



**Università  
degli Studi  
di Ferrara**

DOCTORAL COURSE IN:  
**PHYSICS**

CYCLE XXXVIII

COORDINATOR Prof. Paolo Lenisa

# **Quantum Graph Neural Networks and $S_n$ -based Quantum Inference for particle tracking in High-Energy Physics**

Scientific-Disciplinary Sector (SDS): FIS/01

**Ph.D. Candidate**

**Matteo Argenton**

**Supervisor**

**Dr. Concezio Bozzi**

**Academic Years 2022/2025**



---

# Quantum Graph Neural Networks and $S_n$ -based Quantum Inference for particle tracking in High-Energy Physics

Matteo Argenton

## Abstract

Experimental High Energy Physics (HEP) at CERN's LHC is entering the High Luminosity (HL-LHC) era where both the scale and combinatorics of an expected collision event pose difficulties to the reach of classical track reconstruction algorithms. In this regime the number of proton-proton interactions per beam crossing is scheduled to increase by a factor of 3-5 in the general purpose experiments (from about 50 to 140-200 primary interactions per collision on average), motivating frontier research in new technologies as an answer to the problem of tracking charged particles in future experiments.

Quantum computing and machine learning methods have proved themselves to be two of the most promising emerging computing technologies of our time. This thesis belongs to the intersection of these two fields. I develop Quantum Machine Learning (QML) methods for HEP across two complementary lines of research: a near-term variational algorithm for track segment classification, and a beyond-NISQ (Noisy Intermediate-Scale Quantum) algorithm that leverages non-Abelian quantum Fourier transform (QFTs) on the Symmetric group  $S_n$  to model as a permutations-based task the identity management that is naturally occurring in multi-object tracking.

The first project presented in this thesis is a hybrid Quantum Graph Neural Network (QGNN) for edge-level tasks. In this context particle trajectories are represented as a graph data structure and a quantum graph neural network is trained to perform classification of candidate track segments. Results on the characterization of the original architecture are presented, with scalability tests for increasing pileup values. The performance and criticalities of this approach are discussed, finally presenting an upgrade of the original QGNN architecture on both the classical and quantum constituents. Insights into various aspects of code development are also provided in different quantum programming frameworks, exploiting modern accelerators as emulators of quantum computing resources.

---

Looking beyond NISQ a novel QML approach is developed to treat tracking-related assignment problems as evolutions of probability distributions over elements of the Symmetric group  $S_n$ . The theorized polylog scaling of the non-Abelian QFT in the group size is exploited to devise a fully quantum algorithm, to solve classically inefficient identity management problems. The group theoretic frame derived for the project is discussed together with the first blueprint of the algorithm. In particular the development and complexity analysis for the algorithm's diffusion subroutine is reported, with an outlook on the studies for the Bayes update step, which is currently under development.

Collectively this thesis contributes to the advancement of QML for HEP both from the empirical NISQ era perspective, with a robust study on QGNNs architectures for high luminosity tracking, and setting the ground for a novel family of algorithms for fault tolerant quantum computing, rooted in the non-Abelian group theoretic properties of  $S_n$  for identity management problems.

# Contents

|          |                                                            |           |
|----------|------------------------------------------------------------|-----------|
| <b>1</b> | <b>Introduction</b>                                        | <b>7</b>  |
| <b>2</b> | <b>Machine Learning</b>                                    | <b>11</b> |
| 2.1      | Data and learning paradigms                                | 12        |
| 2.2      | Training a model and risk minimization                     | 14        |
| 2.2.1    | Gradient-based optimization                                | 18        |
| 2.3      | Neural Networks                                            | 19        |
| 2.3.1    | Backpropagation                                            | 22        |
| 2.3.2    | Training metrics                                           | 23        |
| 2.3.3    | Universality arguments                                     | 25        |
| 2.3.4    | Inductive bias                                             | 26        |
| <b>3</b> | <b>Quantum Computing and Quantum Machine Learning</b>      | <b>29</b> |
| 3.1      | Elements of quantum computing                              | 29        |
| 3.1.1    | Quantum states, operators and qubits                       | 30        |
| 3.2      | Quantum Machine Learning                                   | 42        |
| 3.2.1    | A classification for quantum algorithms                    | 43        |
| 3.2.2    | Data embedding                                             | 45        |
| 3.2.3    | Variational quantum algorithms                             | 46        |
| 3.2.4    | Non-variational quantum algorithms                         | 51        |
| <b>4</b> | <b>Particle tracking</b>                                   | <b>53</b> |
| 4.1      | Standard model physics                                     | 54        |
| 4.2      | Colliders                                                  | 56        |
| 4.2.1    | LHC                                                        | 57        |
| 4.3      | Tracking particles                                         | 59        |
| 4.3.1    | A classical tracking pipeline                              | 61        |
| 4.3.2    | Final considerations on classical tracking pipelines       | 67        |
| <b>5</b> | <b>Quantum Graph Neural Networks for particle tracking</b> | <b>69</b> |
| 5.1      | Scientific motivation                                      | 70        |
| 5.2      | The TrackML dataset                                        | 71        |

|                                                                             |            |
|-----------------------------------------------------------------------------|------------|
| 5.2.1 Dataset cuts                                                          | 72         |
| 5.3 Graphs                                                                  | 72         |
| 5.3.1 Graph operations                                                      | 74         |
| 5.3.2 Graph building from hits point cloud                                  | 76         |
| 5.4 Graph Neural Networks                                                   | 76         |
| 5.4.1 Structure of a GNN for edge classification                            | 78         |
| 5.5 Quantum Graph Neural Networks                                           | 80         |
| 5.5.1 Phase I: development and pile up studies on the original architecture | 81         |
| 5.5.2 Phase II: upgrades to the original QGNN model                         | 86         |
| 5.6 Conclusions and acknowledgments                                         | 90         |
| <b>6 Object tracking with non-abelian quantum Fourier methods</b>           | <b>93</b>  |
| 6.1 Identity management and object tracking                                 | 95         |
| 6.2 Representations of the Symmetric group                                  | 96         |
| 6.2.1 Irreducible representations                                           | 100        |
| 6.2.2 Regular representation                                                | 101        |
| 6.2.3 Irreducible decomposition of the symmetric group                      | 102        |
| 6.3 Group Fourier transform                                                 | 104        |
| 6.3.1 Fourier transform for $\mathbb{Z}_n$                                  | 105        |
| 6.3.2 Fourier transform for finite non-commutative groups                   | 107        |
| 6.4 Classical algorithms for identity management on $S_n$                   | 110        |
| 6.4.1 Noise models                                                          | 111        |
| 6.4.2 Bayes update                                                          | 114        |
| 6.5 Development of the quantum algorithm                                    | 119        |
| 6.5.1 Towards a fully quantum pipeline                                      | 119        |
| 6.5.2 State initialization                                                  | 120        |
| 6.5.3 Quantum diffusion                                                     | 121        |
| 6.5.4 Bayes update                                                          | 126        |
| 6.6 Comments                                                                | 128        |
| <b>7 Conclusions</b>                                                        | <b>129</b> |
| <b>Appendices</b>                                                           | <b>133</b> |
| <b>A Appendix A: Kalman filter for particle tracking</b>                    | <b>135</b> |
| <b>B Appendix B: Further characterization of the QGNN architecture</b>      | <b>139</b> |
| <b>Bibliography</b>                                                         | <b>145</b> |

# 1

## Introduction

Quantum computing and High Energy Physics are research fields that in the past several years increasingly intersected across a wide range of applications, from quantum algorithms for experimental particle physics [1], to quantum simulations of gauge theories [2].

This thesis focuses quantum computing algorithms for a central task in collider experiments: the reconstruction of the trajectories of charged particles, or, in short, tracking.

Track reconstruction, while already a computationally demanding task, is going to become even more critical in future, high luminosity environments. New technologies are already being developed to account for the decisive growth in particle flux that is planned, for the ATLAS and CMS experiments at LHC, in the upcoming High Luminosity upgrade [3], scheduled for the years between 2026 and 2030. The number of interactions per bunch crossing at these experiments will increase, up to  $\sim 200$ , compared to the current average of  $\sim 60$ . This substantial increase in complexity makes fast, accurate tracking a key bottleneck for future physics programmes, and a natural target for novel algorithms.

Among these novel approaches, and not restricted to tracking, machine learning (ML) and quantum computing are two fields that have emerged in the last decade as two very active and rapidly developing directions of research. Machine learning, and in particular deep learning, have their roots in ideas developed mainly in the

second half of the 20th century, but only recently they have become pervasive, thanks to algorithmic innovations and the ever increasing computing power, due to the widespread availability of specialized hardware. In HEP, ML methods are now standard tools at a broad variety of different stages in data acquisition, processing, and analysis.

Quantum computing, by contrast, is a field that compared to ML is currently in an earlier development phase, and is such a promising and uniquely new computing framework that it is receiving a lot of attention from the research community.

This attention is motivated by rigorously established asymptotic advantages for certain problems [4][5][6][7][8][9], but quantum computing is at the same time a complex and ever-evolving discipline. The limited specifics and availability of current quantum hardware place quantum computing in its Noisy Intermediate-Scale Quantum (NISQ) era, where even the quantum algorithms that already have provable complexity advantage over their best classical counterparts are often difficult to be applied to real world datasets.

At the intersection of these two domains, quantum machine learning (QML) [10][11] has emerged as a new and exciting hybrid field. QML provides an environment that combines classical and quantum processing, and offers a conceptual bridge between NISQ algorithms, and methods that are instead projected to fault tolerant quantum computing.

The different research paths presented in this thesis aim to expand our knowledge of QML, with a strong emphasis on concrete applications to experimental HEP, and in particular to tracking.

Within QML, one of the most widely studied classes of methods are variational quantum algorithms (VQAs) [12]. These algorithms, although not necessarily limited to NISQ, are often capable of targeting complex applications on NISQ devices through the integration of classical and quantum routines.

VQAs have quickly become the perfect entry points for applications in a variety of research fields outside purely quantum computing studies. Notably, in the past few years HEP has taken great interest in this class of algorithms and many studies have been carried out employing HEP datasets and tasks [13][14][15][16].

The first part of the research work presented in this thesis belongs to this line of research. Chapter [5] is dedicated to the development, characterization and upgrades of Quantum Graph Neural Network (QGNN) architectures for charged particle tracking on a simulated high luminosity dataset.

This research stems from the intersection of classical GNNs studies for HEP [17][18][19] with Quantum Neural Networks (QNNs) [20]. The results of this work have been presented at the main conferences of machine learning, HEP and quantum

---

computing, among which the European AI for Fundamental Physics Conference (EuCAIFCon) in 2024, the International Conference on High Energy Physics (ICHEP) in 2024, the Conference on Computing in High Energy and Nuclear Physics (CHEP) in 2024, and the International Conference on Quantum Technologies for High-Energy Physics (QT4HEP) 2025. This work has been published in the form of proceedings for ICHEP24 [21] and CHEP24 [22].

QML is a relatively young and rapidly evolving field. The landscape of VQAs has changed radically during the last few years. It has become increasingly clear that this class of algorithms, while promising, poses some unexpected difficulties. In particular one of the most debated arguments of these last years has been about trainability and simulability of parametric quantum circuits [23].

Together with new research in the field of VQAs, the QML community is also steering towards the study of non variational algorithms that can learn from data, exploiting some known quantum advantage [24][25], even at the cost of looking beyond NISQ applications, towards more theoretical studies.

The second project presented in this thesis in Chapter 6 belongs to this research direction, and is more long-term in character. It was initiated in the final year of the PhD, and is expected to continue beyond the timescale of this thesis.

The project is developed inside the framework of group representation theory for non abelian groups [26], specifically for the permutation group  $S_n$ , and it represents a novel effort to exploit the privileged relationship between quantum computers and Fourier transforms [27][9].

The goal of this research is twofold: the development of a new quantum computing framework, tailored to  $S_n$ , but inherently generalizable to other compact non-abelian groups. Because the core application behind this formalism revolves around object identity management [28][29], the building of a foundation to specifically study enhancements to tracking algorithms.

Together, these two projects, one variational and NISQ-oriented, and the second aimed at fault tolerant quantum computing, illustrate two complementary directions of QML for HEP. They highlight both the opportunities and the challenges in applying quantum methods to realistic tracking problems, and they aim to contribute to a clearer understanding of how quantum technologies can support the physics of HL-LHC and beyond.

The thesis is organized as follows: Chapter 2 provides an overview on classical machine learning, with a focus on neural network architectures. A review of quantum computing, quantum computing algorithms, and quantum machine learning is given in Chapter 3. Chapter 4 presents the overarching context that the research in this thesis is applied to, namely high energy physics, and specifically particle

track reconstruction, with the description of a classical reconstruction pipeline. Chapters [5](#) and [6](#) are the core of this thesis, and the two original research lines of my work. In particular, Chapter [5](#) details the development, characterization, and upgrade of a NISQ-era quantum graph neural network architecture for particle tracking. Chapter [6](#) shifts the focus to the exploration of a novel class of algorithms for identity management, implementing the formalism of group representation theory for the symmetric group  $S_n$  in a quantum algorithm designed for future fault-tolerant devices. The conclusion and outlook are reported in Chapter [7](#).

# 2

## Machine Learning

### Contents

|                                                   |           |
|---------------------------------------------------|-----------|
| <b>2.1 Data and learning paradigms</b>            | <b>12</b> |
| <b>2.2 Training a model and risk minimization</b> | <b>14</b> |
| 2.2.1 Gradient-based optimization                 | 18        |
| <b>2.3 Neural Networks</b>                        | <b>19</b> |
| 2.3.1 Backpropagation                             | 22        |
| 2.3.2 Training metrics                            | 23        |
| 2.3.3 Universality arguments                      | 25        |
| 2.3.4 Inductive bias                              | 26        |

Machine learning (ML) [30] [31] is the research field that explores how computers can use data and prior assumptions to perform inference. This process typically involves a learning phase, in which models are fitted to a task, although this is not strictly necessary. Inference-only models fall into this second scenario.

Machine learning is traditionally a classical research field in the sense that, although Chapter 3 details its growing relevance in the context of quantum computing, it originally emerged at the intersection of classical computer science and statistics, without any initial connection to quantum technologies.

Within this thesis machine learning constitutes one of the overarching frameworks underpinning both the research on quantum graph neural networks in Chapter 5,

and the studies on novel algorithms for inference over symmetric group elements in Chapter 6.

What follows serves as a foundation for the discussion of quantum machine learning in Chapter 3. Its primary goal is to provide an introduction to the key concepts necessary to present quantum neural networks (QNN) in Section 3.2. The organization is as follows: Section 2.1 is dedicated to the concept of learning from a dataset, and outlines the main learning paradigms; Section 2.2 is an introduction to the concept of risk minimization, which is used to present the main vocabulary, methods, and routines used in supervised learning. Section 2.3 shifts the focus to deep learning, revisiting the concepts previously introduced for ML and specializing them to deep neural networks.

## 2.1 Data and learning paradigms

Data are at the core of machine learning. In the context of machine learning, data can generally be categorized as structured or unstructured. Structured data are organized in a predefined format, such as tables with rows and columns, making it easy to process and analyze them. Unstructured data, on the other hand, lack a fixed format and includes text, images, audio, and video, which require more complex methods for extraction and analysis. An intermediate category, semi-structured data, includes elements of both, such as JSON or XML files, where some organizational structure exists but not as rigid as in traditional databases. The atomic data unit is a data point, and a collection of data points constitutes a dataset, denoted as  $\mathcal{D}$ .

The structure of the dataset depends on factors like presence or absence of a learning phase, and, within the learning framework, the specific type of learning being applied.

For models that include a learning phase, which are the ones implicitly considered in this chapter, the dataset is often divided into subsets. This separation is motivated by the principle of *generalization*: the ability of a model to produce meaningful predictions on previously unseen data. Generalization lies at the foundation of the majority of machine learning methods, and among them neural networks represent a particularly notable example.

To achieve generalization datasets are often split into two or three subsets. In the former case there is a distinction between the *training set* used to learn, and the *test set* used after the learning phase to benchmark the trained model. In addition, a third subset called *validation set* is often used within the training phase to tune the model's hyperparameters and improve generalization, detecting

when the model is losing its capacity to generalize on new data and is instead overfitting the data from the training set.

Specifically, the concept of training refers to a stage where the model's free parameters are iteratively tuned. Because of this, machine learning methods can often be recast as optimization problems: a search for global optimal parameters that minimize or maximize some cost function.

There are three broad classes of learning paradigms: supervised, unsupervised, and reinforcement learning, with many intersections and variations.

Supervised learning is the paradigm underlying the hybrid neural network presented in Chapter 5. In supervised learning there is prior knowledge about data input-output relations. Data points come with an output label  $y_i$ :

$$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N, \quad (x_i, y_i) \stackrel{\text{i.i.d.}}{\sim} P(x, y), \quad (2.1)$$

where  $P$  is the distribution over random variables  $x$  and  $y$  from which the data points are sampled. The label is the ground truth for the task; it is used to optimize the model parameters such that the model prediction  $\hat{y}_i$ , given a new  $x_i$ , is as close as possible to the true label  $y_i$ . There are two possible interpretations for this process, one deterministic and the second probabilistic. In the deterministic interpretation the model is learning a function

$$f_\theta : \mathcal{X} \rightarrow \mathcal{Y} \mid \hat{y} = f_\theta(x) \quad (2.2)$$

In the probabilistic picture instead a supervised model solves a task by approximating the underlying conditional distribution  $P_\theta(y|x)$  starting from independent and identically distributed samples  $(x_i, y_i)$  of the stochastic variables.

Unsupervised learning is instead without labels. The model learns an explicit distribution  $P_\theta(X)$ , or samples from an implicit one, starting from a dataset

$$\mathcal{D} = \{x_i\}_{i=1}^N, \quad x_i \stackrel{\text{i.i.d.}}{\sim} P(x). \quad (2.3)$$

These algorithms have very broad applications, and the architectures include Generative Adversarial Networks (GAN) [32], Variational Autoencoders (VAE) [33], but also clustering algorithms like K-means [34], or dimensionality reduction ones like Principal Component Analysis (PCA) [35].

The third paradigm, Reinforcement Learning (RL), differs fundamentally from the two previous classes, and is based on a feedback loop where an agent interacts with the environment. Learning happens through expected cumulative rewards. Rein-

forcement learning methods have proved themselves as very effective in scenarios where adaptivity to changes in the environment is important. As the methods used in this thesis are not in the RL framework the description won't go into further details.

## 2.2 Training a model and risk minimization

This section lays the foundations for introducing neural networks as supervised machine learning models. The  $\boldsymbol{\theta}$  label in Eq. (2.2) represents in reality the vector  $\boldsymbol{\theta} = (\theta_0, \theta_1, \dots, \theta_{p-1})$  of  $p$  free parameters that the model has at its disposal to model the task. In supervised learning these parameters are optimized minimizing a loss function, which represents a distance measure between the model's output at step  $t$  and the optimal solution represented by or computed from the labels  $y_i$ . There are many possible loss functions for a given task, and the simplest choices express the  $\ell_1$  and  $\ell_2$  norms:

$$\mathcal{L}_{\ell_1} = \frac{1}{N} \sum_{i=1}^N |\hat{y}_i - y_i| = \frac{1}{N} \|\hat{\mathbf{y}} - \mathbf{y}\|_1, \quad (2.4)$$

which is also called Mean Absolute Error (MAE). Similarly the Mean Squared Error loss (MSE) is introduced as:

$$\mathcal{L}_{\ell_2} = \frac{1}{N} \sum_{i=1}^N (\hat{y}_i - y_i)^2 = \frac{1}{N} \|\hat{\mathbf{y}} - \mathbf{y}\|_2^2. \quad (2.5)$$

While both MAE and MSE are great options for regression tasks, neither of them is generally a good choice to train neural network models like the one in Chapter 5. The reason for this is related to an increased risk of stationary training dynamics, resulting from the interplay of these simple losses with both the model's output or intermediate results, and the optimization routine. A solution to this behavior in the context of binary classification within deep learning is presented in Section 2.3.

The theoretical framework guiding and giving an interpretation to learning is not unique, and similar results can for example be obtained both from the principle of Empirical Risk Minimization (ERM) [36], and from Maximum Likelihood Estimation (MLE) [31]. The two have important overlaps, and usually a theory for supervised learning can be introduced conveniently starting from both ERM or MLE, while MLE is the preferred choice for unsupervised learning.

For simplicity, to avoid the sometimes heavier formalism of MLE, the following discussion for supervised machine learning belongs to the ERM framework.

In risk minimization theory, learning is the process that explores the parameters space, searching for the configuration

$$\boldsymbol{\theta}^* = \arg \min_{\boldsymbol{\theta}} \mathcal{R}(f_{\boldsymbol{\theta}}). \quad (2.6)$$

that minimizes the *risk*  $\mathcal{R}$ .

The risk is the expectation value of the chosen loss function  $\mathcal{L}$  on the data distribution  $P(x, y)$ , over the whole domain  $\mathcal{X} \times \mathcal{Y}$ , formally:

$$\mathcal{R}_{f_{\boldsymbol{\theta}}} = \mathbb{E}[\mathcal{L}_{f_{\boldsymbol{\theta}}}] = \int_{\mathcal{X} \times \mathcal{Y}} \mathcal{L}(f_{\boldsymbol{\theta}}, y) P(x, y) dx dy \quad (2.7)$$

where  $f_{\boldsymbol{\theta}}(x)$  is the model being learned.

Computing this integral is generally infeasible due to the dimension of the domain and the fact that  $P$  is not necessarily known. Because of this the more realistic *empirical risk*  $\hat{\mathcal{R}}_{f_{\boldsymbol{\theta}}}$  is introduced as:

$$\hat{\mathcal{R}}_{f_{\boldsymbol{\theta}}} = \hat{\mathbb{E}}[\mathcal{L}_{f_{\boldsymbol{\theta}}}] = \frac{1}{N} \sum_{i=1}^N \mathcal{L}(f_{\boldsymbol{\theta}}(x_i), y_i) \quad (2.8)$$

which is computed from the samples of  $P$  used as dataset instead of the whole space. The framework where empirical risk is minimized instead of risk as in Eq. (2.6) is ERM theory. The intuitive meaning of empirical risk minimization is that the value of the loss function over the training dataset has to be minimized to learn a model that, minimizing the risk, is potentially able to fit the training dataset. However, this mechanism would in most algorithms lead to *overfitting*.

Overfitting is the scenario when the model minimizes the risk on the training dataset, but, as this happens, the generalization error –the expected error on new, unseen, data– on the test (and validation) set increases. This phenomenon is strictly connected to another important concept that is the model’s *capacity*, which is related to the space of functions that a model can explore, often called *hypothesis space*. Other factors influencing generalization are tied to the optimization algorithm used in learning and the dataset size.

It is worth mentioning that, within this classical picture, for a long time it has been believed that necessarily as the model capacity increases, and learning makes the training risk decrease, the error on the test set necessarily first decreases and then increases again. This however has been shown to not be generally true, first from empirical observations, and then with rigorous characterizations [37] [38]. In fact many evidences of double descent started to emerge as soon as the computing re-

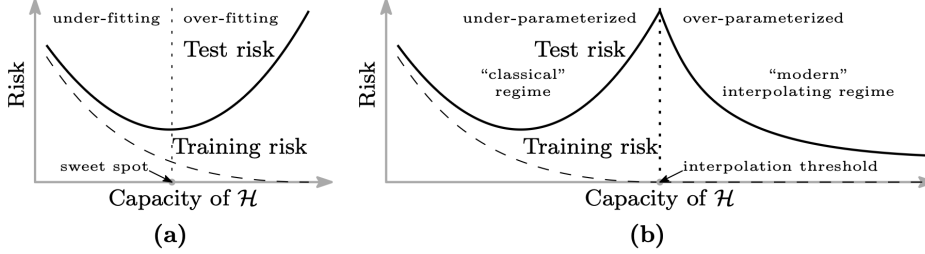


Figure 2.1: Representation of the classical and modern regimes for risk minimization, image from [37]. (a) Classical U-shaped risk curve: defining model capacity as the number of parameters needed to specify a function within the chosen model class  $\mathcal{H}$ , e.g. a fixed neural network architecture, when the number of parameters grows, the expected behavior is for the training risk to get minimized, eventually entering an overfitting regime, and for the test risk to first decrease and then rise again. (b) Modern overparametrized models show that it is possible to enter an interpolation regime, where test risk decreases again; this is called double-descent behavior. Analogous double-descent behavior has been observed as a function of dataset size and number of training epochs.

sources, especially for deep learning, greatly increased. For some overparametrized models, equipped with just loose techniques to reduce overfitting, the observed effect is a second descent of the test error as capacity grows, once an interpolation regime is reached. This behavior serves as an important reminder of the many aspects of learning theory that are still open, but it will not be commented further as it does not influence the discussion of Chapter 5. In this thesis, in fact, models are prevented from going into interpolation regime through regularization techniques.

There are many regularization methods. The two most relevant for this introduction are (i) validation-based early stopping, and (ii) customization of the loss function with regularizer terms.

In the first case (Fig. 2.2) a validation set is used to execute the following routine [39]:

- During training, once in a while evaluate the loss function on the validation set.
- If  $\mathcal{L}_{\text{validation}}$  at present evaluation is higher than the value it had at the last checkpoint, stop the learning. Otherwise continue.
- If the training has been stopped use as model's parameters the ones of that previous checkpoint.

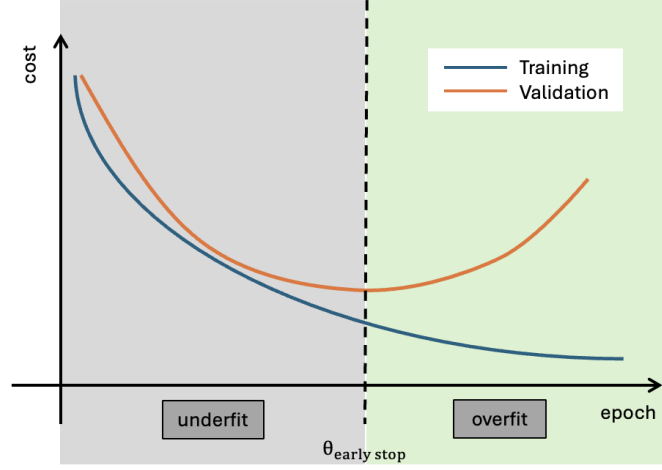


Figure 2.2: Depiction of underfitting, overfitting and early stop regularization strategy. With early stop the model's weights update is stopped at  $\theta_{\text{early stop}}$  in correspondence of the inversion in the slope of the cost function for the validation set, while the model would overfit the training set otherwise.

This has the consequence of heuristically stopping the overfit of the training parameters by checking how the trained model performs on a set of unknown data.

For case (ii) a preference in the choice of the parameters, and hence of the functions the model preferably learns, is enforced by the addition of extra-terms in the loss function. The role of these terms is that of penalties, and the effect is to reduce the generalization error, at the cost of increasing the training error for unwanted configurations [31]. Simple examples are weight decay terms like:

$$C(\theta) = \hat{\mathcal{R}}_{f_\theta} + \lambda \theta^T \theta \quad (2.9)$$

in this above expression  $\lambda$  is a hyperparameter, and  $C(\theta)$  is a global cost function<sup>1</sup> with both the risk term and an  $l_2$ -norm term, enforcing the weights vector  $\theta$  to have the smallest possible norm. For these reasons it is possible to recast Eq. (2.6) in the more general form

$$\theta^* = \arg \min_{\theta} C(\theta). \quad (2.10)$$

which will be the one adopted from now on, with the flexibility of possibly accounting for a regularizing term.

<sup>1</sup>the notation often adopts the letter  $C$  instead of  $\mathcal{L}$  since this global function is not strictly just a loss function.

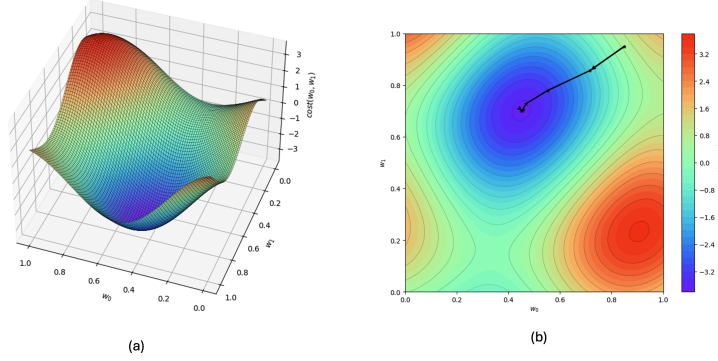


Figure 2.3: Illustration of a generic two-variables cost function landscape and example of gradient descent. (a) 3-D representation of the cost function; (b) projected 2-D contour plot of the surface, the black line represents a gradient descent path.

### 2.2.1 Gradient-based optimization

The discussion so far aimed to build a framework where learning is a well-defined risk minimization process, but nothing has yet been discussed about the parameter optimization process. Optimization is a vast field, and for the purposes of this thesis the focus is just on the small subset relevant to training modern models.

Often the cost function  $C(\boldsymbol{\theta})$  is highly non-convex. This intuitively means that it presents local critical points (e.g. local minima, or saddle points). For this reason finding a solution to the optimization problem of Eq.(2.10) is hard, and in fact it is not uncommon to settle for good enough solutions instead of provably global optima. This is especially true for deep learning where the number of parameters is very high.

A standard foundation for optimization in many machine learning models is *gradient descent* (Fig.2.3):

$$\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) - \eta \nabla C(\boldsymbol{\theta}(t)). \quad (2.11)$$

Here the parameters at step  $t+1$  are computed from the ones at step  $t$  following the direction of maximum change of the cost function  $C(\boldsymbol{\theta})$ , given by its gradient relative to the free parameters  $\boldsymbol{\theta}$ . Since the gradient defines the steepest ascent, the actual direction to minimize the cost is the opposite one.

The hyperparameter  $\eta$  is often called *learning rate* and defines how big the update in the parameters will be. A large learning rate results in wide jumps in the parameter space, which can be useful to escape local minima, while a small  $\eta$  is

usually desired for fine tuning in the final epochs of the training phase.

Full-batch gradient descent is however a computationally expensive algorithm since it requires the evaluation of the gradient of the cost function on the full dataset. For this reason a much more widely spread algorithm is Stochastic Gradient Descent (SGD), where, to obtain the gradient, instead of computing:

$$\nabla_{\theta} C(\theta) = \frac{1}{N} \sum_{i=1}^N \nabla_{\theta} \mathcal{L}(x_i, y_i, \theta) \quad (2.12)$$

on the full training dataset with  $N$  data points, the estimate for the gradient  $\mathbf{g}$  considers just a stochastic subset of points:

$$\mathbf{g} = \frac{1}{N'} \sum_{i=1}^{N'} \nabla_{\theta} \mathcal{L}(x_i, y_i, \theta). \quad (2.13)$$

This method results in important benefits in terms of scalability, and, thanks to the stochastic noise of SGD, also in some form of resistance to plateaus and unwanted minima in the loss landscape.

Modern optimizers add improvements to SGD. A widely used choice is Adam [40], which is also the one used in Chapter 5. Adam implements an adaptive learning rate, which changes during training, together with a technique called momentum, which accounts for previous descent steps and conditions on them the next direction at timestep  $t$ .

## 2.3 Neural Networks

The machine learning models that arguably had the biggest impact in algorithmic research over the past couple of decades are Neural Networks (NN). In the context of this thesis this term will always be a synonym for *feedforward neural networks*, i.e. NNs without feedback loops (in contrast to, e.g. recurrent neural networks [41] where the network's output is being fed back to the network itself<sup>2</sup>).

The foundation of neural networks dates back to both linear models and the first studies on perceptrons.

Linear models are of the form:

$$f(\mathbf{x}; \mathbf{w}, b) = \mathbf{x}^{\top} \mathbf{w} + b \quad (2.14)$$

---

<sup>2</sup>Notice that in this context feeding the same data at different stages of the forward pass of the network, like in ResNet [42], is still within the feedforward NN framework, because, by themselves, residual connections do not introduce cycles

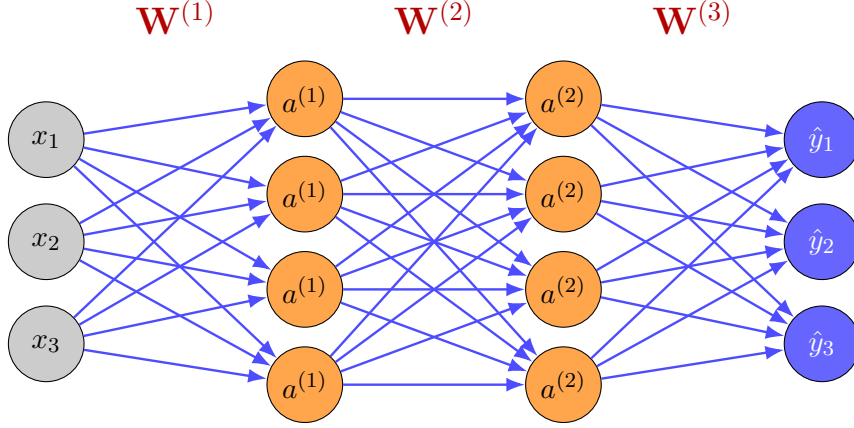


Figure 2.4: Graphical depiction of a fully connected neural network, or multi-layer perceptron, with 3 input nodes  $x_i$ , 2 hidden layers with node-wise activation functions  $a^{(\ell)}$ , and 3 output nodes  $\hat{y}_i$ . Each blue arrow is associated with a weight  $w_{ij}^{(\ell)}$ .

here the model's free parameters  $\theta$  are represented by both  $\mathbf{w}$  and  $b$ , called the weights and bias, respectively. Linear models can be optimized minimizing for example the MSE loss function:

$$\mathcal{L}(\theta) \approx \sum_{\mathbf{x}_{\text{train}}} (f^*(\mathbf{x}) - f(\mathbf{x}; \theta))^2 \quad (2.15)$$

with  $f^*(\mathbf{x})$  the target function to be learned. In practice the target function is unknown and the minimization is done through prior knowledge of the labels of the training samples. It is intuitive that linear models, even when composed with other linear models will just be able to represent linear functions, but linear models are nonetheless the general building blocks of more sophisticated algorithms.

Taking inspiration from linear models and from biological sciences -specifically from the studies of neurons- in the late 1950s Rosenblatt introduced the idea of perceptron[43], as a statistical, adaptive classifier. This is the true progenitor of modern neural networks (also referred to as multi-layer perceptrons). A perceptron is a binary classifier with a linear decision boundary obtained through a step function, hence still very limited in the representational capacity. A historically very notable example of this is the XOR boolean function, which cannot be learned by a single-layer perceptron[44].

A feedforward NN, as in Fig 2.4 stacks affine maps with element-wise nonlinearities  $a^{(\ell)} : \mathbb{R}^{m_\ell} \rightarrow \mathbb{R}^{m_\ell}$ , with  $m_\ell$  the number of neurons of the  $\ell$ -th layer.

Let:

$$\begin{aligned}\Theta &= \{(W^{(\ell)}, b^{(\ell)})\}_{\ell=1}^L \\ W^{(\ell)} &\in \mathbb{R}^{m_\ell \times m_{\ell-1}} \\ b^{(\ell)} &\in \mathbb{R}^{m_\ell}.\end{aligned}\tag{2.16}$$

Then

$$\Phi(x; \Theta) = a^{(L)}\left(W^{(L)} a^{(L-1)}\left(W^{(L-1)} \dots a^{(1)}(W^{(1)}x + b^{(1)}) \dots + b^{(L-1)}\right) + b^{(L)}\right).\tag{2.17}$$

is the function realized by the neural network for input  $x$  and free parameters  $\Theta$ , where  $W^{(\ell)}$  is the matrix of the weights of the layer  $\ell$ , and  $b^{(\ell)}$  is the bias vector for the layer.

The introduction of stacked nonlinearities has the effect of turning the loss landscape of a perceptron in much more complex, non-convex surface. This is in turn the reason why (i) neural networks are almost always trained by iterative, gradient-based optimizers -like SGD- generally resulting in a non global-minimum convergence, but heuristically reaching a close-to-optimum stationary point solution nonetheless; and (ii) the success of the optimization strategy might depend on the choice of the initial value for the free parameters. Altogether this means that in neural networks there aren't in general strong guarantees of convergence [31], and, apart from specific regimes (e.g. overparametrized networks and NTK theory [45]), tests on different datasets, with a variety of hyperparameter sets, are required to empirically characterize performance.

The hierarchical structure of Eq. (2.17) divides the neural network's neurons into three classes (at least for a standard MLP):

- **Input units:** neurons that encode the input to the network.
- **Hidden units:** the sequence of layers placed between input and output, where most of the network's representational capacity resides.
- **Output units:** the final layer of neurons of a neural network.

Each of these categories is associated with the design choice of the per-layer activation function  $a^{(\ell)}$ . Nowadays a general-purpose choice for the hidden units is often *Rectified Linear Unit (ReLU)*:

$$a(z) = \max\{0, z\}.\tag{2.18}$$

Activations for output units are much more task-dependent (e.g. a network used for binary classification needs a different output than a model for multiclass classification). The choice is, for this reason, closely tied to the loss function.

With the specific study case of a binary classification network like the one in Chapter 5, the task is the prediction of a Bernoulli-distributed binary variable  $y$  given  $\mathbf{x}$ .

For this kind of task the common loss function is Binary Cross Entropy (BCE), also called log-loss:

$$\mathcal{L}_{BCE}(z_i, y_i) = -y_i \log \sigma(z_i) - (1 - y_i) \log \sigma(-z_i) \quad (2.19)$$

where

$$\sigma(k) = \frac{1}{1 + e^{-k}} \quad (2.20)$$

is the nonlinear activation function, which is commonly applied to the output of the very last network layer to map  $z_i \in \mathbb{R}$  into probabilities  $\hat{y}_i \in (0, 1)$ .

Binary classification can in principle be trained also with the MSE or MAE loss functions of the previous section, but the overall performance can often be sub-optimal as gradients can vanish near 0 or 1. BCE corresponds to the negative log-likelihood of a Bernoulli distribution, and maintains more informative gradients in those regimes.

### 2.3.1 Backpropagation

The concept of optimization of the model's free parameters has already been introduced in Section 2.2.1, referring specifically to gradient-based optimization, which represents the standard framework by which neural networks are trained.

Learning in a neural network context comprises two phases, the forward and backward passes. The first is happening when data points are being fed to the network and are flowing through the model. Each stacked layer produces a manipulation of the output of the previous one, according to their weights and bias (held fixed during forward pass). During this step, the values of the neuron-wise output from the different activation functions encountered through the computation graph representing the network are stored in a cache for the backward pass. This scales as  $O\left(B \sum_{\ell=1}^L m_{\ell}\right)$ , where the first value refers to the number of inputs in a batch of forward passes, and the second is the total number of activations. Finally, with the output of the last network's layer the scalar value of the loss function gets computed (for each data point, or stacking many together). This value represents how much the model's output differs from some prior reference, as in Eq. (2.8).

Backward pass then starts and is the phase where the gradient of the loss function with respect to the model's free parameters  $\nabla_{\theta} \mathcal{L}(\theta)$  is computed. These values (one for each  $\theta$ ) are then fed to the chosen optimization algorithm (e.g. SGD) to

update both weights and biases for the next forward pass.

Computing the gradients requires in principle the evaluation of complex expressions, and while this is true for a real training, there are two important simplifications, namely the chain rule for the derivatives and the aforementioned cached values for the activations of the intermediate layers.

Chain rule refers to the well known property of derivatives to be splittable into factors, each representing a subset of the neural network. For example, for a 2-layer MLP [\[46\]](#)

$$z^{(1)} = W^{(1)}x + b^{(1)}, \quad a^{(1)} = g(z^{(1)}), \quad z^{(2)} = W^{(2)}a^{(1)} + b^{(2)}, \quad L = L(z^{(2)}, y), \quad (2.21)$$

the gradients split into factors:

$$\frac{\partial \mathcal{L}}{\partial W^{(2)}} = \underbrace{\frac{\partial \mathcal{L}}{\partial z^{(2)}}}_{\text{loss term}} \underbrace{\frac{\partial z^{(2)}}{\partial W^{(2)}}}_{\text{linear layer}} = \delta_2 a^{(1)\top}, \quad (2.22)$$

$$\frac{\partial \mathcal{L}}{\partial W^{(1)}} = \underbrace{\frac{\partial \mathcal{L}}{\partial z^{(1)}}}_{\text{backprop error}} \underbrace{\frac{\partial z^{(1)}}{\partial W^{(1)}}}_{\text{linear layer}} = \delta_1 x^\top, \quad (2.23)$$

with the intermediate error factoring as

$$\delta_1 = \underbrace{\frac{\partial \mathcal{L}}{\partial z^{(2)}}}_{\delta_2} \underbrace{\frac{\partial z^{(2)}}{\partial a^{(1)}}}_{W^{(2)\top}} \odot \underbrace{\frac{\partial a^{(1)}}{\partial z^{(1)}}}_{g'(z^{(1)})} = (W^{(2)\top} \delta_2) \odot g'(z^{(1)}), \quad (2.24)$$

where the two terms are multiplied element-wise by the Hadamard product  $\odot$ , coherently with the node-wise action of the activation function  $g(z^{(1)})$ .

This example illustrates how backward computations are nested, and results from Eq. [\(2.23\)](#) are used in Eq. [\(2.22\)](#). This property is what makes backpropagation both efficient as a stand-alone gradient computation algorithm, and crucial for parameters update when used with stochastic gradient descent.

### 2.3.2 Training metrics

In supervised learning, and in particular in classification tasks like the one in Chapter [\[5\]](#), it is essential to quantify how well a model performs on a given task. Numerical performance metrics provide a way to (i) objectively compare different models or architectures, (ii) guide model selection, and (iii) assess whether a trained model meets the requirements of a specific application. A single scalar

|              |          | Predicted value |          |
|--------------|----------|-----------------|----------|
|              |          | Positive        | Negative |
| Actual value | Positive | TP              | FN       |
|              | Negative | FP              | TN       |

Figure 2.5: Confusion matrix for binary classification. The four classes: TP, TN, FP, and FN correspond to the possible four combinations of real label and classified label.

score rarely captures all aspects of performance.

In a binary classification setting, each prediction can be categorized as a true positive (TP), true negative (TN), false positive (FP), or false negative (FN), depending on whether the predicted label matches the ground truth label, and whether the instance is actually positive or negative. These four quantities form the confusion matrix and constitute the basis for a number of commonly used performance metrics, defined as follows.

**Accuracy** Accuracy measures the proportion of correctly classified instances among all instances. It is defined as

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}. \quad (2.25)$$

This metric provides a single measure of performance but can be misleading in the presence of strong class imbalance, where a trivial classifier that always predicts the majority class may achieve high accuracy while performing poorly on the minority class.

**Specificity** Specificity quantifies how well the classifier identifies negative instances. It is defined as

$$\text{Specificity} = \frac{TN}{TN + FP}. \quad (2.26)$$

A high specificity indicates that few negative examples are incorrectly classified as positive (i.e., the false positive rate is low). This metric is particularly important in scenarios where false positives are costly, such as certain medical screening or anomaly detection tasks.

**Precision** Precision measures the reliability of positive predictions, i.e., the proportion of instances predicted as positive that are actually positive:

$$\text{Precision} = \frac{TP}{TP + FP}. \quad (2.27)$$

High precision means that when the classifier predicts the positive class, it is usually correct. This metric is especially relevant when the cost of a false positive is high, and one wishes to ensure that declared detections are accurate.

**Recall** Recall assesses how well the classifier recovers the positive instances, i.e., the proportion of actual positives that are correctly identified:

$$\text{Recall} = \frac{TP}{TP + FN}. \quad (2.28)$$

High recall indicates that few positive instances are missed (low false negative rate). This metric is crucial in applications where failing to detect a positive instance is particularly undesirable. Precision and recall are often considered jointly, since increasing one may decrease the other, depending on the decision threshold used for classification.

### 2.3.3 Universality arguments

One of the most important statements for deep learning is provided by the *Universal approximation theorem*. This not only illustrates the capabilities of neural networks, but is often also linked with similar concepts for quantum neural networks.

The universal approximation theorem [47][48] in its original formulation states that with at least a single hidden layer and a squashing function (like a sigmoid) a neural network can approximate any Borel function from a subset of  $\mathbb{R}^n$  to another, which in general includes both continuous functions and functions on discrete but finite dimensional spaces [31]. This result is then extended to include activation functions such as ReLU in modern derivations [49].

This theory is of crucial importance as a guarantee of expressivity of a neural network: the theorem states that a wide (possibly very large) shallow network

can approximate, up to arbitrary accuracy, any function of interest for machine learning.

At the same time, even if wide networks have a central role in studies on learning dynamics, they are not usually reliably trainable. In fact, the universality theorem is about expressivity and not trainability: a neural network might be able to approximate a certain function, but the optimization algorithm is not guaranteed to be able to reach a parameter configuration resulting in that function, starting from a random initial state. Moreover there is no general strict guarantee about the number of hidden neurons that such a network could need in the worst case scenario, which can be shown to grow exponentially with the number of input features.

To counteract such a scenario a depth-width argument is often introduced. Up to now the concept of universality here discussed has been the original one on wide shallow networks, but learning theory has also been extended to limited-width deep neural networks [50] [51]. Even if enhancing depth while reducing width is not a simple trade-off, in the sense that keeping similar numbers of free parameters does not guarantee similar results either in expressivity or convergence to solution for the two networks, it is often convenient to work with deep neural networks. An argument for this approach comes from [52], where a deep neural network with ReLU activations is shown to partition the input space in subregions where the network maps the inputs to the output function values with an affine transformation. These sub-regions are defined by stacking the hyperplane boundaries coming from the composition of different ReLUs activations. The result is an exponentially expressive tessellation.

### 2.3.4 Inductive bias

Underlying many of the concepts of the previous sections there is the notion of inductive bias, which refers to the set of prior beliefs about a dataset and the task to be learned from it. These assumptions correspond to the knowledge about a problem that is then transferred to the network in terms of design choices at different levels in an architecture.

From the previous section on universality constraints and the depth-width arguments comes the assumption that, if a target function is believed to be a composition of simpler functions, then then favoring depth introduces an inductive bias toward compositional representations. In other words, deeper architectures bias the hypothesis class toward functions expressible as hierarchical compositions, which theory suggests can be represented more efficiently by depth than by width alone.

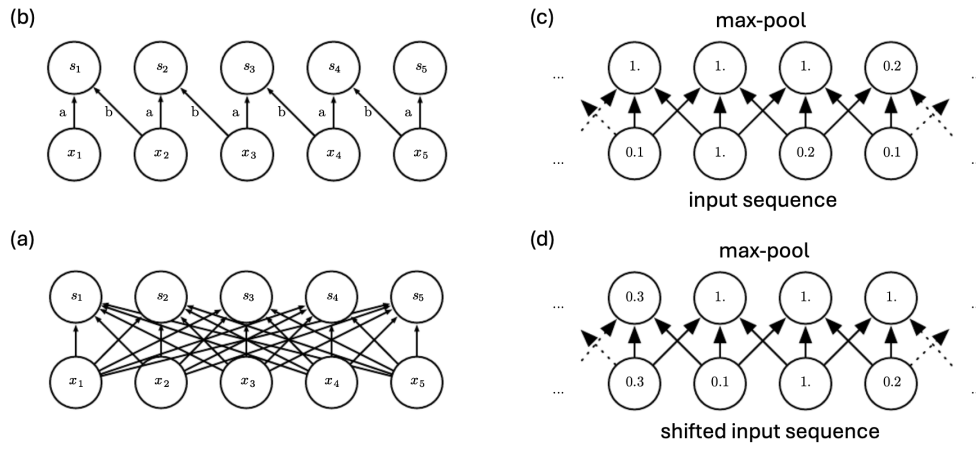


Figure 2.6: Parameter sharing and max-pooling examples from a CNN architecture, images from [31]. (a) A fully connected neural network architecture with no weight sharing. (b) local connectivity with weight sharing: locally connected layer with patch size of two pixels realizing a kernel width of two pixels through parameter sharing. (c) local connectivity with max-pooling operation: only the maximum value of the input is kept at pooling layer. (d) local connectivity with max-pooling on a shifted input: as the input undergoes a translation, and all input values change with respect to (c), the pooling output changes just on half of its values; the sensitivity is not on local changes.

Another important source of bias lies in the connectivity of the network. This thesis specializes in graph neural networks, but in general different tasks benefit from different architectures. MLPs are a quite task-agnostic choice, and the price to pay for this becomes evident in the large number of parameters that have to be optimized compared to more task-oriented architectures.

A notable example are Convolutional Neural Networks (CNNs) [53], which are largely used for image processing. CNNs exhibit two main inductive biases, namely local connectivity and shared weights (Fig 2.6), both of which are strictly related to the prior knowledge about the intrinsic properties of images. Together these biases restrict the hypothesis space explored by a CNN to translation-equivariant functions, resulting in fewer trainable parameters with respect to a full MLP approach, and often to better training and sample efficiency.

Similarly to CNNs, Graph Neural Networks [54] (which can be considered a generalization of CNNs, and to which Chapter 5 is dedicated) exploit specific inductive biases to work with graphs, which are an example of a sparse data structure that benefits from local information aggregation through generalized convolutions.

# 3

## Quantum Computing and Quantum Machine Learning

### Contents

|                                               |           |
|-----------------------------------------------|-----------|
| <b>3.1 Elements of quantum computing</b>      | <b>29</b> |
| 3.1.1 Quantum states, operators and qubits    | 30        |
| <b>3.2 Quantum Machine Learning</b>           | <b>42</b> |
| 3.2.1 A classification for quantum algorithms | 43        |
| 3.2.2 Data embedding                          | 45        |
| 3.2.3 Variational quantum algorithms          | 46        |
| 3.2.4 Non-variational quantum algorithms      | 51        |

### 3.1 Elements of quantum computing

Quantum computing is the paradigm where the information units are quantum systems, and computation is carried out operating on them according to quantum mechanics.

The fundamental difference between quantum and classical computing is the fact that for the digital information unit of the latter, the bit, although still being implemented at subatomic scale and exploiting or controlling the issues coming

from quantum effects, the initial, intermediate and final state of each computation are completely classical, and do not rely on any information-level quantum effect.

To understand how quantum coherence can be used to solve such tasks, this chapter introduces the basics of quantum computing in the limit of infinite measures, from qubits to quantum circuits. Section 3.1.1 begins the discussion with an introduction to quantum mechanics, touching on some of its fundamental concepts: states, operators, unitary evolution and projective measurement. This general formalism is then specialized to quantum computing, with quantum gates and the circuit model.

This will be the foundation for section 3.2, which provides an introduction to quantum machine learning as the main framework where this thesis belongs. The goal of this section is to give an overview of the field, with a discussion on variational and non-variational quantum algorithms.

### 3.1.1 Quantum states, operators and qubits

The basic information-carrying element in quantum computing is the quantum bit: the qubit. A qubit is a two-level quantum system that can be realized starting from different physical systems obeying the laws of quantum mechanics. This section will not delve into the physics of hardware implementations as the context of this work is algorithmic, but some of the most researched implementations include superconducting qubits [55], trapped ions [56], neutral atoms [57], and photonic [58]. Each of these technologies comes with its own advantages and disadvantages.

While having a grasp of the underlying physics is useful to understand the operations that best fit a certain technology, it is not strictly necessary to understand quantum algorithms. What is really needed are the principles of linear algebra and quantum mechanics.

The description of a quantum system is its state. In the commonly used Dirac's notation this is  $|\Psi\rangle$  and it resides in a Hilbert space  $\mathcal{H}$ : a complete complex vector space with a positive-defined inner product. Physical quantities measured on such a space are called observables  $\mathcal{A}$ ; in quantum mechanics physical observables are expressed as Hermitian operators in the Hilbert space:

$$\mathcal{A} = \mathcal{A}^\dagger \tag{3.1}$$

which means that its eigenvalues are real numbers. Let  $|a_i\rangle$  be the eigenvectors of

$\mathcal{A}$ , it is then possible to write the usual secular equation:

$$\mathcal{A} |a_i\rangle = a_i |a_i\rangle \quad (3.2)$$

where  $a_i$  is the  $i$ -th eigenvalue of  $\mathcal{A}$  associated to the eigenket  $|a_i\rangle$ <sup>1</sup>

A generic operator  $\mathcal{A}$  can be written in terms of its eigenvectors and eigenvalues as:

$$\mathcal{A} = \sum_i a_i |a_i\rangle \langle a_i|. \quad (3.3)$$

Eq.(3.3) is the spectral decomposition of operator  $\mathcal{A}$ .

Once a orthonormal basis  $|a\rangle$  is defined on  $\mathcal{H}$ , the state of the system can be in general written as:

$$|\Psi\rangle = \sum_i c_i |a_i\rangle. \quad (3.4)$$

Here the set of  $c_i$  is in general complex-valued and has a few important properties which will keep getting used in the next chapters. First and foremost

$$\sum_i |c_i|^2 = 1 \quad (3.5)$$

which in turn defines how likely it is for the system to collapse to the eigenstate  $|a_i\rangle$  once measured in the basis of operator  $\mathcal{A}$ . Eq.(3.5) implies that the distribution of  $c_i$  defines a probability.

Covectors, or bras in Dirac's notation, are introduced as  $\langle\Psi|$ , defined in the dual space w.r.t. the ones of  $|\Psi\rangle$  and computed as their Hermitian conjugates:

$$\langle\Psi| = \sum_i c_i^* \langle a_i|. \quad (3.6)$$

With vectors and covectors it is possible to write the outer product

$$\Pi_{a_i} = |a_i\rangle \langle a_i|, \quad (3.7)$$

which acts as a projector in the subspace of  $\mathcal{H}$  spanned by  $|a_i\rangle$ . This in turn lets us introduce the completeness relation

$$\mathbb{I} = \sum_i |a_i\rangle \langle a_i| \quad (3.8)$$

---

<sup>1</sup>Notice that the terms vector/ket, eigenvector/eigenket will be used without distinguishing them, since kets are a representation of complex vectors in  $\mathcal{H}$

and, from eq.(3.4) it is possible to recover

$$|\Psi\rangle = \sum_i |a_i\rangle \langle a_i|\Psi\rangle \quad (3.9)$$

where finally it is possible to recognize  $c_i = \langle a_i|\Psi\rangle$ . Here  $\langle a|b\rangle$  is a scalar (inner) product and measures the overlap between two states.

From the fundamental postulate of quantum mechanics, a measurement of operator  $\mathcal{A}$  on the state  $\Psi$  results in one of the eigenvalues of  $\mathcal{A}$  with probability

$$P_i = |\langle a_i|\Psi\rangle|^2 = |c_i|^2. \quad (3.10)$$

After this measurement, the system collapses to the state:

$$|\Psi_i\rangle = \frac{|a_i\rangle \langle a_i|\Psi\rangle}{\sqrt{\langle \Psi|a_i\rangle \langle a_i|\Psi\rangle}}. \quad (3.11)$$

From the formalism presented until now it is possible introduce the concept of expectation value of an operator. When a system is not in an eigenstate  $\bar{a}_i$  of some operator  $\mathcal{A}$ , the measurement will result in some eigenvalue of the operator, but with a certain probability. In this case it is often important estimating the distribution of the outcomes resulting from the repeated sampling of eq.(3.10). A useful statistical quantity is the expectation value of  $\mathcal{A}$  on the state  $|\Psi\rangle$ :

$$\mathbb{E}_{\mathcal{A}} = \langle \Psi|\mathcal{A}|\Psi\rangle \quad (3.12)$$

where the bracket can be estimated from averaging the results of the measurements of  $\mathcal{A}$  on a generic  $|\Psi\rangle$ , in the limit of infinitely many measures:

$$\begin{aligned} \mathbb{E}_{\mathcal{A}} &= \sum_i^d a_i |\langle a_i|\Psi\rangle|^2 \\ &= \sum_i^d a_i \langle \Psi|\Pi_i|\Psi\rangle \\ &= \langle \Psi|\mathcal{A}|\Psi\rangle \\ &= \langle \mathcal{A}\rangle_{\Psi} \\ &\simeq \lim_{N \rightarrow \infty} \frac{1}{N} \sum_{j=1}^N x_j \end{aligned} \quad (3.13)$$

where  $x_j$  are obtained from sampling the distribution.

It is worth highlighting that the formalism introduced up until now is the one for pure states: quantum states where the underlying system is in a definite configuration, not in a statistical mixture. A mixture, often referred to as an *ensemble*, has nothing to do with a coherent superposition of states, and is instead an expression of classical ignorance about which pure state the system has actually been prepared in. To describe this second scenario a complete reference to the density matrix formalism can be found in [59], in this thesis, however, density matrices will not be introduced as they are not directly used in the next chapters.

Some basic concepts related to state evolution are now going to be introduced. From eq. (3.5) the normalization of the state of the global system is constrained to be 1 at each step. This is realized on closed systems by unitary operators. In particular when the generator of the evolution is the physical Hamiltonian of the system and the continuous parameter is time  $t$ , the evolution of a pure quantum state (from now onward "pure" will be omitted and it will be assumed to be working on pure states if not otherwise specified) is expressed through the Schrödinger equation:

$$i\hbar \frac{\partial}{\partial t} |\Psi(t)\rangle = H |\Psi(t)\rangle \quad (3.14)$$

For time independent Hamiltonians it is possible to introduce the unitary action on the quantum state as the operator

$$U(t, t_0) = e^{-i \frac{(t-t_0)H}{\hbar}} \quad (3.15)$$

with

$$|\Psi(t)\rangle = U(t, t_0) |\Psi(t_0)\rangle \quad (3.16)$$

In general the same formalism can be applied for any evolution generated by some hermitian operator and mediated by some continuous parameter  $\theta$ .

Up until this moment, this has been a general introduction to quantum mechanics, now this formalism is specialized to quantum computing. Since a qubit is a 2-level system, its state can be written as:

$$|\Psi\rangle = \alpha |0\rangle + \beta |1\rangle \quad (3.17)$$

where  $|0\rangle$  and  $|1\rangle$  are the eigenvectors of the so called *computational basis*, and form an orthonormal basis of  $\mathbb{C}^2$ . From eq. (3.5) it follows immediately that  $|\alpha|^2 + |\beta|^2 = 1$ , with  $\alpha$  and  $\beta$  in general two complex numbers.

Labeling these two basis elements as 0 and 1 enforces a comparison with the classical bit. The first evident difference between the bit and the qubit is that, for the latter, the two possible basis states can be superimposed before measure-

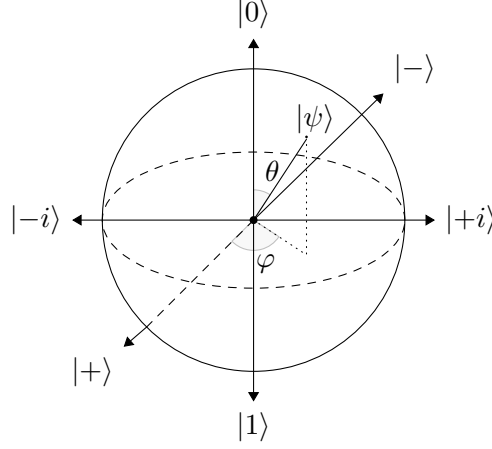


Figure 3.1: Bloch sphere representation of a 1-qubit state parametrized as in eq.(3.18)

ment (or decoherence noise); this is the fundamental building block for quantum algorithms.

A common graphical representation for the state and operations on a qubit is the Bloch sphere (Fig.3.1), which is a unit radius spherical surface where 1-qubit pure states belong. Mixed states would instead belong to the space within the sphere. The most general state parametrization is the following:

$$\gamma = \cos\left(\frac{\theta}{2}\right) |0\rangle + e^{i\phi} \sin\left(\frac{\theta}{2}\right) |1\rangle \quad (3.18)$$

It is possible to extend eq.(3.17) to two and in general multiple qubits. The basis states of a n-qubits system will be labeled as  $\{0, 1\}^{\otimes n}$ , so for example a 2-qubits state will be:

$$|\Psi\rangle = \alpha_0|00\rangle + \alpha_1|01\rangle + \alpha_2|10\rangle + \alpha_3|11\rangle \quad (3.19)$$

and for the general n-qubits case the state will be:

$$|\Psi\rangle = \alpha_0|0 \dots 00\rangle + \alpha_1|0 \dots 01\rangle + \dots + \alpha_{2^n-1}|1 \dots 11\rangle. \quad (3.20)$$

From eqs. (3.19) and (3.20) it is possible to see that the dimension of the associated Hilbert space scales exponentially in the number of qubits, which is a second intrinsic property of quantum states. From this exponential scaling stem both the interest in exploiting it for general purpose algorithms, and also the original idea by Richard Feynman [60] of simulating quantum systems with quantum systems, without paying the classical price of storing and manipulating exponentially big

data structures.

To introduce quantum logic gates it is convenient to work with vector representation of the computational basis. This is possible since the Hilbert space is isomorphic to a complex vector space. Conventionally:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad , \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad (3.21)$$

this in turn lets us write the single qubit state as

$$\alpha|0\rangle + \beta|1\rangle \equiv \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \quad (3.22)$$

and the associated covector as:

$$\alpha^* \langle 0| + \beta^* \langle 1| \equiv (\alpha^*, \beta^*) \quad (3.23)$$

Now that the basic formalism about states has been introduced, the focus shifts to the computation itself.

Performing computation on a quantum state is a matter of applying the right unitary operator, and much about quantum algorithm design consists in determining how to combine ad hoc transformation to extract a meaningful result once the system is measured. This process is most of the times unintuitive and radically different from creating a classical algorithm, and further complicated by the common need for non unitary operations (which can be implemented even in a quantum circuit, wrapping them inside a unitary global action).

The building blocks of a quantum algorithm are quantum logic gates<sup>[61]</sup>: the implementations of unitary operators.

The main one and two qubit gates are introduced as the fundamental building block of any algorithm. Starting from 1-qubit gates, the first, trivial gate is the identity  $\mathbb{1}$  which maps a qubit to itself and does not change its state. More meaningful operators are the X, Y and Z Pauli, associated to matrices

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad (3.24)$$

$$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \quad (3.25)$$

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \quad (3.26)$$

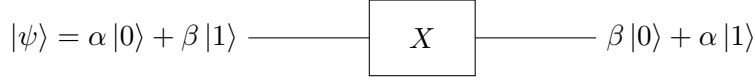


Figure 3.2: Pauli X gate



Figure 3.3: Pauli Z gate

The **Pauli X** (Fig. 3.2) gate is also called the NOT gate and is the quantum analogue to a classical NOT; its action on basis states is:

$$\begin{aligned} X|0\rangle &= |1\rangle \\ X|1\rangle &= |0\rangle \end{aligned} \quad (3.27)$$

**Pauli Z** gate is called bit flip gate because of its action:

$$\begin{aligned} Z|0\rangle &= +1|0\rangle \\ Z|1\rangle &= -1|1\rangle \end{aligned} \quad (3.28)$$

it is also used as a measurement operator as it distinguishes the basis states, which are its eigenkets.

Finally (Fig. 3.4) the **Pauli Y** gate (bit-and-phase flip) acts on the basis states as:

$$\begin{aligned} Y|0\rangle &= i|1\rangle \\ Y|1\rangle &= -i|0\rangle \end{aligned} \quad (3.29)$$

and therefore on a generic qubit:

$$Y(\alpha|0\rangle + \beta|1\rangle) = -i\beta|0\rangle + i\alpha|1\rangle. \quad (3.30)$$

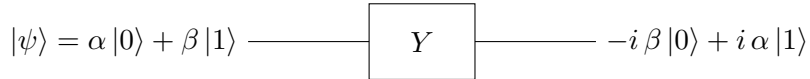


Figure 3.4: Pauli Y gate

Another important 1-qubit operator is the **Hadamard H** (Fig. 3.5), which maps

$$|0\rangle \mapsto \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \quad (3.31)$$

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \longrightarrow \boxed{H} \longrightarrow \frac{\alpha+\beta}{\sqrt{2}}|0\rangle + \frac{\alpha-\beta}{\sqrt{2}}|1\rangle$$

Figure 3.5: Hadamard gate

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \longrightarrow \boxed{R_\phi} \longrightarrow \alpha|0\rangle + e^{i\phi}\beta|1\rangle$$

 Figure 3.6: Phase Shift gate  $R_\phi$ 

and

$$|1\rangle \mapsto \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle). \quad (3.32)$$

This gate is very useful to generate superposition of basis states, and is in fact often the starting point of algorithms where, starting from  $|0\rangle^{\otimes n}$ , all the other  $2^n$  states get superimposed:

$$H^{\otimes n} |0\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_0^{2^n} |x\rangle. \quad (3.33)$$

The Hadamard gate apart from being unitary  $HH^\dagger = H^\dagger H = \mathbb{1}$  is also Hermitian and therefore  $H^2 = \mathbb{1}$ , meaning that a sequential application of two Hadamards results in the action of the identity operator (same holds for the three Pauli gates).

The introduction to single qubit gates ends with a brief mention to the **Phase shift gate**  $R_\phi$  (Fig. 3.6):

$$R_\phi = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{bmatrix} \quad (3.34)$$

which has no classical analogue and introduces a generic phase relation between the two states of the computational basis. The intuitive action of this gate can be understood from the polar (Bloch) equation for the qubit in eq. (3.18):

$$R_\phi|\theta, \Phi\rangle = |\theta, \Phi + \phi\rangle \quad (3.35)$$

It can be proved that  $\{H, R_\phi\}$  form a universal set for the single qubit gates, meaning that any single qubit operation can be built just a string of these two gates [62].

Notice that when  $\phi = \pi$  then  $R_\pi = Z$ , the Z gate is immediately recovered.

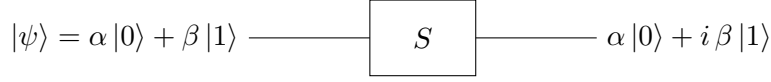


Figure 3.7: Phase gate S

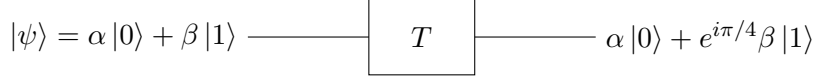


Figure 3.8: T gate

Similarly,  $\phi = \pi/2$  implements the **Phase Gate S** (Fig 3.7):

$$S = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}, \quad (3.36)$$

and  $\phi = \pi/4$  the **T gate** (Fig 3.8):

$$T = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix} \quad (3.37)$$

Finally, from the X, Y and Z gates it is possible to build the corresponding single qubit rotations  $R_x(\theta)$ ,  $R_y(\theta)$  and  $R_z(\theta)$  around the x,y and z axes of the Bloch sphere:

$$R_x(\theta) = e^{-i\frac{\theta}{2}\sigma_x} \quad (3.38)$$

$$R_y(\theta) = e^{-i\frac{\theta}{2}\sigma_y} \quad (3.39)$$

$$R_z(\theta) = e^{-i\frac{\theta}{2}\sigma_z} \quad (3.40)$$

A generic unitary operation on a n qubit system cannot be built with single qubit gates, but there is no need to introduce up to n qubit gates to do so: two qubit gates will suffice [61].

Before delving in these concepts, it is important to enforce a formalism in the ordering of the qubits; from now on, the significance of a qubit is considered to be increasing from left to right in the associated quantum state label, e.g.  $|q_0, q_1, \dots, q_n\rangle$ .

The discussion on 2-qubit gates starts from *controlled-operations*, where the state of a subsystem can be used to choose if a unitary transformation is to be applied to another subsystem. Specifically, in the case of two qubits one will act as control

and the other as target [62](#):

$$c_0 U_1 = |0\rangle\langle 0| \otimes \mathbb{1} + |1\rangle\langle 1| \otimes U \quad (3.41)$$

here for example, if  $q_0$  is in the state  $|0\rangle$  nothing happens to  $q_1$ , instead if  $q_0$  is in the state  $|1\rangle$  the unitary  $U$  is applied to the subsystem corresponding to  $q_1$ .

Controlled operations are very general by nature, and are used to generate entanglement between the subsystems. By far one of the most fundamental is the **CNOT** gate, with matrix representation:

$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad (3.42)$$

and action on a generic two qubits state:

$$c_0 X_1 |x\rangle|y\rangle = |x\rangle \otimes X^x |y\rangle = |x\rangle |x \oplus y\rangle \quad (3.43)$$

where a modulus-2<sup>2</sup> compact notation has been introduced to express eq. [\(3.41\)](#). In eq. [\(3.43\)](#) an X gate is applied to qubit  $q_1$  whenever  $|q_0\rangle = |1\rangle$ .

Another important controlled operation set is composed by controlled rotations, where a single qubit rotation as in eqs [\(3.38\)](#), [\(3.39\)](#), [\(3.40\)](#) is applied conditioned to the state of the other qubit. It is possible to introduce the **CRX** gate for the controlled rotation around the X axis:

$$CRX(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos(\theta/2) & -i \sin(\theta/2) \\ 0 & 0 & i \sin(\theta/2) & \cos(\theta/2) \end{bmatrix} \quad (3.44)$$

**CRY** for the controlled rotation around the Y axis:

$$CRY(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos(\theta/2) & -\sin(\theta/2) \\ 0 & 0 & \sin(\theta/2) & \cos(\theta/2) \end{bmatrix} \quad (3.45)$$

---

<sup>2</sup>Remember that in binary arithmetics  $(1 \oplus 1) = 0 \bmod(2)$

and **CRZ** for the controlled rotation around the Z axis:

$$CRZ(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & e^{-i\theta/2} & 0 \\ 0 & 0 & 0 & e^{i\theta/2} \end{bmatrix} \quad (3.46)$$

these, much like the non-controlled rotations, are parametric gates, and for this reason they will be mentioned in section [3.2](#) as building blocks for variational quantum machine learning models.

A final two-qubit is the **SWAP** gate, which is a non-controlled operation (although it can be made controlled as for any other unitary gate) that exchanges the states of two qubits:

$$\text{SWAP } |x\rangle |y\rangle = |y\rangle |x\rangle, \quad x, y \in \{0, 1\}. \quad (3.47)$$

Equivalently, on the computational basis (ordered as  $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$ ),

$$\text{SWAP} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad (3.48)$$

so that

$$\begin{aligned} \text{SWAP } |00\rangle &= |00\rangle, & \text{SWAP } |01\rangle &= |10\rangle, \\ \text{SWAP } |10\rangle &= |01\rangle, & \text{SWAP } |11\rangle &= |11\rangle. \end{aligned} \quad (3.49)$$

This means that for a generic two-qubit state:

$$\begin{aligned} \text{SWAP}(\alpha_{00} |00\rangle + \alpha_{01} |01\rangle + \alpha_{10} |10\rangle + \alpha_{11} |11\rangle) \\ = \alpha_{00} |00\rangle + \alpha_{01} |10\rangle + \alpha_{10} |01\rangle + \alpha_{11} |11\rangle. \end{aligned} \quad (3.50)$$

The SWAP gate is both unitary and Hermitian, hence  $\text{SWAP}^\dagger = \text{SWAP} = \text{SWAP}^{-1}$  and  $\text{SWAP}^2 = \mathbb{1}$ . Lastly, notice that the SWAP gate can be decomposed in three CNOT gates as:

$$\text{SWAP}(q_0, q_1) = c_0 X_1 \ c_1 X_0 \ c_0 X_1, \quad (3.51)$$

This last equality lets us introduce the central concept of universality of the gate decomposition: a desirable property on a quantum computer is the possibility of implementing any generic unitary operator  $U \in \text{SU}(2^n)$  acting on a n-qubit system as a composition of simpler operators.

A finite and discrete set of quantum operators is said to be universal if it can be used to approximate at arbitrary accuracy any unitary operator [61].

There are different sets of universal gates with different properties. For example within the definition of approximate decomposition the textbook-discrete-set  $\{H, S, T, CNOT\}$  is universal; a stronger exact decomposition is the one provided by the continuous set of all the 1-qubit gates together with an entangling operation, commonly the CNOT gate. This second set is by definition larger than the first and since it is continuous the decomposition is exact.

There are two important distinctions to be made about universal sets, and one is about the scaling of the decomposition, while the second is about error correction codes and which sets are actually implemented on current devices.

The approximate decomposition is in general not efficient as the number of gates needed for a generic  $n$ -qubits unitary may scale more than polynomially. For the exact decomposition with continuous set the latest results [63][64] confirm the  $O(4^n)$  scaling proved initially by Vartiainen et al. [65]. For the approximate decomposition, the one that is actually implemented, the scaling is still dominated by the  $O(4^n)$  term [66].

As for the second distinction the real implementation of the universal set on current error-mitigated hardware differs; each implementation has its own native gates, which are related but different from textbook universal sets. Moreover other sets need to be introduced when the discussion shifts towards error correction [61]. This is related to the important topic of fault-tolerant hardware where one or more physical qubits are grouped in an error-correcting logical qubit. In this context an important set of quantum gates are the ones implementing Clifford group operators.

Clifford group elements are defined by their property of being stabilizers of another important group, the Pauli group [62]:

$$\mathcal{P}_n = \left\{ i^k \hat{P}_1 \otimes \dots \otimes \hat{P}_n : \hat{P}_i \in \{\mathbb{1}, \hat{X}, \hat{Y}, \hat{Z}\}, \quad 0 \leq k \leq 3 \right\} \quad (3.52)$$

which is the group of Pauli strings with a multiplicative scalar factor  $i^k$ . The concept of *stabilizer* will be used again in Chapter 6 together with other group theory formalism, but for the moment the intuitive notion is that any element of the Pauli group is mapped to some element of the same Pauli group by the action of a Clifford operator, formally:

$$\mathcal{C}_n = \left\{ \hat{C} \in U(H) : \hat{C} \hat{P} \hat{C}^\dagger \in \mathcal{P}_n \quad \forall \hat{P} \in \mathcal{P}_n \right\}. \quad (3.53)$$

An important result for the Clifford group is expressed by the Gottesman-Knill

theorem [67], which implies that simulating Clifford operators is efficient on classical machines.

In general the whole set can be built starting for example from  $\{H, S, CNOT\}$  and is not universal. Clifford group elements are therefore cheap in classical simulations, and because of their role of stabilizers of the Pauli group they are employed in Pauli-based error correction codes for fault tolerant quantum computing, which are fundamentally important for quantum error correction.

As already mentioned, the Clifford group, and therefore also the discrete set used for its decomposition, are not universal, and to get to universality the addition is usually the T gate. Since the T gate in many error correction codes doesn't have the convenient properties of the Clifford gates it is not directly implemented in error correction, but its action is obtained by the creation of a noisy T eigenstate, which gets distilled [68] and teleported into the system (*magic state injection*) [69].

This last topic projects quantum computing in its future fault resistant era, and ends this hopefully self-contained introduction to the main building blocks that are needed to understand the next section and chapters.

The next few sections are dedicated to an important application of the formalism derived up until now: quantum machine learning.

## 3.2 Quantum Machine Learning

This section provides an introduction to one of the central research fields of this thesis, namely quantum machine learning (QML), a hybrid research field at the intersection of machine learning and quantum computing. Among the many excellent references, [36] and [11] represent two of the best reviews of this field, and are a source for many of the topics presented in the following sections.

Historically, the term QML was systematically introduced by Lloyd, Mohseni and Rebentrost in 2013 [70], however, some of the concepts underlying QML, like quantum neural networks, can be dated back much earlier than the formal definition of the field, and come directly from the 1990s [71] [72].

In the past ten years quantum machine learning has undergone a rapid and fundamental growth, becoming today one of the most active disciplines among the new computing paradigms. QML evolution is tightly linked with the progress in classical machine learning, from which many of the core ideas of QML inevitably originate. QML however is not merely an extension of classical ML. This uniqueness stems from the underlying formalism inherited from quantum mechanics, which both constrains the operations that can be performed in an algorithm, and offers a new, exotic framework, sometimes without a true classical counterpart.

The evolution of QML has been largely driven by attempts to answer a few basic questions: What are the capabilities of QML, and which are the problems that can be meaningfully approached by it? In particular in which regimes is QML truly providing a learning separation with respect to the best known classical methods?

As it turns out, answering to these questions is not easy, both because of the not always intuitive nature of quantum machine learning, and because, as in classical machine learning, often the ability to bound the complexity of a QML method requires the formulation of a set of simplifying assumptions, that in the end provide just an upper-bound to the empirical performance of the algorithm.

To understand how QML has evolved towards a better understanding of these questions, it is useful to first recall why quantum computing is an attractive candidate for enhancing or redefining machine learning paradigms. As discussed in the previous sections, there are mainly three intrinsic quantum properties that can in principle be exploited by QML algorithms:

- Linearity (quantum parallelism), which is the property of a quantum algorithm to operate coherently on a superposition of states, allowing for the parallel evaluation of multiple computational paths.
- Entanglement, which can lead to the ability to access correlations that are not classically efficient to capture.
- An exponentially scaling Hilbert space in the linear number of qubits, which would require an in principle exponential number of classical resources to be simulated.

### 3.2.1 A classification for quantum algorithms

As a first step in understanding the broad field of quantum algorithms, a convenient classification of quantum algorithms was first introduced by Aïmeur, Brassard and Gambs [73] and shown in Fig. 3.9. There are four possible scenarios for ML-QML interplay:

- Classical data – Classical algorithm (**CC**): classical data are loaded in a fully classical model. This can both identify fully classical machine learning, or quantum-inspired algorithms [74] (e.g. Tensor Networks [75]). Dequantized quantum algorithms handling classical data enter this category as well [36].
- Classical data – Quantum algorithm (**CQ**): here classical data are loaded in a quantum state by a quantum embedding algorithm and are elaborated by a quantum algorithm [76]. This is the category where most of QML belongs, with classical data loaded and manipulated by a quantum circuit;

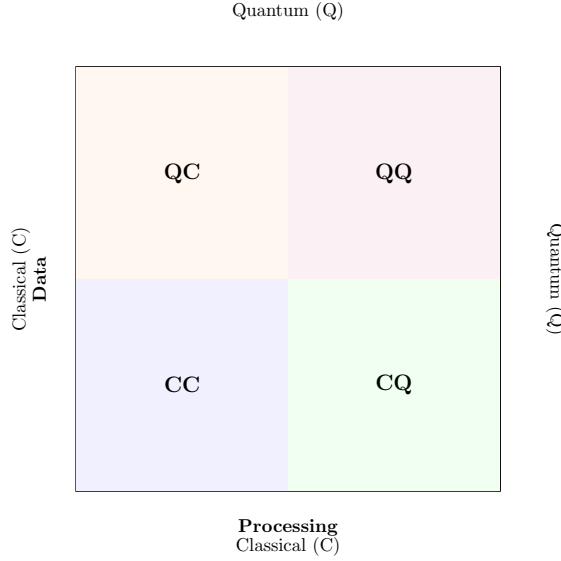


Figure 3.9: Classification for quantum algorithms based on the distinction between classical/quantum input data, and classical/quantum processing algorithm.

in particular it is the category of most quantum neural network applications [77] [20].

- Quantum data – Classical algorithm (QC): for this class definitions vary. The general philosophy is that a classical ML algorithm is used to solve a quantum problem [36], this includes, as an example, neural network architectures to compress a quantum state [78]. Recent literature like [74] restrict the scope to explicit quantum data being classically computed, including classical compilation of a quantum circuit in the case of unitary time-evolution.
- Quantum data – Quantum algorithm (QQ): in this case a quantum system is processed by a quantum algorithm [79] [80]. As for the previous case the interpretations in literature vary. The QQ class includes both explicit quantum data fed to a quantum circuit, or a quantum process fed to a quantum algorithm.

This class is directly linked with the development in CQ, because with minor variations CQ algorithms can be adapted to handle quantum datasets.

Other tasks belonging to this class include for example Quantum Phase Classification [81].

This classification highlights how non trivial it is to provide a simple and self-contained classification of QML methods. In general there are many orthogonal classes of algorithms, but the bounds between them are often not strict.

A differentiation that is very relevant for this thesis is the one between variational and non-variational quantum algorithms; this will be the topic of the next sections.

### 3.2.2 Data embedding

Before shifting the focus to the core of QML algorithms it is important to introduce the first, often overlooked, stage of every QML algorithm: data embedding.

Data embedding is the term used to indicate the circuit's initial state preparation, before the implementation of any other logic. For the remainder of the section it is assumed that the data to be encoded in a quantum state is classical, but in a broader sense a quantum algorithm where no measure operation is operated on the final state can be considered itself as a state preparation routine.

Data embedding is a very active research trend, and has been growing in relevance in the past years, together with a better understanding of the bottlenecks of the most widely adopted embedding schema, which are the ones discussed in this section.

When the input data domain  $\mathcal{X}$  is the one of classical bitstring  $\mathcal{X} \subseteq \{0, 1\}^{\otimes d}$  or real feature vectors  $\mathbf{X} \subseteq \mathbb{R}^d$ , it is natural to encode their binary representation  $x_b$  directly as states of the computational basis:  $|x_b\rangle$ . This is *basis encoding*, and depending on the precision and magnitude of  $x$  this can result in a large number of required qubits, but also to the possibility to coherently encode multiple digits.

A second common embedding is *amplitude encoding*, where input data are encoded as the complex amplitudes of a statevector [36]:

$$\mathbf{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_{2^n} \end{pmatrix} \leftrightarrow |\psi_{\mathbf{x}}\rangle = \sum_{j=0}^{2^n-1} x_j |j\rangle, \quad (3.54)$$

with  $\mathbf{x} \in \mathbb{C}^{2^n}$ ,  $\sum_k |x_k|^2 = 1$ . State preparation for amplitude encoding may be expensive in the number of gates, with a worst case scenario scaling linearly with the qubits' Hilbert space dimension [11]. The advantage is in the compact embedding of  $2^n$  features in the amplitudes of  $n$  qubits.

A final widely spread embedding method is *angle encoding*, where each  $x_i$  is considered as a rotation angle  $\theta_i$  around some axis  $\mathbf{v}$

$$R_{\mathbf{n}}(\theta_i) = e^{-i\theta_i \mathbf{n} \cdot \boldsymbol{\sigma}/2} \quad (3.55)$$

on the  $i$ -th qubit of a system, where  $\sigma_j = (X_j, Y_j, Z_j)$ . This can be considered a

special case of a more general time evolution of some Hamiltonian  $H$ , because on a fundamental level Eq. (3.55) can be rewritten as  $U(t) = e^{-itH}$ .

The advantage of angle encoding is in the reduced number of gates needed to embed a  $d$ -dimensional classical feature vector, which, as an example, with simple  $R_y$  rotations is performed as:

$$U(\mathbf{x}) = R_y(\theta_1) \otimes \cdots \otimes R_y(\theta_d) \quad (3.56)$$

on a system of  $d$  qubits. An intrinsic limitation is in the 1:1 mapping between feature and qubit.

### 3.2.3 Variational quantum algorithms

A quantum circuit where a subset of the quantum gates depends on free parameters is called a *parametrized quantum circuit* (PQC). PQCs are the foundation of *variational quantum algorithms* (VQA), where they are used in conjunction to classical optimizers to find the free parameters' best value to model a task. This is very similar to what a classical machine learning algorithms performs through learning, but in a hybrid context, where the circuit evolves a quantum state, measures the final configuration, and feeds the result to a classical optimizer that is responsible for the search in the (free) parameter's space of the configurations that minimize some cost function.

The following sections are dedicated to present the main concepts underlying VQAs, starting from their interpretation, then discussing how they are trained, and which applications are they commonly modeling.

#### Deterministic and probabilistic models

As for classical machine learning, a variational quantum circuit can be described both as a deterministic, or a probabilistic model.

The deterministic nature of a QML model emerges when, given the input data  $x \in \mathcal{X}$ , a parametrized quantum circuit acting as  $U(x, \boldsymbol{\theta})|0\rangle = |\psi(x, \boldsymbol{\theta})\rangle$ , and a measurement operator  $\mathcal{M}$  corresponding to some quantum observable, then:

$$f_{\boldsymbol{\theta}}(x) = \langle \psi(x, \boldsymbol{\theta}) | \mathcal{M} | \psi(x, \boldsymbol{\theta}) \rangle. \quad (3.57)$$

This expression can be read as the implementation of a deterministic function  $f_{\boldsymbol{\theta}}(x)$  through the measurement of the expectation value of  $\mathcal{M}$  on the circuit's final state.

Fig. 3.10 illustrates this process. Where the circuit action  $U$  is shown as decom-

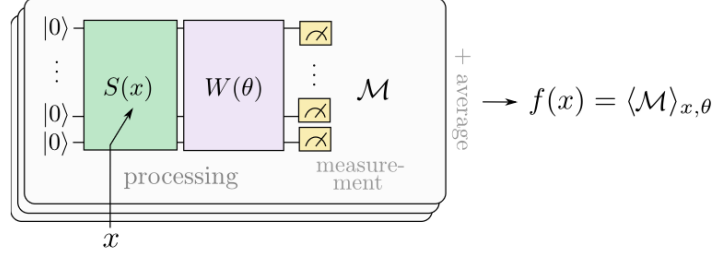


Figure 3.10: Deterministic interpretation of a parametrized quantum circuit, from [36]. The input  $x$  is evolved by a PQC, in this scenario divided between a non-parametrized embedding sub-circuit  $S(x)$ , a parametrized  $W(\theta)$ , and measurement  $\mathcal{M}$ .

posed in its non-parametric data embedding unitary  $S(x)$ , and parametrized operator  $W(\theta)$ :

$$U(x, \theta) = W(\theta)S(x). \quad (3.58)$$

The two blocks may then be further factorized as:

$$S(x) = T_{N+1} \prod_{i=1}^N S_i(x_i) T_i, \quad (3.59)$$

and

$$W(\theta) = V_{K+1} \prod_{k=1}^K W_k(\theta_k) V_k, \quad (3.60)$$

where  $T_i$  and  $V_i$  are truly fixed unitary operators, and the remainder are all parametric operators, but the  $S_i$  gates are non trainable, and the  $W_i$  are the trainable blocks.

The probabilistic interpretation of a quantum model sees instead a PQC as a model evolving a probability distribution.

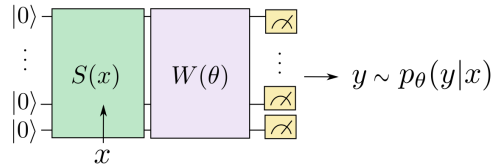


Figure 3.11: Probabilistic interpretation of a PQC for supervised learning, from [36]. Measuring the circuit samples the distribution  $p_\theta(y, x)$ .

In the case of supervised learning (Fig. 3.11) the circuit implements a transformation  $U(x, \theta) |0\rangle = |\psi(x, \theta)\rangle$ , and a measurement

$$\mathcal{M} = \sum_y y |y\rangle \langle y| \quad (3.61)$$

resulting in one of its eigenvalues  $y$ . This models the process:

$$p_\theta(y | x) = |\langle y | \psi(x, \theta) \rangle|^2 \quad (3.62)$$

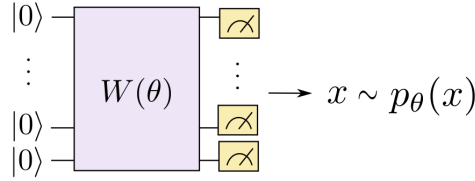


Figure 3.12: Probabilistic interpretation of a PQC for unsupervised learning, from [36]. Measuring the circuit samples the distribution  $p_\theta(x)$ .

Otherwise, for unsupervised learning (Fig. 3.12), a parametrized unitary  $W(\theta)$  evolves a non initialized state  $|0\rangle$ . The measurement operator in this case will be  $\mathcal{M} = \sum_{x \in \mathcal{X}} x |x\rangle \langle x|$ , which is an observable diagonal in the basis of the  $x$  data points. This models:

$$p(x) = |\langle x | \psi(\theta) \rangle|^2. \quad (3.63)$$

### Ansatz

Out of the three main blocks whose composition results in the global action  $U(x, \theta)$  of the circuit, most design choices focus on the parametrized part  $W(\theta)$ , called *ansatz*.

An ansatz may be chosen following different criteria. Some are very general purpose, and are designed to, for example, maximize expressivity and/or entangling capacity. Expressivity is the ability of a circuit to uniformly explore the space of quantum states, while entangling capability is related to the average entanglement of the states obtained by randomly setting the  $\theta$  parameters [12].

It is important to understand that, in general, higher entangling capability does not lead to better modeling; likewise, the higher the expressivity, the more the circuit is able to explore the Hilbert space, but this can imply a more difficult training.

For this reason, some ansatzes are task-specific [82] [83] [84], or designed to exploit

known symmetries of the problem [85][86][87]. This is analogous to equivariant architectures in classical machine learning: imposing meaningful structure and restricting the hypothesis space of the circuit often results in improved trainability and generalisation.

Finally, some ansatzes are architecture specific, like the *hardware efficient ansatz* [88], i.e. designed to maximally adhere to the underlying connectivity and native gates of a device, while others relax the assumption of a fixed structure, exploring the possibility of adaptively changing the circuit [89][90].

### Optimization

Underlying the ability of a PQC to model a deterministic function, or a probabilistic distribution, there is a classical optimization routine. A PQC is trained to model a task through the minimization of some cost function  $C(\theta)$ , similarly to classical ML:

$$\theta^* = \arg \min_{\theta} C(\theta) \quad (3.64)$$

Offloading the complex optimization task to classical computing is a convenient learning strategy, although it is hard to provide theoretical guarantees about the convergence to a good solution, or even to prove that the classical optimizer will be able to solve the optimization problem at all. Intrinsically, variational algorithms are therefore mostly heuristic, but upper-bounds to specific design choices have been studied, as in [91].

The cost function evaluation depends upon both the input data, and the choice of the specific unitary action of the circuit.

The solution to the minimization problem of Eq. (3.64) is searched using different strategies. QML imposes to the training process some additional constraints, inherent in quantum mechanics, like the impossibility of cloning states, and hence to have a real backpropagation algorithm, as it is intended in ML. Nonetheless Eq. (3.64) is approached even within QML, and because the optimizer is inherently classical, through either gradient-based or gradient-free methods.

Gradient-based methods usually rely either on the exact evaluation of the gradient of the cost function, or on approximate estimates. In the former case quantum computing relies on *parameter shift rules* [92][93][94]. One famous and quite general case, which applies to parametrized gates of the form:

$$\mathcal{G}(\mu) = e^{-i\theta G} \quad (3.65)$$

with  $G^2 = \mathbb{1}$ , is:

$$\partial_{\theta} f_{\theta} = \frac{1}{2 \sin(\theta)} (f_{\theta+s} - f_{\theta-s}) \quad (3.66)$$

where  $s$  is often fixed to  $\pi/2$ . The circuit providing  $f$  needs to be evaluated twice for each free parameter  $\theta$ . There exist generalizations to the rule in Eq.(3.66) for gates not in the form of Eq.(3.65), and in general, assuming the ability to change the circuit's parameters and to evaluate new expectation values  $g_{\theta}$ , it is always possible to find new parameter-shift rules [36].

Training VQAs through gradient descent methods based on exact gradient computation methods, like parameter-shift rules, is expensive. Alternatives are provided by natural gradient methods [95], approximations like SPSA [96], and gradient-free methods [12].

## Applications

The landscape of the applications of variational quantum algorithms is extremely diversified and rich, a full review is offered in [12] and the up-to-date [11]; this brief section has the goal to introduce some notable examples.

In parallel with the development of supervised , unsupervised , reinforcement learning architectures, similar models have been adapted to work with PQCs, with, for example QCNNs [84][97][98], QGANs [99][80], and quantum reinforcement learning [100]. This has resulted in the concept of quantum neural networks (QNNs), which is sometimes considered an improper extension of the term "neural network", as PQCs are not working on the same principles [11]. A PQC is inherently a unitary object, where non-linearity is not native as in neural networks. The connectivity itself of a PQC is different from a NN, because of no-cloning theorem. This make PQCs a very interesting an different research field, but not easily comparable to modern neural network architectures.

In this thesis, Chapter 5 belongs to the vast class of classical ML-inspired VQAs, with study on quantum graph neural networks for edge level classification in the context of particle track reconstruction.

VQAs are also largely used for optimization tasks, as in QAOA [101], linear systems [102], and non-linear equations as well [103].

Finally two fundamental branch of these algorithms are used for ground-state studies, as for the Variational Quantum Eigensolver (VQE) [104], and for the dynamical simulation of quantum systems, as a shallow-circuit alternative to Trotter-Suzuki evolution [105].

### Open questions on variational learning

Variational methods are an extremely interesting approach to quantum algorithms, because they turn hard routines into – often heuristic – optimization problems. This comes with some difficulties and open questions for current and future research.

In particular, as already discussed, this trade-off generally results in the difficulty to provide strict upper-bounds to the convergence of the optimization routine, similarly to e.g. neural networks.

Moreover, especially in the very recent years, the understanding of VQAs has dramatically changed thanks to a better understanding of phenomena like *barren plateaus*: as the number of qubits in a PQC increases, the gradients of the cost (loss) functions are exponentially suppressed, hindering learning. This has two important repercussions: a big underlying Hilbert space is not necessarily a good feature for a PQC, and restrictions to smaller subspaces, thanks, for example, to symmetry-preserving ansatzes counteract part of this problem.

However, important empirical and theoretical studies have shown that many of the most common ansatzes, used to limit barren plateaus, and thus guarantee trainability, are classically efficiently simulable [23]. These *dequantization* studies have shaped research in QML in the near past, and are providing important evidences on the potential, but also limitations of some VQA classes.

#### 3.2.4 Non-variational quantum algorithms

It would be a mistake to think of QML only in terms VQAs. Many algorithms modeling tasks on data distributions are not trained in the same sense of VQAs. These algorithms are as widespread as the variational ones.

Some of the most active areas of research in non-variational algorithms include quantum Boltzmann machines [106], and quantum kernels [107]; the latter are particularly relevant as they offer another perspective on many supervised variational quantum algorithms [10].

Other notable non-variational QML methods are quantum approaches to clustering [108], and a broad class of routines for ML-tasks like quantum principal component analysis [109].

Chapter 6 of this thesis is dedicated to expand the current frameworks of non-variational QML through an algorithm that, exploiting the quantum Fourier transform, models a probability distribution without the need for free parameters. Often – as it will also be the case for Chapter 6 – this comes with the need for deeper

circuits with respect to VQAs, and the assumption to design the algorithm for future fault-tolerant devices.

# 4

## Particle tracking

### Contents

---

|                                                            |           |
|------------------------------------------------------------|-----------|
| <b>4.1 Standard model physics</b>                          | <b>54</b> |
| <b>4.2 Colliders</b>                                       | <b>56</b> |
| 4.2.1 LHC                                                  | 57        |
| <b>4.3 Tracking particles</b>                              | <b>59</b> |
| 4.3.1 A classical tracking pipeline                        | 61        |
| 4.3.2 Final considerations on classical tracking pipelines | 67        |

---

Tracking is the overarching term used to identify the set of tasks involved in the reconstruction of charged particle trajectories, starting from raw signals, and aimed at the extraction of the track's parameters and the reconstruction of production and decay vertices of short-lived particles.

This thesis is dedicated to novel algorithmic approaches to particle tracking, and for this reason this chapter is meant as an introduction to the main concepts revolving around tracking fundamentals, and motivations for research and development in particle track reconstruction.

The chapter is organized as follows: Section 4.1 provides a concise overview of the Standard Model. Section 4.2 introduces particle colliders and detectors, as the environments that are targeted by the tracking algorithms discussed in this chapter, with a compact description of the LHC. Section 4.3 reviews tracking subsys-

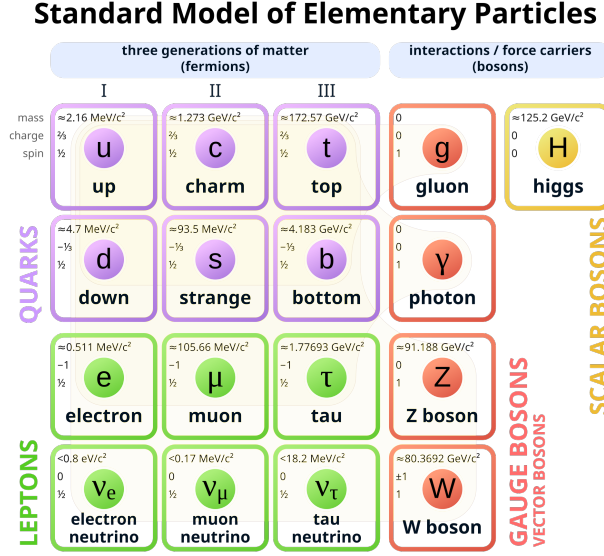


Figure 4.1: Standard model of particles - illustration from [110]. Here shown are the three generations of quarks (violet) and leptons (green), vector gauge bosons (orange), scalar Higgs boson (yellow)

tems, and the classical charged-particle tracking pipeline based on Combinatorial Kalman Filter, with remarks on performance and limitations in high-occupancy environments, that motivate more global approaches in later chapters.

## 4.1 Standard model physics

The study of natural laws and the search for a unifying theory of fundamental interactions is the ultimate goal of particle physics. The model that currently provides the best description for subnuclear interactions is the *Standard Model* (SM) of particle physics, which includes three out of the four fundamental interactions: electromagnetic, weak and strong (with the first two then unified in an electroweak theory).

The standard model classifies elementary particles, which are the targets and carriers of these forces, into leptons, quarks, and gauge bosons, with a special separate category for the Higgs boson (Fig. 4.1).

Leptons are spin-1/2 particles with unit electric charge, further commonly divided in three families: electronic, muonic and tauonic, each comprising an electrically charged particle and an electrically neutral neutrino, together with the respective anti-particles. Charged leptons interact electromagnetically, and, in general, all

leptons are subject to the weak force. Notably, within the standard model neutrinos are supposed to be massless particles, but experimental evidences of flavor oscillation necessitates theories of massive neutrinos beyond the standard model.

Quarks are spin-1/2 particles which interact not only through electromagnetic and weak forces, but are also carriers of the color charge of the strong interaction. As for leptons, they are commonly organized in 3 generations which also correlate with increasing mass: up and down, charm and strange, top and beauty, together with their anti-particles. Among the many notable properties they exhibit *confinement*, i.e., color is never observed in isolation, and *asymptotic freedom*, i.e., the strong coupling decreases at increasing energies and asymptotically quarks become non-interacting at asymptotically high energies. Together these two phenomena are strictly related to *hadronization*, which is the mechanism through which quarks and gluons produced in high-energy collisions (e.g. at LHC) turn into hadrons (bound states of two or more quarks).

Vector gauge bosons mediate the fundamental interactions and ensure the local gauge invariance of the standard model Lagrangian under the gauge group  $SU(3)_C \times SU(2)_L \times U(1)_Y$ . The eight gluons  $g$  are the gauge bosons of  $SU(3)_C$  (strong interaction), while the electroweak gauge fields of  $SU(2)_L \times U(1)_Y$  mix, after spontaneous symmetry breaking, into the massless photon  $\gamma$  (electromagnetic interaction) and the massive weak bosons  $W^\pm$  and  $Z$ . The Higgs boson plays a special role as a massive scalar excitation of the Higgs field, whose non-zero vacuum expectation value both triggers electroweak symmetry breaking and generates masses for the otherwise massless charged leptons, quarks, and the  $W^\pm$  and  $Z$  bosons, while leaving the photon and gluons massless.

Elementary particles can form a broad variety of composite particles. Electromagnetic and strong forces are known to create bound states, while this is not true for weak interaction. The class of *hadrons* represents bound states of the strong interaction, and includes *baryons*, which are composite subatomic particles made up of three valence quarks, *mesons* which are instead built on pairs quarks and anti-quarks, and exotic hadrons [111], which are QCD bound states with more than three quarks and/or possible gluonic degrees of freedom.

In the past hundred years, physics research from accelerators has been a constant driving force for the study of elementary and composite particles, and standard model has been probed in a wide variety of regimes. As a theory SM is already the broadest and most precise theory of nature ever assembled, through decades of theoretical predictions and experimental results. It is however well known that it is not a final theory, not only because gravity is not included, but also because there is a full corpus of open questions that are not answered within the standard

model. Two of the most famous are neutrino masses, which by SM should be zero, or the unsolved problem of the matter-antimatter asymmetry, which is not solved by the SM's CP violation sources. Moreover, arguably the end goal of particle physics is a unification of the interactions, but a Grand Unified Theory remains elusive.

Because of these fundamental questions, and many more not addressed in this short introduction, research in particle physics is an ever expanding field.

## 4.2 Colliders

In the past century the history of high energy physics has been deeply correlated with the access to increasingly higher energy scales, starting from different initial-state particles and processes. This has enabled the possibility of probing a variety of sectors of the standard model, with different classes of dynamics.

Particle accelerators are the human-made machines that, through controlled particle acceleration, represent a way to study particle physics at a wide range of energy scales.

Within physics research at accelerators, a fundamental role is played by colliders: machines where particles are accelerated into beams propagating in opposite directions, and made to collide to produce events.

An accelerator is a complex device which undergoes many design choices, well beyond the scope of this thesis, but two of the most important parameters are *center of mass energy* and *luminosity*.

Specifically, center of mass energy, corresponds to the maximum energy at disposal for the final state particles, and is defined as:

$$\sqrt{s} = \sqrt{\left(\sum_{i=1}^2 E_i\right)^2 - \left(\sum_{i=1}^2 \mathbf{p}_i\right)^2} \quad (4.1)$$

where  $E_i$  is the  $i$ -th initial particle total energy and  $\mathbf{p}_i$  is its 3-momentum. Collider experiments are typically able to access higher  $\sqrt{s}$  than fixed target experiments, where final state particles are generally produced with high momentum<sup>[112]</sup>. Center of mass energy is crucial to make heavy particles -both composite<sup>[113]</sup> and elementary<sup>[114]</sup>- kinematically accessible. Luminosity is instead related to event rates. In particular the *instantaneous luminosity*  $\mathcal{L}_b$  at an interaction point is the event rate  $R$  per unit cross section  $\sigma$ . It measures the number of interacting particle pairs per unit time and transverse area, where the cross section is a measure

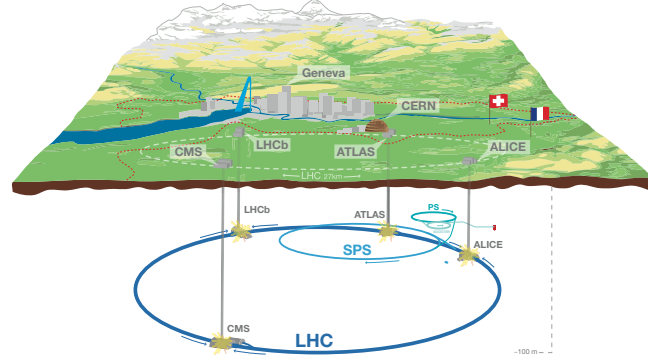


Figure 4.2: LHC illustration from [115]. The main acceleration stages are shown in blue: the Proton Synchrotron (PS), the Super Proton Synchrotron (SPS) and the LHC itself. In yellow the four interaction points, and the corresponding four major CERN experiments: Alice, ATLAS, CMS, and LHCb.

of the probability of a certain process to occur. Formally:

$$\mathcal{L}_b = \frac{R}{\sigma}. \quad (4.2)$$

A cumulative measure of luminosity is the *integrated luminosity*, defined as:

$$\mathcal{L}_{\text{int}} = \int_{t_1}^{t_2} \mathcal{L}_b dt. \quad (4.3)$$

Increasing the luminosity is crucial to improve the sensitivity to processes with low cross-sections. This in turn has the effect of increasing the number of primary interactions per bunch crossing, hence an increased flux of particles to detect and track.

#### 4.2.1 LHC

The Large Hadron Collider (LHC) (Fig. 4.2) is a circular accelerator located at CERN (Geneva). With its length of 27km and center of mass energy of 13.6TeV it is the largest and most energetic collider currently operating. LHC mainly<sup>1</sup> operates as a proton-proton (pp) collider.

<sup>1</sup>LHC has also physics programs for ions

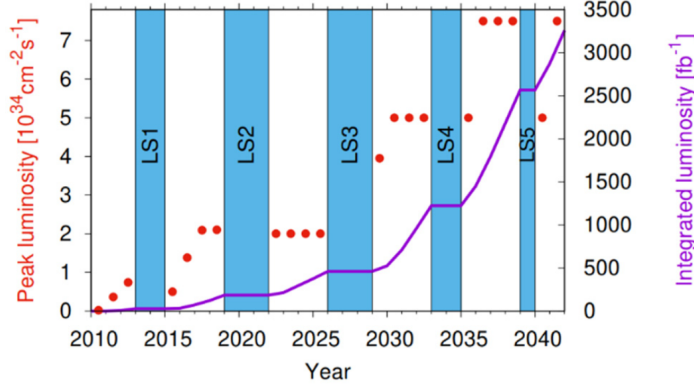


Figure 4.3: Luminosity evolution from [116] for the ATLAS and CMS detectors. Data for Run 1-3 extrapolated, future Runs data from predictions.

Before being actually injected in the LHC, protons are pre-accelerated by previous generation machines (i.e. the Proton Synchrotron (PS) and the Super Proton Synchrotron (SPS)). Then, a sequence of radio-frequency cavities are used to accelerate the beams to energies up to 6.8 TeV each. The proton beams are collimated and kept in a circular trajectory by superconducting magnets, with four interaction points in correspondence of the four major CERN collaborations: ALICE, ATLAS, CMS, and LHCb.

The processes originating all of the pp-based physics studied by the experiments stem from the interactions of protons' constituents, which are governed by QCD. Among these processes, Hard-QCD scattering events –corresponding to large momentum transfer and described by perturbative QCD– are typically the preferred initial interactions. In this case Soft-QCD interactions –with low momentum transfer and therefore belonging to non-perturbative QCD– are often treated as background.

Collisions at the four interaction points happen every 25 ns with different luminosity for each experiment. Specializing the definition of the previous section, because the relevant interactions are proton-proton inelastic collisions, Eq. 4.2 becomes:

$$\mathcal{L}_b = \frac{\mu f_r}{\sigma_{\text{inel}}} \quad (4.4)$$

where  $\mu$  is often referred to as *pile-up*: the average number of inelastic interactions

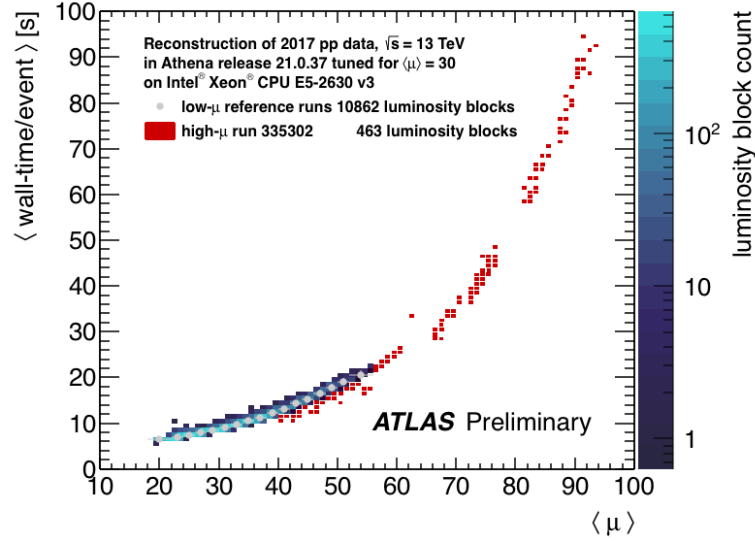


Figure 4.4: Average reconstruction wall-time per event as a function of pileup  $\mu$ . Colored points show the number of luminosity blocks in each bin at reference low- $\mu$ , red squares correspond to high- $\mu$  run.

per bunch crossing<sup>2</sup> and  $f_r$  is the orbit frequency. Fig 4.3 shows a summary of the peak luminosity values for Run 1-3, and estimates for future LHC Runs.

During Long Shutdown 3 (LS3) both LHC and the ATLAS and CMS experiments at CERN will be upgraded to enter a High-Luminosity regime (HL-LHC), where peak luminosity is scheduled to reach  $5 \times 10^{34} \text{cm}^{-2} \text{s}^{-1}$  from current  $2 \times 10^{34} \text{cm}^{-2} \text{s}^{-1}$ . These values correspond to  $\mu \sim 140$ , but CMS and ATLAS detectors are being upgraded to reach up to  $\mu \sim 200$ <sup>3</sup>.

HL-LHC will greatly enhance the amount of data at disposal for analyses, at the cost of a higher computational effort at different stages in the pipeline of data processing, as shown in Fig 4.5 for the annual CPU consumption, and Fig 4.4 for the reconstruction at the ATLAS experiment.

### 4.3 Tracking particles

Particle detectors are the experimental devices responsible for the collection of data coming from a broad variety of fundamental interactions. Their design

<sup>2</sup>this is defined as in [117]. Notice however that for other collaborations, like LHCb there is a distinction between  $\mu$  which is defined on bunch crossings, and pile-up which is defined on visible events, hence as the zero suppressed  $\mu$ . This is due to the very different values of  $\mu$  between LHCb and general purpose experiments like Atlas or CMS.

<sup>3</sup>The Alice and LHCb experiments follow a different upgrade schedule.

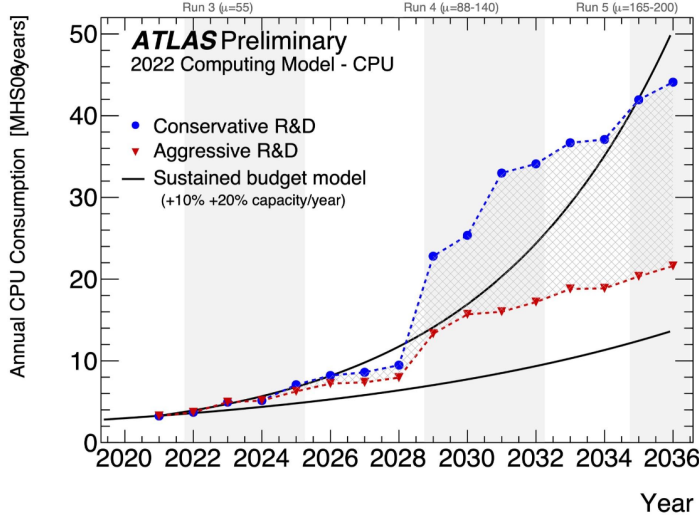


Figure 4.5: Estimates for the CPU usage within the ATLAS collaboration from [118] for Run 1-3 and projections for HL-LHC. Different curves correspond to different assumptions on R&D. MHS06 is a measure of aggregate CPU capacity in high-energy physics computing, corresponding to about  $10^6$  HS06 benchmark units, i.e.  $10^5$  modern CPU cores where each delivers  $O(10 - 15)$  HS06.

choices strongly depend on the physics they target.

The algorithms developed and characterized in this thesis can be considered detector-agnostic in the sense that the research presented in Chapter 5 uses data from a synthetic detector inspired by a mixture of CMS and Atlas. The algorithm in Chapter 6, because of its fundamental novelty and stage of development, is not tailored to a specific dataset and is instead a proposal for a generic multi-object tracking.

Even in the context of the four major LHC collaborations the structure of the respective detectors can differ substantially with the different physics targets. A generic illustration with the majority of the sub-detectors usually found in a high pileup, general purpose experiment, such as ATLAS or CMS, is shown in Fig. 4.6.

Among the different elements of a particle detector this thesis mainly targets trackers: the sub-detectors responsible for precision measurements of particles' positions. Trackers are built off a broad variety of technologies, from silicon pixels and strips (ideal for high granularity and for high-momentum particles, but expensive), to plastic scintillating fibers (minor cost and fast time response), and gaseous detectors (usually, within LHC major experiments, covering the outermost big detector areas, with a notable exception in ALICE). Trackers perform

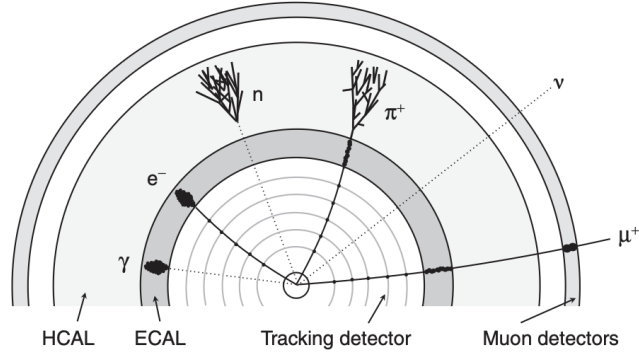


Figure 4.6: Pictorial representation of typical sub-systems in a cylindrical detector, and depiction of their specialized role in the reconstruction pipeline; image from [112]. The innermost sub-detectors are trackers, responsible for high-precision position measurements. Electrons, positrons and photons interact with matter inside the Electromagnetic Calorimeter (ECAL), which is designed for their destructive energy measurements, based on the showers they generate. Similarly the HCAL is responsible for the showering of hadrons. The last stage is dedicated to muon detection; muons are much more massive than electrons and minimally ionize in ECALS. High energy muons are generally not stopped in muon detectors, but still leave a signal at their passage.

non destructive measurements and generally full tracking is achieved by a synergy of different sub-detectors.

#### 4.3.1 A classical tracking pipeline

Charged-particle tracking is not performed by a single algorithm, but by a modular processing chain that integrates measurements from multiple sub-detectors. Although this means that each experiment has its own highly optimized tracking pipeline, there are generally some central routines that characterize the full process.

Among the three major LHC collaborations focused on proton–proton physics, the reconstruction of particle trajectories differs. LHCb’s Run 3 tracking is based on a pipeline that integrates five complementary track classes: VELO, T, upstream, downstream, and long tracks (Fig. 4.7); built from different combinations of its vertex and tracking stations [119] [120]. In this pipeline a Hough transform inspired routine plays a central role in downstream tracking. By contrast, ATLAS and CMS structure their pipelines around the Combinatorial Kalman Filter (CKF) [121].

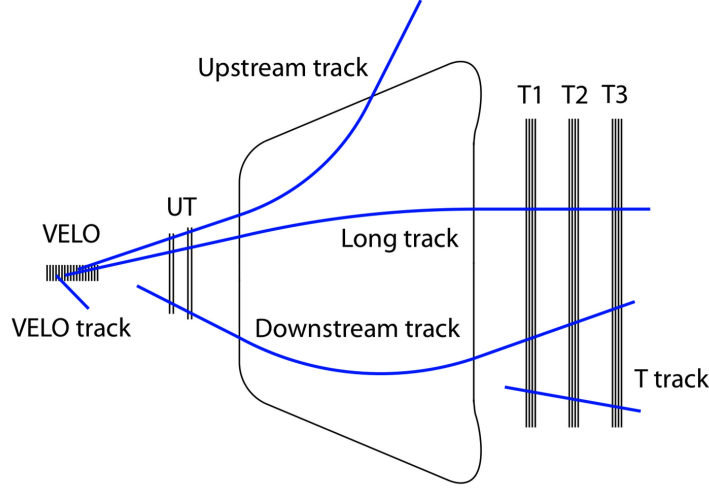


Figure 4.7: LHCb tracking system and track types, from [122]. The tracking sub-detectors are the pixel Vertex Locator (VELO), silicon strip Upstream Tracker (UT), and three stations of scintillating fibers (SciFi). The VELO surrounds the luminous region, and particles are tracked in the forward region. Five different types of tracks are reconstructed from the possible combinations of the sub-detectors.

Reconstruction routines based on CKF (Fig. 4.8) are used in high pileup environments as they offer a solid solution to the combinatorial explosion of candidate tracks. CKF embodies a largely local strategy, extending seeds through successive detector layers. As the objective of this thesis is the exploration of novel approaches to particle tracking for high pileup experiments, CKF routines are a natural reference baseline. The remainder of this chapter reviews the CKF pipeline, discussing the following five steps:

- 1: **hit-points generation**
- 2: **seeding**
- 3: **track finding**
- 4: **track fitting**
- 5: **track selection**

as detailed in [123] and [25], which are tailored to CMS.

In this context Combinatorial Kalman Filter is a powerful integration of track finding and fitting, and represents an extension of the original Kalman Filter

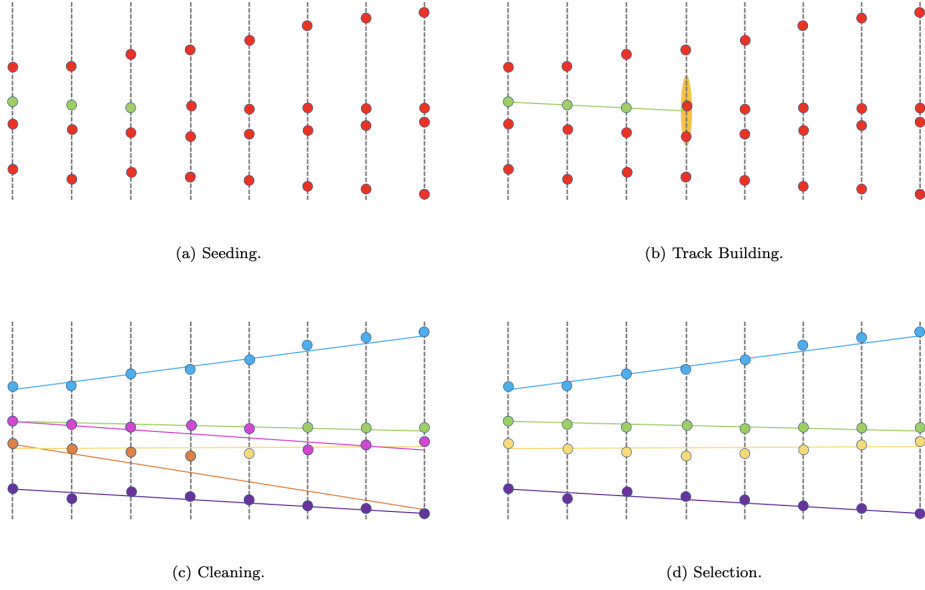


Figure 4.8: Simplified summary of the main steps of a tracking pipeline based on combinatorial Kalman filter for straight tracks; image from [25]. (a) *Seed building* from the innermost tracking layers; green dots represent a possible 3-hit seed. (b) In *track building* the green seed is propagated to the next layer, where new hits are searched for in an orange compatibility window. This procedure will eventually fork different new tracks from the same seed, each with its own track parameters. (c) *Re-fitting and Cleaning* computes new, higher quality fits for the track candidates, and tracks that don't share too many hits are kept in the candidates pool. (d) A *track selection* routine is run on the tracks that survived the previous steps, and a set of criteria is enforced, discarding tracks that don't satisfy them.

[124], which is instead a broadly used fitting algorithm. The goal of the next sections is to disentangle the two concepts and discuss the CKF pipeline.

Step 1 is usually not considered properly as track reconstruction, and usually tracking algorithms assume access to already clustered hits. Steps 2-5 in a CKF-based algorithm can be performed following different pipelines, e.g. a possibility is to iterate them, especially in offline reconstruction, starting from the cleanest tracks, and progressively masking them in successive iterations as in [123], but this is not the only possibility, as in, e.g., ATLAS Run 3 tracking [125].

The following discussion is qualitative and aimed at a general description of the pipeline.

### Hit-points generation

Particles propagating through trackers leave raw signals that are not yet the hit-points of the reconstruction. Depending on the technology of the underlying detector, different hit reconstruction routines are carried over (e.g. hits coming from pixels are reconstructed with a different procedure than hits coming from strips or non-silicon modules). The general idea is that, from multiple energy deposits, a clustered hit is created, with associated uncertainty measurements.

### Seeding

Hits, usually from the innermost, high granularity, detector layers are combined to build short geometrically allowed initial tracks, i.e. *seeds*. This is a general concept, not specific to CKF pipelines.

On its own, as CKF is in its essence following these seeds to prolong the tracks adding new points layer-by-layer, seeding is a step of crucial importance, with strong repercussions in the following track building. For this reason, seeding in the innermost detector region is generally preferred to seed building from the hits belonging to the outermost detectors, which offer by nature minor spatial accuracy.

Seeds are built from just a few hits—typically two or three—as a compromise between efficiency (the probability to find genuine tracks) and fake rate (the fraction of spurious track candidates formed from wrong hit combinations). This depends once again from the detector design: in magnetic-field-free regions (e.g. some vertex sub-detectors [126]), seeds are built under a near-straight-line assumption. Generally, however, trackers are within field regions, and designed to obtain transverse momentum and charge information from the bending of the reconstructed tracks.

A charged particle in a uniform magnetic field follows an helical trajectory, which can be parametrized by five parameters. These five parameters can be extracted by either three 3D hits or two 3D hits and a constraint on the luminous region, origin of the track [\[123\]](#).

### Track finding: Combinatorial Kalman Filter

A Kalman Filter is applied to the seeds built in the previous step to extract the track parameters, together with an associated covariance matrix, which keeps a trace of the uncertainties on each of the parameters, and respective correlations. For a detailed discussion on Kalman Filter refer to Appendix [A](#).

Using the directional information coming from the seed's parameters, a navigator algorithm is responsible for searching the detector layers where the next hit belonging to the same particle is likely to be. This is trivial when the propagation happens in the barrel region of cylindrical trackers, as just neighbor layers are eligible; navigation is more complex e.g. in the endcaps.

Once a layer is selected, a subroutine searches for the layer's modules likely to be in the trajectory. In this step the scope of the search has to be limited to avoid a blow-up of the possible candidates; it is also necessary to account for overlapping modules, multiple scattering, and potential absence of signal due to inefficiency in an otherwise correctly identified module, with the generation of a *ghost hit* as a replacement.

A  $\chi^2$  test is performed between the propagated proto-track state on the new layer, and each of the potential new hits in that layer. A cut-off on the  $\chi^2$  reduces the number of candidates. Each accepted new hit spawns a new, updated proto-track, originated from the old one. To once again avoid a combinatorial blow-up, a threshold on the number of new spawned tracks is set. For the accepted track the associated five parameters and covariance matrices are updated with the addition of the latest hit.

This is in its essence the track building stage of a Combinatorial Kalman Filter algorithm.

The outward track following routine can be complemented by an inward search, which gets triggered as the tracks reach a fixed number of hits, and serves as an attempt to enrich the track hit set with more hits from the regions closest to the interaction point.

It is worth highlighting that it is natural at this stage for the tracks being built to share hits, as the different CKF forks can share a common proto-track portion, or,

similarly, different but generally neighboring seeds can end up aggregating similar tracks. For these reason a first *cleaning* procedure can be performed right after track finding.

A possible cleaning routine from [123] involves the evaluation of

$$f_{\text{shared}} = \frac{N_{\text{shared}}}{\min(N_1, N_2)} \quad (4.5)$$

where  $N_{\text{shared}}$  is the number of shared hits between two candidate tracks,  $N_1$  and  $N_2$  are the total number of hits of which the two tracks consist. The threshold is generally empirical, and when it is exceeded the track with fewer hits gets removed from the pool of candidate tracks. In the limiting case of  $N_1 = N_2$  the track with largest  $\chi^2$  is the one getting cleansed.

A certain number of additional criteria are applied during track building; early-termination conditions on hit aggregation processes are common, e.g., a drop in the estimated  $p_T$ , or an excessive number of missing hits, replaced by ghosts.

### Track fitting

The trajectories created during the previous stage are refitted using Kalman filter and smoother, to get a new, least-biased estimation for the track parameters and covariance. At each layer, fitting uses a high accuracy numerical solver (e.g. a Runge-Kutta integrator) to find estimates of the expected next hit, evolving the first order differential equations of motion. This computationally demanding procedure accounts both for the magnetic field inhomogeneities, and the effects of material, improving on the material propagator of the track finding step. Fitting generally acts in two steps, first an outward direction Kalman filter, from the innermost to outermost hit, and then an inward direction smoother, from outermost and back in the direction of the vertex. The smoother, starting from the state of the last hit  $N$  from the forward pass, updates the state of the track at each position  $k$  starting from  $N$  and back to 1, combining the outward  $k$ -th state with the inward information from future hits ( $k + 1, \dots, N$ ). The resulting smoothed state at  $k$  is therefore conditioned on all measurements ( $1, \dots, N$ ).

Following smoothing there can be further cleaning, with the exclusion of outliers hits from tracks. This is done through different routines, e.g. thresholding once again some  $\chi^2$  estimation, or even, at a lower level, considering the charge distribution from which the hit originated.

### Track selection

Tracks built from the previous stages still present a high portion of fakes: tracks that are nonphysical, i.e. not associated to any particle, just reconstruction by-products.

In the track selection stage sets of criteria are applied to each track to filter away the fakes and improve purity. Among the different possible cuts, some are:

- number of layers with at least one hit belonging to the track (this also accounts for the possibility of, e.g., overlapping modules in a layer);
- number of layers that are not contributing any hit to the track;
- minimum number of 3-D hits (e.g. from pixel trackers or matched strips);
- further constraints on the track  $\chi^2$ ;
- compatibility with primary interaction point.

The strictness of the constraints can vary. In iterative pipelines, for example, in the first iterations tracks with high primary vertex compatibility are favored, while hit counts are a stronger criteria in later iterations, to preserve efficiency on tracks originating from displaced vertices.

#### 4.3.2 Final considerations on classical tracking pipelines

The tracking steps presented in the previous section are simplification of a track reconstruction routine in a real experiment at LHC. Already from this simplified description it is evident that particle tracking is an extremely complex task by its own, without even mentioning the need for fast routines to be used e.g. in online environments.

One of the roads to discover new physics involves luminosity upgrades, which pose important challenges to every aspect of an experiment, from sensors, to data acquisition systems, and storage. Tracking performance is particularly susceptible to increasing pileup.

While reconstruction pipelines based on combinatorial Kalman filter will still be responsible for the highest pileup tracking in HL-LHC, as the number of hits increases, the constraints set in place to avoid a blow-up in the combinatorial will need to become increasingly stronger.

For this reason it is important to research new algorithmic approaches to particle tracking. Modern machine learning methods already support specific sub-tasks (e.g. in LHCb, some matching between tracks from different sub-systems is already done by deep neural networks). In parallel, broader efforts are exploring

reconstruction as a global rather than purely local optimization, using architectures such as the graph neural networks that inspired the project of Chapter [5](#).

# 5

## Quantum Graph Neural Networks for particle tracking

### Contents

---

|                                                                                |    |
|--------------------------------------------------------------------------------|----|
| <b>5.1 Scientific motivation</b>                                               | 70 |
| <b>5.2 The TrackML dataset</b>                                                 | 71 |
| 5.2.1 Dataset cuts                                                             | 72 |
| <b>5.3 Graphs</b>                                                              | 72 |
| 5.3.1 Graph operations                                                         | 74 |
| 5.3.2 Graph building from hits point cloud                                     | 76 |
| <b>5.4 Graph Neural Networks</b>                                               | 76 |
| 5.4.1 Structure of a GNN for edge classification                               | 78 |
| <b>5.5 Quantum Graph Neural Networks</b>                                       | 80 |
| 5.5.1 Phase I: development and pile up studies on the original<br>architecture | 81 |
| 5.5.2 Phase II: upgrades to the original QGNN model                            | 86 |
| <b>5.6 Conclusions and acknowledgments</b>                                     | 90 |

---

## 5.1 Scientific motivation

The reconstruction of particle's trajectories, starting from the clustered energy deposits left during their propagation through trackers' layers, is a computationally intensive task, as detailed in Chapter 4. At the luminosity regimes that LHC is entering in Run 4, and in general in its high luminosity phase, particle tracking will need to account for an increment of a factor 3-5 in the number of primary proton-proton interactions.

While the tracking pipelines currently in use up to Run 3 for general purpose experiments are being upgraded to handle the combinatorial blow-up, which is a consequence of the increased proton flux, these methods are inherently limited by their local approach to track reconstruction.

In recent years, a plethora of novel approaches has emerged, most of them being based on machine learning and targeting different stages of the reconstruction pipeline. In particular, Graph Neural Networks (GNN) are one of the most promising technologies for particle tracking [17] [18] [19] and for particle physics applications in general [127] [128] [129].

As an extension to machine learning models, and parallel to their development, quantum machine learning algorithms have also been proposed

Alongside ongoing developments in classical machine learning for tracking in future high pileup experiments, quantum computing and quantum machine learning models are being explored as alternative paradigms, with current work focused on understanding if and under what conditions they could provide a benefit. Although conceived more than forty years ago, quantum information and computation theory has seen a huge increase in attention in recent years, with the development of both new software frameworks and quantum hardware. In particular quantum machine learning, as a hybrid field researching how learning from data can be performed through quantum algorithms, has seen a considerable development in the last decade, often in parallel with -but not constrained to- the modern machine learning technologies it takes inspiration from.

The research presented in this chapter is about the software development, characterization, and upgrades of a Quantum Graph Neural Network (QGNN) for particle tracking.

QGNNs belong to the broad class of variational quantum machine learning algorithms that can be tested in this NISQ (Noisy Intermediate Scale Quantum) era, where the number of qubits and circuit depth are necessarily limited. In particular, QGNNs are an advanced application of the quantum neural networks

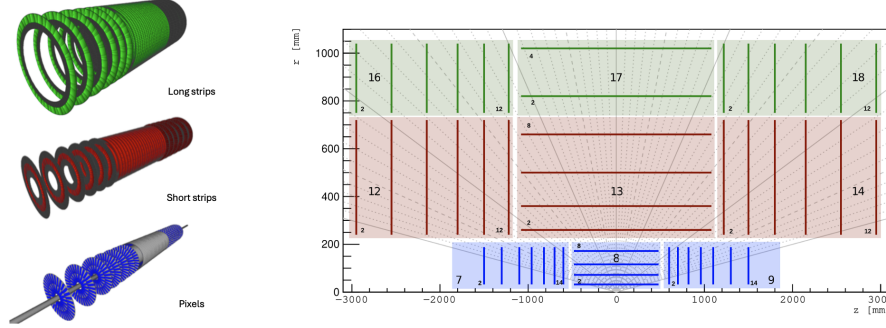


Figure 5.1: TrackML detector layout from [130]. The three different sub-detectors are shown on the left and with the associated colors in the  $z$ - $r$  plane depiction on the right. Grey lines correspond to different pseudo-rapidity  $\eta$  values.

(QNN) of Section 3.2, in an interplay with classical graph neural networks. This class of architectures exploit the flexibility and representation capacity of classical multi-layer perceptrons, to feed and extract data from parametric quantum circuits.

The goal of this research is to characterize and upgrade the original proposal of a QGNN for particle tracking first introduced in [20]. The study is aimed at improving the understanding of hybrid variational quantum technologies, and their role in experimental particle physics.

The chapter is organized as follows: Section 5.2 discusses the dataset used in this research; Section 5.3 then discusses the concept of graph, as the underlying data structure that captures the relations between the hits in a collision event. The classical model of a graph neural network is described in Section 5.4, which serves as a foundation for the discussion, in Section 5.5, on the development, characterization, and upgrades on a quantum graph neural network architecture, and represents the core of the chapter. In particular, the original content is organized in two phases, one for every major development of this project.

## 5.2 The TrackML dataset

The high pileup data used in this research comes from the simulated TrackML dataset [131], which is a standard reference for machine learning research on particle tracking. The dataset is based on a modular, synthetic tracker whose design is based on the high-luminosity upgrades of both CMS and ATLAS inner trackers.

The detector, as shown in Fig. 5.1, is made of three sub-trackers:

- a silicon pixel detector with spatial segmentation  $50\mu m \times 50\mu m$ ;
- a silicon short strip tracker with spatial segmentation  $80\mu m \times 1200\mu m$ ;
- a silicon long strip tracker with spatial segmentation  $0.12mm \times 10.8mm$ .

The dataset is based on a Pythia 8 event generator, with, for each event, a hard QCD interaction producing a  $t\bar{t}$  pair, and 200 additional soft QCD interactions, simulating the maximum predicted pileup of HL-LHC. The detector simulation models an ATLAS-like inhomogeneous magnetic field, and the propagation of the particles through the trackers' layers is obtained by an ACTS-based [132] simulation, accounting for different material interactions (multiple scattering, energy loss, and hadronic interactions). The simulated dataset consists of particles with transverse momentum  $p_T \geq 150\text{MeV}$ .

### 5.2.1 Dataset cuts

Before using the TrackML events as input to the QGNN, they undergo a pre-processing step, necessary to reduce the number of hits and, consequently, of candidate track segments. The preprocessing of this project keeps the same cuts as in the original proposal [20].

In the tests presented in this chapter only the hits in the central (barrel) regions of Fig. 5.1 are considered, hence the volumes labeled 8, 13, and 17; for a total of ten tracking planes. This restriction can be lifted in future studies, but represents a good efficiency compromise, especially in the hybrid quantum framework, between training time and fidelity to a real world tracking scenario.

Similarly (shown in Fig. 5.2), a cut is enforced on the particles' transverse momentum, to only retain  $p_T \geq 1\text{GeV}$  tracks. This cut is also motivated by the training time of the hybrid quantum model, both when the quantum resources are simulated on classical hardware, and when tests are performed on real quantum hardware.

## 5.3 Graphs

A graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  is a collection of objects  $\mathcal{V}$  (named nodes, or vertices), together with a set of relations between said objects  $\mathcal{E}$  (edges, or arcs). Given two nodes  $i, j \in \mathcal{V}$ ; the edge going from  $i$  to  $j$  is  $e = (i, j) \in \mathcal{E}$ . If the edge has no preferential direction, hence  $(i, j) \equiv (j, i)$  the edge is said to be undirected; if the opposite is true it is a directed edge.

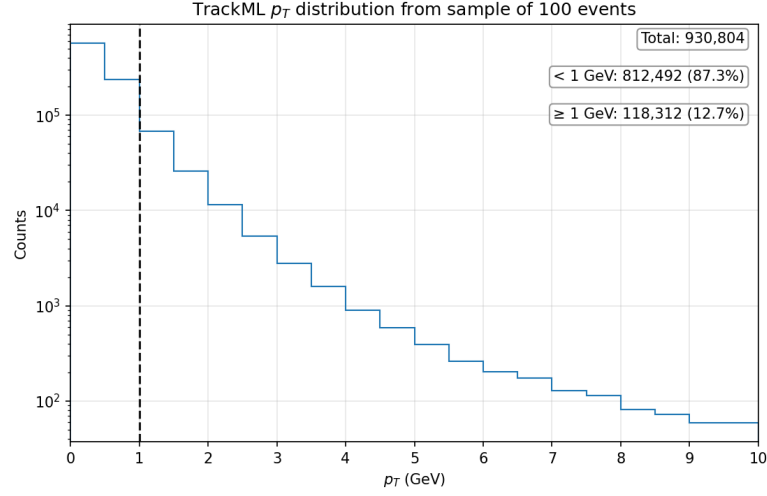


Figure 5.2: Distribution of transverse momentum  $p_T$  in the 100-events sample dataset from [131]. The dashed black line represent the cut enforced at 1GeV.

Edges can be both weighted or unweighted. An arbitrary graph is weighted if it is equipped with a weight function  $w : \mathcal{E} \rightarrow \mathbb{R}$ , where the weights  $w_e$  are all non trivially one.

Using these simple properties it is possible to model an incredible amount of complexity. Graphs, in fact, are such a flexible data structure that examples of entities that can be modeled through them can be found everywhere. Molecules are graphs where particles are nodes, and interactions are the graph's edges; lattices are very regular graphs where points are the nodes and the possible links are the associated edges; even an image in this sense is a bidimensional graph. Graphs are the underlying description of the relationship between people in a room, or in a social network. Neural networks are graphs, and even mathematical expression can be recast to a graph representation.

Most importantly for this research, events from a collider have a convenient representation as graphs. In fact, as shown in Fig 5.3, it is possible to identify as nodes the hits associated with the different particle's trajectories, and with edges the links between the hits. In this context, edges are candidate track segments, and can represent both true, physical links, or fake, wrong paths.

The task of classifying true and fake track segments is a way to recast the task of reconstructing a particle trajectory as a graph problem.

In general there are three macro categories of classifications that can be performed on a graph, and it is useful to introduce them to understand the different kinds

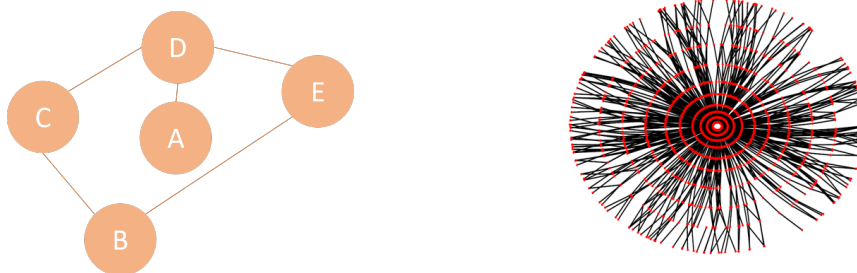


Figure 5.3: Left: representation of a simple, generic, graph. Labeled circles are the graph nodes, and their link is a graph edge. Right: a graph representation of a post-processed event from the TrackML dataset. Red graph nodes are the particle hits, and the black edges are the candidate track segments connecting hits that survive the geometric cuts from the preprocessing.

of (quantum) graph neural networks.

- When the prediction is on the nodes of the graph, it is common to refer to *node-level* tasks. This kind of tasks solve the problem of predicting the labels of the graph's nodes. Some high level applications include node clustering and graph cuts.
- As in the study case of this chapter, the target of the prediction can be the relationship between the graph's nodes, this scenario is commonly referred to as a *edge-level* task. This process is still often based on the node feature, which are aggregated at edge-level.
- Finally a third very common kind of task considers the graph as a whole, i.e. *graph-level* tasks. In this case information is still propagated through the graph, but the final goal is to predict properties of the whole structure. Many tasks that treat molecules as graphs belong to this class, when chemical properties of the whole object are to be inferred.

### 5.3.1 Graph operations

The literature on what can be achieved manipulating information on a graph is extremely broad [133][134]. The goal of this section is to provide the essential formalism to understand the role of graph neural networks in the manipulation of graph-based information [54].

The most natural matrix that can be defined on a graph is the *adjacency*:

$$A(i, j) = \begin{cases} 1 & \text{if } (i, j) \in \mathcal{E} \\ 0 & \text{otherwise} \end{cases} \quad (5.1)$$

this matrix contains the connectivity of the graph  $\mathcal{G}$ ; if nodes  $i$  and  $j$  are connected then the matrix entry  $A_{i,j}$  is one, and zero otherwise. For this reason, in undirected graphs, like the ones of this chapter<sup>1</sup>,  $A$  is a symmetric matrix.

In the context of edge-level tasks and GNNs it is also convenient to introduce *incident matrices*, as in [20]:

$$R_i^{jk} = \begin{cases} 1, & \text{if } k^{th} \text{ edge is input of } j^{th} \text{ node} \\ 0, & \text{otherwise} \end{cases} \quad (5.2)$$

$$R_o^{jk} = \begin{cases} 1, & \text{if } k^{th} \text{ edge is output of } j^{th} \text{ node} \\ 0, & \text{otherwise} \end{cases} \quad (5.3)$$

where to define the  $k^{th}$  edge it is assumed that the set of edges  $\mathcal{E}$  is ordered.

These matrices in Eq.(5.2) and Eq.(5.3) are used to, respectively, address the incoming and outgoing edges, given a node  $j$ .

This essential introduction ends with a brief mention to two other important matrices which will be relevant in Chapter 6. The *degree* matrix  $D$  is the diagonal matrix where the entries are:

$$d_i = \sum_{j=1}^n A_{ij} , \quad (5.4)$$

which represents the number of connections of each node  $i$ .

Finally, the *unnormalized graph Laplacian*, also sometimes called *combinatorial Laplacian* to highlight its action on the discrete domain of the graph's nodes, is introduced as:

$$L = D - A \quad (5.5)$$

The graph Laplacian is the foundation of spectral graph theory, which is the characterization of the properties of graphs, starting from the eigenvalues and eigenvectors of their associated Laplacian.

---

<sup>1</sup>Some tests have also been performed on directed graphs, but with lower general performance.

### 5.3.2 Graph building from hits point cloud

A particle graph, like the one on the right of Fig. 5.3, is built starting from the hits' positions in each dataset event, and after the cuts discussed in Section 5.2.1, which are performed using the event's ground truth.

Hits are grouped according to the tracker layer from which they originate. A graph edge connecting two hits is generated whenever two hits belonging to adjacent layers are geometrically compatible.

In this research the geometrical constraints are aligned with the ones of the original proposal [20], and the classical HEP.TrkX project [135] from which it originates. The coordinates of each hit are expressed in a cylindrical reference system  $(r, \phi, z)$ , where  $r$  is the radial distance from the beam axis  $z$ , and  $\phi$  is the azimuthal angle. For each hit pair  $(h_1, h_2)$  the following quantities are computed and constrained:

- $\phi_{\text{slope}} = \Delta\phi/\Delta r$
- $z_0 = z_1 - r_1\Delta z/\Delta r$

with cuts

- $|\phi_{\text{slope}}| < 6 \times 10^{-4}$
- $|z_0| < 100 \text{ mm}$

with no constraints in pseudorapidity  $\eta$  and  $\phi$  of the single hit. These choices are particularly important in this geometric construction as they guarantee that each graph -with average number of nodes  $\bar{N}_v = (5.1 \pm 0.8) \cdot 10^3$  and number of edges  $\bar{N}_e = (7.7 \pm 1.7) \cdot 10^3$ - has on average  $54 \pm 4\%$  of true track segments. The dataset can therefore be considered balanced.

## 5.4 Graph Neural Networks

Among the different intersections of graph theory and machine learning, graph neural networks represent one of the most transversal and successful deep learning models of the past decade. GNNs are born from the idea of generalizing the convolutions performed by a convolutional neural network in euclidean domain (like a 2D pixel grid) to the non-euclidean domain of a graph [136].

The fundamental idea is that, in graph domain, notions like shift invariance find no obvious counterpart and have to be adapted in a meaningful way from first principles.

Learning on graphs as a concept has first been introduced in the context of recurrent neural networks and random walk models in [137]. Subsequent studies have

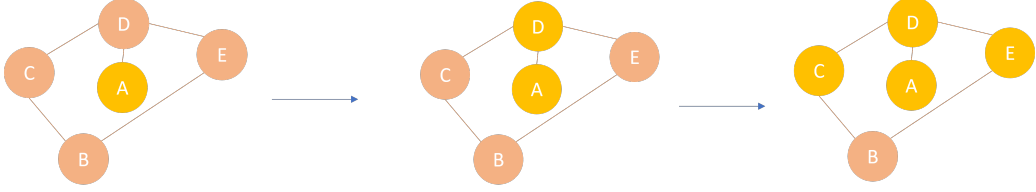


Figure 5.4: Message passing algorithm. Information from node A is first propagated to its nearest neighbor D, and then propagated to the next to nearest neighbors C and E.

however appeared much later, and generally come from two different, but intertwined approaches to graph-like domains: on one side works explicitly based on spectral graph theory [138] [139] [54], and on the other research considering graphs as a mesh, and therefore rooted in differential geometry [140] [141] [142].

Modern approaches [143] draw inspiration from both these lines. In particular the classical computing architecture used in this project is deeply inspired by project Exa.TrkX [17], and, specifically, its attention-based GNN.

There are two important features in this class of GNNs:

- message passing
- edge attention

Message passing refers to the propagation of information within the graph, and is illustrated in a simplified scenario in Fig. 5.4. Each node in the graph iteratively enriches the feature vector that describes its properties, aggregating (for example concatenating) information coming from its neighbors. This is the generalization of the convolution operation that a CNN performs on a regular grid, but on a graph domain, where the connectivity is not necessarily regular. This concept will be further explored in Chapter 6 in the context of graphs with higher regularity than the ones arising from particle collisions.

Every incoming message is weighted by a  $[0, 1]$  attention score, which is a scalar weight associated to each edge.

This double-sided mechanism guarantees that the nodes which in the initial graph are connected with each other -comprising both true and fake candidate track segments-, even at multiple levels of proximity (different detector layers in the case of graphs representing collision events), become aware of their neighborhood.

For these reasons GNNs are an example of a modern, machine-learning-based, global method to particle track building.

Information propagation happens for each node (and edge) at the same time step,

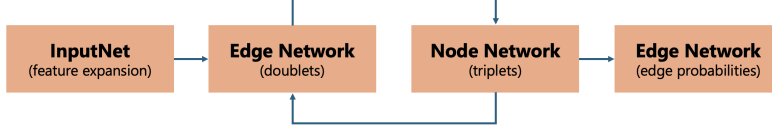


Figure 5.5: Block diagram of a graph neural network for edge classification. The iteration of Edge and Note Networks is responsible for the propagation of the information at different orders of neighborhood in the graph structure, as in Fig. 5.4

within a single global environment, rather than building the track one node at a time for each separate instance of a combinatorial Kalman filter (refer to Chapter 4 for a full description of such a pipeline). It is also worth highlighting that a full, CKF-based track reconstruction pipeline comprises many steps, from seeding, to finding, fitting, and selecting; these are not performed by a single GNN. Modern machine learning projects focused on studying and developing GNNs for particle tracking are multi-stage frameworks [17][18]. However, the central part of these pipelines is the track segment classification, performed by a graph neural network. The output of this stage is a high-accuracy set of true track segments between pairs of hits from adjacent layers (doublets). These doublets are then further aggregated in triples (three connected hits), and, finally, in full tracks, by the subsequent routines in the tracking pipeline.

The quantum machine learning research presented in this chapter targets the track-segment classification stage.

#### 5.4.1 Structure of a GNN for edge classification

The message passing and edge attention methods described in the previous section are implemented at different stages within the GNN, illustrated schematically in Fig. 5.5. The iteration between the Edge and Node network blocks, which are individually discussed in the following sections, is the mechanism responsible for the propagation of the information at an increasingly distant number of hops in the event graphs, and for this reason complements the message passing executed within the Node Network blocks.

### InputNet

A graph built according to the procedure detailed in Section 5.3.2 can be represented through three matrices. The *feature matrix*  $X \in \mathbb{R}^{N_v \times 3}$ , where each row is a tuple consisting of the three hit coordinates, and the incident matrices  $R_i, R_o \in \{0, 1\}^{N_v \times N_e}$  introduced in Eq. (5.2)-(5.3). A final ground truth vector  $y \in \{0, 1\}^{N_e}$  completes the set of preprocessing outputs.

Input features are fed to the network node-by-node for each event. Each step of the GNN acts on all the nodes (or edges) before moving to the following stage. In this sense InputNet is an MLP (see Chapter 2) which, for each row in  $X$ , expands its representation. Details on the hidden dimensions will be provided in the next sections.

### Edge Network

The first and last stage of the GNN are Edge Networks (EN). All ENs share the same learned parameters. For each edge  $e_k$  in the edge set  $\mathcal{E}$ , via the matrices  $R_o$  and  $R_i$ , the feature vectors  $b_o$ , and  $b_i$  of the two nodes at the extremities of  $e_k$  are concatenated in a new edge message vector  $B^k$ :

$$B^k = [b_o^k \oplus b_i^k] \quad (5.6)$$

where

$$b_o^k = \sum_{j=1}^{N_v} R_o^{jk} x_j, \quad (5.7)$$

and

$$b_i^k = \sum_{j=1}^{N_v} R_i^{jk} x_j. \quad (5.8)$$

Each  $B^k$  is then fed to a MLP with a single exit neuron and a sigmoid activation, which bounds the scalar feature associated to each  $e_k$  between 0 and 1. To avoid introducing unnecessary notation, from now on  $e_k$  will refer both to the label of the k-th edge of the graph, and to its scalar weight. The resulting vector of scalar edge scores represents the learned weight associated with each edge.

Track segments with score close to 1 are to be considered more likely to be true, and therefore the information of the nodes at their extremities should be propagated with higher weight during message passing.

The final stage of the GNN is an Edge Network, because the end goal of the pipeline is to get track segments' scores, and, by thresholding them, a classification of true and fake track segments. In this work the final threshold is always 0.5 .

### Node Network

The Node Network (NN) block is responsible for the single-hop message passing, where each node  $j$  concatenates its own feature vector  $x_j$ , with the ones,  $b_o$ ,  $b_i$ , from every other node with which it shares an edge, weighted by the edge score  $e$ . Formally:

$$x'_j = [x'_{j,\text{outgoing}} \oplus x'_{j,\text{incoming}} \oplus x_j] \quad (5.9)$$

$$x'_{j,\text{incoming}} = \sum_{k=1}^{N_\varepsilon} e_k R_i^{jk} b_o^k \quad (5.10)$$

$$x'_{j,\text{outgoing}} = \sum_{k=1}^{N_\varepsilon} e_k R_o^{jk} b_i^k \quad (5.11)$$

As for the previous EN, the concatenated feature vector matrix  $X'$  is then fed row-by-row to an MLP.

## 5.5 Quantum Graph Neural Networks

First introduced in [20], Quantum Graph Neural Networks (QGNN) extend the GNN formalism into the domain of quantum machine learning by implementing parts of the architecture as parameterized quantum circuits (PQC). The core idea is to compare, and potentially enhance, the representational power of a classical multi-layer perceptron used in standard GNNs, alternating classical MLPs blocks with a single PQC as central component of both Edge and Node Networks. PQCs (also referred to as Quantum Neural Networks) have been extensively studied in the past decade, and many open questions remain, as detailed in Chapter 3.

The research questions that this study, as the original one, sets itself to address are:

- 1: *Can a hybrid quantum graph neural network learn relationships in classical data from tracking as good as -or better- than a classical GNN?*
- 2: *What constraints do current NISQ hardware and software frameworks impose on the tasks that quantum machine learning can realistically perform in the context of experimental high energy physics?*
- 3: *Can the original architecture in [20] be upgraded to better fit the particle tracking task?*

To provide answers to these questions this project was developed in two phases, presented in the following discussion. Phase I comprises the software development

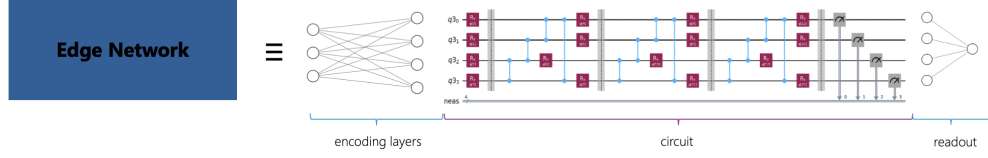


Figure 5.6: Structure of the Edge Network in a QGNN, with encoding and readout layers to manage the dimensionality of the data, and a quantum circuit in between classical networks.

of a new, up-to-date version of the QGNN model, a study at increasing levels of particle pileup, and a final test on quantum hardware, performed in the first quarter of 2024. Phase II builds on the results and new evidence gathered in the previous phase and presents the development of an upgraded QGNN architecture, together with a final characterization.

### 5.5.1 Phase I: development and pile up studies on the original architecture

In the first phase of this project the original implementation of the QGNN in [20] was completely reimplemented and updated to use modern frameworks for both the classical machine learning components, and the integration with quantum circuits. The original study used an integration of TensorFlow, TensorFlow Quantum [144], and Cirq [145]. Instead two new versions have been developed: a first PyTorch - Qiskit [146] version was initially tested, but due to the strong limitations in the Qiskit quantum machine learning interface `TorchConnector` (such as the lack of GPU-accelerated quantum simulation and slow integration of back-propagation through the classical MLPs and simulated circuits), a second and final implementation was developed with Jax [147] and Flax [148] as the machine learning framework, and PennyLane [149] for the quantum neural networks.

In this phase the underlying architecture is kept consistent with the original proposal. In particular the Phase I QGNN exploits the same structure of the GNN presented in Section 5.4, integrating the parametric quantum circuits inside the Edge and Node Network blocks as depicted in Fig. 5.6. While in a purely classical GNN the network itself is a fully connected deep neural network, in this QGNN the dense layers are alternated with parametric quantum circuits, one for each

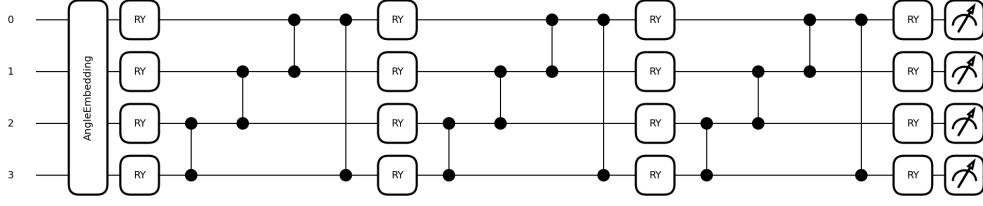


Figure 5.7: The 3-layers and 4-qubits parametric quantum circuit used as foundation to the original hybrid classical-quantum layers of the QGNN. Information is encoded in the initial quantum state through angle encoding.

Edge and Node network.

The encoding and readout layers adjust the number of features to be fed into or read out from the circuit. In this phase, tailored to the original architecture, the number of hidden neurons of the single layer classical networks is 4, to match the number of qubits. The circuit itself consists of three stages. The first stage is the *Information Encoding Circuit* (IEC), which is responsible for injecting classical features into a quantum states. In Phase I the QGNN uses angle encoding on 4 qubits: the classical features, output of the encoding MLP, are each associated to single-qubit rotation gates. This encoding limits the depth of the IEC, but potentially requires a large number of qubits.

The quantum state is directly fed to a parametric quantum circuit with learnable rotation angles. The circuit, as in Fig. 5.7, characterized in [150], is chosen to reproduce the original proposal, and for this reason is composed of 3-layers.

Finally, classical information is extracted through measurement gates and fed to a readout network with a single output neuron in the case of Edge Networks, and 4 neurons for Node Networks.

### Training characterization

The characterization of the QGNN is performed through the creation of three new datasets, simulating increasing pileup values  $\mu = 50, 100, 150$ , in addition to the original dataset at  $\mu = 200$ . This is realized through a random selection of the original  $\sim 200$  primary interaction vertices.

The same QGNN architecture is trained on the four datasets and the results are shown in Fig. 5.8 for the main validation metrics. The datasets are increasingly unbalanced for decreasing pileup, due to the constraint on using the same preprocessing for the four graph-building stages. As the pileup decreases, in fact, the

geometrically available hits resulting in fake track segments also decrease.

Error bars in the plots are obtained by k-folding, a technique where the training set is split in  $k$  equal parts (folds),  $k - 1$  are then used as new training set for each of the  $k$  separate trainings. The remaining fold is, at each training, used as validation set. Averaging the  $k$  validation scores provides a reliable performance estimate.

During training, the quantum circuits are simulated on an ideal noiseless statevector simulator.

The results show that the validation accuracy decreases with increasing pileup, as expected since the events become denser. The numerical value of the accuracy plateau for the  $\mu = 200$  dataset, corresponding to high luminosity events, is slightly dependent on the choice of hyperparameters of the neural network, as the error estimation highlight. In this context the current set of hyperparameters, with learning rate 0.01, and training set consisting of 45 samples, is fixed so that one training for  $\mu = 200$  can be completed in less than one day. The development of the current software implementation has been crucial to achieve this result, overcoming the limitations of the original implementation which is reported to train in the time-frame of a week<sup>[20]</sup>.

The remaining metrics are particularly relevant, as specificity and precision show that the different models are very good at recognizing fake track segments, while recall suggests that there is a very strict classification for true track segments. This means that this architecture struggles to learn how to identify true track segments, at least with respect to this specific input dataset.

These results highlight an unexpected behavior from the original QGNN of <sup>[20]</sup>. The model at this stage is not learning to correctly classify true track segments: this is inferred from the low value of the recall metric for  $\mu = 200$ , which in turn provides an interpretation for the low accuracy score on the same dataset.

### Inference results

Another goal of this work is the characterization of the trained QGNN models on different backends, both simulated and real quantum hardware<sup>[2]</sup>. Fig.<sup>[5.9]</sup> shows the accuracy with increasing pileup values. The results are compatible with the validation accuracy of Fig.<sup>[5.8]</sup>. For the simulated backends, tests rely on PennyLane’s support for local IBM Qiskit Aer quantum simulators. The two lines

---

<sup>2</sup>This work was supported by the PNRR MUR project ICSC under grant CN00000013-ICSC. Access to the IBM Quantum Services was obtained through the IBM Quantum Hub at CERN under the CERN-INFN agreement contract KR5386/IT.

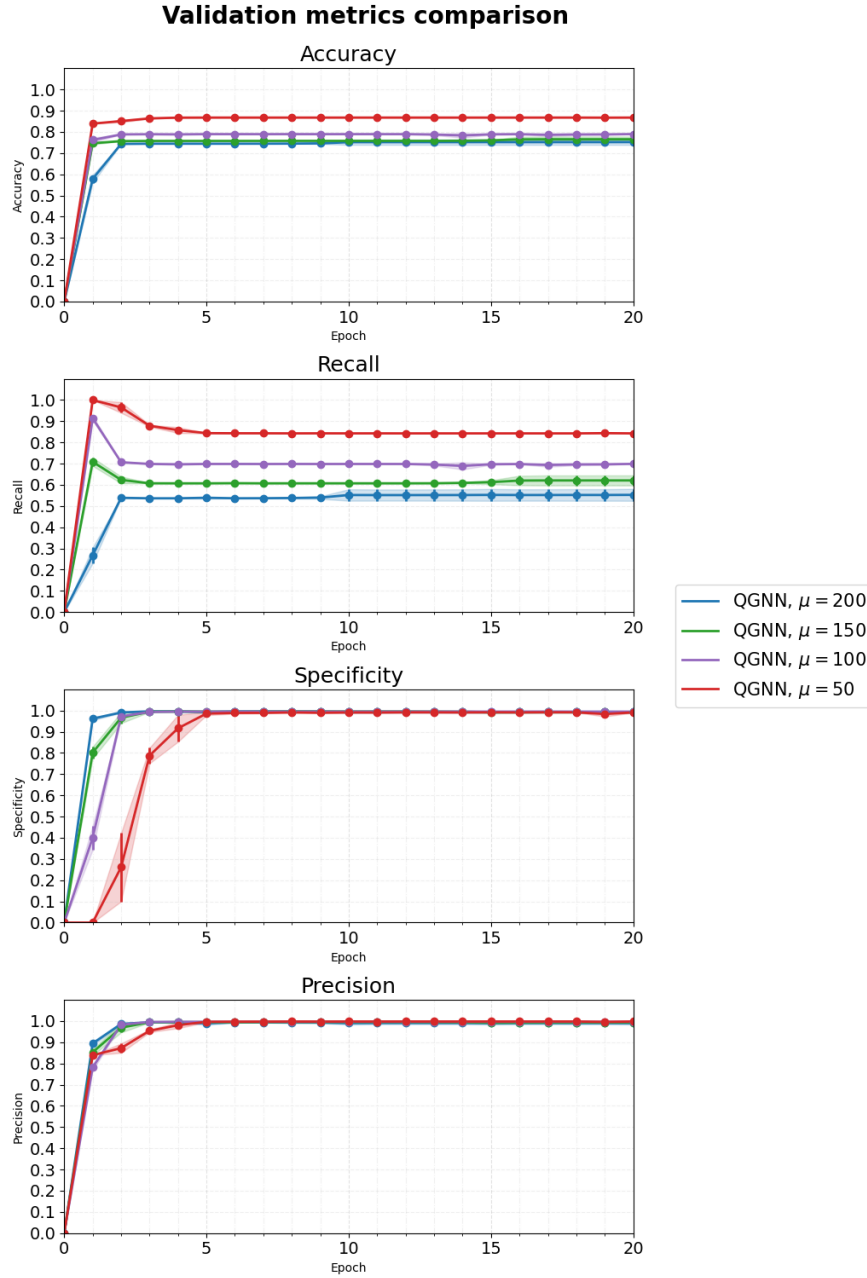


Figure 5.8: Original QGNN architecture accuracy, specificity, precision, and recall measured on the validation set, with four different dataset simulating different pileup values,  $\mu = 50, 100, 150, 200$ .

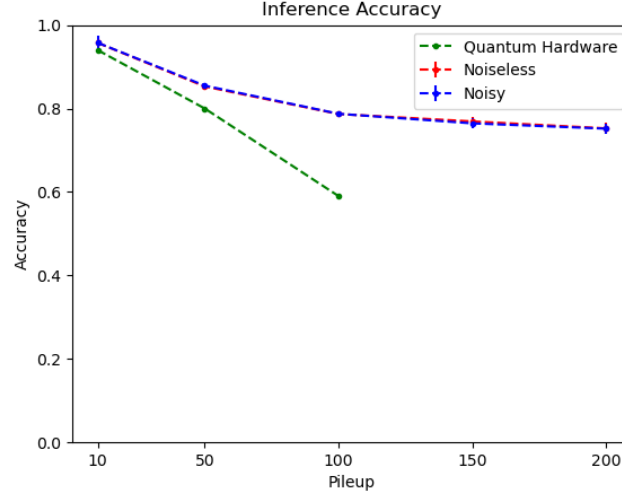


Figure 5.9: Inference accuracy for different pileup datasets. The red line corresponds to the noiseless PennyLane statevector simulator; the blue line to the noisy Qiskit Aer simulator backend, with a noise model imported from the IBM Osaka device; the green line to the real remote IBM Osaka quantum hardware, accessed through PennyLane’s interface.

representing noiseless and noisy simulated backends overlap almost completely. This behavior is compatible with the fact that the quantum circuits are limited in depth and therefore not very complex and subject to non-mitigated noise.

In the first quarter of 2024, thanks to access to IBM’s superconducting quantum hardware, the trained models have been tested up to pileup 100 on a very reduced dataset, therefore this result has to be considered only in its qualitative nature. The limitations in the tests are related to the high queuing times for submitting quantum jobs and the limited access to quantum processing time. The divergence between the green line and the simulated results can be likely attributed to two reasons: (i) the limited statistics due to the small dimension of the test dataset that can be reasonably tested on current quantum hardware, and the consequent inability to perform k-folding; (ii) the sub-optimal IBM remote hardware support offered by PennyLane at the time of testing<sup>3</sup>.

<sup>3</sup>This issue has been dealt with in the meantime, but access to quantum hardware was no longer available for this project.

### 5.5.2 Phase II: upgrades to the original QGNN model

Phase I of this project has revealed the practical limitations of the original QGNN architecture. The low accuracy plateau reached on the pileup 200 dataset, consistent with the results in the original study [20] (small differences, in the order of the percent, can be linked to minor differences in the preprocessing step), indicate that the original QGNN architecture does not model the track classification with sufficient accuracy.

At the start of this second phase the critical aspects motivating an upgrade can be grouped in three classes:

- **Dataset:** Particle tracking on a realistic dataset is extremely resource-demanding in the context of NISQ era quantum machine learning. The presence of classical MLPs can mask some of this complexity.
- **GNN:** The underlying classical GNN in the original proposal is not state of the art anymore, and upgrades are required, taking into account improvements in classical GNN-based tracking pipelines such as those in Refs. [17][18][19].
- **QGNN:** The full quantum-classical pipeline is limited by bottlenecks in the classical-quantum interfaces.

#### GNN upgrades

To answer the initial research questions in a meaningful way, the cuts on the TrackML dataset are kept unchanged with respect to Phase I. The goal of the following upgrades is to improve learning from the same  $\mu = 200$  dataset, in order to understand what the classical and quantum parts of the network are providing in this process.

The upgrades to the classical GNN involve two main aspects:

- introducing residual connections in the forward data flow;
- testing larger MLPs before the quantum circuits, and characterizing the implications of this size mismatch between the classical neural networks and the quantum circuits.

Residual connections improve training and stability of deep message passing models, by explicitly preserving information across different layers. A residual block learns a correction function  $R$  such that, denoting by  $H^{(k)}$  the hidden representation at layer  $k$ :

$$H^{(k+1)} = H^{(k)} + R\left(H^{(k)}\right), \quad (5.12)$$

which (i) provides an explicit identity path so the network can default to the simple non-transformed branch when deeper transformations are unnecessary [151], (ii) improves gradient flow because derivatives can propagate in the additive identity branch [152], and (iii) smooths the loss function landscape, enabling substantially deeper stacks without degradation [153].

These improvements, historically first introduced in deep feedforward neural networks, are beneficial also for graph neural networks. In fact, residual connections counteract feature oversmoothing, happening in the process of propagation, through the reinjection of pre-message-passing features. This way node representations do not lose significance as depth grows.

A similar, but complementary, method used with respect to the original network, and inspired by [17] is an initial residual. Each node network output concatenates the original  $H(0)$  with the current hidden state before the residual add; this preserves raw features that might otherwise be attenuated by repeated neighborhood mixing.

These upgrades alone do not improve the scores of the GNN, and need to be complemented by a wider and deeper MLP, in place of the simple encoding layers of Fig 5.6. In particular, referring to the studies by [17] and [18], a 2-hidden units MLP is introduced before the circuits, with 64 neurons per layer. Further tests with double the number of neurons have not shown significant advantage.

The validation results of the new GNN architecture are shown, together with the results from next section, in Fig 5.10. All the metrics for the upgraded Classical GNN exceed 0.95, and suggest that the learning limitations observed in the previous model are effectively addressed in the classical network.

### QGNN upgrades

The increase in the sizes of the MLPs of the Edge and Node Networks, detailed in the previous sections, leads to a nontrivial interplay with the quantum circuits in the full QGNN architecture. In Phase I, in fact, 4-qubits quantum circuits with angle encoding of the classical information were used, in accordance with the limited dimension of the MLPs. After the upgrade to the classical architecture, however, the new encoding MLPs produce 64-dimensional feature vectors.

A first characterization has been carried out directly on small sized circuits, with an encoding MLP with hidden units sized (64,64,4). This results in the same plateaus of Phase I, indicating that the feature reduction, from 64 to 4, is discarding too much information. The pipeline trains, but again fails to learn the task, in this case due to the reduced dimensions of the quantum circuit.

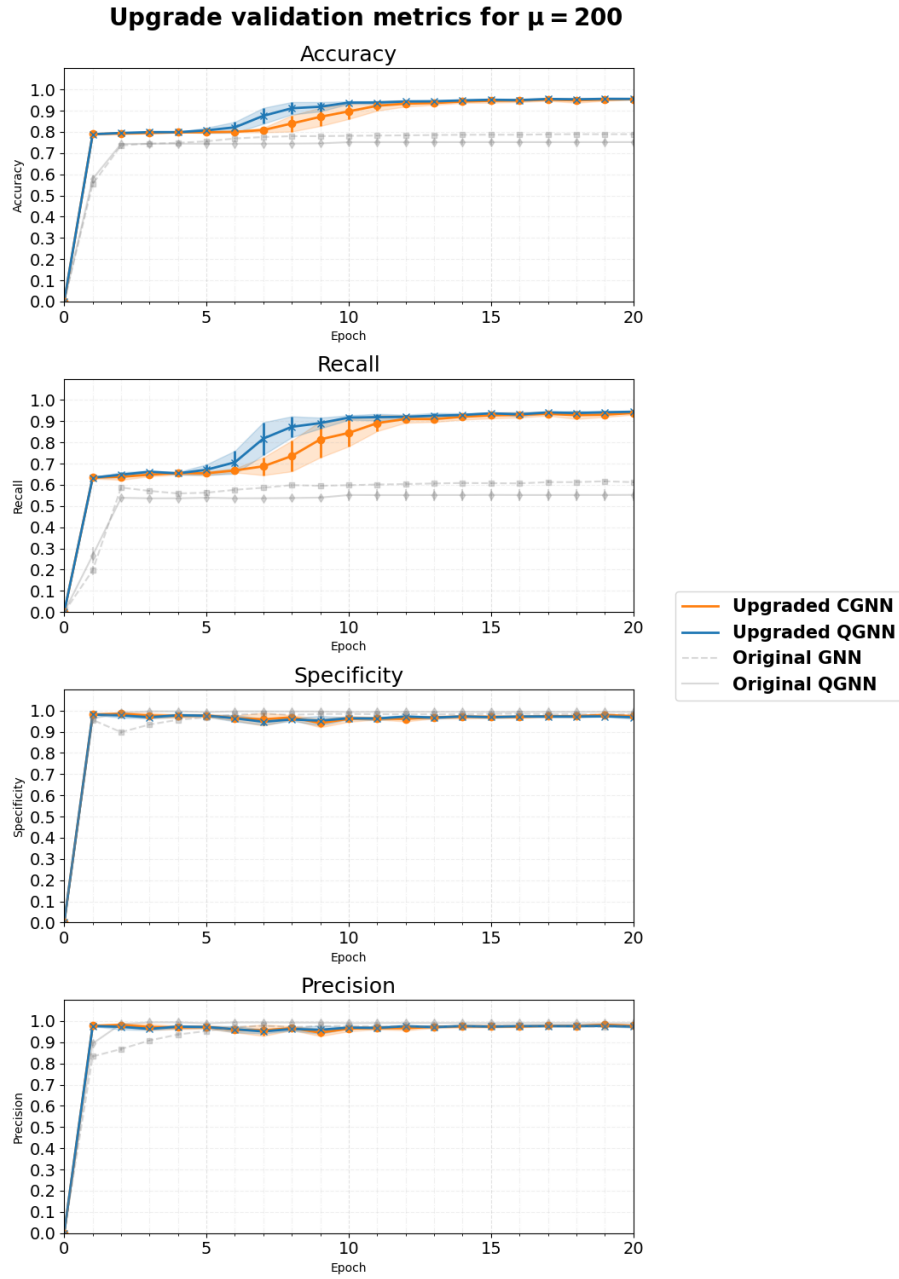


Figure 5.10: Comparison of the learning metrics on the  $\mu = 200$  validation set. Blue lines represent data from the upgraded Quantum-GNN architecture, with [64,64] classical MLPs and 6-qubits quantum circuit with amplitude encoding. Orange lines correspond to data from the upgraded Classical-GNN, with [64,64] MLPs. Grey lines represent the original pre-upgrade architecture scores.

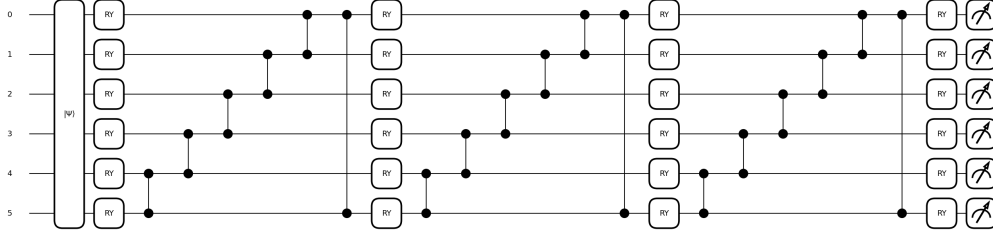


Figure 5.11: The 3-layers and 6-qubits parametric quantum circuit used in the upgrade. The 64 features from the classical MLPs are encoded in the quantum state  $|\psi\rangle$  through amplitude encoding; 64 output amplitudes are then extracted from the circuit.

To circumvent this problem two different paths have been considered:

- 1 Increase the number of qubits while keeping an angle encoding IEC.
- 2 Switch to an amplitude encoding scheme.

The first strategy is conceptually straightforward but severely constrained by the cost of simulating the QGNN. Tests have been performed up to 12 qubits, which have proved to be still not enough to avoid the information loss starting from the 64 features of the MLPs. This constraint comes from two counteracting effects: simulating a single circuit of 12 qubits is extremely fast, but this QGNN operates at edge-level, so for each event (graph) each node network creates  $N_V$  quantum circuits, similarly each edge network creates  $N_E$  circuits. The number of message passing operations is fixed to  $N_{\text{iter}} = 3$  and this amounts to 7 total ENs and NNs, per graph. The total number of circuits to be simulated for graphs sized  $N_V \simeq 5 \cdot 10^3$ , and  $N_E \simeq 8 \cdot 10^3$  is of order  $N_{\text{tot}} \simeq 1 \cdot 10^5$  per event. Caching intermediate results for the backpropagation becomes easily unfeasible for circuits with more than 12 qubits. Similarly, avoiding the majority of the caching results in a very rapid rise in training times, of the order of weeks for meaningfully sized datasets.

For these reasons this architecture adopts amplitude encoding of the classical features, in the  $2^{N_{\text{qubits}}}$  complex amplitudes of a system of  $N_{\text{qubits}}$ . The quantum circuit (Fig. 5.11) is an adaptation of the original circuit in Fig. 5.7, now extended to  $N_{\text{qubits}} = 6$ , in line with the 64-feature vectors.

The results of the tests on this upgraded QGNN architecture are shown in Fig. 5.10. Accuracy and recall show drastic improvements with respect to the same metrics on the Phase I QGNN of Fig. 5.8 for the  $\mu = 200$  dataset. The upgraded QGNN is able to match, within uncertainties, the performance of the upgraded classical

GNN. Comparisons with the original classical and quantum architectures show an overall improvement of both the upgraded classical and upgraded quantum models, both in terms of convergence within the two models, and in the absolute scores.

Additionally, the upgraded QGNN displays a noticeable earlier convergence with respect to the upgraded CGNN. This behavior emerges similarly in the training curves reported in Appendix [B](#).

The total number of classical trainable parameters is the same for both the architectures, full classical and hybrid quantum, and equal to  $N_{\text{classical}} = 29505$ . The total number of trainable rotation angles in the quantum circuits of the QGNN is  $N_{\text{quantum}} = 44$ . This pronounced mismatch is an inevitable feature of a NISQ hybrid QGNN designed for edge classification on graphs from realistic events at  $\mu = 200$ , because much of the learning in this NISQ regime has to be offloaded to the more scalable classical MLPs.

Nonetheless, the difference in convergence introduced by the presence of a quantum circuit, however limited in depth, suggests that quantum neural networks, or classical architectures inspired by them, may provide useful effects such as regularization or alternative inductive biases. Further studies will be required to understand how dataset and preprocessing dependent is this effect.

## 5.6 Conclusions and acknowledgments

The research presented in this chapter illustrates the delicate interplay between deep neural networks and quantum neural networks, in the context of a quantum graph neural network for edge classification. This integration is particularly challenging when the input dataset belongs to a complex environment such as particle collisions at high luminosity.

The characterization, through increasing pileup studies, performed in the first phase of this project, has led to a better understanding on the limitations of the original proposals regarding QGNNs for edge classification. In particular the software development carried out during this first phase has built a solid foundation for the upgrades of the subsequent phase.

In the second phase of the project an upgraded version of the QGNN is characterized. The results show a clear improvement relative to the metrics of the original QGNN, and demonstrate that the upgraded hybrid model can match, within uncertainties, the performance of a classical GNN on the same task. This work therefore constitutes one of the first systematic studies of QGNNs for realistic particle tracking, and provides a concrete performance benchmark for future

developments in quantum-assisted tracking.

The results also indicate that the underlying classical part of the architecture is responsible for the majority of the learning. This is an expected behavior for a NISQ architecture used in a real-world task. There are, however, indications of a possible regularization effect introduced by the parametric quantum circuit during the learning phase, which results in a faster convergence of the QGNN, and motivates further research on hybrid GNNs.

On a computational perspective, this study provides a concrete empirical evidence for the importance of research in the integration between classical and quantum technologies in HPC centers, together with the development of optimized software frameworks for heavy-load quantum simulations.

For all these reasons, although tasks with the complexity and intrinsic size of particle tracking in HEP are still very challenging for NISQ QML algorithms, they provide nonetheless useful insight in the integration between HEP-related machine learning methods and quantum technologies.



# 6

## Object tracking with non-abelian quantum Fourier methods

### Contents

---

|                                                                             |            |
|-----------------------------------------------------------------------------|------------|
| <b>6.1 Identity management and object tracking</b>                          | <b>95</b>  |
| <b>6.2 Representations of the Symmetric group</b>                           | <b>96</b>  |
| 6.2.1 Irreducible representations                                           | 100        |
| 6.2.2 Regular representation                                                | 101        |
| 6.2.3 Irreducible decomposition of the symmetric group                      | 102        |
| <b>6.3 Group Fourier transform</b>                                          | <b>104</b> |
| 6.3.1 Fourier transform for $\mathbb{Z}_n$                                  | 105        |
| 6.3.2 Fourier transform for finite non-commutative groups                   | 107        |
| <b>6.4 Classical algorithms for identity management on <math>S_n</math></b> | <b>110</b> |
| 6.4.1 Noise models                                                          | 111        |
| 6.4.2 Bayes update                                                          | 114        |
| <b>6.5 Development of the quantum algorithm</b>                             | <b>119</b> |
| 6.5.1 Towards a fully quantum pipeline                                      | 119        |
| 6.5.2 State initialization                                                  | 120        |
| 6.5.3 Quantum diffusion                                                     | 121        |

|                              |     |
|------------------------------|-----|
| 6.5.4 Bayes update . . . . . | 126 |
| 6.6 Comments . . . . .       | 128 |

---

Quantum machine learning is increasingly shifting from NISQ-era algorithms to methods projected towards forthcoming fault tolerant quantum computers. This paradigm shift is particularly important for the development of novel QML algorithms for experimental high energy physics, which is an inherently big data and high throughput domain, where the scale of the data poses significant challenges for near-term hardware.

This chapter presents the proposal and current progress in the development of a new class of quantum algorithms, rooted in probabilistic modeling over permutation-valued data, naturally represented as elements of the symmetric group  $S_n$ . These models stem from finite non-abelian group representation theory [26][154], and have since been applied in the context of classical machine learning [155][28]. In particular, symmetric group representations introduce a very convenient framework for the modeling of identity management problems, and, for this reason, a natural application of this formalism is multi-object tracking[29].

These methods address the factorial explosion in the number of possible orderings of  $n$  objects by leveraging the harmonic analysis of non-commutative groups, and by imposing spectral constraints on the models.

In this context, quantum computing can be seen as a potential new paradigm to overcome the remaining bottlenecks, which classical formulations handle only through approximate subroutines, and by restricting the number of objects: the combination of an efficient quantum Fourier transform for the symmetric group [27][9], together with the exponentially large Hilbert space of multi-qubit systems, provides an alternative route to scaling these models.

In turn, an efficient quantum algorithm for identity management has the potential to extend the applications of the classical methods for multi-object tracking to the high-multiplicity environments typical of high-energy physics experiments.

This project has been developed in collaboration with, in no particular order, Maria Schuld and Vasilis Belis from the Xanadu QML team, Giulio Crognaletti from CERN and the University of Trieste, and Michele Grossi from CERN QTI.

At the time of writing the project is well advanced, but not yet complete. This chapter presents the general structure of the algorithm, its foundation, the details on the diffusion over irreducible representations of  $S_n$ , and lays the groundwork for the Bayesian update step, which defines the next stage of the project.

The Chapter is organized as follows: Section [6.1](#) provides an introduction identity management and object tracking, as the underlying general applications of the methods presented in the rest of the chapter. Section [6.2](#) presents the general foundation to group representation theory for finite groups, specializing the formalism to the symmetric group  $S_n$ . Section [6.3](#) expands on this formalism, showing the link with a generalized notion of Fourier transform. These concepts are then applied, first in Section [6.4](#) with the presentation and characterization of two classical algorithms that exploit this theory to perform object tracking, and then in Section [6.5](#) which details the quantum computing algorithm, and future directions of this research.

## 6.1 Identity management and object tracking

Evolving the probability distribution over the possible orderings of a set of  $n$  elements  $\Sigma = \{1, 2, \dots, n\}$  is a notoriously hard classical problem, owing to the  $n!$  scaling of the number of their permutations. In object tracking systems this challenge is generally modeled through probabilistic filters, often derived from Kalman-type algorithms [\[156\]](#) [\[157\]](#).

In this context, [\[158\]](#) introduces the identity management (IM) problem as the evolution of a belief state for the association of object tracks and object identities, accounting for mixing events and noisy observations [\[28\]](#). The typical scenario is that of two or multiple objects moving in space, as in [Fig. 6.1](#). It is generally possible to follow their motion up until either they get very close, or observations become unavailable [\[29\]](#). When these mixing events occur, the exact association between objects and trajectories might get confused, and, to model this uncertainty, from the deterministic domain it is necessary to switch to a probabilistic description.

Identity management and trajectory reconstruction are two very related, but distinct, concepts. The former is about the association of objects (particles in our case) and tracks, the latter about building the tracks themselves. The interplay between the two is very context-dependent, and, while efficient IM could be used to perform track reconstruction via iterative layer-by-layer global track building, IM as an enhancement tool for Kalman filters has already been studied in [\[159\]](#) [\[160\]](#). These previous works, however, often restrict their action to  $n^2$ -sized matrices expressing first order associations of the kind "particle  $i$  belongs to track  $j$ ", and do not model a probability distribution over global permutations. In contrast, higher order associations — joint statements about the identities of multiple objects simultaneously — are crucial to capture consistent relative relationships among objects, particularly in complex mixing scenarios. This gives a

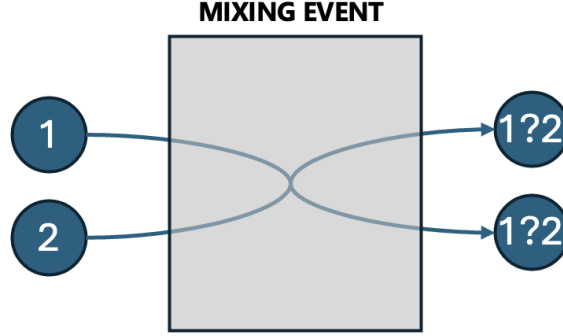


Figure 6.1: Representation of identity confusion after a mixing event. Objects 1 and 2 start from a well defined configuration  $[1,2]$ , but their proximity and the data at disposal for the reconstruction makes it hard to distinguish if the final configuration is  $[1,2]$ , or  $[2,1]$ .

strong motivation to explore the possibility of performing efficient identity management with  $O(poly(n))$  resources, exploiting the full formalism offered by the representations of the symmetric group  $S_n$ , together with quantum processing.

## 6.2 Representations of the Symmetric group

This section serves as an introduction to group representation theory, and specifically to non-commutative Fourier analysis. This is the underlying theoretical foundation for the definition of probability distributions on elements of the symmetric group, and for the operators that act on such functions. Most of the elementary group theory used for this project and presented in this chapter comes from [26] and [161].

**Definition 1** (Group). A group  $G$  is a non-empty set equipped with a binary operation  $\cdot$ , closed under its action:  $G \cdot G \rightarrow G$ . Thus  $\forall g_1, g_2 \in G$ ,  $g_1 \cdot g_2 \equiv g_1 g_2 \in G$ , and the following properties hold:

- **Identity element:**  $\exists e \in G$  such that  $e \cdot g = g \ \forall g \in G$ ;
- **Associativity:**  $(g_1 \cdot g_2) \cdot g_3 = g_1 \cdot (g_2 \cdot g_3) \ \forall g_1, g_2, g_3 \in G$ ;
- **Inverse element:**  $\forall g \in G$ ,  $\exists g^{-1} \mid g \cdot g^{-1} = g^{-1} \cdot g = e$ . The element  $g^{-1}$  is called the inverse of  $g$ .

If the underlying set is finite, the group itself is called *finite*, as opposed to *infinite groups*. The number of the group elements is called group *cardinality* or *order*, and denoted  $|G|$ . In this chapter the focus will be on the symmetric—or permutation—group  $S_n$ , which is finite with  $|S_n| = n!$ .

**Definition 2** (Group representation). Let  $G$  be a group, and  $V$  a vector space over some field  $\mathbb{F}$ ; a (linear) representation of  $G$  on  $V$  is a group homomorphism:

$$\rho : G \rightarrow GL(V) \quad (6.1)$$

where  $GL(V)$  is the group of invertible linear maps  $V \rightarrow V$ .

This formal definition introduces  $\rho$  as a map from the group to a space of linear operators such that:

$$\rho(e) = \mathbb{1} \quad (6.2)$$

where  $\mathbb{1}$  is the identity operator on  $V$ , and:

$$\rho(g_1)\rho(g_2) = \rho(g_1g_2) \quad (6.3)$$

for every  $g_1, g_2 \in G$ . In other words, representations preserve the algebraic structure of the group.

The dimension of the underlying vector space  $V$  is also the dimension of the representation, denoted  $d_\rho$ .

It is often convenient to work with *matrix representations*, i.e. to associate to each group element a matrix  $\rho(g) \in \mathbb{F}^{d_\rho \times d_\rho}$ . Often this field is  $\mathbb{C}$  or  $\mathbb{R}$ , the choice depends on the required underlying algebraic structure. For  $S_n$  it is often possible to work with real representations [162](#).

Specializing the formalism to  $S_n$ , which consists of all the  $n!$  permutations of the set of  $n$  elements, it is common to label its group elements as  $\sigma$ , written in one-line notation as:

$$\sigma = \left[ \begin{array}{cccc} \sigma(1) & \sigma(2) & \dots & \sigma(n) \end{array} \right]. \quad (6.4)$$

This notation is useful to associate to each  $\sigma(i)$  some value  $j$ . E.g. in  $S_3$ ,  $\sigma = [231]$  can be read as  $\sigma(1) = 2$ ,  $\sigma(2) = 3$ ,  $\sigma(3) = 1$ . Notice that each element of the set  $\{1, 2, 3\}$  appears just once. The permutation  $[123]$  is the identity of  $S_3$ , and for each permutation there is always the inverse one, e.g.  $[312]$  is the inverse of  $[231]$ . This is shown for  $S_3$  just out of convenience, but is easily generalizable to  $S_n$ .

The composition law for the permutation group reads:

$$(\sigma_1\sigma_2)(i) = \sigma_1(\sigma_2(i)), \quad (6.5)$$

so that in one-line notation:

$$\sigma_1\sigma_2 = \left[ \begin{array}{cccc} \sigma_1(\sigma_2(1)) & \sigma_1(\sigma_2(2)) & \dots & \sigma_1(\sigma_2(n)) \end{array} \right] \quad (6.6)$$

Another common way of labeling permutations, which will become useful to compactly indicate group elements, is cycle notation. In this context a cycle is written as:

$$\sigma = \left( \begin{array}{cccc} i & j & k & \dots & l \end{array} \right). \quad (6.7)$$

and its action is given by  $i \rightarrow j, j \rightarrow k, \dots, l \rightarrow i$ . Cycles with just one element ( $k$ ) indicate that element  $k$  is fixed. For this reason writing  $\sigma = (12)(3)(4)$  or simply  $\sigma = (12)$  is completely equivalent. Cycle notation is therefore useful because, for example, a transposition between elements  $i$  and  $j$ —the operation that interchanges their respective positions— can be written simply as  $(ij)$ .

This notation can be used to exemplify some of the different representations of  $S_n$ :

- **Trivial representation:** is one dimensional, and is the simplest possible representation, mapping each group element to the scalar 1:

$$\rho_n : S_n \rightarrow \mathbb{R}^{1 \times 1}. \quad (6.8)$$

- **Alternating (sign) representation:** is also one dimensional, and maps each group element of  $S_n$  to +1 if it can be written as a composition of an even number of pairwise swaps, and -1 otherwise. For example, with  $n = 4$ , the alternating representation is usually indicated as  $\rho_{(1,1,1,1)}$ , but notice that here  $(1, 1, 1, 1)$  is not a cycle. This notation will become clear within the next few sections.
- **First-order permutation representation:** is  $n$ -dimensional and associates to each permutation  $\sigma$  a  $n \times n$  matrix, commonly labeled  $\tau_{(n-1,1)}(\sigma)$ , where the entry in position  $(i,j)$  is 1 if  $\sigma(j) = i$ , 0 otherwise. Formally:

$$[\tau_{(n-1,1)}(\sigma)]_{ij} = \mathbb{1}\{\sigma(j) = i\} \quad (6.9)$$

where  $\mathbb{1}$  is an indicator function:

$$\mathbb{1}\{P\} = \begin{cases} 1 & \text{if the statement } P \text{ is true ,} \\ 0 & \text{if } P \text{ is false .} \end{cases} \quad (6.10)$$

To exemplify this, it is common to introduce the permutation representation

for the  $3! = 6$  elements of  $S_3$  as the six  $3 \times 3$  matrices:

$$\begin{aligned}
 \tau_{(2,1)}(\epsilon) &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} & \tau_{(2,1)}(1, 2) &= \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \\
 \tau_{(2,1)}(2, 3) &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} & \tau_{(2,1)}(1, 3) &= \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \\
 \tau_{(2,1)}(1, 2, 3) &= \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} & \tau_{(2,1)}(1, 3, 2) &= \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}
 \end{aligned} \tag{6.11}$$

which correctly combine according to the group law. For instance:

$$(12)(23) = (123) \implies \tau_{(2,1)}(12)\tau_{(2,1)}(23) = \tau_{(2,1)}(123) \tag{6.12}$$

or explicitly:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{6.13}$$

which is precisely  $\tau_{(2,1)}(123)$ .

It is always possible to perform a similarity transformation on a representation  $\rho$ :

$$S : \rho(g) \rightarrow \rho'(g) = S^{-1}\rho(g)S, \tag{6.14}$$

and the two representations  $\rho$  and  $\rho'$  are said to be *equivalent*. This is essentially a change of basis, and it is possible because group representations act on linear spaces.

The concept of equivalent representations is strictly related to following important theorem:

**Theorem 1.** *Every complex representation of a finite group is equivalent to a unitary representation.*

which will not be proved in this context, but whose derivation can be found in [\[161\]](#).

Unitary representations are particularly interesting in the context of quantum computing, as they introduce transformations that can be directly implemented

in a quantum circuit.

### 6.2.1 Irreducible representations

The first step towards a spectral theory in the context of group representations is to introduce the concepts of *reducible* and *irreducible* representations.

The permutation representation of  $S_3$  given in Eq.(6.11) is a convenient starting point for this discussion. In this representation, unlike the the previous two - trivial and sign- which are 1-dimensional, it is possible to identify a structure of subspaces that are mapped into themselves by all group elements.

More generally, fixing a basis  $\{e_1, \dots, e_n\}$  of  $\mathbb{R}^n$ , acting as the underlying vector space, the  $\tau$ -representations of Eq.(6.11) act on these basis vectors as

$$\tau(\sigma)e_j = e_{\sigma(j)}, \quad (6.15)$$

and their entries are  $\tau(\sigma)_{ij} = \delta_{i\sigma(j)}$ .

It is possible to identify as a 1-dimensional subspace the trivial  $\mathbb{1}$ , spanned by  $e_1 + e_2 + \dots + e_n$ . Its orthogonal complement is the subspace  $W$ :

$$W = \left\{ x \in \mathbb{R}^n : \sum_i x_i = 0 \right\} \quad (6.16)$$

Therefore,

$$\mathbb{R}^n = \mathbb{1} \oplus W \quad (6.17)$$

and for every  $\sigma \in S_n$ , and each of the subspaces  $U = \{\mathbb{1}, W\}$  it holds that

$$\tau(\sigma)U \subseteq U \quad (6.18)$$

The existence of these invariant subspaces means that the representation  $\tau$  is *reducible*.

In contrast, when there is no non-trivial invariant subspace  $W \subset V$  a representation is said to be *irreducible*. In the case of the permutation representation, the restriction of  $\tau$  to  $W$  can be proved to define a 2-dimensional irreducible representation  $\rho_{n-1,1}$ . In general  $S_3$  has three irreducible representations: the trivial one, the alternating, and the 2-dimensional; however in the reducible permutation representation only the trivial and 2-dimensional appear.

In conclusion of this section it is important to mention the following theorem:

**Theorem 2.** *Every representation of a finite group  $G$  is a direct sum of irreducible representations.*

When a generic reducible representation  $\rho$  is expressed as the direct sum of its irreducible representations it is said to be *completely reducible*. In a suitable basis, the matrices of the general  $\rho$  are block diagonal, where each block is a sub-matrix associated to an irreducible representation.

### 6.2.2 Regular representation

From the previous list of representations for  $S_n$  there is one more very notable case that expands on the results for trivial, alternating, and first-order representations: the *regular representation*.

The regular representation  $\rho_{\text{reg}}$  is  $|G|$ -dimensional; for the symmetric group this means it is  $n!$ -dimensional. To introduce it, it is customary to consider the  $n!$  group elements of  $S_n$  as forming an orthonormal basis  $\{e_g : g \in G\}$  for the vector space of the group algebra  $\mathbb{C}[G]$ , with  $\langle e_x, e_y \rangle = \delta_{x,y}$ .

**Definition 3.** The group algebra  $\mathbb{K}[G]$  for a field  $\mathbb{K}$  and a group  $G$  is the set of all the linear combinations of a finite number of elements:

$$\sum_{g \in G} a_g e_g, \quad (6.19)$$

with coefficients  $a_g \in \mathbb{K}$ , and basis elements  $e_g$ <sup>1</sup>, with a multiplication operation, canonically defined on the basis elements by extending the group law:  $e_x e_y = e_{xy}$ .

Then

$$[\rho_{\text{reg}}(x)]_{z,y} = \begin{cases} 1 & \text{if } z = xy, \\ 0 & \text{otherwise,} \end{cases} \quad x, y, z \in G \quad (6.20)$$

so that, at the level of basis vectors:  $\rho_{\text{reg}}(x)e_y = e_{xy}$ .

The regular representation matrices are therefore of dimension  $n! \times n!$  for every group element of the symmetric group. This is generally prohibitive to handle even for small  $n$ .

In general, the regular representation is reducible, and its structure is extremely rich: it is possible to show that every and each irreducible representation of  $S_n$  appears at least once in the decomposition. This is detailed in the next section.

The decomposition of the regular representation is the starting point of the spectral analysis for  $S_n$ , and underlies the construction of the non-commutative Fourier

---

<sup>1</sup>Sometimes notation can be confusing, and the basis elements of the algebra  $e_g$  can be indicated simply as  $g$ .

transform on the symmetric group.

### 6.2.3 Irreducible decomposition of the symmetric group

In the previous section irreducible representations have been introduced as the elementary blocks in which reducible representations can be decomposed. This section formalizes the general structure of the irreducible representations for  $S_n$ , which in the remainder of this chapter underlies the non-commutative Fourier analysis at the foundation of this work. The discussion starts from the concepts of partition, Young diagrams, and Young tableaux in the context of the permutation group.

#### Partitions and Young diagrams

A partition of a positive integer  $n$  is a tuple of positive integers:

$$\lambda = (\lambda_1, \lambda_2, \dots, \lambda_k) \quad (6.21)$$

such that:

$$\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_k > 0 \quad \text{and} \quad \sum_{i=1}^k \lambda_i = n. \quad (6.22)$$

The standard notation to denote that  $\lambda$  is a partition of  $n$  is:  $\lambda \vdash n$ . A useful graphical tool to depict partitions are Young diagrams, which are box diagrams where each row  $i$  is composed by  $\lambda_i$  boxes.

Consider for example  $S_3$ ; for  $n = 3$  the partitions are  $(3)$ ,  $(2, 1)$ , and  $(1, 1, 1)$ , associated to the Young diagrams:

$$(3): \begin{array}{|c|c|c|} \hline \square & \square & \square \\ \hline \end{array} \quad (2,1): \begin{array}{|c|c|} \hline \square & \square \\ \hline \square & \\ \hline \end{array} \quad (1,1,1): \begin{array}{|c|} \hline \square \\ \hline \square \\ \hline \square \\ \hline \end{array}$$

Similarly, for  $S_4$  the partitions can be obtained from the ones of  $S_3$ , and adding a box in every possible arrangement that doesn't violate Eq. (6.22):  $(4)$ ,  $(3, 1)$ ,  $(2, 2)$ ,  $(2, 1, 1)$ ,  $(1, 1, 1, 1)$ .

#### Dimension of the irreducible representations

The irreducible representations (shortly named *irreps*) of  $S_n$  are in one-to-one correspondence with partitions  $\lambda \vdash n$ . The irrep associated with the partition  $\lambda$  is in fact labeled  $\rho_\lambda$ , and its dimension  $d_\lambda = \dim \rho_\lambda$  can be described combinatorially in terms of *standard Young tableaux* of shape  $\lambda$ .

**Definition 4** (Standard Young tableau). Let  $\lambda \vdash n$  and let  $Y(\lambda)$  be its Young diagram. A *standard Young tableau* of shape  $\lambda$  is a filling of the boxes of  $Y(\lambda)$  with the integers  $1, 2, \dots, n$ , such that the entries strictly increase along each row and each column.

A fundamental result is that the dimension  $d_\lambda$  of the irrep  $\rho_\lambda$  is equal to the number of standard Young tableaux of shape  $\lambda$ .

The usual simple example of  $S_3$  can be used to illustrate this:

- for  $\lambda = (3)$  there is only one possible way to fill the Young diagram: 

|   |   |   |
|---|---|---|
| 1 | 2 | 3 |
|---|---|---|

 and for this reason  $d_{(3)} = 1$ .
- $\lambda = (2, 1)$  has two possible tableaux: 

|   |   |
|---|---|
| 1 | 2 |
| 3 |   |

 and 

|   |   |
|---|---|
| 1 | 3 |
| 2 |   |

 and in fact  $d_{(2,1)} = 2$
- $\lambda = (1, 1, 1)$  has one single possible filling 

|   |
|---|
| 1 |
| 2 |
| 3 |

 and  $d_{(1,1,1)} = 1$ .

These dimensions satisfy the identity:

$$\sum_{\lambda \vdash 3} d_\lambda^2 = 1^2 + 2^2 + 1^2 = 6 = |S_3| \quad (6.23)$$

which holds true for any  $n$ .

It's important to understand that this simple case is very limited, and that the number of partitions, and tableaux explode rapidly with increasing  $n$ , making once again the problem of working with the full regular representation of  $S_n$  complex to handle by a classical algorithm.

There is however a closed form expression to compute the dimension of an irrep instead of building manually the tableaux; the equation is usually introduced as *Hook-length formula*.

For each box  $c$  in the Young diagram  $Y(\lambda)$ , the *hook length*  $h(c)$  is defined as:

$$h(c) = \#\{\text{boxes to the right of } c \text{ in the same row}\} + \#\{\text{boxes below } c \text{ in the same column}\} + 1.$$

**Theorem 3** (Hook-length formula). Let  $\lambda \vdash n$  be a partition of  $n$ , and  $Y(\lambda)$  its Young diagram. The dimension  $d_\lambda$  of the irreducible representation  $\rho_\lambda$  of  $S_n$  is given by:

$$d_\lambda = \frac{n!}{\prod_{c \in Y(\lambda)} h(c)}. \quad (6.24)$$

Therefore, each partition  $\lambda \vdash n$  labels a unique irreducible representation  $\rho_\lambda$  of  $S_n$  with dimension  $d_\lambda$ , given either by the number of standard Young tableaux of shape  $\lambda$  or equivalently by the Hook-length formula.

Every finite-dimensional representation of  $S_n$  can be decomposed into a direct sum of these irreps. In particular, the regular representation decomposes as

$$\rho_{\text{reg}} \cong \bigoplus_{\lambda \vdash n} d_\lambda \rho_\lambda, \quad (6.25)$$

where each irrep  $\rho_\lambda$  appears with multiplicity equal to its dimension  $d_\lambda$ . This decomposition is the representation theoretic foundation of the non-commutative Fourier transform on  $S_n$ , which will be central to the probabilistic modeling and quantum algorithms developed in the rest of the chapter.

### 6.3 Group Fourier transform

The goal of this section is to introduce the concept of *group Fourier transform* (GFT), which generalizes the more common ones in the space of functions on a unit circle, on the real line, or on a discrete set of integers.

Presenting the full set of theorems supporting the definition of the group Fourier transform would require introducing a substantial amount of algebra and harmonic analysis, which lie beyond the scope of this thesis, an excellent reference is [154]. For this reason in this section only the main results of Fourier analysis on finite non-abelian groups will be mentioned, and presented as the intuitive reason behind the GFT.

The first important result in this direction is the *Peter-Weyl theorem*, which states [26][28] that the matrix entries of the irreducible representations belonging to the regular representation (which includes all of them) form an orthogonal basis for the class  $L(G)$  of all the complex functions on the group  $G$ . In the case of finite groups this space can be identified with the group algebra  $\mathbb{C}[G]$ , and the distinction between  $L(G)$  and  $\mathbb{C}[G]$  is not necessary.

The Fourier transform can then be defined as the map that block diagonalizes the regular representation, and hence projects a function from the algebra  $\mathbb{C}[G]$  onto the irreducible representations of the group. As [155] highlights, this is however not enough to preserve all the useful properties of a Fourier transform,

and therefore it is required, more or less explicitly, that the subspaces over which the irreps are defined are closed with respect to convolutions.

Formally:

**Definition 5** (Group Fourier transform). The *group Fourier transform* of the function  $f \in \mathbb{C}[G]$  on the irrep  $\rho$  is

$$\hat{f}(\rho) = \sum_{x \in G} f(x) \rho(x) \quad \rho \in \mathcal{R} \quad (6.26)$$

where  $\mathcal{R}$  is a complete set of inequivalent irreducible representations of  $G$ .

This definition is completely general for finite groups, both abelian and non-abelian. It can be shown that for abelian groups the GFT exactly reconnects with the results of the spectral theory commonly introduced for periodic functions or functions over  $\mathbb{R}$  and  $\mathbb{Z}$ . Instead, for non-abelian finite groups the generalized relations are often much more convoluted, as the Fourier coefficients  $\hat{f}(\rho)$  are matrices rather than scalars. The basic correspondence between convolution in the group domain and simple multiplication (now matrix multiplication) in the Fourier domain still holds, but the structure of the transform and its inversion formulas becomes substantially richer and more complex.

### 6.3.1 Fourier transform for $\mathbb{Z}_n$

It is worth showing the Fourier transform for at least one special case of abelian group, namely the cyclic group  $\mathbb{Z}_n$ .

The usual definition is:

$$\hat{f}(k) = \sum_{x=0}^{n-1} e^{-2\pi i k x / n} f(x), \quad k \in \{0, 1, 2, \dots, n-1\} \quad (6.27)$$

for  $f$  defined on the discrete set  $\{0, 1, 2, \dots, n-1\}$ . This is often called *Discrete Fourier Transform*, and the underlying group is the cyclic  $\mathbb{Z}_n$ , with group operation given by addition modulo  $n$ . It is possible to show (refer for example to [161]) that the irreducible representations of  $\mathbb{Z}_n$  are:

$$\rho_k(g_j) = e^{2\pi i k j / n} \quad (6.28)$$

which are all 1-dimensional.

While not strictly necessary in this one dimensional case, but useful for later

discussion, it is useful to introduce as *character* the trace of the irreducible matrix:

$$\chi(x) = \text{tr}(\rho(x)) \quad (6.29)$$

As  $\text{tr}(S^{-1}\rho S) = \text{tr}(\rho)$ , equivalent representations are associated to the same characters.

The number and dimensions of irreps are constrained by two general results already introduced for finite groups, and that can be formalized in the following:

**Proposition 1.** *Every irreducible representation  $\rho_\lambda$  is contained in the regular representation with multiplicity equal to its degree.* [26]

from which the following statement on the degrees follows:

**Proposition 2.** *The degrees  $d_i$  of inequivalent irreps satisfy  $\sum_i d_i^2 = |G|$*

As the degrees of the irreps of  $\mathbb{Z}_n$  are all 1, and  $|\mathbb{Z}_n| = n$ , from Eq. (6.28) -generally for abelian groups characters notation is preferred as it is more general, but here they coincide- and Eq. (6.26) it is possible to immediately write the Fourier transform of Eq. (6.27) (up to a sign convention in the exponent):

$$\hat{f}(\rho_k) = \sum_{x=0}^{n-1} f(x) e^{2\pi i k x / n} \quad (6.30)$$

Thus the conventional discrete Fourier transform has a very natural interpretation in terms of group representation theory. The same is true for the transforms of functions defined over  $\mathbb{R}$  or the unit circle.

An important property of the discrete Fourier transform, which reveals a general pattern of the more general group Fourier transform, is the closure to convolution. Defining convolution with the useful notation of [155]:

**Definition 6.** It is defined *translation* of the discrete function  $f$ , defined over  $\mathbb{Z}_n$ , by  $y$  in  $\mathbb{Z}_n$ :

$$f^y(x) = f([x - y]_n) \quad (6.31)$$

From this it is possible to recover the common definition of convolution as:

$$(f * g)(x) = \sum_{y=0}^{n-1} f([x - y]_n) g(y) \quad (6.32)$$

and the following results, formalized by the *convolution theorem* can be proved to

be true:

$$\mathcal{F}(f * g)(k) = \widehat{f}(k) \cdot \widehat{g}(k) \quad (6.33)$$

An important feature of the abelian Fourier transform is to map convolutions in direct space, to products in irrep space. It is not obvious, but it will be shown in the following section that this holds true also on finite non-abelian groups, like  $S_n$ .

### 6.3.2 Fourier transform for finite non-commutative groups

The results of the previous section serve as a link with many intuitive concepts from classical spectral theory. This section expands on them to include finite non-abelian groups, and specifically  $S_n$ .

Understanding why the convolution theorem remains valid in the non-abelian setting can be approached in two equivalent ways: one can define the group Fourier transform so that it diagonalises convolution operators, and then verify that this reduces to the familiar abelian Fourier transform in the commutative case; or one can start from the general definition of the GFT in Eq. (6.26) and derive the convolution property using standard results from representation theory. Here the latter route is followed.

A generic function of the algebra  $\mathbb{C}[S_n]$  of the symmetric group can be written as:

$$f = \sum_{\sigma \in S_n} f(\sigma) \sigma \quad (6.34)$$

which, as already noted is also a generic complex-valued function.

There are many choices for the law of composition of functions belonging to  $\mathbb{C}[S_n]$ , which, in turn, is a law on the composition of the basis vector of the algebra. The choice of convolution as the operator combining two functions from the algebra is canonical, and extends the non-commutative group action in  $S_n$ .

Let  $f_1$  and  $f_2$  be two elements of the group algebra, and write their multiplication

as:

$$\begin{aligned}
 \left( \sum_{g \in G} f_1(g) g \right) \left( \sum_{h \in G} f_2(h) h \right) &= \sum_{g, h \in G} f_1(g) f_2(h) (gh) \\
 &= \sum_{k \in G} \sum_{\substack{g, h \in G \\ gh = k}} f_1(g) f_2(h) k \\
 &= \sum_{k \in G} \sum_{h \in G} f_1(kh^{-1}) f_2(h) k \\
 &= \sum_{k \in G} (f_1 * f_2)(k) k,
 \end{aligned} \tag{6.35}$$

where the convolution is defined as:

$$(f_1 * f_2)(x) = \sum_{t \in G} f_1(xt^{-1}) f_2(t) = \sum_{y \in G} f_1(y) f_2(y^{-1}x). \tag{6.36}$$

Then the Fourier transform of the convolution between the two functions acts as:

$$\begin{aligned}
 \mathcal{F}[f_1 * f_2](\rho) &= \sum_{x \in G} (f_1 * f_2)(x) \rho(x) \\
 &= \sum_{x \in G} \sum_{y \in G} f_1(y) f_2(y^{-1}x) \rho(x) \\
 &\stackrel{x=yz}{=} \sum_{y \in G} \sum_{z \in G} f_1(y) f_2(z) \rho(yz) \\
 &= \left( \sum_{y \in G} f_1(y) \rho(y) \right) \left( \sum_{z \in G} f_2(z) \rho(z) \right) \\
 &= \mathcal{F}[f_1](\rho) \mathcal{F}[f_2](\rho).
 \end{aligned} \tag{6.37}$$

Which can be formalized in the following fundamental theorem:

**Theorem 4** (Convolution theorem). *Consider  $f_1, f_2$  elements of the group algebra  $\mathbb{C}[S_n]$ . Then:*

$$\mathcal{F}[f_1 * f_2](\rho) = \mathcal{F}[f_1](\rho) \mathcal{F}[f_2](\rho) \tag{6.38}$$

Hence the group Fourier transform for a finite group turns convolutions in input space, into an irrep-wise matrix product of the Fourier transforms of the starting functions.

A further specialization of this result happens for *class functions*, which can be introduced from the following definitions from group theory:

**Definition 7** (Conjugation). Elements  $g, h \in G$  are *conjugate*, written  $g \sim h$ , if

there exists  $x \in G$  with

$$h = xgx^{-1}. \quad (6.39)$$

**Definition 8** (Conjugacy class). For  $g \in G$ , the *conjugacy class* of  $g$  is

$$\text{Cl}(g) = \{ xgx^{-1} : x \in G \}. \quad (6.40)$$

For example, in  $S_n$  conjugacy classes correspond to the different types of cycles. E.g. in  $S_3$  there are the following conjugacy classes:

- *identity*:  $\{e\}$
- *transpositions*:  $\{(12), (13), (23)\}$
- *3-cycles*:  $\{(123), (132)\}$

In general, it can be proved that the number of conjugacy classes of a group is equal to the number of its irreps.

Finally, a class function can be defined as a function constant on a conjugacy class:

**Definition 9** (Class function).  $f : G \rightarrow \mathbb{C}$  is a class function if:

$$f(xgx^{-1}) = f(g) \text{ for all } x, g \in G. \quad (6.41)$$

Introducing class functions is crucial to elegantly implement convolutions in Fourier space, because they adhere to the following important lemmas proved in [154], around which much of this chapter revolves, and whose origin can often be traced back to the Schur's lemma [26]:

**Lemma 1.**

1: A function  $f$  defined on  $G$  is a class function if and only if:

$$f * h = h * f, \quad \forall h : G \rightarrow \mathbb{C} \quad (6.42)$$

2: If  $f$  is a class function, then the convolution operator built from it and acting on any other function  $h : G \rightarrow \mathbb{C}$  is not only block diagonal in Fourier space, but diagonal, and acting on each irrep  $\rho_\lambda$  as  $c_\lambda \mathbb{I}$ , with  $c_\lambda$  some constant depending on the irrep.

This result ends the extremely condensed review of the formalism needed to understand the role of convolutions on  $S_n$ .

## 6.4 Classical algorithms for identity management on $S_n$

The formalism derived in the previous sections has been studied in the context of identity management for object tracking by Kondor et. al [29][155], and Huang et. al [28][162]. These two classes of studies are the inspiration for the quantum algorithm this chapter is dedicated to. This section details the characterization of the classical pipelines.

The permutation operations on a set of  $n$  objects can be modeled as the group elements of the symmetric group  $S_n$ , whose basic properties have been introduced in the previous sections.

This regularity, stemming from the underlying group theory, can be exploited to build algorithms that evolve probability distributions defined on the group elements of  $S_n$ . In essence, the fundamental idea is that, from an initial configuration  $P_{t=0}$  (there's no constraint on the possibility to start from a configuration with full probability on a single group element, or multiple), other configurations of elements can be reached acting on it with a sequence of permutations. This evolution will result in a new probability distribution at each  $P_t$ . This can be immediately mapped to a detector setting, where  $t$  becomes the layer index, and the group elements model the particle hits order.

Formally, the probability distribution over elements of the symmetric group is a function  $P : G \rightarrow \mathbb{R}$ , which, as shown in the previous sections, can also be mapped to an element of the algebra as in Eq. (6.34)  $P = \sum_{\sigma \in S_n} P(\sigma)\sigma$ . In general, it is a  $n!$  dimensional vector where each entry is the probability of the associated group element.

As an example with 3 objects, consider here  $S_3$ . A fully localized distribution has all the probability in one of the  $P(\sigma)$ :

- if  $[123]$  is the localized configuration then  $P(e) = 1$  and all the other  $P(\sigma \neq e) = 0$
- if  $[213]$  is the localized configuration then  $P(12) = 1$  and all the other  $P(\sigma \neq (12)) = 0$

Notice that this is a small group with just 6 elements, and the hypothesis of a strong localization of the probability mass usually works just as initial configuration of the system.

The problem of evolving  $n!$ -dimensional vectors of probabilities is classically approached within the framework of spectral models, showing that it is convenient

to evolve  $P$  in Fourier domain, and band-limit the modes. This last approximation is necessary because simply performing a group Fourier transform is not by itself reducing the problem dimensionality, the advantage in spectral approaches comes from the possibility of performing meaningful tasks on subspaces, eventually limiting the action to a subset of them. Restricting to a subspace, in turn, introduces a big difference between operations that can be modeled as convolutions, over which the subspaces are closed, and operations where this doesn't happen in Fourier domain. It is important to mention that, as [29] highlights, band-limiting is a low pass action in spectral domain that might break the exact association between the evolved distribution, and a true probability distribution, but the order relationships between the different  $P(\sigma)$  preserve the possibility of using the filtered result for data association tasks.

Identity management generally assumes three different steps:

- 1: **Mixing/noise model** – the association of objects to tracks becomes probabilistic after mixing events. The number of objects subject to mixing can vary; low order models consider pairwise mixing between couples of objects. This is the assumption of most classical formulations, and is also the baseline for this research, with possible extensions to higher order mixing events already built in the formalism.
- 2: **Bayes update** – the distribution from the previous step is conditioned on the observations from the dataset. The lowest possible order information is of the kind "object  $i$  belongs to track  $j$  with probability  $p$ ".
- 3: **Information extraction** – this step strongly depends on the task; an example can be the estimation of the most likely configuration at the final timestep.

#### 6.4.1 Noise models

Mixing events are modeled from two different perspectives in classical studies. The most intuitive noise model from [28] is the convolution with a function designed as<sup>2</sup>:

$$Q(\sigma) = \begin{cases} p & \sigma = e \\ \frac{1-p}{\binom{n}{2}} & \sigma \text{ is any transposition} \\ 0 & \text{otherwise} \end{cases} \quad (6.43)$$

---

<sup>2</sup>In reality [28] suggests a noise model where  $\sigma$  is a specific transposition, hence it is not acting diagonally in irrep space, but just block diagonally. Here a simplified case is preferred, similar to [29].

The distribution  $Q$  is built as a union of conjugacy classes, and for this reason is a class function. This, by Lemma [1](#), means that:

$$(\widehat{Q * P})(\rho) = \widehat{Q}(\rho)\widehat{P}(\rho) = \lambda_\rho \widehat{P}(\rho) \quad (6.44)$$

Here the action of the convolution by  $Q$  on each irrep  $\rho_\lambda$  is a constant, which is efficient to compute. Convolution, as stated in the previous section, is the function multiplication that respects the underlying non-commutativity of the group operation, and the effect is a noise model that it acts in irrep (Fourier) space without mixing different irreps.

An alternative perspective is offered by the formalism in [29](#), which is based on a heat diffusion kernel [163](#).

In particular,  $S_n$  has a remarkable graph representation in *Cayley graphs*. Such a graph has as nodes the group elements  $\sigma \in S_n$ , and the connectivity is based on transpositions, as the simplest possible set of generators for the symmetric group. Transpositions, formally  $(ij)$ , exchange set elements  $i$  and  $j$ . The nodes of the Cayley graph are, therefore, group elements, or, identically, set configurations that differ by one transposition.

The associated Laplacian operator, introduced in [29](#), and in accordance with the definition of Eq. [\(5.5\)](#) is:

$$L\sigma_1, \sigma_2 = \begin{cases} -1 & \text{if } \sigma_1 = (ij) \cdot \sigma_2 \\ & \text{for some } 1 \leq i < j \leq n \\ n(n-1)/2 & \text{if } \sigma_1 = \sigma_2 \\ 0 & \text{otherwise.} \end{cases} \quad (6.45)$$

From spectral graph theory, its eigenvectors  $\{v_i\}$ , ordered by eigenvalues  $\{\alpha_i\}$ , capture increasingly complex behaviors on the Cayley graph. Single transpositions can be considered as predominantly smooth operations, as on average they don't alterate significantly the probability distribution  $P(\sigma)$ . For this reason the noise model can be implemented as a diffusion process on the Cayley graph built on transpositions.

Modeling noise as a continuous process where, with a small probability  $\beta$  every transposition can happen, the diffusion law becomes:

$$\widehat{P}_{t'}(\rho_\lambda) = e^{-\beta\alpha_\lambda(t'-t)}\widehat{P}_t(\rho_\lambda) \quad (6.46)$$

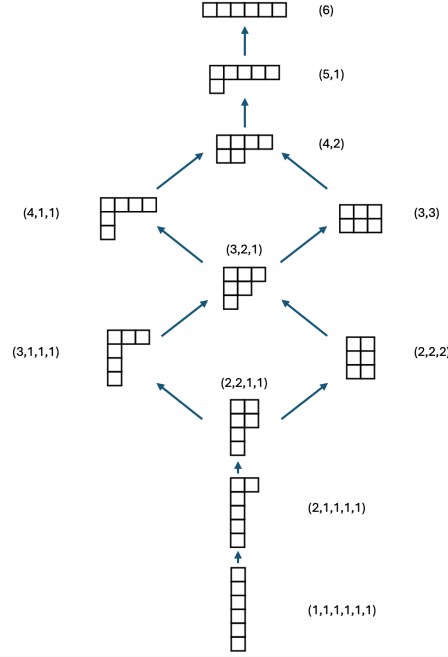


Figure 6.2: Dominance order on partitions of 6, represented by Young diagrams. Partitions become more and more dominant when moving from bottom to top.

where [29]:

$$\alpha_\lambda = \binom{n}{2} \left( 1 - \frac{\text{tr} [\rho_\lambda([1, 2])] }{d_\lambda} \right). \quad (6.47)$$

This expression associates to each  $\lambda$ -indexed partition of  $n$  the correct coefficient. The assumption here is to have an order in the partitions, which is given according to the following definition:

**Definition 10** (Dominance order on partitions). Let  $\lambda = (\lambda_1, \lambda_2, \dots, \lambda_\ell)$  and  $\mu = (\mu_1, \mu_2, \dots, \mu_m)$  be partitions of  $n$ , written in non-increasing order and padded with zeros so that  $\lambda_i = 0$  for  $i > \ell$  and  $\mu_i = 0$  for  $i > m$ . We say that  $\lambda$  *dominates*  $\mu$ , and write  $\lambda \succeq \mu$ , if for all  $k \geq 1$ :

$$\sum_{i=1}^k \lambda_i \geq \sum_{i=1}^k \mu_i. \quad (6.48)$$

An example of this order on partitions of  $n = 6$  is depicted in Fig 6.2

The classical approaches, exploiting the irrep-by-irrep action of the noise models, limit their computations to low order  $\lambda$ , where the problem is still classically tractable. It will be shown that in principle this limitation can be lifted in a

quantum algorithm.

### 6.4.2 Bayes update

Noise models, through convolutions with different distributions, model an identity mixing process where some probability distribution at time-step  $t$   $P_t$  gets diffused to a new  $P_{t+1}$ , at time-step  $t+1$ , which is smoother (less localized) over permutations.

An algorithm built just on the noise model would however just increase the entropy of the distribution, without any possibility of localizing it. The desired property of this algorithm is, however, to evolve the distribution using a dataset of observations of the form:

- *Object  $i$  belongs to position  $j$*  - this is a order 1 observation; it's the only one tested and manifestly used by the classical studies, and might be enough to build a simple model for identity management related to object tracking.
- *Objects  $i_1$  and  $i_2$  belong respectively to positions  $j_1$  and  $j_2$*  - this is a order 2 observation with a relative order information between the two objects associations.
- *Objects  $i_1$  and  $i_2$  belong to positions  $j_1$  or  $j_2$*  - this is still an order 2 observation, but without a specific relative order.
- the number of objects can grow further, but with the same kind of observation model.

The general idea, in direct space (the dual to the spectral basis), is to perform an update of the form:

$$P(\sigma | z) = \eta \cdot P(z | \sigma) \cdot P(\sigma) \quad (6.49)$$

where  $P(\sigma)$  is the prior distribution over permutations: the probability distribution of the assignments in the  $\sigma$  basis;  $P(z | \sigma)$  is the likelihood function, which conditions the prior taking into account the observations  $z$ , while  $P(\sigma | z)$  is the posterior, which, in this case, is the updated distributions of the identities. Finally,  $\eta$  is a normalization constant.

The original classical studies, from Kondor et al. and Huang et al., perform Eq. (6.49) in Fourier space to act just on some specific irreps (think of them as frequencies in the abelian case), thus avoiding the full  $n! \times n!$  operators and  $n!$  vector state. This is both due to their band-limiting, and to their studies on which irreps are involved in first-order updates.

The two approaches are fundamentally different and an important part of this

research has been the study to characterize the feasibility of replicating them through a quantum sub-routine. The next few sections are dedicated to a compact introduction to the two methods, and to their comparison.

### Update through Kronecker conditioning

The element-wise Hadamard multiplication of the likelihood and prior distributions in input space is a composition that, by convolution theorem, becomes a clear convolution in Fourier domain only for abelian groups. For non-commutative groups, like  $S_n$ , it is possible to write a generalized expression in Fourier space, but it is much more complex, because different irreps are combined with each other. The following derivation follows [28].

The inverse group Fourier transform when working with real, orthogonal representations for which  $\rho_\lambda(\sigma^{-1}) = \rho_\lambda(\sigma)^T$ , can be defined as

$$f(\sigma) = \frac{1}{|G|} \sum_{\lambda} d_{\rho_\lambda} \text{Tr} \left[ \hat{f}_{\rho_\lambda}^T \cdot \rho_\lambda(\sigma) \right] \quad (6.50)$$

The goal is to write the product in (6.49) as:

$$f(\sigma) \cdot g(\sigma) = \frac{1}{|G|} \sum_{\nu} d_{\rho_\nu} \text{Tr} \left( R_\nu^T \cdot \rho_\nu(\sigma) \right) \quad (6.51)$$

where

$$\left[ \widehat{fg} \right]_{\rho_\nu} = R_\nu \quad (6.52)$$

is the Fourier transform of the pointwise product between the functions  $f$  and  $g$  on the irrep labeled by  $\nu$ .

The following are the main steps to derive a closed-form expression for (6.52), the full derivation is in [28].

First Eq. (6.50) is applied to the LHS of Eq. (6.51):

$$\begin{aligned} f(\sigma) \cdot g(\sigma) &= \left[ \frac{1}{|G|} \sum_{\lambda} d_{\rho_\lambda} \text{Tr} \left( \hat{f}_{\rho_\lambda}^T \cdot \rho_\lambda(\sigma) \right) \right] \cdot \left[ \frac{1}{|G|} \sum_{\mu} d_{\rho_\mu} \text{Tr} \left( \hat{g}_{\rho_\mu}^T \cdot \rho_\mu(\sigma) \right) \right] \\ &= \left( \frac{1}{|G|} \right)^2 \sum_{\lambda, \mu} d_{\rho_\lambda} d_{\rho_\mu} \left[ \text{Tr} \left( \hat{f}_{\rho_\lambda}^T \cdot \rho_\lambda(\sigma) \right) \cdot \text{Tr} \left( \hat{g}_{\rho_\mu}^T \cdot \rho_\mu(\sigma) \right) \right] \end{aligned} \quad (6.53)$$

It can be proved by properties of the tensor product that:

$$\begin{aligned} \text{Tr}(\hat{f}_\lambda^T \rho_\lambda(\sigma)) \text{Tr}(\hat{g}_\mu^T \rho_\mu(\sigma)) &= \text{Tr}\left((\hat{f}_\lambda^T \rho_\lambda(\sigma)) \otimes (\hat{g}_\mu^T \rho_\mu(\sigma))\right) \\ &= \text{Tr}\left((\hat{f}_\lambda \otimes \hat{g}_\mu)^T (\rho_\lambda(\sigma) \otimes \rho_\mu(\sigma))\right). \end{aligned} \quad (6.54)$$

Since  $\rho_\mu$  and  $\rho_\lambda$  are two irreps, their composition can be re-expressed in terms of irreps  $\rho_\nu$  through a similarity transformation mediated by the C matrices, which are commonly called Clebsch-Gordan:

$$C_{\lambda\mu}^{-1} \cdot [\rho_\lambda \otimes \rho_\mu](\sigma) \cdot C_{\lambda\mu} = \bigoplus_{\nu} \bigoplus_{\ell=1}^{z_{\lambda\mu\nu}} \rho_\nu(\sigma) \quad (6.55)$$

here  $z_{\lambda\mu\nu}$  is a coefficient that controls the number of copies of the irrep  $\rho_\nu$ .

Defining for convenience  $A_{\lambda\mu} \triangleq C_{\lambda\mu}^{-1} \cdot (\hat{f}_{\rho_\lambda} \otimes \hat{g}_{\rho_\mu}) \cdot C_{\lambda\mu}$ , it can be proved -the full derivation is in [28]- that the Fourier transform of the element-wise product of two functions  $f(\sigma)$ ,  $g(\sigma)$  is:

$$[\widehat{fg}]_{\rho_\nu} = \frac{1}{d_{\rho_\nu}|G|} \sum_{\lambda\mu} d_{\rho_\lambda} d_{\rho_\mu} \sum_{\ell=1}^{z_{\lambda\mu\nu}} A_{\lambda\mu}^{(\nu,\ell)} \quad (6.56)$$

Notice how, compared to a convolution in input space which becomes an irrep-wise multiplication in Fourier space, this expression involves different irreps, and combines them through Clebsch-Gordan coefficients.

### Update through twisted Fast Fourier transform

This section is dedicated to a second possibility for the Bayes update step, as in [29]. This method, called twisted fast Fourier transform (FFT) is deeply rooted in the hierarchical structure of the Fourier transform, and avoids the evaluation of the complex Eq. 6.56

Kronecker conditioning is the underlying structure needed to perform in Fourier space, what in input space is a point-wise product. A simplification can be engineered for likelihoods which are built on indicator functions like  $\mathbb{1}_{i,j} \equiv \{\text{object } i \text{ belongs to track } j\}$ :

$$P(O_{i \rightarrow j} | \sigma) = \begin{cases} \pi & \text{if } \sigma(i) = j, \\ (1 - \pi)/(n - 1) & \text{if } \sigma(i) \neq j. \end{cases} \quad (6.57)$$

the difference with a true indicator function is that the image is not  $\{1, 0\}$ , but softer. Another way to see this, closer to the twisted FFT routine is through the introduction of stabilizers.

Let  $H$  be a subset of  $S_n$ , then it is possible to define:

$$H_i = \{h \in S_n : h(i) = i\}, \quad (6.58)$$

hence  $H_i$  acts as a stabilizer for the  $i$ -th element of the set on which a generic group element  $\sigma \in S_n$  acts.  $H_i$  is isomorphic to  $S_{n-1}$  because considers the permutations that affect every set element apart from the  $i$ -th.

The link back to  $S_n$  can be found defining a *coset* of  $H$  as:

$$g_j H_i = \{g_j \circ h : h \in H_i\} \subset S_n \quad (6.59)$$

where  $g_j$  are the group elements in  $S_n$  (there are many of them) bringing the  $i$ -th element in the  $j$ -th position.

In [29] the following formalism is introduced for the cycle  $[[j, n]]$ :

$$J_{j,n} \equiv [[j, n]](i) = \begin{cases} i+1 & \text{if } j \leq i \leq n-1 \\ j & \text{if } i = n \\ i & \text{otherwise} \end{cases} \quad (6.60)$$

which is fixing the elements from 1 to  $j-1$ , and performing a shift to the right by 1 for every element from index  $j$  to index  $n$ , in a cyclic fashion. This means that the  $n$ -th element is brought to the  $j$ -th position, much like  $g_j$  is doing, but with a specialization of eq. (6.58) with  $i$  set to the element  $n$ .

It is now possible to write:

$$\sigma = J_{j,n} \sigma' \quad \text{with } \sigma \in S_n, \sigma' \in S_{n-1} \quad (6.61)$$

where  $\sigma'$  is leaving the  $n$ -th element of the set untouched.

Finally, from the definition of group Fourier transform:

$$\hat{f}(\rho_\lambda) = \sum_{\sigma \in S_n} f(\sigma) \rho_\lambda(\sigma) \quad \rho_\lambda \in \mathcal{R} \quad (6.62)$$

and by eq.(6.61) it is possible to write:

$$\hat{f}(\rho_\lambda) = \sum_{j=1}^n \rho_\lambda([j, n]) \sum_{\sigma' \in S_{n-1}} \rho_\lambda(\sigma') f_j(\sigma') \quad (6.63)$$

where, with a compact notation:

$$f_j(\sigma') = f([j, n]\sigma') \quad (6.64)$$

From Eq.(6.63) it is possible to recognize the group Fourier transform over the  $S_{n-1}$ , labeled by the partitions  $\lambda^-$  of  $n-1$  obtained with the constraint that only one box has been removed from the original partition  $\lambda$  of  $S_n$ :

$$\hat{f}(\rho_\lambda) = \sum_{j=1}^n \rho_\lambda([j, n]) \bigoplus_{\lambda^-} \hat{f}_j(\rho_{\lambda^-}) \quad (6.65)$$

This decomposition is exact working in the Gel'fand-Tsetlin (GT) basis, otherwise some change of basis matrices would enter the above equation .

Specializing this formalism to first order marginals it is possible to introduce two-sided cosets to describe a process where the action on the set elements can be qualitatively described as: "bring the set element  $i$  to  $n$ , then permute all the elements according to some group element belonging to a stabilizer for element  $n$ , then move the untouched set element  $n$  to  $j$ ". This is a possible way of associating element  $i$  and position  $j$  accessing the subgroup structure of  $S_n$ . In a more formal language:

$$f_{i \rightarrow j} = f([j, n]\sigma'[[i, n]]^{-1}) \quad (6.66)$$

Plugging this into eq.(6.65) it is possible to write the final expression for the twisted FFT update:

$$\hat{f}(\rho_\lambda) = \sum_{j=1}^n \rho_\lambda([j, n]) \left[ \bigoplus_{\lambda^-} \hat{f}_{i \rightarrow j}(\rho_{\lambda^-}) \right] \rho_\lambda([i, n]^{-1}) \quad (6.67)$$

In particular, since  $\rho_{(n-1)}$  is the trivial representation of  $S_{n-1}$ , the corresponding Fourier coefficient is exactly the first-order marginal:

$$\hat{f}_{i \rightarrow j}(\rho_{(n-1)}) = P(\sigma(i) = j). \quad (6.68)$$

which means that it is possible to access the marginal probability  $P(\sigma(i) = j)$  exploiting the double coset structure and the hierarchy of Fourier transforms for subsets of  $S_n$  (in the GT basis).

This approach is very useful when the likelihood function is written over cosets, as for the application to object tracking. From [29] the scaling of this approximate approach to Bayes update is bounded to  $O(D^2n)$ , where  $D^2$  is the dimension of the largest irrep block within the band limit.

## 6.5 Development of the quantum algorithm

The previous sections presented the classical formalism behind object identity algorithms performed through the spectral theory of the symmetric group. A substantial part of the effort toward a quantum algorithm has been devoted to characterizing the sometimes scattered classical theory and to formulating a proposal for an algorithm designed for a quantum computer.

This section presents an initial proposal for a quantum pipeline, exploiting the polynomial scaling of the quantum Fourier transform for  $S_n$  [27][9] to devise a non-band-limited algorithm. The study presented here should be regarded as a preliminary version of a full pipeline, which is still under development at the time of writing. This discussion reports on the current status of the research.

Here discussed is a blueprint for the first step of the algorithm, the mixing/diffusion model. For the Bayes update the description will be qualitative, as much of the research in this direction is still under study, and has to be considered work in progress.

### 6.5.1 Towards a fully quantum pipeline

There are two main advantages in engineering the  $S_n$  formalism into a quantum circuit:

- 1: efficient full group quantum Fourier transform;
- 2: efficient coherent encoding of the permutations.

As reported in the introduction, quantum computers have a privileged access to at least some group Fourier transforms. In particular, the seminal work in [9] builds a framework to design efficient quantum circuits for the transform on groups that admit a well-behaved chain of subgroups and sufficiently regular representation theory, so that the transform can be decomposed into a sequence of local steps. One of these groups is  $S_n$ , and the resources needed to implement the QFT are proved to scale  $O(\text{polylog}(|G|))$  as in [27].

This is a very promising starting point, because the best known classical algorithms, performing a Fourier transform of a function defined on the elements of

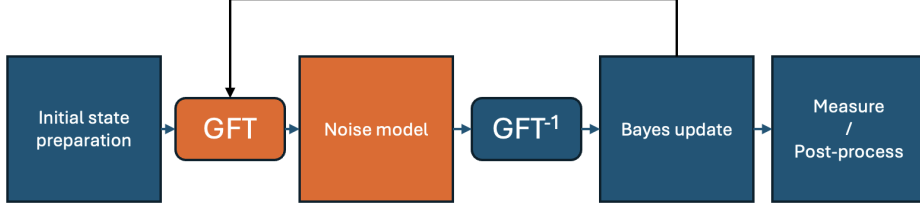


Figure 6.3: Diagram of the quantum pipeline. Blue blocks are stages where the distribution is evolved in input space; orange blocks are operations performed in Fourier (irrep) space;  $GFT$  and  $GFT^{-1}$  are the group Fourier transform and its inverse, used to switch representation from regular to irrep, and vice-versa.

$S_n$ , scale with  $|G|$ , hence with  $O(n!)$ <sup>3</sup>

Exploiting this convenient scaling, the algorithm here discussed evolves a *full* probability distribution, without band-limit approximations, by carefully designing the operators that perform the remaining routines.

The diagram in Fig. 6.3 illustrates the general pipeline for the quantum algorithm. It can be immediately noticed that it's inherently different from any of the classical studies of [29] and [28].

The main difference is in the mixed nature of the noise model and Bayes update. Because this quantum pipeline relies on an efficient quantum group Fourier transform, the basis transformation between input and Fourier spaces is not expensive, and can be repeated for a polynomial number of iterations. This cannot in general be performed efficiently on a classical computer for a generic distribution, and without band-limiting.

The following sections illustrate the current state of the feasibility study on this pipeline for the different sub-routines.

### 6.5.2 State initialization

The initial state preparation routine for the specific application to object tracking prepares the system in the identity configuration  $e$ , where object 1 is associated

---

<sup>3</sup>It is important to highlight that specific symmetries limit this scaling also in the classical case [164], but for a generic function the transform is not efficient nonetheless.

to track 1, object 2 to track 2, and so on up to object  $n$ :

$$P^{(0)}(\sigma) = \delta_{\sigma,e}, \quad (6.69)$$

so that the corresponding quantum state is:

$$|\psi_0\rangle = \sum_{\sigma \in S_n} h^{(0)}(\sigma) |\sigma\rangle = |e\rangle, \quad (6.70)$$

with amplitudes defined from the classical probabilities as:

$$h^{(0)}(\sigma) = \sqrt{P^{(0)}(\sigma)}. \quad (6.71)$$

This guarantees proper normalization,

$$\sum_{\sigma \in S_n} |h^{(0)}(\sigma)|^2 = \sum_{\sigma \in S_n} P^{(0)}(\sigma) = 1. \quad (6.72)$$

The specific implementation of the states  $|\sigma\rangle$  in qubits registers is discussed in the Bayes update section and depends on the assumptions on the circuit for the Fourier transform.

An immediately evident feature of Eq. (6.70) is that while in a classical algorithm the distribution of the  $P(\sigma)$  is evolved, in a quantum algorithm, the unitary evolution of a  $\ell_2$ -normalized vector implies a square root difference with the classical distribution. This is an emerging property of quantum states, and will need to be accounted for if this formalism is to be applied to generative modeling on the symmetric group.

For object tracking, however, the monotonicity of the square root map implies a preservation of the relative order between the probabilities, and the final state amplitudes are guaranteed to have the same maximum as the classical distribution.

### 6.5.3 Quantum diffusion

The state  $|\psi^0\rangle$  is then Fourier transformed, where the unitary Fourier transform can be expressed as:

$$\mathcal{F} = \sum_{\sigma} \sum_{\lambda \vdash n} \sum_{i,j=0}^{d_{\lambda}-1} \sqrt{\frac{d_{\lambda}}{n!}} [\rho_{\lambda}(\sigma)]_{ij} |\lambda ij\rangle \langle \sigma| \quad (6.73)$$

where  $\lambda$  is the irrep index, over the partitions of  $n$ , and  $i, j$  are the row and column indices of the matrix entries associated to the irrep  $\rho_{\lambda}$ .

The normalization coefficient

$$\sqrt{\frac{d_\lambda}{|G|}} = \sqrt{\frac{d_\lambda}{n!}} \quad (6.74)$$

makes  $\mathcal{F}$  unitary, and is explicit in the quantum Fourier transform instead of appearing in the inverse operator like in the classical formalism.

This operator targets any generic function of  $\sigma \in S_n$ ; e.g, for the initial state:

$$\begin{aligned} |\hat{\psi}^{(0)}\rangle &:= \mathcal{F} |\psi^{(0)}\rangle = \sum_{\sigma \in S_n} \sum_{\lambda \vdash n} \sum_{i,j=0}^{d_\lambda-1} \underbrace{\sqrt{\frac{d_\lambda}{n!}} h^{(0)}(\sigma) [\rho_\lambda(\sigma)]_{ij}}_{[\hat{h}_{\rho_\lambda}^{(0)}]_{ij}} |\lambda ij\rangle \\ &= \sum_{\sigma \in S_n} \sum_{\lambda \vdash n} \sum_{i,j=0}^{d_\lambda-1} \sqrt{\frac{d_\lambda}{n!}} \underbrace{\delta_{\sigma,e}}_{h^{(0)}(\sigma)} [\rho_\lambda(\sigma)]_{ij} |\lambda ij\rangle \\ &= \sum_{\lambda \vdash n} \sum_{i,j=0}^{d_\lambda-1} \sqrt{\frac{d_\lambda}{n!}} \underbrace{[\rho_\lambda(e)]_{ij}}_{\mathbb{I}_{ij}^\lambda} |\lambda ij\rangle \\ &= \sum_{\lambda \vdash n} \sum_{i=0}^{d_\lambda-1} \sqrt{\frac{d_\lambda}{n!}} |\lambda ii\rangle. \end{aligned} \quad (6.75)$$

The working hypothesis on the diffusion model is to fully exploit the convolution theorem, as the classical formulations do, which is a guarantee that this simplification is a fair choice for a noise model. Therefore the filter function  $Q(\sigma)$  to be convolved with the state  $|\psi\rangle$  is the one in Eq. (6.43).

This is a class function, and it commutes with the group action, therefore its Fourier transform is a scalar in each irrep block:

$$\hat{Q}_\lambda = c_\lambda \mathbb{I}_\lambda \quad (6.76)$$

It is possible to compute the  $c_\lambda$  coefficients from the characters of the  $\rho_\lambda$  blocks, starting from the general expression for the Fourier transform:

$$\hat{Q}_\lambda = \sum_{\sigma \in S_n} q(\sigma) \rho_\lambda(\sigma) \quad (6.77)$$

computing the trace of both the previous equations:

$$\text{Tr}(\hat{Q}_\lambda) = c_\lambda \text{Tr}(\mathbb{I}_\lambda) = c_\lambda d_\lambda, \quad (6.78)$$

and

$$\text{Tr}(\hat{Q}_\lambda) = \sum_{\sigma} Q(\sigma) \text{Tr}(\rho_\lambda(\sigma)) = \sum_{\sigma} Q(\sigma) \chi_\lambda(\sigma), \quad (6.79)$$

where  $\chi_\lambda$  is the character of  $\rho_\lambda$ . Equating Eq. (6.78) and Eq. (6.79):

$$c_\lambda d_\lambda = \sum_{\sigma} Q(\sigma) \chi_\lambda(\sigma) \quad (6.80)$$

and from the specific choice of the  $Q$  distribution, which is non-zero just on the identity and all of the transpositions (that form the conjugacy class  $C_2$ ):

$$\sum_{\sigma} Q(\sigma) \chi_\lambda(\sigma) = Q(e) \chi_\lambda(e) + \sum_{\sigma \in C_2} Q(\sigma) \chi_\lambda(\sigma). \quad (6.81)$$

Recalling now that  $Q(e) = p$ , and  $\chi_\lambda(e) = d_\lambda$ , the first term at RHS results:

$$Q(e) \chi_\lambda(e) = p d_\lambda. \quad (6.82)$$

For the term on transpositions recall that there are  $\binom{n}{2}$  of them, and by the choice of  $Q$  the coefficient is  $Q_{C_2}(\sigma) = \frac{1-p}{\binom{n}{2}}$ ; moreover all transpositions form a conjugacy class, and the characters are constant over conjugacy classes, so  $\chi_\lambda(\sigma) = \chi_\lambda((1, 2))$ . Finally:

$$\sum_{\sigma \in C_2} Q(\sigma) \chi_\lambda(\sigma) = \binom{n}{2} \cdot \frac{1-p}{\binom{n}{2}} \cdot \chi_\lambda((1, 2)) = (1-p) \chi_\lambda((12)) \quad (6.83)$$

which, equated to Eq. (6.82) gives the final expression for the different  $c_\lambda$ :

$$c_\lambda = p + (1-p) \frac{\chi_\lambda((12))}{d_\lambda} \quad (6.84)$$

The expression for the diffusion operator in Fourier space is therefore:

$$\Delta = \sum_{\lambda} c_\lambda |\lambda\rangle \langle \lambda| \otimes \mathbb{I}_\lambda = \sum_{\lambda} c_\lambda |\lambda\rangle \langle \lambda| \sum_{i,j=0}^{d_\lambda-1} |i\rangle \langle i| \otimes |j\rangle \langle j| \quad (6.85)$$

which is in general not unitary.

To account for this, and because of the studies on convolution operators in quantum algorithms from [165] and [166], the chosen strategy is a block encoding of the diffusion operator  $\Delta$ .

The unitary block encoding is defined as:

$$U_{\Delta} = \begin{pmatrix} \Delta/\alpha & * \\ * & * \end{pmatrix} \quad (6.86)$$

where  $\alpha \geq \|\Delta\|_2$  is a normalization factor chosen so that all singular values of  $\Delta/\alpha$  are at most one.

The action of  $\Delta$  in Fourier space is diagonal on the irrep label:

$$\Delta |\lambda, i, j\rangle = c_{\lambda} |\lambda, i, j\rangle, \quad (6.87)$$

For a generic transposition, like (12), an irrep  $\rho_{\lambda}((12))$  can be chosen as a unitary matrix, satisfying  $\rho_{\lambda}((12))^2 = \mathbb{I}$ , with eigenvalues  $e_{\ell} = \pm 1$ . From this, and since the trace of a matrix is invariant by similarity transformations:

$$\chi_{\lambda}((12)) = \text{Tr}(\rho_{\lambda}((12))) = \sum_{\ell=1}^{d_{\lambda}} e_{\ell}, \quad e_{\ell} \in \{+1, -1\}, \quad (6.88)$$

which implies

$$-d_{\lambda} \leq \chi_{\lambda}((12)) \leq d_{\lambda} \quad \Rightarrow \quad \left| \frac{\chi_{\lambda}((12))}{d_{\lambda}} \right| \leq 1. \quad (6.89)$$

Applying this constraint to Eq. (6.84):

$$c_{\lambda} = p + (1-p) \frac{\chi_{\lambda}(\tau)}{d_{\lambda}} \in [p - (1-p), p + (1-p)] = [2p - 1, 1], \quad (6.90)$$

so in particular  $c_{\lambda} \in [-1, 1]$  and  $|c_{\lambda}| \leq 1$  for all  $\lambda$ . This shows that the spectral norm of  $\Delta$  is at most one, and it is possible to choose  $\alpha = 1$ .

The operator  $U_{\Delta}$  then unitarily evolve the a global state composed of the qubits encoding the distribution  $|\hat{\psi}^{(0)}\rangle$  together with an ancillary register a such as:

$$U_{\Delta} |\hat{\psi}^{(0)}\rangle |0_a\rangle = \Delta |\hat{\psi}^{(0)}\rangle |0_a\rangle + |\perp\rangle |1_a\rangle, \quad (6.91)$$

where the  $|0\rangle_a$  subspace stores the non-unitary action of the diffusion operator  $\Delta$ , and  $|1\rangle_a$  the computation that is non-useful for the algorithm.

From here there are three points to be addressed:

- Can the block encoding be performed with a polynomial number of resources?

- How many diffusion steps are needed to reliably model a tracking task?
- Can the probability of measuring the ancillary register in the 0-state be boosted by amplitude amplification techniques?

### Resource estimation for block encoding

For the first point, two approaches have been considered: (i) a QROM method with the pre-computation of the  $c_\lambda$  coefficients, (ii) an on-the-fly computation through quantum arithmetic.

The first approach scales inefficiently, because the classical computation of the coefficients has to be performed, for every  $n$ , for each partition. A closed-form formula has been derived by Hardy-Ramanujan [167] as:

$$p(n) \sim \frac{1}{4n\sqrt{3}} e^{\pi\sqrt{\frac{2n}{3}}} \quad (6.92)$$

The second approach is the one that is currently being evaluated for the final pipeline. The term  $\frac{\chi_\lambda((12))}{d_\lambda}$  in Eq. (6.84) can be evaluated through the following closed-form result from [26] and [168]:

$$r(\lambda) = \frac{\chi_\lambda((1,2))}{d_\lambda} \equiv \frac{1}{n(n-1)} \sum [\lambda_j^2 - (2j-1)\lambda_j] = \frac{1}{\binom{n}{2}} \sum_j \binom{\lambda_j}{2} - \binom{\lambda'_j}{2} \quad (6.93)$$

which is a special case for transpositions of a more general result from Frobenius. Here  $\lambda'$  labels the transpose of  $\lambda$ .

As every operation can be computed coherently through basic quantum algebra circuits, it can be assumed that block encoding  $\Delta$  is efficient, and needs  $O(\text{poly}(n))$  resources.

As this thesis is being written the third research question *Can the probability of measuring the ancillary register in the 0-state be boosted by amplitude amplification techniques?* is under study.

The probability of successfully post-selecting the state  $|0\rangle_a$  after the first diffusion through block encoding is:

$$P_s = \left\| \Delta \left| \hat{\psi}^{(0)} \right\rangle \right\|^2 \quad (6.94)$$

While it is already possible to prove the efficiency of an amplitude amplification routine that brings  $P_s$  close to 1 using a polynomial number of gates for the specific state  $\left| \hat{\psi}^{(0)} \right\rangle$ , it is under study the possibility of generalizing this result to account for multiple diffusion and Bayes update step.  $P_s$  depends intrinsically on

the data distribution that models the evolution of  $|\hat{\psi}^{(0)}\rangle$ , this means that, as for classical machine learning, it could be extremely difficult to compute meaningful complexity scalings, and much about the efficiency of this step could be deferred to future empirical tests.

This is strictly related to the number of diffusion iteration that can be performed without exponentially suppressing  $P_s$ .

#### 6.5.4 Bayes update

This section briefly outlines the planned next steps of the project as it progresses from the diffusion step to the Bayes update of Eq. (6.49). The detailed construction of a quantum Bayes-update routine is the main active area of development at the time this thesis is being written and is therefore not fully covered here.

The results of the characterization on the classical algorithms, reported in Section 6.4.2, show that, for a quantum algorithm, performing a Bayesian update in Fourier space could be much more complex than developing a routine in the anti-transformed space.

The reason is that both Eq. (6.56), and Eq. (6.67) imply complex operations, much more accessible in the band-limited classical algorithms, than in a quantum circuit. For this reason the collaboration's efforts are converging towards the study of a routine acting in input space, after the anti-Fourier transform  $\mathcal{F}^\dagger$ , and based on the idea of projecting in the subspace of a likelihood function with the simplified form:

$$L(\sigma) = \begin{cases} 1 & \text{if } \sigma(i) = j \\ 0 & \text{if } \sigma(i) \neq j \end{cases} \quad (6.95)$$

This is an indicator-type function, or a *hard* likelihood, and represents the first step to then study also softer functions as:

$$L(\sigma) = \begin{cases} \pi & \text{if } \sigma(i) = j \\ (1 - \pi)/(n - 1) & \text{if } \sigma(i) \neq j \end{cases} \quad (6.96)$$

This will require a block encoding of the non-unitary projector, in a similar fashion to what is done in Fourier space for the diffusion step.

#### Permutation encoding

The details on how the projectors implementing the likelihood action are engineered depend on the encoding of the permutation states  $|\sigma\rangle$ , which, in turn, will

depend on the specific circuit of the QFT. Below are discussed two possible cases, and their scaling in terms of qubits.

Considering the most natural encoding of a permutation, which associates to each element  $\sigma \in S_n$  the tuple  $[i_1, i_2, \dots, i_n]$  it is then possible to encode each digit in a quantum register of  $\log_2(n)$  qubits, with a total cost of  $n \lceil \log_2 n \rceil$  qubits, which scales polynomial in the set size.

A second possible encoding is offered by *Lehmer codes*:

$$\ell(\sigma) \equiv (\ell_1, \ell_2, \dots, \ell_n) \quad (6.97)$$

where

$$\ell_i = \# \{j > i : \sigma_j < \sigma_i\}. \quad (6.98)$$

which can be read as "consider position  $i$  in the original unrolled permutation vector and then count the number of occurrences for which  $\sigma(i) > \sigma(j)$  when  $j > i$ ".

There is a bijection between the Lehmer codes of Eq. (6.97) and the set of integers  $\{0, 1, \dots, n! - 1\}$ , hence this encoding is denser than the natural encoding, which by definition admits no repetition in the digits.

When encoding Lehmer codes as quantum states, the qubit registers reflect the digits in (6.97). Notice that each  $\ell_i$  has a decreasing range as  $i$  grows, in fact:

$$\ell_1 \in \{0, \dots, n-1\}, \quad \ell_2 \in \{0, \dots, n-2\}, \quad \dots, \quad \ell_{n-1} \in \{0, 1\}, \quad \ell_n = 0. \quad (6.99)$$

Thus, the number of qubits required for the  $i$ -th digit is:

$$\lceil \log_2(n - i + 1) \rceil, \quad (6.100)$$

and the total number of qubits is:

$$n_{\text{qubits}}^L = \sum_{i=1}^{n-1} \lceil \log_2(n - i + 1) \rceil = \sum_{k=2}^n \lceil \log_2 k \rceil \quad (6.101)$$

Therefore both the natural and the Lehmer encodings scale polynomial in the set size, and the choice can be entirely based on the assumptions made by the QFT circuit.

## 6.6 Comments

The algorithm discussed in this chapter, although in a development stage, establishes within quantum machine learning a new quantum framework to model the evolution of probability distributions over permutations.

Because this is an active research direction, the discussion has been focused on the current status of the project: consolidating the underlying representation theoretic formalism, and providing an initial blueprint for one of the two central subroutines, namely the mixing/diffusion step.

For this diffusion subroutine the general guarantees for a polynomial scaling of the resources needed for its implementation have been presented, identifying the key open questions that remain to be addressed. The Bayes update step has been introduced in its formal background, and now represents the natural next stage in the development of a full quantum pipeline.

The flexibility offered by the symmetric group in modeling a broad range of phenomena is making studies on the spectral properties of  $S_n$  a growing trend in quantum computing. This study represents a promising novel perspective on how QML can be applied to computing-intensive tasks in high-energy physics, and its future developments will hopefully contribute to improve our understanding on QML approaches to HEP, beyond the NISQ era.

# 7

## Conclusions

Quantum computing is affirming itself as one of the most promising technologies for the years to come. In particular, quantum machine learning (QML), acting as a bridge with the already established modern machine learning, is an ever growing research field that in the past ten years has been explored on a broad variety of applications. Although quantum machine learning experiences year-by-year a steady increase in the quantum resources it can be tested on, together with better software frameworks and theoretical guarantees, the current state of quantum hardware still belongs to a *noisy intermediate-scale quantum* (NISQ) era. Quantum machine learning is, at this point in time, in a similar position as classical machine learning was just a few decades ago.

On the other hand experimental high energy physics is a well consolidated field, currently trying to push its boundaries beyond the Standard Model predictions. Collider physics at CERN, in particular, in the coming years will be entering a high luminosity regime, which represents a major challenge for all the experiments in terms of both hardware and software. Among the many critical points that HEP will need to address in short and long term future experiments, upgrades to particle tracking methods is one of the most urgent. Currently, a combination of efficient heterogeneous classical algorithms, together with new machine learning tools is providing a solid framework for the years to come. However, as particle track reconstruction will always be critical, research at the boundary of new technologies is of primary importance.

This thesis provides an exploration of quantum machine learning methods for particle tracking. It does so under two well defined lines of research, in both of which it provides original contributions.

The first project, presented in Chapter 5, is a development, characterization, and upgrade work on quantum graph neural networks for particle tracking on a synthetic dataset for a pile-up value  $\mu = 200$  at LHC. The study is inherently designed to be tested in this NISQ phase of simulators and quantum hardware. It belongs to the vast corpus of research on the relationship between classical and quantum machine learning, with the additional complexity coming from its application to high energy physics data.

The original contributions in this work span from the software development of a new QGNN implementation with modern frameworks, to the upgrades to the original architecture, both for the classical and the hybrid quantum networks. The hybrid network is shown to not only achieve the same performance of the classical one, but to also converge to the final metrics faster, opening the road to future research on the integration of quantum and classical methods in variational algorithms for NISQ.

The second project looks beyond the constraints of NISQ hardware, and is aimed at the future of quantum computing and quantum machine learning. This research represents a completely novel algorithm within QML, inspired by classical research in the group representation theory of the symmetric group  $S_n$  applied to classical machine learning. Exploiting the privileged, efficient access of quantum computers to the group Fourier transform for  $S_n$ , Chapter 6 presents the characterization, and first blueprint of a new algorithm for the tracking-related task of object identity management.

This quantum algorithm is currently under development. This thesis focuses on the diffusion sub-routine, where a probability distribution over group elements of  $S_n$  is propagated efficiently in Fourier domain, to model identity mixing events. The design and implementation of the second stage of the pipeline—a Bayesian update that conditions the distribution on observations—constitute the next step in this research, and will be the focus of my future work beyond this thesis.

In summary, quantum machine learning is growing to maturity through this challenging, and exciting phase where quantum hardware is still limited, but evolving at incredible pace. Designing quantum algorithms for high energy physics for the current NISQ devices, and building new frameworks for the future is a crucial driving force for the development of both fields. The work presented in this thesis contributes to this effort by exploring concrete QML approaches to particle tracking along two complementary directions. The hope is that these studies will not

---

only inform the development of quantum technologies and their integration into HEP workflows, but also help us access new layers of understanding about the universe we live in, through the combined lens of quantum information and high energy physics.



# Appendices





## Appendix A: Kalman filter for particle tracking

What follows is a simplified description of the Kalman filter algorithm applied to particle tracking, largely based on [25] and [169].

The Kalman filter is, in its essence, a track-following algorithm that, given a sequence of hits compatible with the passage of a charged particle through the detector, produces an optimal estimate of the underlying trajectory parameters by iteratively propagating and updating the state vector and its covariance as new measurements are incorporated.

To illustrate this procedure, let  $\mathbf{m}_{0,j_0}, \mathbf{m}_{1,j_1}, \dots, \mathbf{m}_{l-1,j_{l-1}}$  be the sequence of hits already associated to a proto-track. The first index of  $m_{ij}$  labels the detector layer, and the second index is the relative hit identifier within the  $i$ -th layer.

Under the assumption of modeling charged particles in a quasi-uniform magnetic field, their trajectory is described by a five-dimensional state vector  $\mathbf{p}$ . The notation that will be used in the remainder of this appendix will call:  $\mathbf{p}_{l,t}$  as labels the true track state at the  $l$ -th detector layer, instead  $\mathbf{p}_{l|l-1}$  describes the estimate at layer  $l$ , using the measurements up until layer  $l-1$ ; the shorthand notation  $\mathbf{p}_l$  is used interchangeably with  $\mathbf{p}_{l|l}$ .

Moreover, the *process matrix*  $\mathbf{F}_l$ , and the *measurement matrix*  $\mathbf{H}_l$  are introduced,

as, respectively, the matrix modeling the propagation from the layer  $l - 1$  to the layer  $l$ , and the matrix defining how the internal state vector at  $l - 1$  maps to the measured quantities at step  $l$ .

Then the true state<sup>1</sup> at layer  $l$  is defined as:

$$\mathbf{p}_{l,t} = \mathbf{F}_{l-1}\mathbf{p}_{l-1,t} + w_l \quad (\text{A.1})$$

with  $w_l$  a random variable for the process noise, with distribution  $\mathcal{N}(\mathbf{x} \mid 0, \mathbf{Q}_{k-1})$ , where  $\mathbf{Q}$  is the noise covariance matrix. The predicted state is instead

$$\mathbf{p}_{l|l-1} = \mathbf{F}_{l-1}\mathbf{p}_{l-1}, \quad (\text{A.2})$$

and its predicted covariance matrix  $\mathbf{\Sigma}$  is defined as:

$$\mathbf{\Sigma}_{l|l-1} = \mathbf{F}_{l-1}\mathbf{\Sigma}_{l-1}\mathbf{F}_{l-1}^T + \mathbf{Q}_{l-1} \quad (\text{A.3})$$

A measurement  $\mathbf{m}_l$  at layer  $l$  is instead obtained as:

$$\mathbf{m}_l = \mathbf{H}_l\mathbf{p}_{l,t} + \mathbf{e}_l, \quad (\text{A.4})$$

with  $\mathbf{e}_l$  the gaussian measurement noise distributed as  $\mathcal{N}(\mathbf{x} \mid 0, \mathbf{V}_k)$ , where  $\mathbf{V}$  is the associated covariance matrix.

The full set of update equations to inject measurements in estimates is:

$$\begin{aligned} \mathbf{K}_l &= \mathbf{\Sigma}_{l|l-1}\mathbf{H}_l^T (\mathbf{V}_l + \mathbf{H}_l\mathbf{\Sigma}_{l|l-1}\mathbf{H}_l^T)^{-1} \\ \mathbf{p}_l &= \mathbf{p}_{l|l-1} + \mathbf{K}_l (\mathbf{m}_l - \mathbf{H}_l\mathbf{p}_{l|l-1}) \\ \mathbf{\Sigma}_l &= (\mathbb{I} - \mathbf{K}_l\mathbf{H}_l) \mathbf{\Sigma}_{l|l-1} \end{aligned} \quad (\text{A.5})$$

where  $\mathbf{K}$  is the Kalman gain matrix.

Each new point increases the  $\chi^2$  of the proto-track by:

$$\chi_+^2 = \left( \mathbf{m}_l - \mathbf{H}_l\mathbf{p}_{l|l-1} \right)^T (\mathbf{V}_l - \mathbf{H}_l\mathbf{\Sigma}_{l|l-1}\mathbf{H}_l^T) \left( \mathbf{m}_l - \mathbf{H}_l\mathbf{p}_{l|l-1} \right) \quad (\text{A.6})$$

---

<sup>1</sup>this and the following relation for the measurements can be in principle replaced by non-linear equations.

---

Usually a smoothing pass is performed on the final states, after N hits aggregation:

$$\begin{aligned}
\mathbf{A}_l &= \mathbf{\Sigma}_l \mathbf{F}_l^T \left( \mathbf{\Sigma}_{l+1}^l \right)^{-1} \\
\mathbf{p}_{l|N} &= \mathbf{p}_{l|l} + \mathbf{A}_l \left( \mathbf{p}_{l+1|N} - \mathbf{p}_{l+1|l} \right) \\
\mathbf{\Sigma}_{l|N} &= \mathbf{C}_l + \mathbf{A}_l \left( \mathbf{\Sigma}_{l+1|N} - \mathbf{\Sigma}_{l+1|l} \right) \mathbf{A}_l^T
\end{aligned} \tag{A.7}$$

where  $\mathbf{A}$  is the smoother gain matrix.



# B

## Appendix B: Further characterization of the QGNN architecture

This appendix serves as additional material for Chapter 5. Here reported in Fig. B.1 and Fig. B.2 are the training curves for, respectively, the original QGNN architecture at increasing pileup values, and the upgraded QGNN and CGNN.

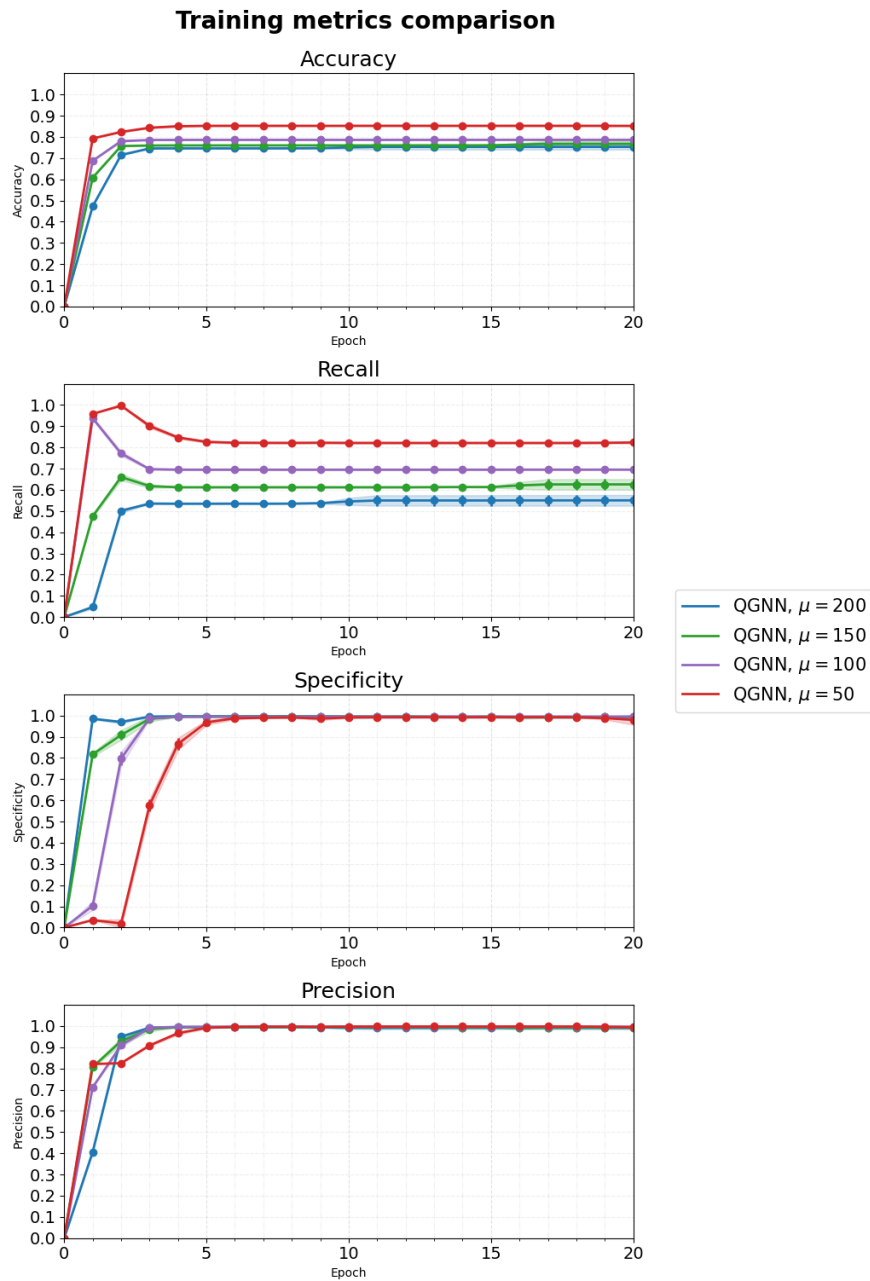


Figure B.1: Original QGNN architecture accuracy, specificity, precision, and recall measured on the training set, with four different dataset simulating different pileup values,  $\mu = 50, 100, 150, 200$ .

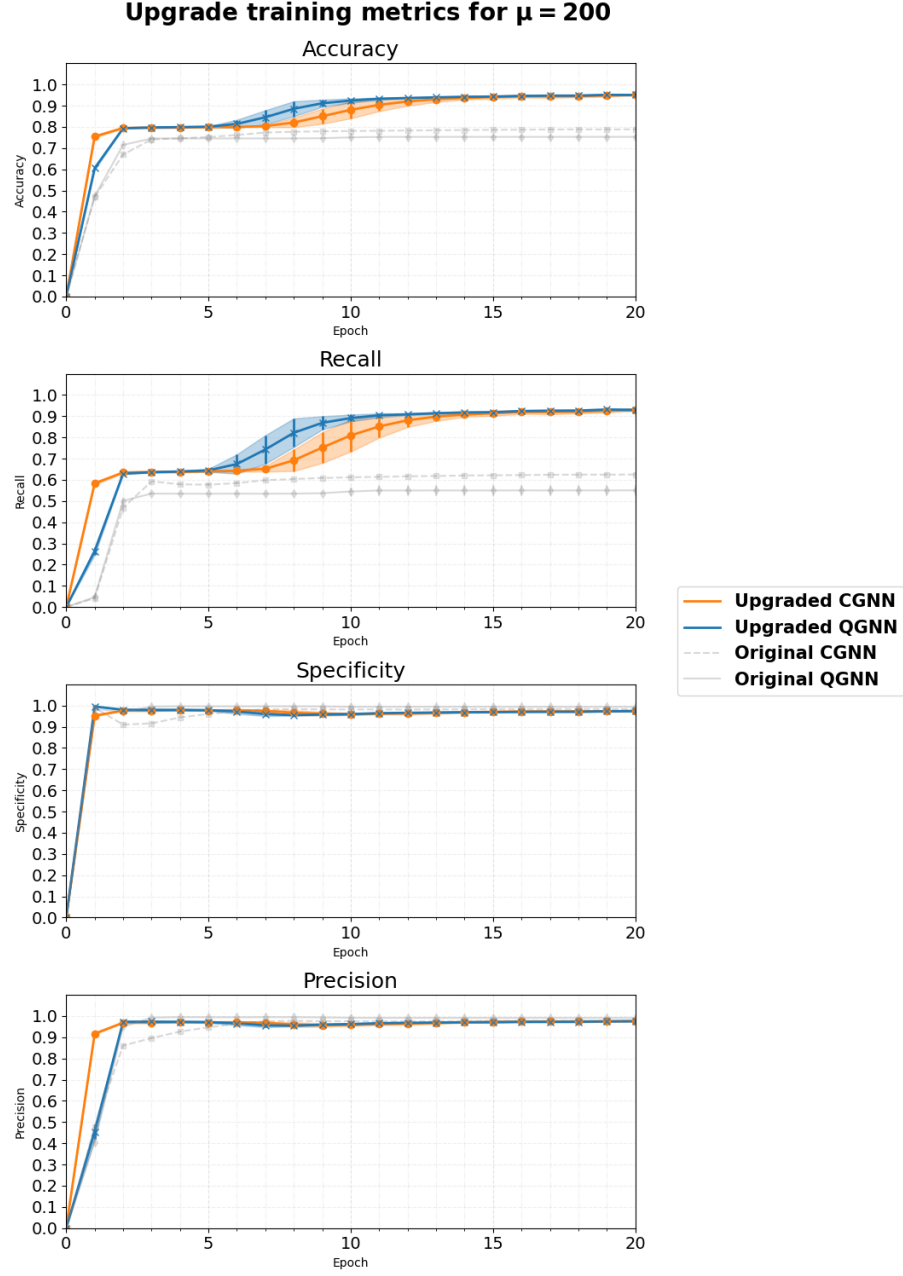


Figure B.2: Comparison of the learning metrics on the  $\mu = 200$  training set. Blue lines represent data from the upgraded Quantum-GNN architecture, with [64,64] classical MLPs and 6-qubits quantum circuit with amplitude encoding. Orange lines correspond to data from the upgraded Classical-GNN, with [64,64] MLPs. Grey lines correspond to the original, pre-upgrade architectures.

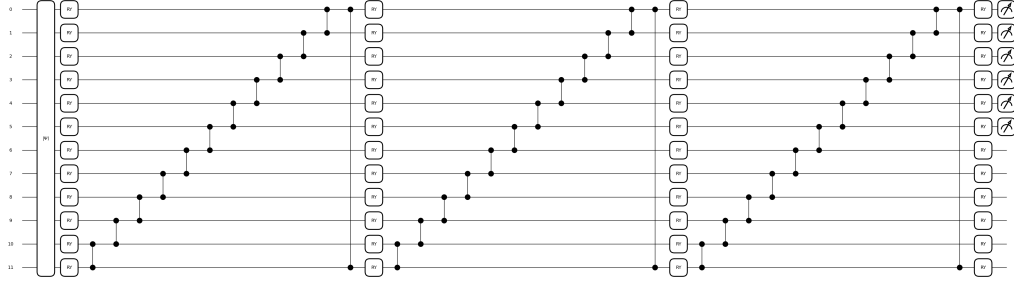


Figure B.3: Extended 12 qubits circuit with  $|\psi\rangle = |\psi_{\text{in}}\rangle \otimes |\psi_{\text{in}}\rangle$  tested within the upgraded QGNN architecture for preliminary studies on expressivity enhancements.

Further preliminary studies have been performed on the possible enhancements to the expressivity of the parametrized quantum circuits. As detailed in Section 5.5.2, tests with data reupload techniques [170][171] through angle embedding are not possible because of the limitations introduced by this kind of information encoding in terms of qubit number.

To circumvent this problem, a repeated parallel amplitude encoding is studied, doubling the number of qubits to perform:

$$|\psi\rangle = |\psi_{\text{in}}\rangle \otimes |\psi_{\text{in}}\rangle, \quad (\text{B.1})$$

where  $|\psi_{\text{in}}\rangle$  is the 64-complex amplitudes state, input of the circuit used in the upgraded QGNN. The total number of qubits is therefore 12, with measure only on the first 6 qubits, to still sample 64-feature vectors. The adapted circuit is shown in Fig. B.3.

Fig. B.4 and Fig. B.5 show a preliminary assessment of the training and validation metrics with this extended circuit. The new circuit introduces bigger variance in the training process, and on validation curves, but does not amount, at least in this preliminary assessment, to either higher metrics, or faster convergence with respect to the single-state amplitude embedding circuit of Chapter 5.5.

The full training process for these 12-qubits models, caching intermediate results to speed up the backpropagation, amounts during training to an average memory occupancy of  $\approx 300GB$ , and are completed, for each of the repeated trainings, in  $\approx 60$  hours on a modern GH200.

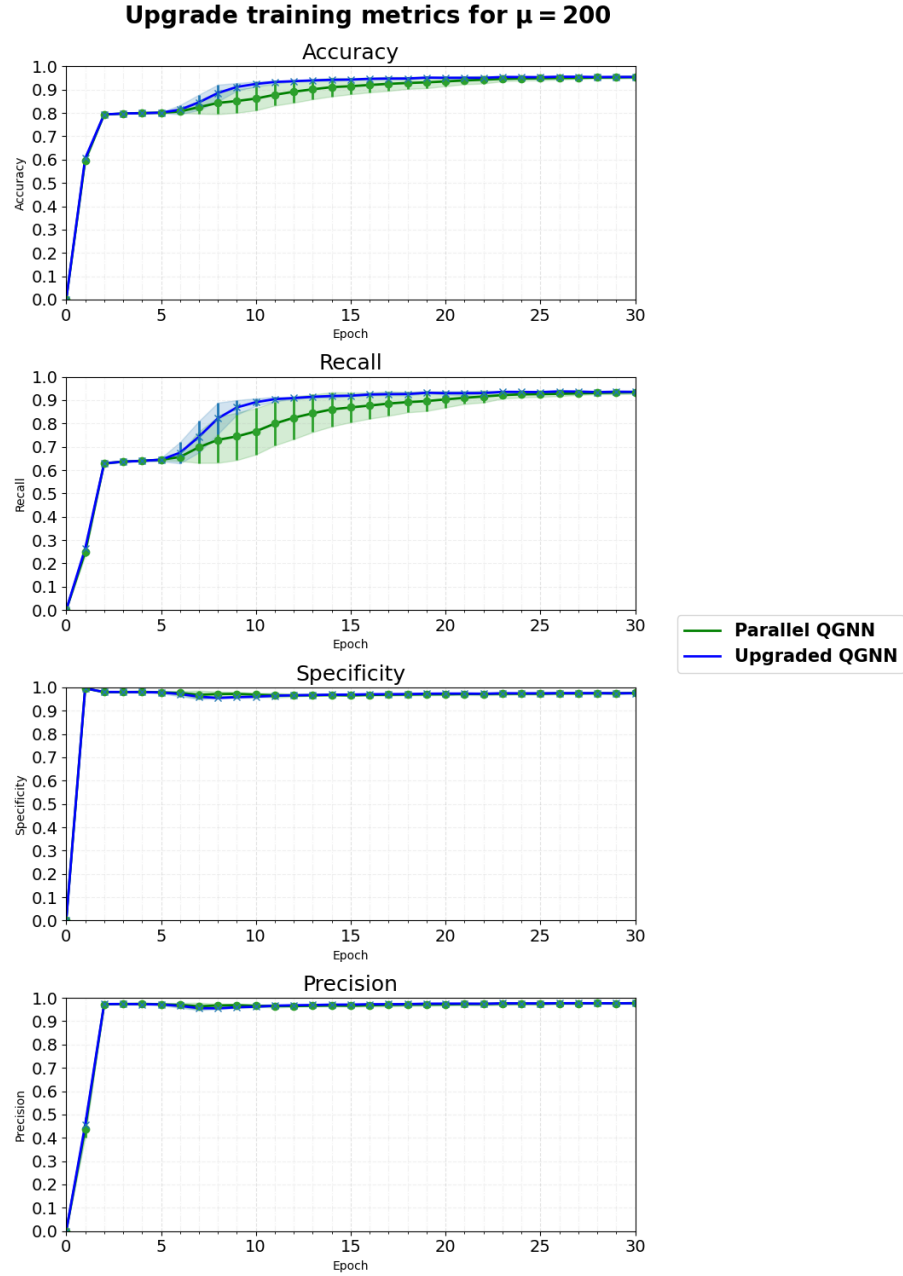


Figure B.4: Comparison of the training metrics between the Upgraded QGNN of Chapter 5 and the same architecture with the circuit in Fig. B.3 labeled Parallel QGNN.

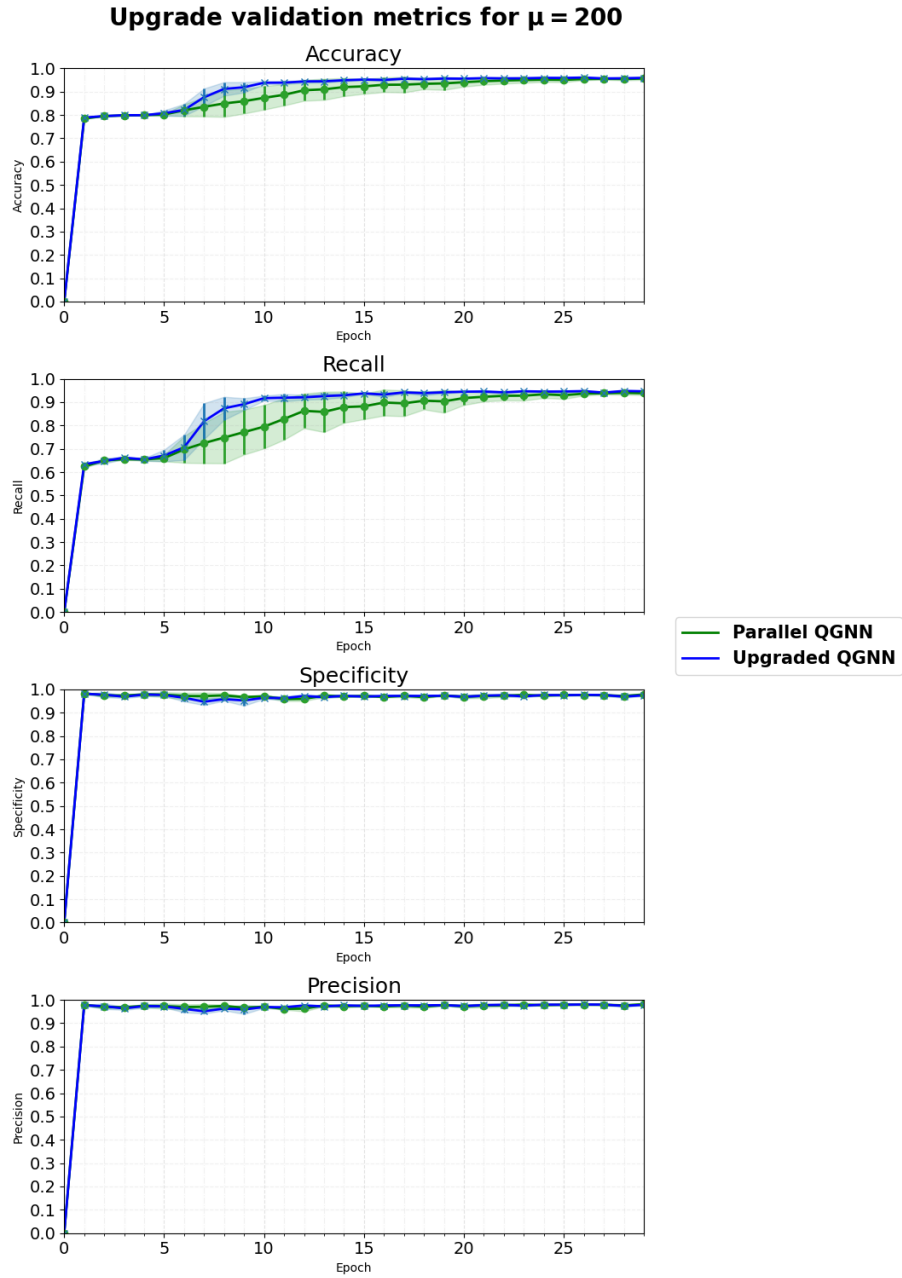


Figure B.5: Comparison of the validation metrics between the Upgraded QGNN of Chapter 5 and the same architecture with the circuit in Fig. B.3, labeled Parallel QGNN.

# Bibliography

- [1] Alberto Di Meglio et al. “Quantum Computing for High-Energy Physics: State of the Art and Challenges”. In: *PRX Quantum* 5.3 (Aug. 2024). ISSN: 2691-3399. DOI: [10.1103/prxquantum.5.037001](https://doi.org/10.1103/prxquantum.5.037001). URL: <http://dx.doi.org/10.1103/PRXQuantum.5.037001>.
- [2] Stephen P. Jordan, Keith S. M. Lee, and John Preskill. “Quantum Algorithms for Quantum Field Theories”. In: *Science* 336.6085 (June 2012), 1130–1133. ISSN: 1095-9203. DOI: [10.1126/science.1217069](https://doi.org/10.1126/science.1217069). URL: <http://dx.doi.org/10.1126/science.1217069>.
- [3] G. Apollinari et al. “High Luminosity Large Hadron Collider HL-LHC”. In: *CERN Yellow Rep.* 5 (2015). Ed. by G Apollinari et al., pp. 1–19. DOI: [10.5170/CERN-2015-005.1](https://doi.org/10.5170/CERN-2015-005.1). arXiv: [1705.08830](https://arxiv.org/abs/1705.08830) [[physics.acc-ph](https://arxiv.org/archive/physics)].
- [4] Peter W. Shor. “Polynomial time algorithms for prime factorization and discrete logarithms on a quantum computer”. In: *SIAM J. Sci. Statist. Comput.* 26 (1997), p. 1484. DOI: [10.1137/S0097539795293172](https://doi.org/10.1137/S0097539795293172). arXiv: [quant-ph/9508027](https://arxiv.org/abs/quant-ph/9508027).
- [5] Seth Lloyd. “Universal quantum simulators”. In: *Science* 273.5278 (1996), pp. 1073–1078.
- [6] Aram W Harrow, Avinatan Hassidim, and Seth Lloyd. “Quantum algorithm for linear systems of equations”. In: *Physical review letters* 103.15 (2009), p. 150502.
- [7] Lov K Grover. “A fast quantum mechanical algorithm for database search”. In: *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing.* 1996, pp. 212–219.
- [8] A Yu Kitaev. “Quantum measurements and the Abelian stabilizer problem”. In: *arXiv preprint quant-ph/9511026* (1995).
- [9] Cristopher Moore, Daniel Rockmore, and Alexander Russell. “Generic quantum Fourier transforms”. In: *ACM Transactions on Algorithms (TALG)* 2.4 (2006), pp. 707–723.
- [10] Maria Schuld. *Supervised quantum machine learning models are kernel methods*. 2021. arXiv: [2101.11020](https://arxiv.org/abs/2101.11020) [[quant-ph](https://arxiv.org/archive/quant)]. URL: <https://arxiv.org/abs/2101.11020>.

- [11] Su Yeon Chang and M Cerezo. “A Primer on Quantum Machine Learning”. In: *arXiv preprint arXiv:2511.15969* (2025).
- [12] Marco Cerezo et al. “Variational quantum algorithms”. In: *Nature Reviews Physics* 3.9 (2021), pp. 625–644.
- [13] Andrea Delgado et al. *Quantum computing for data analysis in high energy physics*. 2022. arXiv: [2203.08805 \[physics.data-an\]](#), URL: <https://arxiv.org/abs/2203.08805>.
- [14] Samuel Yen-Chi Chen et al. “Quantum convolutional neural networks for high energy physics data analysis”. In: *Phys. Rev. Res.* 4 (1 2022), p. 013231. DOI: [10.1103/PhysRevResearch.4.013231](#), URL: <https://link.aps.org/doi/10.1103/PhysRevResearch.4.013231>.
- [15] Sau Lan Wu et al. “Application of quantum machine learning using the quantum kernel algorithm on high energy physics analysis at the LHC”. In: *Physical Review Research* 3.3 (Sept. 2021). ISSN: 2643-1564. DOI: [10.1103/physrevresearch.3.033221](#), URL: <http://dx.doi.org/10.1103/PhysRevResearch.3.033221>.
- [16] Vasilis Belis et al. “Quantum anomaly detection in the latent space of proton collision events at the LHC”. In: *Communications Physics* 7.1 (Oct. 2024). ISSN: 2399-3650. DOI: [10.1038/s42005-024-01811-6](#), URL: <http://dx.doi.org/10.1038/s42005-024-01811-6>.
- [17] Xiangyang Ju et al. “Performance of a geometric deep learning pipeline for HL-LHC particle tracking”. In: *Eur. Phys. J. C* 81.10 (2021), p. 876. DOI: [10.1140/epjc/s10052-021-09675-8](#), arXiv: [2103.06995 \[physics.data-an\]](#).
- [18] Caillou, Sylvain et al. “Novel fully-heterogeneous GNN designs for track reconstruction at the HL-LHC”. In: *EPJ Web of Conf.* 295 (2024), p. 09028. DOI: [10.1051/epjconf/202429509028](#), URL: <https://doi.org/10.1051/epjconf/202429509028>.
- [19] Anthony Correia et al. “Graph Neural Network-Based Track Finding in the LHCb Vertex Detector”. In: *JINST* 19.12 (2024), P12022. DOI: [10.1088/1748-0221/19/12/P12022](#), arXiv: [2407.12119](#), URL: <https://cds.cern.ch/record/2919388>.
- [20] Cenk Tüysüz et al. “Hybrid quantum classical graph neural networks for particle track reconstruction”. In: *Quantum Machine Intelligence* 3.2 (2021), p. 29.
- [21] Matteo Argenton et al. “Charged particle tracking with quantum graph neural networks”. In: *PoS ICHEP2024* (2025), p. 997. DOI: [10.22323/1.476.0997](#).
- [22] Laura Cappelli et al. “Quantum Machine Learning for Track Reconstruction”. In: *EPJ Web Conf.* 337 (2025), p. 01246. DOI: [10.1051/epjconf/202533701246](#).
- [23] M. Cerezo et al. “Does provable absence of barren plateaus imply classical simulability?” In: *Nature Communications* 16.1 (Aug. 2025). ISSN: 2041-

- 
1723. DOI: [10.1038/s41467-025-63099-6](https://doi.org/10.1038/s41467-025-63099-6). URL: <http://dx.doi.org/10.1038/s41467-025-63099-6>.
- [24] Davide Nicotra et al. “A quantum algorithm for track reconstruction in the LHCb vertex detector”. In: *JINST* 18.11 (2023), P11028. DOI: [10.1088/1748-0221/18/11/P11028](https://doi.org/10.1088/1748-0221/18/11/P11028). arXiv: [2308.00619](https://arxiv.org/abs/2308.00619) [quant-ph].
- [25] Duarte Magano et al. “Quantum speedup for track reconstruction in particle accelerators”. In: *Phys. Rev. D* 105.7 (2022), p. 076012. DOI: [10.1103/PhysRevD.105.076012](https://doi.org/10.1103/PhysRevD.105.076012). arXiv: [2104.11583](https://arxiv.org/abs/2104.11583) [quant-ph].
- [26] Persi Diaconis. *Group Representations in Probability and Statistics*. Vol. 11. Institute of Mathematical Statistics Lecture Notes - Monograph Series. 198 pp. Institute of Mathematical Statistics, 1988. ISBN: 0940600145. DOI: [10.1214/lnms/1215467407](https://doi.org/10.1214/lnms/1215467407). URL: <https://doi.org/10.1214/lnms/1215467407>.
- [27] Robert Beals. “Quantum computation of Fourier transforms over symmetric groups”. In: *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*. 1997, pp. 48–53.
- [28] Jonathan Huang, Carlos Guestrin, and Leonidas Guibas. “Fourier Theoretic Probabilistic Inference over Permutations.” In: *Journal of machine learning research* 10.5 (2009).
- [29] Risi Kondor, Andrew Howard, and Tony Jebara. “Multi-object tracking with representations of the symmetric group”. In: *Artificial intelligence and statistics*. PMLR. 2007, pp. 211–218.
- [30] Simon Haykin. *Neural Networks: A Comprehensive Foundation (3rd Edition)*. USA: Prentice-Hall, Inc., 2007. ISBN: 0131471392.
- [31] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. <http://www.deeplearningbook.org>. MIT Press, 2016.
- [32] Ian Goodfellow et al. “Generative adversarial networks”. In: *Communications of the ACM* 63.11 (2020), pp. 139–144.
- [33] Diederik P Kingma, Max Welling, et al. “An introduction to variational autoencoders”. In: *Foundations and Trends® in Machine Learning* 12.4 (2019), pp. 307–392.
- [34] Anil K Jain. “Data clustering: 50 years beyond K-means”. In: *Pattern recognition letters* 31.8 (2010), pp. 651–666.
- [35] Ian T Jolliffe and Jorge Cadima. “Principal component analysis: a review and recent developments”. In: *Philosophical transactions of the royal society A: Mathematical, Physical and Engineering Sciences* 374.2065 (2016), p. 20150202.
- [36] Maria Schuld and Francesco Petruccione. *Machine Learning with Quantum Computers*. Quantum Science and Technology. Cham: Springer, 2021. ISBN: 978-3-030-83097-7, 978-3-030-83100-4, 978-3-030-83098-4. DOI: [10.1007/978-3-030-83098-4](https://doi.org/10.1007/978-3-030-83098-4).
-

- [37] Mikhail Belkin et al. “Reconciling modern machine-learning practice and the classical bias–variance trade-off”. In: *Proceedings of the National Academy of Sciences* 116 (July 2019), p. 201903070. DOI: [10.1073/pnas.1903070116](https://doi.org/10.1073/pnas.1903070116).
- [38] Preetum Nakkiran et al. “Deep double descent: Where bigger models and more data hurt”. In: *Journal of Statistical Mechanics: Theory and Experiment* 2021.12 (2021), p. 124003.
- [39] Lutz Prechelt. “Early Stopping — But When?” In: *Neural Networks: Tricks of the Trade: Second Edition*. Ed. by Grégoire Montavon, Geneviève B. Orr, and Klaus-Robert Müller. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 53–67. ISBN: 978-3-642-35289-8. DOI: [10.1007/978-3-642-35289-8\\_5](https://doi.org/10.1007/978-3-642-35289-8_5). URL: [https://doi.org/10.1007/978-3-642-35289-8\\_5](https://doi.org/10.1007/978-3-642-35289-8_5).
- [40] Kingma DP Ba J Adam et al. “A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* 1412.6 (2014).
- [41] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory”. In: *Neural Comput.* 9.8 (1997), pp. 1735–1780. DOI: [10.1162/neco.1997.9.8.1735](https://doi.org/10.1162/neco.1997.9.8.1735).
- [42] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.
- [43] F. Rosenblatt. “The perceptron: A probabilistic model for information storage and organization in the brain”. In: *Psychological Review* 65 (Nov. 1958), pp. 386–408. DOI: [10.1037/h0042519](https://doi.org/10.1037/h0042519).
- [44] Marvin Minsky and Seymour Papert. “An introduction to computational geometry”. In: *Cambridge tiass., HIT* 479.480 (1969), p. 104.
- [45] Arthur Jacot, Franck Gabriel, and Clement Hongler. “Neural Tangent Kernel: Convergence and Generalization in Neural Networks”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., 2018. URL: [https://proceedings.neurips.cc/paper\\_files/paper/2018/file/5a4be1fa34e62bb8a6ec6b91d2462f5a-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2018/file/5a4be1fa34e62bb8a6ec6b91d2462f5a-Paper.pdf).
- [46] Michael A Nielsen. *Neural networks and deep learning*. Vol. 25. Determination press San Francisco, CA, USA, 2015.
- [47] George V. Cybenko. “Approximation by superpositions of a sigmoidal function”. In: *Mathematics of Control, Signals and Systems* 2 (1989), pp. 303–314. URL: <https://api.semanticscholar.org/CorpusID:3958369>.
- [48] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. “Multilayer feed-forward networks are universal approximators”. In: *Neural Networks* 2.5 (1989), pp. 359–366. ISSN: 0893-6080. DOI: [https://doi.org/10.1016/0893-6080\(89\)90020-8](https://doi.org/10.1016/0893-6080(89)90020-8). URL: <https://www.sciencedirect.com/science/article/pii/0893608089900208>.
- [49] Moshe Leshno et al. “Multilayer feedforward networks with a nonpolynomial activation function can approximate any function”. In: *Neural Networks* 6.6 (1993), pp. 861–867. ISSN: 0893-6080. DOI: <https://doi.org/>

- [10.1016/S0893-6080\(05\)80131-5](https://www.sciencedirect.com/science/article/pii/S0893608005801315), URL: <https://www.sciencedirect.com/science/article/pii/S0893608005801315>.
- [50] Patrick Kidger and Terry Lyons. “Universal approximation with deep narrow networks”. In: *Conference on learning theory*. PMLR. 2020, pp. 2306–2327.
  - [51] Hongzhou Lin and Stefanie Jegelka. “Resnet with one-neuron hidden layers is a universal approximator”. In: *Advances in neural information processing systems* 31 (2018).
  - [52] Guido Montúfar et al. *On the Number of Linear Regions of Deep Neural Networks*. 2014. arXiv: [1402.1869 \[stat.ML\]](https://arxiv.org/abs/1402.1869). URL: <https://arxiv.org/abs/1402.1869>.
  - [53] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Neural Information Processing Systems* 25 (Jan. 2012). DOI: [10.1145/3065386](https://doi.org/10.1145/3065386).
  - [54] TN Kipf. “Semi-supervised classification with graph convolutional networks”. In: *arXiv preprint arXiv:1609.02907* (2016).
  - [55] Morten Kjaergaard et al. “Superconducting qubits: Current state of play”. In: *Annual Review of Condensed Matter Physics* 11.1 (2020), pp. 369–395.
  - [56] Colin D. Bruzewicz et al. “Trapped-ion quantum computing: Progress and challenges”. In: *Applied Physics Reviews* 6.2 (May 2019). ISSN: 1931-9401. DOI: [10.1063/1.5088164](https://doi.org/10.1063/1.5088164). URL: <http://dx.doi.org/10.1063/1.5088164>.
  - [57] Loïc Henriët et al. “Quantum computing with neutral atoms”. In: *Quantum* 4 (Sept. 2020), p. 327. ISSN: 2521-327X. DOI: [10.22331/q-2020-09-21-327](https://doi.org/10.22331/q-2020-09-21-327). URL: <https://doi.org/10.22331/q-2020-09-21-327>.
  - [58] Sergei Slussarenko and Geoff J. Pryde. “Photonic quantum information processing: A concise review”. In: *Applied Physics Reviews* 6.4 (Oct. 2019), p. 041303. ISSN: 1931-9401. DOI: [10.1063/1.5115814](https://doi.org/10.1063/1.5115814). eprint: [https://pubs.aip.org/aip/apr/article-pdf/doi/10.1063/1.5115814/19739502/041303\\_1\\_online.pdf](https://pubs.aip.org/aip/apr/article-pdf/doi/10.1063/1.5115814/19739502/041303_1_online.pdf). URL: <https://doi.org/10.1063/1.5115814>.
  - [59] Jun John Sakurai and Jim Napolitano. *Modern quantum mechanics*. Cambridge University Press, 2020.
  - [60] Richard P Feynman. “Simulating physics with computers”. In: *Feynman and computation*. cRc Press, 2018, pp. 133–153.
  - [61] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, June 2012. ISBN: 978-0-521-63503-5. DOI: [10.1017/cbo9780511976667](https://doi.org/10.1017/cbo9780511976667).
  - [62] Riccardo Manenti and Mario Motta. *Quantum Information Science*. Oxford University Press, Aug. 2023. ISBN: 978-0-19-878748-8.
  - [63] Mikko Möttönen and Juha Vartiainen. “Decompositions of general quantum gates”. In: *Frontiers in Artificial Intelligence and Applications* (May 2005).

- [64] V.V. Shende, S.S. Bullock, and I.L. Markov. “Synthesis of quantum-logic circuits”. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 25.6 (2006), pp. 1000–1010. DOI: [10.1109/TCAD.2005.855930](https://doi.org/10.1109/TCAD.2005.855930).
- [65] Juha J. Vartiainen, Mikko Möttönen, and Martti M. Salomaa. “Efficient Decomposition of Quantum Gates”. In: *Phys. Rev. Lett.* 92 (17 2004), p. 177902. DOI: [10.1103/PhysRevLett.92.177902](https://doi.org/10.1103/PhysRevLett.92.177902), URL: <https://link.aps.org/doi/10.1103/PhysRevLett.92.177902>.
- [66] Neil J. Ross and Peter Selinger. “Optimal ancilla-free Clifford+T approximation of Z-rotations”. In: *Quant. Inf. Comput.* 16.11-12 (2016), pp. 0901–0953. DOI: [10.26421/QIC16.11-12-1](https://doi.org/10.26421/QIC16.11-12-1), arXiv: [1403.2975 \[quant-ph\]](https://arxiv.org/abs/1403.2975).
- [67] Daniel Gottesman. *The Heisenberg Representation of Quantum Computers*. 1998. arXiv: [quant-ph/9807006 \[quant-ph\]](https://arxiv.org/abs/quant-ph/9807006), URL: <https://arxiv.org/abs/quant-ph/9807006>.
- [68] Ben W. Reichardt. *Quantum universality by state distillation*. 2009. arXiv: [quant-ph/0608085 \[quant-ph\]](https://arxiv.org/abs/quant-ph/0608085), URL: <https://arxiv.org/abs/quant-ph/0608085>.
- [69] Sergey Bravyi and Alexei Kitaev. “Universal quantum computation with ideal Clifford gates and noisy ancillas”. In: *Physical Review A* 71.2 (Feb. 2005). ISSN: 1094-1622. DOI: [10.1103/physreva.71.022316](https://doi.org/10.1103/physreva.71.022316), URL: <http://dx.doi.org/10.1103/PhysRevA.71.022316>.
- [70] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. “Quantum algorithms for supervised and unsupervised machine learning”. In: (July 2013). arXiv: [1307.0411 \[quant-ph\]](https://arxiv.org/abs/1307.0411).
- [71] Subhash C Kak. “Quantum neural computing”. In: *Advances in imaging and electron physics* 94 (1995), pp. 259–313.
- [72] Alexander Yu Vlasov. “Quantum computations and images recognition”. In: *arXiv preprint quant-ph/9703010* (1997).
- [73] Esma Aïmeur, Gilles Brassard, and Sébastien Gambs. “Machine learning in a quantum world”. In: *Conference of the Canadian Society for Computational Studies of Intelligence*. Springer. 2006, pp. 431–442.
- [74] Marco Cerezo et al. “Challenges and opportunities in quantum machine learning”. In: *Nature computational science* 2.9 (2022), pp. 567–576.
- [75] Jacob C Bridgeman and Christopher T Chubb. “Hand-waving and interpretive dance: an introductory course on tensor networks”. In: *Journal of Physics A: Mathematical and Theoretical* 50.22 (2017), p. 223001. DOI: [10.1088/1751-8121/aa6dc3](https://doi.org/10.1088/1751-8121/aa6dc3), URL: <https://dx.doi.org/10.1088/1751-8121/aa6dc3>.
- [76] Nana Liu and Patrick Rebentrost. “Quantum machine learning for quantum anomaly detection”. In: *Physical Review A* 97.4 (2018), p. 042315.
- [77] Seth Lloyd et al. “Quantum embeddings for machine learning”. In: *arXiv preprint arXiv:2001.03622* (2020).

- [78] Giuseppe Carleo and Matthias Troyer. “Solving the quantum many-body problem with artificial neural networks”. In: *Science* 355.6325 (2017), pp. 602–606.
- [79] Jonathan Romero, Jonathan P Olson, and Alan Aspuru-Guzik. “Quantum autoencoders for efficient compression of quantum data”. In: *Quantum Science and Technology* 2.4 (2017), p. 045001.
- [80] Pierre-Luc Dallaire-Demers and Nathan Killoran. “Quantum generative adversarial networks”. In: *Phys. Rev. A* 98 (1 2018), p. 012324. DOI: [10.1103/PhysRevA.98.012324](https://doi.org/10.1103/PhysRevA.98.012324). URL: <https://link.aps.org/doi/10.1103/PhysRevA.98.012324>.
- [81] Juan Carrasquilla and Roger G Melko. “Machine learning phases of matter”. In: *Nature Physics* 13.5 (2017), pp. 431–434.
- [82] Roeland Wiersema et al. “Exploring entanglement and optimization within the Hamiltonian variational ansatz”. In: *PRX quantum* 1.2 (2020), p. 020319.
- [83] Joonho Lee et al. “Generalized unitary coupled cluster wave functions for quantum computation”. In: *Journal of chemical theory and computation* 15.1 (2018), pp. 311–324.
- [84] Iris Cong, Soonwon Choi, and Mikhail D. Lukin. “Quantum convolutional neural networks”. In: *Nature Phys.* 15 (2019), pp. 1273–1278. DOI: [10.1038/s41567-019-0648-8](https://doi.org/10.1038/s41567-019-0648-8).
- [85] Zhihui Wang et al. “XY mixers: Analytical and numerical results for the quantum alternating operator ansatz”. In: *Physical Review A* 101.1 (2020), p. 012320.
- [86] Martín Larocca et al. “Group-invariant quantum machine learning”. In: *PRX quantum* 3.3 (2022), p. 030341.
- [87] Andrea Skolik et al. “Equivariant quantum circuits for learning on weighted graphs”. In: *npj Quantum Information* 9.1 (2023), p. 47.
- [88] Abhinav Kandala et al. “Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets”. In: *nature* 549.7671 (2017), pp. 242–246.
- [89] Matias Bilkis et al. “A semi-agnostic ansatz with variable structure for variational quantum algorithms”. In: *Quantum Machine Intelligence* 5.2 (2023), p. 43.
- [90] Daniel Claudino et al. “Benchmarking adaptive variational quantum eigensolvers”. In: *Frontiers in Chemistry* 8 (2020), p. 606863.
- [91] Aram W Harrow and John C Napp. “Low-depth gradient measurements can improve convergence in variational hybrid quantum-classical algorithms”. In: *Physical Review Letters* 126.14 (2021), p. 140502.
- [92] K. Mitarai et al. “Quantum circuit learning”. In: *Physical Review A* 98.3 (Sept. 2018). ISSN: 2469-9934. DOI: [10.1103/physreva.98.032309](https://doi.org/10.1103/physreva.98.032309). URL: <http://dx.doi.org/10.1103/PhysRevA.98.032309>.
- [93] Maria Schuld et al. “Evaluating analytic gradients on quantum hardware”. In: *Phys. Rev. A* 99 (3 2019), p. 032331. DOI: [10.1103/PhysRevA.99.032331](https://doi.org/10.1103/PhysRevA.99.032331).

032331. URL: <https://link.aps.org/doi/10.1103/PhysRevA.99.032331>.
- [94] Andrea Mari, Thomas R. Bromley, and Nathan Killoran. “Estimating the gradient and higher-order derivatives on quantum hardware”. In: *Physical Review A* 103.1 (Jan. 2021). ISSN: 2469-9934. DOI: [10.1103/PhysRevA.103.012405](https://doi.org/10.1103/PhysRevA.103.012405). URL: <http://dx.doi.org/10.1103/PhysRevA.103.012405>.
- [95] Sam McArdle et al. “Variational ansatz-based quantum simulation of imaginary time evolution”. In: *npj Quantum Information* 5.1 (Sept. 2019). ISSN: 2056-6387. DOI: [10.1038/s41534-019-0187-2](https://doi.org/10.1038/s41534-019-0187-2). URL: <http://dx.doi.org/10.1038/s41534-019-0187-2>.
- [96] James C Spall. “Multivariate stochastic approximation using a simultaneous perturbation gradient approximation”. In: *IEEE transactions on automatic control* 37.3 (2002), pp. 332–341.
- [97] Saverio Monaco et al. “Quantum phase detection generalization from marginal quantum neural network models”. In: *Phys. Rev. B* 107 (8 2023), p. L081105. DOI: [10.1103/PhysRevB.107.L081105](https://doi.org/10.1103/PhysRevB.107.L081105). URL: <https://link.aps.org/doi/10.1103/PhysRevB.107.L081105>.
- [98] Guillaume Verdon et al. “Quantum Graph Neural Networks”. In: (Sept. 2019). arXiv: [1909.12264 \[quant-ph\]](https://arxiv.org/abs/1909.12264).
- [99] Pierre-Luc Dallaire-Demers and Nathan Killoran. “Quantum generative adversarial networks”. In: *Phys. Rev. A* 98 (1 2018), p. 012324. DOI: [10.1103/PhysRevA.98.012324](https://doi.org/10.1103/PhysRevA.98.012324). URL: <https://link.aps.org/doi/10.1103/PhysRevA.98.012324>.
- [100] Nico Meyer et al. *A Survey on Quantum Reinforcement Learning*. 2024. arXiv: [2211.03464 \[quant-ph\]](https://arxiv.org/abs/2211.03464). URL: <https://arxiv.org/abs/2211.03464>.
- [101] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. *A Quantum Approximate Optimization Algorithm*. 2014. arXiv: [1411.4028 \[quant-ph\]](https://arxiv.org/abs/1411.4028). URL: <https://arxiv.org/abs/1411.4028>.
- [102] Carlos Bravo-Prieto et al. “Variational Quantum Linear Solver”. In: *Quantum* 7 (Nov. 2023), p. 1188. ISSN: 2521-327X. DOI: [10.22331/q-2023-11-22-1188](https://doi.org/10.22331/q-2023-11-22-1188). URL: <http://dx.doi.org/10.22331/q-2023-11-22-1188>.
- [103] Michael Lubasch et al. “Variational quantum algorithms for nonlinear problems”. In: *Phys. Rev. A* 101 (1 2020), p. 010301. DOI: [10.1103/PhysRevA.101.010301](https://doi.org/10.1103/PhysRevA.101.010301). URL: <https://link.aps.org/doi/10.1103/PhysRevA.101.010301>.
- [104] Alberto Peruzzo et al. “A variational eigenvalue solver on a photonic quantum processor”. In: *Nature Commun.* 5.1 (2014), p. 4213. DOI: [10.1038/ncomms5213](https://doi.org/10.1038/ncomms5213). arXiv: [1304.3061 \[quant-ph\]](https://arxiv.org/abs/1304.3061).
- [105] Xiao Yuan et al. “Theory of variational quantum simulation”. In: *Quantum* 3 (Oct. 2019), p. 191. ISSN: 2521-327X. DOI: [10.22331/q-2019-10-07-191](https://doi.org/10.22331/q-2019-10-07-191). URL: <http://dx.doi.org/10.22331/q-2019-10-07-191>.

- 
- [106] Luuk Coopmans and Marcello Benedetti. “On the sample complexity of quantum Boltzmann machine learning”. In: *Communications Physics* 7.1 (Aug. 2024). ISSN: 2399-3650. DOI: [10.1038/s42005-024-01763-x](https://doi.org/10.1038/s42005-024-01763-x). URL: <http://dx.doi.org/10.1038/s42005-024-01763-x>.
- [107] Vojtěch Havlíček et al. “Supervised learning with quantum-enhanced feature spaces”. In: *Nature* 567.7747 (Mar. 2019), 209–212. ISSN: 1476-4687. DOI: [10.1038/s41586-019-0980-2](https://doi.org/10.1038/s41586-019-0980-2). URL: <http://dx.doi.org/10.1038/s41586-019-0980-2>.
- [108] Iordanis Kerenidis et al. *q-means: A quantum algorithm for unsupervised machine learning*. 2018. arXiv: [1812.03584 \[quant-ph\]](https://arxiv.org/abs/1812.03584). URL: <https://arxiv.org/abs/1812.03584>.
- [109] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. “Quantum principal component analysis”. In: *Nature Physics* 10.9 (July 2014), 631–633. ISSN: 1745-2481. DOI: [10.1038/nphys3029](https://doi.org/10.1038/nphys3029). URL: <http://dx.doi.org/10.1038/nphys3029>.
- [110] Wikipedia contributors. *Elementary particle — Wikipedia, The Free Encyclopedia*. [Online; accessed 22-October-2025]. 2025. URL: [https://en.wikipedia.org/w/index.php?title=Elementary\\_particle&oldid=1313971361](https://en.wikipedia.org/w/index.php?title=Elementary_particle&oldid=1313971361).
- [111] Daniel Johnson et al. “Exotic Hadrons at LHCb”. In: *Annual Review of Nuclear and Particle Science* 74.1 (Sept. 2024), 583–612. ISSN: 1545-4134. DOI: [10.1146/annurev-nucl-102422-040628](https://doi.org/10.1146/annurev-nucl-102422-040628). URL: <http://dx.doi.org/10.1146/annurev-nucl-102422-040628>.
- [112] Mark Thomson. *Modern particle physics*. New York: Cambridge University Press, Oct. 2013. ISBN: 978-1-107-03426-6, 978-1-139-52536-7. DOI: [10.1017/CB09781139525367](https://doi.org/10.1017/CB09781139525367).
- [113] Roel Aaij et al. “Observation of the resonant character of the  $Z(4430)^-$  state”. In: *Phys. Rev. Lett.* 112.22 (2014), p. 222002. DOI: [10.1103/PhysRevLett.112.222002](https://doi.org/10.1103/PhysRevLett.112.222002). arXiv: [1404.1903 \[hep-ex\]](https://arxiv.org/abs/1404.1903).
- [114] Georges Aad et al. “Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC”. In: *Phys. Lett. B* 716 (2012), pp. 1–29. DOI: [10.1016/j.physletb.2012.08.020](https://doi.org/10.1016/j.physletb.2012.08.020). arXiv: [1207.7214 \[hep-ex\]](https://arxiv.org/abs/1207.7214).
- [115] CERN Service graphique. “Overall view of the LHC”. In: (2014). General Photo. URL: <https://cds.cern.ch/record/1708849>.
- [116] O Brüning and L Rossi. “Chapter 1: The High Luminosity Large Hadron Collider – HL-LHC”. In: *Adv. Ser. Direct. High Energy Phys.* 31 (2024), pp. 1–53. DOI: [10.1142/9789811278952\\_0001](https://doi.org/10.1142/9789811278952_0001). URL: <https://cds.cern.ch/record/2906155>.
- [117] Atlas Collaboration et al. “Luminosity determination in  $pp$  collisions at  $\sqrt{s} = 13$  TeV using the ATLAS detector at the LHC”. In: *arXiv preprint arXiv:2212.09379* (2022).
-

- [118] *ATLAS Software and Computing HL-LHC Roadmap*. Tech. rep. Geneva: CERN, 2022. URL: <https://cds.cern.ch/record/2802918>.
- [119] Alessandro Scarabotto. “Tracking on GPU at LHCb’s fully software trigger”. In: *Proceedings of the CTD 2022 - PROC-CTD2022-28* (2022). URL: <https://cds.cern.ch/record/2823783>.
- [120] Paul Andre Günther. “LHCb’s Forward Tracking algorithm for the Run 3 CPU-based online track-reconstruction sequence”. In: (July 2022). DOI: [10.5281/zenodo.8119731](https://doi.org/10.5281/zenodo.8119731). arXiv: [2207.12965](https://arxiv.org/abs/2207.12965) [[physics.ins-det](#)].
- [121] Pierre Billoir. “Progressive track recognition with a Kalman like fitting procedure”. In: *Comput. Phys. Commun.* 57 (1989), pp. 390–394. DOI: [10.1016/0010-4655\(89\)90249-X](https://doi.org/10.1016/0010-4655(89)90249-X).
- [122] Salvatore Aiola et al. “Hybrid seeding: A standalone track reconstruction algorithm for scintillating fibre tracker at LHCb”. In: *Comput. Phys. Commun.* 260 (2021). 7 pages, 6 figures, p. 107713. DOI: [10.1016/j.cpc.2020.107713](https://doi.org/10.1016/j.cpc.2020.107713). arXiv: [2007.02591](https://arxiv.org/abs/2007.02591). URL: <https://cds.cern.ch/record/2748557>.
- [123] Serguei Chatrchyan et al. “Description and Performance of Track and Primary-Vertex Reconstruction with the CMS Tracker”. In: *JINST* 9.10 (2014), P10009. DOI: [10.1088/1748-0221/9/10/P10009](https://doi.org/10.1088/1748-0221/9/10/P10009). arXiv: [1405.6569](https://arxiv.org/abs/1405.6569) [[physics.ins-det](#)].
- [124] R. Frühwirth. “Application of Kalman filtering to track and vertex fitting”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 262.2 (1987), pp. 444–450. ISSN: 0168-9002. DOI: [https://doi.org/10.1016/0168-9002\(87\)90887-4](https://doi.org/10.1016/0168-9002(87)90887-4). URL: <https://www.sciencedirect.com/science/article/pii/0168900287908874>.
- [125] Georges Aad et al. “Software Performance of the ATLAS Track Reconstruction for LHC Run 3”. In: *Comput. Softw. Big Sci.* 8.1 (2024), p. 9. DOI: [10.1007/s41781-023-00111-y](https://doi.org/10.1007/s41781-023-00111-y). arXiv: [2308.09471](https://arxiv.org/abs/2308.09471) [[hep-ex](#)].
- [126] R. Aaij et al. “Performance of the LHCb Vertex Locator”. In: *JINST* 9 (2014), P09007. DOI: [10.1088/1748-0221/9/09/P09007](https://doi.org/10.1088/1748-0221/9/09/P09007). arXiv: [1405.7808](https://arxiv.org/abs/1405.7808) [[physics.ins-det](#)].
- [127] Arnaud Duperrin. “Flavour tagging with graph neural networks with the ATLAS detector”. In: *30th International Workshop on Deep-Inelastic Scattering and Related Subjects*. June 2023. arXiv: [2306.04415](https://arxiv.org/abs/2306.04415) [[hep-ex](#)].
- [128] Shah Rukh Qasim et al. “End-to-end multi-particle reconstruction in high occupancy imaging calorimeters with graph neural networks”. In: *Eur. Phys. J. C* 82.8 (2022), p. 753. DOI: [10.1140/epjc/s10052-022-10665-7](https://doi.org/10.1140/epjc/s10052-022-10665-7). arXiv: [2204.01681](https://arxiv.org/abs/2204.01681) [[physics.ins-det](#)].
- [129] A. Aurisano et al. “Graph neural network for neutrino physics event reconstruction”. In: *Phys. Rev. D* 110.3 (2024), p. 032008. DOI: [10.1103/PhysRevD.110.032008](https://doi.org/10.1103/PhysRevD.110.032008). arXiv: [2403.11872](https://arxiv.org/abs/2403.11872) [[physics.data-an](#)].

- 
- [130] Moritz Kiehn et al. “The TrackML high-energy physics tracking challenge on Kaggle”. In: *EPJ Web of Conferences*. Vol. 214. EDP Sciences. 2019, p. 06037.
  - [131] Anaderi et al. *TrackML Particle Tracking Challenge*. <https://kaggle.com/competitions/trackml-particle-identification>. Kaggle. 2018.
  - [132] Xiacong Ai et al. “A common tracking software project”. In: *Computing and Software for Big Science* 6.1 (2022), p. 8.
  - [133] Ulrike Von Luxburg. “A tutorial on spectral clustering”. In: *Statistics and computing* 17.4 (2007), pp. 395–416.
  - [134] Daniel Spielman. “Spectral and algebraic graph theory”. In: *Yale lecture notes* 4 (2019), p. 47.
  - [135] Steven Farrell et al. *Novel deep learning methods for track reconstruction*. 2018. arXiv: [1810.06111 \[hep-ex\]](https://arxiv.org/abs/1810.06111). URL: <https://arxiv.org/abs/1810.06111>.
  - [136] Michael M Bronstein et al. “Geometric deep learning: going beyond euclidean data”. In: *IEEE Signal Processing Magazine* 34.4 (2017), pp. 18–42.
  - [137] Marco Gori, Gabriele Monfardini, and Franco Scarselli. “A new model for learning in graph domains”. In: *Proceedings. 2005 IEEE international joint conference on neural networks, 2005*. Vol. 2. IEEE. 2005, pp. 729–734.
  - [138] Joan Bruna et al. “Spectral networks and locally connected networks on graphs”. In: *arXiv preprint arXiv:1312.6203* (2013).
  - [139] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. “Convolutional neural networks on graphs with fast localized spectral filtering”. In: *Advances in neural information processing systems* 29 (2016).
  - [140] Jonathan Masci et al. “Geodesic convolutional neural networks on riemannian manifolds”. In: *Proceedings of the IEEE international conference on computer vision workshops*. 2015, pp. 37–45.
  - [141] Davide Boscaini et al. “Learning shape correspondence with anisotropic convolutional neural networks”. In: *Advances in neural information processing systems* 29 (2016).
  - [142] Federico Monti et al. “Geometric deep learning on graphs and manifolds using mixture model cnns”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 5115–5124.
  - [143] Petar Veličković et al. “Graph attention networks”. In: *arXiv preprint arXiv:1710.10903* (2017).
  - [144] Michael Broughton et al. *TensorFlow Quantum: A Software Framework for Quantum Machine Learning*. 2021. arXiv: [2003.02989 \[quant-ph\]](https://arxiv.org/abs/2003.02989). URL: <https://arxiv.org/abs/2003.02989>.
  - [145] Cirq Developers. *Cirq*. Zenodo, 2025. DOI: [10.5281/ZENODO.4062499](https://zenodo.org/doi/10.5281/ZENODO.4062499). URL: <https://zenodo.org/doi/10.5281/zenodo.4062499>.
  - [146] Ali Javadi-Abhari et al. *Quantum computing with Qiskit*. 2024. arXiv: [2405.08810 \[quant-ph\]](https://arxiv.org/abs/2405.08810). URL: <https://arxiv.org/abs/2405.08810>.
-

- [147] James Bradbury et al. *JAX: composable transformations of Python+NumPy programs*. Version 0.3.13. 2018. URL: <http://github.com/jax-ml/jax>.
- [148] Jonathan Heek et al. *Flax: A neural network library and ecosystem for JAX*. Version 0.12.1. 2024. URL: <http://github.com/google/flax>.
- [149] Ville Bergholm et al. *PennyLane: Automatic differentiation of hybrid quantum-classical computations*. 2022. arXiv: [1811.04968 \[quant-ph\]](https://arxiv.org/abs/1811.04968). URL: <https://arxiv.org/abs/1811.04968>.
- [150] Sukin Sim, Peter D. Johnson, and Alán Aspuru-Guzik. “Expressibility and Entangling Capability of Parameterized Quantum Circuits for Hybrid Quantum-Classical Algorithms”. In: *Advanced Quantum Technologies* 2.12 (Oct. 2019). ISSN: 2511-9044. DOI: [10.1002/qute.201900070](https://doi.org/10.1002/qute.201900070). URL: <http://dx.doi.org/10.1002/qute.201900070>.
- [151] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.
- [152] Kaiming He et al. “Identity mappings in deep residual networks”. In: *European conference on computer vision*. Springer. 2016, pp. 630–645.
- [153] Hao Li et al. “Visualizing the loss landscape of neural nets”. In: *Advances in neural information processing systems* 31 (2018).
- [154] Audrey Terras. *Fourier analysis on finite groups and applications*. 43. Cambridge University Press, 1999.
- [155] Imre Risi Kondor. *Group theoretical methods in machine learning*. Columbia University, 2008.
- [156] Brian Cheung et al. “Probabilistic multi hypothesis tracker for an event based sensor”. In: *2018 21st International Conference on Information Fusion (FUSION)*. IEEE. 2018, pp. 1–8.
- [157] Samuel S Blackman and Robert Popoli. *Design and analysis of modern tracking systems*. 1999.
- [158] J. Shin et al. “Lazy inference on object identities in wireless sensor networks”. In: *IPSN 2005. Fourth International Symposium on Information Processing in Sensor Networks, 2005*. 2005, pp. 174–180. DOI: [10.1109/IPSN.2005.1440922](https://doi.org/10.1109/IPSN.2005.1440922).
- [159] Brad Schumitsch et al. “The Identity Management Kalman Filter (IMKF).” In: *Robotics: science and systems*. 2006.
- [160] Hanwen Cao, George J Pappas, and Nikolay Atanasov. “Pkf: Probabilistic data association kalman filter for multi-object tracking”. In: *IEEE Robotics and Automation Letters* (2025).
- [161] Howard Georgi. *Lie algebras in particle physics: from isospin to unified theories*. Taylor & Francis, 2000.
- [162] Jonathan Huang. *Probabilistic Reasoning and Learning on Permutations: Exploiting structural decompositions of the symmetric group*. Carnegie Mellon University, 2011.

- [163] Risi Imre Kondor and John Lafferty. “Diffusion kernels on graphs and other discrete structures”. In: *Proceedings of the 19th international conference on machine learning*. Vol. 2002. 2002, pp. 315–322.
- [164] Michael Clausen and Paul Hühne. “Linear time fourier transforms of  $\text{sn-k}$ -invariant functions on the symmetric group  $\text{sn}$ ”. In: *Proceedings of the 2017 ACM International Symposium on Symbolic and Algebraic Computation*. 2017, pp. 101–108.
- [165] Grecia Castelazo et al. “Quantum algorithms for group convolution, cross-correlation, and equivariant transformations”. In: *Physical Review A* 106.3 (Sept. 2022). ISSN: 2469-9934. DOI: [10.1103/physreva.106.032402](https://doi.org/10.1103/physreva.106.032402). URL: <http://dx.doi.org/10.1103/PhysRevA.106.032402>.
- [166] Chris Lomont. “Quantum convolution and quantum correlation algorithms are physically impossible”. In: *arXiv preprint quant-ph/0309070* (2003).
- [167] Gert Almkvist and George E Andrews. “A Hardy-Ramanujan formula for restricted partitions”. In: *Journal of Number Theory* 38.2 (1991), pp. 135–144.
- [168] SJ RE Ingram. “Some characters of the symmetric group”. In: *Proceedings of the American Mathematical Society* (1950), pp. 358–369.
- [169] Benjamin Huth. “Track reconstruction for future high-energy-physics experiments with classical and machine-learning methods”. PhD thesis. Universität Regensburg, 2024.
- [170] Adrián Pérez-Salinas et al. “Data re-uploading for a universal quantum classifier”. In: *Quantum* 4 (Feb. 2020), p. 226. ISSN: 2521-327X. DOI: [10.22331/q-2020-02-06-226](https://doi.org/10.22331/q-2020-02-06-226). URL: <http://dx.doi.org/10.22331/q-2020-02-06-226>.
- [171] Maria Schuld, Ryan Sweke, and Johannes Jakob Meyer. “Effect of data encoding on the expressive power of variational quantum-machine-learning models”. In: *Physical Review A* 103.3 (Mar. 2021). ISSN: 2469-9934. DOI: [10.1103/physreva.103.032430](https://doi.org/10.1103/physreva.103.032430). URL: <http://dx.doi.org/10.1103/PhysRevA.103.032430>.



