. I first introduce some related work (Subsection 2.3.1), then present three different strategies to sample those frequencies to build classical models (Subsection 2.3.2). I finally provide theoretical bounds on the number of samples required (Subsection 2.3.3) and present potential limitations to the methods (Subsection 2.3.4), opening the way for VQCs with strong advantage over classical methods.

2.3.1 Related work

A recent work (SEM22) independently proposed a similar approach where classical surrogate methods approximate VQCs. The difference with this work is the necessity of having access to all Ω, the totality of the frequencies of the VQC considered, without sampling from them.

Indeed, if Ω is known, the coefficients a_ω and b_ω of the VQC function (see Equation (2.4)) can be easily fitted by solving the classical least square problem. Namely, one determines $\mathbf{w}^*$ such that

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \frac{1}{M}\sum_{i=1}^M |\mathbf{w}^T \phi(x_i) - y_i|^2 + \frac{\lambda_0}{M}||\mathbf{w}||^2 \tag{2.15}$$

where $\phi(x) = \begin{bmatrix}\cos(\omega x) \\ \sin(\omega x)\end{bmatrix}_{\omega \in \Omega}$, and λ_0 is the regularisation parameter. As explained in the previous section, with a dataset of M points, this can be solved exactly using matrix inversion

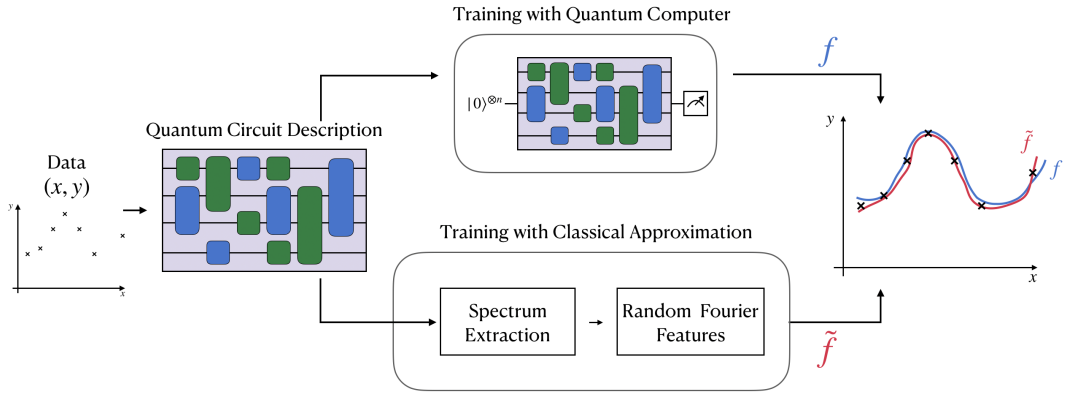


Figure 2.1: **Random Fourier features as a classical approximator of quantum models.** Instead of training a Variational Quantum Circuit by using a quantum computer, I propose to train a classical linear model built by sampling a few frequencies of the quantum model. These frequencies can be derived from the quantum circuit architecture, in particular from the encoding gates. Using random Fourier features, one can build a classical model which could perform as good as the quantum model with a bounded error and a tractable number of random features.

in $\mathcal{O}(M|\Omega|^2 + |\Omega|^3)$ operations if $M \geq 2|\Omega|$. If the inequality is not fulfilled or if $|\Omega|^3$ is too big, one would use stochastic gradient descent instead of matrix inversion.

However, this method assumes that Ω is known, and is not too large, which will usually be the case as we show in Section 1.4. One should also be able to enumerate all individual frequencies $\omega \in \Omega$. Moreover, as we will show the redundancy of some frequencies in Ω has a key importance, which is not captured by such a method.

For completeness, we note from the seminal work (Sch21) that the author briefly mentions the idea of approximating kernels with RFF. Similarly, in a more recent work (PS22), the authors mention RFF as a sampling strategy on VQCs with shift invariant kernels, without further details.

2.3.2 RFF sampling strategies

We now propose 3 types of sampling strategies of the spectrum Ω. We will explain in which case these strategies are suited, according to the type of the encoding circuit, the dimension of the input vectors, the number of training points, and so on.

As shown in Equation (2.7) and Equation (2.8), the corresponding kernel k of the VQC is built from frequencies $\omega \in \Omega$. In section Section 2.1, we have shown that this kernel is shift-invariant, which ensures us the efficiency of RFF to approximate them. Applying the RFF method would consist in sampling from Ω and reconstructing a classical model which should approximate the VQC output function f.

2.3.2.1 RFF with Distinct sampling

This strategy describe the basic approach of using RFF for approximating VQCs. We assume the ability to sample from Ω. It is straightforwardly following the method described in Section Section 2.2, and given in Algorithm 1.

The benefit of this method, compared to the one presented in Subsection 2.3.1, is the ability to use far fewer frequencies than the actual size of Ω. As shown in Subsection 2.3.3, one might require a number D of samples scaling linearly with the dimension of the input d, and logarithmically with the size of Ω.

Algorithm 1 RFF with Distinct sampling

Require: a VQC model f, and M data points $\{x_j\}_{j\in[M]}$
Ensure: Approximate function $\tilde{f}$
1: Diagonalize the Hamiltonians of the VQC's encoding gates.
2: Use their eigenvalues to obtain all frequencies $\omega \in \Omega$, as in Equation (1.13)
3: Sample D frequencies $(\omega_1, \cdots, \omega_D)$ from Ω
4: Construct the approximated kernel $\tilde{k}(x,y) = \tilde{\phi}(y)^T\tilde{\phi}(x)$ with $\tilde{\phi}(x) = \frac{1}{\sqrt{D}}\begin{bmatrix} cos(\omega_i^T x) \\ sin(\omega_i^T x)\end{bmatrix}_{i\in[\![1,D]\!]}$
5: Solve the LRR problem (Section 1.2), and obtain a weight vector $\tilde{\mathbf{w}}$.
6: Obtain the approximated function $\tilde{f}(x) = \tilde{\mathbf{w}}^T\tilde{\phi(x)}$

2.3.2.2 RFF with Tree sampling

The abovementioned method requires constructing explicitly Ω, which can become exponentially large if the dimension d of the datapoints is high (see Section 1.4). The size of Ω is also increased if the encoding Hamiltonians have many eigenvalues (when the Hamiltonian is complex and acts on many qubits), or if many encoding gates are used. A large Ω is indeed the main interest of VQCs in the first place, promising a large expressivity.

In some cases, and in particular for a VQC using many Pauli encoding gates, a lot of redundancy occurs in the final frequencies. Indeed, if many eigenvalues are equal, the tree leaves will become redundant. Very small eigenvalues can also create groups of frequencies extremely close to each other, which in some use cases, when tuning their coefficients a_ω and b_ω, can be considered as redundancy. In the numerical experiments (see Figure 2.3), we observe an interesting phenomenon: On average, the frequencies with the more redundancy tend to obtain larger coefficients. Conversely, isolated frequencies are very likely to have small coefficients in comparison, making them "ghost" frequencies in Ω. For VQC with solely Pauli encoding gates, we observe that the coefficients of high frequencies are almost stuck to zero during training. Therefore, one can argue that the *Distinct Sampling* described above can reach even more frequencies than the corresponding VQC. However, if one wants to closely approximate a given

VQC with RFF, one would not want to sample such isolated frequencies from Ω but instead draw with more probability the redundant frequencies.

Algorithm 2 RFF with Tree sampling

Require: a VQC model f, and M datapoints $\{x_j\}_{j\in[M]}$
Ensure: Approximate function $\tilde{f}$
1: Diagonalize the Hamiltonians of the VQC's encoding gates.
2: Sample D paths from the tree shown in Figure 1.1, obtain D frequencies $(\omega_1, \cdots, \omega_D)$ from Ω
3: Follow steps 4-6 of Algorithm 1.

This is what we try to achieve with *Tree Sampling*. Knowing the eigenvalue decomposition of each encoding's Hamiltonian, we propose to directly sample from the tree shown in Figure 1.1. The first advantage of this method is that it does not require computing the whole set Ω, but only draw D paths through the tree (which can be used to generate up to $\binom{D}{2} + 1$ positive frequencies, with potential redundancy). Second, it naturally tends to sample more frequencies that are redundant, and therefore more key to approximate the VQC's function. Overall, it could speed up the running time and necessitate fewer samples.

2.3.2.3 RFF with Grid sampling

The two above methods suffer from a common caveat: if one or more of the encoding Hamiltonians are hard to diagonalize, sampling the VQC's frequencies is not possible as it prevents us from building some of the branches of the tree shown in Figure 1.1.

Even in this case, we propose a method to approximate the VQC. If the frequencies are unknown, but one can guess an upper bound or their maximum value, we propose the following strategy: We create a grid of frequencies regularly disposed between zero and the upper bound ω_{max}, on each dimension. In practice, if unknown, the value of ω_{max} can simply be the largest frequency learnable by the Shannon criterion (see Section Subsection 2.3.3) hence half of the number of training points. Letting $s > 0$ be the step on this grid, the number of frequencies on a single dimension is given by ω_{max}/s. Over all dimensions, there are $\lceil(\omega_{max}/s)\rceil^d$ frequency vectors.

Therefore, instead of sampling from actual frequencies in Ω, one could sample blindly from this grid, hence the name *Grid Sampling*. At first sight, it might seem ineffective, since none of the frequencies actually in Ω may be represented in the grid. But I show in Theorem 2.4 that the error between the VQC's model f and the approximation $\tilde{f}$ coming from the grid can be bounded by s. When s is small enough, the number D of samples necessary to reach an error $\epsilon > 0$ grows like $log(1/s)/\epsilon^2$ which is surprisingly efficient. However, the trade-off comes from the fact that a small s means a very large grid, in particular in high dimension.

Algorithm 3 RFF with Grid Sampling

Require: Assumption on the highest frequency ω_{max}, a step $s > 0$ and M datapoints $\{x_j\}_{j\in[M]}$
Ensure: Approximate function $\tilde{f}$
1: Create a regular grid in $[0, \omega_{max}]^d$ with step s.
2: Sample D frequencies $(\omega_1, \cdots, \omega_D)$ from the grid.
3: Follow steps 4-6 of Algorithm Algorithm 1.

2.3.3 Number of samples and approximation error

In (SS15), authors bound the resulting approximation error in the RFF method, see Theorem 2.2. In this theorem, the error between the final functions f and $\tilde{f}$ is considered. One can see that if the error must be constrained such that $|f(x) - \tilde{f}(x)| \leq \epsilon$, for $\epsilon > 0$, one can derive a lower bound on the number D of samples necessary for the approximation. Fortunately, the bound on D grows linearly with the input dimension d, and logarithmically with σ_p which linked to the variance of the frequency distribution $p(\omega)$.

In our case, the continuous distribution $p(\omega)$ will be replaced by the actual set of frequencies Ω,

$$p(\omega) = \frac{1}{|\Omega|} \sum_{\omega\in\Omega} \delta_\omega \tag{2.16}$$

where δ_ω represents the Dirac distribution at ω. As a result, we can write the discretized variance as

$$\sigma_p = \sum_{\omega\in\Omega} p(\omega)\omega^T\omega \tag{2.17}$$

From this, we want to know the relation between the number D of samples necessary and the size of Ω, or even the number L of encoding gates per dimension.

In the general case, we consider that σ_p is the average value of $\omega^T\omega$. The more the frequencies are spread, the higher σ_p will be, but the number of frequencies in itself doesn't seem to play the key role here.

Finally, note that it is important to take into account the Shannon criterion, stating that one needs at least $2\omega_{\max}$ training points to estimate the coefficients of a Fourier series of maximum frequency $\omega_{\max}$. In practice, it puts some limitation on the largest frequency one can expect to learn (both classically and quantumly) given an input dataset. Large frequencies with VQCs, as in (STJ22) would have a limited interest with a restricted number of training points. The efficiency of RFF against VQCs in such cases becomes even more interesting, as it allows reducing the number D of sample compared to the actual exponential size of Ω.

2.3.3.1 Pauli encoding

We provide here a bound on the minimum of samples required to achieve a certain error between the RFF model and the complete model in the case of Pauli encoding in the distinct sampling strategy. The proof and details for this theorem is shown in Appendix A.1.

Theorem 2.3. *Let $\mathcal{X}$ be a compact set of $\mathbb{R}^d$, and $\epsilon > 0$. We consider a training set $\{(x_i, y_i)\}_{i=1}^M$. Let f be a VQC model with L encoding Pauli gates on each of the d dimensions and full freedom on the associated frequency coefficients, trained with a regularization λ. Let $\sigma_y^2 = \frac{1}{M}\sum_{i=1}^M y_i^2$ and $|\mathcal{X}|$ the diameter of $\mathcal{X}$. Let $\tilde{f}$ be the RFF model with D samples in the distinct sampling strategy trained on the same dataset and the same regularization. Then we can guarantee $|f(x) - \tilde{f}(x)| \leq \epsilon$ for all $x \in \mathcal{X}$ with probability $1 - \delta$ for a number D of samples given by:*

$$D = \Omega\left(\frac{dC_1(1+\lambda)^2}{\lambda^4\epsilon^2}\left[\log\left(dL^2|\mathcal{X}|\right) + \log\frac{C_2(1+\lambda)}{\epsilon\lambda^2} - \log\delta\right]\right) \tag{2.18}$$

with C_1, C_2 being constants depending on σ_y, $|\mathcal{X}|$. We recall that in Equation (2.18) the notation Ω stands for the computational complexity "Big-Ω" notation.

We can conclude that the number D of samples grows linearly with the dimension d, and logarithmically with the size of Ω (see Equation (2.5)). It means that even though the number of frequencies in the spectrum of the quantum circuit is high, only a few of them are useful in the model. This fact limits the quantum advantage of such circuits.

One should note however that this theorem assumes that the inductive biases of the quantum and classical models are similar, which is not necessarily true. The case where they are different is the subject of Chapter 3.

However, the scaling in ϵ and λ, respectively in $\Omega(1/\epsilon^2)$ and $\Omega(1/\lambda^4)$ is not favorable, and can limit in practice the use of the RFF method.

One may think at first sight that the RFF method would be efficient to approximate the outputs of any VQC, or find a "classical surrogate" (SEM22). Instead, the bound that is provided is on the error between a VQC trained on an independent data source and a RFF model trained on the same data. There is contained in this result the potential incompleteness of the dataset to render an accurate representation of the underlying data distribution. If the dataset fails to correctly represent the data distribution, then the VQC will fail to correctly model it, and the theorem provide the minimal number of samples to perform "as badly". This was tested in the numerical simulations in Section 2.4. However, we recall that these are bounds that can hardly be reached in practice with the current classical and quantum computing resources. They give an intuition on the asymptotic scaling as quantum circuits become larger.

2.3.3.2 Grid sampling

We provide here a bound on the minimum number of samples required to achieve a certain error between the RFF model and the complete model in the case of a general encoding in the gird sampling strategy. The proof and details for this theorem is shown in Appendix A.1.

Theorem 2.4. *Let $\mathcal{X}$ be a compact set of $\mathbb{R}^d$, and $\epsilon > 0$. We consider a training set $\{(x_i, y_i)\}_{i=1}^M$. Let f be a VQC model with any hamiltonian encoding, with a maximum individual frequency*

ω_{max} and full freedom on the associated frequency coefficients, trained with a regularization λ. Let $\sigma_y^2 = \frac{1}{M}\sum_{i=1}^{M} y_i^2$ and $|\mathcal{X}|$ the diameter of $\mathcal{X}$. Let $\tilde{f}$ be the RFF model with D samples in the grid strategy trained on the same dataset and the same regularization. Let $C = |f|_\infty|\mathcal{X}|$ and s the sampling rate defined in the grid sampling strategy. Then we can guarantee $|f(x) - \tilde{f}(x)| \leq \epsilon$ for all $x \in \mathcal{X}$ for $0 < s < \frac{\epsilon}{C}$ with probability $1 - \delta$ for a number D of samples given by:

$$D = \Omega\left(\frac{dC_1(1+\lambda)}{\lambda^4(\epsilon - sC)^2}\left[\log(\omega_{max}|\mathcal{X}|/s) \; + \; \log\frac{C_2(1+\lambda)}{\lambda^2(\epsilon - sC)} - \log\delta\right]\right) \tag{2.19}$$

with C_1 and C_2 being constants depending on σ_y and $|\mathcal{X}|$. We recall that in Equation (2.19) the notation Ω stands for the computational complexity "Big-Ω" notation.

The bound depends on the maximum individual frequency ω_{max} that is equal to the sum of the largest eigenvalues of encoding hamiltonians. It can be made arbitrarily big by introducing a scaling coefficient α in $\exp(-ix\,\alpha H)$. In the case of Pauli encoding, we have that $\omega_{max} = L$.

2.3.4 Limitations of RFF for approximating VQCs

We now have seen the theoretical power of Random Fourier Features and three different adaptations to approximate VQCs in practice. Since many parameters are to be taken into account (size and structure of Ω, number of qubits, circuit depth, number of training points, input dimension, encoding Hamiltonians, etc.), it is natural to ask ourselves which of the three strategies is recommended given a use case, and whether there are any use cases for which none of them work.

As seen in Subsection 2.3.3, we know the lower bound on the number of samples to draw in RFF, to reach a specific error. This bound grows linearly with the input dimension d, and logarithmically with the size of Ω (itself growing like L^d so exponentially in d). Nonetheless, one could see in practice that very large spectrum are hard to approximate, simply because it would require much more samples. This scaling will be judged once such VQCs will be actually implemented on large enough quantum computers (with enough qubits and/or long coherence).

The size of Ω increases as well when the encoding Hamiltonians have distinct eigenvalues and are acting on many qubits. Therefore, quantum computers allowing for many qubits and various high locality Hamiltonians would be a plus for enlarging the spectrum.

As the Hamiltonians become larger and their eigenvalues complex, we could reach a limit where it becomes impossible to diagonalize them. In such a case, without sampling access to Ω, the *Distinct* and *Tree* sampling strategies would be unavailable. The *Grid* sampling scheme would suffice until suffering from the high dimensionality or other factors detailed above.

Another important limitation of the theoretical bounds is the scaling in the regularization parameter. As illustrated in Figure 2.9, this constant prefactor make the bound still reach intractable numbers, so it is of little value to determine *a priori* if the RFF method will be efficient.

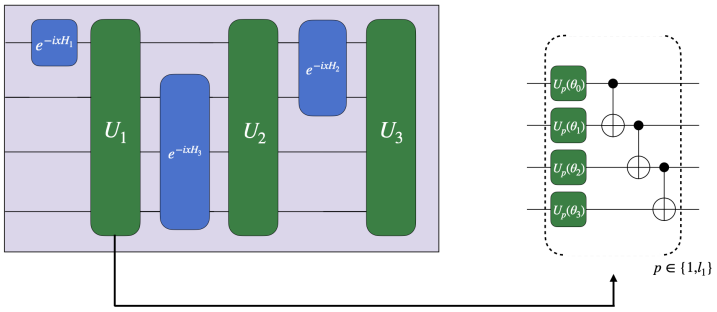


Figure 2.2: **Random instance of a VQC.** In this example, three encoding Hamiltonians $\{H_1, H_2, H_3\}$ are randomly assigned over four qubits, and load a 1-dimensional vector x. Following each encoding gate H_i, an ansatz with trainable parameters and a ladder of CNOTs is applied, l_i times in a row.

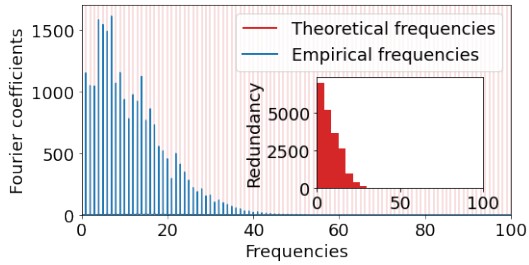
Overall, some limits for our classical methods can be guessed and observed already, but the main ones remain to be measured on real and larger scale quantum computers. This research is left for future work. On another hand, one could want to understand better the relation between the available frequencies and their amplitude in practice, to find potential properties that could be in favor or not of the VQCs.

Finally, we want to insist on the fact that the assumptions on VQCs are crucial on the whole construction that we propose, and that some of them could be questioned, especially concerning the encoding. For instance, when encoding vectors $x = (x_1, \cdots, x_d)$, not having encoding gates expressed as $exp(-x_i H)$ could potentially change the expression of $f(x; \theta)$ (Equation (1.11)) and therefore could change the fact that the associated kernel would be easily expressed as a Fourier series, with shift-invariance. For instance, in (KPE21), the authors use $exp(-arcsin(x_i)H)$ to encode data, resulting in f being expressed in the Chebyshev basis instead of the Fourier one. More generally, understanding what happens with encodings of the form $exp(-g(x_i)H)$, and whether one can still use our classical approximation methods is tackled in Chapter 3.

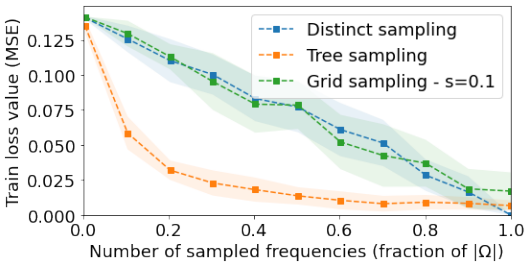
2.4 Numerical experiments

In this section, I aim to assess the accuracy and efficiency of the classical methods to approximate VQCs in practice. Three types of simulations have been conducted. In Subsection 2.4.1, I instantiate random VQCs and try to mimic their output using our RFF methods. In Subsection 2.4.2, I create artificial functions and compare VQCs and RFF on the same task of learning it. In Section Subsection 2.4.3, I compare VQCs and RFF on real datasets. Finally, I observe the scaling of our methods in Section Subsection 2.4.4.

As shown in Figure 2.2, a typical random VQC instance is built from a list of general



(a) Average Fourier Transform of the VQC's quantum models. The frequencies with high coefficients are the ones with high redundancy in Ω (seen in the inner red histogram). Frequencies over 100 have negligible coefficients and redundancy, and therefore are not shown.



(b) Evolution of RFF train loss as a function of the relative number of frequencies sampled. The *Tree* sampling strategy takes advantage of the high redundancy to sample less frequencies to reach a good approximation.

Figure 2.3: Random 1d VQCs with L=200 Pauli encoding gates, averaged over 10 different random initialization.

encoding Hamiltonians $\{H_1, \cdots, H_k\}$, applied to randomly selected qubits according to their locality. The number of qubits is fixed to 5 in the following experiments (Note that the number of qubits has no impact on the expressivity on the spectrum a priori, it will only influence the spans of coefficients).

2.4.1 Using RFF to mimic random VQCs

In a first stage, I focus on approximating random VQCs (i.e. with random parameter initialization) using Random Fourier features. To this end, the quantum spectrum Ω is fixed by making a certain choice about the structure of the encoding gates. The training dataset is $\{X_{grid}, Y_{grid}\}$ with X_{grid} being a set of d-dimensional data points spaced uniformly on the interval $\prod_{i=1}^{d}[0, x_{max_i}]$ and Y_{grid} the evaluation of the quantum circuit on the input dataset X_{grid}. I then observe and evaluate the performance of our three RFF strategies (*Distinct*, *Tree*, and *Grid* sampling, see Section Subsection 2.3.2) to approximate the quantum model. For completeness, I have tested our methods on different types of VQCs: some with basic Pauli encoding in 1 dimension (Figure 2.3), in higher dimension (Figure 2.4), some with more elaborate Hamiltonians (Figure 2.5), and with the scaled Pauli encoding as in (STJ22) (Figure 2.6). For each type, I have also observed the actual Fourier Transform of the random VQCs model on average, to understand which frequencies appear more frequently in their spectrum.

It can be noted that the number of data points in X_{grid} needed to efficiently learn the quantum function is $N > \prod_{i=1}^{d} \frac{x_{max_i} w_{max_i}}{\pi}$. This choice is basically related to the *Shannon criterion* for effective sampling in order to reconstruct the full function covering all of its frequencies. Moreover, for the solution to be unique and hence for the least square problem introduced in Equation (2.15) to be well defined, we better choose N to be bigger than the number of features in the regression problem (these two criteria coincide in the case of Pauli

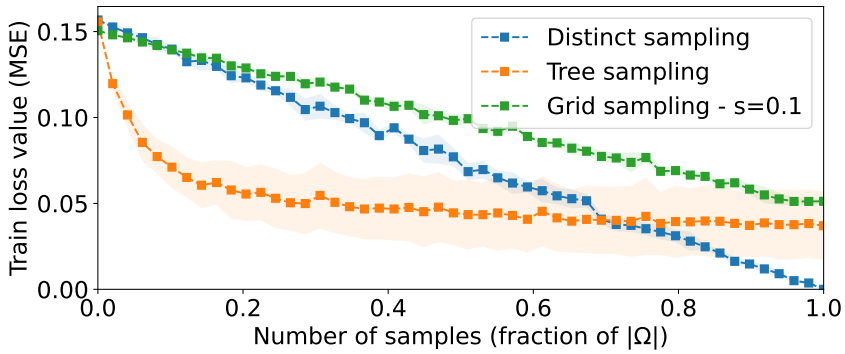


Figure 2.4: RFF performance for $L = 5, d = 4$, to approximate random VQCs with Pauli encoding.

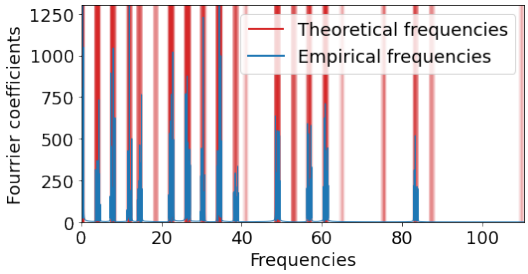
encoding).

2.4.1.1 Pauli encoding

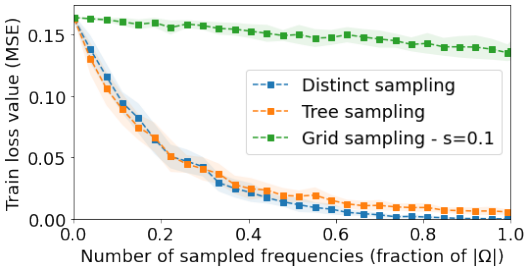
I first consider a quantum model with L Pauli encoding gates per feature resulting in an integer-frequency spectrum (half of $[\![-L, L]\!]^d$). In this case, the corresponding quantum model is a periodic function of period $T = (2\pi)^d$ and thus, we choose $x_{max} = 2\pi$ for X_{grid} construction.

In Figure 2.3, I implement a VQC with L=200 Pauli encoding gates, for a 1-dimensional input. One can observe that our classical approximation methods are indeed able to reproduce such VQCs. On average, the RFF training error for *Distinct* and *Grid* sampling is a linear function of the number D of samples taken from Ω. On the other hand, the error using *Tree* sampling exhibits a faster decreasing trend, reaching relatively low errors with only 20% of the spectrum size.

I conjecture that the efficiency of *Tree* sampling is closely related to the redundancy in the discrete frequency distribution over Ω. In fact, as shown in Figure 2.3, Fourier coefficients of the VQC are, on average, correlated to the frequency redundancy in the empirical quantum spectrum. Frequencies above a certain threshold $\omega_{\text{effective}}$ are merely redundant for this particular encoding scheme, and one observes that they are cut from the quantum model empirical spectrum. The effective spectrum of the VQC is therefore smaller than what the theory predicts. Consequently, the fast decreasing trend of the *Tree* sampling stems from the fact that we sample according to the redundancies, therefore requiring less frequency samples. One can see that $0.2 \times |\Omega|$ samples are sufficient to sample approximately all frequencies in $[|0, \omega_{\text{effective}}|]$. In Figure 2.4, I show similar simulations with a d-dimensional input ($d = 4$) and $L = 5$ Pauli gates per dimension. According to Equation (2.5), the theoretical number of distinct positive frequencies is 7321. In this case in the tree sampling procedure, we didn't correct for the fact that one can sample both a frequency ω and its opposite $-\omega$. Therefore the scheme is a bit less performant than in dimension 1.



(a) Fourier transform averaged over different random initialization of the VQC. The intensity of the vertical red lines indicates the concentration of the theoretical frequencies in Ω.



(b) RFF train loss with different sampling methods on the random VQCs. The Distinct sampling benefit from the concentration of frequencies in packets to approximate with less samples.

Figure 2.5: Random 1d VQC with 4 scaled Paulis and and a 3-qubits H_{XYZ} Hamiltonian

2.4.1.2 More elaborate Hamiltonian encodings

For Pauli encoding, we have shown in the previous part that *Tree* sampling is highly effective for approximating the quantum model. Consequently, I designed VQCs with different spectrum distributions to study the RFF approximation performance in these cases.

As explained in Section 1.3, I consider encoding gates of the form $exp(-ix_iH)$ for each dimension i. One way to alter the spectrum distribution is the use of more general Hamiltonians H. I restrict the experiment to physical Hamiltonians (ie involving only two-bodies interactions), and use the generic expression

$$H_{XYZ} = \sum_{\langle i,j \rangle} \alpha_{ij} X_i X_j + \beta_{ij} Y_i Y_j + \gamma_{ij} Z_i Z_j + \sum_i \delta_i P_i \tag{2.20}$$

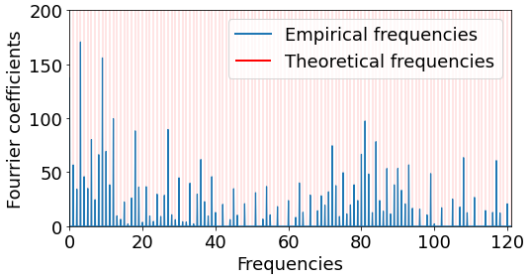
with the first term describing the interactions: $\langle i, j \rangle$ indicates a pair of connected particles and the second term describing a single particle's energy ($P_i = \{X_i, Y_i \text{ or } Z_i\}$).

In Figure 2.5, I construct VQCs mixing both such Hamiltonians[1] and scaled Paulis[2] as encoding gates, on 1-dimensional inputs. In these cases, the corresponding quantum model is no longer 2π-periodic, thus we have to find empirically a good value for x_{max} (by increasing it until the performance reaches a limit).

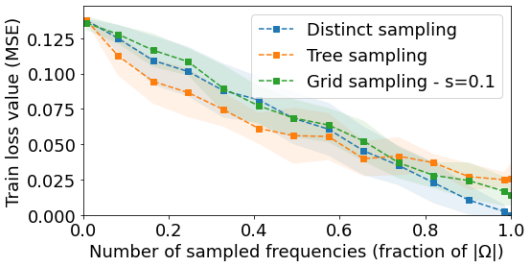
With such complex encoding, we witness a different behavior for the *Distinct* sampling method, in comparison to the previous basic Pauli encoding scenario. Essentially, *Distinct* sampling has a faster than linear scaling, showing a clear and unexpected efficiency of RFFs in this case. We also notice that the *Tree* sampling method has a similar scaling. This observation points to the fact that, with the chosen encoding strategy, the frequencies in the spectrum Ω

[1] in Figure 2.5, we used a 3-qubits Hamiltonian defined by: $H_{XYZ} = 7X_0X_1 + 7X_1X_0 + 0.11X_0X_2 + 0.1X_2X_0 + 8[Y_1Y_2 + Y_2Y_1 + Z_0Z_2 + Z_2Z_0]$

[2] Scaling factors are $[26.4309, 34.4309, 22.4309, 0.4309]$



(a) Fourier transform averaged over different random initializations of the exponential encoding VQC with $L = 5$.



(b) RFF approximation performance of an the exponential encoding VQC with $L = 5$.

Figure 2.6: Random VQCs with exponentially large spectrum, using scaled Pauli encoding as in (STJ22).

are concentrated in many packets or groups. This behavior is displayed with the concentrated red lines in Figure 2.5. Therefore, even though the frequencies in Ω have a low redundancy (545 distincts frequencies out of 2017), sampling just one of the many frequencies in a narrow packet is enough for the RFF to approximate it all. To put it diffrently, we can consider that there is a $\Omega_{\text{effective}}$ where each packet can be replaced by its main frequency, and RFF manages to approximate it with fewer samples than the actual size of Ω. To conclude, many distinct frequencies is not a guarantee of high expressivity.

As for *Grid* sampling, the choice of s seemed too high for this solution to work in this case, in line with the theoretical bounds for this sampling method given in Theorem 2.3.

2.4.1.3 Exponential Pauli encoding

In order to obtain VQCs with a large number of frequencies, but low redundancy and no concentrated packets, we exploit the exponential encoding scheme proposed in (STJ22), resulting in a non degenerate quantum spectrum with zero redundancies and thus a uniform probability distribution over integers. In this encoding strategy, encoding Pauli gates are enhanced with a scaling coefficient β_{kl} for the l^{th} Pauli rotation gate encoding the component x_k. This gives us a total of 3^{Ld} positive and negative frequencies. These frequencies can be all distinct with the particular choice of $\beta_{kl} = 3^{l-1}$, resulting in an exponentially large and uniform Ω. Note however that Ω is analytically known and contains only integer frequencies, mostly very high frequencies for which the usefulness in practice remain to be studied.

The RFF strategies have been tested, shown in Figure 2.6, and we obtained again the confirmation that RFF can approximate such an exponential feature space with a fraction of $|\Omega|$. This fraction might however be too large in practice. We also observe as expected that all three strategies have a linear scaling, in line with the absence of redundancy and frequency packets.

With these experiments, we conclude a few important properties. One can observe that

when some frequencies in the spectrum Ω have much redundancy (*e.g.* Pauli encoding), these frequencies are empirically the ones with higher coefficients. In such case, the *Tree* sampling strategy is able to approximate the VQC's model with fewer samples than the other methods as expected. With more complex Hamiltonians, concentrated packets of frequencies appear, and even without much redundancy, both *Tree* and *Distinct* sampling require fewer frequency samples to cover these packets. According to these experiments, the worst case scenario for the RFF is a uniform probability distribution where all the three sampling techniques will be equivalent. Nonetheless, the theoretical bounds prove that the number of Fourier Features will scale linearly with respect to the spectrum size that scales itself exponentially.

2.4.2 Comparing VQC and RFF on artificial target functions

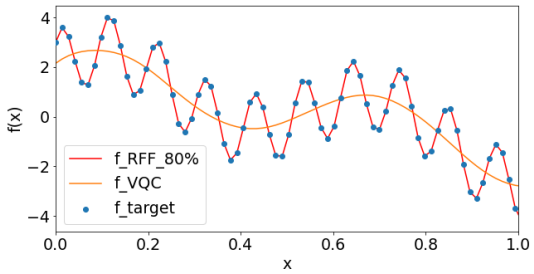
In the above section, a RFF is trained to mimic the output of random VQCs. In practical use cases, the ground truth relies on a classical dataset with an underlying function to find. Therefore a relevant comparison is to measure the efficiency of both VQC and RFF on a common target function. One wants to see when RFF can obtain a similar or better results than the VQC on the same task.

We have seen that for VQCs with Pauli encoding, the Fourier coefficients are rapidly decreasing, cutting out frequencies higher than $\omega_{effective}$ from the empirical spectrum. For this reason, I have chosen a particular synthetic target function: I create a sparse Fourier series (i.e. having only few non-zero coefficients) as a target function: $s(x) = \sum_{\omega \in \{4,10,60\}} cos(\omega x) + sin(\omega x)$ and a VQC with $L = 200$ Pauli encoding gates as the quantum model.

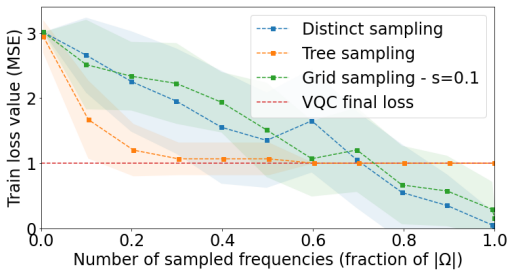
In Figure 2.7, it can be observed that the VQC, as well as RFF with *Tree* sampling, can not learn the frequency $\omega = 60 > \omega_{\text{effective}}$ (their train loss reach a high limit) while the RFF models based on *Distinct* and *Grid* sampling can effectively learn the target function with enough frequency samples. This result shows that even when a VQC with Pauli encoding is trained, it cannot reach all of its theoretical spectrum, thus questioning the expressivity of such a quantum model. On the other hand, its classical RFF approximators (*Distinct* and *Grid*) succeed in learning the function in this case. This is due to the specific choice of the frequencies in s and of the VQC's structure, which has a high redundancy in its spectrum. Indeed, the VQC is not able in practice to obtain non-negligible coefficients for frequencies higher than $\omega_{\text{effective}}$. This limit appears similarly for the *Tree Sampling* RFF, but not for the two other types. Of course, when one choses an artificial function for which all frequencies are below $\omega_{\text{effective}}$, the VQC and all RFF methods manage to fit it.

2.4.3 Comparing VQC and RFF on real datasets

In order to compare the learning performances of a VQC and its corresponding RFF approximator on real-world data, we choose the fashion-MNIST dataset (XRV17) and consider a binary image classification task(*coat* and *dress*). We pre-process the input images by performing a

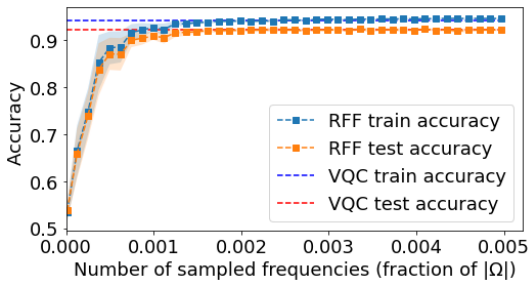


(a) Predictions of the target function $s(x)$ with the quantum model and its corresponding RFF approximator using *Distinct* sampling on 80% of all frequencies.

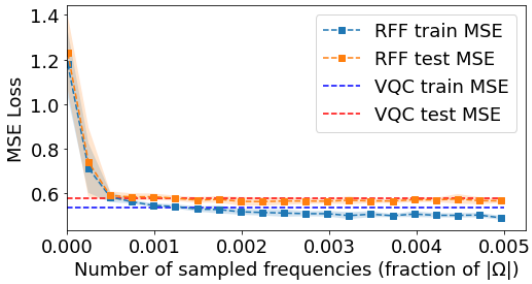


(b) RFF learning curves for sparse target fitting based on the VQC description. Distinct and Grid sampling are able to outperform the VQC.

Figure 2.7: **Fitting a target function** $s(x) = \sum_{\omega \in \{4,10,60\}} cos(\omega x) + sin(\omega x)$ with a VQC architecture of $L = 200$ Pauli gates.



(a) Fashion-MNIST dataset (classification)



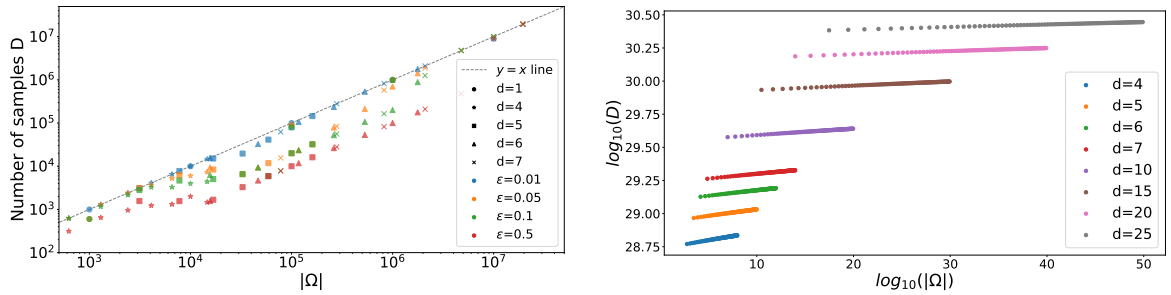
(b) California Housing dataset (regression)

Figure 2.8: **On real datasets.** Prediction results of a VQC and the its classical RFF approximator (Tree Sampling) on two 5-dimensional real datasets. A very low number of frequency samples is necessary to obtain similar results.

principal component analysis keeping the first 5 features (therefore $d = 5$) and by re-scaling the new input features between $-\pi$ and π. We also use the California Housing dataset for a regression task with 5 features and a re-scaling between 0 and π.

We chose to solve these two problems by training VQCs with Pauli encoding ($L = 5$ for each dimension). According to Equation (2.5), the number of distinct positive frequencies in Ω is 80526. In Figure 2.8, we observe that, with very few frequencies sampled, the RFF model with Tree sampling succeeded as well in learning the distribution of the input datasets. We conclude that this RFF method allows to closely approximate the trained quantum models. We observe that for the two tasks, with a radically lower number of frequencies ($0.002 \times |\Omega| \simeq 160$), the RFF accuracy/MSE loss performance mimics with fidelity the VQC performance even when more frequencies are used.

This result could indicate that the underlying distributions to learn were simple enough, such that the VQC had an excessive expressivity. In this case, and despite the large size of Ω, the RFF manage to find similar solutions more quickly.



(a) Experimental bounds for different values of L, d.

(b) Theoretical bound for different values of L, d.

Figure 2.9: **Evolution of D as a function of input dimension d and of L encoding gates per dimension, and theoretical bounds.** The number of samples D given as a fraction of $|\Omega|$ decreases with the growth of the data input dimension and the number of encoding gates. The regime where the experiments were performed is however limited to see the dependency of $D \sim \log |\Omega|$ according to the theoretical bounds.

2.4.4 Numerical test of the theoretical bounds

In this section, we test the theoretical bound provided by the Theorem 2.3. Given a spectrum $\Omega = [\![0, L]\!]^d$, a Fourier series model trained on a specific dataset, the theorem bounds the necessary number of samples for a RFF model to approximate the original model with an ϵ error. This is an approximation to the Pauli encoding VQCs where the spectrum is $\Omega = [\![-L, L]\!]^d$. For fixed values of L, d, and a spectrum $\Omega = [\![0, L]\!]^d$, we implement the following protocol to test this bound:

- Generate a dataset of 10^5 points sampled uniformly from $[0, 1]^d$ and labels coming from a Fourier series on Ω with coefficients chosen uniformly from $[0, 1/\sqrt{|\Omega|}]$, split into a train set and a test set with respective fractions .9 and .1.
- For each value of D in $\{1, k|\Omega|/10 \text{ for } k \in [\![1, 10]\!]\}$, sample D frequencies from Ω without replacement, and train a linear ridge regression with $\lambda = 10^{-6}$ on the train set. We performed the training with a Adam optimizer, a learning rate of .001, and between 50 and 200 epochs depending on the size of the spectrum with the constraint of computation time. Compute the output on the test set.
- Compute the mean absolute error between the output of the trained model with all the frequencies and the output of all other model. Select the model with the lowest number of samples that has an error below ϵ.

The results of the application of this protocol are shown Figure 2.9. For the values of $|\Omega|$ between 10^3 and 10^6, one can see a reduction (more or less pronounced depending on the experiment setting) of the number of samples needed to approximate the whole model. For

$\epsilon = .05$, one can expect to need only half of the spectrum, whereas for $\epsilon = .5$, one only need about 10% of the spectrum. The trend does not continue above $|\Omega| = 10^7$.

There are several limitations to this experiment. The main one is the limited training of the models. For the biggest values of $|\Omega|$ we limit ourselves to 50 epochs, which may be not enough to reach the optimal parameters, and thus blur the interpretation. Furthermore although the theorem is valid for every number of data points, the overparameterized regime where there are much more parameters than data points is known to exhibit unusual effects in linear regression (HMRT22).

Given the choices of λ and ϵ, the theoretical bounds are very high for the regimes we experimentally tested, so they are not relevant. The effect that is quantified by the theory appears from $|\Omega| = 10^{30}$, e.g one need approximately $D = 10^{30}$ samples to approximate 10^{40} frequency which is still unfeasible on a classical computer. ignificant

2.5 Conclusion

In this work, we have studied the potential expressivity advantage of Variational Quantum Circuits (VQCs) for machine learning tasks, by providing novel classical methods to approximate VQCs. Our three methods use the fact that sampling few random frequencies from a large dimensional kernel (exponentially large in the case of VQCs) can be enough to provide a good approximation. This can be done given only the description of the VQCs and does not require running it on a quantum computer. We studied in depth the number of samples and its dependence on key aspects of the VQCs (input dimension, encoding Hamiltonians, circuit depth, VQC spectrum, number of training points). On the theoretical side, we conclude that our classical sampling method can approximate VQCs for machine learning tasks on Hamiltonian encoding of classical data, with a number of samples that scales favorably but with potentially large constant overheads. Experimentally, we have tested our classical approximators on several use cases, using both artificial and real datasets, and our classical methods were able to match or exceed VQC results. By providing a new way of comparing classical and quantum models, these results may help to understand where the true power of VQCs comes from, and define more rigorously quantum advantage in this context. In particular, it opens up questions about alternative encoding schemes, harnessing the effective expressivity of VQCs, and the link between a full expressivity and trainability.

Chapter 3

Learning beyond minimum norm least square

It is known that variational quantum circuits are linear models in some *feature space* of finite dimension (see Chapter 1). Optimizing the parameters of a quantum circuit amounts then to a specific way of searching in the space of linear models for a given feature map (Sch21). At first sight, if this feature map can be explicitly computed classically, one may wonder what is the interest of searching the best parameters of the quantum circuit instead of performing classically a linear regression on the same feature map, using a so called *classical surrogate* model (SEM23). Even when the feature space is too large to be computed classically, Chapter 2 introduced a method to reduce its dimension by random sampling, realizing approximated classical models. A refined analysis of this technique has been also made in (SRJ+23). However, the sampling technique must have some limitations because it has been proven that quantum circuits can still offer advantages in learning tasks with examples related to cryptography (JFPN+23; GD23; LAT21).

In this chapter, I present necessary conditions for a quantum model to avoid such dequantization, and I highlight that this could be only satisfied for high dimensional feature maps. I propose conditions that guarantee a quantum model to remain *far* from its equivalent classical model. For that, I use the important fact that classical linear regression tends to make the optimized weight vector converge towards a specific solution called *minimum norm least square* (MNLS). The study focuses on showing when the quantum weight vector doesn't possess the same bias. Figure 3.1 summarizes the methodology.

Those conditions are then analyzed for several usual frameworks and architectures, showing that the methodology can be seen as a new tool for one to rule out some quantum circuits of

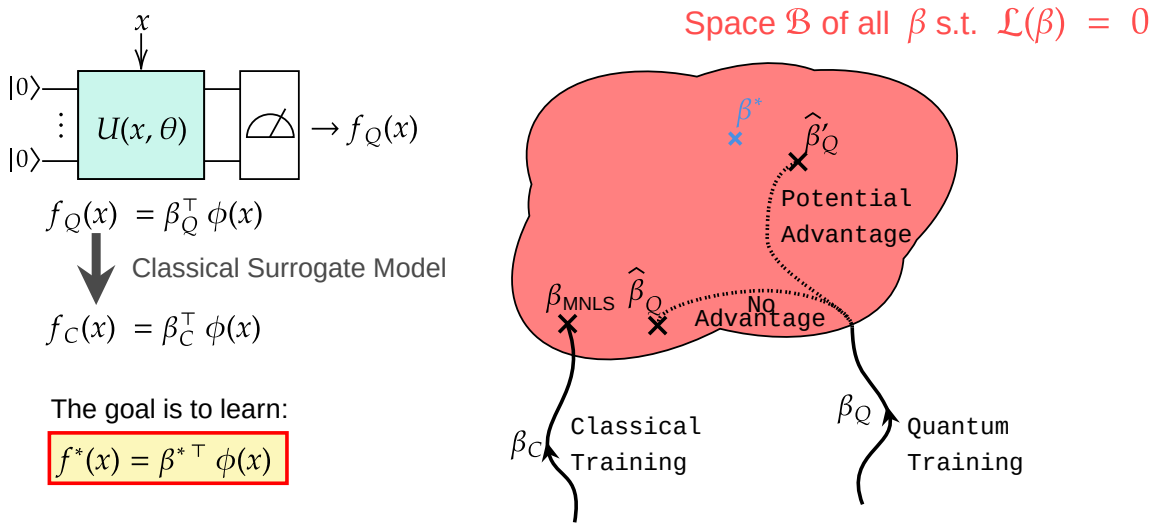


Figure 3.1: A quantum model $f_Q(x) = \beta_Q(\theta) \cdot \phi(x)$ is trained by optimizing its weight vector $\beta_Q(\theta)$. If one can train a surrogate model on a classical computer, using the same (or approximated) feature map $\phi(x)$, it would constitute an obstacle to quantum advantage. It has been shown that during classical linear regression, the weight vector converges towards a specific point β_{MNLS} called the *minimum norm least squares* estimator. Ensuring that the quantum weight vector β_Q converges far from β_{MNLS} is therefore a necessary condition to avoid such dequantization.

their choice. Using Weingarten calculus, I show that some proposed quantum models can be far from the MNLS.

In addition, I study the link between these dequantization schemes and concentration, another crucial issue of quantum circuits. I prove that there should exist models that avoids both of these problems. In Section 3.1, I introduce the setup and define the quantum models their classical surrogates. In addition, I discuss the connection between model distances and their weight vector norms.

In Section 3.2, I highlight results from classical learning theory about the convergence of linear models towards the minimum norm least square estimator (MNLS). I also introduce the approximated classical models with random feature sampling, and show the link between the number of random features and the norm of the weight vector. Then, in Section 3.3, I point out that a quantum model will not necessarily converge to the same solution as its classical surrogate (MNLS). As specific examples, we analyze two classes of VQCs and see if their quantum models can be far from their classical surrogates. Finally in Section 3.4, I address the question on whether it is possible to construct a quantum model that avoids dequantization and that is not concentrated.

This chapter is based on the following preprint:

- (TML24) Slimane Thabet, Léo Monbroussou, Eliott Z Mamon, Jonas Landman (2024). *When Quantum and Classical Models Disagree: Learning Beyond Minimum Norm Least Square arXiv:2411.04940*

The most common variables and notations of this chapter are summarized in Table 3.1.

Notation	Object
x	Input vector
M	Size of the dataset
d	Input dimension
$\mathbf{\Phi}$	Data matrix of shape (M, p)
$\phi(x)$	Feature map
$\boldsymbol{y}$	Vector of targets of shape $(M, 1)$
p	Given context: dimension of $\phi(x)$, size of Ω
D	Number of random features
$f_Q(x;\theta), \beta_Q$	Quantum model and its weight vector
Ω	Spectrum of the quantum model
θ	Parameters of the Quantum circuit
Ω^*	Spectrum of the quantum model without 0
f^*, β^*	Target function and its weight vector
Ω_+	Subset of Ω^* such that $\forall\omega \in \Omega_+, -\omega \in \Omega\backslash\Omega_+$
$f_{\text{MNLS}}, \beta_{\text{MNLS}}$	Minimum norm estimator and its weight vector
n	Number of qubits
K	Kernel matrix
N	$N = 2^n$, size of the Hilbert space

Table 3.1: Main notations used in this chapter.

3.1 VQC in arbitrary basis and surrogate models

In this section I define the setup of the analysis. In Chapter 1 I explained that VQCs can be considered as linear models in the Fourier basis, here I generalize these notions by considering a data input with any preprocessing function, and show that quantum models can be expressed as linear models in a feature map

3.1.1 Setup

We consider a learning problem with an input vector x distributed in $\mathcal{X} \subseteq \mathbb{R}^d$, a feature map $\phi : \mathbb{R}^d \longrightarrow \mathbb{R}^p$, and a target function expressed as

$$f^*(x) = \beta^{*\top}\phi(x)\,, \tag{3.1}$$

where $\beta^* \in \mathbb{R}^p$. Furthermore we assume $\|\phi(x)\| \leq 1$ which is a very usual assumption. One needs to consider this assumption to ensure that the KRR solution is defined (BN06; SHS01).

If the upper bound of the feature map is greater than 1, one can always renormalize it. We are given a dataset $\{(x_i, y_i) = (x_i,\ f^*(x_i))\}_{i=1}^M$ where x_is are iid given a distribution $\mu(x)$. We also assume that $||f^*||_\infty \leq 1$. This assumption comes from the fact that expectations of Pauli strings on quantum circuits are bounded by 1, so the target function has to respect this property. Note that f^* can always be rescaled.

The goal of the learning task is to train $f(x) = \beta^\top \phi(x)$ with β the *weight vector* to be optimized. To quantify the error between an estimator f and the true solution f^*, various distances between functions can be considered. We will consider the square of the L_2 distance (with respect to a distribution μ on the input space)

$$\|f - g\|_\mu^2 := \int_{\mathcal{X}} (f(x) - g(x))^2 d\mu(x)\,, \tag{3.2}$$

which in the case of a uniform distribution on the training dataset $\{x_i\}_{i=1}^M$ takes the form of:

$$\mathcal{L}(f, g) := \frac{1}{M} \sum_{i=1}^{M} (f(x_i) - g(x_i))^2\,. \tag{3.3}$$

Lastly, for bounded functions f, g, we may also consider their ∞-distance:

$$\|f - g\|_\infty := \sup_{x \in \mathcal{X}} |f(x) - g(x)|\,. \tag{3.4}$$

$\|f - f^*\|_\mu^2$ represents the *true risk*, while $\mathcal{L}(f, f^*)$ represents the *empirical risk*. In the *over-parametrized* case ($p > M$), note that many functions f can have an empirical risk equal to zero without minimizing the true loss, as the empirical risk concerns only the M data points from the training set. A model with low true risk implies better *generalization* properties.

3.1.2 Quantum models

We consider a quantum circuit corresponding to a n-qubit unitary matrix $U(x, \theta)$ with $x \in \mathbb{R}^d$ the input vector and θ the set of trainable parameters, and O the observable matrix corresponding to the final measurement. We define the *quantum model* as the parameterized function $f : \mathcal{X} \times \Theta \longrightarrow \mathbb{R}$ that can be expressed as:

$$f_Q(x; \theta) = \langle 0 |\, U(x; \theta) O U(x; \theta)\, | 0 \rangle\ . \tag{3.5}$$

We consider unitary matrices composed of successions of layers of the form

$$U(x; \theta) = \Big[\prod_{l=1}^{L} V^l(\theta) S^l(x) \Big] V^0(\theta)\,, \tag{3.6}$$

where the V^ls are trainable unitary matrices depending on θ (for the rest of the paper, the dependency on θ will be dropped when there is no ambiguity. $S^l(x)$ are encoding gates, and we assume that they are of the form

$$S^l(x) = \exp\Big(-i g^l(x) H^l\Big)\,, \tag{3.7}$$

where H_l is the *encoding Hamiltonian* of the layer l and $g_l : \mathbb{R}^d \longrightarrow \mathbb{R}$ is a *preprocessing function*. Both H_l and g_l are independent of θ.

We propose similar computations as (SSM21a) in the case of general preprocessing functions. It generalizes the above Fourier basis into any arbitrary basis. Given an encoding gate $S^l(x) = \exp\Big(-ig^l(x)H^l\Big)$, one can decompose the generator Hamiltonian as $H^l = P^{l\dagger}\Lambda^l P^l$ such that $S^l(x) = P^{l\dagger}\exp\Big(-ig^l(x)\Lambda^l\Big)P^l$. Without loss of generality, the unitaries P^l and $P^{l\dagger}$ can be absorbed into the trainable unitaries.

The component i of the state vector $U(x;\theta)|0\rangle^{\otimes n}$ can be written as

$$[U(x;\theta)|0\rangle^{\otimes n}]_i = \sum_{j_1\dots j_L=1}^{2^n} \exp\Big(-i\sum_{l=1}^{L}\lambda_{j_1}^{(l)}g^l(x)\Big)W^L_{ij_l}\dots W^1_{j_2j_1}W^0_{j_11}\,, \tag{3.8}$$

where $\lambda_{j_i}^{(l)}$ is the j_i-th eigenvalue of H^l, or the j_i-th coefficient in the diagonal of Λ^l. By taking into account the observable, one can write

$$f_Q(x) = \sum_{\boldsymbol{j},\boldsymbol{j'}} \exp\Big(-i\Big(\sum_l \lambda_{j_l}^{(l)}g_l(x) - \sum_l \lambda_{j'_l}^{(l)}g_l(x)\Big)\Big)\boldsymbol{a_j}\boldsymbol{a_{j'}}\,, \tag{3.9}$$

$$f_Q(x) = \sum_{\omega\in\Omega} c_\omega\, e^{-i\varphi(x;\,\omega)} \tag{3.10}$$

$$= \sum_{\omega\in\Omega} (\beta_Q)_\omega\, \phi(x;\,\omega) \tag{3.11}$$

$$= \beta_Q^\top\, \phi(x), \tag{3.12}$$

$$\tag{3.13}$$

where $\boldsymbol{j} = (j_1,\dots j_L)$ is a L uplet of integers whose entries go from 1 to 2^n, and $\boldsymbol{a_j}$ is a coefficient that can be expressed solely with the trainable unitaries. In the case where there are arbitrary preprocessing functions, one is able to express the output of the quantum model as a linear combination of basis functions, different from the Fourier basis, but still individually computable. The quantum model can be expressed as a linear function in a feature map $\beta_Q^\top\,\phi(x)$.

3.1.3 Classical surrogate models

Given a quantum circuit, one can design a *surrogate* model by considering a classical model with the same feature map ϕ, defining a new linear model:

$$f_C(x) = \beta_C^\top\,\phi(x)\,, \tag{3.14}$$

with β_C a weight vector that is explicitly optimized using gradient descent or kernel ridge regression as explained in Subsection 3.2.1. Even if the feature maps is too large for classical memory, approximation techniques have been presented to reduce the dimensionality of the feature map. We analyze these techniques in Subsection 3.2.3, and studies such as (SRJ+23) have shown that quantum models can sometimes be dequantized using random features regression

techniques, with limitations exist for resource-constrained circuits. The sampling technique must have some limitations because it has been proven that quantum circuits can still offer advantages in learning tasks with examples related to cryptography (JFPN+23; GD23; LAT21).

3.2 Bias of classical models and random feature regression

In this section, I study the bias of the classical model training that makes the quantum weight vector converges to a unique solution called the Mean Norm Least Square. Then, I study Random Feature regression (generalized from Chapter 2), a randomization technique used to lower computational costs for kernel methods. I show that this technique is adapted to construct surrogate models.

3.2.1 Classical methods : gradient descent

The linear regression and kernel ridge regression were introduced in Section 1.2. I remind in this section the important elements, while adding supplementary information relevant to this chapter. Classical learning is done by minimizing the mean square error over the training set (BN06). One wants to converge to the vector β^*, and obtain $\hat{\beta}$ that minimizes

$$\mathcal{L}(\beta) = \frac{1}{M}\sum_{i=1}^{M}(\phi(x_i)^\top\beta - y_i)^2 = \frac{1}{M}\|\Phi\beta - \boldsymbol{y}\|^2\,, \tag{3.15}$$

with ϕ the feature map, Φ the data matrix, and $\boldsymbol{y}$ the target vector of size M. We will call finding such a $\hat{\beta}$ the *least square problem.*

In order to prevent overfitting, one can add a regularization term to the loss, and minimize

$$\mathcal{L}_\lambda(\beta) = \frac{1}{M}\|\Phi\beta - \boldsymbol{y}\|^2 + \lambda\|\beta\|^2\,. \tag{3.16}$$

We will call this problem the *regularized least square problem.*

The optimization is usually done with gradient descent (GD) or stochastic gradient descent (SGD). In the following section, we will look at the properties of the solution given by gradient descent.

If $p < M$, the solution to the least square problem is unique, and can be expressed as

$$\hat{\beta} = (\Phi^\top\Phi)^{-1}\Phi^\top\boldsymbol{y}\,. \tag{3.17}$$

And if there is no noise in the target data, and the samples are adequately chosen (ie $(\Phi^\top\Phi)$ is invertible), we have $\hat{\beta} = \beta^*$

If $p > M$, there exists an infinite number of solutions such that $\mathcal{L}(\beta) = 0$. The set of solutions forms an affine space of dimension $M - p$.

The linear regression problem can also be related to kernels. A kernel function is a function $k : \mathbb{R}^d \times \mathbb{R}^d \longrightarrow \mathbb{R}$ symmetric positive definite, representing an inner product in a potentially infinite dimensional Hilbert space, called the Reproducing Kernel Hilbert Space (RKHS).

The kernel ridge regression is the equivalent of a linear regression in the RKHS (BN06; HSS08). The goal is to construct a linear model $f(x) = \sum_{i=1}^{N} \alpha_i k(x, x_i)$ where $x_i \in \mathbb{R}^d$. Given a dataset of size M, the representer theorem (SHS01) states that the minimizer of the empirical loss

$$\mathcal{L}(f) = \sum_{i=1}^{M} (f(x_i) - y_i)^2 + \lambda \|f\|^2 , \tag{3.18}$$

can be written as $\hat{f} = \sum_{i=1}^{M} \alpha_i k(x, x_i)$.

A search in a high dimensional space has become a search in a M dimensional space, since we are guaranteed that the optimal solution can be described by the α_i coefficients.

Given a feature map ϕ, we can define a kernel $k(x, y) = \phi(x)^\top \phi(y)$. Solving the linear regression with ϕ is equivalent of solving the KRR with k. We can define a dual problem to the least square problem by defining $\alpha \in \mathbb{R}^M$ such that $\beta = \Phi^T \alpha$. Solving the least square problem on β is equivalent to solving it on α, and we have that the optimal α is given by

$$\hat{\alpha} = (\Phi\Phi^\top)^{-1} \boldsymbol{y} = K^{-1} \boldsymbol{y} . \tag{3.19}$$

where K is the kernel matrix, and then $\hat{\beta} = \Phi^\top (\Phi\Phi^\top)^{-1} \boldsymbol{y}$. This expressison is correctly defined only in the case where $p > M$. The obtained $\hat{\beta}$ is the same $\hat{\beta}$ obtained by GD. This formulation is helpful because it will help derive bounds on the norm of $\hat{\beta}$ in the following of the work.

3.2.2 Bias of classical models

I just described the classical linear regression and kernel ridge regression. In this subsection, I detail the known results (BN06; HMRT22) about the solution of these methods. I distinguish two regimes:

- The **underparameterized** regime where the number of features is lower or equal to the number of datapoints: $p \leq M$.
- The **overparameterized** regime where the number of features is greater than the number of datapoints: $p > M$.

In the underparameterized regime, the solution to the least square problem is unique, and can be expressed as

$$\hat{\beta} = (\Phi^\top \Phi)^{-1} \Phi^\top \boldsymbol{y} . \tag{3.20}$$

And if there isn't noise in the target data, and the samples are adequately chosen, we have $\hat{\beta} = (\Phi^\top \Phi)^{-1} \Phi^\top \boldsymbol{y} = (\Phi^\top \Phi)^{-1} \Phi^\top \Phi \, \beta^* = \beta^*$.

In the overparameterized regime, there exists an infinite number of solutions such that $\mathcal{L}(\beta) = 0$. The set of solutions forms an affine space of dimension $M - p$. However, the algorithms of GD and KRR will converge towards a specific solution β_{MNLS} called *minimum norm least*

square estimator (MNLS), which is the unique solution that minimizes the empirical loss with a minimal norm:

$$\beta_{\text{MNLS}} = \arg\min \|\beta\|_2 \text{ such that } \mathcal{L}(\beta) = 0\,. \tag{3.21}$$

This is formalized in the following theorem

Theorem 1 (From (HMRT22))**.** *Let $\beta_0 = 0$ the initialization of a gradient descent algorithm. Let the following iterations be defined by*

$$\beta_{k+1} = \beta_k + \gamma\, \Phi^\top(\boldsymbol{y} - \Phi\beta_k)\,, \tag{3.22}$$

with γ the learning rate such that $0 \leq \gamma \leq 1/\lambda_{\max}(\Phi^\top\Phi)$ where $\lambda_{\max}(\Phi^\top\Phi)$ is the largest eigenvalue of $\Phi^\top\Phi$. Then

- *β_k converges towards the minimum norm least square estimator β_{MNLS} defined in Equation (3.21).*
- *$\beta_{MNLS} = \Phi^\top(\Phi\Phi^\top)^{-1}\boldsymbol{y}$.*

For completeness, I give a proof in Appendix A.2. The key reason for such a property is that when performing GD, each iterate of β stays in the subspace $V = \text{span}(\phi(x_1), \dots \phi(x_M))$, the *row space* of the data. Each vector of $\mathbb{R}^p$ can be decomposed as $\beta = \beta_V + \beta_{V^\perp}$ where $\beta_{V^\perp} \in V^\perp$, the orthogonal of V in $\mathbb{R}^p$, such that $\Phi\beta_{V^\perp} = 0$.

Each vector $\hat{\beta}$ such that $\mathcal{L}(\hat{\beta}) = 0$ can then be written

$$\hat{\beta} = \beta_{\text{MNLS}} + u\,, \tag{3.23}$$

where $u \in V^\perp$, and $\Phi\beta_{\text{MNLS}} = \boldsymbol{y}$, $\Phi u = 0$, $\|\hat{\beta}\|^2 = \|\beta_{\text{MNLS}}\|^2 + \|u\|^2$. Since the iterates of GD stay in V, then $u = 0$ all along.

The kernel matrix is defined as:

$$[K]_{i,j} = \phi(x_i)^\top\phi(x_j) = k(x_i, x_j)\,. \tag{3.24}$$

We also define the data matrix Φ and the target vector $\boldsymbol{y}$:

$$\Phi = \begin{bmatrix} \phi(x_1)^\top \\ \vdots \\ \phi(x_M)^\top \end{bmatrix}, \quad \text{and} \quad \boldsymbol{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_M \end{bmatrix}. \tag{3.25}$$

The weight vector β obtained through the data will be the Minimum Norm Least Square (MNLS) one, defined as:

$$\beta_{\text{MNLS}} = \Phi^\top(K^{-1}\boldsymbol{y})\,. \tag{3.26}$$

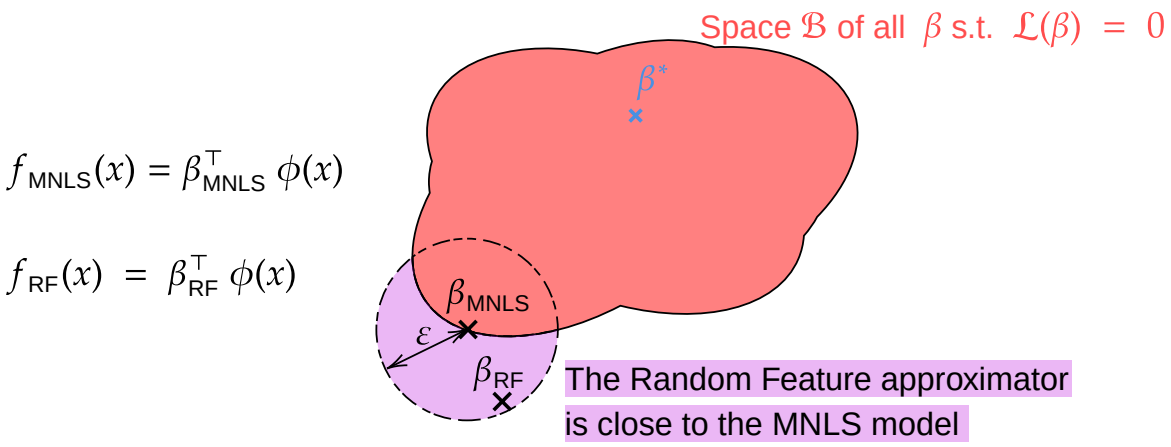


Figure 3.2: The random feature approximator f_{RF} approximates the MNLS f_{MNLS} of a feature map, given suitable behavior of the kernel matrix, which depends on the feature map and the data distribution.

3.2.3 Classical linear regression models can be approximated with randomization

In this section, we show how the above method can be generalized for many other cases in classical linear regression (RR08a; RR08b). It applies therefore to the arbitrary basis quantum models defined in Subsection 3.1.2 . Again, the goal is to reduce the number of features required to approximate the target function. The problem is to learn a function of the form $f(x) = \sum_{i=1}^{p} \beta_i \phi_i(x)$ where one wants to learn the vector of coefficient β. The continuous version of this problem would be to learn a continuous function $\beta(\omega)$ such that $f(x) = \int_{\omega \in \Omega} \alpha(\omega)\phi(x;\omega)$. If the number of basis function is too big to be stored in memory, or infinite in the case of continuous function learning, the problem can become too difficult to solve.

However, it has been shown that this problem can be simplified by learning a function of the form $\hat{f}(x) = \sum_{k=1}^{D} \beta_i \phi_k(x)$ with $D << p$ where the functions $\phi(\,\cdot\,;\omega_k)$ are sampled from $[\![1,\, p]\!]$. In this case, one only has to learn a vector of dimension D.

Theorem 2. *Let $\phi(x) = [\sqrt{q_1}\ \phi_1(x) \dots \sqrt{q_p}\ \phi_p(x)]^{\top}$ where $\phi_i(x)$ are basis functions such that $\forall x,\ |\phi_i(x)| \leq 1$ and $q = (q_1, \dots q_p)$ represents a discrete probability distribution, and let $f(x) = \beta^{\top}\phi(x)$. Let S be a subset of $[\![1, p]\!]$ sampled independently with the probability density q, with $D = |S|$. Then there exists coefficients $c_1, \dots c_D$ such that $\hat{f}(x) = \sum_{k \in S} c_k \phi_k(x)$ satisfies*

$$\|\hat{f} - f\|_{\mu} \leq \frac{\max_i |\beta_i / \sqrt{q_i}|}{\sqrt{D}} (1 + \sqrt{2 \log \frac{1}{\delta}}) \tag{3.27}$$

As a consequence, if one applies the above to β_{MNLS} obtained from a kernel matrix K and target

vector $\boldsymbol{y}$ *such that* $\|\boldsymbol{y}\|_\infty \leq 1$ *there exists coefficients* $c_1, \dots c_D$ *such that*

$$\|\hat{f} - f_{MNLS}\|_\mu \leq \frac{M}{\sqrt{D}\,\lambda_{\min}(K)}(1 + \sqrt{2\log\frac{1}{\delta}})\,. \tag{3.28}$$

We illustrate this in Figure 3.2 and we present the proof of this theorem in Appendix A.3.

Other bounds can be obtained with better scaling in M (SRJ+23; RR17) but involve other quantities like the norm of the kernel operator.

3.2.4 Analysis of the kernel matrix for Fourier models with integer frequencies

Previous results such as Theorem 2 depend on the kernel matrix K. In this section, I consider Fourier models, the case where the frequencies are vectors of integers and the uniform measure in $[0, 2\pi]^d$. As shown in Section 1.4, this case is very important in the quantum machine learning literature (SSM21a; SEM22; SRJ+23; PS22), and could help us understand what quantum circuit design needs to be done in order to do variational circuit learning.

I consider the Fourier feature map:

$$\phi(x) = \frac{1}{\sqrt{p}}\begin{bmatrix}\cos\left(\omega^\top x\right)\\ \sin\left(\omega^\top x\right)\\ \vdots\end{bmatrix}_{\omega\in\Omega_+} \tag{3.29}$$

with $\Omega \subset [-L, L]^d/2$, $L \in \mathbb{N}$, and $p = |\Omega|$. I consider the input vector x to be uniformly distributed in $[0, 2\pi]^d$. The domain $[0, 2\pi]^d$ is enough to consider because the function is periodic, the distribution could be chosen non uniform and it could change the results.

Let $x \in [0, 2\pi]^d$ be the input vector. The kernel is defined as:

$$\begin{aligned} k(x, x') &= \frac{1}{|\Omega|}\sum_{\omega\in\Omega}\cos\left(\omega^\top x\right)\cos\left(\omega^\top x'\right) + \quad \sin\left(\omega^\top x\right)\sin\left(\omega^\top x'\right)\\ &= \frac{1}{|\Omega|}\sum_{\omega\in\Omega}\cos\left(\omega^\top (x - x')\right) \end{aligned} \tag{3.30}$$

Theorem 3. *Let* $(x_1, \dots x_M)$ *uniformly distributed on* $[0, 2\pi]^d$*, and let* K *be the empirical kernel matrix. Then there exists a constant* C *such that*

$$\mathbb{P}(\lambda_{\min}(K) > \frac{1}{2}) \geq 1 - \frac{C}{p}\frac{(M-1)^2p^2}{(p - 4(M-1)^2)^2} = 1 - \frac{C}{p}\frac{(M-1)^2}{1 - 4\frac{(M-1)^2}{p^2}} \tag{3.31}$$

A proof is presented in Appendix A.4. If $p > CM^2$ there is a high probability that the smallest eigenvalue of K is constant. It is the most favorable case to apply random feature regression, from the Theorem 2 it is then enough to have on the order of M^2 random features to approximate the MNLS estimator.

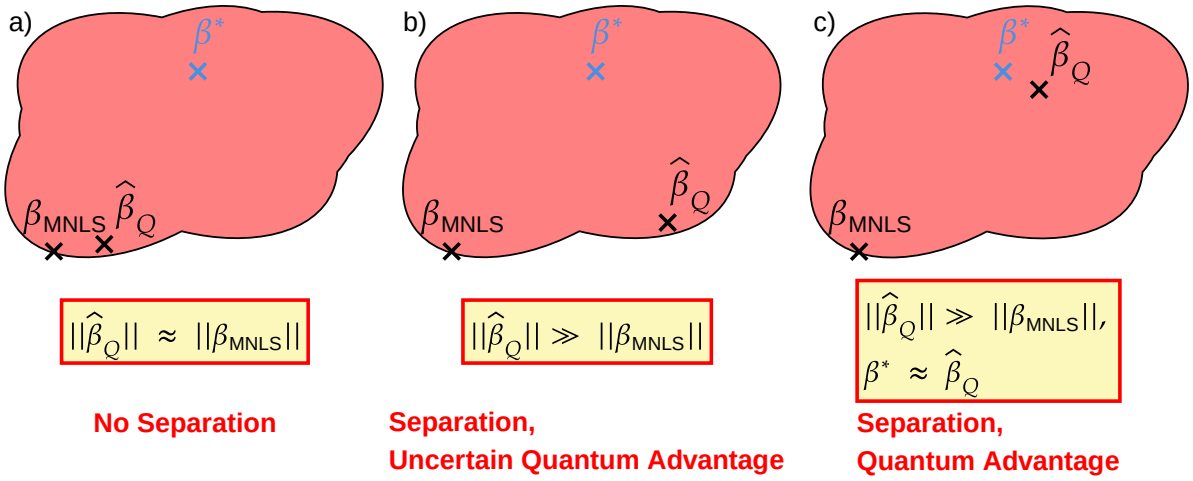


Figure 3.3: Illustration of the potential quantum advantage. If β_Q is close to β_{MNLS} there is no separation between the quantum estimator and the classical one. If β_Q and β_{MNLS} are far from each other and far from the ground truth, there is a separation but uncertain quantum advantage. If β_Q is closer to the ground truth than β_{MNLS}, there is a suggestion of quantum advantage.

3.3 Bias of quantum models and potential advantage

In this section, we discuss about the potential advantage of using parameterized quantum circuits for learning tasks. In our work, we would like to focus on the case where the input variable is continuous. Most proven theoretical results in quantum machine learning come from problems where the input data take discrete values (GD23; MGD24; JFPN+23; LAT21; JGM+24), typically $\{0,1\}^n$. It is convenient because the problems can be linked to cryptography problems which are known or strongly supposed to be hard to solve classically. However many real world use cases utilize continuous vectors, so it is important to have a better understanding in that domain.

First of all, we note that in the underparameterized regime, where the feature space has less dimensions than the number of samples ($p \leq M$), there is little possibility of solving the linear regression problem in a better way with a quantum computer. The optimal solution to the least square problem has indeed a closed form, and if there is no noise in the data, it is equal to the true weight vector. It means that any other optimization technique will converge towards that optimal solution. Moreover, since we assume that the number of data points is small enough to be handled with a classical computer, the total number of operations in the procedure is still polynomial in the size of the dataset. We do not exclude an advantage using a quantum computer to invert the covariance matrix (HHL09), or other more modest polynomial advantages (CVH+22) but it will require fault tolerant quantum computers.

In this Section, we will first explain why having a quantum weight vector norm far from the one of the MNLS is a necessary condition for a potential advantage, as illustrated in Figure 3.3.

Then, we present a class of VQCs that can fulfill this condition in Subsection 3.3.2.

3.3.1 Quantum models can differ from Minimum Norm Least Square

In the setup discussed in this chapter, it makes the most sense to use quantum computers in the overparameterized regime, i.e. $p > M$.

I described in Subsection 3.2.2 the implicit bias of classical learning algorithms. We showed that a classical linear regression trained with gradient descent, or a kernel ridge regression will output a model $f_{\text{MNLS}}(x) = \beta_{\text{MNLS}}^{\top}\phi(x)$ with β_{MNLS} the weight vector of minimum norm which minimizes the training loss. I also showed (See Appendix A.3) that in lots of cases, if one provides a sampling access to the entries of $\phi(x)$, then f_{MNLS} could be classically approximated.

Because VQCs are parametrized in a different way from classical models, they do not necessarily converge towards the MNLS as a result of the training. The weight vector associated to the quantum model may indeed not be contained in the row space of the data, as it is the case for classical linear regression. Contrary to the classical case, one does not have access directly to the coefficients β while tuning a quantum model. One instead optimizes a vector of parameters θ such that $\beta = \beta(\theta)$ and optimizes the loss function $\mathcal{L}(\theta) = \|y - X\beta(\theta)\|^2$. The update rule defined in Equation (3.22) becomes

$$\beta_{k+1} = \beta(\theta_{k+1}) = \beta\Big(\theta_k - t\frac{\partial \mathcal{L}}{\partial \theta}\Big|_{\theta=\theta_k}\Big) \tag{3.32}$$

$$\frac{\partial \mathcal{L}}{\partial \theta} = 2\,\frac{\partial \beta}{\partial \theta}^{\top} X^{\top}(X\beta - y) \tag{3.33}$$

If t is small enough, one can linearize Equation (3.32) and write

$$\beta_{k+1} = \beta_k - t\frac{\partial \beta}{\partial \theta}\Big|_{\theta=\theta_k}^{\top} \frac{\partial \mathcal{L}}{\partial \theta}\Big|_{\theta=\theta_k} \tag{3.34}$$

$$= \beta_k - t\frac{\partial \beta}{\partial \theta}\Big|_{\theta=\theta_k}^{\top} \frac{\partial \beta}{\partial \theta}\Big|_{\theta=\theta_k} 2\,X^{\top}(X\beta - y) \tag{3.35}$$

In the general case, β does not remain in the row space of X therefore it does not necessarily converge towards the minimum least square estimator. This constitutes a crucial potential distinction between quantum and classical models. It remains to see when β_Q can converge far from β_{MNLS} or an approximation of MNLS via Random Features.

Since f_{MNLS} is the interpolating model of minimum norm, any quantum interpolating model f_Q must verify $||\beta_Q|| \geq ||\beta_{\text{MNLS}}||$. A sufficient condition for separation between $||\beta_Q||$ and $||\beta_{\text{MNLS}}||$ would be that $||\beta_Q|| \gg ||\beta_{\text{MNLS}}||$. We state it in the informal theorem

Theorem 4 (Informal)**.** *Let f_Q be an interpolating quantum model, ie $\mathcal{L}(f_Q) = 0$. Therefore f_Q has a potential quantum advantage if $||\beta_Q|| \gg ||\beta_{MNLS}||$.*

In practice, we can consider $\|\beta_Q\| \geq \text{poly}(N)$ in order to have a clear separation. Such a separation would have to be confirmed for a larger class of classical algorithms for training, which

is reserved for future work. In the examples that are developped in section Subsection 3.3.2, we have that p is of the order of N^2 and $||\beta_Q||^2$ is of the order of N. Having a weight vector of large norm will provide a difference with classical models (see Subsection 3.2.3), but a true advantage will be reached if in addition the quantum models is closer to the ground truth than the MNLS. We illustrate these views in Figure 3.3.

If the quantum models would converge to f_{MNLS}, they could be approximated with random feature regression techniques. Therefore we suggest that the best usage of quantum computers would not be to reproduce classical linear regressions. The quantum circuit should be used to provide a model β_Q such that $\beta_Q \neq \beta_{\text{MNLS}}$. This should be possible in principle because the optimization trajectory of the parameters would not lead to converge to β_{MNLS}.

Other works have outlined the differences between quantum and classical linear regression, but none of them mentions the criteria about the norm of the weight vector. The authors in (JFPN+23) study the fact that variational circuits express a different solution than the kernel ridge regression (therefore the MNLS). They point out that there exists functions that are learnable with variational quantum circuits but that require exponentially more resources to learn with quantum kernels. In (YCCW23), the authors analyse the optimization dynamics of QNNs and conclude that they are different from the neural tangent kernel. They study in detail the convergence rate of the respective methods, but do not study the actual solution reached.

3.3.2 Example of quantum Fourier models far from their classical counterparts

In this section, we study an example of quantum circuit generating a Fourier model. We show that the norm of the weight vector reached by this circuit can be much bigger than the norm of the MNLS estimator. We consider a circuit with a specific type of encoding layer $S(x)$ followed by a trainable unitary V and an observable O such that $\text{Tr}(O) = 0$. The circuit outputs the model:

$$f_Q(x) = \text{Tr}[V^\dagger O V S(x)|0\rangle\langle 0|S(x)^\dagger] = \text{Tr}[V^\dagger O V \,|\psi(x)\rangle\,\langle\psi(x)|] = \text{Tr}[V^\dagger O V \rho(x)] \tag{3.36}$$

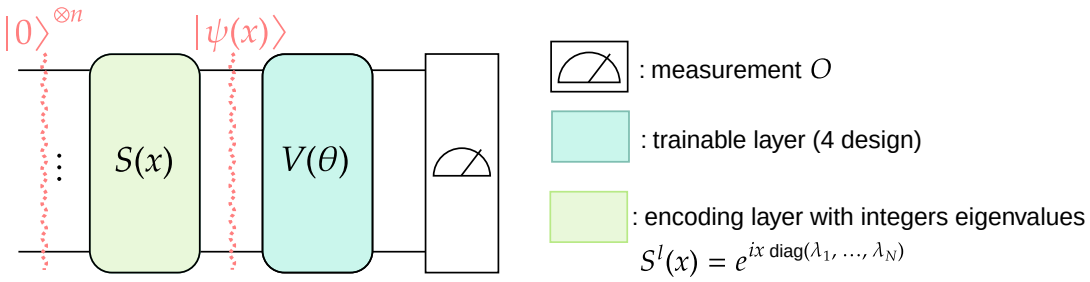


Figure 3.4: Parameterized quantum models considered: hamiltonian encoding with no integer eigenvalues.

Suppose that there exists integer uplets $(\lambda_1, \dots \lambda_N)$ such that:

$$|\psi(x)\rangle = \frac{1}{\sqrt{N}} \begin{pmatrix} e^{i\lambda_1^\top x} \\ \vdots \\ e^{i\lambda_N^\top x} \end{pmatrix} \tag{3.37}$$

with some λ_i that can be equals. This can be achieved with a Hadamard gate followed by a RZ gate on each qubit.

Therefore, after the encoding layer $S(x)$, we have the following density matrix:

$$\rho(x) = |\psi(x)\rangle\langle\psi(x)| = \frac{1}{N} \begin{pmatrix} 1 & \dots & e^{i(\lambda_k - \lambda_\ell)^\top x} & \dots \\ \vdots & \ddots & & \\ e^{-i(\lambda_k - \lambda_\ell)^\top x} & & \ddots & \\ \vdots & & & 1 \end{pmatrix} = \frac{1}{N} \begin{pmatrix} 1 & \dots & e^{i\omega^\top x} & \dots \\ \vdots & \ddots & & \\ e^{-i\omega^\top x} & & \ddots & \\ \vdots & & & 1 \end{pmatrix} \tag{3.38}$$

The frequencies ωs are given by all the pairwise differences between λ_is.

The feature map is given by:

$$\phi(x) = \frac{1}{\sqrt{p}} \begin{pmatrix} \cos\left(\omega^\top x\right) \\ \sin\left(\omega^\top x\right) \end{pmatrix}_{\omega \in \Omega_+} \tag{3.39}$$

We call $R(\omega) = \left|\{(\lambda_i, \lambda_j),\ \lambda_i - \lambda_j = \omega\}\right|$ the number of redundancies of ω distinguishing ω and $-\omega$, and $R(0)$ the number of redundancies of the 0 frequency on the whole matrix, out of the diagonal. For each ω we note the set of indices of which ω appears by $(m_i^{(\omega)}, n_i^{(\omega)})_{i \in R(\omega)}$. We have that

$$2 \sum_{\omega \in \Omega_+^*} R(\omega) + R(0) = N^2 - N \tag{3.40}$$

We can discard the diagonal because its contribution will amount to the trace of O which is 0 by definition.

The quantum model can then be written as

$$f_Q(x;\theta) = \frac{1}{N} \sum_{i=1}^{R(0)} (V^\dagger O V)_{m_j^{(0)}, n_j^{(0)}} + \frac{1}{N} \sum_{\omega \in \Omega_+} \sum_{i=1}^{R(\omega)} (V^\dagger O V)_{m_i^{(\omega)}, n_i^{(\omega)}} e^{i\omega^\top x} + (V^\dagger O V)^*_{m_i^{(\omega)}, n_i^{(\omega)}} e^{-i\omega^\top x} \tag{3.41}$$

We have

$$\begin{aligned} f_Q(x;\theta) = \frac{1}{N} \sum_{i=1}^{R(0)} (V^\dagger O V)_{m_j^{(0)}, n_j^{(0)}} &+ \frac{1}{N} \sum_{\omega \in \Omega} \sum_{i=1}^{R(\omega)} [(V^\dagger O V)_{m_i^{(\omega)}, n_i^{(\omega)}} + i(V^\dagger O V)^*_{m_i^{(\omega)}, n_i^{(\omega)}}] \cos\left(\omega^\top x\right) \\ &+ \frac{1}{N} \sum_{\omega \in \Omega} \sum_{i=1}^{R(\omega)} [(V^\dagger O V)_{m_i^{(\omega)}, n_i^{(\omega)}} - i(V^\dagger O V)^*_{m_i^{(\omega)}, n_i^{(\omega)}}] \sin\left(\omega^\top x\right) \end{aligned} \tag{3.42}$$

We can write the weight vector components as:

$$\beta_{\cos}(\omega) = \frac{\sqrt{p}}{N} \sum_{i=1}^{R(\omega)} [(V^\dagger O V)_{m_i^{(\omega)}, n_i^{(\omega)}} + i(V^\dagger O V)^*_{m_i^{(\omega)}, n_i^{(\omega)}}] \tag{3.43}$$

$$\beta_{\sin}(\omega) = \frac{\sqrt{p}}{N} \sum_{i=1}^{R(\omega)} [(V^\dagger O V)_{m_i^{(\omega)}, n_i^{(\omega)}} - i(V^\dagger O V)^*_{m_i^{(\omega)}, n_i^{(\omega)}}] \tag{3.44}$$

and

$$\beta(0) = \frac{\sqrt{p}}{N} \sum_{j=1}^{R(0)} (V^\dagger O V)_{m_j, n_j} \tag{3.45}$$

Theorem 5. *Let V be drawn from a 2 design. We have that*

$$\mathbb{E}_V[\|\beta_Q\|^2] = \frac{p}{N+1} \tag{3.46}$$

Furthermore, if V is drawn from a 4 design, we have that

$$\mathbb{V}_V[\|\beta_Q\|^2] = \Theta\left(\frac{p^2}{N^6} \sum_{\omega \in \Omega} R(\omega)^2 + \frac{p^2}{N^4}\right) \tag{3.47}$$

We obtain this result by integrating order 4 moments of the Haar measure (CŚ06; Fuk99), the proof is given in Appendix A.5. This result shows that we can have a separation between quantum and classical models. If [1] $R(\omega) = 1$ for all ω, and $p \sim N^2$, then $\mathbb{E}[\|\beta_Q\|^2] = N$ and $\mathbb{V}[\|\beta_Q\|^2] = \Theta(1)$. In Subsection 3.2.2, we have shown that the norm of the MNLS on the same basis scales like $\mathcal{O}(M)$, and thus we have $||\beta_Q|| \gg ||\beta_{\text{MNLS}}||$.

In this example, considering that the trainable unitary is drawn from a 2-design implies the model concentration and vanishing gradient phenomenon called Barren Plateau (HSCC22; MBS+18; LTW+24).

3.4 Discussion

3.4.1 Avoiding concentration issues

In this Section, we present the connection between the model concentration and the existence of a quantum model that is far from the MNLS solution. We prove the existence of such a potential quantum model in Theorem 6.

Concentration phenomenon of parameterized quantum circuits have been studied a lot in the literature. We say that a function is concentrated if the variance is too small. I quantify it by computing the quantity $\mathbb{V}_x[f(x)]$. Typically, for a quantum model we say that f is concentrated

[1] Note that having a dimension of the feature map close to the maximal value implies that the number of redundancies are all close to 1 (see (MMHG+24) for more details).

if $\mathbb{V}[f] \leq \frac{1}{poly(N)}$, with N being the dimension of the Hilbert space so $N = 2^n$. It is equivalent to the Barren Plateau phenomenon (MBS+18), where the gradient of the loss function is exponentially close to 0.

The quantum model f_Q can only be estimated by taking an average of N_{shots} measurements with a precision of $\frac{1}{\sqrt{N_{\text{shots}}}}$. Thus if f is concentrated, it would take an exponential amount of shots to evaluate it reliably. Therefore it would not be useful as a model.

For the Random Fourier features (Subsection 3.2.3), the variance of the function is given by the norm of β, $\mathbb{V}_x[f] = ||\beta||^2/p$. For the weight vectors of the proposed quantum circuits, we have that $\mathbb{V}_x[f]$ is of the order of $1/2^n$ which is concentrated. We would like to find functions such that $||\beta_Q||^2 \geq p$, and I wonder if it can be compatible with the fact that f should be bounded independently of p, ie $|f(x)| \leq 1$ for all x. The fact that f should be bounded comes from the fact that it is the expectation value of an observable.

In the following, I propose a special family of Fourier model such that the norm of the weight vector is large (thus far from MNLS), and that is not concentrated. In addition, I show that this function is bounded. If one could find a quantum circuit architecture that realizes a function from this family, the conditions in Theorem 4 would be satisfied, which would constitute a potential quantum advantage.

Theorem 6. *Let Ω a subset of $[\![-L, L]\!]^d$, where L is an integer. We consider the following function $f : \mathbb{R}^d \longrightarrow \mathbb{R}$*

$$f(x) = \frac{1}{\sqrt{p}} \sum_{\omega \in \Omega} (\beta_{\omega,\cos} \cos\left(\omega^\top x\right) + \beta_{\omega,\sin} \sin\left(\omega^\top x\right)) \tag{3.48}$$

with $p = |\Omega|$, and $\beta_{\omega,\cos}$, $\beta_{\omega,\sin}$ are all iid uniform random variables in the interval $[-\sigma, \sigma]$ with $\sigma = \Theta(1/(d\,(\log d + \log L)))$ We have the following properties:

1. $|\|\beta\|^2 - \frac{2}{3}p\sigma^2| \leq \sigma^2\sqrt{p\log(2/\delta)}$ *with probability at least* $1 - \delta$
2. $\mathbb{V}_x[f(x)] \geq \frac{2}{3}\sigma^2 - \frac{\sigma^2}{\sqrt{p}}\sqrt{\log(2/\delta)}$ *with probability at least* $1 - \delta$
3. $|f(x)| \leq 1 \ \forall x \in \mathbb{R}^d$ *with high probability*

The high probabilities are with respect to the choice of β.

In the above theorem, (1) shows that $\|\beta\|^2$ is of the order of $p\sigma^2$ therefore of potentially higher norm than β_{MNLS} (which scales like M), (2) shows that$f$ is not concentrated, and (3) shows that f is bounded by a constant, which leaves open the amenability to realize it as an quantum expectation value of an observable O with $\|O\|_\infty$ bounded by a constant, which is a property of commonplace quantum observables . Finally, thanks to Theorem 3 we know that $\|\beta_{\text{MNLS}}\|$ would be small in this case.

I give a proof and the probability of (3) in Appendix A.6. This Theorem gives a function that seems possible to achieve from a VQC, far from the classical model, and not concentrated.

However, one needs to find a quantum circuit capable of implementing such a function. In 3.3.2, we present an example of a quantum model that is far from the MNLS solution and bounded, but that is concentrated.

3.4.2 Limitations of the analysis

We list some limitations of our framework. Situations where our analysis doesn't apply and give therefore more or less hope for quantum models to avoid dequantization.

Non standard classical linear regression. We focused our analysis on usual classical gradient descent and KRR, giving rise to the bias of converging towards the MNLS estimator. We did not explore other classical learning algorithms that may not converge to the same solution. One could envision another classical model that is equivalent to a linear model with a high norm weight vector. A simple way to do so would be to add an arbitrary component of the null space of the data to the MNLS, by definition the training loss will still be 0. This would constitute another type of dequantization for quantum models.

Discrete data distributions. A lot of the results in this work depend on the feature map chosen and the associated data distribution. In particular if the input data take discrete values (GD23; MGD24; JFPN+23; LAT21; JGM+24), typically in $\{0,1\}^n$, it can greatly affect the values of the average error. In such cases, the analysis has to be refined.

Different feature maps. One could envision a change of the feature map. For example, instead of using ϕ one may use arbitrary projections and create a new feature vector $\phi' = (\alpha_1^\top \phi, \dots \alpha_{p'}^\top \phi)$. Even though both the MNLS associated to ϕ and ϕ' are linear models of ϕ, their characteristics may differ.

Different quantum circuits. We envisioned the model to be composed of a single quantum circuit, we can imagine a succession of quantum circuits interleaved with classical post-processing, in the way of a multi layer neural networks, and then we don't know how the results will hold.

Classically hard feature maps. An intuitive case where one expects to find a quantum advantage would be when the individual components of the feature map are functions that are easy to compute on a quantum computer, but hard to do so on a classical one. Therefore even

trying to train a classical surrogate wouldn't be possible. We can give two examples. Using the discrete logarithm (Sho94), (LAT21) constructs the quantum feature map such that:

$$|\phi(x)\rangle = \frac{1}{2^k} \sum_{i=0}^{2^k-1} |C_i(x)\rangle \tag{3.49}$$

$$C_i(x) = x \cdot g^i \bmod p \tag{3.50}$$

If one had to compute it classically, one could reformulate it as

$$\phi_i(x) = \begin{cases} 1 \text{ if } \exists k,\ x \cdot g^k \bmod p \\ 0 \text{ otherwise} \end{cases} \tag{3.51}$$

So one would need to compute the discrete logarithm of i in order to evaluate $\phi_i(x)$, which is known to be a hard problem classically.

We can also mention feature maps involving the ground state of data dependent hamiltonians. In (UK24), the authors propose a feature map $\phi(x)$ equal to the ground state of the single chain Ising hamiltonian where the longitudinal part depends on the input x

$$H(x) = \sum_{i=1}^{n} Z_i Z_{i-1} + x \sum_{i=1}^{n} X_i + \sum_{i=1}^{n} Z_i \tag{3.52}$$

In the general case where all interaction coefficients depend on x, the feature map is hard to compute classically.

In this work, we will not talk much about this case, we will assume that each element of the feature map can be easily computed classically. We are interested about the inductive bias of the optimization dynamics of the variational circuits.

3.5 Conclusion

In this chapter, I point out that quantum and classical models for the same feature map do not converge towards the same solution. Classical linear regression trained with gradient descent converges towards the minimum norm least square estimator (MNLS), which is also the output of kernel ridge regression (KRR). Such a bias is due to the fact that the gradient of the mean square error loss function is contained in the row space of the data (*ie* it is a linear combination of the data features). Because of the fact that quantum circuits are a special parameterization one does not have the same bias for quantum neural networks.

In the underparametrized regime where there are more training data points M than dimensions p of the feature space ($p < M$), I show that there is little value to be brought by quantum computers, since the MNLS has a close form and is an optimal solution. In the overparametrized regime ($p > M$), there is an infinity of weight vectors that will minimize the training loss, including the MNLS. It could then be that VQCs will converge to another solution than the MNLS, but closer to the ground truth hence resulting in a better model.

It would also not always be useful to use quantum computers in order to look for the MNLS estimator, because this estimator can be approximated with regression on random features, with a number of random features polynomial on the number of data points. The quality of approximation is also influenced by the spectrum of the empirical kernel matrix, which depends on the exact feature map and on the data distribution. So we do not rule out a case where quantum computers could provide an advantage by evaluating the MNLS.

I investigate in greater details the case of the Fourier feature map with integer coefficients. In this case, the condition number of the kernel matrix is constant in high probability whenever $p > M^2$, so it is the most favorable case to apply random features regression. I show examples of quantum circuits that implement the same feature map and whose weight vector has a norm much bigger than the norm of the MNLS. Therefore there could exist a separation between quantum and classical models in this cases. Such a separation would have to be confirmed for a larger class of classical algorithms for training.

Unfortunately, the proposed quantum models are highly concentrated, which makes them unusable in practice. The concentration of a linear function for integer Fourier features with uniform distribution also directly depends on the norm of the weight vector. We asked whether it was possible to simultaneously have a weight vector norm scaling like the number of features and the function to be bounded. I show that it is possible and we exhibit a function with these properties. The quantum circuit realizing these models remains to be found.

I see the following open questions:

- Similarly than classical linear regression, can we prove a inductive bias of variational quantum circuits linked to the optimization dynamics? (YCCW23) started going into this direction.
- Can we analyze the norm of weight vectors for generalized reuploading circuits, for example using the Pauli string representations of the circuits (RFHC23; BBR$^+$24) ?
- Can we investigate other input data distributions?
- Can we generalize the comparisons we made to other classical learning algorithms?

Part II

Quantum algorithms for graph structured data

Chapter 4

Preliminaries

This part is dedicated to explore the use of quantum computers to perform machine learning tasks on graph structured data. The protocols that are developed are especially relevant for neutral atoms quantum hardware. In this chapter, I introduce the elements needed to understand the content of this part. I give a review of the important concepts of classical machine learning for graph based data, and I explain neutral atoms quantum computers. In Section 4.1, I give concrete examples of machine learning on graph structured data and I define general concepts. Therefore, I detail important families of algorithms popular in the classical machine learning community. Section 4.2 will focus on graph kernels, and Section 4.4 will focus on graph neural networks and graph transformers. Section 4.5 will introduce the Weisfeiler-Lehman test, a useful tool to quantify the expressivity of graph neural networks. Finally Section 4.6 will introduce the basics of the neutral atoms quantum hardware.

4.1 Graph machine learning and applications

A graph is is a set of nodes (or vertices) $\mathcal{V}$ linked by a set of edges (or links) which are tuples of nodes $\mathcal{E} = \{(u, v), u \in \mathcal{V}, v \in \mathcal{V}\}$. The graph is said *directed* if the order of the edge is important, ie (u, v) is different from (v, u) and *undirected* otherwise. Graph data often possess features linked to the nodes or edges. We note for each node $v \in \mathcal{V}$ the node feature vector $h_v \in \mathbb{R}^{d_V}$ and for each edge $(u, v) \in \mathcal{E}$ the edge feature vector $e_{uv} \in \mathbb{R}^{d_E}$. We note by H the matrix of shape $(|\mathcal{V}|, d_V)$, where each row v represents the feature vector of the node v and by E the matrix of shape $(|\mathcal{E}|, d_E)$, where each row (u, v) represents the feature vector of the edge (u, v).

Graph machine learning (GML) is an expanding field of research with applications in

Domain	Node features	Edge features	Task
Recommender systems	User's characteristics	User-item interactions: likes, ratings	Predict new items users may like
Molecules	Atom types	Chemical bonds types	Toxicity prediction
Social networks	User's posts	Friendship ties	Predict interests of users

Table 4.1: A few examples of real world applications of graph machine learning. The type of graphs, the node features, the edge features and the corresponding tasks are indicated.

chemistry (VB12; GSR+17), biology (ZAL18; MOB20; BOS+05), drug design (Kon14), social networks (Sco11; PARS14), natural language processing (NMR+17; GŠ13), computer vision (HB07) and science (SGGP+20; XHLJ18). A few examples of machine learning tasks for graph data are enumerated in Table 4.1. In those fields, some problems can be efficiently tackled by machine learning, and observations have an inherent graph structure. Usual machine learning algorithms cannot be directly used to exploit this graph structure, requiring the development of specific algorithms.

4.2 Graph kernels

A large body of work exists in the classical machine learning literature, trying to study graphs through the use of *graph kernels* that are measures of similarity between graphs(LR15; KJM20; BGLL+20; SS01).

The idea behind the graph kernel approach is very generic, and consists first in finding a way to associate any graph with a *feature vector* encapsulating its relevant characteristics (the *feature map*) and then to compute the similarity between those vectors, in the form of a scalar product in the feature space. A graph kernel K constitutes such an inner product and therefore allows to perform machine learning tasks on graphs. More specifically, a graph kernel is a symmetric, positive semidefinite function defined on the space of graphs $\mathbb{G}$. Given a kernel K, there exists a map $\phi : \mathbb{G} \to \mathcal{H}$ into a Hilbert space $\mathcal{H}$ such that $K(\mathcal{G}_1, \mathcal{G}_2) = \langle \phi(\mathcal{G}_1) | \phi(\mathcal{G}_2) \rangle$ for all $\mathcal{G}_1, \mathcal{G}_2 \in \mathbb{G}$(NSV19). The design of a kernel always comes down to a trade-off between capturing enough characteristics of the graph structure while still being algorithmically efficient.

For completeness, we detail here two algorithms in which graph kernels can be used: Support Vector Machine (SVM) for classification (*i.e.* sorting data in categories) and Kernel Ridge Regression (KRR) for regression (*i.e.* the prediction of continuous values)(SS01; BN06). These methods have been successfully applied to data sets of graphs of up to a few dozen nodes (MKB+20). KRR has been already introduced in another context in Section 1.2.

4.2.1 Support Vector Machine

The SVM algorithm aims at splitting a dataset in two classes by finding the best hyperplane that separates the data points in the feature space, in which the coordinates of each data point (here each graph) is determined according to the kernel K.

For a training graph dataset $\{\mathcal{G}_i\}_{i=1\dots M}$, and a set of labels $y = \{y_i\}_{i=1\dots M}$ (where $y_i = \pm 1$ depending on which class the graph $\mathcal{G}_i$ belongs to), the dual formulation of the SVM problem consists in finding $\tilde{\boldsymbol{\alpha}} \in \mathcal{A}_C(y) = \left\{ \boldsymbol{\alpha} \in [0, C]^M \middle| \boldsymbol{\alpha}^T y = 0 \right\}$ such that

$$\frac{1}{2}\tilde{\boldsymbol{\alpha}}^T Q \tilde{\boldsymbol{\alpha}} - e^T \tilde{\boldsymbol{\alpha}} = \min_{\boldsymbol{\alpha} \in \mathcal{A}_C(y)} \left\{ \frac{1}{2}\boldsymbol{\alpha}^T Q \boldsymbol{\alpha} - e^T \boldsymbol{\alpha} \right\} \tag{4.1}$$

where e is the vector of all ones, Q is a $M \times M$ matrix such that $Q_{ij} = y_i y_j K(\mathcal{G}_i, \mathcal{G}_j)$, and C is the penalty hyperparameter, to be adjusted. Setting C to a large value increases the range of possible values of α and therefore the flexibility of the model. But it also increases the training time and the risk of overfitting.

The data points for which $\tilde{\alpha}_i > 0$ are called support vectors (SV). Once the α_i are trained, the class of a new graph $\mathcal{G}$ is predicted by the decision function, given by:

$$y(\mathcal{G}) = \operatorname{sgn}\left\{ \langle \phi(\mathcal{G}) | \phi_0 \rangle \right\} \tag{4.2}$$

$$= \operatorname{sgn}\left\{ \sum_{i \in SV} y_i \tilde{\alpha}_i K(\mathcal{G}, \mathcal{G}_i) \right\}, \tag{4.3}$$

with

$$\phi_0 = \sum_{i \in SV} y_i \tilde{\alpha}_i \phi(\mathcal{G}_i) \tag{4.4}$$

In this case, the training of the kernel amounts to finding the optimal feature vector ϕ_0. It is worth noting that in many cases, equation (4.3) is evaluated directly, without explicitly computing ϕ_0.

If the dataset is to be split into more than two classes, a popular approach is to combine several binary classification in a one-vs-one scheme(POG+20). This means that a classifier is constructed for each pair of classes in the dataset. Namely, for a dataset with n_c classes, $n_c(n_c - 1)/2$ classifiers will be constructed and trained (one for each pair of classes). This is the strategy that will be used here, whenever necessary.

4.2.2 Kernel Ridge Regression

The regression is similar to the classification task, but here the aim is to attribute a continuous value to each graph. Given a training graph dataset $\{\mathcal{G}_i\}_{i=1\dots M}$, and a set of labels $y = \{y_i\}$ (that we assumed here to be in $\mathbb{R}$), the problem of linear regression consists in finding weights $\boldsymbol{\alpha} = \{\alpha_i\}_{i=1\dots d}$ (where d is the dimension of the embedding space $\mathcal{H}$), such that for any new

input graph $\mathcal{G}$, $y_{\boldsymbol{\alpha}}(\mathcal{G}) = \boldsymbol{\alpha}^T \phi(\mathcal{G})$. The solution $\boldsymbol{\alpha}$ is found by minimizing

$$J(\boldsymbol{\alpha}) = \frac{1}{2}\sum_{i=1}^{M} [y_{\boldsymbol{\alpha}}(\mathcal{G}_i) - y_i]^2 + \frac{\lambda}{2}\boldsymbol{\alpha}^T\boldsymbol{\alpha}, \tag{4.5}$$

where λ is the regularization hyperparameter. This problem has a dual formulation by setting $\boldsymbol{a} = \boldsymbol{\Phi}^T\boldsymbol{\alpha}$ where $\boldsymbol{\Phi}$ is the matrix whose rows are the embedded vectors $\phi(\mathcal{G}_i)$. By injecting the value of $\boldsymbol{a} = \{a_i\}_{i=1\dots M}$ in (4.5) the solution to the problem is given by

$$\boldsymbol{a} = (\boldsymbol{K} + \lambda I)^{-1}\boldsymbol{y} \tag{4.6}$$

where $\boldsymbol{K} = \{K(\mathcal{G}_i, \mathcal{G}_j)\}_{ij} = \{\langle\phi(\mathcal{G}_i)|\phi(\mathcal{G}_j)\rangle\}_{ij}$ is the kernel matrix and $\boldsymbol{y}$ is the vector of targets.

The prediction for a new input $\mathcal{G}$ is then given by:

$$y(\mathcal{G}) = \sum_{i=1}^{M} a_i K(\mathcal{G}, \mathcal{G}_i) \tag{4.7}$$

4.3 Examples of graph kernels

In this section, we give a brief description of a few examples of graph kernels. More examples can be found in Appendix A.7. These kernels will be used as a benchmark for the quantum algorithm QEK in Chapter 6.

4.3.1 Size kernel

The Size kernel only depends on the number of vertices of the graphs. It may seem trivial, but is a relevant baseline for the experiments described in Chapter 6. Given two graphs $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1)$ and $\mathcal{G}_2 = (\mathcal{V}_2, \mathcal{E}_2)$, the Size kernel is defined as:

$$K_{\text{size}}(\mathcal{G}_1, \mathcal{G}_2) := e^{-\gamma(|\mathcal{V}_1| - |\mathcal{V}_2|)^2} \tag{4.8}$$

with a choice of hyperparameter $\gamma > 0$.

4.3.2 Graphlet Sampling kernel

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and $H = (\mathcal{V}_H, \mathcal{E}_H)$ be two graphs. We say that $\mathcal{H}$ is a subgraph of $\mathcal{G}$ if there exists an injective map $\alpha : \mathcal{V}_{\mathcal{H}} \to \mathcal{V}$ such that $(u, v) \in \mathcal{E}_H \iff (\alpha(u), \alpha(v)) \in \mathcal{E}$. In general it might be possible to map $\mathcal{H}$ into $\mathcal{G}$ in several different ways, i.e. the mapping α, if it exists, is not necessarily unique.

Given two graphs $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1)$ and $\mathcal{G}_2 = (\mathcal{V}_2, \mathcal{E}_2)$, the idea behind the Graphlet kernel is to pick an integer $k < \min\{|\mathcal{V}_1|, |\mathcal{V}_2|\}$, enumerate all possible graphs of size k and find the number of ways they can be mapped to $\mathcal{G}_1$ and $\mathcal{G}_2$. Denote by $f_{\mathcal{G}_i}^{(k)}$ the vector where each entry counts

the way a specific graph of size k can be mapped as a subgraph of $\mathcal{G}_i$. A kernel can then be defined as the dot product $f_{\mathcal{G}_1}^{(k)} \cdot f_{\mathcal{G}_2}^{(k)}$ between the two vectors.

The complexity of computing such a kernel scales as $O(n^k)$, as there are $\binom{n}{k}$ size-k subgraphs in a graph of size n. For this reason it is preferable to resort to sampling rather than complete enumeration (SVP+09). Given a choice of integer N, graphs $g_1, \ldots, g_N$ of size between 3 and k are randomly sampled. The number of ways each g_i can be mapped as a subgraph of $\mathcal{G}_j$ is computed and stored in a vector $f_{\mathcal{G}_j}$, and the Graphlet Sampling kernel is defined as the dot product:

$$K_{\mathrm{GS}}(\mathcal{G}_1, \mathcal{G}_2) := f_{\mathcal{G}_1} \cdot f_{\mathcal{G}_2} \tag{4.9}$$

To account for the different size of $\mathcal{G}_1$ and $\mathcal{G}_2$, each vector can be normalized by the total number of its subgraphs.

4.3.3 Random Walk kernel

The Random Walk kernel is one of the oldest and most studied graph kernels (GFW03). Given two graphs $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1)$ and $\mathcal{G}_2 = (\mathcal{V}_2, \mathcal{E}_2)$, the idea is to measure the probability of simultaneous random walks of a certain length between two vertices in $\mathcal{G}_1$ and $\mathcal{G}_2$.

Simultaneous random walks can be conveniently encoded in powers of the adjacency matrix on the product graph. The product graph $\mathcal{G}_1 \times \mathcal{G}_2 = \mathcal{G}_\times = (\mathcal{V}_\times, \mathcal{E}_\times)$ is defined as follows:

$$\mathcal{V}_\times := \{(u_i, u_r) \mid u_i \in \mathcal{V}_1,\ u_r \in \mathcal{V}_2\} \tag{4.10}$$

$$\mathcal{E}_\times := \{((u_i, u_r), (v_j, v_s)) \mid (u_i, v_j) \in \mathcal{E}_1, \\ (u_r, v_s) \in \mathcal{E}_2\}. \tag{4.11}$$

In other words, an edge in the product graph indicates that an edge exists between the endpoints in both $\mathcal{G}_1$ and $\mathcal{G}_2$. If $A_\times$ is the adjacency matrix of the product graph, then the entries of $A_\times^k$ indicate the probability of a simultaneous random walk of length k between two vertices $u_i, v_j \in \mathcal{V}_1$ and $u_r, v_s \in \mathcal{V}_2$.

If $p, q \in \mathbb{R}^{|\mathcal{V}_\times|}$ are vectors representing the probability distribution of respectively starting or stopping the walk at a certain node of $\mathcal{V}_\times$, the first idea for a kernel would be to compute the sum $\sum_k q^{\mathrm{T}} A_\times^k p$, which however may fail to converge. A simple modification to make the sum convergent is to choose an appropriate length-dependent weight $\mu(k)$:

$$K(\mathcal{G}_1, \mathcal{G}_2) := \sum_{k=0}^{\infty} \mu(k)\ q^{\mathrm{T}} A_\times^k p. \tag{4.12}$$

The Geometric Random Walk kernel is obtained by choosing the weights to be the coefficients of a geometric series $\mu(k) = \lambda^k$, and p, q to be uniform. If λ is tuned in such a way as to make the series convergent, the kernel reads:

$$K_{\mathrm{RW}}(\mathcal{G}_1, \mathcal{G}_2) := \sum_{k=0}^{\infty} \lambda^k\ e^{\mathrm{T}} A_\times^k e = e^{\mathrm{T}} \left(I - \lambda A_\times\right)^{-1} e \tag{4.13}$$

where e denote vectors with all the entries equal to 1.

The cost of matrix inversion scales as the cube of the matrix size. If $|\mathcal{V}_1| = |\mathcal{V}_2| = n$, then the cost of the algorithm scales as $O(n^6)$, as it involves the inversion of an adjacency matrix of size $n^2 \times n^2$. Several methods are proposed in (VSKB10) to make the computation faster. The Spectral Decomposition method in particular allows to reduce the complexity for unlabeled graphs to $O(n^3)$. Essentially, one exploits the fact that the adjacency matrix of the product graph can be decomposed in the tensor product of the individual adjacency matrices:

$$A_\times = A_1 \otimes A_2 \tag{4.14}$$

which allows to diagonalize each $n \times n$ adjacency matrix in $O(n^3)$ time and perform the inversion only on the diagonal components.

4.4 Graph neural networks and graph transformers

Graph Neural Networks (GNNs) are neural networks that take into account the graph structure. The objective is to learn suitable vector embeddings of the nodes and edges that enable efficient solutions to the original problem. In the past few years, significant effort has been put into the design of GNNs (Ham20).

A GNN is composed of a succession of layers. The node and edge features at the layer $\ell + 1$ are computed from

$$H_{\ell+1} = f_{\ell+1}(H_\ell, E_\ell; W_l) \tag{4.15}$$

$$E_{\ell+1} = f'_{\ell+1}(H_\ell, E_\ell, W_\ell) \tag{4.16}$$

f_ℓ, f'_ℓ are the functions of the layer ℓ, and W_ℓ is the associated weight matrix. Training the neural network amounts to minimizing a loss function which depends on the problem to solve. For example, in the case of a graph regression task over a dataset $\{(\mathcal{G}_i, y_i), i \in [1, M]\}$, the loss function to minimize can be expressed as

$$\mathcal{L}(W) = \sum_{i=1}^{M} (f(\mathcal{G}_i;\, W) - y_i)^2 \tag{4.17}$$

The optimization is usually done with a gradient descent algorithm (Rud16).

There are little limitations on how the functions f'_ℓ and f_ℓ can be, it must however respect the property of equivariance by relabeling the nodes. It means that one must obtain the same output for a given node if the indexing is changed. It means that for every permutation matrices P and P' associated a node relabelling, one must have:

$$f_{\ell+1}(PH_\ell, P'E_\ell; W_l) = Pf_{\ell+1}(H_\ell, E_\ell; W_l) \tag{4.18}$$

$$f'_{\ell+1}(PH_\ell, P'E_\ell, W_\ell) = P'f_{\ell+1}(H_\ell, E_\ell; W_l) \tag{4.19}$$

4.4.1 Message passing neural networks

Message Passing Neural Networks are a family of GNNs where for each layer the feature vector $h_v^{\ell+1}$ is a function of h_v^ℓ and $\{h_v^\ell, v \in \mathcal{N}(v)\}$. More precisely, the update equations can be written the following way

$$m_v = \text{UPDATE}(h_v) \tag{4.20}$$
$$h_v^{l+1} = \text{AGGREGATE}(m_v, \{m_v,\ v \in \mathcal{N}(v)\}) \tag{4.21}$$

where UPDATE and AGGREGATE are parameterized functions. For each node, a message m_v is computed from h_v with the function UPDATE, and the messages of the neighbors of v are aggregated to v with the function AGGREGATE. Plenty of variants alongside this architecture have been proposed. We detail here a few of them. In the following, σ is an activation function.

Graph Convolution Networks.
Graph Convolution Networks (GCN) have been originally introduced in (KW16), and are now one of the most effective GNN baseline. The aggregation is made by summing the normalized messages of the neighbors. The update rule can be written as

$$h_v^{\ell+1} = \sigma\Big(\sum_{u \in \mathcal{N}(v)\cup\{v\}} W^{\ell+1} \frac{h_u^\ell}{\sqrt{|\mathcal{N}(v)||\mathcal{N}(u)|}}\Big) \tag{4.22}$$

Graph Attention Networks.
Another strategy of aggregation of messages is to apply attention (BCB14). Attention between two elements can be understood as an importance weight of the relationship between those elements. (VCC+18) proposed to use attention to improve the aggregation procedure of GCNs. The authors suggest to compute an aggregation weight between a node and its neighbors. The update can then be written as

$$h_v^{\ell+1} = \sigma\Big(\sum_{u \in \mathcal{N}(v)\cup\{v\}} \alpha_{uv} W^{\ell+1} h_u^\ell\Big) \tag{4.23}$$

where

$$\alpha_{uv} = \frac{\exp\Big(a^\top [W^{\ell+1} h_u \oplus W^{\ell+1} h_v]\Big)}{\sum_{u \in \mathcal{N}(v)\cup\{v\}} \exp\Big(a^\top [W^{\ell+1} h_u \oplus W^{\ell+1} h_v]\Big)} \tag{4.24}$$

4.4.2 Transformers

The transformer architecture has been first introduced by (VSP+17). It is now one of the most popular neural network architecture used for sequence modelling (Dev18; ADL+22) but also for

computer vision tasks (CMS+20; DBK+20). Transformers are neural networks only composed of attention layers, and were originally introduced for sequence modelling tasks. Attention was already a feature accompanying Recurrent Neural Networks (RNNs) (WHZZ16). A transformer layer is a parameterized function $f_\theta : \mathbb{R}^{N\times d} \longrightarrow \mathbb{R}^{N\times d_{\text{model}}}$. An attention head is defined as

$$\text{HEAD}(X; Q, K, V) = \text{softmax}(XQK^\top X)XV \tag{4.25}$$

Q, K, V are matrices of shape (d, d_{head}), and are usually called query, key and value matrices respectively.

A transformer layer is the concatenation of attention heads such that

$$\text{LAYER}(X) = \Big|\Big|_{i=1}^{n_{\text{heads}}} \text{HEAD}(X; Q, K, V) \tag{4.26}$$

$||$ denotes the concatenation of matrices by the columns, and n_{heads} are concatenated such that $n_{\text{heads}} \times d_{\text{head}} = d_{\text{model}}$

4.4.3 Graph transformers and positional encodings

Despite some successes, it has been shown that MPNNs suffer several flaws. First and foremost, their theoretical expressivity is related to the Weisfeiler-Lehman (WL) test. It means that two graphs who are indistinguishable via the WL test will lead to the same MPNN output (MRF+19). This can cause several problems because two different substructures will not be differentiated. More details can be found in Section 4.5. MPNNs also perform best with homophilic data and seem to fail on heterophilic graphs (ZYZ+20). Homophilic graphs mean that two nodes have the same labels if they are close to each other in the graph, which is not necessarily the case. Finally, MPNNs suffer from oversmoothing (CLL+20) and oversquashing (TDGC+21). Oversmoothing means that the output features of all nodes will converge to the same value as the number of layers increases. Oversquashing occurs when few links on the graph separates two dense clusters of nodes. The information that circulates through these links is then an aggregation of many nodes and is much poorer compared to the information initially present.

Solutions to circumvent those issues are currently investigated by the community. The main idea is not to limit the aggregation to the neighbors, but to include the whole graph, or a larger part of it. Graph Transformers were created in this spirit with success on standard benchmarks (YCL+21; RGD+22). Similarly to the famous transformer architecture, an aggregation rule is provided to every pair of nodes in the graph with incorporation of global structural features.

"Positional" or "structural" embeddings are features computed from the graph that are concatenated to original node or edge features to enrich GNN architectures (either MPNN or GT). These two terms are used interchangeably in the literature and we denote them as "positional encodings" (PEs) in the rest of this work. PEs can include random walk probabilities (RGD+22; MLL+23), spectral information (DJL+20; RGD+22; KBH+21), shortest path distances (LHW18), or heat kernels (MCSM21). They can also be learned (DLL+21). Table 4.2

extracted from (RGD+22) gives a few examples of positional encodings in the literature. We detail below the most common ones that will be included in our benchmarks.

Laplacian Eigenvectors. The spectral information of the graph can be used as PE, more precisely the eigenvectors of the Laplacian matrix with the smallest eigenvalues, or laplacian eigenvectors (LE). For a line graph, the laplacian eigenvectors almost correspond to positional embeddings in the transformer architecture for sequences (VSP+17). The main issue of this encoding is to ensure that the model remains invariant by changing the sign of eigenvectors, and a solution has been proposed by (LRZ+22).

Relative Random Walk Probabilities (RRWP). The authors of (MLL+23) introduced the RRWP with which they initialize their model. For a graph $\mathcal{G}$, let A be the adjacency matrix and D the degree matrix. Let P be a 3 dimensional tensor such that $P_{k,i,j} = (M^k)_{ij}$ with $M = D^{-1}A$. For each pair of node (i, j), we associate the vector $P_{:,i,j}$, i.e., the concatenation of the probabilities for all k to get from node i to node j in k steps in a random walk. $P_{:,i,i}$ is the same as the Random Walk Structural Encodings (RWSE) defined in (RGD+22). The authors of (MLL+23) highlight the benefits of RRWP. They prove that the Generalized Distance WL (GD-WL) test introduced by (ZLWH23) with RRWP is strictly more powerful than GD-WL test with the shortest path distance, and they prove universal approximation results of multi-layer perceptrons (MLP) initialized with RRWP. They also achieve state of the art results on most of benchmark datasets.

(DLL+21) proposed a way to learn the position embedding from the initial embedding that can either be the laplacian eigenvectors or the diagonal of the random walk matrix. They define an architecture in which position embeddings are concatenated to node features and are updated separately at each layers. We detail here the equations of their model. Consider at a layer ℓ the feature vector of the node i noted h_i^ℓ, the feature vector to the edge (i, j) noted e_{ij}^ℓ, and position vector of the node i noted p_i^ℓ. Then the updates are computed with the following formulas.

$$h_i^{\ell+1} = f_h\left(\begin{bmatrix} h_i^\ell \\ p_i^\ell \end{bmatrix}, \left\{ \begin{bmatrix} h_j^\ell \\ p_j^\ell \end{bmatrix} \right\}_{j\in\mathcal{N}(i)}, e_{ij}^\ell\right) \tag{4.27}$$

$$e_i^{\ell+1} = f_e(h_i^\ell, h_j^\ell, e_{ij}^\ell) \tag{4.28}$$

$$p_i^{\ell+1} = f_p(p_i^\ell, \{p_j^\ell\}_{j\in\mathcal{N}(i)}, e_{ij}^\ell) \tag{4.29}$$

This scheme gives a supplementary expressivity compared to just taking the position features as an input to the GNN. f_p and f_h follow the same analytical formula, but the activation function used for f_p is tanh, allowing negative values.

The authors separate the training of the position encoding. The final loss function is written

Global PE (node features)	Allow a node to know its global position within the graph	• Eigenvectors of the Adjacency, Laplacian or distance matrices. (KBH+21; DJL+20) • Distance from the graph's centroid. • Unique identifier for each connected component of the graph.
Relative PE (edge features)	Allow two nodes to understand their distances or directional relationships	• Pair-wise node distances from heat kernels, random-walks, Green's function, graph geodesic, or any local/global PE. (KBH+21; BPL+21; MCSM21) • Gradient of eigenvectors or any local/global PE. (KBH+21; BPL+21) • Boolean indicating if two nodes are in the same cluster.
Local SE (node features)	Allow a node to understand what substructures it is a part of	• Degree of a node (YCL+21) • Diagonal of the m-steps random-walk matrix (DLL+21) • Time-derivative of the heat-kernel diagonal (gives the degree at t = 0). • Enumerate or count predefined structures such as triangles, rings, etc. (BFZB22; ZJAS21) • Ricci curvature (TDGC+21)

Table 4.2: A few examples of different positional (PE) encodings and structural encodings (SE) identified from (RGD+22).

as

$$\text{Loss} = \text{Loss}_{\text{Task}}(h^L, p^L) + \alpha \text{Loss}_{\text{Pos}}(p^L) \tag{4.30}$$

$$\text{Loss}_{\text{Pos}}(p) = \frac{1}{k} Tr(p^T L p) + \frac{\lambda}{k} ||p^T p - I_k||_F^2 \tag{4.31}$$

with $\alpha, \lambda > 0$ hyperparameters and $|| \cdot ||_F$. the Frobenius norm.

4.5 Theoretical expressivity of GNNs

4.5.1 Weisfeiler Lehman test

Evaluating the expressivity of graph neural networks is a rich area of research. The Weisfeiler-Lehman test (WL test) is an important tool introduced by (WL68). It is a widely popular fast to compute isomorphism test. It is not universal, (Bam22) proposed a method to generate graphs non distinguishable by the WL test. One can also improve the test by adding a notion of distance (ZLWH23).

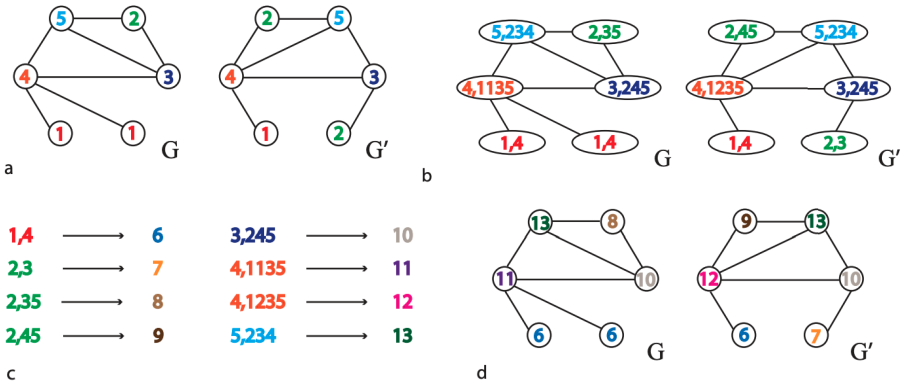


Figure 4.1: General steps of the WL test algorithm. a. Two graphs with node colorings. b. Node coloring and the nodes of the neighbors. c. New mapping from the set of colors and colors of neighbors to new colors. d. The two graphs with new colors from the previous mapping. Elements of the figure are taken from (SSVL+11), figure 2 in accordance with the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/legalcode) .

Wesfeiler-Lehman test

The Weisfeiler-Lehman test (WL test) is an isomorphism test developed from a color refinement algorithm (WL68; BK79). The algorithms outputs for each graph a set of node colorings that are computed iteratively. We start with a graph $\mathcal{G}$ and labeled nodes. Each node v is labeled by a color c_v. The colors of the nodes are computed iteratively, the color of the node v at step t, noted $c_v^{(t)}$ can be computed by hashing $c_v^{(t-1)}$ with the colors of the neighbors of v, $\{c_u^{(t-1)} \mid u \in \mathcal{N}(v)\}$.

The update rule can be written as

$$c_v^{(t)} = HASH(\{c_v^{(t-1)}, \{c_u^{(t-1)} \mid u \in \mathcal{N}(v)\}\}) \tag{4.32}$$

where $HASH$ is an injective hash function. The process is repeated until convergence. The final isomorphism test is done by comparing the histograms of final colors for two graphs.

Higher order WL tests can be defined by aggregating the k neighbors of the nodes, giving birth to the family of $k-WL$ tests.

Generalized Distance Wesfeiler-Lehman test
The GDWL test (ZLWH23) is similar to the WL test, but add a notion of distance between all pairs of nodes. The iterative update can be written as

$$c_v^{(t)} = HASH(\{c_v^{(t-1)}, \{(c_u^{(t-1)}, d(u,v) \mid u \in \mathcal{N}(v)\}\}) \tag{4.33}$$

4.5.2 Properties

The WL test is especially relevant for the study of GNNs. It has been shown that the theoretical expressivity of MPNNs can be directly linked to the WL test. It means that two graphs who are indistinguishable via the WL test will lead to the same MPNN output (MRF+19). To alleviate the limitation, the authors have proposed to use higher The GD-WL has been used to characterize the expressivity of the GRIT architecture developed by (MLL+23).

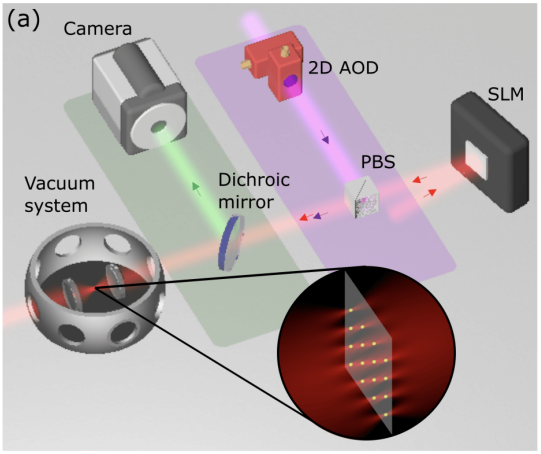
4.6 Neutral atoms quantum computing

In this section, I explain the inner working of a neutral atom quantum computer. I start by describing the physical set up of the machine, and I expand on the analog quantum computation that the platform is able to perform. This ability is its main advantage compared to other types of quantum hardware.

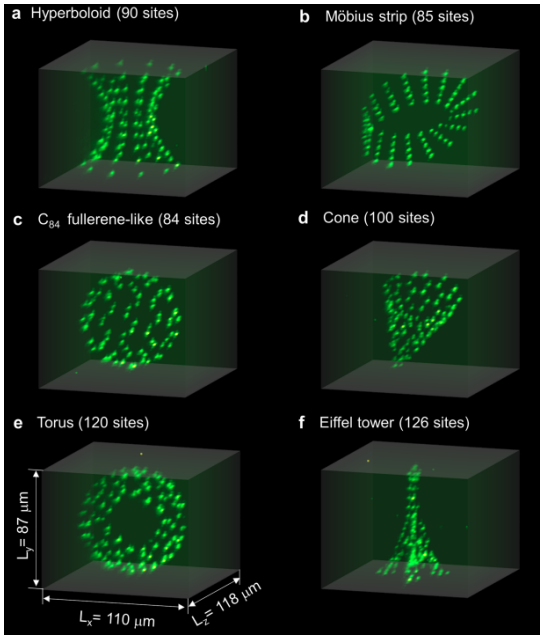
4.6.1 Rydberg atoms and optical tweezers

Neutral atoms quantum computers use atoms in their rydberg state as qubits. The most common element used is the rubidium, where the states $|0\rangle$ and $|1\rangle$ are chosen as energy levels of the atom. These atoms are manipulated by light and trapped in a vacuum chamber.

A laser and a Spatial Light Modulator (SLM) will create a set of traps in the chamber (HBS+20), and each trap will capture exactly one atom. Experimental progress enabled the manipulation of hundreds of atoms (SSW+21; EKC+22), approaching the thousands (PLB+24), in comparison to one atom at the beginning of the century (SRPG01). It also enabled a very large degree of freedom in the positioning of the atoms, allowing to realize different shapes and lattices (BLL+18), as shown in Figure 4.2a. Information processing is done by applying a laser pulse to the register of atoms in their Rydberg state while shutting off the traps. Measurement is realized by capturing emitted photons by the atoms after turning on the traps again. If the atom is in the $|0\rangle$ state, it will be captured by the traps and be detected whereas if it is in the $|1\rangle$ state it will be ejected.



(a) Set up of a neutral atom device from (HBS+20)



(b) Array of atoms from (BLL+18)

Figure 4.2: An illustration of the experimental set up of a neutral atom quantum computer and some shapes that can be produced with arrays of atoms.

4.6.2 Analog quantum computing

Although neutral atom devices can be used as digital quantum computer, they are currently most effectively useful for performing analog computation. If we consider n atoms located at the positions $(r_1, \dots r_n)$, the global hamiltonian of the system is given by

$$\hat{\mathcal{H}}_{\text{ising}} = \Omega(t) \sum_{i=1}^{n} X_i - \delta(t) \sum_{i=1}^{n} N_i + C_6 \sum_{i,j} \frac{N_i N_j}{|r_i - r_j|^6} \tag{4.34}$$

with $N = (I + Z)/2$. $\Omega(t)$ and $\delta(t)$ are time dependent functions depending on the laser pulse applied to the register. C_6 is a constant which depends on the atomic levels used to encode the $|0\rangle$ state and the $|1\rangle$ state. Ω is called the Rabi frequency and depends on the power of the laser, δ is called the detuning, and depends on the frequency of the laser. The quantum state of the system will be given by the solution to the Schrodinger equation

$$\frac{d|\psi\rangle}{dt} = -i\hat{\mathcal{H}}(t)|\psi\rangle \tag{4.35}$$

In the rest of this thesis unless specified otherwise we assume that $\hbar = 1$.

Another important hamiltonian that can be realized on the platform is the XY hamiltonian

$$\hat{\mathcal{H}}_{\text{XY}} = \Omega(t) \sum_{i=1}^{n} X_i - \delta(t) \sum_{i=1}^{n} N_i + C_3 \sum_{i,j} \frac{X_i X_j + Y_i Y_j}{|r_i - r_j|^3} \tag{4.36}$$

C_3 is a constant just like C_6 which depends on the atomic levels used for the encoding.

Analog quantum computing opens new possibilities in comparison to digital quantum computing. One can indeed in the analog mode prepare unitaries that would take a number of 2-qubits and 1-qubits gates much bigger than any available digital hardware. (DBK+22) showed as an example what were the resources needed to simulate a 10x10 Hubbard model at the same error rate as in current experiments (GB17). They showed that one would need at least 200 logical qubits, 10^6 error corrected gates at 10^-7 error rate. This is beyond any available hardware on other platforms.

Chapter 5

Quantum Algorithms for graph machine learning

This chapter will develop core contributions of this thesis. I will describe the different ways to construct a machine learning model that takes a graph as an input, using a quantum computer. Unlike the previous part, this chapter is written from different parts of the following papers:

- (ADL+23) Boris Albrecht, Constantin Dalyac, Lucas Leclerc, Luis Ortiz-Gutiérrez, Slimane Thabet, Mauro D'Arcangelo, Julia RK Cline, Vincent E Elfving, Lucas Lassablière, Henrique Silvério, Bruno Ximenez, Louis-Paul Henry, Adrien Signoles, Loïc Henriet. (2022) *Quantum feature maps for graph machine learning on a neutral atom quantum processor.* Physical Review A, 107(4), 042615.

- (TFH22) Slimane Thabet, Romain Fouilland, Loic Henriet (2022). *Extending Graph Transformers with Quantum Computed Correlations arXiv:2210.10610*

- (TDS+) Slimane Thabet, Mehdi Djellabi, Igor Sokolov, Sachin Kasture, Louis-Paul Henry, Loïc Henriet (2024). *Quantum Positional Encodings for Graph Neural Networks.* International Conference on Machine Learning, 2024

The process of constructing quantum models for graph data has been done iteratively in the presented papers, and by following the evolution of the classical graph machine learning community. In this thesis, I wished to unify the work I have done under a single framework. I will therefore present all the algorithms I developed, and for which I have performed experiments,

as variations of a single workflow. This way of presenting will allow inspirations for future work by encouraging to create more variations.

In Section 5.1, I give an overview of the elements of a quantum algorithm for graph machine learning. From Section 5.2 to Section 5.5, I develop each one of these elements. In Section 5.6, I provide some theoretical results about the capabilities of the constructed quantum algorithms.

5.1 General view

Building a graph machine learning model with a quantum computer amounts to constructing a function

$$f(\mathcal{G};\, \theta,\, W) = g(|\psi(\mathcal{G};\, \theta)\rangle;\, W) \tag{5.1}$$

where $\mathcal{G}$ is a graph, θ is a vector of parameters, $|\psi(\mathcal{G};\, \theta)\rangle$ is a quantum state depending on $\mathcal{G}$ and θ, and W is a vector of classical parameters.

The process can be decomposed into 4 steps and is illustrated in Figure 5.1.

1. Mapping of the graph $\mathcal{G}$ to a hamiltonian $\hat{\mathcal{H}}_{\mathcal{G}}$ with each node being mapped to one qubit.
2. Creating a quantum state dependent on the graph with the hamiltonian previously defined. The quantum state may involve some parameters to be optimized
3. Performing measurements on the quantum state previously defined.
4. Performing a post processing of the measurements. The post processing can be minimal (*i.e* the measurement is the final output), or be a significant part of the procedure.

At the end, one can add a feedback loop to any step of the procedure. It means that each of the previous step can be updated depending on the output of the model.

One may think intuitively that the potential advantage of using a quantum computer to perform graph machine learning tasks is the ability to potentially extract intractable topological features. It is known that quantum computers can help to solve combinatorial optimization problems involving graphs (FGG14; AAA+24; Had18). Two examples of such problems are the maximum independent set problem (MIS) and the maximum cut problem (MAX-CUT). The goal of the MIS is to find the largest set of vertices in the graph such that there exists no edge between any nodes. The goal of MAX-CUT is to find a partition of two sets such that the number of edges that cross going from one partition to the other is maximum. For a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ of n nodes, these problems can be formulated as finding binary variables $z = (z_1, \dots z_n) \in \{0, 1\}^n$ that maximize the following cost functions

$$C_{\text{MIS}} = \sum_{i \in \mathcal{V}} z_i - U \sum_{(i,j) \in \mathcal{E}} z_i z_j \tag{5.2}$$

$$C_{\text{MAX-CUT}} = \sum_{(i,j) \in \mathcal{E}} z_i (1 - z_j) \tag{5.3}$$

$$\tag{5.4}$$

These problems are believed to be hard to compute on a classical computer (GJ78; Spi99). The research on how to solve them more efficiently with classical computers is still very active (DHK+23), with concrete real world applications in mind (Ves22). Since one is able to construct hard to compute functions of a graph with a quantum computer, one may therefore think about using theses functions for machine learning tasks.

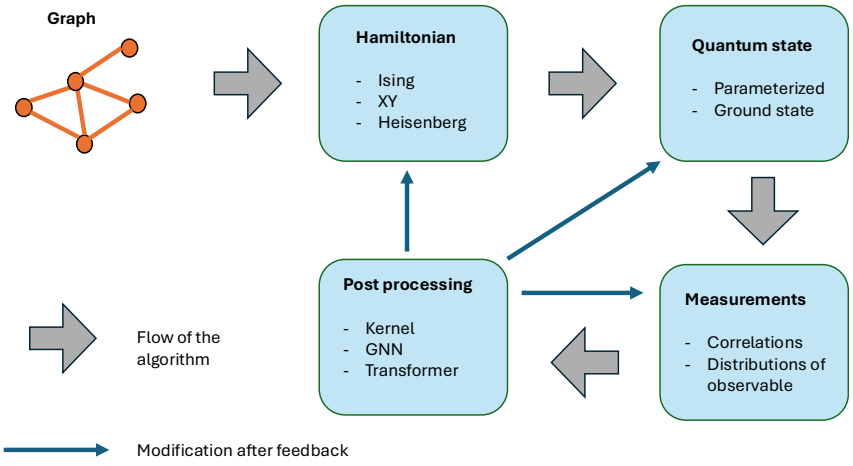


Figure 5.1: General steps of a graph machine learning algorithm with a quantum computer.

At the begining of this research, few works have explored the idea of using quantum computers to do graph machine learning. (SBI+20) proposed to use a gaussian boson sampler to extract graph features. The input graph is embedded into the interferometer such that the covariance matrix of the gaussian state is equal to the adjacency matrix of the graph. The device outputs photon counts then proportional to the permanent of the adjacency matrix of the graph. (VML+19) proposed a general framework for quantum variational ansatzes using a graph as input data, and proposed a few applications, like graph isomorphism or hamiltonian learning. Both previous works follow the different steps described.

During this thesis, a few other works have been developed by the quantum computing community about graph machine learning, following adjacent directions of this thesis (MMC22; LZF24). Reviews of the progress can be found in (TYE22; CMDF+24).

5.2 Graph to hamiltonian mapping

We associate a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, to a hamiltonian $\hat{\mathcal{H}}_\mathcal{G}$ of $|\mathcal{V}|$ qubits of the form

$$\hat{\mathcal{H}}_\mathcal{G} = \sum_{(i,j)\in\mathcal{E}} \hat{\mathcal{H}}_{ij} \tag{5.5}$$

where $\hat{\mathcal{H}}_{ij}$ is a Pauli string acting non-trivially on i and j only. Such examples of hamiltonians are the Ising hamiltonian

$$\hat{\mathcal{H}}^I = \sum_{(i,j)\in\mathcal{E}} Z_i Z_j \tag{5.6}$$

the XY hamiltonian

$$\hat{\mathcal{H}}^{XY} = \sum_{(i,j)\in\mathcal{E}} X_i X_j + Y_i Y_j \tag{5.7}$$

or the Heisenberg hamiltonian

$$\hat{\mathcal{H}}^{XXZ} = \sum_{(i,j)\in\mathcal{E}} X_i X_j + Y_i Y_j + Z_i Z_j \tag{5.8}$$

In the following of this work, I will mainly focus on the Ising and XY hamiltonian. Those two Hamiltonians are analyzed here because they are ubiquitous spin models, that can be rather easily implemented on currently existing platforms, particularly in the case of neutral-atom processors (SSW+21; SWB+22).

The neutral atom quantum computer offers the flexibility to change the topology of the system at each run which makes it a prime tool to run these procedures. The change of geometry of the atomic register only introduces a constant overhead cost due to the calibration of the devices. The calibration parameters can then be stored in memory and reused when needed. If one would want to do the same with another platform like superconducting qubits, one would need to introduce SWAP gates since the topology of the chip is fixed, which would significantly increase the depth of the circuit.

5.3 Graph quantum states

In the previous section I described how to create a mapping from a graph to a Hamiltonian. I describe now how to create a quantum state dependent on the graph, that I will call *graph quantum states*, not to be mistaken by other common definitions of graph states in quantum computing (HDE+06).

5.3.1 Ground states

One family of quantum states that is natural to explore is the ground state of the graph hamiltonian (GHL+15). Ground state properties are indeed widely studied in many-body physics and their properties depend on the topology of the graph. The ground state of a system is defined as the lowest-energy eigenstate of its hamiltonian (when it is degenerate, one considers the *ground state manifold* $\mathbb{H}_{GS}$).

Preparing this state is the purpose of quantum annealing (DC08). When using neutral atom quantum processors (HBS+20), one can natively address hamiltonians of the form

$$\hat{\mathcal{H}}_{\mathcal{G}} = \sum_{(i,j)\in\mathcal{E}} J_{ij}(Z_i - \alpha_i I)(Z_j - \alpha_j I) \tag{5.9}$$

with α_i real coefficients. Its eigenstates are the basis states $|\mathbf{b}\rangle = |b_1 \dots b_N\rangle$ described above. In the case where $\alpha_i = 1 - \delta/(2z_i)$ with $z_i = \sum_{j|(i,j)\in\mathcal{E}} J_{ij}$ and $J_{ij} = 1/4$, the eigenenergies (or eigenvalues) are

$$E(\mathbf{b}) = \sum_{i,j\in\mathcal{E}} b_i b_j - \delta \sum_{i=1}^{N} b_i. \tag{5.10}$$

When $0 < \delta < 1$, this is the cost function associated with the maximum independent set problem, a NP-hard problem (GJ79). In the absence of degeneracy-lifting or symmetry-breaking effects, a quantum annealing scheme would prepare an equal-weight superposition of all maximum independent sets. With that in mind, I will call *ground state of the graph* the state

$$|\psi_{GS}\rangle = \frac{1}{\sqrt{|\mathbb{H}_{GS}|}} \sum_{\mathbf{b}\in\mathbb{H}_{GS}} |\mathbf{b}\rangle. \tag{5.11}$$

Such a quantum annealing scheme is not however applicable to all systems. It would indeed only work if the energy gap between the ground state and the first excited state is big enough (Ami09; Dal23).

5.3.2 Parameterized quantum states

Alternate layered quantum state

We consider the quantum state obtained by alternated action of p layers of $\hat{\mathcal{H}}_\mathcal{G}$ and a *mixing* hamiltonian $\hat{\mathcal{H}}_M$ (that doesn't commute with $\hat{\mathcal{H}}_\mathcal{G}$, for instance $\hat{\mathcal{H}}_M \propto \sum_{i=1}^{|\mathcal{V}|} Y_i$) on an initial state $|\psi_0\rangle$

$$|\psi_\mathcal{G}(\boldsymbol{\theta})\rangle = \prod_{k=1}^{p} \left(\mathrm{e}^{-i\hat{\mathcal{H}}_M\theta_k}\mathrm{e}^{-i\hat{\mathcal{H}}_\mathcal{G}t_k}\right)\mathrm{e}^{-i\hat{\mathcal{H}}_M\theta_0}\,|\psi_0\rangle\,, \tag{5.12}$$

where $\boldsymbol{\theta} = (\theta_0, t_0, \theta_1, t_1, \dots \theta_p)$ is a real vector of parameters. The choice of these states is motivated by their similarity with the *Trotterized* dynamics of several quantum systems(Suz76). They are also heavily used in other quantum algorithms for combinatorial optimization like the Quantum Approximate Optimization Algorithm (FGG14).

Continuously parameterized graph states

The *layered* time-evolution scheme, can be easily extended to any kind of parametrized time-evolution $\hat{\mathcal{H}}(t)$ such that the final state is

$$|\psi_f\rangle = \mathcal{T}\exp\left[-i\int_0^{t_f} dt\,\hat{\mathcal{H}}(t)\right]|\psi_0\rangle\,, \tag{5.13}$$

where $\mathcal{T}$ is the time-ordering operator. This can be thought of as taking the limit of an infinite number of layers. Analog quantum processing platforms are particularly well suited to the evaluation of this kind of quantities. One can indeed prepare quantum states resulting from the time evolution of a time dependent hamiltonian that can be written as

$$\hat{\mathcal{H}}(t) = \Omega(t)\sum_{i=1}^{|\mathcal{V}|} X_i - \delta(t)\sum_{i=1}^{|\mathcal{V}|} Z_i + \hat{\mathcal{H}}_\mathcal{G} \tag{5.14}$$

Indeed in those systems, some of the parameters of the Hamiltonian can be continuously tuned (and it is often difficult to generate strictly piece-wise constant Hamiltonians).

5.4 Measurements

In this section, I will describe the quantities that can be measured from the quantum state that has been created. Subsection 5.4.1 will present the probability distribution of an observable, Subsection 5.4.2 will present the correlation matrices, Subsection 5.4.3 will introduce the k-particles quantum random walks, and Subsection 5.4.4 will present a quantum inspired version of the quantum random walks. All these quantities satisfy the equivariance property crucial for graph algorithms Section 4.4.

The quantities defined here are well known in the quantum physics community, but their use as signature of a graph is an original contribution of this thesis.

5.4.1 Probability distribution of an observable

Let $\hat{\mathcal{O}}$ be an observable, $\{\lambda_1, \dots \lambda_K\}$ its eigenvalues (*i.e.* the possible outcomes of the measure), and $\{|o_1\rangle, \dots |o_K\rangle\}$ the corresponding eigenstates. The normalized histogram of measured values approaches in the large M limit the following probability distribution

$$\mathcal{P}_{\mathcal{G}}^{\hat{\mathcal{O}}}(\Lambda) = (p_1, \dots p_K), \text{ where } p_k = |\langle o_k|\psi_f\rangle|^2 . \tag{5.15}$$

This probability distribution can then be used as a graph feature.

Remark: Note that if some eigenvalues are degenerate, one would get instead $p_k = \sum_i \delta(\lambda_i - \lambda_k) |\langle o_i|\psi_f\rangle|^2$, where the p_k are restricted to the $\tilde{K} < K$ distinct eigenvalues of $\hat{\mathcal{O}}$.

In practice, if K is large, one would resort to *binning* the values of λ_i, *i.e.* by defining a set of $K' < K$ intervals $\{I_k = [\tilde{\lambda}_k, \tilde{\lambda}_{k+1}]\}_{k=1\dots,K'}$, with $\tilde{\lambda}_1 \leq \min_k \lambda_k$ and $\tilde{\lambda}_{K+1'} \geq \max_k \lambda_k$, such that $\mathcal{P}_{\mathcal{G}}^{\hat{\mathcal{O}}}(\Lambda) = (\tilde{p}_1, \dots \tilde{p}_{K'})$, where

$$\tilde{p}_k = \frac{|\{m_i | m_i \in I_k\}|}{M} \underset{M\to\infty}{\equiv} \sum_{i|\lambda_i \in I_k} |\langle o_i|\psi_f\rangle|^2 . \tag{5.16}$$

Typically $\hat{\mathcal{O}}$ will be a sum of Pauli strings, and must be invariant by the relabelling of the nodes. An observable that stisfies this property would be $\sum_i Z_i$.

5.4.2 Correlations

The correlations (or *correlators*) C_{ij} of local operators $\hat{\mathcal{O}}_i$ and $\hat{\mathcal{O}}_j$ acting respectively on qubits i and j can be defined either as the expectation value of their product $\langle \hat{\mathcal{O}}_i \hat{\mathcal{O}}_j \rangle$, or their covariance $\langle \hat{\mathcal{O}}_i \hat{\mathcal{O}}_j \rangle - \langle \hat{\mathcal{O}}_i \rangle \langle \hat{\mathcal{O}}_j \rangle$ (note that the orders matters if $\hat{\mathcal{O}}_i$ and $\hat{\mathcal{O}}_j$ don't commute). In the rest of

the thesis, I will indifferently call correlation the two former expressions, and give details when necessary. We will be focusing on the case where $\hat{\mathcal{O}}_i$ is a Pauli string of length 1 (i.e., X_i, Y_i or Z_i). This includes single Pauli types correlators (*i.e.* X_iX_j, Y_iY_j, Z_iZ_j) or multiple types Pauli correlators (*i.e.* X_iY_j, Y_iZ_j, X_iZ_j etc).

5.4.3 k-particles quantum random walks (k-QRW).

In this subsection, I introduce the k-particles (or walkers) random walk probabilities that can be obtained using $\hat{\mathcal{H}}^{XY}$ (see Equation (5.7)). We denote by $\mathbb{H}_k$ the k-particles subspace (*i.e.* the Hilbert space obtained as the span of states $|\mathbf{b}\rangle$ of Hamming weight k, noted $|i_1 \dots i_k\rangle$, parameterized by k integers $i_1 \dots i_k \in \{0,1\}^k$). It is a well-known property that $\hat{\mathcal{H}}^{XY}$ stabilizes each of the $\mathbb{H}_k$s and I denote by $\hat{\mathcal{H}}_k^{XY}$ the XY hamiltonian restricted to $\mathbb{H}_k$ (HTDH21a). $\hat{\mathcal{H}}^{XY}$ can also be called Hamming weight preserving.
$\hat{\mathcal{H}}_k^{XY}$ can be seen as the adjacency matrix of a graph called the k *occupation graph*. Therefore, a quantum evolution of $\hat{\mathcal{H}}_k^{XY}$ can be seen as a quantum walk on the k occupation graph. I use the hamiltonian $\hat{\mathcal{H}}_k^{XY}$ to prepare a quantum state as in Equation (5.12) that will represent a superposition over all k-tuples of nodes, and I measure observables for each pair of nodes which will give edge features. For a 1-particle QRW, I calculate the probability

$$[X^{(1)}(t)]_{ij} = |\langle j| e^{-i\hat{\mathcal{H}}_1^{XY}t} |i\rangle|^2 \tag{5.17}$$

to find particle at node j coming from node i after time t. Similarly for a 2-particle QRW, I calculate

$$[X^{(2)}(t)]_{ij} = |\langle ij| e^{-i\hat{\mathcal{H}}_2^{XY}t} |\psi_{\text{init}}\rangle|^2 \tag{5.18}$$

where $|i,j\rangle \in \hat{\mathcal{H}}_2^{XY}$ is the state with walkers at nodes i and j and

5.4.4 Quantum inspired encodings

I propose a discrete version of the quantum features described above, where I consider powers of the hamiltonian $(\hat{\mathcal{H}}^{XY})^p$ for integers p instead of continuous evolutions as explained in the previous paragraph. The discrete powers are not implemented natively on a quantum computer, hence the name *quantum-inspired*. They are however directly comparable to the RRWP scheme of (MLL+23), and they are cheaper to compute than the continuous quantum random walks. $\hat{\mathcal{H}}^{XY}$ is indeed of size $\mathcal{O}(|\mathcal{V}|^k)$ when restricted in the subspace of Hamming weight k, and is furthermore a sparse matrix for low density graphs, so the powers can be efficiently computed. We consider a *discrete* 2-particle quantum-inspired RW (2-QiRW) encoding that reads

$$\mathbf{P}_{ij} = \left[\langle ij| ((D_2^{XY})^{-1}\hat{\mathcal{H}}_2^{XY})^k |\psi_{\text{init}}\rangle \,|k \in [0,K]\right]_{ij} \tag{5.19}$$

where D_2^{XY} is the diagonal matrix sum of the rows of $\hat{\mathcal{H}}_2^{XY}$.

5.5 Post processing

In the previous sections, I described the features that can be obtained from a graph using a quantum computer. In this section, I will describe how these features can be used in a machine learning algorithm.

5.5.1 Quantum evolution kernel (QEK)

As a first example of algorithm, I will describe a way to construct a graph kernel using the probability distributions computed in Subsection 5.4.1. I contributed to the development of this method (HTDH21b) prior this thesis, and part of this thesis work is the hardware implementation of the procedure. We can naturally define a graph kernel by computing the distances between the probability distributions. There are many choices of distances between probability distributions. I will here use the Jensen-Shannon divergence (Lin91), which is commonly used in machine learning. Given two probability distributions $\mathcal{P}$ and $\mathcal{P}'$, the Jensen-Shannon divergence can be defined as

$$JS(\mathcal{P}, \mathcal{P}') = H\left(\frac{\mathcal{P} + \mathcal{P}'}{2}\right) - \frac{H(\mathcal{P}) + H(\mathcal{P}')}{2}, \tag{5.20}$$

where $H(\mathcal{P}) = -\sum_k p_k \log p_k$ is the Shannon entropy of $\mathcal{P}$. $JS(\mathcal{P}, \mathcal{P}')$ takes values in $[0, \log 2]$. In particular $JS(\mathcal{P}, \mathcal{P}) = 0$, and $JS(\mathcal{P}, \mathcal{P}') = \log 2$ is maximal if $\mathcal{P}$ and $\mathcal{P}'$ have disjoint supports. For two graphs $\mathcal{G}$ and $\mathcal{G}'$, and their respective probability distributions $\mathcal{P}$ and $\mathcal{P}'$ (computed as described in Subsection 5.4.1), I define the graph kernel as

$$\mathcal{K}_\mu(\mathcal{G}, \mathcal{G}') = \exp\left[-\mu\, JS(\mathcal{P}, \mathcal{P}')\right] \in \left[2^{-\mu}, 1\right]. \tag{5.21}$$

The kernel is then positive by construction (BH09). μ is a hyperparameter that can be tuned. For the following, I will set $\mu = 1$, but it might be helpful to adjust this value to improve the results.

We can then use this kernel in usual algorithms like SVM or KRR (see Section 4.2), and find the best dual coefficients α_i as defined in Equation (4.3) and Equation (4.7). The target of a new graph $\mathcal{G}$ is then predicted by computing its probability distribution $\mathcal{P}_{\mathcal{G}}^{\hat{\mathcal{O}}}$ and then computing its kernel values with respect to graphs in the training dataset. One should also optimize the quantum state to be constructed by finding the best time evolution parameters in Equation (5.12). Since the whole procedure is in general non differentiable, one would compute a cross validation score on a training set, and find the parameters that give the best score using gradient free methods. It is similar as the way one would find hyperparameters in classical machine learning models. One can think of randomized seach, bayesian optimization (Fra18), SPSA (HCT+19). In the case of neutral atoms quantum computers, methods have been developed to find the analog pulse (LH22) giving the best performances.

5.5.2 Graph Transformer with Quantum Correlations

This section presents GTQC (Graph Transformer with Quantum Correlations), an architecture of Graph Neural Network based on Graph Transformers and incorporating global graph features computed with quantum dynamics. A global view of the algorithm is represented on figure 5.2. Representation learning on graphs using neural network has become the state of the art of graph machine learning (WPC+20). Scaling deep learning models has brought lots of benefits as shown by the success of large language models (BMR+20; ADL+22). The goal was to bring the best of both worlds, meaning large overparameterized deep learning models, and structural graph features intractable with a classical computer. The architecture I propose only uses nodes features, but similar techniques could be implemented for edges features.

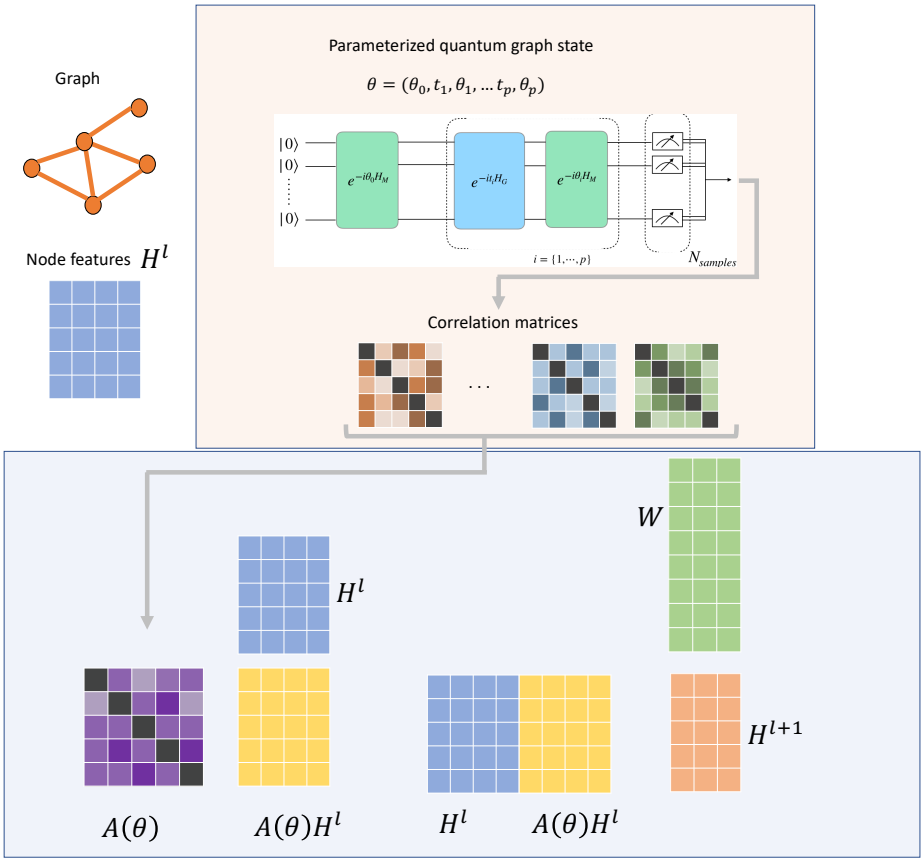


Figure 5.2: Overview of the model. The graph is translated into a hamitonian that will drive a quantum system. After a parameterized evolution, the correlations are measured and aggregated into a single attention matrix.

Here I develop a method to compute a parameterized transition matrix or quantum *attention* matrix from the correlations of a quantum dynamic. This matrix will later be used in the update mechanism of my architecture. Once the quantum attention matrix is computed, the rest of the architecture is purely classical, and all existing classical variations could be implemented.

Finally, the quantum attention matrix is by construction equivariant to a permutation of the nodes.

I consider a parameterized graph state as defined in equation 5.12, parameterized by the trainable parameter $\boldsymbol{\theta} = (\theta_0, t_0, \theta_1, t_1, \dots \theta_p)$, and noted $|\psi(\theta)\rangle$. $\boldsymbol{\theta}$ will be called the *quantum parameters* in the rest of the paper. I then compute for every pair of nodes (i, j) the vector of 2-bodies observables $C_{ij} = [\langle Z_iZ_j\rangle, \langle X_iX_j\rangle, \langle Y_iY_j\rangle, \langle X_iZ_j\rangle, \langle X_iY_j\rangle, \langle Y_iZ_j\rangle, \langle X_jZ_i\rangle, \langle X_jY_i\rangle, \langle Y_jZ_i\rangle]^T$ where $\langle O\rangle = \langle\psi(\theta)| O |\psi(\theta)\rangle$. These observables are all possible correlators.

The quantum attention matrix is computed by taking a linear combination of the previous correlation vector and optionally a softmax over the lines.

$$A(\theta)_{ij} = \gamma^T C_{ij} \tag{5.22}$$

$$A(\theta)_{ij} = softmax(\gamma^T C_{ij}) \tag{5.23}$$

where γ is a trainable vector of size 9. Multiple correlators are measured to enrich the number of features that can be extracted out of the quantum state. Limiting ourselves to e.g $\langle Z_iZ_j\rangle$ might be inefficient because $\langle\psi_f| Z_iZ_j |\psi_f\rangle$ could be written $XX^\dagger$ where X is a matrix with row i equal to $(Z_i |\psi_f\rangle)^\dagger$. The resulted weight matrix is therefore a symmetric positive semi-definite matrix, and can be reduced by a Choleski decomposition $A = LL^T$ where L is a real matrix of shape $N \times N$. The same model can then be constructed by learning the matrix L even though it is unclear if that would be efficient.

The quantum weight matrix previously computed is used as a transition matrix, or attention matrix, in the update mechanism of the model. Given H^l the node features matrix at the layer l, the node features matrix at the next layer is computed with the following formula :

$$H^{l+1} = \sigma((A(\theta)H^l||H^l)W) \tag{5.24}$$

where σ is a non-linearity, H is of size $(N \times d)$ where each row represents a node feature of dimension d, W is a learnable weight matrix of size $(2d \times d_h)$, $A(\theta)$ is the attention matrix with parameters θ computed in 5.5.2 and $||$ is the concatenation on the columns.

With the same approach as Transformers or Graph Attention Networks, one can use several attention heads per layer to improve the expressivity of the architectures. The update formula is given by:

$$H^{l+1} = \Big|\Big|_i^{N_{heads}} HEAD_i(H^l) \tag{5.25}$$

where each head is computed with the formula 5.5.2. The total dimension of the feature vector is $N_{heads}d_h$. Each head has a different quantum circuit attached, and can be computed in parallel if one possesses several QPUs.

We provide here some precisions on how the model would be implemented on real quantum devices.

Decoupling between quantum and classical parts

The parameters of the quantum states and the classical weight parameters are independent in the proposed algorithm. One can then asynchronously measure all the quantum states of the model and run the classical part. This may be particularly important for near term implementation since the access of QPUs are quite restricted in time. Furthermore, the gradients of the classical parameters depend only on the correlation matrices, so they can be easily computed with backpropagation without any supplementary circuit run.

Training the parameters of the quantum state

Computing the gradients of parameterized quantum circuits is a challenge source of numerous research in the quantum computing community (KE21; WIWL22; BC21). Finite-difference methods can be challenging to use because of the sampling noise of quantum measurements and the hardware noise. Some algorithms named parameter-shift rules were then created to circumvent this issue (MNKF18). In some cases, the derivative of a parameterized quantum state can be expressed as an exact difference between two other quantum states with the same architecture and other values of the parameters.

I detail here how I would compute the gradient in a simple case of the proposed architecture. Let $\hat{\mathcal{H}}$ be a hamiltonian, $\hat{\mathcal{O}}$ an observable, $|\psi_0\rangle$ an initial state. I introduce

$$|\psi(\theta)\rangle = U(\theta)\,|\psi_0\rangle = \exp\left(-i\theta\hat{\mathcal{H}}\right)|\psi_0\rangle \tag{5.26}$$

$$f(\theta) = \langle\psi(\theta)|\,\hat{\mathcal{O}}\,|\psi(\theta)\rangle \tag{5.27}$$

It is known (WIWL22) that f can be expressed as a trigonometric polynomial

$$f(\theta) = \sum_{\omega\in\Omega} a_\omega \cos(\omega\theta) + b_\omega \cos(\omega\theta) \tag{5.28}$$

where Ω is a finite set of frequencies depending on the eigenvalues of $\hat{\mathcal{H}}$. In the case of $\hat{\mathcal{H}} = \hat{\mathcal{H}}_M$ (see Equation (5.12)), the frequencies are the integers between 0 and N. One can then evaluate f on $2N+1$ points and solve the linear equations to determine $\{a_\omega\}$, $\{b_\omega\}$ and the derivative of f. This is the same for the $\hat{\mathcal{H}}^I$ hamiltonian which associated frequencies are the integers between 0 and $|\mathcal{E}|$.

For the other hamiltonians listed, there is no known analytical formula to compute the gradients like previously. The only method would be to use finite difference schemes which are sensitive to noise.

Random parameters

Optimizing over the quantum parameters can be costly and ineffective with current quantum hardware. Even with emulation, back-propagating the loss through a system of more than 20 qubits is very difficult. I encounter memory errors for more than 21 qubits on A100 GPUs, even though the implementation is certainly not optimal. Therefore I propose an alternative scheme to the model, to help with both actual hardware implementations and classical emulation.

The main idea in the spirit of (RR08b) is to evaluate the attention matrices on many random quantum parameters, and only training the classical weights. From a model $f(x; W, \theta) = \Big|\Big|_i^{N_{heads}} \sigma((A(\theta_i)H^l||H^l)W_i)$ with one layer, I would normally find the parameters that minimize a loss between inputs x and labels y

$$W^*, \theta^* = \arg\max_{W,\theta} \sum_{i=1}^{M} l(f(x_i; W, \theta), y_i) \tag{5.29}$$

Instead, I create a model with more heads and fixed random values. θ' expressed as $f(x; W, \theta') = \Big|\Big|_i^{N'_{heads}} \sigma((A(\theta'_i)H^l||H^l)W_i)$ and I minimize only on θ

$$W^* = \arg\max_{W} \sum_{i=1}^{M} l(f(x_i; W, \theta'), y_i) \tag{5.30}$$

5.5.3 Positional encodings with quantum features

In this section, I detail my proposals to incorporate quantum features in GNN models and discuss the potential benefits and drawbacks. I focus on two types of encoding: the first uses the ground state of the graph as defined in section Subsection 5.3.1, the second uses the XY hamiltonian on the k-particles subspace (detailed in Subsection 5.4.3). Both methods use a quantum state that is difficult to prepare in the general case, so I expect to obtain features that are not available with classical approaches. For details of the corresponding algorithms, see Appendix A.8.

5.5.3.1 Eigenvectors of the correlation on the ground state.

I propose to use the correlation matrix $C_{ij} = \langle Z_i Z_j \rangle$ on the ground state of the graph defined in Subsection 5.3.1. Since this matrix is symmetric with nonnegative eigenvalues, it can formally be used in the same place as the Laplacian matrix in graph learning models. Hence, I use the eigenvectors of this correlation matrix in the same way Laplacian eigenvectors (LE) are used in other architectures of graph transformers. Instead of taking the eigenvectors with the lowest eigenvalues as for the Laplacian eigenmaps, I take the ones with highest eigenvalues, since they are the ones in which most of the information about the correlation matrix is contained. I expect to face the same challenges due to the sign ambiguity (DLL+21; KBH+21), and to implement the same techniques to alleviate them (LRZ+22).

5.6 Theory

I present in this section the main theoretical results I established regarding the expressive power of the quantum algorithms. In this section, I aim to investigate the theoretical properties of

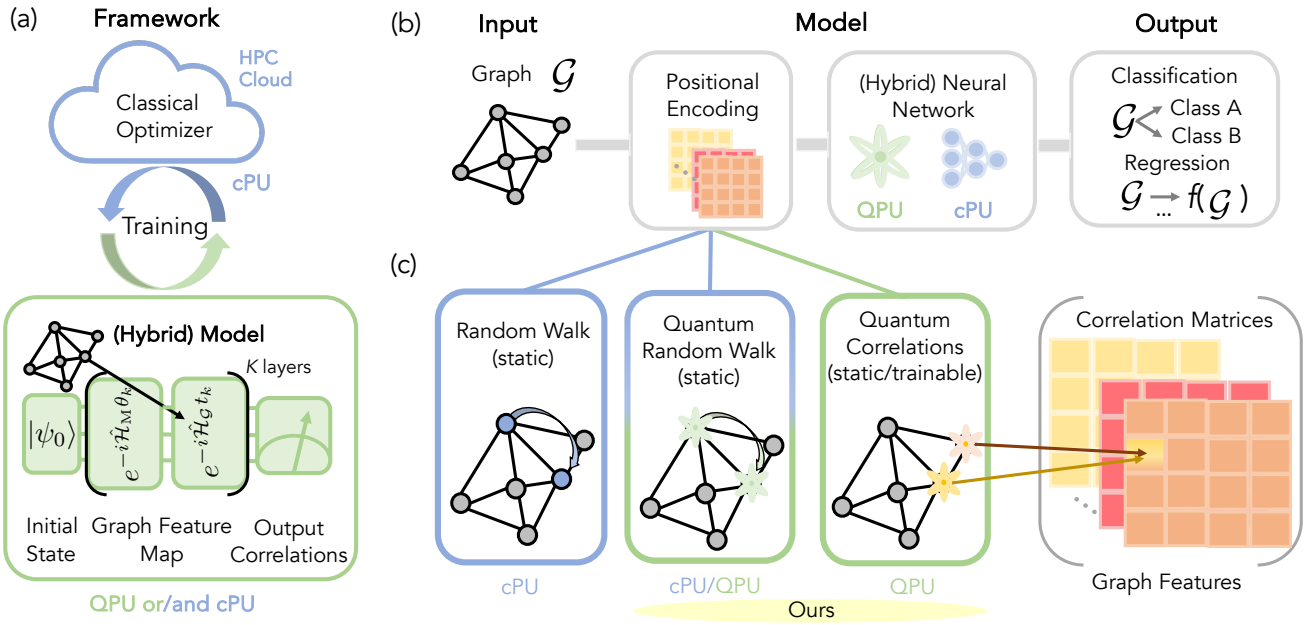


Figure 5.3: Summary of the proposed method. **(a)** The hybrid quantum-classical framework utilizes a classical computer for parameter optimization (if required) and employs a hybrid model using a Quantum Processing Unit (QPU) and a CPU and/or GPU, denoted as classical Processing Unit (cPU). In the quantum graph NN, I initialize QPU at a quantum state $|\psi_0\rangle$, apply a mixing Hamiltonian $\hat{\mathcal{H}}_M$ evolution for a duration θ, and utilize a Hamiltonian $\hat{\mathcal{H}}_{\mathcal{G}}$ evolution for the graph feature map with a duration t. K layers are used to obtain a sufficiently expressive quantum model. Finally, the output is obtained by measuring correlators, e.g., $\langle Z_i Z_j \rangle$. See Section 5.4.2 for details. **(b)** Static or trainable PE is constructed for a graph $\mathcal{G}$ via **(c)** (quantum) random walk (static PE) or a quantum graph NN (static/trainable PE), which computes quantum correlations. Note that the proposed PEs are not restricted to classical models (such as the transformer studied in this work) but are also applicable to all quantum models.

the quantum walk features. I show that they are able to distinguish strongly regular graphs whereas usual positional encoding features cannot.

5.6.1 Quantum walks and strongly regular graphs

Definition 5.1. *A strongly regular graph (SRG), noted $srg(\nu, k, \lambda, \mu)$, is a graph with ν vertices of fixed degree k, such that every pair of adjacent vertices have a fixed number λ of common neighbors, and every pair of non-adjacent vertices have a fixed number μ of common neighbors.*

Each tuple (ν, k, λ, μ) defines a family of SRGs, and it is possible to find multiple non isomorphic graphs within the same family (Spe24). We chose to work on these particular graphs because their regularity makes them especially difficult to distinguish within the same family. For instance, (ZLWH23) provide a worst-case analysis for the GD-WL test, a provably more

powerful version of 1-WL , in the case of distance-regular graphs. Their examples include Rook's and Shrikhande graphs, both of diameter 2. We know on the other hand that a SRG is a distance-regular graph with a diameter 2 when $\mu \neq 0$ (Big93). In (MLL+23), proposition 3.2., the authors show that GD-WL with RRWP distance is strictly more powerful than GD-WL with shortest path distance. They test their approach on 2 distance regular graphs. In the following, we propose to analyze the distinguishability of SRGs through the GD-WL test with RRWP distance, as well as a test involving a particular case of Hamiltonian evolution by the Ising model, constituting one of the rare configurations where it is possible to extract generic formulas without any costly simulation or quantum computing.

Proposition 5.1. *(BFW+21) It requires a 3-WL test or higher to distinguish two non-isomorphic strongly regular graphs from the same family.*

This result highlights the difficulty of the task, as a 3-WL test requires overlapping information from all the triplets in the graph, and is therefore costly in terms of memory and time as the size of the graph analyzed increases. We continue in the same line with the following result:

Proposition 5.2. *GD-WL with RRWP distance cannot, even with eigen-decomposition of the distance matrix, distinguish any two non isomorphic SRGs from the same family.*

Proof. One can show (SJ03; GFZ+10) that for strongly regular graphs, the powers of the adjacency matrix A can be expressed as

$$A^n = \alpha_n I + \beta_n J + \gamma_n A$$

where α_n, β_n, γ_n only depend on N, k, λ, μ. I is the identity matrix, J is the matrix full of 1s. The degree matrix D is also equal to kI, then $(D^{-1}A)^n = A^n/k^n$. Hence the information about distance contained in $\mathbf{P}_{uv}$ for strongly regular graphs is the same as in their adjacency matrices. Therefore, for strongly regular graphs, the GD-WL with RRWP test is equivalent to the WL test. The F.I. consists of eigenvalues and eigenvectors. It has been shown that F.I is stronger than 1-WL, but weaker than 2-WL (RS). Therefore, *GD*-WL along with eigenvectors cannot distinguish SRGs, which require 3-WL or higher. □

This extends to the RRWP distance the results recovered in (ZLWH23), in which they show the same for shortest path distance and resistance distance.We provide in the next section a set of experiments as empirical evidence that show that in some cases, a GD-WL with correlations on 2-QW distinguishes SRGs.

5.6.2 Empirical study : Ising and XY models for the distinguishability of SRGs

This result shows that the case in which it is possible to derive formal expressions actually fails to distinguish non isomorphic SRGs. We have proven that in the previous subsection that

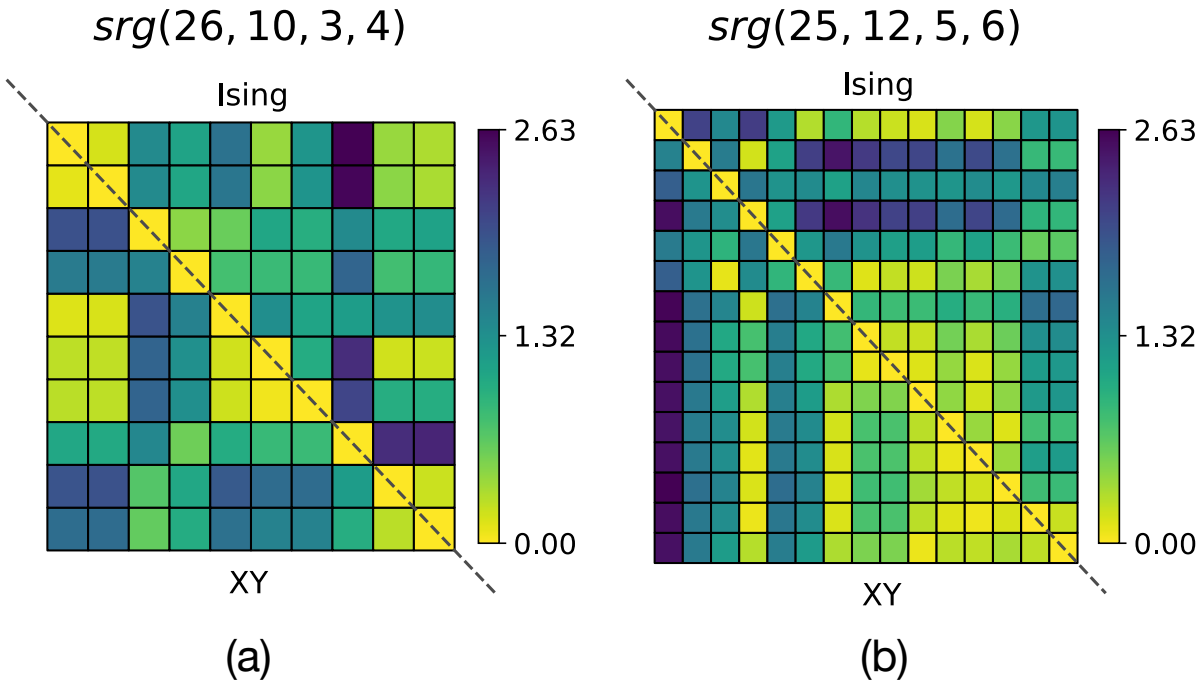


Figure 5.4: Normalized values of Ising correlations (upper triangular part) and XY model correlations (lower triangular part) for (a) the family $srg(26, 10, 3, 4)$ that includes 10 non-isomorphic graphs, and (b) the family of $srg(25, 12, 5, 6)$ that includes 15 non-isomorphic graphs.

the GD-WL test cannot distinguish non isomorphic SRGs. Figure 5.4 shows the results of the distinguishability in a set of 2 families, with 25 and 26 nodes. We compute both Ising and XY-hamiltonian evolutions, for $p = 2$ in the former and for a 2-QW in the latter. We use the following permutation invariant measure to compute the distance between a pair of graphs $d(G_i, G_j) = \frac{1}{2}\sum_{k=1}^{\nu^2} ||S(C(G_i)) - S(C(G_j)||_k$ where C is the correlation matrix, $S(M)$ a function that receives a matrix $M \in \mathbb{R}^{N\times N}$ as input and returns a vector $m \in \mathbb{R}^{N^2}$ containing the sorted elements of M. Using this formula, a non- zero value implies that the two graphs are not isomorphic, but the opposite is not necessarily true. We can see from this figure that it is possible to distinguish these graphs. We also ran experiments to verify that the distance between any of these graphs and a set of 5 randomly selected isomorphic counterparts is zero as expected.This shows empirically that for certain data sets, a two layers Ising evolution and a 2-QW are strictly more powerful than 2-WL. We also run the GD-WL test with the edge features $S(M)$ and we obtain that all pairs of graphs are successfully distinguished. Finally, it is important to point out that the classical k-WL test requires comparisons between pairs of subsets of nodes of size k, rendering its complexity to at least N^k, N being the size of the graph. On the other hand, for quantum evolutions, the complexity of the algorithm is characterized by the number of shots that need to be measured in order to reconstruct the distribution of the

desired observable. The number of shots is $\mathcal{O}(\frac{1}{\epsilon^2})$ for a precision up to ϵ (HKP20). This number does not increase with the number of layers in the Ising hamiltonian case, where we only have a linear increase in the evolution time (which is short in practice), or with the number of quantum walkers that only depends on the initial input state preparation. This gives our approach an attractive potential for quantum advantage, albeit limited to datasets on which large values of k in the k-WL test are relevant. This also assumes that the k-QW as well as the k-layers Ising, are both strictly more powerful than the k-WL for any values of k. This property has been demonstrated in some cases for $k = 1$ and observed for $k = 2$ in the case of SRGs, but not yet demonstrated in the general case and for any value of k. Our comparison is however limited to graph neural networks of the message passing neural networks and graph transformers family.

Chapter 6

Experiments

In this chapter, I detail the experimental implementations of the algorithms described in Chapter 5. Unlike the previous part, this chapter is written from different parts of the following papers:

- (ADL+23) Boris Albrecht, Constantin Dalyac, Lucas Leclerc, Luis Ortiz-Gutiérrez, Slimane Thabet, Mauro D'Arcangelo, Julia RK Cline, Vincent E Elfving, Lucas Lassablière, Henrique Silvério, Bruno Ximenez, Louis-Paul Henry, Adrien Signoles, Loïc Henriet. (2022) *Quantum feature maps for graph machine learning on a neutral atom quantum processor.* Physical Review A, 107(4), 042615.

- (TFH22) Slimane Thabet, Romain Fouilland, Loic Henriet (2022). *Extending Graph Transformers with Quantum Computed Correlations arXiv:2210.10610*

- (TDS+) Slimane Thabet, Mehdi Djellabi, Igor Sokolov, Sachin Kasture, Louis-Paul Henry, Loïc Henriet (2024). *Quantum Positional Encodings for Graph Neural Networks.* International Conference on Machine Learning 2024

I show numerically that the performance of state-of-the-art models can be improved on standard benchmarks and large-scale datasets by computing tractable versions of quantum features. Our findings highlight the potential of leveraging quantum computing capabilities to enhance the performance of transformers in handling graph data. In Section 6.1 and Section 6.2, I present numerical experiments of the transformer based algorithms, and in Section 6.3 I analyze the results of the experimental implementation of graph kernels done in collaboration

with the Pasqal hardware team. To that end, we introduce a quantum feature map to encode the information about graphs in the parameters of a tunable Hamiltonian acting on an array of qubits. Using this tool, we first show that interactions in the quantum system can be used to distinguish non isomorphic graphs that are locally equivalent. We then realize a toxicity screening experiment, consisting of a binary classification protocol on a biochemistry data set comprising 286 molecules of sizes ranging from 2 to 32 nodes, and obtain results which are comparable to the implementation of the best classical kernels on the same data set. Using techniques to compare the geometry of the feature spaces associated with kernel methods, we then show evidence that the quantum feature map perceives data in an original way, which is hard to replicate using classical kernels.

It remains unclear whether an actual performance improvement will be observed on real-world use cases, and whether this improvement will justify the cost of using quantum hardware in the future.

6.1 Numerical experiments on graph transformers

I benchmarked our model GTQC (see Subsection 5.5.2) and its randomized version with different GNN architectures. I selected different datasets from various topics and with diverse tasks to show the general capabilities of our approach. I limited the size of the graphs to 20 nodes in order to be able to simulate the quantum dynamics, and performing a backpropagation. I then chose datasets with the majority of graphs falling below this size limit. The details of each dataset can be found in Appendix A.9.1.

I implemented the models with a classical emulator of quantum circuits implemented in *pytorch* (PGM+19) and with *dgl* (WZY+19), and ran them on A100 GPUs. All experiments were done using one GPU, except QM9 which required 4. We used a Adam optimizer with a learning rate .001, no weight decay for 500 epochs. The quantum parameters were only updated every 10 epochs because of computation time. As an order of magnitude, one epoch of QM7 takes 6 min with 1 GPU, and one epoch of QM9 takes 1h with 4 GPUs. Most of the time is allocated to compute the quantum dynamics, the size of classical parameters has little effect on the compute time except for the 2048 hidden layers. I do not consider any shot noise or potential hardware noise in the experiments.

I compare my model to three architectures of message passing models : GCN (KW16), SAGE (HYL18), GAT (VCC+18). In order to have a fair comparison between the models, we employ similar hyperparameters for all of them. I use ReLU function for all activation functions. Each model has 2 layers and 1 head for multi-head ones like ours and GAT, except on the randomized version of our model. I don't use softmax as described in Subsection 5.5.2, except for the randomized instance because I observed the training was more stable without using it. We also report the results by comparing models that have the same number of hidden

neurons per layer and therefore approximately the same number of parameters. Each dataset is randomly split in train, validation, test with respective ratios of .8, .1, .1 and the models are run on 5 seeds, except QM9 for which we have only one seed.

All metrics are such that the lower the better, for classification tasks we display the ratio of misclassified objects. For QM9 the loss is aggregated over all the targets. Figure 6.1 shows results as box plots accompanied by the underlying points. Though the variability of the results is a bit higher, GTQC reaches similar results than usual classical well-known approaches on QM7 and DBLP_v1 (even beating GCN on this dataset) and seems relevant on QM9 as well. GTQC random usually outperforms GTQC, illustrating the complexity of the optimisation of the quantum system. GTQC random provides very promising results on DBLP_v1 and outperforms all other approaches on QM9. Letter-med seems to represent a difficult task for our quantum methods as they both perform very poorly and way worse than classical methods. QM7 also seems to be challenging for GTQC random. Table 6.1 shows the results grouped by breadth, compared to the results for GTQC with the same breadth and averaged other various dataset splits. GTQC random has only been trained for breadth of 128 neurons and yields impressive results by clearly outperforming other methods on QM9 and being on par with the best method on DBLP_v1.

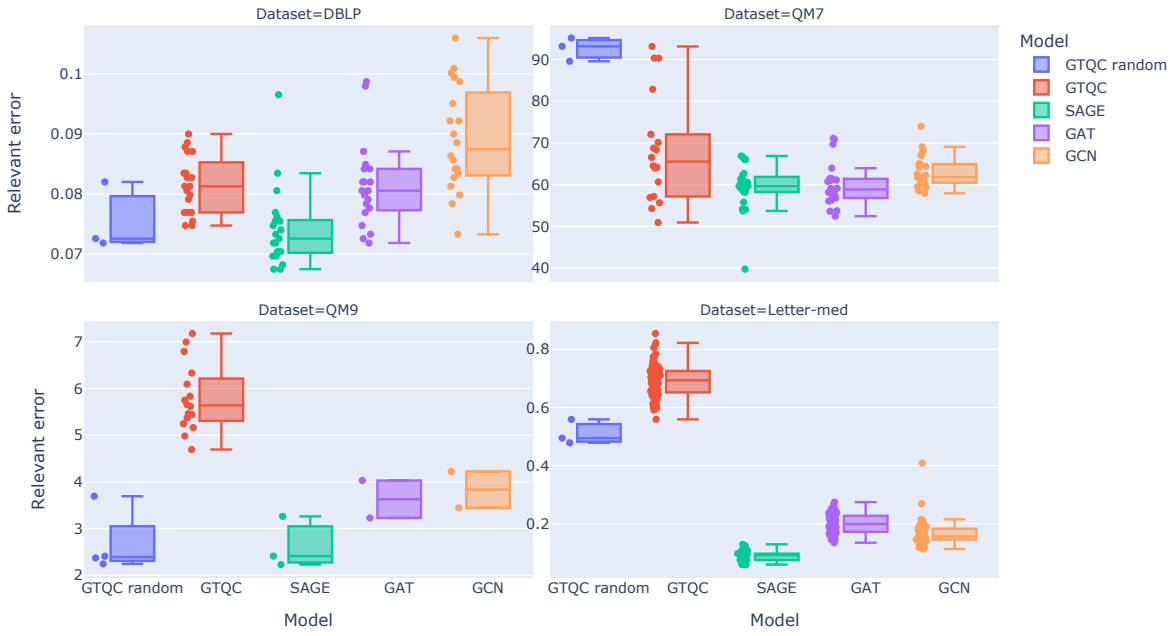


Figure 6.1: Summary graph of the results of the different models on the studied datasets. Each point is an instance of the model with specific hyperparameters and specific seed for dataset splits.

Table 6.1: Relative best test error compared to GTQC, grouped by breadth of the hidden layer. We colored in blue the classical models for which at least one of GTQC or GTQC random is better. The standard deviations are over different splits of the datasets.

model	dataset breadth	DBLP	Letter-med	QM7	QM9
GAT	128	0.08 ± 0.00	0.22 ± 0.02	64.27 ± 6.40	4.03 ± 0.00
	512	0.08 ± 0.01	0.20 ± 0.03	60.42 ± 5.56	3.23 ± 0.00
	1024	0.08 ± 0.01	0.21 ± 0.04	56.06 ± 2.77	
	2048	0.08 ± 0.00	0.19 ± 0.04	59.27 ± 3.33	
GCN	128	0.09 ± 0.01	0.21 ± 0.05	67.50 ± 4.70	4.22 ± 0.00
	512	0.09 ± 0.01	0.16 ± 0.02	63.30 ± 2.67	3.44 ± 0.00
	1024	0.09 ± 0.01	0.15 ± 0.02	62.94 ± 2.90	
	2048	0.09 ± 0.01	0.14 ± 0.02	60.70 ± 2.30	
GTQC	128	0.08 ± 0.00	0.71 ± 0.06	68.95 ± 8.50	8.01 ± 0.00
	512	0.08 ± 0.00	0.66 ± 0.04	70.09 ± 18.48	6.23 ± 0.00
	1024	0.08 ± 0.01	0.67 ± 0.06	67.85 ± 16.78	5.27 ± 0.00
	2048	0.08 ± 0.00	0.73 ± 0.06	65.53 ± 2.48	5.30 ± 0.00
GTQC random	32		0.56 ± 0.00		
	64		0.48 ± 0.00		
	128	0.08 ± 0.01	0.49 ± 0.00	92.62 ± 2.82	2.34 ± 0.00
SAGE	128	0.08 ± 0.01	0.09 ± 0.02	62.30 ± 3.47	3.26 ± 0.00
	512	0.07 ± 0.00	0.09 ± 0.02	61.47 ± 3.06	2.41 ± 0.00
	1024	0.07 ± 0.00	0.09 ± 0.01	61.30 ± 3.07	2.23 ± 0.00
	2048	0.08 ± 0.01	0.09 ± 0.02	54.41 ± 6.42	

6.2 Numerical experiments on positional encodings

6.2.1 Experiments on random walk models

In this subsection, I test concatenating the QRW encodings (see Subsection 5.4.3, Subsection 5.4.4) to the RRWP in the GRIT model (MLL+23). The (continuous) 1-CQRW for K random times and the discrete 2-QiRW are computed for K steps. Those encodings are computed numerically since they are still tractable for graphs below 200 nodes compared to the higher order k-QiRW ones. I benchmark our method on 7 datasets from (DJL+20), following the experimental setup of (RGD+22) and (MLL+23). The method is compared to many other architectures and the results directly taken from (MLL+23). I do not perform an extensive hyperparameter search for each architecture and only run myself the GRIT model by taking the same hyperparameters as the authors. The experiments are done by building on the codebase of (MLL+23) which is itself built on (RGD+22). More details about the datasets can be found in Appendix A.9.1. The results are included in Table 6.2. The proposed methods performs better on ZINC, MNIST and CIFAR10 than all others, and comes second for PATTERN and CLUSTER. I also benchmark my methods on large-scale datasets, ZINC-full (a bigger

version of ZINC (ISM+12)) and PCQM4MV2 (HFR+21). For these datasets, I run a variety of models (GINE, GatedGCN, and GRIT) with different position encodings (LE, RRWP, 2-QiRW, and a mix of RRWP and 2-QiRW. The results are reported figure 6.2 and the full numbers are reported in Appendix A.11. Quantum features perform better for all models in the case of ZINC-full and for some models of PCQM4Mv2. All hyperparameters for this sections are reported in the appendix in Appendix A.10.

Table 6.2: Test performance in five benchmarks from (DJL+20). We show the mean ± s.d. of 4 runs with different random seeds as in (MLL+23). Highlighted are the top first, second, and third results. Models are restricted to $\sim 500K$ parameters for ZINC, PATTERN, CLUSTER $\sim 100K$ for MNIST and CIFAR10. We compare our model to our run of GRIT and indicate the results obtained by the authors for information. Figures other than the last 3 lines are taken from (MLL+23). Models in bold are our models.

Model	ZINC	MNIST	CIFAR10	PATTERN	CLUSTER
	MAE↓	Accuracy↑	Accuracy↑	Accuracy↑	Accuracy↑
GIN	0.526 ± 0.051	96.485 ± 0.252	55.255 ± 1.527	85.387 ± 0.136	64.716 ± 1.553
GatedGCN	0.282 ± 0.015	97.340 ± 0.143	67.312 ± 0.311	85.568 ± 0.088	73.840 ± 0.326
EGT	0.108 ± 0.009	**98.173 ± 0.087**	68.702 ± 0.409	86.821 ± 0.020	**79.232 ± 0.348**
GPS	0.070 ± 0.004	98.051 ± 0.126	72.298 ± 0.356	86.685 ± 0.059	78.016 ± 0.180
GRIT (our run)	**0.060 ± 0.002**	**98.164 ± 0.054**	**76.198 ± 0.744**	**90.405 ± 0.232**	**79.856 ± 0.156**
GRIT 1-CQRW	**0.058 ± 0.002**	98.108 ± 0.111	**76.347 ± 0.704**	**87.205 ± 0.040**	78.895 ± 0.1145
GRIT 2-QiRW	**0.059 ± 0.004**	**98.204 ± 0.048**	**76.442 ± 1.07**	**90.165 ± 0.446**	**79.777 ± 0.171**

6.2.2 Synthetic experiments

In this section, I provide one example of dataset with a binary graph classification task for which the use of the correlation matrix on the ground state as defined in 5.5.3.1 is more powerful than other commonly used features like the laplacian eigenvectors or RRWP. The idea is to construct graphs that will exhibit very different Ising ground states but similar spectral properties or random walk transition probabilities. I illustrate the differences between the encodings in the appendix A.9.2. I train classical models like GINE, GatedGCN, and GRIT on these datasets with LEs and RRWP as node features and edge features, and I compare it to a simple GCN model with the eigenvectors of the correlation matrix as node features.

I train the GCN model for 200 epochs using the Adam optimizer, 0.001 learning rate, no weight decay. I split randomly the dataset on train/validation/test with a proportion 0.8/0.1/0.1, and I measure the test accuracy of the model having the highest validation accuracy. For the other models we use the same hyperparameters as in table A.8, but with hidden dimensions of 32 for normal models and 64 for big models. I also use a dimension 20 for all positional encodings, and initialize with uniform node and edge features full of 1s.

I also benchmark the GRIT model with RRWP. The results are shown in table 6.3. The quantum encoding models achieve 100% accuracy whereas all other classical models achieve

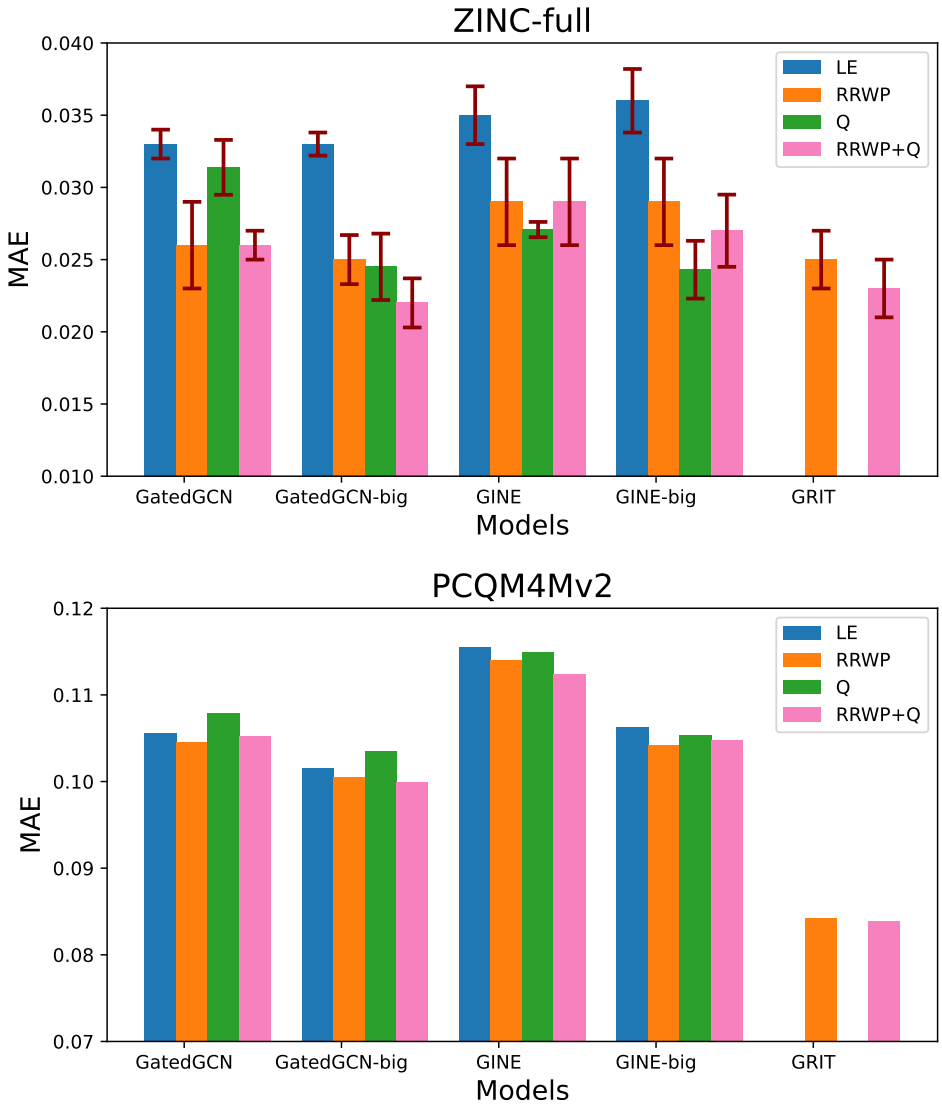


Figure 6.2: Test performance (mean absolute error) of different models with different positional encodings on large scale datasets. Q: 2-QiQRW features, RRWP+Q: RRWP and 2-QiQRW concatenated. Top : ZINC-full . Bottom: PCQM4Mv2. For ZINC-full, we show the mean and s.d of 4 runs with different random seeds. For PCQM4Mv2 we show the output of a single run.GRIT has 500k parameters for ZINC-full and 11.8M for PCQM4MV2. Normal models have about 200k parameters and big models about 700M.

45% accuracy.

6.2.3 Discussion

I performed several experiments comparing the quantum encodings to the classical ones. Including the quantum walk features into state-of-the-art models improves their performances

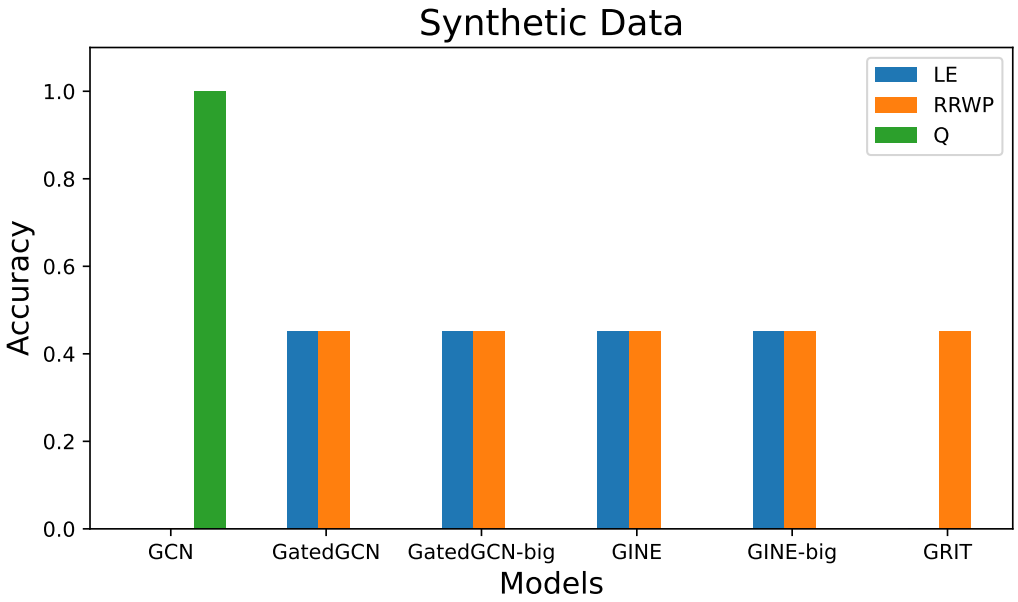


Figure 6.3: Results on synthetic data. We show the accuracy on the test set. Q: eigenvectors of the correlation on the ground state. GCN model has about 4k parameters, GRIT model has 500k parameters, other normal models have about 10k parameters, big models have about 60k parameters.

on most of the datasets tested. It is not surprising that the method works well for datasets such as ZINC for which random walks are known to provide relevant features (RGD+22). I only experimented on versions of quantum features that can be easily computed with classical computers and we were able to show a small gain in performance compared to state-of-the-art models. It is then plausible that using quantum features that cannot be classically accessible could lead to a greater improvement of models, if quantum hardware can be made widely available. I was able to engineer an artificial dataset for which classical approaches fail to perform the associated binary classification tasks and the quantum encoding perfectly realizes it, even with only 4k parameters where classical models have between 10k and 500k.

6.3 Hardware implementation of quantum feature maps

In this section, we describe implementation of the methods developed in Subsection 5.3.2, Subsection 5.4.1, and Subsection 5.5.1 on the neutral atom quantum hardware of Pasqal. The actual experiment was performed by the experimental team of the company, in this section I describe the protocol and the results.

Starting from a unit disk (UD) graph $\mathcal{G}$ reproduced in the array of tweezers with qubits all starting in $|0\rangle$, we apply a parameterized laser pulse onto the atoms in order to generate a

wavefunction $|\psi_{\mathcal{G}}\rangle$ of the form

$$|\psi_{\mathcal{G}}\rangle = U(\mathcal{G};t)\,|0\rangle^{\otimes|\mathcal{G}|}, \tag{6.1}$$

where we define the time-evolution operator $U(\mathcal{G};t) = \mathcal{T}\left[\exp\left(-i/\hbar \int_{s=0}^{t} \hat{\mathcal{H}}_{\mathcal{G}}(s)ds\right)\right]$ to be our *quantum feature map* unitary for graph structured data. We will restrict ourselves to laser pulses with constant detuning δ and Rabi frequency Ω, with an adjustable duration t (see Subsection 4.6.2). Depending on the task at hand, we consider various observables $\hat{O}$ to evaluate on $|\psi_{\mathcal{G}}\rangle$. Measurements of a site-dependent (respectively global) observable give rise to a probability distribution $\mathcal{P}$ which is node (graph) specific and can be used for various machine learning tasks at the node (graph) level.

In Subsection 6.3.1, we show theoretically and experimentally that the graph quantum feature map already shows interesting properties when associated with local or global observables built from single-body expectation values $\langle \hat{O}_{j=1,\dots,|\mathcal{G}|}\rangle$.

In the rest of the section, we implement a graph classification task corresponding to the prediction of toxicity of molecules on the dataset PTC-FM (HKKS01).

6.3.1 Insight on a quantum feature map

The graph quantum feature map already shows interesting properties when associated with single-body observables $\langle \hat{O}_{j=1,\dots,|\mathcal{G}|}\rangle$. The measured values are not only affected by local graph properties such as node degrees, but also by more global ones such as the presence of cycles. This enrichment provided by the quantum dynamics contrasts with the locality of node representations in many classical graph machine learning algorithms. This key feature comes from the fact that the quantum dynamics of a given spin model (*e.g.* an Ising model) will be significantly influenced, beyond short times (given by the Lieb-Robinson bound (LR72; TGS+19)), by the complete structure of the graph.

We illustrate experimentally this behavior for two graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ that are non-isomorphic but locally identical. In these graphs, nodes can be separated into two equivalence classes according to their neighborhood: border nodes B have one degree-3 neighbor and one degree-2 neighbor, while center nodes C have two degree-2 neighbors and one degree-3 neighbor (see Figure 6.5**a**). We will see that the presence of interactions will enable us to discriminate between $\mathcal{G}_1$ and $\mathcal{G}_2$ by comparing the dynamics of local observables on border and center nodes.

We first map the graphs in a tweezers array with a nearest-neighbor (NN) distance of $r_{NN} = 5.3\,\mu$m and apply a constant pulse with $\Omega/2\pi = 1.0\,$MHz and $\delta/2\pi = 0.7\,$MHz. We then measure the local mean Rydberg excitation $\langle n_j\rangle_{j\in B/C}$ for varying pulse duration $t \in [0, 2.5]\,\mu$s. As illustrated in Figure 6.5**b**, a qualitative difference in the dynamics of both graph appears after $t \sim 0.25\,\mu$s. Precisely, the excitation of the border nodes (see Figure 6.5**b**, left panel) is initially increasing with indistinguishable behavior between the two graphs. Then, a distinction appears between the two graph instances. The mean density for the border qubits of $\mathcal{G}_1$ exhibits damped oscillations around $\langle n_B\rangle \sim 0.15$ with period of the order of $0.5\,\mu$s while for $\mathcal{G}_2$ it

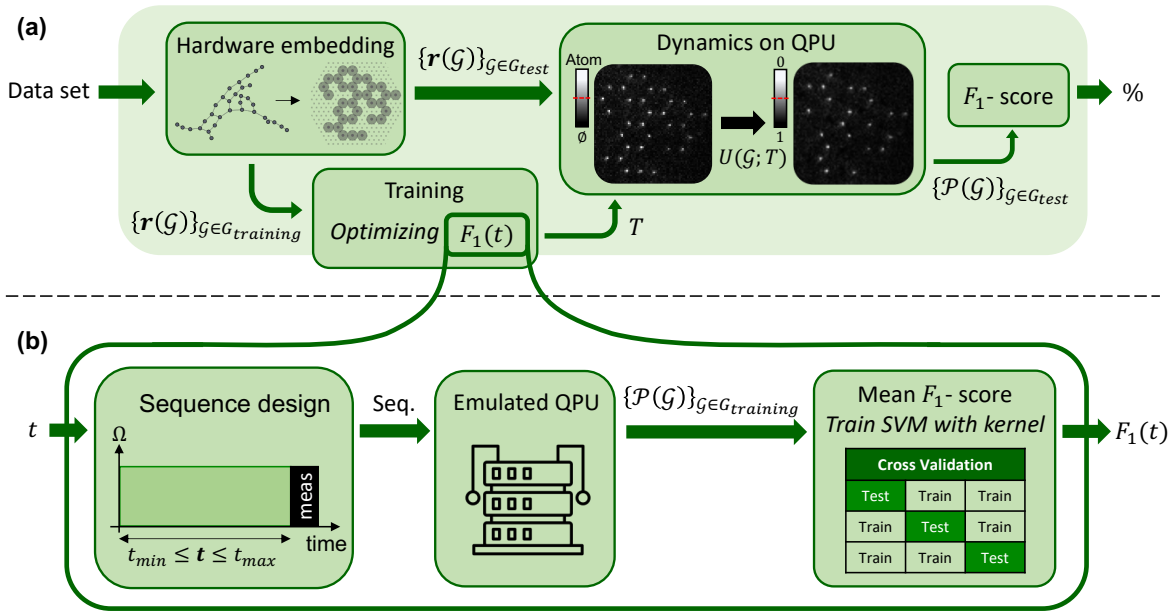


Figure 6.4: **a**. A dataset of graphs $\mathcal{G}$ is first mapped onto atomic registers $\mathbf{r}(\mathcal{G})$ implementable on the QPU, and separated between a training set $\mathcal{G}_{training}$ and a test set $\mathcal{G}_{test}$. We use the training set to determine numerically the optimal pulse sequence to be applied on the hardware using a grid search algorithm for optimizing $F_1(t)$ (see **b**). This training phase outputs the optimal parameter T used to design the laser-pulse sequence applied experimentally on each register of the test set. The resulting dynamics performed on the QPU generates $U(\mathcal{G};T)$, driving the system from $|0\rangle^{\otimes\mathcal{G}}$ to $|\psi_{\mathcal{G}}\rangle$. F_1 is then derived from the measured probability distributions $\{\mathcal{P}(\mathcal{G})\}_{\mathcal{G}\in G_{test}}$. **b.** The optimization of the score function F_1 during the training includes several steps. The input t, taken from the parameter space $[t_{min}, t_{max}]$ defines a laser sequence with Ω and δ fixed parameters followed by a measurement. The dynamics of the system is emulated and enables us to compute the probability distributions associated to this particular value of t for the whole training part of the dataset. Finally, $F_1(t)$ is obtained by fitting the SVM with the kernel constructed from those probability distributions.

exhibits flatter oscillations centered around 0.25 with period around 1 μs. We can observe a comparable distinction between the two graphs for the center qubits (see Figure 6.5**b**, right panel). The experimental measurements are consistent with the theoretical predictions and with the expected level of noise (see Appendix A.12.2 for more details).

When restricted to the mean-field approximation (or similarly in the classical limit), the qubits' dynamics on either graphs are far more similar, as illustrated in the insets of panels in Figure 6.5b. We still observe distinct dynamics between the two graphs, which is due to next nearest neighbors (NNN) interactions (more pronounced for the center nodes). If we neglected those NNN interactions, the mean-field equations governing the dynamics of each qubit would only depend on its direct neighborhood, i.e. the local structure of the graph. In that case, the qubits dynamics for $\mathcal{G}_1$ and $\mathcal{G}_2$ obey the exact same equations (see black dashed line in the

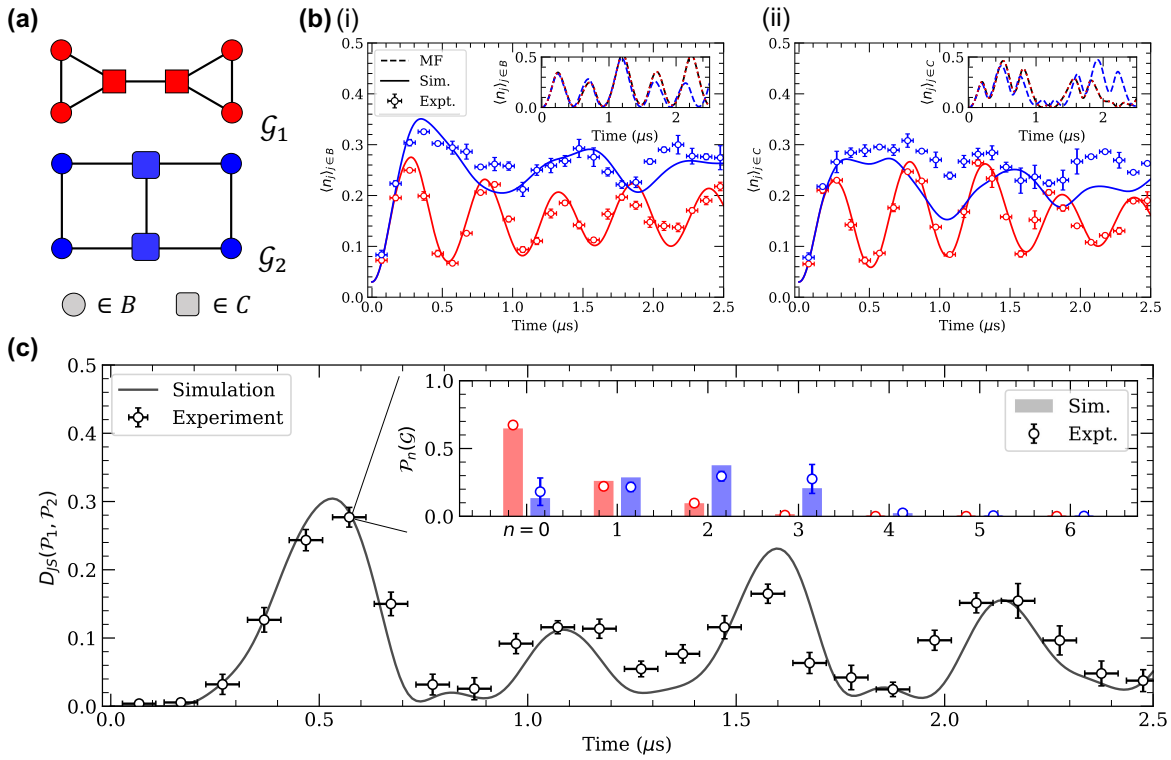


Figure 6.5: **a**. $\mathcal{G}_1$ (red) and $\mathcal{G}_2$ (blue) are two different graphs with identical local structure. Based on their neighborhood, the nodes either belong to the *border* B (circle) or to the *center* C (square). **b**. We plot the evolution of the mean occupation $\langle n_i \rangle$ of the two regions B (left) and C (right) for both graphs $\mathcal{G}_1$ (red) and $\mathcal{G}_2$ (blue). The dots represent the experimental results while the full curves show noisy simulation results. Horizontal error bars account for the sequence-trigger uncertainty (≈ 40ns) while the vertical ones account for the sampling noise. The insets show the corresponding mean field dynamics (dashed) with only NN (black) or full (colored) interactions. **c**. The evolution of the Jensen-Shannon divergence obtained experimentally (dot) is compared to the noisy simulation (plain). At each point in time, $JS(\mathcal{P}_1, \mathcal{P}_2)$ is computed using the excitation distributions $\mathcal{P}_{1/2} = \{\mathcal{P}_n(\mathcal{G}_{1/2})\}_{n=0\ldots6}$ obtained either numerically (bar) or experimentally (dot). The inset depicts $\mathcal{P}_{1/2}$ obtained at $t \approx 0.57$ μs which yields the maximum value $JS_{max} \approx 0.28$ reached.

insets of Figure 6.5b). We therefore conclude that the presence of interactions in the system enables us to discriminate between the two non-isomorphic graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ by evaluating node-level local observables $\langle n_B \rangle$ or $\langle n_C \rangle$.

6.3.2 Dataset and mapping on hardware

In this section, we describe the realization of a toxicity screening experiment, consisting of a binary classification protocol on the Predictive Toxicology Challenge on Female Mice (PTC-FM)

dataset (HKKS01) comprising 286 molecules of sizes ranging from 2 to 32 nodes. Between 200 and 1000 usable shots (meaning with no rearranging defects) per graphs were taken, with a data acquisition phase that lasted 3 days. In the original PTC-FM dataset, the 349 molecules are represented under the form of graphs where each node is labeled by atomic type and each edge is labeled according to its bond type. We first truncate the dataset to small graph sizes in order to be able to train the kernel in reasonable time, and discard larger molecules. For the $M = 286$ remaining graphs of this dataset, we take into account the adjacency matrix of the graphs representing the compounds and discard the nodes and edges labels. Note that the results of our implementation are therefore not directly comparable to kernel results in the literature which take into account edge and node labels (see Ref. (KJM20) for example).

We will estimate the quality of the classification by using the F_1 score $F_1 = \frac{t_p}{t_p+(f_p+f_n)/2}$. Here, t_p, f_p and f_n are respectively the number of true positives, false positives and false negatives of the predicted distribution.

Each node of a graph will be represented by a qubit in the QPU. We first need to determine the positions of these qubits, in order to implement an interaction term in Equation (4.34) that effectively reflects the graph topology. For completeness, we detail the whole procedure in Appendix A.12.1, and give the outline here.

We first design a local optimizer detailed in Appendix A.12.1 to estimate in free space a preliminary 2D layout for each graph. Starting from a Reingold-Fruchterman layout (FR91), our optimizer minimizes the average distance between two connected nodes while maximizing the distances between unconnected nodes. Then taking advantage of our ability to tailor the spatial disposition of the tweezers generated by a Spatial Light Modulator (SLM) to fit the optimized layout, we can replicate the graph in the hardware. Following a batching method also detailed in Appendix A.12.1, we group similar graphs and superimpose them on the same SLM pattern, effectively mapping the whole dataset on only 6 different SLM patterns over a triangular grid. We therefore reduce the time needed to implement the whole dataset on the QPU.

6.3.3 Model training

To test the performance of our implementation, we perform a standard procedure called cross-validation. Cross-validation consists of dividing the dataset in 5 equal parts called 'splits', and using each split for testing while the rest of the dataset is used for training. During the training phase, we construct for each pulse duration t the corresponding kernel and train a SVM model with it (see Subsection 4.2.1). We then evaluate the F_1-score on the part of the dataset that was left as a test set. We repeat the splitting 10 times, and the cross-validation score is defined as the average of the F_1-score of each split (50 splits in total). We perform a grid search on the penalty hyperparameter C of the SVM on the range $[10^{-3}, 10^3]$ such that the final score of a given pulse is the best cross-validation score among the tested values of C.

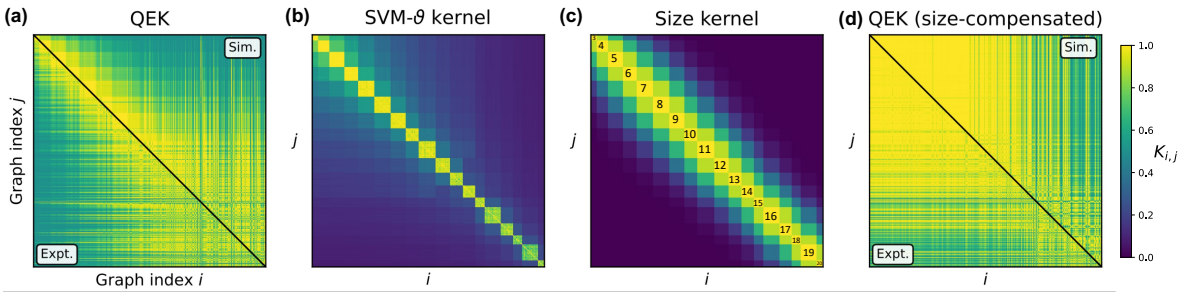


Figure 6.6: Each kernel is represented by a $M \times M$ matrix where $K_{i,j} = K(\mathcal{G}_i, \mathcal{G}_j)$ as defined in Equation (5.21). The graph indices are sorted by increasing size. A separation (black line) is drawn between numerically simulated (top right) and experimentally measured (bottom left) QEK matrices. **a.** QEK kernel obtained using directly the raw distributions $\mathcal{P}_i$ and $\mathcal{P}_j$. **b.** Kernel obtained via SVM-ϑ method. **c.** Size kernel obtained with $K^{\text{size}}(\mathcal{G}_i, \mathcal{G}_j) = \exp(-\gamma(|\mathcal{G}_i| - |\mathcal{G}_j|)^2)$ with $\gamma = 0.1$. **d.** QEK kernel obtained using modified distributions $\tilde{\mathcal{P}}_i$ and $\mathcal{P}_j$, where graphs of smaller sizes are convoluted with binomial distributions when compared to larger graphs.

Including graphs with sizes $|\mathcal{G}| \leq 20$, we numerically compute the score for a nearest-neighbor distance of $r_{NN} = 5.6\,\mu$m and a resonant constant pulse with fixed $\Omega/2\pi = 1$ MHz, and we vary its duration between $t_{min} = 0.1\,\mu$s and $t_{max} = 2.5\,\mu$s. We select the optimal duration $T = 0.66\,\mu s$ that exhibits the maximum F_1-score. We then implement this pulse on the QPU. The whole process is illustrated in Figure 6.4.

6.3.4 Classification results

After a training of our model, we experimentally obtain an F_1-score of $60.4 \pm 5.1\%$. For comparison purposes, we examine the performances of other kernels on this dataset: the Graphlet Sampling (GS), Random Walk (RW), Shortest Path (SP) and SVM-ϑ kernels, all these kernels being described in detail in Section 4.3. The F_1-scores reached by the various kernels are collected in Table 6.3. Obtained scores range from $49.8 \pm 6.0\%$ up to $58.2 \pm 5.5\%$. Those results show that the Quantum Evolution Kernel is competitive with standard classical kernels on this dataset. The SVM-ϑ kernel is found to be, among the classical kernels tested, the one with the best performance. As described in Appendix A.7.1, it is defined up to a choice of base kernel between real numbers, which gives it a certain degree of flexibility.

We show in Figure 6.6**a** the kernel matrix associated with QEK, with indices sorted by increasing size of the graphs. Using the same noise model as in the previous section, we find adequate agreement between the numerically $\mathcal{P}^{num}$ and experimentally $\mathcal{P}^{exp}$ obtained data. Quantitatively, we make use of the JS divergence to estimate this agreement for any $\mathcal{G}_i$ and observe that $\langle JS(\mathcal{P}_i^{num}, \mathcal{P}_i^{exp})\rangle_i \approx 0.03 \pm 0.01$ is one order of magnitude below

Kernel	F_1-score (%)
QEK	60.4 ± 5.1
QEK (size-compensated)	45.1 ± 3.7
SVM-ϑ	58.2 ± 5.5
Size	56.7 ± 5.6
Graphlet Sampling	56.9 ± 5.0
Random Walk	55.1 ± 6.9
Shortest Path	49.8 ± 6.0

Table 6.3: F_1-score reached experimentally on the PTC-FM dataset by QEK ($\pm$ std. on the splits). In addition, the scores reached numerically by the classical kernels SVM$-\vartheta$, Size, Graphlet Sampling, Random Walk and Shortest-Path. The values reported are the average over a 5-fold cross-validation repeated 10 times.

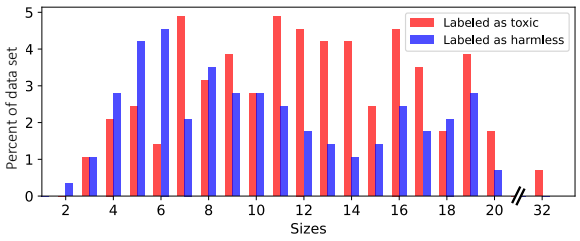


Figure 6.7: The PTC-FM dataset exhibits a strong size imbalance. For small number of nodes ($\lesssim 10$) more graphs are labeled as harmless (blue) while it is the opposite for larger graphs, more prone to be labeled as toxic (red).

$\langle JS(\mathcal{P}_i^{exp}, \mathcal{P}_j^{exp})\rangle_{i\neq j} \approx 0.33 \pm 0.01$. An interesting feature of both QEK and SVM-ϑ (Figure 6.6**b**) kernel matrices is the emergence of size-related diagonal blocks, signaling that the models identify the size of the graphs as an important feature for classification. Examining more closely the dataset, we indeed remark that the subset of PTC-FM that we used is significantly size imbalanced, as illustrated in Figure 6.7. Since the graph size seems to be a relevant feature for this particular dataset, we define a Size kernel as a Gaussian in the size difference, which reaches an F_1-score of $56.7 \pm 5.6\%$. The corresponding kernel matrix is displayed in Figure 6.6**c** and exhibits a block-diagonal shape with a Gaussian tail.

It is interesting to note that the quantum model was able to identify size as a relevant parameter for this dataset, leading to classification results which are on par with the best classical kernels.

Going forward, we modify the QEK procedure in order to make the kernel insensitive to size. To that end, we compare the measurement distributions obtained for different graph sizes using a convolution operation. Let us consider two graphs $\mathcal{G}_i$ and $\mathcal{G}_j$ of N_i and $N_j = N_i + \Delta N > N_i$ nodes respectively; and note their respective observable distributions $\mathcal{P}_i$ and $\mathcal{P}_j$. From $\mathcal{P}_i$ we

construct $\tilde{\mathcal{P}}_i = \mathcal{P}_i \star b_{\Delta N}^{(i/j)}$ the convolution of $\mathcal{P}_i$ and a binomial distribution :

$$b_{\Delta N}^{(p)}(n) = \binom{\Delta N}{n} p^n (1-p)^{\Delta N - n}. \tag{6.2}$$

$\tilde{\mathcal{P}}_i$ corresponds to the distribution one would get by adding to the graph ΔN non-interacting qubits, submitted to the same laser pulse as the other. Each of these isolated qubits undergoes Rabi oscillations, induced by the applied pulse sequence. They are therefore measured either in $|0\rangle$ with probability p or in $|1\rangle$ with probability $1-p$, where $p = \sin^2(\pi\Omega T)$ (≈ 0.768 here). We finally define the modified graph kernel as

$$K_{conv}(\mathcal{G}_i, \mathcal{G}_j) = \exp\left[-JS(\tilde{\mathcal{P}}_i, \mathcal{P}_j)\right]. \tag{6.3}$$

Using this procedure on the data obtained experimentally, we obtain the kernel matrix shown in Figure 6.6d, with a corresponding $F1$-score of $45.1 \pm 3.7\%$. If this size-compensated version of QEK had been implemented without interaction between atoms, its score would be 42%, which is the lowest score reachable by any kernel. We therefore see that this version of QEK cannot capture useful features beyond the graph size, meaning that the presence of interactions by itself is not sufficient to produce an interesting kernel for the task at hand. While the size-compensated QEK does not give results that are comparable with classical kernels, we study in the following part its expressive power, and show that the geometry induced by this method is hardly reproducible by a classical kernel.

6.3.5 Geometric test with respect to classical kernels

In order to obtain an advantage over classical approaches it is not sufficient to implement a quantum feature map based on quantum dynamics that are hard to simulate classically. As shown in (HBM+21), classical ML algorithms can in certain instances learn efficiently from intractable quantum evolutions if they are allowed to be trained on data. The authors consequently propose another metric between kernels in the form of an asymmetric metric function called the geometric difference g_{12}. It compares two kernels K_1 and K_2 in the following way:

$$g_{12} = \sqrt{||\sqrt{K_2}\,(K_1)^{-1}\,\sqrt{K_2}||_\infty} \tag{6.4}$$

where $||.||_\infty$ is the spectral norm. Intuitively, g_{12} measures the difference between how kernels K_1 and K_2 perceive the relation between data. Precisely, it characterizes the disparity regarding how each of them maps data points to their respective feature spaces. In our case, we take K_1 to be the size-compensated QEK K_{conv}, and K_2 is selected from a set of classical kernels. If the geometric difference is small, it means that there exists no underlying function mapping the data to the targets for which K_{conv} outperforms the classical kernel. On the other hand, a high geometric difference between a quantum and a classical kernel guarantees that there exists such a function for which the quantum model outperforms the classical one. Estimating

the geometric difference is therefore a sanity check to stating that the encoding of data to the feature space through the Quantum Evolution Kernel could not be closely replicated by a classical model.

We compute the geometric difference between QEK and various classical kernels over the PTC-FM dataset and report the results in Table 6.4. The threshold for a high geometric difference is typically taken to be $\sqrt{M}$, where M is the size of the dataset. Here, the obtained g_{12} is always far beyond $\sqrt{M} \sim 10^1$, indicating that the embedding of data through our quantum-enhanced kernel is not trivial and cannot be replicated by a classical machine learning algorithm.

To summarize, while the F_1-score on PTC-FM is rather similar using quantum or classical models, we see nonetheless that the geometry created by our quantum model is non-trivial. A possible interpretation of the non-superiority of quantum approaches on PTC-FM would be that the relationship between the data and the targets is not better captured by our quantum model, although its feature space is not reproducible by classical means. To further confirm this understanding, we find a function that increases and even maximizes the utility of our rich quantum feature space. We build such a function by artificially relabeling the targets according to a procedure presented in (HBM+21). If v is the eigenvector of $\sqrt{K_2}\,(K_1)^{-1}\sqrt{K_2}$ corresponding to the eigenvalue g_{12}^2, the vector of new labels is given by $y_{\text{new}} = \sqrt{K_2}v$.

When dealing with a finite amount of training data, Equation (6.4) should be regularized in order to stabilize the inversion of K_1. The regularized expression reads:

$$g_{12}(\lambda) = \sqrt{||\sqrt{K_2}\sqrt{K_1}\,(K_1 + \lambda I)^{-2}\,\sqrt{K_1}\sqrt{K_2}||_\infty} \tag{6.5}$$

where λ is the regularization parameter. The geometric difference $g_{12}(\lambda)$ has a plateau for small λ, when the regularization parameter becomes smaller than the smallest eigenvalue of K_1, and decreases for increasing λ. The effect of λ is to introduce a certain amount of training error. The training error can be upper bounded by a quantity proportional to:

$$g_{\text{tra}}(\lambda)^2 = \lambda^2||\sqrt{K_2}\,(K_1 + \lambda I)^{-2}\,\sqrt{K_2}||_\infty. \tag{6.6}$$

Practically, one should look at the regime where g_{12} has not plateaued but the training error is still small enough.

A regularization should be introduced also in the relabeling procedure. The new labels are taken to be $y_{\text{new}} = \sqrt{K_Q}v$, where v is the eigenvector of the regularized matrix

$$\sqrt{K_Q}\sqrt{K_C}\,(K_C + \lambda I)^{-2}\,\sqrt{K_C}\sqrt{K_Q}$$

corresponding to the eigenvalue $g_{12}(\lambda)^2$.

We observe that QEK, without retraining, retains an F_1-score of around 99% on the relabeled dataset, while the closest classical kernel reaches a score of at most 82% even after retraining it on the new labels. The results are summarized in Table 6.5, where the difference in F_1-score between QEK and the various classical kernels is shown.

Geometric Difference w.r.t. QEK	
SVM-ϑ	10^3
Size	10^5
Graphlet Sampling	10^4
Random Walk	10^5
Shortest Path	10^5

Table 6.4: Order of magnitude of the geometric difference between QEK and various classical kernels.

F_1-score gap (%) w.r.t. QEK (relabeled)	
SVM-ϑ	17.2 ± 4.5
Size	17.8 ± 4.2
Graphlet Sampling	20.1 ± 4.5
Random Walk	17.3 ± 4.3
Shortest Path	18.2 ± 4.4

Table 6.5: Gap in F_1-score between QEK and various classical kernels after relabeling the dataset.

In light of the geometric difference assessment and the observed gap of F_1-score between QEK and classical kernels on an artificial function, it remains an open question to generally characterize which types of dataset naturally offer a structure that better exploits the geometry offered by our quantum model, without requiring artificial tweaking of the labels. In the following section, we present a synthetic dataset on which QEK is able to outperform classical methods without any relabeling.

6.3.6 Synthetic dataset

This binary classification dataset is created by sampling weighted random walks on a triangular lattice. In class A, sites belonging to a honeycomb-type sublattice are favored. They are explored with a weight $p_0 = 1$ while the rest of the triangular lattice sites are explored with a weight $p < 1$. Class B is constructed in a similar fashion, but taking a kagome instead of a honeycomb sublattice. The construction of this artificial dataset is illustrated in Fig. 6.8. In the case where $p = 0$, the differences in their local structure make the two classes easily distinguishable. However, with increasing p, their local structure becomes more and more similar, as additional triangular lattice sites are incorporated. When p is large enough, a lot of triangular local substructures are shared by the two classes, rendering them potentially hard to distinguish by classical methods. At $p = 1$, the underlying triangular lattice is explored uniformly, rendering the datasets indistinguishable.

Building on our ability to distinguish between graphs with similar local structure but globally distinct, we apply QEK on this synthetic dataset. We expect our method to be hardly affected

by the presence of sparse defects and therefore be able to outperform classical approaches.

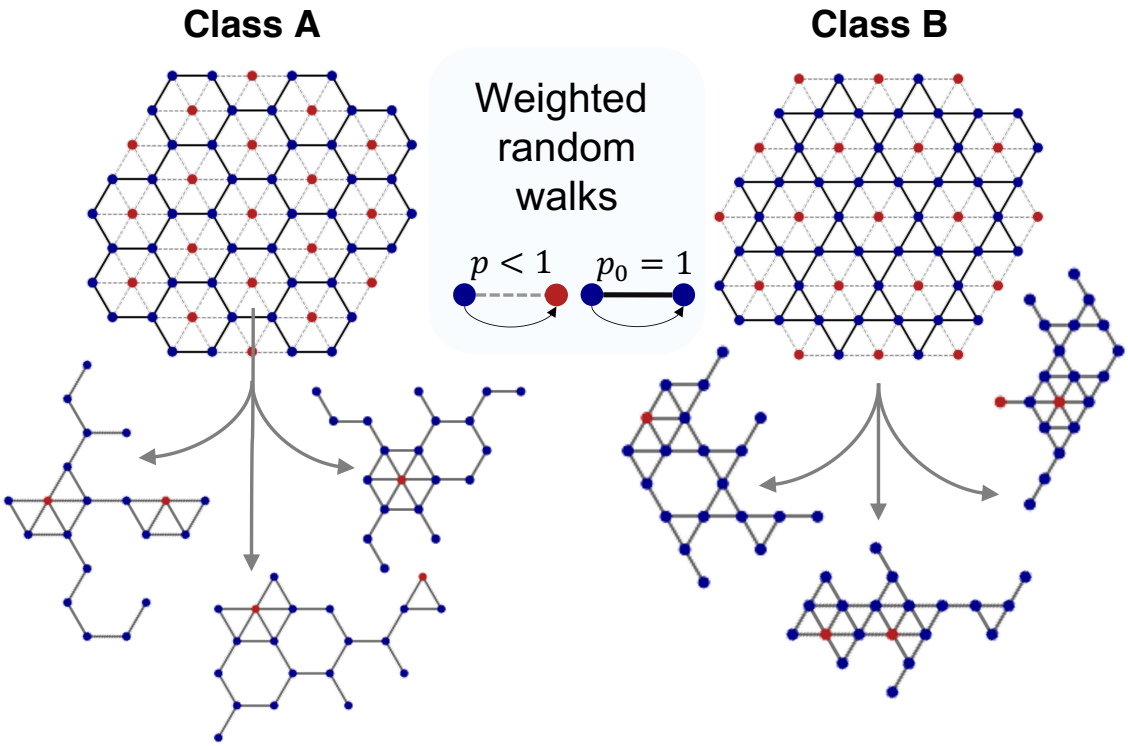


Figure 6.8: Graphs in Class A contains honeycomb sites (blue) with inclusions of non-honeycomb sites (red) with probability p. Graphs in Class B contains kagome sites (blue) with inclusions of non-kagome sites (red) with probability p. We show examples of generated graph with the aforementioned process.

We investigate numerically this assumption, for several values of p. In each case, we create 200 graphs of 20 nodes each, 100 in each class. The graphs are mapped to a triangular lattice with 5 µm spacing. Here, we consider two alternative schemes of pulse sequences. The first one remains almost the same as the experimentally implemented one, i.e. a unique resonant pulse of $\Omega/2\pi = 2$ MHz with parameterized duration up to $8\,\mu$s. The second one is an alternate layer scheme with 4 parameters as described in (HTDH21a), where we evaluate 500 random values of the parameters and select the best one. The procedure is designed such that it would be directly implementable on the hardware, as we did for the PTC-FM dataset. We then compare the F_1-score reached by QEK to those reached by other classical kernels, namely: SVM-ϑ, GS, RW and SP. The results are summarized in Fig. 6.9. With decreasing proportion of defects, all methods perform increasingly better, as expected. Overall, regarding the mean F_1-score reached, the two QEK schemes outperform the four other classical kernels tested for all $p \leq 0.5$. Noticeably, at $p = 0.1$ (*resp* $p = 0.2$), the mean gap in F_1-score between the QEK scheme and the the best classical scheme is 4.5% (*resp* 7.1%) while the mean gap obtained with the alternate QEK scheme is even larger with 13.7% (*resp* 21%), thus showing that QEK can significantly

surpass classical approaches on certain types of datasets. When adding too many defects, i.e. $p = 0.5$, our Quantum Evolution Kernel exhibits similar performance to the SVM-ϑ.

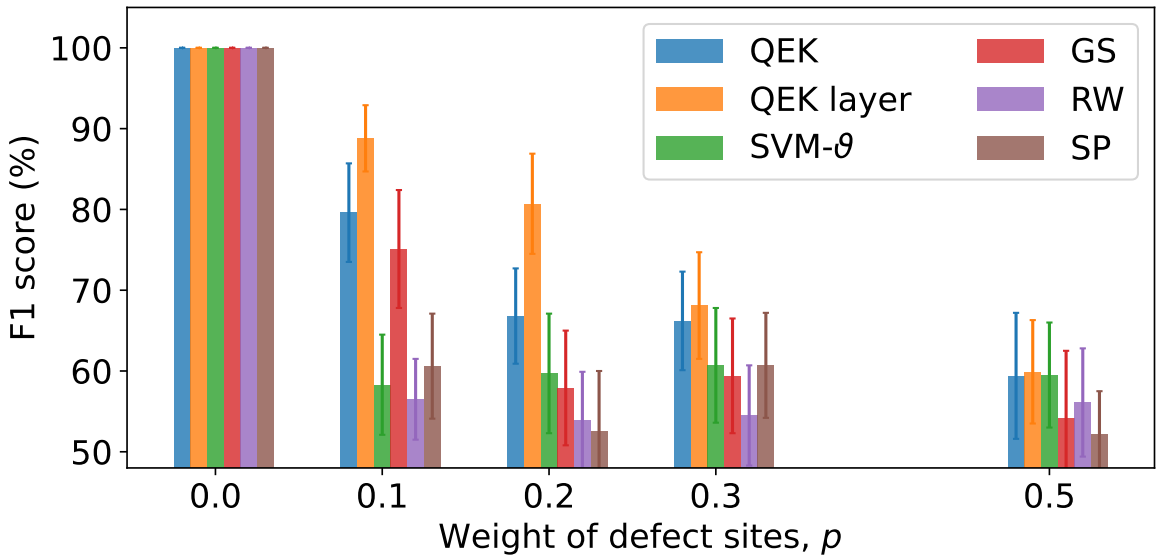


Figure 6.9: F1-score (%) reached on the synthetic dataset for different probabilities p of including non-sublattice sites, by the Quantum Evolution Kernel (the alternate scheme is noted QEK layer) as well as by the best SVM-ϑ, GS, RW and SP kernels. The values reported are the average over a 5-fold cross-validation repeated 10 times. Each kernel reaches an F_1-score of 100% when $p = 0$.

Conclusion and outlook

Machine learning enabled the resolution of many real world problems that other traditional computational methods struggled to solve, or could solve in a more expensive way. Quantum computing is a new paradigm of computation using the quantum states of matter that enables a large computational speed up on some problems. The recent development of quantum processing units encouraged the research of concrete applications of quantum computers. It then became natural to look for ways in which quantum computers can be applied for machine learning.

The first part of this thesis aimed at understanding the capabilities of variational quantum circuits (VQC) for machine learning tasks on unstructured vector data, and have a clearer idea on the necessary conditions in order to expect a quantum advantage. Much of the intuition of the community about the advantage of VQCs over classical learning methods was the fact that VQCs were high dimensional functions. VQCs could in principle express functions that could not be expressed otherwise. However, it is not enough to guarantee a quantum advantage. For high dimension linear models, one can indeed construct an approximate representation of quantum models using the technique of random features regression. I showed that a necessary condition for a quantum model to avoid dequantization by its classical surrogate is to have a large weight vector norm. This is due to the fact that learning a classical model on the same feature map will lead to a solution called the Minimum Norm Least Square (MNLS) estimator, but the training dynamics of the quantum circuits will not necessarily lead to the same solution. I give concrete examples of this separation.

The previous part studied the properties of VQCs with little hypothesis on the structure of the data. The second part explores methods to perform machine learning tasks on graph structured data. The main idea is to encode the graph into a hamiltonian that has the same topology. One then prepares a quantum state by driving this hamiltonian, and the measurements are incorporated in a classical machine learning algorithm. This approach is especially suited to neutral atoms quantum computers. With such platforms, one can indeed easily create a quantum system with the desired connectivity, and the geometry can be changed at each run. I developed a large family of algorithms, inspired by kernels, graphs neural networks, and transformers with the intention to be ran on current hardware. I performed numerical experiments on large scale datasets, and described the results of an experimental implementation on the hardware of Pasqal.

Some topics were out of the scope of this thesis, but are relevant to the theme of quantum machine learning:

- In this thesis I did not provide many theoretical results about the advantage in terms of complexity (only a little in Chapter 2 and Chapter 3), in contrary to other works in quantum machine learning (GD23; JFPN+23). Concerning the part on learning on graphs, a promising research direction would be to identify real world learning tasks related to combinatorial optimization (e.g classify graphs related to their MIS or max-CUT) that can be solved better on a quantum computer.
- The noise of the hardware can severely hinder the execution of any quantum algorithms. Quantum error correction (QEC) is the design of algorithm to detect and correct error as the circuit is executed. The idea is to encode a logical qubit into several noisy qubits which allow some redundancy and robustness to noise. The idea is quite old (Got02; Kit97) but has been experimentally implemented only very recently at scale (BEG+24; goo23). Several works provided resource estimate to execute usual algorithms like Shor with QEC (HLH22; GRLR+23). In contrary, there is little work on estimating the cost of error correction in the context of a machine learning task.
- There are some difficulties in using neutral atoms quantum computers for quantum machine learning. The main issue in the short term is the low repetition rate. At the optimistic rate of 10 Hz, if one wants to acquire 1000 measurements per data input on a dataset of 10^6 datapoints, one would need around 3 years for acquiring all the data, assuming a non stop running QPU. In addition to hardware improvements, a way to overcome this problem would be to look for ways to use as less measurement shots as possible. Instead of measuring the correlation matrix with good fidelity as in Subsection 5.5.2 and Subsection 5.5.3, one could investigate if some error is tolerated in the process.
- This thesis was mostly about supervised learning. There are other learning paradigms like unsupervised learning, reinforcement learning, and generative learning. Generative learning has several real life applications like natural language processing or drug discovery. One wants to generate new data with the same distribution as the training data. Generative tasks can be more interesting than prediction tasks for gaining a quantum advantage, because they use samples instead of expectation values of observables. There are indeed classes of circuits like the IQP circuits for which sampling is difficult (BMS16) but computing expectation values are easy on a classical computer (RAAB25; dN10).

I conclude by giving my perspective about the evolution of the research on quantum machine learning for classical data. An aspect of understanding quantum algorithms for machine learning that will gain increasing importance in my opinion is analyzing the signal extracted from the data. In classical machine learning, this concept has been extensively explored. As highlighted in (BBCV21), many successful algorithms owe their effectiveness to leveraging the geometric

structure of input data—such as convolutional neural networks, which excel in image processing due to their translation invariance. To achieve a quantum advantage with classical data, it is essential to identify relevant inductive biases. Almost all existing proofs for quantum advantage in learning tasks (GD23; JGM+23; LAT21) come from cryptography problems where we know quantum computation provide an advantage. It seems also likely that quantum advantage will manifest in quantum data, or data coming from quantum systems.

Until now, most of the work on variational quantum circuits, and near term quantum algorithms (including this thesis) were guided primarily by the capabilities of the hardware, more than a theoretical analysis. Most of the progress of classical machine learning was also done by experimenting, and theoretical explanations of the capabilities of the algorithms often came *a posteriori*. Since quantum hardware is expensive to run, it is likely that progress of quantum machine learning in practical use cases will come from theoretical analysis *a priori*.

APPENDIX A

Appendix

A.1 Approximation results for RFF in the context of VQCs

A.1.1 Distinct sampling in the Pauli encoding case

In the case of Pauli encoding only, we know that $\Omega = [\![-L, L]\!]^d$ (considered here to be the full spectrum, not Ω_+ defined in Section 2.1, which would have been equivalent). In one dimension, we simply have $\sigma_p = 1/L \sum_{\ell=-L,\cdots,L} \ell^2 = O(L^2)$. In dimension d, a frequency ω is given by its values on each dimension $(j_1, \cdots, j_d)$ with $j_k \in [|-L, L|]$. We similarly have

$$\sigma_p = \frac{1}{(2L+1)^d} \sum_{j_1,\cdots,j_d} j_1^2 + \cdots + j_d^2 \tag{A.1}$$

Note that $\sum_{j_1,\cdots,j_d} j_1^2 + \cdots + j_d^2$ is $d(2L+1)^{d-1}$ times the sum of all squares,

$$\begin{aligned} \sigma_p = \frac{d(2L+1)^{d-1}}{(2L+1)^d} \sum_{\ell=-L}^{L} \ell^2 &= \frac{d}{2L+1} \frac{2L(L+1)(2L+1)}{6} \\ &= O(dL^2) = O(d|\Omega|^{2/d}) \end{aligned} \tag{A.2}$$

The expression is then obtained by replacing the value of σ_p in theorem 2.2.

We note that we can generalize this results to scaled Pauli encoding, as done in (KPE22; STJ22), by replacing L by a term growing as c^L where c is a constant. D would grow linearly in L and not logarithmically anymore.

A.1.2 Grid sampling with a general hamiltonian

We provide here a bound on the minimum of samples required to achieve a certain error between the RFF model and the complete model in the case of a general encoding in the gird sampling

strategy. The proof and details for this theorem is shown in Appendix Section A.1.2.

Theorem A.1. *Let $\mathcal{X}$ be a compact set of $\mathbb{R}^d$, and $\epsilon > 0$. We consider a training set $\{(x_i, y_i)\}_{i=1}^M$. Let f be a VQC model with any hamiltonian encoding, with a maximum individual frequency $\omega_{\max}$ and full freedom on the associated frequency coefficients, trained with a regularization λ. Let $\sigma_y^2 = \frac{1}{M}\sum_{i=1}^M y_i^2$ and $|\mathcal{X}|$ the diameter of $\mathcal{X}$. Let $\tilde{f}$ be the RFF model with D samples in the grid strategy trained on the same dataset and the same regularization. Let $C = |f|_\infty|\mathcal{X}|$ and s the sampling rate defined in the grid sampling strategy. Then we can guarantee $|f(x) - \tilde{f}(x)| \leq \epsilon$ for $0 < s < \frac{1}{C}$ with probability $1 - \delta$ for a number D of samples given by:*

$$D = \Omega\left(\frac{dC_1}{\lambda^4(\epsilon - sC)^2}\left[log(\omega_{\max}|\mathcal{X}|/s) + log\frac{C_2}{\lambda^2(\epsilon - sC)} - log\delta\right]\right) \tag{A.3}$$

with C_1 and C_2 being constants depending on σ_y, $d(X)$. We recall that in Eq.A.3 the notation Ω stands for the computational complexity "Big-Ω" notation.

Proof. The following theorem bounds the approximation between a function defined by its Fourier series and another function with frequencies distant by at most a constant s of the original frequencies.

Let $\mathcal{X}$ a compact set of $\mathbb{R}^d$ with diameter $|\mathcal{X}|$ and Ω a finite subset of $\mathcal{X}$. Let $f(x) = \sum_{\omega\in\Omega} a_\omega cos(\omega^T x) + b_\omega sin(\omega^T x)$. Let Ω' a subset of $\mathcal{X}$ and $s > 0$ such that $\forall\omega \in \Omega$, $\exists\omega' \in \Omega$, $|\omega - \omega'| \leq s$.

Let $\mathcal{F}_{\Omega'} = \left\{ \sum_{\omega\in\Omega'} a_\omega cos(\omega^T x) + b_\omega sin(\omega^T x), a_\omega, b\omega \in \mathbb{R} \right\}$.

Theorem A.2. *It exists $f' \in \mathcal{F}_{\Omega'}$ such that*

$$\sup_{x\in\mathcal{X}} |f'(x) - f(x)| \leq sC \tag{A.4}$$

with $C = |\mathcal{X}||f|_\infty$.

Proof. For each $\omega \in \Omega$ let $b(\omega) \in \Omega'$ be such that $|\omega - b(\omega)| \leq s$. Such element exists by definition but is not necessarily unique. Let $f'(x) = \sum_{\omega\in\Omega} a_\omega cos(b(\omega)^T x) + b_\omega sin(b(\omega)^T x)$. The $b(\omega)$s are not necessarily different therefore there might be less frequencies in f' than in f.

$$|f(x) - f'(x)| = 2\Big|\sum_{\omega\in\Omega} sin(\frac{(\omega - b(\omega))^T}{2}x) \tag{A.5}$$

$$[b_\omega sin(\frac{(\omega + b(\omega))^T}{2}x) - a_\omega cos(\frac{(\omega + b(\omega))^T}{2}x)]\Big| \tag{A.6}$$

$$\leq 2\sum_{\omega\in\Omega} |\frac{(\omega - b(\omega))^T}{2}||x||[|b_\omega| + |a_\omega|] \tag{A.7}$$

$$\leq s|x|\sum_{\omega\in\Omega} |b_\omega| + |a_\omega| \tag{A.8}$$

$$\leq s|\mathcal{X}||f|_\infty \tag{A.9}$$

□

We shall here extend the proof where we sample from the grid described above. Let us note $\hat{f}_s$ the RFF model with the whole grid and $\tilde{f}$ the RFF model with D samples from the grid below. For all $x \in \mathcal{X}$ we have

$$|\tilde{f}(x) - f(x)| \leq |\tilde{f}(x) - \hat{f}_s| + |\hat{f}_s - f(x)| \tag{A.10}$$
$$\leq |\tilde{f}(x) - \hat{f}_s| + sC \tag{A.11}$$

Then

$$\mathbb{P}(|\tilde{f}(x) - f(x)| \geq \epsilon) \leq \mathbb{P}(|\tilde{f}(x) - \hat{f}_s| \geq \epsilon - sC) \tag{A.12}$$

for $s < \epsilon/C$.

In this case $|\Omega| = (\omega_{\max}/s)^d$ Using the expression of Section A.1.1, we can guarantee that $|f(x) - \tilde{f}(x)| \leq \epsilon$ with probability $1 - \delta$ if

$$D = \Omega\left(d\frac{1}{(\epsilon - sC)^2}\left[log(\omega_{\max}|\mathcal{X}|/s) + log\frac{1}{\epsilon - sC} - log\delta\right]\right) \tag{A.13}$$

□

A.2 Minimum norm least square estimator

We recall Theorem 1:

Theorem 1 (From (HMRT22)). *Let $\beta_0 = 0$ the initialization of a gradient descent algorithm. Let the following iterations be defined by*

$$\beta_{k+1} = \beta_k + \gamma\, \Phi^\top(\boldsymbol{y} - \Phi\beta_k)\,, \tag{3.22}$$

with γ the learning rate such that $0 \leq \gamma \leq 1/\lambda_{\max}(\Phi^\top\Phi)$ where $\lambda_{\max}(\Phi^\top\Phi)$ is the largest eigenvalue of $\Phi^\top\Phi$. Then

- *β_k converges towards the minimum norm least square estimator β_{MNLS} defined in Equation (3.21).*
- *$\beta_{MNLS} = \Phi^\top(\Phi\Phi^\top)^{-1}\boldsymbol{y}$.*

Proof. The algorithm converges to a minimizer of $||y - X\beta||^2$ noted $\hat{\beta}$ for the choice of the step size. Furthermore, at each iteration iteration of the algorithm, β_k lies in the row space of X. Then $\hat{\beta}$ also lies in the row space of X, and we will show that it is necessary the minimum norm least square.

If $(X^\top X)$ is low rank, then the minimum least square is unique as shown previously.

Otherwise, if $(XX^\top)$ is full rank, let us denote $\beta^* = X^\top\alpha^\star = X^\top(XX^\top)^{-1}y$. β^* is a minimizer of the least square loss and is in the row space of X. We also have that $X\beta = y$. We will show that there are no other minimizer of the least square loss on the row space of X and that β^* is the minimum norm least square. Let $\beta = X^\top\alpha$ another minimizer of the least square loss in the row space of X. We then necessarily have $X\beta = y = X\beta^*$. Then $X(\beta - \beta^*) = XX^\top(\alpha - \alpha^*) = 0$, and $\alpha = \alpha^*$ since $XX^\top$ is full rank.

Let $\beta = X^\top\alpha + v$ be a minimizer of the least square loss where v is in the orthogonal of the row space of X. We then have

$$\begin{aligned} X\beta &= X\beta^* \\ XX^\top\alpha + Xv &= XX^\top\alpha^* \\ XX^\top(\alpha^* - \alpha) &= Xv \end{aligned}$$

Let us compute $||X^\top\alpha||^2 - ||X^\top\alpha^*||^2$.

$$\begin{aligned} ||X^\top\alpha||^2 - ||X^\top\alpha^*||^2 &= (X^\top\alpha)^\top(X^\top\alpha - X^\top\alpha^*) + (X^\top\alpha - X^\top\alpha^*)^\top(X^\top\alpha^*) && \text{(A.14)} \\ &= (X^\top\alpha + X^\top\alpha^*)^\top(X^\top\alpha - X^\top\alpha^*) && \text{(A.15)} \\ &= (\alpha + \alpha^*)^\top XX^\top(\alpha - \alpha^*) && \text{(A.16)} \\ &= -(\alpha + \alpha^*)^\top Xv && \text{(A.17)} \\ &= -(X^\top\alpha + X\top\alpha^*)^\top v && \text{(A.18)} \\ &= 0 && \text{(A.19)} \end{aligned}$$

where at the last equality we use the fact that v is orthogonal to the row space of X. Finally we have $||\beta||^2 - ||\beta^*||^2 = ||X^\top\alpha||^2 + ||v||^2 - ||X^\top\alpha^*||^2 = ||v||^2 > 0$ if v is non zero. Then the minimum norm least square is the unique least square estimator to be in the row space of X. □

A.3 Random Feature regression

We recall Theorem 2:

Theorem 2. *Let $\phi(x) = [\sqrt{q_1}\,\phi_1(x) \dots \sqrt{q_p}\,\phi_p(x)]^\top$ where $\phi_i(x)$ are basis functions such that $\forall x$, $|\phi_i(x)| \leq 1$ and $q = (q_1, \dots q_p)$ represents a discrete probability distribution, and let $f(x) = \beta^\top\phi(x)$. Let S be a subset of $[\![1, p]\!]$ sampled independently with the probability density q, with $D = |S|$. Then there exists coefficients $c_1, \dots c_D$ such that $\hat{f}(x) = \sum_{k\in S} c_k\phi_k(x)$ satisfies*

$$\|\hat{f} - f\|_\mu \leq \frac{\max_i |\beta_i / \sqrt{q_i}|}{\sqrt{D}}(1 + \sqrt{2\log\frac{1}{\delta}}) \tag{3.27}$$

As a consequence, if one applies the above to β_{MNLS} obtained from a kernel matrix K and target vector $\boldsymbol{y}$ such that $\|\boldsymbol{y}\|_\infty \leq 1$ there exists coefficients $c_1, \dots c_D$ such that

$$\|\hat{f} - f_{MNLS}\|_\mu \leq \frac{M}{\sqrt{D}\,\lambda_{\min}(K)}(1 + \sqrt{2\log\frac{1}{\delta}})\,. \tag{3.28}$$

Proof. The Equation (3.27) is a direct consequence of theorem 3.1 from (RR08a) (reminded in Appendix A.3).

To derive Equation (3.28), we note that $\beta_i = \sqrt{q_i}\,\boldsymbol{k}^\top K^{-1}\boldsymbol{y}$ where $\boldsymbol{k} = [\phi_i(x_1)\dots\phi_i(x_M)]^\top$. We have then

$$|\beta_i/\sqrt{q_i}| \leq |\boldsymbol{k}^\top K^{-1}\boldsymbol{y}| \tag{A.20}$$
$$\leq ||\boldsymbol{k}||||K^{-1}\boldsymbol{y}|| \tag{A.21}$$
$$\leq \frac{M}{\lambda_{\min}(K)} \tag{A.22}$$

The second equation is obtained using Cauchy Schwarz inequality, and the last one by using the facts that $||\boldsymbol{k}|| \leq \sqrt{M}$ because $||\boldsymbol{k}||_\infty \leq 1$, and $||\boldsymbol{y}|| \leq \sqrt{M}$ because $||\boldsymbol{y}||_\infty \leq 1$ □

A.4 Concentration of eigenvalues of the kernel matrix for the Fourier feature map with integer coefficients

In this section, we establish concentration bounds on the eigenvalues of the kernel matrix. We consider the Fourier feature map:

$$\phi(x) = \frac{1}{\sqrt{p}}\begin{bmatrix}\cos\left(\omega^\top x\right)\\ \sin\left(\omega^\top x\right)\end{bmatrix} \tag{A.23}$$

with $\Omega \subset \mathbb{Z}^d$,.

The kernel is defined as:

$$\begin{aligned} k(x,x') &= \phi(x)^\top\phi(x') \\ &= \frac{1}{p}\sum_{\omega\in\Omega}\cos\left(\omega^\top x\right)\cos\left(\omega^\top x'\right) + \sin\left(\omega^\top x\right)\sin\left(\omega^\top x'\right) \end{aligned} \tag{A.24}$$

Let $(x_1, \dots x_M)$ uniformly distributed on $[0, 2\pi]^d$, and let K be the empirical kernel matrix. We want to lower bound the smallest eigenvalue of K, and we use the following lemma

Lemma A.1. *Theorem 2.1 from (WS80). Let A be a $M \times M$ complex matrix with real eigenvalues. Let $m = tr(A)/M$ and $s^2 = tr(A^2)/M - m^2$. We have that*

$$m - s\sqrt{M-1} \leq \lambda_{min}(A) \leq m - \frac{s}{\sqrt{M-1}} \tag{A.25}$$

For the kernel matrix, $m = 1$ and $s^2 = 1 + \frac{1}{M}\sum_{j\neq i}^{M} k(x_i, x_i)^2 - 1 = \frac{1}{M}\sum_{j\neq i}^{M} k(x_i, x_j)^2$. We prove the following result:

Lemma A.2. *The expectation of s^2 is given by*

- $\mathbb{E}[s^2] = \dfrac{M(M-1)}{2Mp} = \dfrac{(M-1)}{2p}$

Furthermore, there exists a constant C such that

- $\mathbb{V}[s^2] \leq \dfrac{C}{p}$

Proof. With the use of the lemma A.3, we have that

$$\mathbb{E}[s^2] = \frac{M(M-1)}{M}\mathbb{E}(k(x, x')^2) = \frac{M-1}{2p} \tag{A.26}$$

$$\mathbb{E}[s^4] = \mathbb{E}\left[\frac{1}{M^2}\left(\sum_{j\neq i}^{M} k(x_i, x_j)^2\right)^2\right] = \frac{1}{M^2}\mathbb{E}\left[2\sum_{j\neq i}^{M} k(x_i, x_j)^4 + \sum_{j\neq i\neq k\neq l=1}^{M} k(x_i, x_j)^2 k(x_k, x_l)^2\right] \tag{A.27}$$

$$\leq \frac{C'}{p}\frac{M(M-1)}{M^2} + \frac{M(M-1)(M-2)(M-3)}{M^2}\frac{(M-1)^2}{4p^2} \tag{A.28}$$

$$\mathbb{V}[s^2] \leq \frac{C'}{p}\frac{M(M-1)}{M^2} + \frac{M(M-1)(M-2)(M-3)}{M^2}\frac{1}{4p^2} - \frac{(M-1)^2}{4p^2} \tag{A.29}$$

$$\leq \frac{C'}{p}\frac{M(M-1)}{M^2} + \frac{1}{4p^2}((M-2)(M-3) - (M-1)^2) \tag{A.30}$$

$$\leq \frac{C}{p} \tag{A.31}$$

□

Now we prove the following inequality

$$\mathbb{P}(\lambda_{\min}(K) > \frac{1}{2}) \geq 1 - \frac{C}{p}\frac{4(M-1)^2p^2}{(p-8(M-1)^2)^2} = 1 - \frac{C}{p}\frac{4(M-1)^2}{1-8\frac{(M-1)^2}{p^2}} \tag{A.32}$$

Proof. With Chebyshev's inequality

$$\mathbb{P}(|s^2 - \frac{2(M-1)}{p}| > \epsilon) \leq \frac{\mathbb{V}[s^2]}{\epsilon^2} \tag{A.33}$$

$$\mathbb{P}(s^2 \geq \frac{1}{4(M-1)}) \leq \mathbb{P}(s^2 - \frac{2(M-1)}{p} \geq \frac{1}{4(M-1)} - \frac{2(M-1)}{p}) \tag{A.34}$$

$$\leq \mathbb{P}(s^2 - \frac{(p-8(M-1)^2)}{4(M-1)p}) \tag{A.35}$$

$$\leq \frac{C}{p}\frac{4(M-1)^2p^2}{(p-8(M-1)^2)^2} \tag{A.36}$$

With lemma A.1,

$$\mathbb{P}(\lambda_{\min}(K) > \frac{1}{2}) \geq \mathbb{P}(s \leq \frac{1}{2\sqrt{M-1}}) = \mathbb{P}(s^2 \leq \frac{1}{4(M-1)}) \tag{A.37}$$

$$\geq 1 - \mathbb{P}(s^2 \geq \frac{1}{4(M-1)}) \tag{A.38}$$

$$\geq 1 - \frac{C}{p}\frac{4(M-1)^2p^2}{(p-8(M-1)^2)^2} \tag{A.39}$$

□

We just need to prove the following lemma

Lemma A.3. *Let x_i, x_j independent and uniformly distributed on $[0, 2\pi]^d$. There exists a constant $C' > 0$ such that*

- $\mathbb{E}[k(x_i, x_j)^2] = \frac{1}{2p}$
- $\mathbb{E}[k(x_i, x_j)^4] \leq \frac{C'}{p}$

$$\mathbb{E}[k(x_i, x_j)^2] = \frac{1}{p^2}\mathbb{E}[\sum_{\omega}(\cos\Big(\omega^\top x_i\Big)\cos\Big(\omega^\top x_j\Big) + \sin\Big(\omega^\top x_i\Big)\sin\Big(\omega^\top x_j\Big))^2] \tag{A.40}$$

$$+ \frac{1}{p^2}\mathbb{E}[\sum_{\omega\neq\omega'}(\cos\Big(\omega^\top x_i\Big)\cos\Big(\omega^\top x_j\Big) + \sin\Big(\omega^\top x_i\Big)\sin\Big(\omega^\top x_j\Big))\times \tag{A.41}$$

$$(\cos\Big(\omega'^\top x_i\Big)\cos\Big(\omega'^\top x_j\Big) + \sin\Big(\omega'^\top x_i\Big)\sin\Big(\omega'^\top x_j\Big))] \tag{A.42}$$

We have that $\mathbb{E}[\cos^2(\omega^\top x_i)\cos^2(\omega^\top x_j)] = \frac{1}{4}$ and $\mathbb{E}[\sin^2(\omega^\top x_i)\sin^2(\omega^\top x_j)] = \frac{1}{4}$.
Furthermore $\mathbb{E}[\cos\Big(\omega^\top x_i\Big)\cos\Big(\omega^\top x_j\Big)\sin\Big(\omega^\top x_i\Big)\sin\Big(\omega^\top x_j\Big)] = 0$.
Therefore,

$$\mathbb{E}[k(x_i, x_j)^2] = \frac{1}{2p} \tag{A.43}$$

We introduce some simplifying notations that will be used for the rest of the calculation.

$$\sum_{\omega\in\Omega} c_ic_j + s_is_j = \sum_{\omega\in\Omega}\cos\Big(\omega^\top x_i\Big)\cos\Big(\omega^\top x_j\Big) + \sin\Big(\omega^\top x_i\Big)\sin\Big(\omega^\top x_j\Big) \tag{A.44}$$

$$\sum_{\omega_1,\omega_2\in\Omega}(c_i^{(1)}c_j^{(1)} + s_i^{(1)}s_j^{(1)})(c_i^{(2)}c_j^{(2)} + s_i^{(2)}s_j^{(2)}) \tag{A.45}$$

$$= \sum_{\omega_1,\omega_2\in\Omega}\Big(\cos\Big(\omega_1^\top x_i\Big)\cos\Big(\omega_1^\top x_j\Big) + \sin\Big(\omega_1^\top x_i\Big)\sin\Big(\omega_1^\top x_j\Big)\Big)\Big(\cos\Big(\omega_2^\top x_i\Big)\cos\Big(\omega_2^\top x_j\Big) + \sin\Big(\omega_2^\top x_i\Big)\sin\Big(\omega_2^\top x_j\Big)\Big) \tag{A.46}$$

By applying the multinomial formulas, we have that

$$k(x_i, x_j)^4 = \frac{1}{p^4}\Big(\sum_{\omega\in\Omega} c_i c_j + s_i s_j\Big)^4 \tag{A.47}$$

$$= \frac{1}{p^4}\sum_{\omega\in\Omega}(c_i c_j + s_i s_j)^4 \tag{A.48}$$

$$+ \frac{4}{p^4}\sum_{\omega_1,\omega_2\in\Omega,\omega_1\neq\omega_2}(c_i^{(1)}c_j^{(1)} + s_i^{(1)}s_j^{(1)})^3(c_i^{(2)}c_j^{(2)} + s_i^{(2)}s_j^{(2)}) \tag{A.49}$$

$$+ \frac{6}{p^4}\sum_{\omega_1,\omega_2\in\Omega,\omega_1\neq\omega_2}(c_i^{(1)}c_j^{(1)} + s_i^{(1)}s_j^{(1)})^2(c_i^{(2)}c_j^{(2)} + s_i^{(2)}s_j^{(2)})^2 \tag{A.50}$$

$$+ \frac{4}{p^4}\sum_{\omega_1,\omega_2,\omega_3\in\Omega,\omega_1\neq\omega_2\neq\omega_3}(c_i^{(1)}c_j^{(1)} + s_i^{(1)}s_j^{(1)})^2(c_i^{(2)}c_j^{(2)} + s_i^{(2)}s_j^{(2)})(c_i^{(3)}c_j^{(3)} + s_i^{(3)}s_j^{(3)}) \tag{A.51}$$

$$+ \frac{24}{p^4}\sum_{\omega_1,\omega_2,\omega_3,\omega_4\in\Omega,\omega_1\neq\omega_2\neq\omega_3\neq\omega_4}(c_i^{(1)}c_j^{(1)} + s_i^{(1)}s_j^{(1)})(c_i^{(2)}c_j^{(2)} + s_i^{(2)}s_j^{(2)})(c_i^{(3)}c_j^{(3)} + s_i^{(3)}s_j^{(3)})(c_i^{(4)}c_j^{(4)} + s_i^{(4)}s_j^{(4)}) \tag{A.52}$$

Our goal is to show that there exists a constant C such that $\mathbb{E}[k(x_i, x_j)^4] \leq \frac{C}{p}$. All the sums except the last one have at most p^3 terms, and we can bound each one of them using the fact that $|c_i c_j + s_i s_j| \leq 2$. We need to look at the last sum.

$$\mathbb{E}[(c_i^{(1)}c_j^{(1)} + s_i^{(1)}s_j^{(1)})(c_i^{(2)}c_j^{(2)} + s_i^{(2)}s_j^{(2)})(c_i^{(3)}c_j^{(3)} + s_i^{(3)}s_j^{(3)})(c_i^{(4)}c_j^{(4)} + s_i^{(4)}s_j^{(4)})] \tag{A.53}$$

$$= \sum_{f_1,f_2,f_3,f_4\in\{\cos,\sin\}}\mathbb{E}[f_1(\omega_1^\top x_i)f_2(\omega_2^\top x_i)f_3(\omega_3^\top x_i)f_4(\omega_4^\top x_i)f_1(\omega_1^\top x_j)f_2(\omega_2^\top x_j)f_3(\omega_3^\top x_j)f_4(\omega_4^\top x_j)] \tag{A.54}$$

$$= \sum_{f_1,f_2,f_3,f_4\in\{\cos,\sin\}}\mathbb{E}[f_1(\omega_1^\top x_i)f_2(\omega_2^\top x_i)f_3(\omega_3^\top x_i)f_4(\omega_4^\top x_i)]^2 \tag{A.55}$$

$$\begin{aligned}\cos\Big(\omega_1^\top x_i\Big)\cos\Big(\omega_2^\top x_i\Big)\cos\Big(\omega_3^\top x_i\Big)\cos\Big(\omega_4^\top x_i\Big) = \frac{1}{8}[&\cos\Big((\omega_1+\omega_2+\omega_3+\omega_4)^\top x_i\Big)\\ &+\cos\Big((\omega_1+\omega_2-\omega_3-\omega_4)^\top x_i\Big)\\ &+\cos\Big((\omega_1+\omega_2+\omega_3-\omega_4)^\top x_i\Big)\\ &+\cos\Big((\omega_1+\omega_2-\omega_3+\omega_4)^\top x_i\Big)\\ &+\cos\Big((\omega_1-\omega_2+\omega_3+\omega_4)^\top x_i\Big)\\ &+\cos\Big((\omega_1-\omega_2-\omega_3-\omega_4)^\top x_i\Big)\\ &+\cos\Big((\omega_1-\omega_2+\omega_3-\omega_4)^\top x_i\Big)\\ &+\cos\Big((\omega_1-\omega_2-\omega_3+\omega_4)^\top x_i\Big)\\ &]\end{aligned}$$

$$\begin{aligned}\cos\Big(\omega_1^\top x_i\Big)\cos\Big(\omega_2^\top x_i\Big)\cos\Big(\omega_3^\top x_i\Big)\sin\Big(\omega_4^\top x_i\Big) = \frac{1}{8}[&\sin\Big((\omega_1+\omega_2+\omega_3+\omega_4)^\top x_i\Big)\\ &-\sin\Big((\omega_1+\omega_2-\omega_3-\omega_4)^\top x_i\Big)\\ &-\sin\Big((\omega_1+\omega_2+\omega_3-\omega_4)^\top x_i\Big)\\ &+\sin\Big((\omega_1+\omega_2-\omega_3+\omega_4)^\top x_i\Big)\\ &+\sin\Big((\omega_1-\omega_2+\omega_3+\omega_4)^\top x_i\Big)\\ &-\sin\Big((\omega_1-\omega_2-\omega_3-\omega_4)^\top x_i\Big)\\ &-\sin\Big((\omega_1-\omega_2+\omega_3-\omega_4)^\top x_i\Big)\\ &+\sin\Big((\omega_1-\omega_2-\omega_3+\omega_4)^\top x_i\Big)\\ &]\end{aligned}$$

All terms in $\sin\Big(\omega^\top x_i\Big)$ for all ω and all terms in $\cos\Big(\omega^\top x_i\Big)$ for all $\omega \neq 0$ have a zero expectation, therefore the only contribution comes from the terms such that there is an even number of sin among $\{f_1, f_2, f_3, f_4\}$.

Given $\omega_1, \omega_2, \omega_3$, there is only a constant number of ω_4 such that there is a zero element in the set $\{\omega_1 + \epsilon\omega_2 + \epsilon'\omega_3 + \epsilon''\omega_4, \epsilon, \epsilon', \epsilon'' \in \{+1, -1\}\}$.

Therefore

$$\sum_{\omega_4 \in \Omega \setminus \{\omega_1,\omega_2,\omega_3\}} \sum_{f_1,f_2,f_3,f_4 \in \{\cos,\sin\}} \mathbb{E}[f_1(\omega_1^\top x_i) f_2(\omega_2^\top x_i) f_3(\omega_3^\top x_i) f_4(\omega_4^\top x_i)]^2 \leq C \tag{A.56}$$

$$\frac{24}{|\Omega|^4} \sum_{\omega_1,\omega_2,\omega_3,\omega_4 \in \Omega, \omega_1 \neq \omega_2 \neq \omega_3 \neq \omega_4} (c_i^{(1)} c_j^{(1)} + s_i^{(1)} s_j^{(1)})(c_i^{(2)} c_j^{(2)} + s_i^{(2)} s_j^{(2)})(c_i^{(3)} c_j^{(3)} + s_i^{(3)} s_j^{(3)})(c_i^{(4)} c_j^{(4)} + s_i^{(4)} s_j^{(4)}) \tag{A.57}$$

$$\leq \frac{24C|\Omega|^3}{|\Omega|^4} \leq \frac{C'}{|\Omega|} \tag{A.58}$$

where C is a constant

A.5 Computation of Weingarten sums

A.5.1 Diagonal observable

In order to evaluate the norm of the weight vector, we need to be able to bound expressions coming from Haar unitaries.

We prove the following result:

Theorem A.3. *Let U be a Haar random unitary and O be a Pauli string composed only of I and Z, different from identity. Let*

$$u = \sum_{i=1}^{q} (UOU^\dagger)_{m_i n_i}, \quad \forall i, m_i \neq n_i \quad \textit{and} \quad \forall i \neq j,\ (m_i, n_i) \neq (m_j, n_j) \tag{A.59}$$

Then we have

$$\mathbb{E}_{U \sim Haar}(|u|^2) = \frac{qN}{N^2 - 1} \tag{A.60}$$

Proof. We have

$$|u|^2 = \sum_{i,j}^{q} (UOU^\dagger)_{m_i n_i} (UOU^\dagger)^*_{m_j n_j} = \sum_{i,j=1}^{q} \Big(\sum_{k=1}^{N} u_{m_i k} O_k u^*_{n_i k}\Big) \Big(\sum_{k=1}^{N} u_{m_j k} O_k u^*_{n_j k}\Big)^* \tag{A.61}$$

$$= \sum_{i,j=1}^{q} \sum_{k,\ell=1}^{N} u_{m_i k} u_{n_j \ell} u^*_{n_i k} u^*_{m_j \ell}\, O_k O_\ell \tag{A.62}$$

$$\tag{A.63}$$

With the Weingarten calculus we have (CŚ06; CSV+21):

$$\mathbb{E}(u_{m_i k} u_{n_j \ell} u^*_{n_i k} u^*_{m_j \ell}) = \frac{1}{N^2 - 1} [\delta_{m_i n_i} \delta_{m_j n_j} \delta_{kk} \delta_{\ell\ell} + \delta_{m_i m_j} \delta_{n_i n_j} \delta^2_{k\ell}] \tag{A.64}$$

$$- \frac{1}{N(N^2 - 1)} [\delta_{m_i n_i} \delta_{m_j n_j} \delta^2_{kl} + \delta_{m_i m_j} \delta_{n_i n_j} \delta_{kk} \delta_{\ell\ell}] \tag{A.65}$$

$\sum_{k,\ell=1}^{N} O_k O_l = 0$ therefore we can eliminate all the terms in $\delta_{kk}\delta_{\ell\ell}$. Furthermore $\forall i\ \delta_{m_i n_i} = 0$ and we have

$$\mathbb{E}(|u|^2) = \frac{1}{N^2 - 1} \sum_{i,j=1}^{q} \sum_{k=1}^{N} \delta_{m_i m_j} \delta_{n_i n_j} O_k^2 = \frac{N}{N^2 - 1} \sum_{i,j=1}^{q} \delta_{m_i m_j} \delta_{n_i n_j} \tag{A.66}$$

We also made the hypothesis that $\forall i \neq j,\ (m_i, n_i) \neq (m_j, n_j)$ therefore $\delta_{m_i m_j} \delta_{n_i n_j} = 1$ iif $m_i = m_j$ and $n_i = n_j$ iif $i = j$. Finally we have

$$\mathbb{E}(|u|^2) = \frac{N}{N^2 - 1} \sum_{i=1}^{q} \delta_{m_i m_i} \delta_{n_i n_i} = \frac{qN}{N^2 - 1} \tag{A.67}$$

□

We will prove the following result

Theorem A.4. *Let* $u = \sum_{i=1}^{q}(UOU^\dagger)_{m_i n_i}, u' = \sum_{i=1}^{q'}(UOU^\dagger)_{m'_i n'_i}$. *If* U *is sampled from a* 4*-design then*

$$\mathbb{E}_{U\sim Haar}[|u|^4] = 2q^2 W_4(N)N^2 + \mathcal{O}(\frac{q}{N^3}) \tag{A.68}$$

If $\forall i_1$*, there exists* i_2 *such that* $(n_{i2}, m_{i2}) = (m_{i1}, n_{i1})$*, then*

$$\mathbb{E}_{U\sim Haar}[|u|^4] = 3q^2 W_4(N)N^2 + \mathcal{O}(\frac{q}{N^3}) \tag{A.69}$$

$$\mathbb{E}_{U\sim Haar}[|u|^2|u'|^2] = qq' W_4(N)N^2 + \mathcal{O}(\frac{qq'}{N^3}) \tag{A.70}$$

where we note

$$W_4(N) = \frac{N^4 - 8N^2 + 6}{N^8 - 14N^6 + 49N^4 - 36N^2}, \quad W_2(N) = -\frac{1}{N^5 - 14N^3 + 9N} \tag{A.71}$$

Proof. We have

$$|u|^4 = \sum_{i1,i2,i3,i4}(UOU^\dagger)_{m_{i1},n_{i1}}(UOU^\dagger)^*_{m_{i2},n_{i2}}(UOU^\dagger)_{m_{i3},n_{i3}}(UOU^\dagger)^*_{m_{i4},n_{i4}} \tag{A.72}$$

$$= \sum_{i1,i2,i3,i4}\sum_{k_1,k_2,k_3,k_4} u_{m_{i1},k_1}O_{k_1}u^*_{n_{i1},k_1}u^*_{m_{i2},k_2}O_{k_2}u_{n_{i2},k_2}u_{m_{i3},k_3}O_{k_3}u^*_{n_{i3},k_3}u^*_{m_{i4},s}O_{k_4}u_{n_{i4},k_4} \tag{A.73}$$

$$= \sum_{i1,i2,i3,i4}\sum_{k_1,k_2,k_3,k_4} u_{m_{i1},k_1}u_{n_{i2},k_2}u_{m_{i3},k_3}u_{n_{i4},s}u^*_{n_{i1},k_1}u^*_{m_{i2},k_2}u^*_{n_{i3},k_3}u^*_{m_{i4},k_4}O_{k_1}O_{k_2}O_{k_3}O_{k_4} \tag{A.74}$$

By using the Weingarten calculus (CŚ06), we have that

$$\mathbb{E}(u_{m_{i1},k_1}u_{n_{i2},k_2}u_{m_{i3},k_3}u_{n_{i4},s}u^*_{n_{i1},k_1}u^*_{m_{i2},k_2}u^*_{n_{i3},k_3}u^*_{m_{i4},k_4}) \tag{A.75}$$

$$= \sum_{\sigma,\tau\in S_4}\delta_{m_{i1},\sigma(m_{i1})}\cdot\delta_{m_{i3},\sigma(m_{i3})}\delta_{n_{i2},\sigma(n_{i2})}\delta_{n_{i4},\sigma(n_{i4})}\delta_{k_1,k_{\tau(1)}}\delta_{k_2,k_{\tau(2)}}\delta_{k_3,k_{\tau(3)}}\delta_{k_4,k_{\tau(4)}}W_g(\tau\sigma^{-1},N) \tag{A.76}$$

where W_g is the Weingarten function and S_4 is the set of permutations on $\{1,2,3,4\}$.

$$\mathbb{E}[|u|^4] = \sum_{i1,i2,i3,i4}\sum_{k_1,k_2,k_3,k_4}\sum_{\sigma,\tau\in S_4} O_{k_1}O_{k_2}O_{k_3}O_{k_4} \tag{A.77}$$

$$\delta_{m_{i1},\sigma(m_{i1})}\delta_{m_{i3},\sigma(m_{i3})}\delta_{n_{i2},\sigma(n_{i2})}\delta_{n_{i4},\sigma(n_{i4})}\delta_{k_1,k_{\tau(1)}}\delta_{k_2,k_{\tau(2)}}\delta_{k_3,k_{\tau(3)}}\delta_{k_4,k_{\tau(4)}}W_g(\tau\sigma^{-1},N) \tag{A.78}$$

$$= \sum_{i1,i2,i3,i4}\sum_{\sigma\in S_4}\delta_{m_{i1},\sigma(m_{i1})}\delta_{m_{i3},\sigma(m_{i3})}\delta_{n_{i2},\sigma(n_{i2})}\delta_{n_{i4},\sigma(n_{i4})} \tag{A.79}$$

$$\sum_{\tau\in S_4}W_g(\tau\sigma^{-1},N)\sum_{k_1,k_2,k_3,k_4}\delta_{k_1,k_{\tau(1)}}\delta_{k_2,k_{\tau(2)}}\delta_{k_3,k_{\tau(3)}}\delta_{k_4,k_{\tau(4)}}O_{k_1}O_{k_2}O_{k_3}O_{k_4} \tag{A.80}$$

$$= \sum_{i1,i2,i3,i4}\sum_{\sigma,\tau\in S_4}\delta_{m_{i1},\sigma(m_{i1})}\delta_{m_{i3},\sigma(m_{i3})}\delta_{n_{i2},\sigma(n_{i2})}\delta_{n_{i4},\sigma(n_{i4})}A(\tau)W_g(\tau\sigma^{-1},N) \tag{A.81}$$

$$\tag{A.82}$$

where we note $A(\tau) = \sum_{k_1,k_2,k_3,k_4} \delta_{k_1,k_{\tau(1)}}\delta_{k_2,k_{\tau(2)}}\delta_{k_3,k_{\tau(3)}}\delta_{k_4,k_{\tau(4)}}O_{k_1}O_{k_2}O_{k_3}O_{k_4}$.

Let us look at the term $A(\tau)$ for a given permutation τ. We can verify that only for the permutations included in table A.1 the sum will be non zero. For instance, $A(\mathrm{id}) = \sum_{k_1,k_2,k_3,k_4} O_{k_1}O_{k_2}O_{k_3}O_{k_4} = \mathrm{Tr}(O)^4 = 0$, and $A(\tau_1) = \sum_{k_1,k_2,k_3,k_4} \delta_{k_1,k_2}\delta_{k_2,k_1}\delta_{k_3,k_4}\delta_{k_4,k_3}O_{k_1}O_{k_2}O_{k_3}O_{k_4} = \sum_{k_1,k_3} O_{k_1}^2 O_{k_3}^2 = N^2$.

So we have

$$\mathbb{E}[|u|^4] = \sum_{\sigma\in S_4}\sum_{i1,i2,i3,i4} \delta_{m_{i1},\sigma(m_{i1})}\delta_{m_{i3},\sigma(m_{i3})}\delta_{n_{i2},\sigma(n_{i2})}\delta_{n_{i4},\sigma(n_{i4})} \sum_{\tau\in\{\tau_1,\tau_2,\tau_3\}} W_g(\tau\sigma^{-1},N)N^2 \tag{A.83}$$

Let us look at the term $\sum_{i1,i2,i3,i4} \delta_{m_{i1},\sigma(m_{i1})}\delta_{m_{i3},\sigma(m_{i3})}\delta_{n_{i2},\sigma(n_{i2})}\delta_{n_{i4},\sigma(n_{i4})}$ for a given permutation σ.

Since we assumed all elements $(UOU^\dagger)_{m_in_i}$ are off-diagonal, we have that $\delta_{m_i,n_i} = 0$, and it restricts the permutations potentially giving non-zero terms. These permutations are enumerated in table A.2.

For each permutation σ we compute the dominant term in $\sum_\tau A(\tau)W_g(\tau\sigma^{-1},N)$. These numbers are enumerated in table A.3. We compute the number of non zero terms in

$\sum_{i1,i2,i3,i4} \delta_{m_{i1},\sigma(m_{i1})}\delta_{m_{i3},\sigma(m_{i3})}\delta_{n_{i2},\sigma(n_{i2})}\delta_{n_{i4},\sigma(n_{i4})}$. These numbers are enumerated in table A.2

From (Fuk99), if $\sigma \neq \mathrm{id}$, $W_g(\sigma,N) = \mathcal{O}(\frac{1}{N^5})$ and $W_g(\mathrm{id},N) = \dfrac{N^4-8N^2+6}{N^8-14N^6+49N^4-36N^2} = \mathcal{O}(\dfrac{1}{N^4})$. We can verify that only for $\sigma_1,\sigma_4,\sigma_7$ there exist $\tau \in \tau_1,\tau_2,\tau_3$ such that $\tau\sigma^{-1} = \mathrm{id}$, and τ is unique.

By combining the numbers in tables A.2 and A.4 we have the final result.

	k_1	$\sigma_O(k_2)$	k_3	$\sigma_O(k_4)$	1	2	3	4	$A(\tau)$
τ_1	k_2	$\sigma_O(k_1)$	k_4	$\sigma_O(k_3)$	2	1	4	3	N^2
τ_2	k_4	$\sigma_O(k_3)$	k_2	$\sigma_O(k_1)$	4	3	2	1	N^2
τ_3	$\sigma_O(k_3)$	k_4	$\sigma_O(k_1)$	k_2	3	4	1	2	N^2
τ_4	k_2	k_4	$\sigma_O(k_1)$	$\sigma_O(k_3)$	2	4	1	3	N
τ_5	k_2	$\sigma_O(k_3)$	k_4	$\sigma_O(k_1)$	2	3	4	1	N
τ_6	k_4	$\sigma_O(k_1)$	k_2	$\sigma_O(k_3)$	4	1	2	3	N
τ_7	k_4	$\sigma_O(k_3)$	$\sigma_O(k_1)$	k_2	4	3	1	2	N
τ_8	$\sigma_O(k_3)$	k_4	k_2	$\sigma_O(k_1)$	3	4	2	1	N
τ_9	$\sigma_O(k_3)$	k_1	k_4	k_2	3	1	4	2	N

Table A.1: τ permutations that give non zero values for non diagonal O. For diagonal O, consider $\sigma_O = id$.

	m_{i_1}	n_{i_2}	m_{i_3}	n_{i_4}	Number of non-zero terms
σ_1	m_{i_2}	n_{i_1}	m_{i_4}	n_{i_3}	q^2
σ_2	m_{i_2}	n_{i_3}	m_{i_4}	n_{i_1}	$q + \mathcal{O}(q)$
σ_3	m_{i_2}	m_{i_4}	n_{i_1}	n_{i_3}	$\mathcal{O}(q)$
σ_4	m_{i_4}	n_{i_3}	m_{i_2}	n_{i_1}	q^2
σ_5	m_{i_4}	n_{i_1}	m_{i_2}	n_{i_3}	$q + \mathcal{O}(q)$
σ_6	m_{i_4}	n_{i_3}	n_{i_1}	m_{i_2}	$\mathcal{O}(q)$
σ_7	n_{i_3}	m_{i_4}	n_{i_1}	m_{i_2}	0 or q^2

Table A.2: σ permutations to be considered

Let us now look at

$$u_1 = \sum_{i=1}^{p_1} (UOU^\dagger)_{m_i^{(1)} n_i^{(1)}}, \quad \forall i, m_i^{(1)} \neq n_i^{(1)} \quad \text{and} \quad \forall i \neq j,\ (m_i^{(1)}, n_i^{(1)}) \neq (m_j^{(1)}, n_j^{(1)}) \tag{A.84}$$

$$u_2 = \sum_{i=1}^{p_2} (UOU^\dagger)_{m_i^{(2)} n_i^{(2)}}, \quad \forall i, m_i^{(2)} \neq n_i^{(2)} \quad \text{and} \quad \forall i \neq j,\ (m_i^{(2)}, n_i^{(2)}) \neq (m_j^{(2)}, n_j^{(1)}) \tag{A.85}$$

$$|u_1|^2 |u_2|^2 = \Big(\sum_{i,j=1}^{p_1} \sum_{k,\ell=1}^{N} u_{m_i^{(1)} k} u_{n_j^{(1)} \ell} u^*_{n_i^{(1)} k} u^*_{m_j^{(1)} \ell}\, O_k O_\ell \Big) \Big(\sum_{i,j=1}^{p_2} \sum_{k,\ell=1}^{N} u_{m_i^{(2)} k} u_{n_j^{(2)} \ell} u^*_{n_i^{(2)} k} u^*_{m_j^{(2)} \ell}\, O_k O_\ell \Big) \tag{A.86}$$

$$= \sum_{i1,i2,i3,i4} \sum_{k_1,k_2,k_3,k_4} u_{m_{i1}^{(1)},k_1} u_{n_{i2}^{(1)},k_2} u_{m_{i3}^{(2)},k_3} u_{n_{i4}^{(2)},s} u^*_{n_{i1}^{(1)},k_1} u^*_{m_{i2}^{(1)},k_2} u^*_{n_{i3}^{(2)},k_3} u^*_{m_{i4}^{(2)},k_4} O_{k_1} O_{k_2} O_{k_3} O_{k_4} \tag{A.87}$$

$$\tag{A.88}$$

$\mathbb{E}[|u|^4] = 2q^2(W_4(N)N^2 + 4NW_2(N)) + \Theta(q)(2W_2(N)N^2 + NW_4(N)) = 2q^2 W_4(N)N^2 + \mathcal{O}(\frac{q}{N^3})$

We do the same reasoning as above, but this time the number of non zero term for the permutations σ are given table A.3.

□

	$m_{i_1}^{(1)}$	$n_{i_2}^{(1)}$	$m_{i_3}^{(2)}$	$n_{i_4}^{(2)}$	Number of non-zero terms
σ_1	$m_{i_2}^{(1)}$	$n_{i_1}^{(1)}$	$m_{i_4}^{(2)}$	$n_{i_3}^{(2)}$	$q_1 q_2$
σ_2	$m_{i_2}^{(1)}$	$n_{i_3}^{(2)}$	$m_{i_4}^{(2)}$	$n_{i_1}^{(1)}$	$\mathcal{O}(q_1 q_2)$
σ_3	$m_{i_2}^{(1)}$	$m_{i_4}^{(2)}$	$n_{i_1}^{(1)}$	$n_{i_3}^{(2)}$	$\mathcal{O}(q_1 q_2)$
σ_4	$m_{i_4}^{(2)}$	$n_{i_3}^{(2)}$	$m_{i_2}^{(1)}$	$n_{i_1}^{(1)}$	0
σ_5	$m_{i_4}^{(2)}$	$n_{i_1}^{(1)}$	$m_{i_2}^{(1)}$	$n_{i_3}^{(2)}$	$\mathcal{O}(q_1 q_2)$
σ_6	$m_{i_4}^{(2)}$	$n_{i_3}^{(2)}$	$n_{i_1}^{(1)}$	$m_{i_2}^{(1)}$	$\mathcal{O}(q_1 q_2)$
σ_7	$n_{i_3}^{(2)}$	$m_{i_4}^{(2)}$	$n_{i_1}^{(1)}$	$m_{i_2}^{(1)}$	0

Table A.3: σ permutations to be considered in the cross terms

σ	$n(\sigma, W_4, N^2)$	$n(\sigma, W_2, N^2)$	$n(\sigma, W_4, N)$	$n(\sigma, W_2, N)$	$\sum_\tau A(\tau)W_g(\tau\sigma^{-1}, N)$
σ_1	1	0	0	4	$N^2W_4(N) + 4NW_2(N)$
σ_2	0	2	1	0	$NW_4(N) + 2N^2W_2(N)$
σ_3	0	2	1	0	$NW_4(N) + 2N^2W_2(N)$
σ_4	1	0	0	4	$N^2W_4(N) + 4NW_2(N)$
σ_5	0	2	1	0	$NW_4(N) + 2N^2W_2(N)$
σ_6	0	2	1	0	$NW_4(N) + 2N^2W_2(N)$
σ_7	1	0	0	4	$N^2W_4(N) + 4NW_2(N)$

Table A.4: σ permutations to be considered in the cross terms. For each permutation σ, we compute the dominant factors in the term $\sum_\tau A(\tau)W_g(\tau\sigma^{-1}, N)$. To do so, for each σ, we compute the number of permutations τ such that $A(\tau)W_g(\tau\sigma^{-1}, N)$ is equal to $N^2W_4(N), NW_4(N), N^2W_2(N), NW_2(N)$. We use a computer to find the numbers in the first four columns.

Theorem A.5. *Let U be drawn from a 2 design. We have that*

$$\mathbb{E}[||\beta||^2] = \frac{p}{N+1} \tag{A.89}$$

Proof.

$$\mathbb{E}[||\beta||^2] = \frac{p}{N^2}\mathbb{E}[|u_0|^2] + 2\frac{p}{N^2}\sum_{\omega\in\Omega^*}\mathbb{E}[|u_\omega|^2] = \frac{p}{N^2}\frac{N}{N^2-1}[R(0) + 2\sum_{\omega\in\Omega^*}R(\omega)] \tag{A.90}$$

$$= \frac{p}{N}\frac{N(N-1)}{N^2-1} = \frac{p}{N+1} \tag{A.91}$$

□

Theorem A.6. *Let U be drawn from a 4 design. We have that*

$$\mathbb{V}[||\beta||^2] = \Theta(\frac{p^2}{N^6}\sum_{\omega\in\Omega}R(\omega)^2 + \frac{p^2}{N^4}) \tag{A.92}$$

Proof. We note

$$\mathbb{E}[||\beta||^4] = \Big(\frac{p}{N^2}(|u_0|^2 + 2\sum_{\omega\in\Omega^*}|u_\omega|^2)\Big)^2 \tag{A.93}$$

$$= \frac{p^2}{N^4}\Big(\mathbb{E}[|u_0|^4| + 2\sum_{\omega\in\Omega^*}\mathbb{E}[|u_\omega|^2|u_0|^2] + 4\sum_{\omega\in\Omega^*}\mathbb{E}[|u_\omega|^4] + 4\sum_{\omega\neq\omega'\in\Omega^*}\mathbb{E}[|u_\omega|^2|u_{\omega'}|^2]\Big) \tag{A.94}$$

$$= \frac{p^2}{N^4}\Big([3R(0)^2 + 4\sum_{\omega\in\Omega^*}2R(\omega)^2]N^2W_4(N) + \mathcal{O}[R(0) + 2\sum_{\omega\in\Omega^*}R(\omega)]N^2W_2(N) \tag{A.95}$$

$$+ [2R(0)\sum_{\omega\in\Omega^*}R(\omega) + 4\sum_{\omega\neq\omega'\in\Omega^*}R(\omega)R(\omega')]N^2W_4(N) \tag{A.96}$$

$$+ \mathcal{O}[2R(0)\sum_{\omega\in\Omega^*}R(\omega) + 4\sum_{\omega\neq\omega'\in\Omega^*}R(\omega)R(\omega')]N^2W_2(N)) \tag{A.97}$$

$$= \frac{p^2}{N^2}\Big([3R(0)^2 + 8\sum_{\omega\in\Omega^*}R(\omega)^2 + 2R(0)\sum_{\omega\in\Omega^*}R(\omega) + 4\sum_{\omega\neq\omega'\in\Omega^*}R(\omega)R(\omega')]W_4(N) \tag{A.98}$$

$$+ \mathcal{O}[R(0) + 2\sum_{\omega\in\Omega^*}R(\omega) + 2R(0)\sum_{\omega\in\Omega^*}R(\omega) + 4\sum_{\omega\neq\omega'\in\Omega^*}R(\omega)R(\omega')]W_2(N)) \tag{A.99}$$

$$= \frac{p^2}{N^2}\Big([(R(0) + 2\sum_{\omega\in\Omega^*}R(\omega))^2 + 2R(0)^2 + 4\sum_{\omega\in\Omega^*}R(\omega)^2 - 2R(0)\sum_{\omega\in\Omega^*}R(\omega)]W_4(N) \tag{A.100}$$

$$+ \mathcal{O}[R(0) + 2\sum_{\omega\in\Omega^*}R(\omega)(1 + R(0) + 2\sum_{\omega\in\Omega^*}R(\omega))]W_2(N)) \tag{A.101}$$

$$= \frac{p^2}{N^2}\Big([N^2(N-1)^2 + 2R(0)^2 + 4\sum_{\omega\in\Omega^*}R(\omega)^2 - 2R(0)\sum_{\omega\in\Omega^*}R(\omega)]W_4(N) \tag{A.102}$$

$$+ \mathcal{O}[R(0) + N^2]W_2(N)) \tag{A.103}$$

$$\mathbb{E}[||\beta||^4] - \mathbb{E}[||\beta||^2]^2 = \frac{p^2}{N^2}\Big([N^2(N-1)^2 + 2R(0)^2 + 4\sum_{\omega\in\Omega^*}R(\omega)^2 - 2R(0)\sum_{\omega\in\Omega^*}R(\omega)]W_4(N) \tag{A.104}$$

$$+ \mathcal{O}[R(0) + N^2]W_2(N)) - \frac{p^2}{(N+1)^2} \tag{A.105}$$

$$\frac{p^2}{N^2}N^2(N-1)^2W_4(N) - \frac{p^2}{(N+1)^2} = \frac{p^2(N-1)^2(N^4-8N^2+6)}{N^8-14N^6+49N^4-36N^2} - \frac{p^2}{(N+1)^2} \tag{A.106}$$

$$= \frac{p^2(N-1)^2(N^4-8N^2+6)}{N^2(N^2-1)(N^4-13N^2+36))} - \frac{p^2}{(N+1)^2} \tag{A.107}$$

$$= \frac{p^2(N-1)(N^4-8N^2+6)}{N^2(N+1)(N^4-13N^2+36)} - \frac{p^2}{(N+1)^2} \tag{A.108}$$

$$= \frac{p^2(N-1)(N+1)(N^4-8N^2+6) - p^2N^2(N^4-13N^2+36))}{N^2(N+1)^2(N^4-13N^2+36))} \tag{A.109}$$

$$= \frac{4p^2N^4 + \mathcal{O}(p^2N^2)}{(N+1)^2N^2(N^4-13N^2+36))} \tag{A.110}$$

$$= \Theta(\frac{p^2}{N^4}) \tag{A.111}$$

$$W_4(N) = \Theta(\frac{1}{N^4}) \tag{A.112}$$

$$N^2W_2(N) = \mathcal{O}(\frac{1}{N^3}) \tag{A.113}$$

$$\mathbb{E}[||\beta||^4] - \mathbb{E}[||\beta||^2]^2 = \Theta(\frac{p^2}{N^6}\sum_{\omega\in\Omega} R(\omega)^2) + \Theta(\frac{p^2}{N^4}) \tag{A.114}$$

□

A.5.2 Non diagonal observable

We do the same computation as above, but we no longer assume that $O \in \{I, Z\}^{\otimes n}$. We indeed assume that O is a general Pauli observable. Then there exists a permutation $\sigma_O \neq id$ such that for each row k of O, $O_{k,\sigma_O(k)} \neq 0$. Because O is hermitian we have $\sigma_O^2 = id$ And we have

$$(UOU^\dagger)_{ij} = \sum_{k=1}^{n} u_{ik}O_{k,\sigma_O(k)}u^*_{j,\sigma_O(k)} \tag{A.115}$$

and

$$|(UOU^\dagger)_{ij}|^2 = \Big(\sum_{k=1}^{N} u_{ik}O_{k,\sigma_O(k)}u^*_{j,\sigma_O(k)}\Big)\Big(\sum_{k=1}^{N} u_{ik}O_{k,\sigma_O(k)}u^*_{j,\sigma_O(k)}\Big)^* \tag{A.116}$$

$$= \sum_{k,\ell=1}^{N} u_{ik}O_{k,\sigma_O(k)}u^*_{j,\sigma_O(k)} \quad u^*_{i\ell}O^*_{\ell,\sigma_O(\ell)}u_{j,\sigma_O(\ell)} \tag{A.117}$$

$$= \sum_{k,\ell=1}^{N} u_{ik}u_{j,\sigma_O(\ell)}u^*_{j,\sigma_O(k)}u^*_{i\ell}O_{k,\sigma_O(k)}O^*_{\ell,\sigma_O(\ell)} \tag{A.118}$$

More precisely, we want to evaluate $\mathbb{E}[|u|^2]$ where

$$u = \sum_{i=1}^{q}(UOU^\dagger)_{m_in_i}, \quad \forall i, m_i \neq n_i \quad \text{and} \quad \forall i \neq j,\ (m_i, n_i) \neq (m_j, n_j) \tag{A.119}$$

We have

$$|u|^2 = \sum_{i,j}^{q} (UOU^\dagger)_{m_i n_i} (UOU^\dagger)^*_{m_j n_j} \tag{A.120}$$

$$= \sum_{i,j=1}^{q} \Big(\sum_{k=1}^{N} u_{m_i k} O_{k,\sigma_O(k)} u^*_{n_i,\sigma_O(k)} \Big) \Big(\sum_{k=1}^{N} u_{m_j k} O_{k,\sigma_O(k)} u^*_{n_j \sigma_O(k)} \Big)^* \tag{A.121}$$

$$= \sum_{i,j=1}^{q} \sum_{k,\ell=1}^{N} u_{m_i k} u_{n_j \sigma_O(\ell)} u^*_{n_i \sigma_O(k)} u^*_{m_j \ell}\, O_{k,\sigma_O(k)} O^*_{\ell,\sigma_O(\ell)} \tag{A.122}$$

$$\mathbb{E}(u_{m_i k} u_{n_j \sigma_O(\ell)} u^*_{n_i \sigma_O(k)} u^*_{m_j \ell}) = \frac{1}{N^2-1} [\delta_{m_i n_i} \delta_{m_j n_j} \delta_{k,\sigma_O(k)} \delta_{\ell,\sigma_O(l)} + \delta_{m_i m_j} \delta_{n_i n_j} \delta_{kl} \delta_{\sigma_O(l)\sigma_O(k)}] \tag{A.123}$$

$$- \frac{1}{N(N^2-1)} [\delta_{m_i n_i} \delta_{m_j n_j} \delta_{kl} \delta_{\sigma_O(l)\sigma_O(k)} + \delta_{m_i m_j} \delta_{n_i n_j} \delta_{k,\sigma_O(k)} \delta_{\ell,\sigma_O(l)}] \tag{A.124}$$

We have that

$$\forall k, l,\ \delta_{k,\sigma_O(k)} \delta_{\ell,\sigma_O(l)} = 0, \quad \sum_{k,\ell=1}^{N} \delta_{kl} \delta_{\sigma_O(l)\sigma_O(k)} O_{k,\sigma_O(k)} O_{\ell,\sigma_O(\ell)} = \sum_{k=1}^{N} O_{k,\sigma_O(k)} O^*_{k,\sigma_O(k)} = N \tag{A.125}$$

$$\mathbb{E}[|u|^2] = \frac{1}{N^2-1} \sum_{i,j=1}^{q} \delta_{m_i m_j} \delta_{n_i n_j} N - \frac{1}{N(N^2-1)} \sum_{i,j=1}^{q} \delta_{m_i n_i} \delta_{m_j n_j} N \tag{A.126}$$

$$= \frac{qN}{N^2-1} \tag{A.127}$$

We note

$$u_1 = \sum_{i=1}^{q} (UOU^\dagger)_{m_i^{(1)} n_i^{(1)}} \tag{A.128}$$

$$u_2 = \sum_{i=1}^{q} (UOU^\dagger)_{m_i^{(2)} n_i^{(2)}} \tag{A.129}$$

$$\tag{A.130}$$

We want to compute $\mathbb{E}[u_1 u_2^*]$ and $\mathbb{E}[u_1 u_2]$.

We have

$$u_1 u_2^* = \sum_{i=1}^{q_1}\sum_{j=1}^{q_2}(UOU^\dagger)_{m_i^{(1)}n_i^{(1)}}(UOU^\dagger)^*_{m_j^{(2)}n_j^{(2)}} \tag{A.131}$$

$$= \sum_{i=1}^{q_1}\sum_{j=1}^{q_2}\Big(\sum_{k=1}^{N} u_{m_i^{(1)}k}O_{k,\sigma_O(k)}u^*_{n_i^{(1)},\sigma_O(k)}\Big)\Big(\sum_{k=1}^{N} u_{m_j^{(2)}k}O_{k,\sigma_O(k)}u^*_{n_j^{(2)}\sigma_O(k)}\Big)^* \tag{A.132}$$

$$= \sum_{i=1}^{q_1}\sum_{j=1}^{q_2}\sum_{k,\ell=1}^{N} u_{m_i^{(1)}k}u_{n_j^{(2)}\sigma_O(\ell)}u^*_{n_i^{(1)}\sigma_O(k)}u^*_{m_j^{(2)}\ell}\,O_{k,\sigma_O(k)}O^*_{\ell,\sigma_O(\ell)} \tag{A.133}$$

$$\mathbb{E}(u_{m_i^{(1)}k}u_{n_j^{(2)}\sigma_O(\ell)}u^*_{n_i^{(1)}\sigma_O(k)}u^*_{m_j^{(2)}\ell}) = \frac{1}{N^2-1}[\delta_{m_i^{(1)}n_i^{(1)}}\delta_{m_j^{(2)}n_j^{(2)}}\delta_{k,\sigma_O(k)}\delta_{\ell,\sigma_O(l)} + \delta_{m_i^{(1)}m_j^{(2)}}\delta_{n_i^{(1)}n_j^{(2)}}\delta_{kl}\delta_{\sigma_O(l)\sigma_O(k)}] \tag{A.134}$$

$$- \frac{1}{N(N^2-1)}[\delta_{m_i^{(1)}n_i^{(1)}}\delta_{m_j^{(2)}n_j^{(2)}}\delta_{kl}\delta_{\sigma_O(l)\sigma_O(k)} + \delta_{m_i^{(1)}m_j^{(2)}}\delta_{n_i^{(1)}n_j^{(2)}}\delta_{k,\sigma_O(k)}\delta_{\ell,\sigma_O(l)}] \tag{A.135}$$

$$u_1 u_2 = \sum_{i=1}^{q_1}\sum_{j=1}^{q_2}\Big(\sum_{k=1}^{N} u_{m_i^{(1)}k}O_{k,\sigma_O(k)}u^*_{n_i^{(1)},\sigma_O(k)}\Big)\Big(\sum_{k=1}^{N} u_{m_j^{(2)}k}O_{k,\sigma_O(k)}u^*_{n_j^{(2)}\sigma_O(k)}\Big) \tag{A.136}$$

$$= \sum_{i=1}^{q_1}\sum_{j=1}^{q_2}\sum_{k,\ell=1}^{N} u_{m_i^{(1)}k}u_{m_j^{(2)}\sigma_O(\ell)}u^*_{n_i^{(1)}\sigma_O(k)}u^*_{n_j^{(2)}\ell}\,O_{k,\sigma_O(k)}O^*_{\ell,\sigma_O(\ell)} \tag{A.137}$$

$$\mathbb{E}(u_{m_i^{(1)}k}u_{n_j^{(2)}\sigma_O(\ell)}u^*_{n_i^{(1)}\sigma_O(k)}u^*_{m_j^{(2)}\ell}) = \frac{1}{N^2-1}[\delta_{m_i^{(1)}n_i^{(1)}}\delta_{m_j^{(2)}n_j^{(2)}}\delta_{k,\sigma_O(k)}\delta_{\ell,\sigma_O(l)} + \delta_{m_i^{(1)}n_j^{(2)}}\delta_{n_i^{(1)}m_j^{(2)}}\delta_{kl}\delta_{\sigma_O(l)\sigma_O(k)}] \tag{A.138}$$

$$- \frac{1}{N(N^2-1)}[\delta_{m_i^{(1)}n_i^{(1)}}\delta_{m_j^{(2)}n_j^{(2)}}\delta_{kl}\delta_{\sigma_O(l)\sigma_O(k)} + \delta_{m_i^{(1)}n_j^{(2)}}\delta_{n_i^{(1)}m_j^{(2)}}\delta_{k,\sigma_O(k)}\delta_{\ell,\sigma_O(l)}] \tag{A.139}$$

$$\mathbb{E}[u_1 u_2^*] = \frac{1}{N^2-1}\sum_{i=1}^{q_1}\sum_{j=1}^{q_2}\delta_{m_i^{(1)}m_j^{(2)}}\delta_{n_i^{(1)}n_j^{(2)}}N - \frac{1}{N(N^2-1)}\sum_{i=1}^{q_1}\sum_{j=1}^{q_2}\delta_{m_i^{(1)}n_i^{(1)}}\delta_{m_j^{(2)}n_j^{(2)}}N = 0 \tag{A.140}$$

$$\mathbb{E}[u_1 u_2] = \frac{1}{N^2-1}\sum_{i=1}^{q_1}\sum_{j=1}^{q_2}\delta_{m_i^{(1)}n_j^{(2)}}\delta_{n_i^{(1)}m_j^{(2)}}N - \frac{1}{N(N^2-1)}\sum_{i=1}^{q_1}\sum_{j=1}^{q_2}\delta_{m_i^{(1)}n_i^{(1)}}\delta_{m_j^{(2)}n_j^{(2)}}N \tag{A.141}$$

$$= 0 \tag{A.142}$$

$$|u|^4 = \sum_{i1,i2,i3,i4} (UOU^\dagger)_{m_{i1},n_{i1}} (UOU^\dagger)^*_{m_{i2},n_{i2}} (UOU^\dagger)_{m_{i3},n_{i3}} (UOU^\dagger)^*_{m_{i4},n_{i4}} \tag{A.143}$$

$$= \sum_{i1,i2,i3,i4} \sum_{k_1,k_2,k_3,k_4} \tag{A.144}$$

$$u_{m_{i1},k_1} O_{k_1,\sigma_O(k_1)} u^*_{n_{i1},\sigma_O(k_1)} u^*_{m_{i2},k_2} O^*_{k_2,\sigma_O(k_2)} u_{n_{i2},\sigma_O(k_2)} u_{m_{i3},k_3} O_{k_3,\sigma_O(k_3)} u^*_{n_{i3},\sigma_O(k_3)} u^*_{m_{i4},k_4} O_{k_4,\sigma_O(k_4)} u^*_{n_{i4},\sigma_O(k_4)} \tag{A.145}$$

$$= \sum_{i1,i2,i3,i4} \sum_{k_1,k_2,k_3,k_4} \tag{A.146}$$

$$u_{m_{i1},k_1} u_{n_{i2},\sigma_O(k_2)} u_{m_{i3},k_3} u_{n_{i4},\sigma_O(k_4)} u^*_{n_{i1},\sigma_O(k_1)} u^*_{m_{i2},k_2} u^*_{n_{i3},\sigma_O(k_3)} u^*_{m_{i4},k_4} O_{k_1,\sigma_O(k_1)} O^*_{k_2,\sigma_O(k_2)} O_{k_3,\sigma_O(k_3)} O^*_{k_4,\sigma_O(k_4)} \tag{A.147}$$

$$\mathbb{E}[|u|^4] = \sum_{i1,i2,i3,i4} \sum_{\sigma,\tau \in S_4} \delta_{m_{i1},\sigma(m_{i1})} \delta_{m_{i3},\sigma(m_{i3})} \delta_{n_{i2},\sigma(n_{i2})} \delta_{n_{i4},\sigma(n_{i4})} A(\tau) W_g(\tau\sigma^{-1}, N) \tag{A.148}$$

$$\tag{A.149}$$

where $A(\tau) = \sum_{k_1,k_2,k_3,k_4} \delta_{k_1,\tau(k_1)} \delta_{\sigma_O(k_2),\tau(\sigma_O(k_2))} \delta_{k_3,\tau(k_3)} \delta_{\sigma_O(k_4),\tau(\sigma_O(k_4))} O_{k_1,\sigma_O(k_1)} O^*_{k_2,\sigma_O(k_2)} O_{k_3,\sigma_O(k_3)} O^*_{k_4,\sigma_O(k_4)}$

We then do the same reasoning as with O diagonal.

A.6 A Fourier model with random coefficients

We consider the following function $f : \mathbb{R}^d \longrightarrow \mathbb{R}$

$$f_\beta(x) = \frac{1}{\sqrt{p}} \sum_{\omega \in [-L,L]^d} (\beta_{\omega,\cos} \cos\left(\omega^\top x\right) + \beta_{\omega,\sin} \sin\left(\omega^\top x\right)) \tag{A.150}$$

with $p = (2L+1)^d$, and β_ω are iid uniform random variables in the interval $[-\sigma, \sigma]$ with $\sigma = \Theta(1/\text{poly(d, L)})$. Unless needed, we will simplify the notation with $f = f_\beta$.

Theorem A.7. *We have the following properties:*

- $|||\beta||^2 - p\sigma^2| \leq \sigma\sqrt{2p\log(1/\delta)}$ *with probability at least* $1-\delta$
- $\mathbb{V}_x[f(x)] \geq \sigma^2 - \frac{\sigma}{\sqrt{p}}\sqrt{2\log(1/\delta)}$ *with probability at least* $1-\delta$
- $|f(x)| \leq 1 \ \forall x \in \mathbb{R}^d$ *with high probability*

Proof. We first remark that for all ω, $\mathbb{E}[\beta^2_{\omega,\cos}] = \mathbb{E}[\beta^2_{\omega,\sin}] = \sigma^2/2$.

The first item is proven by applying the Hoeffding inequality to the random variable $||\beta||^2 = \sum_\omega \beta^2_{\omega,\cos} + \beta^2_{\omega,\sin}$ since every element of the sum is iid, and $\mathbb{E}[||\beta||^2] = p\sigma^2$.

To prove the second item, we remind that for all $\omega \neq 0$, $\mathbb{E}_x[\cos\left(\omega^\top x\right)] = \mathbb{E}_x[\cos\left(\omega^\top x\right)] = 0$ and $\mathbb{E}_x[\cos\left(\omega^\top x\right)^2] = \mathbb{E}_x[\sin\left(\omega^\top x\right)^2] = 1/2$, $\mathbb{E}_x[\cos\left(\omega^\top x\right)\sin\left(\omega^\top x\right)] = 0$ and for all $\omega \neq \omega'$, $\mathbb{E}_x[\cos\left(\omega^\top x\right)\sin\left(\omega'^\top x\right)] = \mathbb{E}_x[\cos\left(\omega^\top x\right)\cos\left(\omega'^\top x\right)] = \mathbb{E}_x[\sin\left(\omega^\top x\right)\sin\left(\omega'^\top x\right)] = 0$.

Then we have $\mathbb{E}[f(x)] = 0$ and $\mathbb{E}[f(x)^2] = \frac{1}{p}\sum_{\omega} \beta_{\omega,\cos}^2 + \beta_{\omega,\cos}^2 = \frac{||\beta||^2}{p}$ and we apply the result of the first item.

We will now show that $|f(x)| \leq 1$ on $\mathbb{R}^d$.

First, we show that $\forall x, \mathbb{P}(|f(x)| \leq \epsilon) \leq \exp\left(-\frac{\epsilon^2}{2\sigma^2}\right)$. Let $i_0 \in [1, p]$ and $\beta' = (\beta_{\omega_1,\cos}, \beta_{\omega_1,\sin} \dots \beta'_{\omega_{i_0},\cos}, \beta_{\omega_{i_0},\sin}, \dots \beta_{\omega_p})$.

$$|f_\beta(x) - f_{\beta'}(x)| = \frac{1}{\sqrt{p}} \sum_{\omega \neq \omega_{i_0}} (\beta_{\omega,\cos} \cos\left(\omega^\top x\right) + \beta_{\omega,\sin} \sin\left(\omega^\top x\right)) + \frac{\beta'_{\omega_{i_0},\cos}}{\sqrt{p}} \cos\left(\omega^\top x\right) + \frac{\beta_{\omega_{i_0},\sin}}{\sqrt{p}} \sin\left(\omega^\top x\right) \tag{A.151}$$

$$- \frac{1}{\sqrt{p}} \sum_{\omega \neq \omega_{i_0}} (\beta_{\omega,\cos} \cos\left(\omega^\top x\right) + \beta_{\omega,\sin} \sin\left(\omega^\top x\right)) - \frac{\beta'_{\omega_{i_0},\cos}}{\sqrt{p}} \cos\left(\omega^\top x\right) - \frac{\beta_{\omega_{i_0},\sin}}{\sqrt{p}} \sin\left(\omega^\top x\right) \tag{A.152}$$

$$= \frac{1}{\sqrt{p}} \cos\left(\omega^\top x\right) |\beta_{\omega_{i_0},\cos} - \beta'_{\omega_{i_0},\cos}| \tag{A.153}$$

$$\leq \frac{2\sigma}{\sqrt{p}} \tag{A.154}$$

The last inequality comes from $|\beta_{\omega_{i_0}} - \beta'_{\omega_{i_0}}| \leq 2\sigma$ and $|\cos\left(\omega^\top x\right)| \leq 1$. The same result can be obtain with $\beta = (\beta_{\omega_1,\cos}, \beta_{\omega_1,\sin} \dots \beta_{\omega_{i_0},\cos}, \beta'_{\omega_{i_0},\sin}, \dots \beta_{\omega_p})$. Furthermore, for all x, $\mathbb{E}_\beta[f_\beta(x)] = 0$

From McDiarmid's inequality

$$\mathbb{P}(|f_\beta(x)| \geq \epsilon) \leq \exp\left(-\frac{2\epsilon^2}{\sum_{i=1}^p \frac{4\sigma^2}{p}}\right) \leq \exp\left(-\frac{\epsilon^2}{2\sigma^2}\right) \tag{A.155}$$

Now we prove that $|f| \leq 1$ on $\mathbb{R}^d$. f is a periodic function, so we only need to prove that f is bounded over the domain $\mathcal{X} = [0, 2\pi]^d$. We consider a covering net of radius r of $\mathcal{X}$. Let $T = \{r_1, \dots r_T\}$ be the set of centers of the balls composing the covering net.

Let $r = (r_1, \dots r_d) \in \S$ and $u = (u_1, \dots u_d) \in \S$ such that $||r - u|| \leq \epsilon$.

Then we have, by applying Taylor's formulas

$$f(r+u) - f(r) = \sum_{i=1}^d \partial_i f(r) u_i + \sum_{i,j} R_{ij}(r+u) u_i u_j \tag{A.156}$$

$$= u^\top \nabla f(r) + u^\top R(r+u) u \tag{A.157}$$

$$\text{with} \quad [R(r+u)]_{ij} = R_{ij}(r+u) \; = \; S_{ij} \int_0^1 (1-t) \partial_{ij} f(r+tu) dt \tag{A.158}$$

where $S_{ij} = 1$ if $i = j$ and $S_{ij} = 2$ if $i \neq j$.

$$\partial_i f(r) = \sum_{\omega \in \Omega} \frac{\omega_i}{\sqrt{p}} (-\beta_{\omega,\cos} \sin\Big(\omega^\top r\Big) + \beta_{\omega,\sin} \cos\Big(\omega^\top r\Big))$$

$$\partial_{ij} f(r+tu) = -\sum_{\omega \in \Omega} \frac{\omega_i \omega_j}{\sqrt{p}} (\beta_{\omega,\cos} \cos\Big(\omega^\top r\Big) + \beta_{\omega,\sin} \sin\Big(\omega^\top r\Big))$$

$$\forall a, b \in \mathbb{R} \int_0^1 (1-t)\cos(at+b)dt = -\frac{1}{a}\sin(b) - \frac{1}{a^2}\cos(a+b) + \frac{1}{a^2}\cos(b)$$

$$\text{and} \int_0^1 (1-t)\sin(at+b)dt = -\frac{1}{a}\cos(b) - \frac{1}{a^2}\sin(a+b) + \frac{1}{a^2}\sin(b)$$

$$R_{ij}(r+u) = -S_{ij} \sum_{\omega \in \Omega} \frac{\omega_i \omega_j}{\sqrt{p}} \left[\beta_{\omega,\cos} \int_0^1 (1-t)\cos\Big(\omega^\top (r+tu)\Big)dt \; + \; \beta_{\omega,\sin} \int_0^1 (1-t)\sin\Big(\omega^\top (r+tu)\Big)dt\right]$$

$$= -S \sum_{\omega \in \Omega} \frac{\omega_i \omega_j}{\sqrt{p}} \left[\beta_{\omega,\cos}(-\frac{1}{\omega^\top u}\sin\Big(\omega^\top r\Big) - \frac{1}{(\omega^\top u)^2}\cos\Big(\omega^\top u + \omega^\top r\Big) + \frac{1}{(\omega^\top u)^2}\cos\Big(\omega^\top r\Big)) + \right.$$

$$\left. \beta_{\omega,\sin}(-\frac{1}{\omega^\top u}\cos\Big(\omega^\top r\Big) - \frac{1}{(\omega^\top u)^2}\sin\Big(\omega^\top u + \omega^\top r\Big) + \frac{1}{(\omega^\top u)^2}\sin\Big(\omega^\top r\Big))\right]$$

$$= -S_{ij} \sum_{\omega \in \Omega} \frac{\omega_i \omega_j}{\sqrt{p}} \beta_{\omega,\cos}(\cos\Big(\omega^\top r\Big) + c_u) + \beta_{\omega,\sin}(\sin\Big(\omega^\top r\Big) + s_u)$$

$$= A(r)_{ij} + B(r)_{ij} c'_u$$

$$A(r)_{ij} = -S_{ij} \sum_{\omega \in \Omega} \frac{\omega_i \omega_j}{\sqrt{p}} \beta_{\omega,\cos} \cos\Big(\omega^\top r\Big) + \beta_{\omega,\sin} \sin\Big(\omega^\top r\Big)$$

$$B(r)_{ij} = -S_{ij} \sum_{\omega \in \Omega} \frac{\omega_i \omega_j}{\sqrt{p}} \beta_{\omega,\cos} + \beta_{\omega,\sin}$$

with $c_u, s_u, c'_u = \mathcal{O}(|\omega^\top u|)$, ie there exist a constant C such that $\forall u, c_u, s_u, c'_u \leq C|\omega^\top u|$. c_u, s_u, c'_u are obtained by applying Taylor formulas to sin and cos.

Then

$$|f(r+u) - f(r)| \leq |u^\top \nabla f(r)| + |u^\top R(r+u)u| \tag{A.159}$$

$$\leq ||u|| ||\nabla f(r)|| + ||R(r+u)||_F ||u||^2 \tag{A.160}$$

$$\leq \epsilon ||\nabla f(r)|| + \epsilon^2 ||R(r+u)||_F \tag{A.161}$$

$$|\omega^\top u| \leq dL\epsilon \tag{A.162}$$

since $||u|| \leq \epsilon$.

If we have for all $r \in T$, and for all $i, j \in [\![1, d]\!]$,

1. $|f(r)| \leq \frac{1}{3}$
2. $|\partial_i f(r)| \leq \frac{1}{3d\epsilon}$
3. $|A(r)_{ij}| \leq \frac{1}{6d\epsilon^2}$
4. $|B(r)_{ij}| \leq \frac{1}{6d^2 L\epsilon^3}$

then we will have $\forall x \in \mathcal{X}, |f(x)| \leq 1$.

Now let us lower bound the probability over drawing βs of each condition . Each time we adapt the equation A.155 derived from McDiarmid's inequality

1. $\mathbb{P}(|f(r)| \leq 1/3) \geq 1 - \exp\Big(-\frac{1}{18\sigma^2}\Big)$
2. $\mathbb{P}(|\partial_i f(r)| \leq \frac{1}{3d\epsilon}) \geq 1 - \exp\Big(-\frac{1}{9d^2\epsilon^2 2\sigma^2 L^2}\Big) = 1 - \exp\Big(-\frac{1}{18d^2\epsilon^2\sigma^2 L^2}\Big)$
3. $\mathbb{P}(|A(r)_{ij}| \leq \frac{1}{6d\epsilon^2}) \geq 1 - \exp\Big(-\frac{1}{72d^2\epsilon^4\sigma^2 L^4}\Big)$
4. $\mathbb{P}(|B(r)_{ij}| \leq \frac{1}{6d^2 L\epsilon^3}) \geq 1 - \exp\Big(-\frac{1}{72d^4\epsilon^6 2\sigma^2 L^6}\Big)$

By performing a union bound on all conditions, we have that

$$\mathbb{P}(|f| \leq 1) \geq \left(1 - \exp\left(-\frac{1}{18\sigma^2}\right)\right)^{|T|} \left(1 - \exp\left(-\frac{1}{18d^2\epsilon^2\sigma^2 L^2}\right)\right)^{d|T|} \left(1 - \exp\left(-\frac{1}{72d^2\epsilon^4\sigma^2 L^4}\right)\right)^{d^2|T|} \tag{A.163}$$

$$\left(1 - \exp\left(-\frac{1}{72d^4\epsilon^6\sigma^2 L^6}\right)\right)^{d^2|T|} \tag{A.164}$$

$$\geq 1 - |T|\exp\left(-\frac{1}{18\sigma^2}\right) - d|T|\exp\left(-\frac{1}{18d^2\epsilon^2\sigma^2 L^2}\right) - \tag{A.165}$$

$$d^2|T|\exp\left(-\frac{1}{72d^2\epsilon^4\sigma^2 L^4}\right) - d^2|T|\exp\left(-\frac{1}{72d^4\epsilon^6\sigma^2 L^6}\right) \tag{A.166}$$

If $\epsilon \leq \frac{1}{2^{1/2}dL}$, then $\exp\Big(-\frac{1}{18d^2\epsilon^2\sigma^2 L^2}\Big), \exp\Big(-\frac{1}{72d^2\epsilon^4\sigma^2 L^4}\Big)$, and $\exp\Big(-\frac{1}{72d^4\epsilon^6\sigma^2 L^6}\Big) \leq \exp\Big(-\frac{1}{18\sigma^2}\Big)$ Therefore

$$\mathbb{P}(|f| \leq 1) \geq 1 - (1 + d + 2d^2)\Big(Ld2^{3/2}\pi\Big)^d \exp\left(-\frac{1}{18\sigma^2}\right) \tag{A.167}$$

To satisfy these inequalities, it is enough to have σ on the order of $\Theta(1/(d\,(\log d + \log L)))$ □

A.7 Further examples of graph kernels

A.7.1 SVM-ϑ kernel

The SVM-ϑ kernel was proposed as an alternative to the more computationally intensive Lovasz-ϑ kernel. Both ϑ kernels leverage the so-called orthogonal representation of a graph. Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the orthogonal representation is an assignment of unit vectors $\{\mathbf{u}_i\}$ to each node of the graph, subject to the constraint that unit vectors associated to vertices that are not joined by an edge are orthogonal: $\langle \mathbf{u}_i, \mathbf{u}_j \rangle = 0$ if $\{i, j\} \notin \mathcal{E}$.

Orthogonal representations are not unique, but there is a particular representation associated with the ϑ number (Lov79) of a graph. Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with n vertices, denote $U_{\mathcal{G}}$ an

orthogonal representation of $\mathcal{G}$, and C the space of unit vectors in $\mathbb{R}^n$. The ϑ number is defined as:

$$\vartheta(\mathcal{G}) := \min_{\mathbf{c}\in C} \min_{U_{\mathcal{G}}} \max_{\mathbf{u}_i \in U_{\mathcal{G}}} \frac{1}{\langle \mathbf{c}, \mathbf{u}_i \rangle^2}. \tag{A.168}$$

From now on, we will always be referring to the particular orthogonal representation $U_{\mathcal{G}}$ that minimizes (A.168).

Now consider a subset of vertices $B \subset \mathcal{V}$, and call $U_{\mathcal{G}|B}$ the orthogonal representation obtained from $U_{\mathcal{G}}$ by removing the vectors that are not in B:

$$U_{\mathcal{G}|B} := \{\mathbf{u}_i \in U_{\mathcal{G}} : \ i \in B\}. \tag{A.169}$$

Note that $U_{\mathcal{G}|B}$ preserves the global properties encoded in $U_{\mathcal{G}}$ through the orthogonal constraint, and that $U_{\mathcal{G}|B}$ is not in general the orthogonal representation of the subgraph of $\mathcal{G}$ containing only the vertices in B. Define the ϑ_B number:

$$\vartheta_B(\mathcal{G}) := \min_{\mathbf{c}\in C} \max_{\mathbf{u}_i \in U_{\mathcal{G}|B}} \frac{1}{\langle \mathbf{c}, \mathbf{u}_i \rangle^2}. \tag{A.170}$$

We are ready now to give the definition of the Lovasz-ϑ kernel. Given two graphs $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1)$, $\mathcal{G}_2 = (\mathcal{V}_2, \mathcal{E}_2)$, define:

$$K_{\text{Lo}}(\mathcal{G}_1, \mathcal{G}_2) := \sum_{B_1 \subset V_1} \sum_{B_2 \subset V_2} \delta_{|B_1|,|B_2|} \ \frac{1}{Z} \ k\left(\vartheta_{B_1}, \vartheta_{B_2}\right) \tag{A.171}$$

where $Z = \binom{|V_1|}{|B_1|}\binom{|V_2|}{|B_2|}$, δ is the Kronecker delta, and k is a freely specifiable kernel (called base kernel) from $\mathbb{R} \times \mathbb{R}$ to $\mathbb{R}$.

The SVM-ϑ kernel is defined as (A.171), but it uses an approximation for the ϑ numbers. Consider a graph $\mathcal{G}$ with n vertices and adjacency matrix A, and let $\rho \geq -\lambda$, where λ is the minimum eigenvalue of A. The matrix

$$\kappa := \frac{1}{\rho} A + I \tag{A.172}$$

is positive semi-definite. Define the maximization problem:

$$\max_{\alpha_i \geq 0} \ 2\sum_{i=1}^{n} \alpha_i - \sum_{i,j=1}^{n} \alpha_i \alpha_j \kappa_{ij}. \tag{A.173}$$

If $\{\alpha_i^*\}$ are the maximizers of (A.173), then it can be proven that on certain families of graphs the quantity $\sum_i \alpha_i^*$ is with high probability a constant factor approximation to $\vartheta(\mathcal{G})$:

$$\vartheta(\mathcal{G}) \leq \sum_{i=1}^{n} \alpha_i^* \leq \gamma \vartheta(\mathcal{G}) \tag{A.174}$$

for some γ. The SVM-ϑ kernel then replaces the ϑ_B numbers on subgraphs with:

$$\vartheta_B(\mathcal{G}) \to \sum_{j \in B} \alpha_j^*. \tag{A.175}$$

The SVM-ϑ kernel requires a choice of base kernel $k : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$. We choose a translation invariant universal kernel (MXZ06) $k(x, y) = (\beta + ||x - y||^2)^{-\alpha}$, where α and β are two trainable hyperparameters.

A.7.2 Shortest Path kernel

Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, an edge path between two vertices $u, v \in \mathcal{V}$ is a sequence of edges $(e_1, \ldots, e_n)$ such that $u \in e_1$, $v \in e_n$, e_i and e_{i+1} are contiguous (*i.e.* they have one of the endpoints in common) and $e_i \neq e_j$ for $i \neq j$. Computing the shortest edge path between any two nodes of a graph can be done in polynomial time with the Dijkstra (Dij59) or Floyd-Warshall (Flo62) algorithms, which makes it a viable feature to be probed by a graph kernel.

The first step of the Shortest Path kernel is to transform the graphs into shortest path graphs. Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the shortest path graph $\mathcal{G}^S = (\mathcal{V}^S, \mathcal{E}^S)$ associated to $\mathcal{G}$ is defined as:

$$\mathcal{V}^S = \mathcal{V} \tag{A.176}$$

$$\mathcal{E}^S = \{(u, v) \mid \exists \text{ an edge path } (e_1, \ldots, e_n) \text{ between } u \text{ and } v \text{ in } \mathcal{G}\} \tag{A.177}$$

In addition, to each edge $e \in \mathcal{E}^S$ a label $l(e)$ is assigned given by the length of the shortest path in $\mathcal{G}$ between its endpoints. The Shortest Path kernel is then defined as:

$$K_{\mathrm{SP}}(\mathcal{G}_1, \mathcal{G}_2) := \sum_{e \in \mathcal{E}_1^S} \sum_{p \in \mathcal{E}_2^S} k(e, p) \tag{A.178}$$

with k being a kernel between edge paths such as the Brownian bridge kernel:

$$k(e, p) := \max\{0,\ c - |l(e) - l(p)|\} \tag{A.179}$$

for a choice of c.

A.8 Algorithms for quantum positional encodings

We detail the algorithms corresponding to the use of positional encodings obtained with a quantum computer.
Firstly, we give a generic algorithm that allows the simulation of quantum features on a classical computer, according to an arbitrary choice of Hamiltonian and quantum observables.

Algorithm 4 Positional encoding with a generic classical simulation of a Hamiltonian evolution

Input : a graph $\mathcal{G}(\mathcal{V},\mathcal{E})$, a hyper-parameter set $\{\boldsymbol{\theta}_1,\boldsymbol{\theta}_2,...,\boldsymbol{\theta}_K\}$[1] and a set of quantum observables $\{\hat{\mathcal{O}}_i\}_{i\in\mathcal{V}}$ [2]

Output : the matrices of node pairs positional encoding $\mathbf{P_E}\in\mathbb{R}^{d_e\times K}$, and single nodes positional encoding $\mathbf{P_N}\in\mathbb{R}^{N\times K}$ where $d_e=\binom{N}{2}$, and N the number of nodes in the graph.

for $k\in\{1,2,...,K\}$ **do**
  Compute $|\psi_{\mathcal{G}}(\boldsymbol{\theta}_k)\rangle$ according to equation (5.12)
  $(\mathbf{P_E})_{ij,k}=\langle\psi_{\mathcal{G}}(\boldsymbol{\theta}_k)|\hat{\mathcal{O}}_i\hat{\mathcal{O}}_j|\psi_{\mathcal{G}}(\boldsymbol{\theta}_k)\rangle-\langle\psi_{\mathcal{G}}(\boldsymbol{\theta}_k)|\hat{\mathcal{O}}_i|\psi_{\mathcal{G}}(\boldsymbol{\theta}_k)\rangle\langle\psi_{\mathcal{G}}(\boldsymbol{\theta}_k)|\hat{\mathcal{O}}_j|\psi_{\mathcal{G}}(\boldsymbol{\theta}_k)\rangle$
  $(\mathbf{P_N})_{i,k}=\langle\psi_{\mathcal{G}}(\boldsymbol{\theta}_k)|\hat{\mathcal{O}}_i^2|\psi_{\mathcal{G}}(\boldsymbol{\theta}_k)\rangle-\langle\psi_{\mathcal{G}}(\boldsymbol{\theta}_k)|\hat{\mathcal{O}}_i|\psi_{\mathcal{G}}(\boldsymbol{\theta}_k)\rangle^2$
end forreturn $\mathbf{P_E}$ and $\mathbf{P_N}$.

Next we provide an efficient classical algorithm for the simulation of the 2 particles quantum walk (which also applies to the 1 particle random walk, if one applies the following on the original adjacency matrix of the graph).

Algorithm 5 Positional encoding with a classically simulated 2 particles QW

Input : a graph $\mathcal{G}(\mathcal{V},\mathcal{E})$, a hyper-parameter set $\{t_1,t_2,...,t_K\}$

Output : the matrices of node pairs positional encoding $\mathbf{P_E}\in\mathbb{R}^{d_e\times K}$, and single nodes positional encoding $\mathbf{P_N}\in\mathbb{R}^{N\times K}$ where $d_e=\binom{N}{2}$, and N the number of nodes in the graph.

- Compute the adjacency matrix $\boldsymbol{A}_2$ of the (2 particles) product graph : we draw an edge in the latter if the symmetric difference $\{i_1,j_1\}\Delta\{i_2,j_2\}$ is a pair of adjacent nodes in the input graph[3].
- Define $\hat{\mathcal{H}}_1^{XY}$ as in equation (5.7) on the edges of the product graph, indexed as (i_1j_1,i_2j_2)
for $k\in\{1,2,...,K\}$ **do**
  if Continuous random walk scheme **then**
    $(\mathbf{P_E})_{ij,k}=|\langle ij|e^{-i.t_k.\hat{\mathcal{H}}_1^{XY}}|\psi_0\rangle|^2\ \ \forall i\in\{1,..,N\}$ and $\forall j\in\{i+1,...,N\}$ [4]
    $(\mathbf{P_N})_{i,k}=|\langle ii|e^{-i.t_k.\hat{\mathcal{H}}_1^{XY}}|\psi_0\rangle|^2\ \ \forall i\in\{1,..,N\}$
  else if Quantum-inspired random walk scheme **then**
    $(\mathbf{P_E})_{ij,k}=[(\boldsymbol{D}_2^{-1}.\boldsymbol{A}_2)^k.\psi_0]_{ij}\ \ \forall i\in\{1,..,N\}$ and $\forall j\in\{i+1,...,N\}$
    $(\mathbf{P_N})_{i,k}=[(\boldsymbol{D}_2^{-1}.\boldsymbol{A}_2)^k.\psi_0]_{ii}\ \ \forall i\in\{1,..,N\}$[5]
  end if
end forreturn $\mathbf{P_E}$ and $\mathbf{P_N}$.

Finally, we present an algorithm for computing our quantum features using a quantum computer.

Algorithm 6 Positional encoding with a generic Hamiltonian evolution on a quantum computer

Input : a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, a hyper-parameter set $\{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, ..., \boldsymbol{\theta}_K\}$, the number of measurements N_m for the estimation of the average of quantum observable $\hat{\mathcal{O}}_i$

Output : the matrices of node pairs positional encoding $\mathbf{P_E} \in \mathbb{R}^{d_e \times K}$, and single nodes positional encoding $\mathbf{P_N} \in \mathbb{R}^{N \times K}$ where $d_e = \binom{N}{2}$, and N the number of nodes in the graph.

for $k \in \{1, 2, ..., K\}$ **do**
  for $m \in \{1, 2, ..., N_m\}$ **do**
    Let the system evolve according to equation (5.12) to prepare the quantum state $|\psi_{\mathcal{G}}(\boldsymbol{\theta}_k)\rangle$
    Perform a measurement of the observables[6] $\{\hat{\mathcal{O}}_i\}_{i \in \mathcal{V}}$
  end for

$(\mathbf{P_E})_{ij,k} = \langle\psi_{\mathcal{G}}(\boldsymbol{\theta}_k)|\hat{\mathcal{O}}_i\hat{\mathcal{O}}_j|\psi_{\mathcal{G}}(\boldsymbol{\theta}_k)\rangle - \langle\psi_{\mathcal{G}}(\boldsymbol{\theta}_k)|\hat{\mathcal{O}}_i|\psi_{\mathcal{G}}(\boldsymbol{\theta}_k)\rangle \langle\psi_{\mathcal{G}}(\boldsymbol{\theta}_k)|\hat{\mathcal{O}}_j|\psi_{\mathcal{G}}(\boldsymbol{\theta}_k)\rangle$ estimated from the measurements

$(\mathbf{P_N})_{i,k} = \langle\psi_{\mathcal{G}}(\boldsymbol{\theta}_k)|\hat{\mathcal{O}}_i^2|\psi_{\mathcal{G}}(\boldsymbol{\theta}_k)\rangle - \langle\psi_{\mathcal{G}}(\boldsymbol{\theta}_k)|\hat{\mathcal{O}}_i|\psi_{\mathcal{G}}(\boldsymbol{\theta}_k)\rangle^2$ estimated from the measurments

end forreturn $\mathbf{P_E}$ and $\mathbf{P_N}$.

A.9 Datasets

A.9.1 Description about the benchmark datasets

QM7 and QM9 molecules and graph regression

Context

QM7 dataset is a subset of the GDB-13 database (BR09), a database of nearly 1 billion stable and synthetically accessible organic molecules, containing up to seven heavy atoms (C, N, O, S). Similarly QM9 is a subset of the GDB-17 database consisting of molecules with up to nine heavy atoms. Learning methods using QM7 and QM9 are predicting the molecules electronic properties given stable conformational coordinates.

QM7 figures

QM7 consists of 7165 molecule graphs. Each node is an atom with its 3D coordinates and atomic number Z. The only edge feature is the entry of the Coulomb matrix. Each graph is thus fully connected and has one regression target corresponding to its atomization energy.

QM9 figures

QM9 consists of 130831 molecule graphs of between 1 and 29 nodes with an average of 18 nodes (see Figure A.1). Each node is an atom with its 3D coordinates and its atomic number Z. Edges are purely distance based and have no feature. Each graph is thus fully connected and has 12 regression targets corresponding to diverse chemical electronic properties. In our implementation, all the targets are recentered and rescaled by their standard deviation.

Benchmarks

On the QM7 dataset, Quantum Machine benchmark reached MAE of 3.5 and 9.9 (MRG+13).

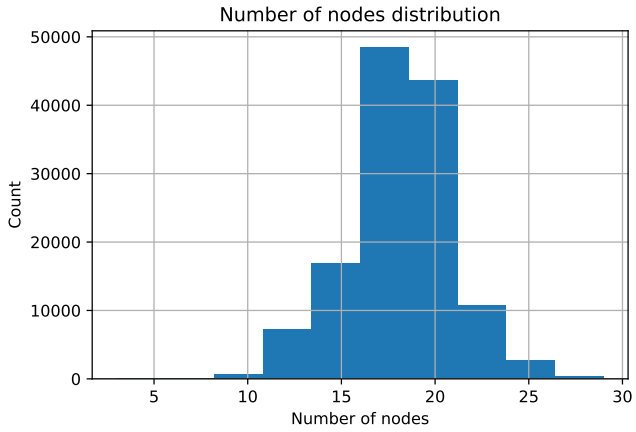


Figure A.1: Distribution of the number of nodes in a graph for QM9 dataset.

Conf. category	Conferences	#Papers	#Graphs
DBDM	SIGMOD, VLDB, ICDE, EDBT, PODS, DASFAA, SSDBM, CIKM, DEXA, KDD, ICDM, SDM, PKDD, PAKDD	20601	9530
CVPR	ICCV, CVPR, ECCV, ICPR, ICIP, ACM Multimedia, ICME	18366	9926

Table A.5: DBLP_v1 details.

The best models of the MoleculeNet benchmark (WRF+17) reached a test MAE of 2.86 ± 0.25 on QM7 and 2.4 ± 1.1 on QM9.

DBLP_v1 and node classification

Context

DBLP_v1 is a graph stream built out of the DBLP dataset (PZZY13) consisting of bibliography data in computer science. To build a graph stream, a list of conferences from DBDM (database and data mining) and CVPR (computer vision and pattern recognition) fields are selected (as shown in Table A.5). The papers published in these conferences are then used (in chronological order) to form a binary-class graph stream where the classification task is to predict whether a paper belongs to DBDM or CVPR field by using the references and the title of each paper.

Papers without references are filtered out. Then, the top 1000 most frequent words (excluding stop words) in titles are used as keywords to construct the graph (see Figure A.2).

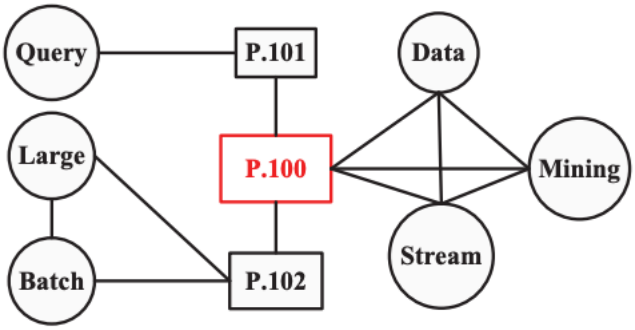


Figure A.2: Graph representation for a paper (P.100) in the DBLP_v1 dataset. The rectangles are paper ID nodes and circles are keyword nodes from titles. The paper P.100 cites (connects) paper P.101 and P.102, and P.100 has keywords Data, Stream, and Mining in its title. Paper P.101 has keyword Query in its title, and P.102's title include keywords Large and Batch. For each paper, the keywords in the title are linked with each other

Figures

DBLP_v1 consists of 19456 graphs evenly split between the two groups of conferences (the two classes) from 2 to 39 nodes with an average of 10 nodes.

These graphs are actually local parts of a bigger graph. To perform node classification (on nodes representing a paper), the local neighborhood of each graph is extracted and a graph classification task is run.

There are 3 types of edges:

- 0: paper - paper
- 1: keyword - paper
- 2: keyword - keyword

Node features are only a unique ID to be identified between multiple graphs (cf P.100 in Figure A.2 which will also appear in P.101 graph). There are 41325 unique IDs so keywords don't have a single keyword identifier among all graphs.

Benchmarks

DBLP_v1 (PZZY13) benchmarks show accuracy ranging between 0.55 and 0.80 in a chunked graph stream classification setup.

Data augmentation

The map from IDs to topics and paper IDs is also provided and has been used to perform data augmentation to provide node features. USing Stanford GloVe word embedding pre-trained on Wikipedia 2014 and Gigaword 5 in a 50-dimension space, each topic node could be enriched with its embedding. A boolean flag was also added to identify a node as a topic or not (a paper).

Computer vision: letters and graph classification

Context

Letters datasets (RB08) are 3 datasets of distorted letter drawings with low, medium or high distorsion levels. Only the 15 capital letters of the Roman alphabet that consist of straight lines (A, E, F, H, I, K, L, M, N, T, V, W, X, Y, Z) are represented. Distorted letter drawings are converted into graphs by representing lines by undirected edges and ending points of lines by nodes.

Figures

Node attributes are their 2D positions and edges have no attribute. The graphs are uniformly distributed over the 15 letters. We focused on the medium distorsion dataset consisting of 2250 graphs.

Benchmarks

Benchmark results from k-NN are given by (RB08): 99.6% (low), 94.0% (medium), and 90.0% (high). The best classical algorithm we trained on this dataset was GraphSAGE with results of 100% (low), 94.5% (medium), and 80% (high)).

Datasets used in experiments on quantum random walks

The datasets used for benchmarking the use of quantum random walks encodings are standard in the GNN community. The first five are from (DJL+20), the last one is from (HFR+21). We reproduce the table of statistics A.6 taken from (MLL+23), and we also refer the reader to (RGD+22) for more information about the datasets.

Table A.6: Overview of the graph learning datasets involved in this work (DJL+20), (ISM+12), (HFR+21) .

Dataset	# Graphs	Avg. # nodes	Avg. # edges	Directed	Prediction level	Prediction task	Metric
ZINC(-full)	12,000 (250,000)	23.2	24.9	No	graph	regression	Mean Abs. Error
MNIST	70,000	70.6	564.5	Yes	graph	10-class classif.	Accuracy
CIFAR10	60,000	117.6	941.1	Yes	graph	10-class classif.	Accuracy
PATTERN	14,000	118.9	3,039.3	No	inductive node	binary classif.	Weighted Accuracy
CLUSTER	12,000	117.2	2,150.9	No	inductive node	6-class classif.	Accuracy
PCQM4Mv2	3,746,620	14.1	14.6	No	graph	regression	Mean Abs. Error

A.9.2 Construction of artificial datasets

In this subsection, we explain how to construct our artificial dataset. Our building blocks are 3 types of graphs, called types 0, 1, 2. Each type is composed of one ladder graphs with crossings inserted at different places. All crossings are in the same fixed arbitrary direction. Type 0 graphs are plain ladder graphs and their Ising hamiltonian has two ground states. Type 1 graphs are type 0 graphs with crossings separated with an odd number of nodes. The crossings are located such that they have one possible Ising ground state which is one of the ground states of the

type 0 associated graph. The crossings will effectively select one of the two possible ground states. Type 2 graphs are ladder graphs of odd length with crossings at the beginning and the end. An illustration of the types of graphs is provided figure A.3.

We construct a graph given two graphs of same length but different types concatenated to each other. The first class is determined by graphs of type 0 and type 1 concatenated, and the second class is composed of graphs of type 0 concatenated to graphs of type 1. The concatenation is made by adding edges to continue the ladder, the process is illustrated figure A.4. The ground state of the total graph is included in a union of the groundstates of the subgraph, so it can be efficiently computed. The length of graphs are taken between 100 and 400, our dataset consists of 400 graphs per class, so 800 in total.

In figure A.4 we see that the RRWP features are very similar for the two classes whereas the correlations on the ground state are very different.

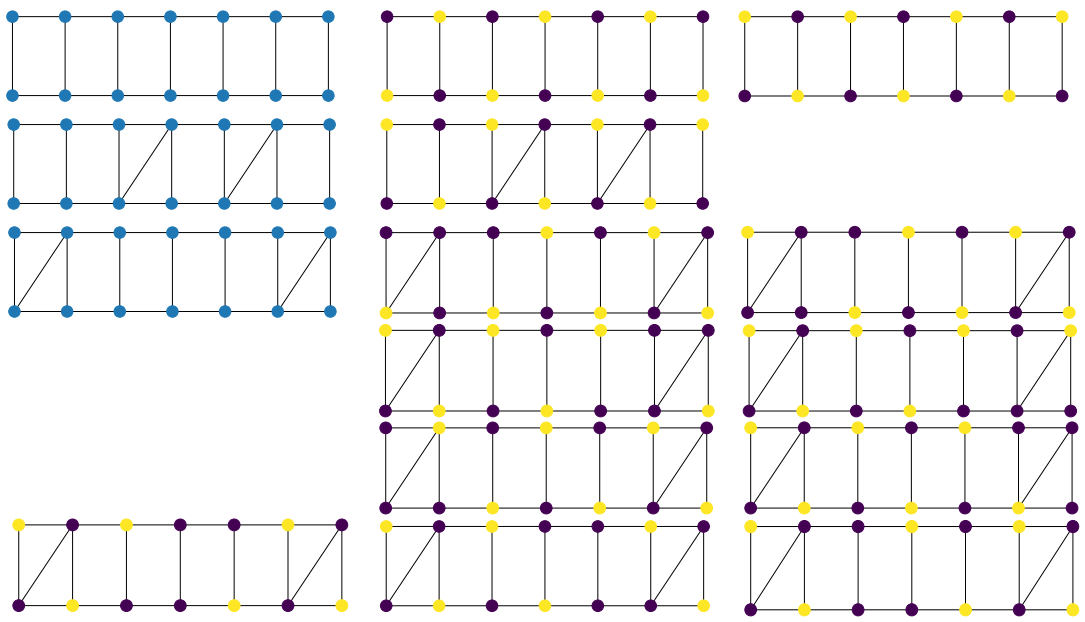


Figure A.3: The base subgraphs (type 0, type 1, type 2) and their possible ground state. Top : type 0 graph of length 7, 2 possible ground states. Middle: type 1 graph of length 7, 1 possible ground state. Bottom : type 2 graph of length 7, 9 possible ground states.

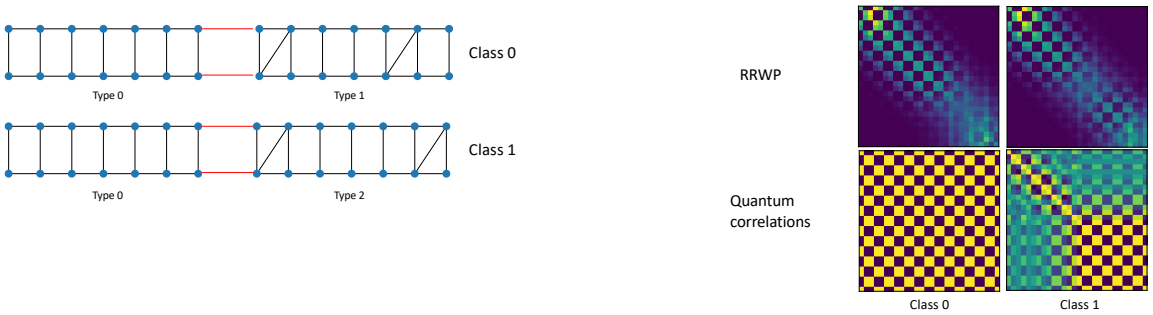


Figure A.4: Left: construction of our artificial dataset. Right: RRWP and quantum features for each class of the dataset on a 40 nodes graph.

A.10 Hyperparameters

Table A.7: Hyperparameters for GRIT model five datasets from BenchmarkingGNNs (DJL+20), ZINC-full (ISM+12) and (HFR+21)

Hyperparameter	ZINC/ZINC-full	MNIST	CIFAR10	PATTERN	CLUSTER	PCQM4Mv2
# Transformer Layers	10	3	3	10	16	16
Hidden dim	64	52	52	64	48	256
# Heads	8	4	4	8	8	8
Dropout	0	0	0	0	0.01	0.1
Attention dropout	0.2	0.5	0.5	0.2	0.5	0.1
Graph pooling	sum	mean	mean	–	–	mean
PE dim (RW-steps)	21	18	18	21	32	16
PE encoder	linear	linear	linear	linear	linear	linear
QPE dim (1CQRW steps)	20	18	18	20	32	16
Max duration	π	π	π	π	π	π
Min duration	0.1	0.1	0.1	0.1	0.1	0.1
Initial distribution	local	local	local	local	local	local
QPE dim (2QiRW steps)	20	18	18	20	32	16
Initial distribution	adjacency	adjacency	adjacency	adjacency	adjacency	adjacency
Batch size	32/256	16	16	32	16	256
Learning Rate	0.001	0.001	0.001	0.0005	0.0005	0.0002
# Epochs	2000	200	200	100	100	150
# Warmup epochs	50	5	5	5	5	10
Weight decay	$1e-5$	$1e-5$	$1e-5$	$1e-5$	$1e-5$	0
# Parameters GRIT	473,473	102,138	99486	477,953	432,206	11.8M
# Parameters 2QiRW GRIT	476,033	104,010	101,358	480,513	434,742	11.8M

Table A.8: Hyperparameters for non transformer base models large scale datasets , ZINC-full (ISM+12) and PCQM4Mv2 (HFR+21). Each entry has to be read as the values for ZINC-full/PCQM4Mv2, when there is a single entry, the value is the same for both datasets. * : same as the first column. - : non applicable.

Hyperparameter	GINE	GINE-big	GatedGCN	GatedGCN-big
# Layers	5/3	*	*	*
Hidden dim	128	256	128	256
Dropout	0	*	*	*
Aggregation	mean	*	*	*
# Layers MLP postprocessing	3	*	*	*
PE encoder	linear	*	*	*
PE dim (RRWP)	21	40	21	40
PE dim (LE)	32	*	*	*
PE dim (Q)	21	40	20	40
Initial distribution	adjacency	*	*	*
PE dim RRWP(RRWP+Q)	20	*	*	*
PE dim Q(RRWP+Q)	20	*	*	*
Batch size	256	*	*	*
Learning Rate	0.001/0.0002	*	*	*
# Epochs	2000/150	*	*	*
# Warmup epochs	50/10	*	*	*
Weight decay	$1e-5/0$	*	*	*

A.11 Results on large scale datasets

Table A.9: Test performance on ZINC-full and PCQM4MV2.

Method	**Model**	**ZINC-full** (MAE ↓)	**PCQM4MV2** (MAE ↓)
GatedGCN	LE	$.033 \pm .001$	.1056
	RRWP	$.026 \pm .003$	.1045
	Q	$.031 \pm .002$	.1079
	RRWP+Q	$.026 \pm .001$	.1052
GatedGCN-big	LE	$.033 \pm .0008$	.1016
	RRWP	$.025 \pm .0017$	.1005
	Q	$.025 \pm .0023$	.1035
	RRWP+Q	$.022 \pm .0017$	.0999
GINE	LE	$.035 \pm .002$	.1155
	RRWP	$.029 \pm .003$	.114
	Q	$.027 \pm .0005$	.1149
	RRWP+Q	$.029 \pm .003$	.1124
GINE-big	LE	$.036 \pm .0022$	.1063
	RRWP	$.029 \pm .003$	.1041
	Q	$.024 \pm .002$	.1054
	RRWP+Q	$.027 \pm .0025$	.1048
GRIT	RRWP	0.025 ± 0.002	.0842
	RRWP+Q	0.023 ± 0.002	.0838

A.12 Hardware implementations

A.12.1 Mapping and Batching

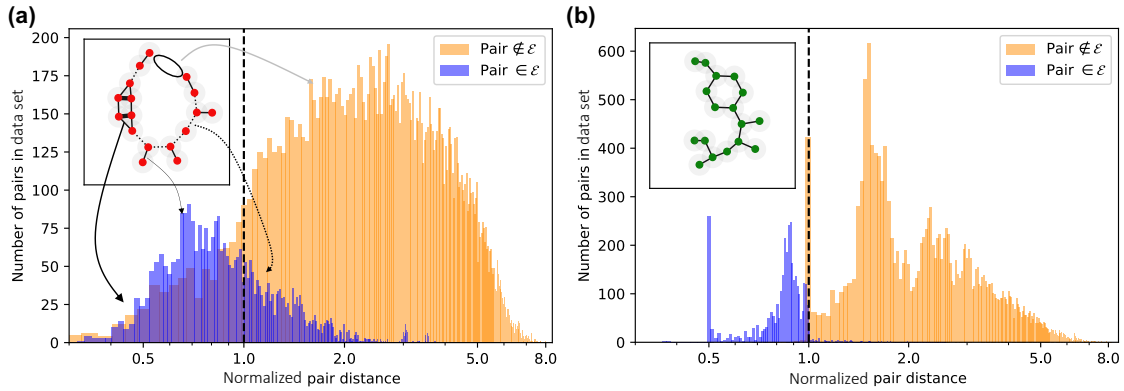


Figure A.5: Histograms of normalised pairwise distances between atoms in the 286 graphs of the truncated dataset when performing the embedding with **a.** only a Fruchterman-Reingold layout or **b.** when adding a local optimization step afterwards. For a given graph (insets), two atoms forming a pair $\in \mathcal{E}$ (blue) can be close enough to form a bond via interaction (plain) or too far, creating a missing bond (dotted). Likewise, two atoms forming a pair $\notin \mathcal{E}$ can be placed too close and form a fake edge (thick line).

We present in detail our method to embed the graphs of the PTC-FM dataset. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph of the dataset for which we have a layout of the nodes. Embedding the graph amounts to replace its nodes with atoms, the latter interacting between themselves with the $1/R^6$ dependence. Moving two atoms slightly apart can therefore drastically reduce their interaction strength but it remains non-zero. In order for the Hamiltonian to reflect the topology of $\mathcal{G}$, this $1/R^6$ dependence needs to be approximated by the Heaviside function defined as:

$$h(r) = \begin{cases} \infty & \text{if } r \leq r_b \\ 0 & \text{else} \end{cases} \tag{A.180}$$

For the Heaviside approximation to be correct, we have to ensure that the largest distance between a pair sharing an edge in the graph is always far less than the shortest distance between a pair not sharing an edge. In other words, in theory, $\min\{U_{ij}, (i,j) \in \mathcal{E}\} / \max\{U_{ij}, (i,j) \notin \mathcal{E}\} \gg 1$.

We use a local optimizer to maximize this ratio and find good solutions in polynomial time. The method optimizes the position of each node in turn, depending on the previously mapped nodes and the presence of cycles in the graph. For the dataset used in this study, we achieve a significant increase of the mean ratio up to 16.8, starting from 5.9 with the classical

Fruchterman-Reingold layout. We report that more than half the dataset exhibits a ratio higher than 10 and less than 5% of the dataset is embedded with some defects, i.e. a ratio smaller than 1. We also assess the benefit of this approach in Fig. A.5 by comparing the distributions of distance of pairs $\in \mathcal{E}$ and pairs $\notin \mathcal{E}$ (**a**) before and (**b**) after the optimization. While some defects, such as fake or missing bonds, frequently appear in the pre-optimisation embedding, the optimised positions are constrained such that a clear cut is visible between the two distributions, easing the approximation.

In order to further characterize the effect of these defects, we analyzed their effect on the measurement histograms. For each graph, we first compute the histogram that would have been obtained with a perfect embedding. We then compute the Jensen-Shannon divergence between this histogram and the one measured in the QPU. From these, we can also estimate that if one were to perform the SVM using histograms resulting from ideal embedding, one would expect a slightly worst F1-score of 58.8 ± 4.0.

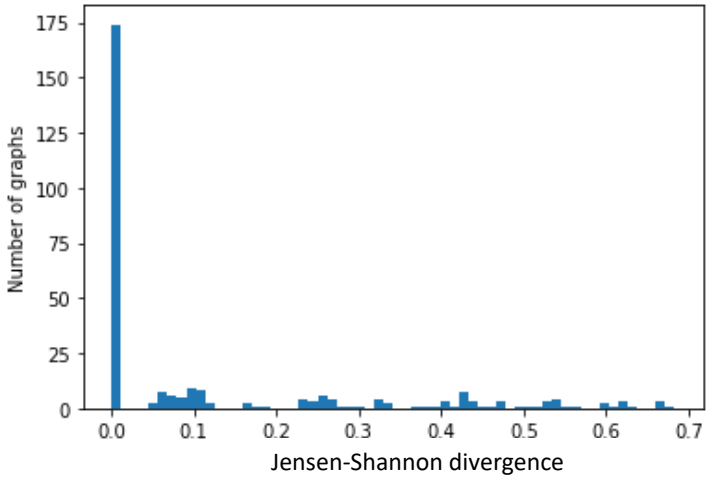


Figure A.6: Distribution of Jensen-Shannon divergences between measured and expected histograms for a pulse of duration $T = 0.66\mu s$. In the inset are shown the emulated traces of each of these divergences as a function of the duration of the pulse. The implementation was done in a regime where this difference can be large for some graphs.

In principle, we can program a different SLM pattern for the layout of each graph from the dataset. In practice however the SLM calibration step can be quite time-consuming, *i.e.* of the order of the minute. We can compare it to the duration of hundreds of shot, each of which consisting in applying a sequence and measuring a quantum state, performed at a frequency of 1 Hz. Then for each graph, calibrating the SLM and obtaining the probability distribution take approximately the same order of time.

We therefore seek to regroup many graphs onto the same SLM pattern, to be able to reduce the number of calibrations needed for the whole dataset. We do so by clustering the graphs according to similarities in their structures. Because the dataset consists in representations of organic molecules, many of the graphs share common structures. We thus focus on retrieving the presence and multiplicity of pentagons and hexagons. We then build a similarity measure between the graphs. For the pentagons for example, the similarity can be written under the form:

$$s(\mathcal{G}_1, \mathcal{G}_2) = 1 - \exp\Big(-\alpha |N_1^P - N_2^P|\Big) \tag{A.181}$$

where N^P represents the number of pentagons in $\mathcal{G}$ and α is a hyper-parameter. We then use a linear combination of similarity measures in order to build a similarity matrix between all graphs of the dataset. We then apply a k-means clustering algorithm (C.04) using the similarity matrix in order to separate the graphs into different batches. Furthermore, since the laser power is distributed over all the traps, we want to reduce the total number of traps, in order to maximize the intensity provided to each trap. This ensures that the traps are deep enough to obtain a satisfying filling efficiency ($\sim 55\%$) over the whole pattern. For each batch, we thus apply the following mapping algorithm

Algorithm 7 Creating a triangular SLM pattern by batching M graphs

Require: Graphs $\{\mathcal{G}_1, \ldots, \mathcal{G}_M\}$ in sorted sizes and optimized positions $\{x_1, \ldots, x_M\}$
Ensure: Single SLM pattern that embeds M graphs with optimal positions on a triangular lattice.

1: $traps = \{\}$
2: for i in range $1, \cdots, M$:
3: find $r_{\mathcal{G}_i} = \{r_1, \ldots, r_{|\mathcal{G}_i|}\}$ triangular grid points that best conserve the pairwise distances between points in x_i and maximizes overlap with existing $traps$.
4: $traps \leftarrow traps + r_{\mathcal{G}_i} \setminus traps$
5: if $|traps| < 2|\mathcal{G}_M|$, add additional random triangular grid points to guarantee the filling property for re-arrangement.

We successfully map the entire dataset of 286 graphs into only 6 SLM patterns. For example, we batch 66 graphs together onto the 71-trap SLM pattern presented in Fig. A.7. On average, the 6 SLM patterns use 70 traps each to encode 48 graphs each.

A.12.2 Noise model

Despite the precise calibration of the control devices which enable to monitor quantities such as the SLM pattern spacing or the pulse shapes, several experimental imperfections may alter the data measured on the experiment. All experimental data obtained during this study, including those presented in Figure 6.5 and 6.6, are uncorrected and thus needs to be benchmarked with respect to their simulated counterpart, taking into account the following main sources of noise.

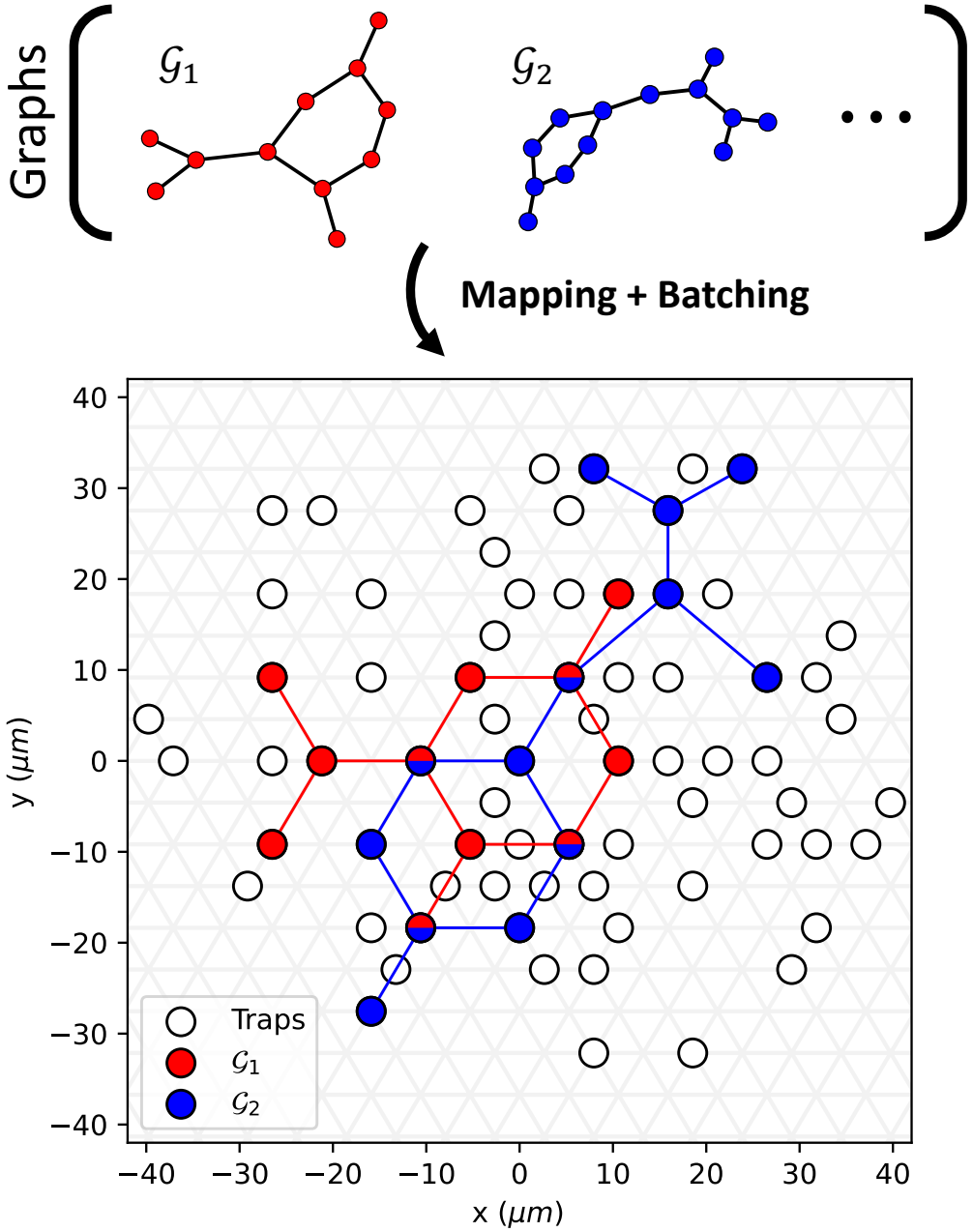


Figure A.7: A family of 66 graphs, ranging in sizes from 4 to 19 nodes, is mapped and batched to the same SLM pattern (white dots) over a triangular grid with spacing 5.6 μm. The traps used when implementing $\mathcal{G}_1$ ($\mathcal{G}_2$) are colored in red (blue). The bi-colored traps are those used for both graphs.

First and foremost, due to the nature of the quantum state and the limited budget of shots, measurements are subject to sampling noise. For instance, on average, each of the 25 experimental points on Figure 6.5 is obtained using 600 shots and the uncertainty related to

this effect (vertical error bars) can be estimated using the Jackknife resampling method (ST95).

The finite sampling is also inherently flawed by several physical processes like atoms thermal motion, background-gas collisions or Rydberg state finite lifetime, whose effects can all be encompassed as first approximation into two detection error terms, ε and ε'. ε (*resp* ε') yield the probability to get false positive (*resp* negative), i.e. measure an atom in $|0\rangle$ (*resp* $|1\rangle$) as being in $|0\rangle$ (*resp* $|1\rangle$). ε can be measured with a regular release-and-recapture experiment and ε' with a more advanced method (Ldk18) involving π and pushout pulses. To replicate the probabilistic effect of detection errors, the simulated distributions of bitstrings are altered using the following rule to compute the probability of measuring j instead of i:

$$P_{j|i} = \prod_k (1 - |i-j|_k) - (-1)^{|i-j|_k} \left[(1-i_k)\varepsilon + i_k\varepsilon'\right]. \tag{A.182}$$

$i, j \in \mathbb{B}^N$, $i_k = 0$ (*resp* 1) if atom k is in $|0\rangle$ (*resp* $|1\rangle$). On our device, we measure $\varepsilon \approx 3\%$ and $\varepsilon' \approx 8\%$; thus as an example, we can compute $P_{1001|0101} = \varepsilon\varepsilon'(1-\varepsilon)(1-\varepsilon') \approx 0.2\%$. Those detection errors can deeply modify the measured excitation distributions, with a noticeable effect shown on Figure 6.5**b** at $t = 0$ where the simulated $\langle n_j \rangle$ does not start at 0 despite $|\psi(t=0)\rangle = |0\ldots0\rangle$.

Additional errors can also lead to decoherence in the system (Ldk18), affecting the atom dynamics in ways costly to emulate. For instance, since the Rydberg transition used is addressed by a two-photon process, misalignments and power fluctuations of the two lasers are twice as likely to occur. Atoms are subject to positional disorder between each shot and their finite velocities make them sensitive to the Doppler effect. Since taking all those effects into consideration becomes quickly intractable, they were only individually simulated in order to assess their limited action on the implemented protocols. However, in order to replicate the experimental data presented in Figure 6.5, we resort to an effective decoherence model in the form of solving the Master equation with a relaxation rate of $2\pi \times 0.06$ MHz (BVC+13). This value was obtained by fitting with the above model damped Rabi oscillations measured on the same device. Thus, reaching similar behaviour within error bars between numerically simulated and experimentally obtained $JS(\mathcal{P}_1, \mathcal{P}_2)$ was achieved with no free parameter.

Bibliography

[AAA+24] A. Abbas, A. Ambainis, B. Augustino, A. Bärtschi, H. Buhrman, C. Coffrin, G. Cortiana, V. Dunjko, D. J. Egger, B. G. Elmegreen, et al.
Challenges and opportunities in quantum optimization.
Nature Reviews Physics, pages 1–18, 2024.

[ADL+22] J.-B. Alayrac, J. Donahue, P. Luc, A. Miech, I. Barr, Y. Hasson, K. Lenc, A. Mensch, K. Millican, M. Reynolds, et al.
Flamingo: a visual language model for few-shot learning.
arXiv preprint arXiv:2204.14198, 2022.

[ADL+23] B. Albrecht, C. Dalyac, L. Leclerc, L. Ortiz-Gutiérrez, S. Thabet, M. D'Arcangelo, J. R. Cline, V. E. Elfving, L. Lassablière, H. Silvério, et al.
Quantum feature maps for graph machine learning on a neutral atom quantum processor.
Physical Review A, 107(4):042615, 2023.

[AI23] O. AI.
Gpt-4 technical report.
arXiv preprint arXiv:2303.08774, 2023.

[Ami09] M. H. Amin.
Consistency of the adiabatic theorem.
Physical review letters, 102(22):220401, 2009.

[Bam22] J. Bamberger.
A topological characterisation of weisfeiler-leman equivalence classes.
In *Topological, Algebraic and Geometric Learning Workshops 2022*, pages 17–27. PMLR, 2022.

[BBCV21] M. M. Bronstein, J. Bruna, T. Cohen, and P. Veličković.
Geometric deep learning: Grids, groups, graphs, geodesics, and gauges.
arXiv preprint arXiv:2104.13478, 2021.

[BBR+24] P. Bermejo, P. Braccia, M. S. Rudolph, Z. Holmes, L. Cincio, and M. Cerezo.
Quantum convolutional neural networks are (effectively) classically simulable.
arXiv preprint arXiv:2408.12739, 2024.

[BC21] L. Banchi and G. E. Crooks.
Measuring analytic gradients of general quantum evolution with the stochastic parameter shift rule.
Quantum, 5:386, 2021.

[BCB14] D. Bahdanau, K. Cho, and Y. Bengio.
Neural machine translation by jointly learning to align and translate. arxiv 2014.
arXiv preprint arXiv:1409.0473, 2014.

[BEG+24] D. Bluvstein, S. J. Evered, A. A. Geim, S. H. Li, H. Zhou, T. Manovitz, S. Ebadi, M. Cain, M. Kalinowski, D. Hangleiter, et al.
Logical quantum processor based on reconfigurable atom arrays.
Nature, 626(7997):58–65, 2024.

[BFW+21] C. Bodnar, F. Frasca, Y. Wang, N. Otter, G. F. Montufar, P. Lió, and M. Bronstein.
Weisfeiler and lehman go topological: Message passing simplicial networks.
In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 1026–1037. PMLR, 18–24 Jul 2021.
URL https://proceedings.mlr.press/v139/bodnar21a.html.

[BFZB22] G. Bouritsas, F. Frasca, S. Zafeiriou, and M. M. Bronstein.
Improving graph neural network expressivity via subgraph isomorphism counting.
IEEE Transactions on Pattern Analysis and Machine Intelligence, 45(1):657–668, 2022.

[BGLL+20] K. M. Borgwardt, M. E. Ghisu, F. Llinares-López, L. O'Bray, and B. Rieck.
Graph kernels: State-of-the-art and future challenges.
Foundations and Trends in Machine Learning, 13(5-6), 2020.
doi:10.1561/2200000076.

[BH09] J. Briët and P. Harremoës.
Properties of classical and quantum jensen-shannon divergence.
Phys. Rev. A, 79:052311, May 2009.
doi:10.1103/PhysRevA.79.052311.

[Big93] N. Biggs.
Algebraic graph theory.
Cambridge Univ Pr, 1993.

[BK79] L. Babai and L. Kucera.
Canonical labelling of graphs in linear average time.
In *20th annual symposium on foundations of computer science (sfcs 1979)*, pages 39–46. IEEE, 1979.

[BLL+18] D. Barredo, V. Lienhard, S. d. Léséleuc, T. Lahaye, and A. Browaeys.
Synthetic three-dimensional atomic structures assembled atom by atom.
Nature, 561(7721):79–82, Sept. 2018.
doi:10.1038/s41586-018-0450-2.

[BMR+20] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al.
Language models are few-shot learners.
Advances in neural information processing systems, 33:1877–1901, 2020.

[BMS16] M. J. Bremner, A. Montanaro, and D. J. Shepherd.
Average-case complexity versus approximate simulation of commuting quantum computations.
Physical review letters, 117(8):080501, 2016.

[BN06] C. M. Bishop and N. M. Nasrabadi.
Pattern recognition and machine learning, volume 4.
Springer, 2006.

[BOS+05] K. M. Borgwardt, C. S. Ong, S. Schönauer, S. V. Vishwanathan, A. J. Smola, and H. P. Kriegel.
Protein function prediction via graph kernels.
Bioinformatics, 21, 2005.
doi:10.1093/bioinformatics/bti1007.

[BPL+21] D. Beaini, S. Passaro, V. Létourneau, W. Hamilton, G. Corso, and P. Liò.
Directional graph networks.
In *International Conference on Machine Learning*, pages 748–758. PMLR, 2021.

[BR09] L. C. Blum and J.-L. Reymond.
970 million druglike small molecules for virtual screening in the chemical universe database gdb-13.
Journal of the American Chemical Society, 131(25):8732–8733, 2009, https://doi.org/10.1021/ja902302h.

doi:10.1021/ja902302h.
PMID: 19505099.

[BVC+13] L. Beguin, A. Vernier, R. Chicireanu, T. Lahaye, and A. Browaeys.
Direct measurement of the van der waals interaction between two rydberg atoms.
Physical review letters, 110(26):263201, 2013.

[BWP+17] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd.
Quantum machine learning.
Nature, 549(7671):195–202, 2017.

[C.04] M. D. J. C.
An Example Inference Task: Clustering.
Cambridge University press, 2004.

[CAB+21] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, et al.
Variational quantum algorithms.
Nature Reviews Physics, 3(9):625–644, 2021.

[CCL19] I. Cong, S. Choi, and M. D. Lukin.
Quantum convolutional neural networks.
Nature Physics, 15(12):1273–1278, 2019.

[CGFM+21] M. C. Caro, E. Gil-Fuster, J. J. Meyer, J. Eisert, and R. Sweke.
Encoding-dependent generalization bounds for parametrized quantum circuits.
Quantum, 5:582, Nov. 2021.
doi:10.22331/q-2021-11-17-582.

[cir95] Quantum computations with cold trapped ions.
Physical review letters, 74(20):4091, 1995.

[CLL+20] D. Chen, Y. Lin, W. Li, P. Li, J. Zhou, and X. Sun.
Measuring and relieving the over-smoothing problem for graph neural networks from the topological view.
In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3438–3445, 2020.

[CMDF+24] A. Ceschini, F. Mauro, F. De Falco, A. Sebastianelli, A. Verdone, A. Rosato, B. L. Saux, M. Panella, P. Gamba, and S. L. Ullo.
From graphs to qubits: A critical review of quantum graph neural networks.
arXiv preprint arXiv:2408.06524, 2024.

[CMS+20] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko.
End-to-end object detection with transformers.
In *European conference on computer vision*, pages 213–229. Springer, 2020.

[CŚ06] B. Collins and P. Śniady.
Integration with respect to the haar measure on unitary, orthogonal and symplectic group.
Communications in Mathematical Physics, 264(3):773–795, 2006.

[CSV+21] M. Cerezo, A. Sone, T. Volkoff, L. Cincio, and P. J. Coles.
Cost function dependent barren plateaus in shallow parametrized quantum circuits.
Nature communications, 12(1):1791, 2021.

[CVH+22] M. Cerezo, G. Verdon, H.-Y. Huang, L. Cincio, and P. J. Coles.
Challenges and opportunities in quantum machine learning.
Nature Computational Science, 2(9):567–576, 2022.

[Dal23] C. Dalyac.
Quantum many-body dynamics for combinatorial optimisation and machine learning.
PhD thesis, Sorbonne Université, 2023.

[DAR+22] A. Dawid, J. Arnold, B. Requena, A. Gresch, M. Płodzień, K. Donatella, K. A. Nicoli, P. Stornati, R. Koch, M. Büttner, et al.
Modern applications of machine learning in quantum sciences.
arXiv preprint arXiv:2204.04198, 2022.

[DBK+20] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al.
An image is worth 16x16 words: Transformers for image recognition at scale.
In *International Conference on Learning Representations*, 2020.

[DBK+22] A. J. Daley, I. Bloch, C. Kokail, S. Flannigan, N. Pearson, M. Troyer, and P. Zoller.
Practical quantum advantage in quantum simulation.
Nature, 607(7920):667–676, 2022.

[DC08] A. Das and B. K. Chakrabarti.
Colloquium: Quantum annealing and analog quantum computation.
Reviews of Modern Physics, 80(3):1061, 2008.

[dev13] Superconducting circuits for quantum information: an outlook.
Science, 339(6124):1169–1174, 2013.

[Dev18] J. Devlin.
Bert: Pre-training of deep bidirectional transformers for language understanding.
arXiv preprint arXiv:1810.04805, 2018.

[DHK+23] C. Dalyac, L.-P. Henry, M. Kim, J. Ahn, and L. Henriet.
Exploring the impact of graph locality for the resolution of mis with neutral atom devices, 2023, 2306.13373.

[Dij59] E. W. Dijkstra.
A note on two problems in connexion with graphs.
Numerische Mathematik, 1(1):269–271, Dec 1959.
doi:10.1007/BF01386390.

[DJL+20] V. P. Dwivedi, C. K. Joshi, T. Laurent, Y. Bengio, and X. Bresson.
Benchmarking graph neural networks.
2020.

[DLL+21] V. P. Dwivedi, A. T. Luu, T. Laurent, Y. Bengio, and X. Bresson.
Graph neural networks with learnable structural and positional representations.
arXiv preprint arXiv:2110.07875, 2021.

[dN10] M. V. den Nest.
Simulating quantum computers with probabilistic methods, 2010, 0911.1624.
URL https://arxiv.org/abs/0911.1624.

[EKC+22] S. Ebadi, A. Keesling, M. Cain, T. T. Wang, H. Levine, D. Bluvstein, G. Semeghini, A. Omran, J.-G. Liu, R. Samajdar, et al.
Quantum optimization of maximum independent set using rydberg atom arrays.
Science, 376(6598):1209–1215, 2022.

[FGG14] E. Farhi, J. Goldstone, and S. Gutmann.
A quantum approximate optimization algorithm.
arXiv preprint arXiv:1411.4028, 2014.

[Flo62] R. W. Floyd.
Algorithm 97: Shortest path.
Commun. ACM, 5(6):345, jun 1962.
doi:10.1145/367766.368168.

[FR91] T. M. J. Fruchterman and E. M. Reingold.
Graph drawing by force-directed placement.
Software: Practice and Experience, 21(11):1129–1164, 1991, https://onlinelibrary.wiley.com/doi/pdf/10.1002/spe.4380211102.
doi:https://doi.org/10.1002/spe.4380211102.

[Fra18] P. I. Frazier.
A tutorial on bayesian optimization.
arXiv preprint arXiv:1807.02811, 2018.

[Fuk99] M. Fukuda.
Weingarten functions for p = 4, 1999.
URL https://motohisafukuda.github.io/RTNI/PYTHON/Weingarten/functions4.html.

[GB17] C. Gross and I. Bloch.
Quantum simulations with ultracold atoms in optical lattices.
Science, 357(6355):995–1001, 2017.

[GD23] C. Gyurik and V. Dunjko.
Exponential separations between classical and quantum learners.
arXiv preprint arXiv:2306.16028, 2023.

[GFW03] T. Gärtner, P. Flach, and S. Wrobel.
On graph kernels: Hardness results and efficient alternatives.
In B. Schölkopf and M. K. Warmuth, editors, *Learning Theory and Kernel Machines*, pages 129–143, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[GFZ+10] J. K. Gamble, M. Friesen, D. Zhou, R. Joynt, and S. Coppersmith.
Two-particle quantum walks applied to the graph isomorphism problem.
Physical Review A, 81(5):052313, 2010.

[GHL+15] S. Gharibian, Y. Huang, Z. Landau, S. W. Shin, et al.
Quantum hamiltonian complexity.
Foundations and Trends® in Theoretical Computer Science, 10(3):159–282, 2015.

[GJ78] M. R. Garey and D. S. Johnson.
"strong"np-completeness results: Motivation, examples, and implications.
Journal of the ACM (JACM), 25(3):499–508, 1978.

[GJ79] M. R. Garey and D. S. Johnson.
Computers And Intractability: A Guide To The Theory Of NP-Completeness.
1979.

[goo23] Suppressing quantum errors by scaling a surface code logical qubit.
Nature, 614(7949):676–681, 2023.

[Got02] D. Gottesman.
An introduction to quantum error correction.
In *Proceedings of Symposia in Applied Mathematics*, volume 58, pages 221–236, 2002.

[GRLR+23] É. Gouzien, D. Ruiz, F.-M. Le Régent, J. Guillaud, and N. Sangouard.
Performance analysis of a repetition cat code architecture: Computing 256-bit elliptic curve logarithm in 9 hours with 126 133 cat qubits.
Physical review letters, 131(4):040602, 2023.

[GŠ13] G. Glavaš and J. Šnajder.
Recognizing identical events with graph kernels.
In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 797–803, 2013.

[GSR+17] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl.
Neural message passing for quantum chemistry.
In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1263–1272. PMLR, 06–11 Aug 2017.
URL http://proceedings.mlr.press/v70/gilmer17a.html.

[Had18] S. A. Hadfield.
Quantum algorithms for scientific computing and approximate optimization.
Columbia University, 2018.

[Ham20] W. L. Hamilton.
Graph representation learning.
Morgan & Claypool Publishers, 2020.

[HB07] Z. Harchaoui and F. Bach.
Image classification with segmentation graph kernels.
In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.

[HBM+21] H.-Y. Huang, M. Broughton, M. Mohseni, R. Babbush, S. Boixo, H. Neven, and J. R. McClean.
Power of data in quantum machine learning.
Nature Communications, 12(1):2631, May 2021.
doi:10.1038/s41467-021-22539-9.

[HBS+20] L. Henriet, L. Beguin, A. Signoles, T. Lahaye, A. Browaeys, G.-O. Reymond, and C. Jurczak.
Quantum computing with neutral atoms.
Quantum, 4:327, 2020.

[HCT+19] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta.
Supervised learning with quantum-enhanced feature spaces.
Nature, 567(7747):209–212, Mar 2019.
doi:10.1038/s41586-019-0980-2.

[HDE+06] M. Hein, W. Dür, J. Eisert, R. Raussendorf, M. Van den Nest, and H.-J. Briegel.
Entanglement in graph states and its applications.
In *Quantum computers, algorithms and chaos*, pages 115–218. IOS Press, 2006.

[HFR+21] W. Hu, M. Fey, H. Ren, M. Nakata, Y. Dong, and J. Leskovec.
Ogb-lsc: A large-scale challenge for machine learning on graphs.
arXiv preprint arXiv:2103.09430, 2021.

[HHL09] A. W. Harrow, A. Hassidim, and S. Lloyd.
Quantum algorithm for linear systems of equations.
Physical review letters, 103(15):150502, 2009.

[HKKS01] C. Helma, R. D. King, S. Kramer, and A. Srinivasan.
The Predictive Toxicology Challenge 2000–2001 .
Bioinformatics, 17(1):107–108, 01 2001, https://academic.oup.com/bioinformatics/article-pdf/17/1/107/576297/170107.pdf.
doi:10.1093/bioinformatics/17.1.107.

[HKP20] H.-Y. Huang, R. Kueng, and J. Preskill.
Predicting many properties of a quantum system from very few measurements.
Nature Physics, 16(10):1050–1057, 2020.

[HL11] M. Hilbert and P. López.
The world's technological capacity to store, communicate, and compute information.
Science, 332(6025):60–65, 2011, https://www.science.org/doi/pdf/10.1126/science.1200970.
doi:10.1126/science.1200970.

[HLH22] J. Ha, J. Lee, and J. Heo.
Resource analysis of quantum computing with noisy qubits for shor's factoring algorithms.
Quantum Information Processing, 21(2):60, 2022.

[HMRT22] T. Hastie, A. Montanari, S. Rosset, and R. J. Tibshirani.
Surprises in high-dimensional ridgeless least squares interpolation.
The Annals of Statistics, 50(2):949–986, 2022.

[HSCC22] Z. Holmes, K. Sharma, M. Cerezo, and P. J. Coles.
Connecting ansatz expressibility to gradient magnitudes and barren plateaus.
PRX Quantum, 3:010313, Jan 2022.
doi:10.1103/PRXQuantum.3.010313.

[HSS08] T. Hofmann, B. Schölkopf, and A. J. Smola.
Kernel methods in machine learning.
The Annals of Statistics, 36(3):1171–1220, 2008.
doi:10.1214/009053607000000677.

[HTDH21a] L.-P. Henry, S. Thabet, C. Dalyac, and L. Henriet.
Quantum evolution kernel: Machine learning on graphs with programmable arrays of qubits.
Phys. Rev. A, 104:032416, Sep 2021.
doi:10.1103/PhysRevA.104.032416.

[HTDH21b] L.-P. Henry, S. Thabet, C. Dalyac, and L. Henriet.
Quantum evolution kernel: Machine learning on graphs with programmable arrays of qubits.
Physical Review A, 104(3):032416, 2021.

[HYL18] W. L. Hamilton, R. Ying, and J. Leskovec.
Inductive representation learning on large graphs, 2018, 1706.02216.

[ISM+12] J. J. Irwin, T. Sterling, M. M. Mysinger, E. S. Bolstad, and R. G. Coleman.
Zinc: a free tool to discover chemistry for biology.
Journal of chemical information and modeling, 52(7):1757–1768, 2012.

[JEP+21] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, et al.
Highly accurate protein structure prediction with alphafold.
Nature, 596(7873):583–589, 2021.

[JFPN+23] S. Jerbi, L. J. Fiderer, H. Poulsen Nautrup, J. M. Kübler, H. J. Briegel, and V. Dunjko.
Quantum machine learning beyond kernel methods.
Nature Communications, 14(1):517, 2023.

[JGM+23] S. Jerbi, C. Gyurik, S. C. Marshall, R. Molteni, and V. Dunjko.
Shadows of quantum machine learning.
arXiv preprint arXiv:2306.00061, 2023.

[JGM+24] S. Jerbi, C. Gyurik, S. C. Marshall, R. Molteni, and V. Dunjko.
Shadows of quantum machine learning.
Nature Communications, 15(1):5676, 2024.

[Kat50] T. Kato.
On the adiabatic theorem of quantum mechanics.
Journal of the Physical Society of Japan, 5(6):435–439, 1950.

[KB14] D. P. Kingma and J. Ba.
Adam: A method for stochastic optimization.
arXiv preprint arXiv:1412.6980, 2014.

[KBH+21] D. Kreuzer, D. Beaini, W. Hamilton, V. Létourneau, and P. Tossou.
Rethinking graph transformers with spectral attention.
Advances in Neural Information Processing Systems, 34:21618–21629, 2021.

[KE21] O. Kyriienko and V. E. Elfving.
Generalized quantum circuit differentiation rules.
Physical Review A, 104(5):052417, 2021.

[Kit97] A. Y. Kitaev.
Quantum computations: algorithms and error correction.
Russian Mathematical Surveys, 52(6):1191, 1997.

[KJM20] N. M. Kriege, F. D. Johansson, and C. Morris.
A survey on graph kernels.
Applied Network Science, 5(6), 2020.
doi:10.1007/s41109-019-0195-3.

[KLLP19] I. Kerenidis, J. Landman, A. Luongo, and A. Prakash.
q-means: A quantum algorithm for unsupervised machine learning.
Advances in neural information processing systems, 32, 2019.

[KLP19] I. Kerenidis, J. Landman, and A. Prakash.
Quantum algorithms for deep convolutional neural networks.
arXiv preprint arXiv:1911.01117, 2019.

[Kon14] M. I. Konaklieva.
Molecular targets of β-lactam-based antimicrobials: beyond the usual suspects.
Antibiotics, 3(2):128–142, 2014.

[KP16] I. Kerenidis and A. Prakash.
Quantum recommendation systems.
arXiv preprint arXiv:1603.08675, 2016.

[KPE21] O. Kyriienko, A. E. Paine, and V. E. Elfving.
Solving nonlinear differential equations with differentiable quantum circuits.
Physical Review A, 103(5):052416, 2021.

[KPE22] O. Kyriienko, A. E. Paine, and V. E. Elfving.
Protocols for trainable and differentiable quantum generative modelling.
arXiv preprint arXiv:2202.08253, 2022.

[KSH12] A. Krizhevsky, I. Sutskever, and G. E. Hinton.
Imagenet classification with deep convolutional neural networks.
Advances in neural information processing systems, 25, 2012.

[KW16] T. N. Kipf and M. Welling.
Semi-supervised classification with graph convolutional networks.
arXiv preprint arXiv:1609.02907, 2016.
doi:10.48550/ARXIV.1609.02907.

[LAT21] Y. Liu, S. Arunachalam, and K. Temme.
A rigorous and robust quantum speed-up in supervised machine learning.
Nature Physics, 17(9):1013–1017, 2021.

[LB+] Y. LeCun, Y. Bengio, et al.
Convolutional networks for images, speech, and time series.

[LBH15] Y. LeCun, Y. Bengio, and G. Hinton.
Deep learning.
nature, 521(7553):436–444, 2015.

[Ldk18] S. d. Leseleuc de kerouara.
Quantum simulation of spin models with assembled arrays of Rydberg atoms.
Theses, Université Paris Saclay (COmUE), Dec. 2018.
URL https://pastel.archives-ouvertes.fr/tel-02088297.

[lec89] Backpropagation applied to handwritten zip code recognition.
Neural computation, 1(4):541–551, 1989.

[LH22] L. Leclerc and L. Henriet.
Pusho, a pulse-level variational quantum algorithm for rydberg atom platforms.
In *2022 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pages 839–841, 2022.
doi:10.1109/QCE53715.2022.00138.

[LHW18] Q. Li, Z. Han, and X.-M. Wu.
Deeper insights into graph convolutional networks for semi-supervised learning.
In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[Lin91] J. Lin.
Divergence measures based on the shannon entropy.
IEEE Transactions on Information Theory, 37(1):145–151, 1991.
doi:10.1109/18.61115.

[Lov79] L. Lovasz.
On the shannon capacity of a graph.
IEEE Transactions on Information Theory, 25(1):1–7, 1979.
doi:10.1109/TIT.1979.1055985.

[LR72] E. H. Lieb and D. W. Robinson.
The finite group velocity of quantum spin systems.
Communications in Mathematical Physics, 28(3):251 – 257, 1972.
doi:cmp/1103858407.

[LR15] P. Latouche and F. Rossi.
Graphs in machine learning: an introduction.
European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN), pages 207–218, Apr. 2015.

[LRZ+22] D. Lim, J. Robinson, L. Zhao, T. Smidt, S. Sra, H. Maron, and S. Jegelka.
Sign and basis invariant networks for spectral graph representation learning.
arXiv preprint arXiv:2202.13013, 2022.

[LTD+22] J. Landman, S. Thabet, C. Dalyac, H. Mhiri, and E. Kashefi.
Classically approximating variational quantum machine learning with random fourier features, 2022, 2210.13200.

[LTOS19] Z. Li, J.-F. Ton, D. Oglic, and D. Sejdinovic.
Towards a unified analysis of random fourier features.
In *International conference on machine learning*, pages 3905–3914. PMLR, 2019.

[LTW+24] M. Larocca, S. Thanasilp, S. Wang, K. Sharma, J. Biamonte, P. J. Coles, L. Cincio, J. R. McClean, Z. Holmes, and M. Cerezo.
A review of barren plateaus in variational quantum computing.
arXiv preprint arXiv:2405.00781, 2024.

[Luc14] A. Lucas.
Ising formulations of many np problems.
Frontiers in physics, 2:5, 2014.

[LZF24] Y. Liao, X.-M. Zhang, and C. Ferrie.
Graph neural networks on quantum computers.
arXiv preprint arXiv:2405.17060, 2024.

[MBS+18] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven.
Barren plateaus in quantum neural network training landscapes.
Nature communications, 9(1):1–6, 2018.

[MCSM21] G. Mialon, D. Chen, M. Selosse, and J. Mairal.
Graphit: Encoding graph structure in transformers.
arXiv preprint arXiv:2106.05667, 2021.

[MGD24] R. Molteni, C. Gyurik, and V. Dunjko.
Exponential quantum advantages in learning quantum observables from classical data.
arXiv preprint arXiv:2405.02027, 2024.

[MKB+20] C. Morris, N. M. Kriege, F. Bause, K. Kersting, P. Mutzel, and M. Neumann.
Tudataset: A collection of benchmark datasets for learning with graphs.
In *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*, 2020, 2007.08663.
URL www.graphlearning.io.

[MLL+23] L. Ma, C. Lin, D. Lim, A. Romero-Soriano, P. K. Dokania, M. Coates, P. Torr, and S.-N. Lim.
Graph inductive biases in transformers without message passing.
arXiv preprint arXiv:2305.17589, 2023.

[MMC22] P. Mernyei, K. Meichanetzidis, and I. I. Ceylan.
Equivariant quantum graph circuits.
In *International Conference on Machine Learning*, pages 15401–15420. PMLR, 2022.

[MMHG+24] H. Mhiri, L. Monbroussou, M. Herrero-Gonzalez, S. Thabet, E. Kashefi, and J. Landman.
Constrained and vanishing expressivity of quantum fourier models, 2024, 2403.09417.

[MNKF18] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii.
Quantum circuit learning.
Physical Review A, 98(3):032309, 2018.

[MOB20] G. Muzio, L. O'Bray, and K. Borgwardt.
Biological network analysis with deep learning.
Briefings in Bioinformatics, 22(2):1515–1530, 11 2020, https://academic.oup.com/bib/article-pdf/22/2/1515/36655309/bbaa257.pdf.
doi:10.1093/bib/bbaa257.

[MRF+19] C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, and M. Grohe.
Weisfeiler and leman go neural: Higher-order graph neural networks.
In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 4602–4609, 2019.

[MRG+13] G. Montavon, M. Rupp, V. Gobre, A. Vazquez-Mayagoitia, K. Hansen, A. Tkatchenko, K.-R. Müller, and O. A. von Lilienfeld.
Machine learning of molecular electronic properties in chemical compound space.

New Journal of Physics, 15(9):095003, sep 2013.
doi:10.1088/1367-2630/15/9/095003.

[MXZ06] C. A. Micchelli, Y. Xu, and H. Zhang.
Universal kernels.
Journal of Machine Learning Research, 7(95):2651–2667, 2006.
URL http://jmlr.org/papers/v7/micchelli06a.html.

[NC11] M. A. Nielsen and I. L. Chuang.
Quantum Computation and Quantum Information: 10th Anniversary Edition.
Cambridge University Press, New York, NY, USA, 10th edition, 2011.
doi:10.1017/CBO9780511976667.

[NM65] J. A. Nelder and R. Mead.
A simplex method for function minimization.
The computer journal, 7(4):308–313, 1965.

[NMR+17] G. Nikolentzos, P. Meladianos, F. Rousseau, Y. Stavrakas, and M. Vazirgiannis.
Shortest-path graph kernels for document similarity.
In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1890–1900, 2017.

[NSV19] G. Nikolentzos, G. Siglidis, and M. Vazirgiannis.
Graph kernels: A survey, 2019, 1904.12218.

[O'b07] J. L. O'brien.
Optical quantum computing.
Science, 318(5856):1567–1570, 2007.

[PARS14] B. Perozzi, R. Al-Rfou, and S. Skiena.
Deepwalk: Online learning of social representations.
In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.

[PGM+19] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al.
Pytorch: An imperative style, high-performance deep learning library.
Advances in neural information processing systems, 32, 2019.

[PLB+24] G. Pichard, D. Lim, É. Bloch, J. Vaneecloo, L. Bourachot, G.-J. Both, G. Mériaux, S. Dutartre, R. Hostein, J. Paris, et al.
Rearrangement of individual atoms in a 2000-site optical-tweezer array at cryogenic temperatures.
Physical Review Applied, 22(2):024073, 2024.

[PMS+14] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'brien.
A variational eigenvalue solver on a photonic quantum processor.
Nature communications, 5(1):4213, 2014.

[POG+20] P. Pawara, E. Okafor, M. Groefsema, S. He, L. R. Schomaker, and M. A. Wiering.
One-vs-one classification for deep neural networks.
Pattern Recognition, 108:107528, 2020.
doi:https://doi.org/10.1016/j.patcog.2020.107528.

[PS22] E. Peters and M. Schuld.
Generalization despite overfitting in quantum machine learning models.
arXiv preprint arXiv:2209.05523, 2022.

[PZZY13] S. Pan, X. Zhu, C. Zhang, and P. S. Yu.
Graph stream classification using labeled and unlabeled graphs.
In *2013 IEEE 29th International Conference on Data Engineering (ICDE)*, pages 398–409, 2013.
doi:10.1109/ICDE.2013.6544842.

[RAAB25] E. Recio-Armengol, S. Ahmed, and J. Bowles.
Train on classical, deploy on quantum: scaling generative quantum machine learning to a thousand qubits.
arXiv preprint arXiv:2503.02934, 2025.

[RB08] K. Riesen and H. Bunke.
Iam graph database repository for graph based pattern recognition and machine learning.
In N. da Vitoria Lobo, T. Kasparis, F. Roli, J. T. Kwok, M. Georgiopoulos, G. C. Anagnostopoulos, and M. Loog, editors, *Structural, Syntactic, and Statistical Pattern Recognition*, pages 287–297, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[RFHC23] M. S. Rudolph, E. Fontana, Z. Holmes, and L. Cincio.
Classical surrogate simulation of quantum systems with lowesa.
arXiv preprint arXiv:2308.09109, 2023.

[RGD+22] L. Rampášek, M. Galkin, V. P. Dwivedi, A. T. Luu, G. Wolf, and D. Beaini.
Recipe for a general, powerful, scalable graph transformer.
arXiv preprint arXiv:2205.12454, 2022.

[Ros58] F. Rosenblatt.
The perceptron: a probabilistic model for information storage and organization in the brain.
Psychological review, 65(6):386, 1958.

[RR07] A. Rahimi and B. Recht.
Random features for large-scale kernel machines.
Advances in neural information processing systems, 20, 2007.

[RR08a] A. Rahimi and B. Recht.
Uniform approximation of functions with random bases.
In *2008 46th annual allerton conference on communication, control, and computing*, pages 555–561. IEEE, 2008.

[RR08b] A. Rahimi and B. Recht.
Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning.
Advances in neural information processing systems, 21, 2008.

[RR17] A. Rudi and L. Rosasco.
Generalization properties of learning with random features.
Advances in neural information processing systems, 30, 2017.

[RS] G. Rattan and T. Seppelt.
Weisfeiler-Leman and Graph Spectra, pages 2268–2285.
doi:10.1137/1.9781611977554.ch87.

[Rud16] S. Ruder.
An overview of gradient descent optimization algorithms.
arXiv preprint arXiv:1609.04747, 2016.

[Rud17] W. Rudin.
Fourier analysis on groups.
Courier Dover Publications, 2017.

[SBI+20] M. Schuld, K. Brádler, R. Israel, D. Su, and B. Gupt.
Measuring the similarity of graphs with a gaussian boson sampler.
Phys. Rev. A, 101:032314, Mar 2020.
doi:10.1103/PhysRevA.101.032314.

[SBSW20] M. Schuld, A. Bocharov, K. M. Svore, and N. Wiebe.
Circuit-centric quantum classifiers.
Physical Review A, 101(3):032308, 2020.

[Sch21] M. Schuld.
Supervised quantum machine learning models are kernel methods.
arXiv preprint arXiv:2101.11020, 2021.

[Sco11] J. Scott.
Social network analysis: developments, advances, and prospects.
Social Network Analysis and Mining, 1(1), 01 2011.
doi:10.1007/s13278-010-0012-6.

[SEM22] F. J. Schreiber, J. Eisert, and J. J. Meyer.
Classical surrogates for quantum learning models.
arXiv preprint arXiv:2206.11740, 2022.

[SEM23] F. J. Schreiber, J. Eisert, and J. J. Meyer.
Classical surrogates for quantum learning models.
Physical Review Letters, 131(10):100803, 2023.

[SGGP+20] A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, and P. Battaglia.
Learning to simulate complex physics with graph networks.
In *International Conference on Machine Learning*, pages 8459–8468. PMLR, 2020.

[Sho94] P. W. Shor.
Algorithms for quantum computation: discrete logarithms and factoring.
In *Proceedings 35th annual symposium on foundations of computer science*, pages 124–134. Ieee, 1994.
doi:10.1109/sfcs.1994.365700.

[SHS01] B. Schölkopf, R. Herbrich, and A. J. Smola.
A generalized representer theorem.
In *International conference on computational learning theory*, pages 416–426. Springer, 2001.

[SJ03] S.-y. Shiau and R. Joynt.
Physically-motivated dynamical algorithms for the graph isomorphism problem.
arXiv preprint quant-ph/0312170, 2003.

[SP18] M. Schuld and F. Petruccione.
Supervised learning with quantum computers, volume 17.
Springer, 2018.

[Spe24] E. Spence, 2024.
URL https://www.maths.gla.ac.uk/~es/srgraphs.php.
Accessed on 31st January 2024.

[Spi99] F. C. Spieksma.
On the approximability of an interval scheduling problem.
Journal of Scheduling, 2(5):215–227, 1999.

[SRJ+23] R. Sweke, E. Recio, S. Jerbi, E. Gil-Fuster, B. Fuller, J. Eisert, and J. J. Meyer.
Potential and limitations of random fourier features for dequantizing quantum machine learning.
arXiv preprint arXiv:2309.11647, 2023.

[SRPG01] N. Schlosser, G. Reymond, I. Protsenko, and P. Grangier.
Sub-poissonian loading of single atoms in a microscopic dipole trap.
Nature, 411(6841):1024–1027, 2001.

[SS01] B. Schlköpf and A. J. Smola.
Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.
The MIT Press, 2001.

[SS15] D. J. Sutherland and J. Schneider.
On the error of random fourier features.
arXiv preprint arXiv:1506.02785, 2015.

[SSM21a] M. Schuld, R. Sweke, and J. J. Meyer.
Effect of data encoding on the expressive power of variational quantum-machine-learning models.
Physical Review A, 103(3):032430, 2021.

[SSM21b] M. Schuld, R. Sweke, and J. J. Meyer.
Effect of data encoding on the expressive power of variational quantum-machine-learning models.
Physical Review A, 103(3):1–16, 2021, 2008.08605.
doi:10.1103/PhysRevA.103.032430.

[SSS+17] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al.
Mastering the game of go without human knowledge.
nature, 550(7676):354–359, 2017.

[SSVL+11] N. Shervashidze, P. Schweitzer, E. J. Van Leeuwen, K. Mehlhorn, and K. M. Borgwardt.
Weisfeiler-lehman graph kernels.
Journal of Machine Learning Research, 12(9), 2011.
URL https://www.jmlr.org/papers/volume12/shervashidze11a/shervashidze11a.pdf.

[SSW+21] P. Scholl, M. Schuler, H. J. Williams, A. A. Eberharter, D. Barredo, K.-N. Schymik, V. Lienhard, L.-P. Henry, T. C. Lang, T. Lahaye, et al.
Quantum simulation of 2d antiferromagnets with hundreds of rydberg atoms.
Nature, 595(7866):233–238, 2021.

[ST95] J. SHAO and D. TU.
The Jackknife and Bootstrap.
SPRINGER, 1995.

[STJ22] S. Shin, Y. Teo, and H. Jeong.
Exponential data encoding for quantum supervised learning.
arXiv preprint arXiv:2206.12105, 2022.

[Suz76] M. Suzuki.
Generalized trotter's formula and systematic approximants of exponential operators and inner derivations with applications to many-body problems.
Communications in Mathematical Physics, 51(2):183–190, 1976.

[SVP+09] N. Shervashidze, S. Vishwanathan, T. Petri, K. Mehlhorn, and K. Borgwardt.
Efficient graphlet kernels for large graph comparison.
In *JMLR Workshop and Conference Proceedings Volume 5: AISTATS 2009*, pages 488–495, Cambridge, MA, USA, Apr. 2009. Max-Planck-Gesellschaft, MIT Press.

[SWB+22] P. Scholl, H. J. Williams, G. Bornet, F. Wallner, D. Barredo, L. Henriet, A. Signoles, C. Hainaut, T. Franz, S. Geier, et al.
Microwave engineering of programmable x x z hamiltonians in arrays of rydberg atoms.
PRX Quantum, 3(2):020303, 2022.

[Tan19] E. Tang.
A quantum-inspired classical algorithm for recommendation systems.
Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, pages 217–228, June 2019.
doi:10.1145/3313276.3316310.

[Tan23] E. Tang.
Quantum Machine Learning Without Any Quantum.
PhD thesis, University of Washington, 2023.

[TDGC+21] J. Topping, F. Di Giovanni, B. P. Chamberlain, X. Dong, and M. M. Bronstein.
Understanding over-squashing and bottlenecks on graphs via curvature.
arXiv preprint arXiv:2111.14522, 2021.

[TDS+] S. Thabet, M. Djellabi, I. O. Sokolov, S. Kasture, L.-P. Henry, and L. Henriet.
Quantum positional encodings for graph neural networks.
In *Forty-first International Conference on Machine Learning.*

[TFH22] S. Thabet, R. Fouilland, and L. Henriet.
Extending graph transformers with quantum computed aggregation.
arXiv preprint arXiv:2210.10610, 2022.

[TGS+19] M. C. Tran, A. Y. Guo, Y. Su, J. R. Garrison, Z. Eldredge, M. Foss-Feig, A. M. Childs, and A. V. Gorshkov.
Locality and digital quantum simulation of power-law interactions.
Phys. Rev. X, 9:031006, Jul 2019.
doi:10.1103/PhysRevX.9.031006.

[TML24] S. Thabet, E. Z. Monbroussou, Léo sand Mamon, and J. Landman.
When quantum and classical models disagree: Learning beyond minimum norm least square.
arXiv preprint arXiv:2411.04940, 2024.

[TYE22] Y. Tang, J. Yan, and H. Edwin.
From quantum graph computing to quantum graph learning: A survey.
arXiv preprint arXiv:2202.09506, 2022.

[UK24] C. Umeano and O. Kyriienko.
Ground state-based quantum feature maps.
arXiv preprint arXiv:2404.07174, 2024.

[VB12] A. Varnek and I. Baskin.
Machine learning methods for property prediction in chemoinformatics: Quo vadis?
Journal of Chemical Information and Modeling, 52(6):1413–1437, 2012, https://doi.org/10.1021/ci200409x.
doi:10.1021/ci200409x.
PMID: 22582859.

[VCC+18] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio.
Graph attention networks, 2018, 1710.10903.

[Ves22] M. Veshchezerova.
Quantum algorithms for energy management optimization problems.
Theses, Université de Lorraine, page 13, 2022.

[VML+19] G. Verdon, T. McCourt, E. Luzhnica, V. Singh, S. Leichenauer, and J. Hidary.
Quantum graph neural networks.
arXiv preprint arXiv:1909.12264, 2019.

[VSKB10] S. Vishwanathan, N. N. Schraudolph, R. Kondor, and K. M. Borgwardt.
Graph kernels.
Journal of Machine Learning Research, 11(40):1201–1242, 2010.
URL http://jmlr.org/papers/v11/vishwanathan10a.html.

[VSP+17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin.
Attention is all you need.
Advances in neural information processing systems, 30, 2017.

[WHZZ16] Y. Wang, M. Huang, X. Zhu, and L. Zhao.
Attention-based lstm for aspect-level sentiment classification.
In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 606–615, 2016.

[WIWL22] D. Wierichs, J. Izaac, C. Wang, and C. Y.-Y. Lin.
General parameter-shift rules for quantum gradients.
Quantum, 6:677, 2022.

[WL68] B. Weisfeiler and A. Leman.
The reduction of a graph to canonical form and the algebra which appears therein.
nti, Series, 2(9):12–16, 1968.

[WPC+20] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip.
A comprehensive survey on graph neural networks.
IEEE transactions on neural networks and learning systems, 32(1):4–24, 2020.

[WRF+17] Z. Wu, B. Ramsundar, E. N. Feinberg, J. Gomes, C. Geniesse, A. S. Pappu, K. Leswing, and V. Pande.
Moleculenet: A benchmark for molecular machine learning, 2017.
doi:10.48550/ARXIV.1703.00564.

[WS80] H. Wolkowicz and G. P. Styan.
Bounds for eigenvalues using traces.
Linear Algebra and its Applications, 29:471–506, 1980.
doi:https://doi.org/10.1016/0024-3795(80)90258-X.
Special Volume Dedicated to Alson S. Householder.

[WZY+19] M. Wang, D. Zheng, Z. Ye, Q. Gan, M. Li, X. Song, J. Zhou, C. Ma, L. Yu, Y. Gai, T. Xiao, T. He, G. Karypis, J. Li, and Z. Zhang.
Deep graph library: A graph-centric, highly-performant package for graph neural networks.
arXiv preprint arXiv:1909.01315, 2019.

[XHLJ18] K. Xu, W. Hu, J. Leskovec, and S. Jegelka.
How powerful are graph neural networks?
arXiv preprint arXiv:1810.00826, 2018.

[XRV17] H. Xiao, K. Rasul, and R. Vollgraf.
Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms.
arXiv preprint arXiv:1708.07747, 2017.

[YCCW23] X. You, S. Chakrabarti, B. Chen, and X. Wu.
Analyzing convergence in quantum neural networks: Deviations from neural tangent kernels.
In *International Conference on Machine Learning*, pages 40199–40224. PMLR, 2023.

[YCL+21] C. Ying, T. Cai, S. Luo, S. Zheng, G. Ke, D. He, Y. Shen, and T.-Y. Liu.
Do transformers really perform badly for graph representation?
Advances in Neural Information Processing Systems, 34:28877–28888, 2021.

[ZAL18] M. Zitnik, M. Agrawal, and J. Leskovec.
Modeling polypharmacy side effects with graph convolutional networks.
Bioinformatics, 34(13):i457–i466, 2018.

[ZJAS21] L. Zhao, W. Jin, L. Akoglu, and N. Shah.
From stars to subgraphs: Uplifting any gnn with local structure awareness.
arXiv preprint arXiv:2110.03753, 2021.

[ZKH+25] Z. Zimborás, B. Koczor, Z. Holmes, E.-M. Borrelli, A. Gilyén, H.-Y. Huang, Z. Cai, A. Acín, L. Aolita, L. Banchi, et al.
Myths around quantum computation before full fault tolerance: What no-go theorems rule out and what they don't.
arXiv preprint arXiv:2501.05694, 2025.

[ZLWH23] B. Zhang, S. Luo, L. Wang, and D. He.
Rethinking the expressive power of gnns via graph biconnectivity.
arXiv preprint arXiv:2301.09505, 2023.

[ZYZ+20] J. Zhu, Y. Yan, L. Zhao, M. Heimann, L. Akoglu, and D. Koutra.
Beyond homophily in graph neural networks: Current limitations and effective designs.
Advances in Neural Information Processing Systems, 33:7793–7804, 2020.