UNIVERSIDAD DE CONCEPCIÓN
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
Doctorado en Ciencias Físicas

ARTIFICIAL NEURAL NETWORKS FOR THE STUDY OF PHYSICAL SYSTEMS IN QUANTUM INFORMATION

Tesis presentada a la Facultad de Ciencias Físicas y Matemáticas de la Universidad de Concepción para optar al grado académico de Doctor en Ciencias Físicas

Autor : Antonio Segundo González Guerra
Profesor Guía : Esteban Fernando Sepúlveda Gómez

Junio 2026
Concepción, Chile

ACKNOWLEDGEMENTS

This thesis is the result of a journey that I did not walk alone, and these lines are a small attempt to thank those who walked it with me.

To my wife, Carolina, for her love, her patience, and her unwavering support through every long day and every late night. You have been my calm and my strength throughout this entire process, and none of this would have been the same without you by my side.

To my friends, for the laughter, the encouragement, and the welcome distractions that kept me grounded; and to my mother-in-law, for her generosity and her constant support, which I deeply appreciate.

To my pets, my faithful companions, who kept me company through every working day at home, curling up beside me during the long hours and reminding me, in their own way, to take a break now and then.

To my advisor and to the colleagues who served on my committee, for their guidance, their careful reading, and the many discussions that shaped and sharpened this work. I am grateful for their time, their insight, and their trust.

To the Universidad de Concepción, for supporting me in presenting this work at various conferences and events, an experience that enriched both this thesis and my growth as a researcher; and to ANID, for the doctoral fellowship that made this research possible, through the doctoral scholarship No. 2022-21221096.

Finally, and most especially, to my mother, may she rest in peace. For her love, her sacrifices, and everything she gave me, this achievement is as much hers as it is mine. Without her I would never have reached where I am today. This thesis is dedicated to her memory.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

RESUMEN

El crecimiento exponencial del espacio de Hilbert con el número de constituyentes cuánticos hace que numerosas tareas de información cuántica sean intratables mediante métodos clásicos exactos, lo que motiva el desarrollo de aproximaciones escalables. Esta tesis presenta dos enfoques de physics-informed deep learning que abordan este tipo de problemas incorporando los principios físicos subyacentes directamente en la arquitectura de la red neuronal y en la función objetivo de entrenamiento, garantizando que las soluciones aprendidas permanezcan consistentes con las leyes de la mecánica cuántica.

El primer enfoque estudia la dinámica de sistemas cuánticos cerrados gobernados por Hamiltonianos dependientes del tiempo, cuya evolución está descrita por la ecuación de von Neumann. En este contexto, el operador de evolución temporal se expresa mediante expansiones integrales infinitas. A diferencia del caso independiente del tiempo, donde la solución admite una forma exponencial simple, obtener una expresión análoga requiere la introducción del operador de ordenamiento temporal. En consecuencia, cuando el Hamiltoniano no conmuta consigo mismo en distintos instantes de tiempo, la complejidad matemática del problema aumenta considerablemente. El método propuesto reduce el costo computacional asociado a la simulación de estas dinámicas. A partir del operador de evolución temporal conocido únicamente en un conjunto disperso de instantes de tiempo, una Physics-Informed Neural Network interpola su evolución sobre todo el dominio temporal sin requerir conocimiento explícito del Hamiltoniano. En su lugar, el modelo incorpora la unitariedad como una restricción física y, para sistemas de mayor tamaño, predice un Hamiltoniano efectivo cuya evolución se propaga mediante integradores que preservan la estructura del sistema. Para sistemas de entre 2 y 8 qubits, el enfoque alcanza fidelidades promedio cercanas a 0.99, reproduce con precisión la dinámica incluso bajo un muestreo temporal poco denso y logra acelerar la inferencia aproximadamente en un orden de magnitud con respecto a un método de interpolación geodésica.

El segundo enfoque aborda el Quantum Marginal Problem, cuyo ob-

jetivo es reconstruir un estado cuántico global a partir de un conjunto dado de matrices densidad reducidas (marginales cuánticos), garantizando simultáneamente que todas ellas sean mutuamente compatibles. Este problema se vuelve cada vez más desafiante a medida que aumenta el tamaño del sistema debido al crecimiento exponencial del espacio de Hilbert y a las complejas restricciones de compatibilidad entre las marginales. Mediante la combinación de un convolutional denoising autoencoder con el Marginal Imposition Operator, y representando las matrices densidad como imágenes de dos canales junto con una función de pérdida que impone Hermiticidad, positividad y normalización, el modelo reconstruye estados globales físicamente válidos para sistemas de entre 3 y 8 qubits. Un esquema híbrido de reconstrucción recupera, en numerosos casos, estados cuyas marginales coinciden con las marginales objetivo con fidelidad perfecta, mientras que transfer learning permite extender el modelo a distintos tamaños de sistema con un reentrenamiento mínimo. Una vez entrenado, el enfoque realiza inferencias sustancialmente más rápidas que los solucionadores de programación semidefinida más avanzados, logrando además reconstruir estados en regímenes donde dichos métodos se vuelven computacionalmente impracticables.

En conjunto, estos resultados demuestran que la incorporación explícita de principios físicos dentro del proceso de aprendizaje permite que el deep learning actúe no como un reemplazo del modelamiento físico, sino como una extensión natural de éste. Al combinar aprendizaje basado en datos con la estructura matemática de la mecánica cuántica, los enfoques propuestos permiten abordar de manera eficiente problemas cuyo tratamiento clásico exacto resulta computacionalmente prohibitivo.

ABSTRACT

The exponential growth of the Hilbert space with the number of quantum constituents renders many quantum information tasks intractable for exact classical computation, motivating the development of scalable approximation methods. This thesis presents two physics-informed deep learning frameworks that address such challenges by embedding the underlying physical principles directly into the network architecture and training objective, ensuring that the learned solutions remain consistent with the laws of quantum mechanics.

The first framework concerns the dynamics of closed quantum systems governed by time-dependent Hamiltonians, whose evolution is described by the von Neumann equation. In this setting, the time-evolution operator is expressed through infinite integral expansions. Unlike the time-independent case, where the solution admits a simple exponential form, obtaining an analogous expression requires the introduction of the time-ordering operator. Consequently, when the Hamiltonian does not commute with itself at different times, the mathematical complexity of the problem increases significantly. The proposed method reduces the computational cost of simulating such dynamics. Given the time-evolution operator at a sparse set of sampled time points, a Physics-Informed Neural Network interpolates its evolution over the entire time domain without requiring explicit knowledge of the Hamiltonian. Instead, the model incorporates unitarity as a physical constraint and, for larger systems, predicts an effective Hamiltonian that is propagated using structure-preserving integrators. Across systems ranging from 2 to 8 qubits, the framework achieves mean gate fidelities close to 0.99, accurately reproduces the dynamics even under coarse temporal sampling, and delivers inference speedups of approximately one order of magnitude compared with a geodesic interpolation baseline.

The second framework addresses the Quantum Marginal Problem, which seeks to reconstruct a global quantum state from a prescribed set of reduced density matrices (quantum marginals) while ensuring that all marginals are mutually compatible. The problem becomes increasingly challenging as the system size grows because of the exponential scaling

of the Hilbert space and the highly nontrivial compatibility constraints among the marginals. By combining a convolutional denoising autoencoder with the Marginal Imposition Operator, and representing density matrices as two-channel images with a loss function that enforces Hermiticity, positivity, and normalization, the proposed framework reconstructs physically valid global states for systems of 3 to 8 qubits. A hybrid reconstruction scheme recovers states whose marginals match the target marginals with perfect fidelity in many cases, while transfer learning enables the model to generalize across different system sizes with only limited retraining. Once trained, the framework performs inference substantially faster than state-of-the-art semidefinite programming solvers, while successfully reconstructing states in regimes where those methods become computationally impractical.

Taken together, these results demonstrate that incorporating physical principles directly into the learning process enables deep learning to serve not as a replacement for physical modeling, but as its natural extension. By combining data-driven learning with the mathematical structure of quantum mechanics, the proposed frameworks efficiently address problems whose exact classical treatment is computationally prohibitive.

INTRODUCTION

Over the past decade, deep learning has evolved from a promising research direction into one of the most influential tools in modern science and engineering. Built upon artificial neural networks trained on large volumes of data, deep learning models have achieved, and in several cases surpassed, human-level performance on tasks once considered intractable for machines [1]. Convolutional architectures transformed computer vision, enabling accurate image classification, segmentation, and object detection [2]; sequence and attention-based models reshaped natural language processing, culminating in large language models capable of fluent reasoning, translation, and code generation [3]. Beyond these flagship applications, deep learning now underpins systems for speech recognition, recommendation, medical diagnosis, and autonomous decision-making, establishing itself as a general-purpose framework for learning complex, nonlinear relationships directly from data [4].

This success has rapidly propagated into the natural sciences, where deep learning has accelerated discovery across disciplines. Neural networks have driven breakthroughs in protein structure prediction, the discovery of new stable materials [5], and medium-range weather forecasting [6], and they have become powerful tools in fundamental physics, from the search for new particles to the analysis of gravitational-wave signals [7]. A notable development has been the emergence of physics-informed learning, in which known physical laws, expressed as conservation principles, symmetries, or differential equations, are embedded directly into the training process. Physics-Informed Neural Networks (PINNs) [8,9] exemplify this paradigm, restricting the space of admissible solutions to those consistent with the governing physics and thereby achieving accurate, data-efficient models even in regimes where purely data-driven approaches would falter. It is this physics-informed perspective that motivates and unifies the work developed in this thesis.

Within physics, quantum information science is a natural and demanding domain for these methods. Quantum systems are described by mathematical representations such as state vectors, density operators,

and unitary evolution operators, whose dimension grows exponentially with the number of constituents, rendering exact classical treatments intractable beyond modest system sizes [10]. This exponential barrier, together with the abundance of structured data produced by quantum experiments, makes quantum information a setting where the representational power and scalability of deep learning can have substantial impact. Indeed, machine learning has already been applied across a broad spectrum of quantum-information tasks: the reconstruction of quantum states from measurement data, or quantum state tomography [11, 12]; the characterization of quantum processes through quantum process tomography [13]; the solution of the quantum many-body problem via neural quantum states [14–17]; the learning of N-representability conditions in quantum chemistry [18–20]; and the control of quantum systems through physics-informed networks that solve the underlying differential equations [21]. Across these applications, a consistent theme emerges. Deep learning does not merely accelerate existing computations, but learns the latent, low-dimensional structure that often underlies high-dimensional quantum problems.

This thesis contributes to this growing body of work by developing two deep-learning frameworks for the study of physical systems in quantum information, each addressing a problem whose exact treatment is computationally demanding and each guided by the physics-informed principle of embedding known structure into the learning process.

The first problem concerns the dynamics of quantum systems governed by time-dependent Hamiltonians. Unlike the time-independent case, where the time-evolution operator reduces to a simple matrix exponential e^{-iHt}, time-dependent Hamiltonians give rise to a time-ordered exponential that precludes a closed-form solution and introduces substantial mathematical and numerical complexity. Series representations such as the Dyson or Magnus expansions involve nested time integrals whose number grows with the expansion order, and the associated numerical errors accumulate over long evolution times [22, 23]; even on quantum hardware, simulating such dynamics is generally more demanding than the time-independent case, particularly for rapidly varying or highly oscillatory Hamiltonians [24]. Despite these difficulties, time-dependent Hamiltonians are central to quantum technologies, including quantum control and gate optimization [25] and the quantum simulation of complex many-body systems [26, 27]. A variety of approaches have been developed to address this challenge, including truncated Dyson

series [28, 29], Magnus-expansion-based algorithms for quantum control [30–32], modified Suzuki–Trotter decompositions for unitary interpolation [33], and tensor-network methods [34–36], as well as machine-learning approaches that learn quantum dynamics or reconstruct time-dependent Hamiltonians [37–39]. In Chapter 4, we introduce a physics-informed deep learning framework that, given the time-evolution operator on a finite set of sampled times, interpolates it over the entire time domain, without ever being supplied the Hamiltonian explicitly, by incorporating unitarity as a physical constraint and, for larger systems, predicting an effective Hamiltonian propagated through structure-preserving integrators. This work has been submitted for publication [40].

The second problem concerns the Quantum Marginal Problem (QMP): given a set of reduced density matrices describing subsystems of a multipartite system, determining whether they are consistent with some global quantum state, and, when they are, reconstructing such a state. This is a fundamental question in quantum theory whose consistency version is complete for the Quantum Merlin–Arthur (QMA) complexity class [41, 42], reflecting its computational intractability in the general case. The QMP carries important implications. It would enable more efficient computation of ground states of local Hamiltonians [43], and it lies at the heart of quantum chemistry, where it appears as the N-representability problem [44], itself QMA-complete. Special cases have been addressed theoretically, from Klyachko's conditions for the compatibility of one-body marginals [45] to studies on the uniqueness of pure global states determined by their marginals [46–48] and semidefinite-programming formulations of symmetric instances [49]. In Chapter 5, we introduce a deep-learning approach that combines a convolutional denoising autoencoder with the Marginal Imposition Operator [50] to reconstruct physically valid global density matrices compatible with a prescribed set of marginals, for systems of up to eight qubits, and demonstrate that, once trained, it offers a faster alternative to state-of-the-art semidefinite programming solvers without compromising solution quality. This work has been published in *Machine Learning: Science and Technology* [51].

The remainder of this thesis is organized as follows. Chapters 1, 2, and 3 establish the necessary background in linear algebra, quantum information, and deep learning, respectively, providing the unified mathematical language in which the two applications are formulated. Chapter 4 presents the interpolation of unitary time-evolution operators via physics-informed neural networks, and Chapter 5 presents the recon-

struction of density matrices from quantum marginals. Finally, Chapter 6 draws together the conclusions of both works, discusses their implications for large-scale deep learning in quantum information, and outlines directions for future research.

Parte I

Theoretical background

Chapter 1
LINEAR ALGEBRA

The purpose of this chapter is to introduce the fundamental elements of linear algebra that are required in Chapters 2 and 3. Rather than providing a comprehensive and fully demonstrative exposition of linear algebra, the presentation is intentionally restricted to the definitions and structures that are directly used in the subsequent chapters.

In particular, the concepts introduced here are essential for formulating quantum states, observables, and dynamics in terms of linear operators acting on finite-dimensional Hilbert spaces, as well as for expressing neural network architectures, loss functions, and optimization procedures within a unified and mathematically transparent framework.

Throughout this chapter, all vector spaces are assumed to be finite-dimensional, which ensures the equivalence of norms and the validity of the operator representations employed in the remainder of this thesis.

1.1 Vector space

In physics, mathematical objects are assigned physical meaning in order to describe observable quantities and dynamical laws. Physical theories are therefore formulated in terms of abstract mathematical structures whose elements acquire a direct physical interpretation. A simple example arises in classical mechanics, where the position of a particle is represented by a vector in the three-dimensional Euclidean space, $\mathbb{R}^3$. In this representation, each vector consists of three real-valued components corresponding to the spatial coordinates of the particle.

This construction naturally generalizes to n dimensions, giving rise to the vector space $\mathbb{R}^n$. More generally, a vector space is defined over a field $\mathbb{F}$, whose elements serve as the scalars used in linear combinations of vectors. The most common choices are the fields of real numbers $\mathbb{R}$, and complex numbers $\mathbb{C}$. By varying both the dimension of the space and the field over which it is defined, a broad class of vector spaces can be constructed.

To provide a precise mathematical definition, it is necessary to specify not only the elements of the space, but also the operations that can be performed on them. A vector space $\mathscr{V}$ over a field $\mathbb{F}$ is therefore defined as a set equipped with vector addition and scalar multiplication satisfying the vector-space axioms, such as those presented in Ref. [52]. Throughout this work, both real and complex vector spaces are employed, with complex vector spaces playing a central role in the formulation of quantum mechanics.

In order to introduce geometric notions such as lengths, angles, and orthogonality, vector spaces are often endowed with an additional structure known as an inner product. An inner product on $\mathscr{V}$ is a map

$$\langle \cdot, \cdot \rangle : \mathscr{V} \times \mathscr{V} \to \mathbb{F}, \tag{1.1}$$

which, in an orthonormal basis of a finite-dimensional space, takes the coordinate form

$$\langle \mathbf{v}, \mathbf{w} \rangle = \sum_{i=1}^{n} v_i^* w_i = \mathbf{v}^\dagger \mathbf{w}, \tag{1.2}$$

and satisfies the following properties for all $\mathbf{u}, \mathbf{v}, \mathbf{w} \in \mathscr{V}$ and $\alpha \in \mathbb{F}$:

- **Linearity in the second argument:** $\langle \mathbf{u}, \alpha\mathbf{v} + \mathbf{w} \rangle = \alpha \langle \mathbf{u}, \mathbf{v} \rangle + \langle \mathbf{u}, \mathbf{w} \rangle$.
- **Conjugate symmetry:** $\langle \mathbf{u}, \mathbf{v} \rangle = \langle \mathbf{v}, \mathbf{u} \rangle^*$.
- **Positive definiteness:** $\langle \mathbf{v}, \mathbf{v} \rangle \geq 0$, with equality if and only if $\mathbf{v} = \mathbf{0}$.

Note that conjugate symmetry implies that $\langle \mathbf{v}, \mathbf{v} \rangle$ is real for every $\mathbf{v}$, so that the positive-definiteness condition is well defined.

The inner product induces a norm,

$$\|\mathbf{v}\| = \sqrt{\langle \mathbf{v}, \mathbf{v} \rangle}, \tag{1.3}$$

which provides a natural notion of length and distance between vectors and forms the basis for the geometric and analytical structures employed in the following chapters. More generally, a norm on $\mathscr{V}$ is any map $\|\cdot\| : \mathscr{V} \to \mathbb{R}$ satisfying:

1. **Definiteness:** $\|\mathbf{v}\| \geq 0$, with equality if and only if $\mathbf{v} = \mathbf{0}$.
2. **Homogeneity:** $\|\alpha\mathbf{v}\| = |\alpha| \|\mathbf{v}\|$.

3. **Triangle inequality:** $\|\mathbf{v}+\mathbf{w}\| \leq \|\mathbf{v}\| + \|\mathbf{w}\|$.

The norm induced by the inner product corresponds to one particular choice; not every norm arises from an inner product. In finite-dimensional spaces, the most commonly used norms are the p-norms. For a vector $\mathbf{v} \in \mathbb{F}^n$, the p-norm is defined as

$$\|\mathbf{v}\|_p = \left(\sum_{i=1}^{n} |v_i|^p \right)^{1/p}, \qquad 1 \leq p < \infty, \tag{1.4}$$

and

$$\|\mathbf{v}\|_\infty = \max_{1 \leq i \leq n} |v_i|. \tag{1.5}$$

A particularly important case is the Euclidean norm ($p = 2$), which is the one induced by the inner product and possesses a clear geometric interpretation in terms of lengths and angles.

Distances A norm naturally induces a metric (or distance function) on $\mathscr{V}$ defined by

$$d(\mathbf{v}, \mathbf{w}) = \|\mathbf{v} - \mathbf{w}\|. \tag{1.6}$$

This distance satisfies the standard metric properties:

1. $d(\mathbf{v}, \mathbf{w}) \geq 0$,
2. $d(\mathbf{v}, \mathbf{w}) = 0$ if and only if $\mathbf{v} = \mathbf{w}$,
3. $d(\mathbf{v}, \mathbf{w}) = d(\mathbf{w}, \mathbf{v})$,
4. $d(\mathbf{v}, \mathbf{u}) \leq d(\mathbf{v}, \mathbf{w}) + d(\mathbf{w}, \mathbf{u})$.

Distances provide a natural measure of similarity between elements of a vector space. The smaller the distance, the more closely the elements resemble one another. Consequently, they offer a rigorous way to quantify approximation errors and deviations. In the context of this thesis, distance measures play a fundamental role in defining loss functions, convergence criteria, and stability measures for learning algorithms and operator approximations.

1.2 Bases of vector space

To represent vectors and linear operators in a concrete and computationally useful form, it is necessary to introduce the concept of a basis of a vector space. A basis provides a minimal set of vectors from which every element of the space can be expressed uniquely as a linear combination, thereby enabling coordinate representations and matrix formulations.

Let $\mathscr{V}$ be a vector space over a field $\mathbb{F}$. A set of vectors $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n\} \subset \mathscr{V}$ is said to be *linearly independent* if the linear combination

$$\sum_{i=1}^{n} \alpha_i \mathbf{v}_i = \mathbf{0}, \tag{1.7}$$

implies $\alpha_i = 0$ for all i, with $\alpha_i \in \mathbb{F}$. If this condition is not satisfied, the set is said to be *linearly dependent*. Linear independence ensures that no vector in the set can be expressed as a linear combination of the others.

A basis of $\mathscr{V}$ is defined as a linearly independent set of vectors that spans the entire space. That is, every vector $\mathbf{v} \in \mathscr{V}$ can be written uniquely as

$$\mathbf{v} = \sum_{i=1}^{n} v_i \mathbf{v}_i, \tag{1.8}$$

where $\{v_i\}$ are the coordinates of $\mathbf{v}$ with respect to the chosen basis. The number of elements in a basis defines the dimension of the vector space.

In finite-dimensional vector spaces, all bases contain the same number of elements. Once a basis is fixed, vectors and linear operators admit concrete representations in terms of column vectors and matrices, respectively. This construction forms the foundation for the operator and matrix-based formulations employed throughout this thesis, particularly in the context of quantum mechanics and deep learning models.

1.3 Bracket notation and Hilbert spaces

In quantum mechanics, it is customary to represent vectors and linear mappings using the bracket notation introduced by Dirac. This notation provides a compact and expressive language that is particularly well suited for operator-based formulations and will be adopted throughout this thesis. The definition of linear functionals and the associated dual

space follows the presentation in Ref. [53].

Let $\mathscr{V}$ be a complex vector space. A *linear functional* on $\mathscr{V}$ is defined as a linear map

$$f : \mathscr{V} \to \mathbb{C}, \tag{1.9}$$

which assigns a complex number to each vector in $\mathscr{V}$ while preserving linearity. Linear functionals naturally arise whenever scalar quantities are associated with vectors, such as projections, expectation values, or inner products.

The set of all linear functionals on $\mathscr{V}$ forms a vector space itself, known as the *dual space* of $\mathscr{V}$ and denoted by $\mathscr{V}^*$,

$$\mathscr{V}^* = \{f : \mathscr{V} \to \mathbb{C} \mid f \text{ is linear}\}. \tag{1.10}$$

Elements of the dual space act on vectors to produce scalar values and play a fundamental role in the formulation of quantum mechanics.

Once a basis of $\mathscr{V}$ is fixed, vectors admit a coordinate representation as column vectors, while linear functionals in the dual space are naturally represented as row vectors. This distinction reflects their different algebraic roles. Vectors represent elements of the space itself, whereas linear functionals act on vectors to produce scalars. In this representation, the action of a functional on a vector corresponds to the matrix product between a row vector and a column vector, yielding a scalar. This row–column distinction is a feature of the chosen representation, not of the abstract definitions of $\mathscr{V}$ and $\mathscr{V}^*$.

When working with Hilbert spaces, it is convenient to adopt Dirac's bra-ket notation, which provides a compact representation of vectors, linear functionals, and inner products. In this notation, vectors in $\mathscr{V}$ are denoted by *kets* $|v\rangle$, while linear functionals acting on $\mathscr{V}$ are denoted by *bras* $\langle v|$, which belong to the dual space $\mathscr{V}^*$.

When $\mathscr{V}$ is endowed with an inner product, every ket $|v\rangle$ uniquely defines a corresponding bra through the mapping

$$|u\rangle \mapsto \langle v|u\rangle\,. \tag{1.11}$$

This correspondence, known as the *Riesz representation theorem*, establishes a one-to-one, conjugate-linear mapping between vectors in $\mathscr{V}$ and linear functionals in $\mathscr{V}^*$, allowing both objects to be treated within a unified mathematical framework.

The inner product between two vectors $|u\rangle$ and $|v\rangle$ is written as

$$\langle u|v\rangle\,, \tag{1.12}$$

which is linear in its second argument and conjugate-linear in its first, following the convention adopted throughout quantum mechanics.

A complex vector space equipped with an inner product and complete with respect to the norm induced by that inner product, that is, such that every Cauchy sequence converges to an element of the space, is called a *Hilbert space*. In this work, all Hilbert spaces are assumed to be finite-dimensional, so that completeness is automatically guaranteed and no technical distinction between inner-product spaces and Hilbert spaces is required.

The Hilbert space formalism provides the natural mathematical setting for the description of quantum states and linear operators. At the same time, its finite-dimensional structure enables a direct correspondence with matrix representations and numerical implementations, which is essential for the learning-based approaches developed in the subsequent chapters.

1.4 Linear operators and matrix representations

A natural extension of the concept of vector spaces is given by linear operators acting on them. A linear operator is a map

$$A : \mathscr{V} \to \mathscr{V}, \tag{1.13}$$

which associates to each vector in the vector space another vector in the same space. Linearity requires that, for all $|v\rangle\,, |w\rangle \in \mathscr{V}$ and scalars $\alpha, \beta \in \mathbb{F}$,

$$A\big(\alpha\,|v\rangle + \beta\,|w\rangle\big) = \alpha A\,|v\rangle + \beta A\,|w\rangle\,. \tag{1.14}$$

Once an orthonormal basis $\{|i\rangle\}_{i=1}^{n}$ of the n-dimensional vector space $\mathscr{V}$ is fixed, any linear operator admits a matrix representation with respect to that basis. The matrix elements of the operator A are defined by

$$A_{ij} \equiv \langle i|A|j\rangle\,, \tag{1.15}$$

which fully characterize the action of the operator. Indeed, the action of A on an arbitrary vector $|v\rangle = \sum_j v_j |j\rangle$ can be written as

$$A|v\rangle = \sum_{i,j} \langle i|A|j\rangle \, v_j \, |i\rangle \,, \tag{1.16}$$

showing explicitly that the abstract operator action corresponds, in a chosen basis, to standard matrix multiplication.

This correspondence allows linear operators to be treated equivalently as matrices acting on column vectors of components, while preserving the basis-independent operator formulation. Throughout this work, operator expressions are written in basis-independent form whenever possible, with matrix representations introduced only when required for explicit calculations or numerical implementations.

In a fixed orthonormal basis, that is, a basis whose elements are normalized and mutually orthogonal, the adjoint operator corresponds to the conjugate transpose of the matrix representation of A. Explicitly, the matrix elements of the adjoint operator satisfy

$$\langle i|A^\dagger|j\rangle = \langle j|A|i\rangle^* \,. \tag{1.17}$$

Operators satisfying $A = A^\dagger$ are called *Hermitian* and play a fundamental role in quantum mechanics, where they represent physical observables. In contrast, operators obeying

$$A^\dagger A = A A^\dagger = \mathbb{I} \tag{1.18}$$

are called *unitary* and describe reversible transformations that preserve inner products and norms. As a consequence, unitary operators generate norm-preserving dynamics and constitute the mathematical framework for the time evolution of closed quantum systems.

Within the scope of this thesis, linear operators provide a unified language for describing quantum states, observables, and dynamical generators. At the same time, their matrix representations enable direct connections with numerical methods and learning-based models, which are essential for the computational approaches developed in the subsequent chapters.

1.5 Spectral decomposition

Since a vector space generally admits infinitely many possible bases, it is often advantageous to select a basis that is naturally adapted to the linear transformations acting on the space. In this context, the spectral decomposition provides a particularly useful representation, as it expresses a linear operator in terms of its eigenvectors, which define invariant directions under the action of the operator. When such a decomposition exists, the eigenvectors form a distinguished basis that diagonalizes the operator and reveals its intrinsic structure.

Given a linear operator $A : \mathscr{V} \to \mathscr{V}$, a nonzero vector $|v\rangle \in \mathscr{V}$ is called an eigenvector of A with corresponding eigenvalue $\lambda \in \mathbb{C}$ if it satisfies

$$A|v\rangle = \lambda |v\rangle . \tag{1.19}$$

Eigenvectors characterize directions that are left invariant by the action of the operator and play a central role in the analysis of linear transformations.

When the operator A is Hermitian, its spectral properties exhibit particularly strong structure. In finite-dimensional Hilbert spaces, Hermitian operators possess real eigenvalues and admit a complete set of eigenvectors. Moreover, eigenvectors associated with distinct eigenvalues are orthogonal with respect to the inner product. As a consequence, the eigenvectors of a Hermitian operator can be chosen to form an orthonormal basis of the space.

This property allows the operator to be expressed in terms of mutually orthogonal projectors. In particular, any Hermitian operator A admits a spectral decomposition of the form

$$A = \sum_i \lambda_i |i\rangle \langle i| , \tag{1.20}$$

where $\{\lambda_i\}$ are the real eigenvalues of A and $\{|i\rangle\}$ is an orthonormal basis of eigenvectors. Each term $|i\rangle \langle i|$ represents an orthogonal projector onto the one-dimensional subspace spanned by the corresponding eigenvector.

The spectral decomposition provides a basis-independent characterization of the operator, revealing its action as a weighted sum of projections along mutually orthogonal directions. In quantum mechanics, this decomposition acquires direct physical significance: Hermitian opera-

tors represent observables, the eigenvalues correspond to possible measurement outcomes, and the associated projectors encode the idealized state-update rule induced by measurements.

In the context of this thesis, the spectral decomposition plays a fundamental role in the analysis of quantum states, density operators, and dynamical generators, and constitutes a key mathematical tool for both analytical developments and numerical implementations.

1.6 Matrix exponential

The matrix exponential is one of the central mathematical tools for describing continuous transformations and time evolution in Physics. In Quantum Mechanics, in particular, evolution operators are naturally expressed as exponentials of linear operators.

Although writing an exponential with a matrix in the exponent may initially appear formal, it is rigorously defined through the Taylor series expansion of the exponential function. For a general square matrix $A \in \mathbb{C}^{d\times d}$, the matrix exponential is defined as

$$e^A = \sum_{n=0}^{\infty} \frac{A^n}{n!}, \tag{1.21}$$

where $A^0 = \mathbb{I}$ and A^n denotes the n-th matrix power. Since the series converges absolutely for all finite matrices, the matrix exponential is well defined for any finite-dimensional linear operator.

Spectral representation If A is normal, that is, $AA^\dagger = A^\dagger A$ (in particular, if A is Hermitian), the computation of matrix functions becomes particularly simple, since the spectral theorem guarantees an orthonormal eigenbasis. Suppose

$$A = \sum_i \lambda_i |i\rangle \langle i| , \tag{1.22}$$

where $\{|i\rangle\}$ is an orthonormal basis of eigenvectors and λ_i are the corresponding eigenvalues. Then, for any analytic function f, one defines

$$f(A) = \sum_i f(\lambda_i) |i\rangle \langle i| . \tag{1.23}$$

In particular, the exponential of A can be written as

$$e^A = \sum_i e^{\lambda_i} |i\rangle \langle i| . \tag{1.24}$$

This result follows directly from substituting the spectral decomposition of A into the Taylor series definition.

Hermitian generators and unitary operators A particularly important case arises when A is Hermitian, i.e., $A^\dagger = A$. In this situation, all eigenvalues λ_i are real. Consider now the operator

$$U = e^{-iA}. \tag{1.25}$$

Using the spectral representation,

$$U = \sum_i e^{-i\lambda_i} |i\rangle \langle i| . \tag{1.26}$$

Since $|e^{-i\lambda_i}| = 1$ for real λ_i, it immediately follows that

$$U^\dagger U = \mathbb{I}, \tag{1.27}$$

which means that U is unitary. More generally, for any matrix A,

$$\left(e^A\right)^\dagger = e^{A^\dagger}. \tag{1.28}$$

Therefore, if A is Hermitian, e^{-iA} is unitary.

Connection with dynamical evolution The matrix exponential plays a fundamental role in the description of continuous dynamical processes generated by linear operators. Whenever the evolution of a system is governed by a linear generator A, the associated evolution operator can often be written in the form

$$U(t) = e^{tA}. \tag{1.29}$$

If the generator is skew-Hermitian, that is, of the form $A = -iH$ with H Hermitian, the exponential $U(t) = e^{tA}$ is unitary and therefore norm-preserving. This property ensures the mathematical consistency of the

evolution and the stability of the underlying dynamics.

In many physical applications, including quantum mechanics, evolution operators naturally take this exponential form. When the generator depends explicitly on time, however, additional subtleties arise due to possible non-commutativity at different times. This leads to more sophisticated constructions, such as time-ordered exponentials and the Magnus expansion, which will be discussed in later sections.

1.7 Tensor products

In many areas of Physics and Mathematics, and particularly in Quantum Mechanics, the tensor product provides the natural mathematical structure to describe composite systems. When two subsystems are combined, their corresponding vector spaces are not added but rather tensored, producing a larger space capable of encoding correlations and non-classical features.

Let V and W be vector spaces over the same field $\mathbb{F}$. The tensor product space, denoted by $V \otimes W$, is a new vector space together with a bilinear map

$$\otimes : V \times W \longrightarrow V \otimes W,$$

such that for all $\mathbf{v}, \mathbf{v}' \in V$, $\mathbf{w}, \mathbf{w}' \in W$, and scalars $\alpha, \beta \in \mathbb{F}$, the following bilinearity conditions hold:

$$\begin{aligned}(\alpha\mathbf{v} + \beta\mathbf{v}') \otimes \mathbf{w} &= \alpha(\mathbf{v} \otimes \mathbf{w}) + \beta(\mathbf{v}' \otimes \mathbf{w}),\\ \mathbf{v} \otimes (\alpha\mathbf{w} + \beta\mathbf{w}') &= \alpha(\mathbf{v} \otimes \mathbf{w}) + \beta(\mathbf{v} \otimes \mathbf{w}').\end{aligned}$$

Elements of the form $\mathbf{v} \otimes \mathbf{w}$ are called *simple tensors*. A general element of $V \otimes W$ is a finite linear combination of simple tensors,

$$\mathbf{z} = \sum_k \mathbf{v}_k \otimes \mathbf{w}_k.$$

If $\{\mathbf{e}_i\}_{i=1}^n$ is a basis of V and $\{\mathbf{f}_j\}_{j=1}^m$ is a basis of W, then the set

$$\{\mathbf{e}_i \otimes \mathbf{f}_j\}_{i,j}$$

forms a basis of $V \otimes W$. Consequently,

$$\dim(V \otimes W) = \dim(V)\,\dim(W).$$

In finite-dimensional spaces with column-vector representations, if

$$\mathbf{v} = \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix}, \qquad \mathbf{w} = \begin{pmatrix} w_1 \\ \vdots \\ w_m \end{pmatrix},$$

their tensor product (also known as the Kronecker product in matrix representation) is given by

$$\mathbf{v} \otimes \mathbf{w} = \begin{pmatrix} v_1\mathbf{w} \\ v_2\mathbf{w} \\ \vdots \\ v_n\mathbf{w} \end{pmatrix} = \begin{pmatrix} v_1 w_1 \\ \vdots \\ v_1 w_m \\ v_2 w_1 \\ \vdots \\ v_n w_m \end{pmatrix}.$$

This construction naturally extends to linear operators. If $A : V \to V$ and $B : W \to W$ are linear maps, their tensor product $A \otimes B$ acts on simple tensors as

$$(A \otimes B)(\mathbf{v} \otimes \mathbf{w}) = (A\mathbf{v}) \otimes (B\mathbf{w}), \tag{1.30}$$

and extends linearly to the entire space.

The tensor product is therefore the fundamental algebraic mechanism that allows one to construct structured, high-dimensional spaces from simpler building blocks. In the context of composite quantum systems, this structure becomes essential and will be introduced formally in the next chapter.

1.8 Differentiation with respect to vectors and matrices

A central task in Machine Learning and Deep Learning consists of optimizing a large number of parameters in order to construct models capable of approximating complex mappings. This optimization process requires evaluating and differentiating functions whose arguments are vectors or matrices. Throughout this section the underlying field is taken to be real, $\mathbb{F} = \mathbb{R}$; complex-valued quantities, such as the density matrices and unitary operators appearing in later chapters, are handled by

treating their real and imaginary parts as independent real variables.

Derivatives with respect to vectors Let $f : \mathbb{F}^n \to \mathbb{F}$ be a scalar-valued function of a vector $\mathbf{x} = (x_1, \ldots, x_n)^\top$. If f is differentiable with respect to each component, its derivative is described by the *gradient*, defined as

$$\nabla f(\mathbf{x}) := \begin{pmatrix} \dfrac{\partial f}{\partial x_1} \\ \vdots \\ \dfrac{\partial f}{\partial x_n} \end{pmatrix}. \tag{1.31}$$

The gradient encodes the direction of steepest ascent of the function. In optimization problems, this object determines how parameters must be updated in order to minimize (or maximize) a given objective function. Iterative schemes such as gradient descent rely precisely on this geometric interpretation.

More generally, if $\mathbf{F} : \mathbb{F}^n \to \mathbb{F}^m$ is a vector-valued function, its derivative is represented by the *Jacobian matrix*,

$$J_{\mathbf{F}}(\mathbf{x}) = \left(\frac{\partial F_i}{\partial x_j} \right)_{i,j}. \tag{1.32}$$

The Jacobian provides the best linear approximation of $\mathbf{F}$ around a point and generalizes the usual one-dimensional derivative to higher dimensions. In local coordinates, it describes how small variations in the input propagate to variations in the output.

Matrix derivatives In many applications, the parameters of interest are arranged in matrix form. Let $f : \mathbb{F}^{m\times n} \to \mathbb{F}$ be a scalar function of a matrix $X = (X_{ij})$. The derivative of f with respect to X is defined componentwise as

$$\left(\frac{\partial f}{\partial X} \right)_{ij} = \frac{\partial f}{\partial X_{ij}}, \tag{1.33}$$

producing a matrix of the same size as X.

In practice, matrix derivatives are most conveniently handled using trace identities. For instance, if A is a constant matrix of compatible

dimension, the identity

$$\frac{\partial}{\partial X}\mathrm{Tr}(A^\top X) = A \tag{1.34}$$

follows directly from the linearity of the trace and provides a compact way to compute gradients.

Two frequently used examples illustrate this approach. The Frobenius norm of a matrix is defined by

$$\|X\|_F = \sqrt{\mathrm{Tr}(X^\top X)}, \tag{1.35}$$

and its squared value satisfies

$$\frac{\partial}{\partial X}\|X\|_F^2 = 2X. \tag{1.36}$$

Similarly, for a quadratic form of the type $\mathrm{Tr}(X^\top AX)$, one obtains

$$\frac{\partial}{\partial X}\mathrm{Tr}(X^\top AX) = AX + A^\top X = (A + A^\top)X. \tag{1.37}$$

These identities appear repeatedly in optimization problems involving least-squares objectives and operator learning.

Differentials and linear approximation A conceptually cleaner formulation of derivatives in finite-dimensional spaces is given by the *differential*. The differential of a function f at a point X in the direction of a perturbation H is defined as

$$df(X)[H] = \frac{d}{d\varepsilon} f(X + \varepsilon H)\bigg|_{\varepsilon=0}. \tag{1.38}$$

This definition emphasizes that the derivative is fundamentally a linear map acting on perturbations. If f is differentiable, there exists a matrix $\nabla_X f$ such that

$$df(X)[H] = \mathrm{Tr}\Big((\nabla_X f)^\top H\Big), \tag{1.39}$$

which identifies $\nabla_X f$ as the gradient of f with respect to X.

This differential viewpoint is especially convenient when dealing with operator-valued functions, variational formulations, and loss functions defined on spaces of matrices. It provides a compact and coordinate-independent framework that will be essential in the derivation of learning rules and operator optimization procedures developed in later chapters.

1.9 Summary

This chapter introduced the finite-dimensional linear algebraic framework that underpins the theoretical and computational developments presented in the remainder of this thesis. Rather than pursuing a fully general or abstract treatment, the exposition was intentionally restricted to the structures that are directly required for quantum information theory and deep learning models.

The exposition began with vector spaces over $\mathbb{R}$ and $\mathbb{C}$, emphasizing inner products, induced norms, and metric structures. These notions provide the geometric foundation necessary to quantify lengths, angles, orthogonality, and approximation errors, concepts that are central both to quantum state representations and to the definition of loss functions in learning problems.

The introduction of bases enabled coordinate representations of vectors and linear maps, leading naturally to matrix formulations. Through Dirac's bracket notation and the identification between vectors and elements of the dual space, the Hilbert space framework that serves as the mathematical setting for quantum states and observables was established.

Linear operators were then examined both abstractly and through their matrix representations. Special attention was given to Hermitian and unitary operators, whose structural properties play a fundamental role in quantum mechanics and in norm-preserving transformations. The spectral decomposition was presented as a powerful tool for revealing intrinsic operator structure and for defining analytic functions of operators, including the matrix exponential.

The matrix exponential was introduced as the natural mechanism for generating continuous linear transformations. Its spectral characterization and its relation to unitary dynamics provide the mathematical foundation for describing time evolution processes, which will be further

developed in subsequent chapters.

Tensor products were then introduced as the algebraic construction that allows composite systems to be modeled within a unified vector space framework. This structure is essential for describing multipartite quantum systems and for understanding the exponential growth of state-space dimension.

Finally, differentiation with respect to vectors and matrices was presented, emphasizing gradients, Jacobians, trace-based identities, and the differential formulation. These tools constitute the mathematical backbone of optimization procedures and learning algorithms that operate on high-dimensional parameter spaces.

Collectively, the concepts developed in this chapter establish a unified linear-algebraic language in which quantum mechanics and machine learning can be expressed within the same formal framework. This shared structure enables the analytical derivations and computational implementations that will be developed in the following chapters.

Chapter 2
QUANTUM INFORMATION

This chapter introduces the fundamental concepts of quantum information theory required to describe and analyze the applications developed in this thesis.

In much of the quantum mechanics literature, the postulates of the theory are formulated in terms of the wave function, commonly denoted by $|\psi\rangle$. In this approach, the wave function, representing the state of the isolated physical system, is described as a vector in a Hilbert space, namely a complex inner-product space that is complete with respect to the norm induced by the inner product. While this formulation is sufficient to describe a wide range of quantum phenomena, it does not constitute the most general framework.

In the context of quantum information theory, a more comprehensive description is obtained by employing the density-operator formalism, which extends the state description beyond pure states and naturally incorporates statistical mixtures. For this reason, a density-operator–centric formulation of quantum mechanics is adopted, in which all physical states are represented by density operators and quantum dynamics is described as linear maps acting on them.

2.1 Postulates

A central aspect in the formulation of any mathematical model describing the physical behavior of a system is the precise specification of its postulates, namely a set of fundamental assumptions from which the entire theoretical framework is derived. In the case of quantum mechanics, such a formulation is not unique, as different equivalent formulations may be employed to explain experimental outcomes. In this work, the formulation presented in Ref. [54] is adopted, which constitutes a standard and widely accepted reference in the field of quantum information theory.

Postulate 1

The density operator was originally introduced to describe ensembles of quantum states. More generally, the quantum state of an isolated physical system can be equivalently represented in terms of a density operator. In this formalism, an operator ρ represents a statistical ensemble $\{p_i, |\psi_i\rangle\}$ and is defined by the following properties:

(1) Hermiticity: $\rho = \rho^\dagger$.

(2) Normalization: $\mathrm{Tr}(\rho) = 1$.

(3) Positivity: ρ is positive semidefinite, i.e., $\langle\psi|\,\rho\,|\psi\rangle \geq 0$ for all $|\psi\rangle$, or equivalently, all its eigenvalues satisfy $\lambda_i \geq 0$.

Under these conditions, the density operator ρ can be written as

$$\rho \equiv \sum_i p_i \,|\psi_i\rangle\,\langle\psi_i|\,, \tag{2.1}$$

where $\{p_i\}$ is a probability distribution satisfying $p_i \geq 0$ and $\sum_i p_i = 1$.

If the quantum system is known to be in a definite state $|\psi\rangle$, the state is said to be *pure*, and the corresponding density operator reduces to $\rho = |\psi\rangle\,\langle\psi|$. Otherwise, the density operator represents a *mixed state*, corresponding either to a statistical description arising from classical uncertainty about the underlying pure state or to the reduced state of a system entangled with external degrees of freedom.

Postulate 2

The evolution of a closed quantum system is described by a unitary transformation. Specifically, if the system is in the state ρ_0 at an initial time t_0, its state at a later time t is given by

$$\rho(t) = U(t,t_0)\,\rho_0\,U^\dagger(t,t_0), \tag{2.2}$$

where $U(t,t_0)$ denotes the unitary time-evolution operator. This postulate applies exclusively to closed quantum systems; extensions to open-system dynamics are discussed separately.

Postulate 3

Quantum measurements are described by a collection of measurement operators $\{M_m\}$ acting on the system Hilbert space and satisfying the completeness relation

$$\sum_m M_m^\dagger M_m = \mathbb{I}, \tag{2.3}$$

where $\mathbb{I}$ is the identity operator.

If the quantum system is described by a density operator ρ, the probability of obtaining the measurement outcome labeled by m is given by

$$p(m) = \mathrm{Tr}\left(M_m \rho M_m^\dagger\right). \tag{2.4}$$

Conditioned on obtaining the outcome m, the post-measurement state of the system is

$$\rho_m = \frac{M_m \rho M_m^\dagger}{\mathrm{Tr}\left(M_m \rho M_m^\dagger\right)}. \tag{2.5}$$

A particularly important class of measurements corresponds to projective measurements associated with observables. Any observable O is represented by a Hermitian operator admitting a spectral decomposition of the form

$$O = \sum_m o_m \Pi_m, \tag{2.6}$$

where $\{o_m\}$ are the real eigenvalues corresponding to the possible outcomes of the measurement and $\{\Pi_m\}$ are orthogonal projection operators satisfying $\Pi_m \Pi_{m'} = \delta_{mm'} \Pi_m$ and $\sum_m \Pi_m = \mathbb{I}$.

In this case, the measurement operators reduce to $M_m = \Pi_m$, and the probability of obtaining the outcome o_m is given by

$$p(m) = \mathrm{Tr}(\rho\, \Pi_m). \tag{2.7}$$

The expectation value of the observable O with respect to the quantum state ρ is defined as

$$\langle O \rangle = \sum_m o_m\, p(m) = \mathrm{Tr}(\rho\, O). \tag{2.8}$$

In this work, quantum measurements are employed primarily through expectation values and statistical outcomes, without explicitly modeling

the state update induced by measurement.

Postulate 4

For a composite physical system, the state space is described as the tensor product of the Hilbert spaces associated with its subsystems. If the composite system consists of n subsystems, each described by a density operator ρ_i, and the subsystems are prepared independently, the joint state of the global system is given by

$$\rho = \bigotimes_{i=1}^{n} \rho_i. \tag{2.9}$$

In general, the joint state of a composite system need not be of product form and may exhibit quantum correlations, including entanglement.

2.2 Density operators: structure and properties

Section 2.1 introduced the density operator formalism. This section discusses several structural properties and derived quantities that follow directly from the definition of the density operator and will be used throughout this work.

Spectral decomposition. Since ρ is Hermitian, the spectral theorem discussed in Section 1.5 guarantees that any density operator acting on a d-dimensional Hilbert space admits a spectral decomposition of the form

$$\rho = \sum_{i=1}^{d} \lambda_i |i\rangle \langle i|, \tag{2.10}$$

where $\{\lambda_i\}_{i=1}^{d}$ are the eigenvalues of ρ and $\{|i\rangle\}_{i=1}^{d}$ is an orthonormal eigenbasis of the system Hilbert space. The eigenvalues satisfy $\lambda_i \geq 0$ and $\sum_{i=1}^{d} \lambda_i = 1$.

Moreover, the density operator ρ represents a pure quantum state if and only if it has rank one, which is equivalent to having a single eigenvalue equal to 1 and all remaining eigenvalues equal to 0.

Purity. The purity of a quantum state described by a density operator ρ is defined as

$$\gamma(\rho) = \mathrm{Tr}\Big(\rho^2\Big). \tag{2.11}$$

This quantity satisfies $\frac{1}{d} \leq \gamma(\rho) \leq 1$ for a d-dimensional system, with $\gamma(\rho) = 1$ corresponding to pure states and the lower bound $\gamma(\rho) = 1/d$ attained by the maximally mixed state $\rho = \mathbb{I}/d$; smaller values indicate increasing degrees of mixedness. In particular, if $\gamma(\rho) < 1$, the density operator cannot be written in the form $\rho = |\psi\rangle\langle\psi|$ for some state vector $|\psi\rangle$.

Ensemble representations. While any density operator can be written as a convex combination of pure states, such a decomposition is not unique. In contrast, the spectral decomposition of ρ is unique up to degeneracies in the eigenvalue spectrum. This distinction plays an important role in the interpretation of mixed states and in practical quantum state reconstruction tasks.

Hilbert–Schmidt inner product. The space of linear operators acting on the system Hilbert space can be endowed with the Hilbert–Schmidt inner product, defined as

$$\langle A, B\rangle_{\mathrm{HS}} := \mathrm{Tr}\Big(A^\dagger B\Big). \tag{2.12}$$

This inner product induces a natural notion of orthogonality between operators and will be used extensively in the representation of quantum states, Hamiltonians, and dynamical generators.

Corollary (Born rule for pure states). For a generalized measurement described by operators $\{M_m\}$ and a pure state $\rho = |\psi\rangle\langle\psi|$, the

probability of obtaining outcome m is

$$\begin{aligned} p(m) &= \mathrm{Tr}\Big(M_m \rho M_m^\dagger\Big) \\ &= \mathrm{Tr}\Big(M_m \left|\psi\right\rangle \left\langle\psi\right| M_m^\dagger\Big) \\ &= \mathrm{Tr}\Big(\left|\psi\right\rangle \left\langle\psi\right| M_m^\dagger M_m\Big) \qquad \text{(cyclicity of the trace)} \\ &= \left\langle\psi\right| M_m^\dagger M_m \left|\psi\right\rangle \\ &= \|M_m \left|\psi\right\rangle\|^2. \end{aligned} \tag{2.13}$$

This expression recovers the standard form of the Born rule for pure states, as commonly presented in wave-function–based formulations of quantum mechanics.

2.3 Composite systems and Partial trace

Postulate 4 defines how to construct joint states for composite systems from the Hilbert spaces of their subsystems. However, in many situations one is interested in describing only a subsystem of a larger quantum system. In such cases, the reduced density operator provides a systematic way to obtain the state of a subsystem from the density operator of the composite system.

Partial trace. Let $\{\left|j\right\rangle_B\}$ be an orthonormal basis of the Hilbert space associated with subsystem B. The partial trace over subsystem B is defined as

$$\mathrm{Tr}_B\Big(\rho^{AB}\Big) := \sum_j (\mathbb{I}_A \otimes \left\langle j\right|_B)\, \rho^{AB}\, (\mathbb{I}_A \otimes \left|j\right\rangle_B)\,. \tag{2.14}$$

This operation yields an operator acting on the Hilbert space of subsystem A and preserves Hermiticity, positivity, and trace. The reduced operator is therefore itself a valid density operator.

Reduced density operator. The reduced density operator associated with subsystem A is defined as

$$\rho^A \equiv \mathrm{Tr}_B\Big(\rho^{AB}\Big)\,, \tag{2.15}$$

where ρ^{AB} denotes the density operator describing the joint system composed of subsystems A and B.

Reduced states and expectation values. The reduced density operator ρ^A fully characterizes all measurement statistics associated with observables acting solely on subsystem A. In particular, for any observable O_A acting on subsystem A, one has

$$\mathrm{Tr}\Big(\rho^{AB}\,(O_A \otimes \mathbb{I}_B)\Big) = \mathrm{Tr}\Big(\rho^A\, O_A\Big)\,. \tag{2.16}$$

This property justifies the interpretation of ρ^A as the effective quantum state of subsystem A.

Entanglement. Entanglement is one of the most intensively studied resources in quantum information theory, as it captures the existence of correlations in composite quantum systems that have no classical counterpart. These correlations manifest themselves in the measurement outcomes of entangled quantum states and cannot be explained within the framework of classical physics, thus highlighting one of the most fundamental differences between classical and quantum theories.

From a formal perspective, entanglement is naturally defined within the density operator formalism. Consider a bipartite quantum system composed of two subsystems A and B, described by a joint density operator ρ_{AB} acting on the composite Hilbert space $\mathscr{H}_A \otimes \mathscr{H}_B$. The state ρ_{AB} is said to be *separable* if it can be expressed as a convex combination of product states, namely,

$$\rho_{AB} = \sum_k p_k\, \rho_A^{(k)} \otimes \rho_B^{(k)}, \qquad p_k \geq 0, \quad \sum_k p_k = 1, \tag{2.17}$$

where $\rho_A^{(k)}$ and $\rho_B^{(k)}$ are valid density matrices associated with subsystems A and B, respectively.

Separable states describe correlations that can be interpreted as purely classical, arising from statistical mixtures of local quantum states. In contrast, if the joint state ρ_{AB} cannot be written in the form of Eq. (2.17), it is classified as an *entangled* state. Entangled states exhibit intrinsically quantum correlations that cannot be reproduced by any classical probabilistic model based on local descriptions of the subsystems.

A particularly relevant property arises when considering pure states. If the joint state ρ_{AB} is pure and separable, i.e., of the form $\rho_{AB} = \rho_A \otimes \rho_B$, then the reduced state ρ_A is also pure. In contrast, when the joint state is entangled, the subsystems cannot be described independently, and the reduced density operator ρ_A is necessarily mixed, even if the global state ρ_{AB} is pure. This demonstrates that mixed states may arise not only from classical statistical uncertainty but also as a direct physical consequence of entanglement with external degrees of freedom.

The quantum marginal problem. When dealing with composite quantum systems, a natural question arises regarding the relationship between global states and their reduced descriptions. Given a multipartite quantum state $\rho_{AB\ldots}$, its reduced states (or *marginals*) are obtained by taking partial traces over subsystems. While this procedure is straightforward, the inverse problem is highly nontrivial.

The *quantum marginal problem* asks whether a given set of reduced density matrices is compatible with a global quantum state. More precisely, given a collection of density operators describing subsystems of a composite system, one seeks to determine whether there exists a joint density operator whose reduced states coincide with the given marginals.

The quantum marginal problem exhibits genuinely quantum features with no classical counterpart. A representative example is the monogamy of entanglement. If two subsystems A and B are maximally entangled, then neither can be entangled with a third subsystem C, which constrains the compatibility of overlapping marginals such as ρ_{AB} and ρ_{BC}. As a consequence, reduced density matrices do not, in general, uniquely determine the global state, and different global states may share the same set of marginals.

The quantum marginal problem highlights fundamental limitations in the reconstruction of global quantum states from local information and plays an important role in quantum information theory, quantum chemistry, and many-body physics. In particular, it provides a conceptual framework for understanding the emergence of mixed reduced states, the role of entanglement, and the challenges associated with state and process reconstruction in composite quantum systems. This problem will be discussed in more detail in the following chapters.

2.4 Quantum dynamics of density matrices

Equation of motion for the density operator. *Postulate 2* establishes the time evolution of quantum states by relating the state of a system at different times. In quantum mechanics, however, several equivalent formulations, known as *pictures*, exist, which differ in how the time dependence is distributed between states and observables. The two most commonly used formulations are the *Schrödinger picture* and the *Heisenberg picture*. In the Schrödinger picture, the time dependence is entirely carried by the quantum states, while observables are taken to be time-independent, unless they possess an explicit time dependence.

Throughout this work, the Schrödinger picture is adopted. Within this framework, the evolution of a closed quantum system is described by unitary operators acting on the state of the system, which may be represented either by state vectors or, more generally, by density operators.

To derive the equation of motion for the density operator, consider a state vector $|\psi(t)\rangle$ at time t, represented as a normalized vector in the system Hilbert space. Assuming a (possibly time-dependent) Hamiltonian $H(t)$, the Schrödinger equation reads

$$i\frac{\mathrm{d}}{\mathrm{d}t}|\psi(t)\rangle = H(t)\,|\psi(t)\rangle\,, \tag{2.18}$$

where $\hbar = 1$ is set.

The formal solution of Eq. (2.18) can be written in terms of the unitary time-evolution operator $U(t,t_0)$, which maps the state at an initial time t_0 to the state at time t,

$$|\psi(t)\rangle = U(t,t_0)\,|\psi(t_0)\rangle\,. \tag{2.19}$$

Substituting Eq. (2.19) into Eq. (2.18) yields the operator equation governing the time-evolution operator,

$$i\frac{\mathrm{d}}{\mathrm{d}t}U(t,t_0) = H(t)\,U(t,t_0), \tag{2.20}$$

supplemented with the initial condition $U(t_0,t_0) = \mathbb{I}$.

The density operator $\rho(t)$ describing the state of the system at time

t is related to the initial density operator ρ_0 at time t_0 through

$$\rho(t) = U(t,t_0)\,\rho_0\,U^\dagger(t,t_0), \tag{2.21}$$

as stated in *Postulate 2*.

Differentiating this expression with respect to time, using Eq. (2.20), and exploiting the Hermiticity of the Hamiltonian, $H(t) = H^\dagger(t)$, one obtains

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\rho(t) &= \frac{\mathrm{d}}{\mathrm{d}t}U(t,t_0)\,\rho_0\,U^\dagger(t,t_0) + U(t,t_0)\,\rho_0\,\frac{\mathrm{d}}{\mathrm{d}t}U^\dagger(t,t_0)\\ &= -iH(t)\rho(t) + i\rho(t)H(t)\\ &= -i\,[H(t),\rho(t)],\end{aligned} \tag{2.22}$$

where $[\cdot,\cdot]$ denotes the commutator between two operators.

This equation is known as the *von Neumann equation*, and governs the unitary dynamics of closed quantum systems in the Schrödinger picture. A detailed discussion can be found in Ref. [55].

Solution. To properly describe the dynamics of density operators, it is necessary to determine the corresponding time-evolution operator associated with the Hamiltonian governing the system. Two relevant cases can be distinguished: (i) evolution generated by a time-independent Hamiltonian, and (ii) evolution generated by a time-dependent Hamiltonian.

In both cases, the system may remain closed and the dynamics is unitary. A time-dependent Hamiltonian does not necessarily imply an open system, but may arise, for instance, from externally controlled fields or explicitly time-modulated parameters acting on an otherwise isolated quantum system.

For the case of a time-independent Hamiltonian H, the solution of Eq. (2.20) can be obtained in closed form, and the time-evolution operator reads

$$U(t,t_0) = \exp[-iH(t-t_0)]\,. \tag{2.23}$$

In contrast, when the Hamiltonian depends explicitly on time, $H(t)$, the time-evolution operator is formally given by

$$U(t,t_0) = \mathscr{T}\exp\left[-i\int_{t_0}^{t}\mathrm{d}\tau\,H(\tau)\right], \tag{2.24}$$

where $\mathscr{T}$ denotes the time-ordering operator, which arranges products of operators such that terms evaluated at later times appear to the left. The presence of this operator is essential because, in general, a time-dependent Hamiltonian does not commute with itself at different times, i.e., $[H(t_1), H(t_2)] \neq 0$ for $t_1 \neq t_2$. As a consequence, different temporal orderings of the operators lead to different physical results.

For this reason, the exact time-evolution operator is typically expressed through systematic approximation schemes or infinite series representations, such as the Dyson series [56], the Magnus expansion [22], or the Trotter product formula [57].

Dyson series. Equation (2.20) can be rewritten as the integral equation

$$U(t,t_0) = \mathbb{I} - i\int_{t_0}^{t} H(\tau)\,U(\tau,t_0)\,\mathrm{d}\tau. \tag{2.25}$$

A formal solution is obtained by iteratively substituting the operator $U(\tau,t_0)$ inside the integral. This procedure generates a perturbative expansion given by

$$\begin{aligned} U(t,t_0) = \mathbb{I} - i\int_{t_0}^{t} \mathrm{d}\tau\, H(\tau) + (-i)^2 \int_{t_0}^{t} \mathrm{d}\tau \int_{t_0}^{\tau} \mathrm{d}\tau'\, H(\tau)H(\tau') \\ + \cdots + (-i)^n \int_{t_0}^{t} \mathrm{d}\tau \int_{t_0}^{\tau} \mathrm{d}\tau' \cdots \int_{t_0}^{\tau^{(n-1)}} \mathrm{d}\tau^{(n)}\, H(\tau)H(\tau')\cdots H(\tau^{(n)}) + \ldots \end{aligned} \tag{2.26}$$

This expansion is known as the *Dyson series* and provides a systematic perturbative construction of the time-evolution operator for time-dependent Hamiltonians. Despite its formal exactness, the Dyson series presents two major practical limitations. First, in any numerical implementation the series must be truncated at a finite order, which generally leads to an approximate evolution operator that is no longer unitary. As a consequence, probability conservation is violated, requiring artificial re-normalization procedures and potentially driving the quantum states outside the space of physical states, *i.e.*, violating the conditions imposed by *Postulate 1* during time evolution.

Second, the explicit evaluation of the nested time-ordered integrals becomes computationally expensive, even at relatively low orders of truncation. This rapidly increasing computational cost severely limits

the applicability of the Dyson series in large-scale or long-time numerical simulations, motivating the development of alternative approaches that preserve unitarity by construction.

Magnus expansion. Alternatively, the solution of Eq. (2.20) can be expressed using the Magnus expansion, which provides a representation of the time-evolution operator in exponential form,

$$U(t,t_0) = \exp\big[\Omega(t,t_0)\big], \tag{2.27}$$

where the operator $\Omega(t,t_0)$ is given as an infinite series,

$$\Omega(t,t_0) = \sum_{k=1}^{\infty} \Omega_k(t,t_0). \tag{2.28}$$

The first terms of the Magnus expansion are given by

$$\Omega_1(t,t_0) = -i\int_{t_0}^{t} \mathrm{d}\tau\, H(\tau), \tag{2.29}$$

$$\Omega_2(t,t_0) = -\frac{1}{2}\int_{t_0}^{t} \mathrm{d}\tau \int_{t_0}^{\tau} \mathrm{d}\tau' \,\big[H(\tau),H(\tau')\big], \tag{2.30}$$

$$\begin{aligned}\Omega_3(t,t_0) = \frac{i}{6}\int_{t_0}^{t} \mathrm{d}\tau \int_{t_0}^{\tau} \mathrm{d}\tau' \int_{t_0}^{\tau'} \mathrm{d}\tau'' \Big(&[H(\tau),[H(\tau'),H(\tau'')]] \\ &+ [H(\tau''),[H(\tau'),H(\tau)]]\Big),\end{aligned} \tag{2.31}$$

while higher-order terms involve increasingly nested commutators of the Hamiltonian evaluated at different times. The Magnus series converges, and the exponential representation is therefore valid, provided that $\int_{t_0}^{t} \|H(\tau)\|_2 \,\mathrm{d}\tau < \pi$, where $\|\cdot\|_2$ denotes the spectral (operator) norm [22]; this convergence condition underlies the normalization of the Hamiltonian introduced in the applications of Chapter 4.

A key advantage of the Magnus expansion is that, at any finite truncation order, the time-evolution operator is explicitly written as the exponential of an anti-Hermitian operator. Consequently, since the Hamiltonian $H(t)$ is Hermitian, the truncated evolution operator remains unitary by construction.

From a numerical standpoint, the Magnus expansion offers improved stability and physical consistency, as it preserves norm and probability

conservation without requiring re-normalization procedures. Nevertheless, its practical applicability is limited by the rapid growth in the number and complexity of nested commutators appearing at higher orders, which can significantly increase the computational cost. Despite this limitation, low-order Magnus expansions often provide accurate and reliable approximations, making them well suited for controlled simulations of time-dependent quantum dynamics.

If the Hamiltonian $H(t)$ commutes with itself at different times, all higher-order Magnus terms vanish and the expansion reduces to the first-order contribution $\Omega_1(t,t_0)$. The necessity of including higher-order terms is therefore tied to the non-commutativity of the Hamiltonian at different times, and hence to the complexity of the time-dependent dynamics.

Trotter formula. An alternative approximation to the time-evolution operator for time-dependent Hamiltonians is provided by the Trotter product formula. This approach is particularly useful when the Hamiltonian can be written as a sum of simpler (generally non-commuting) contributions,

$$H(t) = \sum_k H_k(t), \tag{2.32}$$

for which the individual exponential operators can be efficiently evaluated.

To construct the time-evolution operator from an initial time t_0 to a final time t, the time interval is discretized into N subintervals of equal length $\Delta t = (t - t_0)/N$, such that

$$t_n = t_0 + n\Delta t, \qquad n = 0, 1, \ldots, N.$$

Over each small interval $[t_n, t_{n+1}]$, the Hamiltonian is assumed to be approximately constant and equal to its value at t_n. Under this assumption, the short-time evolution operator can be approximated as

$$U(t_{n+1}, t_n) \approx \exp[-iH(t_n)\,\Delta t]\,. \tag{2.33}$$

If the Hamiltonian is further decomposed into a sum of terms $H_k(t_n)$, the exponential of the total Hamiltonian can be approximated using the

first-order Trotter product formula,

$$\exp[-iH(t_n)\,\Delta t] \approx \prod_k \exp[-iH_k(t_n)\,\Delta t] + \mathscr{O}(\Delta t^2), \tag{2.34}$$

where the error arises from the non-commutativity of the operators $H_k(t_n)$.

By composing the evolution over all time steps, the full time-evolution operator is approximated as

$$U(t,t_0) = \lim_{N\to\infty} \prod_{n=0}^{N-1} \left[\prod_k \exp(-iH_k(t_n)\,\Delta t) \right], \tag{2.35}$$

where the product over n is time-ordered, with earlier times appearing to the right. In the limit $\Delta t \to 0$ (or equivalently $N \to \infty$), this construction converges to the exact time-evolution operator.

The Trotter formula provides a simple and physically transparent approximation scheme for the time-evolution operator. At each time step, unitarity is preserved by construction, since the approximation is expressed as a product of unitary operators. In the present work, the first-order Trotter approximation is sufficient to accurately capture the relevant dynamics.

As in the Magnus expansion, the Trotter formula guarantees unitarity. However, in contrast to both the Dyson series and the Magnus expansion, the Trotter approach does not require the explicit evaluation of time-ordered integrals or nested commutators. This feature makes it particularly well suited for numerical implementations, as the computation of the time-evolution operator reduces to evaluating matrix exponentials of simpler Hamiltonian terms at each discretized time step.

2.5 Pauli basis and operator representations

Throughout this work, the physical systems of interest are two-level quantum systems, or composite systems formed by subsystems with two accessible energy levels. The quantum states considered therefore describe such systems, commonly referred to as *qubits*. Although quantum information theory is not restricted to two-level systems and can be naturally extended to n-level systems, known as *qudits*, the focus of this work is exclusively on qubit-based systems.

The dynamics and structure of qubits are conveniently described

within a mathematical framework associated with the special unitary group $SU(2)$ and its Lie algebra $\mathfrak{su}(2)$, which captures the symmetry properties of isolated two-level quantum systems. Within this framework, a set of four operators plays a central role: the identity operator and the three Pauli matrices. Together, these operators form a complete basis for the vector space of linear operators acting on a single-qubit Hilbert space. As a result, any single-qubit density operator, Hamiltonian, or unitary evolution operator can be expressed as a linear combination of these matrices. In what follows, this operator basis will be referred to as the *Pauli basis*, defined by

$$\sigma_0 \equiv \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \tag{2.36}$$

$$\sigma_1 \equiv \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \tag{2.37}$$

$$\sigma_2 \equiv \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \tag{2.38}$$

$$\sigma_3 \equiv \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}. \tag{2.39}$$

These matrices satisfy several important algebraic relations,

$$[\sigma_i, \sigma_j] = 2i\,\varepsilon_{ijk}\,\sigma_k, \tag{2.40}$$
$$\{\sigma_i, \sigma_j\} = 2\,\delta_{ij}\,\mathbb{I}, \tag{2.41}$$
$$\sigma_i\sigma_j = \delta_{ij}\,\mathbb{I} + i\,\varepsilon_{ijk}\,\sigma_k, \tag{2.42}$$

where δ_{ij} denotes the Kronecker delta, $\{\cdot,\cdot\}$ denotes the anticommutator and $i, j, k \in \{1, 2, 3\}$.

Pauli vectors. It is useful to introduce a correspondence between vectors in $\mathbb{R}^3$ and the space of traceless Hermitian 2×2 matrices. This correspondence is naturally established through the Pauli matrices, which span the traceless Hermitian 2×2 matrices, i.e. the generators of $SU(2)$ (the elements of $\mathfrak{su}(2)$ up to a factor of i). Within this framework, any single-qubit density operator can be parametrized in terms of a real

three-dimensional vector, known as the Bloch vector.

Explicitly, a one-qubit density operator can be written as

$$\rho = \frac{1}{2}(\mathbb{I} + \vec{\mathbf{r}} \cdot \boldsymbol{\sigma}), \tag{2.43}$$

where $\vec{\mathbf{r}} \in \mathbb{R}^3$ is the Bloch vector and $\boldsymbol{\sigma}$ denotes the Pauli vector. The Pauli vector is defined as the operator-valued vector

$$\boldsymbol{\sigma} = \sigma_1 \hat{\mathbf{x}}_1 + \sigma_2 \hat{\mathbf{x}}_2 + \sigma_3 \hat{\mathbf{x}}_3, \tag{2.44}$$

with $\hat{\mathbf{x}}_1$, $\hat{\mathbf{x}}_2$, and $\hat{\mathbf{x}}_3$ being the canonical unit vectors of $\mathbb{R}^3$.

The components of the Bloch vector are given by the expectation values of the Pauli operators,

$$r_i = \mathrm{Tr}(\rho\, \sigma_i), \qquad i \in \{1,2,3\}, \tag{2.45}$$

which ensures that $\vec{\mathbf{r}}$ is real-valued due to the Hermiticity of ρ and the Pauli matrices. The positivity and unit-trace conditions imposed on density operators restrict the norm of the Bloch vector to satisfy

$$|\vec{\mathbf{r}}| \leq 1. \tag{2.46}$$

Geometrically, this condition defines a unit-radius sphere in $\mathbb{R}^3$, where each point inside the sphere is uniquely associated with a valid density operator. This geometric representation is known as the Bloch sphere. Pure states correspond to vectors lying on the surface of the sphere, whereas mixed states are represented by vectors strictly inside it.

This parametrization provides a geometrically intuitive description of single-qubit quantum states. In particular, unitary evolutions generated by Hamiltonians proportional to the Pauli vector correspond to rotations of the Bloch vector in $\mathbb{R}^3$. Consequently, the Pauli vector formalism establishes a direct link between the algebraic structure of $\mathfrak{su}(2)$ and the geometric representation of qubit states.

Dynamics in the Pauli basis. The Pauli vector formalism provides a natural representation for single-qubit Hamiltonians. Any Hermitian operator acting on a two-dimensional Hilbert space can be expanded in the Pauli basis. In particular, a general time-dependent single-qubit

Hamiltonian can be written as

$$H(t) = \vec{\mathbf{h}}(t) \cdot \boldsymbol{\sigma}, \tag{2.47}$$

where $\vec{\mathbf{h}}(t) \in \mathbb{R}^3$ is a real vector characterizing the Hamiltonian. In this representation, the full time dependence of the Hamiltonian is entirely encoded in the vector $\vec{\mathbf{h}}(t)$.

Consider the simpler case in which the Hamiltonian is time-independent. In this case, the time-evolution operator is given by

$$U(t,t_0) = \exp\left[-i(t-t_0)H\right] = \exp\left[-i(t-t_0)\vec{\mathbf{h}} \cdot \boldsymbol{\sigma}\right], \tag{2.48}$$

where the exponent satisfies

$$\left(\vec{\mathbf{h}} \cdot \boldsymbol{\sigma}\right)^2 = h^2 \mathbb{I}, \tag{2.49}$$

where $h = |\vec{\mathbf{h}}|$.

Using the series definition of the exponential,

$$e^{-iH(t-t_0)} = \sum_{n=0}^{\infty} \frac{\left[-i(t-t_0)\right]^n}{n!} H^n, \qquad H = \vec{\mathbf{h}} \cdot \boldsymbol{\sigma}, \tag{2.50}$$

the even-order terms yield

$$\begin{aligned} \sum_{n=0}^{\infty} \frac{\left[-i(t-t_0)\right]^{2n}}{(2n)!} H^{2n} &= \sum_{n=0}^{\infty} \frac{(-1)^n \left[h\,(t-t_0)\right]^{2n}}{(2n)!} \mathbb{I} \\ &= \cos\left[h\,(t-t_0)\right] \mathbb{I}, \end{aligned} \tag{2.51}$$

while the odd-order terms give

$$\begin{aligned} \sum_{n=0}^{\infty} \frac{\left[-i(t-t_0)\right]^{2n+1}}{(2n+1)!} H^{2n+1} &= -i \sum_{n=0}^{\infty} \frac{(-1)^n \left[h\,(t-t_0)\right]^{2n+1}}{(2n+1)!} \hat{\mathbf{h}} \cdot \boldsymbol{\sigma} \\ &= -i \sin\left(h\,(t-t_0)\right) \hat{\mathbf{h}} \cdot \boldsymbol{\sigma}, \end{aligned} \tag{2.52}$$

where $\hat{\mathbf{h}} = \vec{\mathbf{h}}/h$.

Substituting Eqs. (2.51) and (2.52) into Eq. (2.50), the time-evolution

operator can be written in closed form as

$$U(t,t_0) = \cos\left(h\left(t-t_0\right)\right)\mathbb{I} - i\sin\left(h\left(t-t_0\right)\right)\hat{\mathbf{h}}\cdot\boldsymbol{\sigma}. \tag{2.53}$$

This expression shows that the unitary evolution induces a rigid rotation of the Bloch vector associated with a quantum state around the axis defined by the unit vector $\hat{\mathbf{h}}$, with angular frequency $2\left|\vec{\mathbf{h}}\right|$. This establishes a direct correspondence between unitary dynamics in Hilbert space and rotations in $\mathbb{R}^3$, further highlighting the usefulness of the Pauli vector formalism as a bridge between algebraic and geometric descriptions of single-qubit dynamics.

Extension to multi-qubit systems. The Pauli vector formalism can be naturally extended to composite quantum systems composed of n qubits by following *Postulate 4*. In this case, operators acting on the composite Hilbert space can be conveniently represented using tensor products of single-qubit Pauli operators. Specifically, an operator basis for an n-qubit system can be constructed as the set of all n-fold tensor products

$$\sigma_{\boldsymbol{\mu}} = \sigma_{\mu_1}\otimes\sigma_{\mu_2}\otimes\cdots\otimes\sigma_{\mu_n}, \qquad \mu_k \in \{0,1,2,3\}. \tag{2.54}$$

These operators are commonly referred to as *Pauli strings*. The Pauli strings constitute an orthogonal basis with respect to the Hilbert–Schmidt inner product,

$$\mathrm{Tr}\big(\sigma_{\boldsymbol{\mu}}\,\sigma_{\boldsymbol{\nu}}\big) = 2^n\,\delta_{\boldsymbol{\mu},\boldsymbol{\nu}}, \tag{2.55}$$

where $\delta_{\boldsymbol{\mu},\boldsymbol{\nu}}$ denotes the Kronecker delta over multi-indices, defined as

$$\delta_{\boldsymbol{\mu},\boldsymbol{\nu}} = \prod_{k=1}^{n}\delta_{\mu_k\nu_k}. \tag{2.56}$$

Therefore any operator O acting on an n-qubit system can be expanded as

$$O = \sum_{\boldsymbol{\mu}} c_{\boldsymbol{\mu}}\,\sigma_{\boldsymbol{\mu}}, \qquad c_{\boldsymbol{\mu}} = \frac{1}{2^n}\,\mathrm{Tr}\big(O\,\sigma_{\boldsymbol{\mu}}\big). \tag{2.57}$$

In particular, an arbitrary n-qubit density operator admits the expan-

sion

$$\rho = \frac{1}{2^n} \sum_{\boldsymbol{\mu}} r_{\boldsymbol{\mu}} \, \sigma_{\boldsymbol{\mu}}, \tag{2.58}$$

where the coefficients $r_{\boldsymbol{\mu}} = \mathrm{Tr}\big(\rho \, \sigma_{\boldsymbol{\mu}}\big)$ are real, by the Hermiticity of ρ and of the Pauli strings, and generalize the notion of the Bloch vector to multi-qubit systems. Similarly, a general n-qubit Hamiltonian can be written as

$$H(t) = \sum_{\boldsymbol{\mu}} h_{\boldsymbol{\mu}}(t) \, \sigma_{\boldsymbol{\mu}}, \tag{2.59}$$

where the time dependence of the Hamiltonian is entirely encoded in the real coefficients $h_{\boldsymbol{\mu}}(t)$.

This representation provides a unified and scalable framework for describing the structure and dynamics of multi-qubit systems, and serves as a natural language for many-body Hamiltonians, quantum control protocols, and machine-learning-based approaches to quantum dynamics.

2.6 Fidelity

Fidelity between density matrices. To quantitatively assess the similarity between quantum states and quantum operations, it is necessary to introduce appropriate fidelity measures. These quantities play a central role in the validation of quantum dynamics, the characterization of control protocols, and the evaluation of machine-learning-based models for quantum systems.

The fidelity between two quantum states ρ and σ is defined as [58]

$$F(\rho, \sigma) = \left[\mathrm{Tr}\left(\sqrt{\sqrt{\rho} \, \sigma \, \sqrt{\rho}} \right) \right]^2. \tag{2.60}$$

This definition admits simplified expressions in relevant special cases. If one of the states is pure, $\rho = |\psi\rangle \langle\psi|$, the fidelity reduces to

$$F(|\psi\rangle, \sigma) = \langle\psi| \, \sigma \, |\psi\rangle, \tag{2.61}$$

while for two pure states, $\rho = |\psi\rangle \langle\psi|$ and $\sigma = |\phi\rangle \langle\phi|$, it further reduces to

$$F(|\psi\rangle, |\phi\rangle) = |\langle\psi|\phi\rangle|^2. \tag{2.62}$$

For any pair of density operators ρ and σ, the fidelity satisfies the following properties:

- **Bounds:** $0 \leq F(\rho,\sigma) \leq 1$.
- **Normalization:** $F(\rho,\sigma) = 1 \quad$ if and only if $\quad \rho = \sigma$.
- **Symmetry:** $F(\rho,\sigma) = F(\sigma,\rho)$.
- **Unitary invariance:** For any unitary operator U,

$$F(U\rho U^\dagger, U\sigma U^\dagger) = F(\rho,\sigma). \tag{2.63}$$

- **Joint concavity of the square-root fidelity:** For any sets $\{\rho_i\}$, $\{\sigma_i\}$ and probabilities $\{p_i\}$,

$$\sqrt{F}\left(\sum_i p_i\rho_i, \sum_i p_i\sigma_i\right) \geq \sum_i p_i\sqrt{F(\rho_i,\sigma_i)}. \tag{2.64}$$

This expression does not define a metric on the space of density operators, since it does not satisfy the triangle inequality. It nevertheless remains a useful quantity for comparing density matrices.

Fidelity between operations. To compare quantum operations, it is convenient to adopt the framework of quantum channels, where a quantum operation is described as a linear map $\mathscr{E}$ acting on density operators,

$$\rho' = \mathscr{E}(\rho). \tag{2.65}$$

The Choi–Jamiołkowski representation provides a convenient way to compare quantum channels by establishing a one-to-one correspondence between completely positive trace-preserving maps and positive semidefinite operators acting on an extended Hilbert space [59].

Consider a unitary operator U acting on a d-dimensional Hilbert space. The associated unitary quantum channel $\mathscr{U}$ is defined as

$$\mathscr{U}(\rho) = U\rho U^\dagger, \tag{2.66}$$

and its corresponding Choi matrix is given by

$$J_U = (\mathscr{U} \otimes \mathbb{I})(|\Phi\rangle\langle\Phi|), \tag{2.67}$$

where

$$|\Phi\rangle = \frac{1}{\sqrt{d}} \sum_{i=1}^{d} |i\rangle \otimes |i\rangle \tag{2.68}$$

is a maximally entangled reference state.

Since the Choi–Jamiołkowski isomorphism maps quantum operations to density matrices, the fidelity defined in Eq. (2.60) can be directly used to compare quantum channels. Consider two unitary operators U and V, with associated Choi matrices

$$J_U = (\mathscr{U} \otimes \mathbb{I})(|\Phi\rangle \langle\Phi|), \tag{2.69}$$
$$J_V = (\mathscr{V} \otimes \mathbb{I})(|\Phi\rangle \langle\Phi|). \tag{2.70}$$

Unitary channels correspond to pure Choi states, which can be written as $J_U = |J_U\rangle \langle J_U|$ and $J_V = |J_V\rangle \langle J_V|$. For the comparison of unitary channels, the *gate fidelity* is adopted, defined simply as the fidelity of Eq. (2.60) evaluated on these pure Choi states,

$$F(J_U, J_V) = |\langle J_U | J_V \rangle|^2. \tag{2.71}$$

Using Eq. (2.67), this expression becomes

$$F(J_U, J_V) = \left| \langle\Phi| \left(U^\dagger V \otimes \mathbb{I}\right) |\Phi\rangle \right|^2 = \left| \frac{1}{d} \operatorname{Tr}\left(U^\dagger V\right) \right|^2. \tag{2.72}$$

This quantity, which corresponds to the transition probability between the pure Choi states associated with U and V, is commonly referred to as the *gate fidelity*. In particular, if $U = V$, then $\operatorname{Tr}\left(U^\dagger V\right) = d$, yielding $F(J_U, J_V) = 1$, which corresponds to maximal fidelity. The fidelity between Choi matrices therefore provides a consistent and operationally meaningful measure for comparing unitary quantum operations.

2.7 Summary

In this chapter, the fundamental concepts of quantum information theory required throughout this thesis have been introduced. The formulation of quantum mechanics based on density operators was adopted as the primary framework, as it provides a general and operationally meaningful description of both pure and mixed quantum states, as well as a

natural language for open and composite systems.

Starting from the postulates of quantum mechanics, the structural properties of density operators were discussed, including their spectral decomposition, purity, ensemble representations, and operator inner products. The formalism was then extended to composite systems through the tensor-product structure of Hilbert spaces and the partial trace operation, leading to the definition of reduced density operators and the introduction of entanglement as a genuinely quantum form of correlation.

The role of entanglement was further emphasized by discussing the quantum marginal problem, which highlights the fundamental limitations associated with reconstructing global quantum states from local information. This problem provides important conceptual insight into the structure of composite quantum systems and motivates the need for alternative approaches when dealing with incomplete or indirect descriptions of quantum dynamics.

The chapter also addressed the unitary dynamics of closed quantum systems through the von Neumann equation, together with several analytical and numerical approaches for solving time-dependent dynamics, including the Dyson series, the Magnus expansion, and the Trotter product formula. These methods establish the theoretical basis for the simulation and approximation of quantum evolution operators.

Finally, the Pauli basis was introduced as a convenient operator representation for qubit systems, enabling a compact and scalable description of quantum states, Hamiltonians, and dynamical generators. Within this framework, the concept of fidelity was presented as a quantitative measure of similarity between quantum states and quantum operations, including its extension to unitary channels via the Choi–Jamiołkowski representation.

Together, the tools and concepts developed in this chapter provide the mathematical and conceptual foundation required for the learning-based modeling of quantum dynamics explored in the subsequent chapters.

Chapter 3
DEEP LEARNING

This chapter provides all the background required to understand, reproduce, and critically analyze the deep learning tools employed throughout this thesis. Owing to the versatility of deep learning models and their capacity to approximate complex, high-dimensional mappings, these methods offer a powerful framework for addressing challenging problems in Physics and Mathematics.

By introducing the essential concepts needed to formulate, train, test, and analyze deep learning models, this chapter establishes the foundations for constructing architectures tailored to specific physical constraints and learning objectives. In particular, such models enable the treatment of problems that are otherwise computationally prohibitive using conventional numerical methods, including the quantum marginal problem discussed in the previous chapter and the simulation of complex quantum dynamics with reduced computational resources.

Given the broad scope of Deep Learning as a research field, the discussion in this chapter is intentionally limited to the concepts and techniques that are directly relevant to the developments presented in this thesis. Throughout this chapter, the references [3, 60] are adopted as the primary sources for the theoretical and methodological aspects of deep learning.

3.1 Motivation and scope

Deep learning has demonstrated notable versatility across a wide range of disciplines, spanning from large-scale commercial applications to advanced scientific research. In many contexts, deep learning models have outperformed traditional numerical and algorithmic approaches when dealing with high-dimensional, nonlinear, or poorly structured problems. This capability has enabled the development of practical solutions to tasks that, prior to the emergence of modern deep learning techniques, were widely regarded as computationally infeasible or even impossible.

As mentioned in the Introduction, deep learning has been successfully applied to a variety of problems, including image and signal processing, natural language modeling, and the approximation of complex dynamical systems. Beyond their empirical success, these models provide a flexible mathematical framework for learning representations and mappings directly from data, without requiring explicit handcrafted features or closed-form analytical models.

Despite their empirical success, deep learning models are often informally referred to as *black-box* models. This characterization reflects the perceived opacity of their internal mechanisms rather than a lack of underlying mathematical structure. However, several authors argue that this description is imprecise, and that deep learning models are more appropriately described as *grey-box* models. Although such models may involve a large number of parameters, their internal operations remain partially traceable and analyzable, which contradicts a strictly black-box interpretation.

This chapter clarifies how these models operate internally by introducing the key principles governing their architecture, training, and optimization. This perspective is particularly relevant in physics-based applications, where interpretability, consistency with known physical laws, and controlled generalization play a central role.

3.2 Machine Learning models

Since Deep Learning is a subfield of Machine Learning, many of the concepts employed in this area originate from the broader Machine Learning framework. In both Machine Learning and Deep Learning, the central objective is to construct algorithms that are able to learn from data rather than being explicitly programmed through fixed rules.

This naturally leads to a fundamental question: what does it mean for a model to *learn*? A widely accepted and operational definition, originally formulated by Mitchell and reproduced in Ref. [3], states that

> A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

This definition highlights three essential components of any learn-

ing system: the task T to be performed, the experience E from which the system learns, and the performance measure P used to evaluate its success. Together, these elements provide a precise and practical framework for formulating learning problems in a mathematically and computationally well-defined manner.

The following sections analyze these components in the context of deep learning models and introduce the key elements required to understand neural network architectures, training procedures, and evaluation strategies. This establishes the foundation necessary to construct and analyze learning-based models capable of addressing the complex physical and mathematical problems considered in this thesis.

The task, T. In Machine Learning, the task T specifies the type of problem that the algorithm is designed to solve and determines how input data are processed to produce meaningful outputs. Data are typically represented as collections of features extracted or measured from a physical, mathematical, or computational system, and the task defines the mapping that the model is expected to learn from these features.

A wide variety of tasks can be addressed within this framework. Common examples include classification, where the objective is to assign discrete labels to inputs; regression, which aims to predict continuous-valued outputs; transcription and representation learning, where data are mapped into alternative representations; anomaly detection, which focuses on identifying atypical patterns; interpolation and denoising tasks; and probability density estimation. The specific choice of the task T is dictated by the nature of the problem under study and by the type of information one aims to extract from the data.

The experience, E. The experience E refers to the data from which the learning algorithm acquires information. A useful distinction arises when categorizing Machine Learning approaches according to the type of experience provided during training.

In *supervised* learning, the experience consists of labeled data pairs, typically of the form $(\mathbf{x}, \mathbf{y})$, where $\mathbf{x}$ denotes the input features and $\mathbf{y}$ represents the desired output. The learning process aims to infer a functional relationship that maps inputs to outputs by minimizing a discrepancy between model predictions and known targets.

In contrast, *unsupervised* learning relies on unlabeled data, where

only the input samples **x** are available. In this case, the objective is to uncover hidden structures, correlations, or representations intrinsic to the data itself, without explicit target values. Both paradigms play an important role in Deep Learning, depending on the availability of data and the nature of the task being addressed.

The performance measure, *P*. The performance measure P quantifies how well a learning algorithm performs the assigned task and plays a central role during the training stage. Since learning-based models aim to approximate an unknown mapping from data, discrepancies between model predictions and observed values naturally arise. The purpose of the performance measure is to provide a scalar quantity that evaluates these discrepancies and allows different candidate models to be compared.

There exists considerable flexibility in the choice of performance measures, depending on the task under consideration. In regression problems, a common choice is to evaluate the deviation between predicted and measured values through an averaged error criterion.

To illustrate this idea, consider the simple example of *linear regression*. Given a set of measured data points $\{(x_i, y_i)\}_{i=1}^{N}$, the task consists of finding a linear function

$$y(x) = mx + n, \tag{3.1}$$

that best describes the relationship between the variables x and y. Although two exact data points are sufficient to determine a unique linear function in an ideal noiseless setting, real measurements are typically affected by noise. As a consequence, infinitely many linear functions may be compatible with the data within experimental uncertainty.

To compare different candidate functions, a performance function can be introduced. For instance, one may define

$$\mathscr{L}(m,n) = \frac{1}{N}\sum_{i=1}^{N}(mx_i + n - y_i)^2, \tag{3.2}$$

which quantifies the average squared deviation between the model predictions and the observed data.

In this example, the three elements of learning are clearly identified: the task T is to approximate the functional relationship between

two variables, the experience E is given by the measured data points (x_i, y_i), and the performance measure P provides a quantitative criterion to assess how well the model performs the task.

3.3 Artificial neural networks

A central pillar of this thesis is the use of artificial neural networks (ANNs). In this work, they are employed to construct models (i.e., computer programs) designed to accomplish the tasks T previously introduced. An artificial neural network is built from basic computational units commonly referred to as neurons or perceptrons, which constitute the fundamental elements of these architectures.

The terminology neuron originates from the fact that neural networks were originally inspired by the interconnected structure of biological neural systems, such as those found in the human brain. However, despite this inspiration, artificial neural networks do not operate according to the same principles as biological neural systems and exhibit substantial differences in their underlying mechanisms [61].

Nevertheless, the computational behavior of artificial perceptrons is sufficient to construct robust and versatile models that have proven effective across a wide range of tasks. In this thesis, some of these tasks that can be successfully addressed using artificial neural networks are explored and analyzed.

Artificial neurons The basic structure of an artificial neuron can be conveniently described using linear algebra. Let $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$ denote the input vector of the neuron, and $\boldsymbol{\omega} = (\omega_1, \ldots, \omega_n)$ the corresponding vector of trainable weights. The neuron computes an affine transformation of the input followed by a nonlinear activation function f, yielding the output

$$y = f(\boldsymbol{\omega} \cdot \mathbf{x} + b). \tag{3.3}$$

The bias parameter b allows the activation function to be shifted independently of the input, thereby increasing the expressive capacity of the neuron. For a fixed activation function, the response of the neuron is entirely determined by the values of the weights and the bias, which are adjusted during training in order to satisfy the requirements of the target task.

Forward propagation Once the functioning of a single perceptron is understood, it is natural to extend the construction to a collection of perceptrons arranged in layers, forming a neural network capable of processing data. Consider a first layer composed of n_1 artificial neurons. The output of the i-th neuron in this layer is given by

$$a_i^{(1)} = f\left(\boldsymbol{\omega}_i^{(1)} \cdot \mathbf{x} + b_i^{(1)}\right), \qquad i = 1, \ldots, n_1, \tag{3.4}$$

where $\boldsymbol{\omega}_i^{(1)}$ and $b_i^{(1)}$ denote the weights and bias associated with the i-th neuron of the layer, respectively, and f is the activation function applied element-wise. Here the superscript labels the layer index, so that $\mathbf{a}^{(1)} = (a_1^{(1)}, \ldots, a_{n_1}^{(1)})$ collects the outputs of the first layer.

Subsequently, additional layers can be stacked on top of the first one. For instance, if the input to the second layer is given by the output vector of the first layer, $\mathbf{a}^{(1)}$, then the output of the j-th neuron in the second layer is

$$a_j^{(2)} = g\left(\boldsymbol{\omega}_j^{(2)} \cdot \mathbf{a}^{(1)} + b_j^{(2)}\right), \tag{3.5}$$

where $\boldsymbol{\omega}_j^{(2)}$ and $b_j^{(2)}$ are the weights and bias of the corresponding neuron in the second layer. The activation function g may differ from f, although in practice it is common, but not mandatory, to use the same activation across multiple layers.

This construction can be generalized to a neural network composed of L layers, where each layer introduces additional weights and biases, increasing the number of free parameters available for optimization during training. These parameters are adjusted in order to approximate a target mapping associated with a given task. In this context, the matrix representation of the neural network can be written in a compact form. Let $\mathbf{a}^{(0)} \equiv \mathbf{x}$ denote the input vector, and consider the l-th layer characterized by a weight matrix $\mathbf{W}^{(l)} \in \mathbb{R}^{n_l \times n_{l-1}}$, a bias vector $\mathbf{b}^{(l)} \in \mathbb{R}^{n_l}$, and an activation function $\sigma^{(l)}(\cdot)$. The forward propagation through the network is then given by

$$\mathbf{a}^{(l)} = \sigma^{(l)}\left(\mathbf{W}^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}\right), \tag{3.6}$$

where the activation function is applied element-wise. The output of the network corresponds to $\mathbf{y} = \mathbf{a}^{(L)}$, with L denoting the total number of layers. A similar description can be found in Ref. [62], where the authors

develop a Python-based framework to solve differential equations using this class of neural networks. This application will be discussed later in this chapter.

Models of this type are referred to as *feedforward* (or *forward*) neural networks, since the output is obtained through a forward propagation of information from the input layer, across successive hidden layers, up to the final output layer. In particular, there exist multiple ways to connect the internal neurons within a neural network. The configuration presented in Figure 3.1 corresponds to the so-called *fully connected* architecture, in which each perceptron in a given layer is connected to every perceptron in the preceding and subsequent layers.

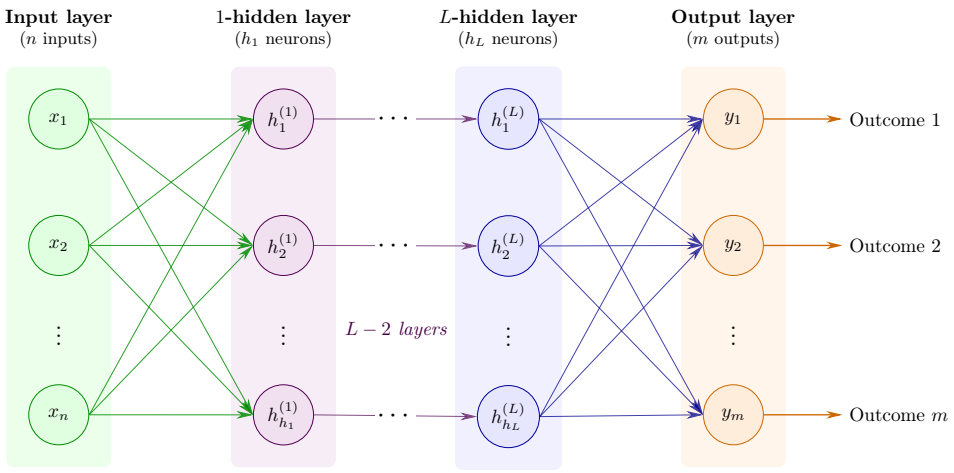


Figure 3.1: Schematic representation of a deep feedforward neural network using a combination of fully-connected layers

In general, while the structure of the input and output layers is strongly dictated by the specific task for which the network is designed, the internal layers, commonly referred to as *hidden layers*, may vary in their connectivity patterns and may incorporate additional mathematical regularization mechanisms, such as batch normalization or dropout. These techniques are task-dependent and implementation-specific; therefore, they are not considered in the present work. The specific choice of layer organization, connectivity, and associated transformations is referred to as the *network architecture*.

Convolutional layers The fully connected architecture described previously provides a general framework for constructing neural networks; however, it does not explicitly exploit any structure present in the input

data. In many applications, particularly those involving spatially distributed data such as images, grids, or discretized physical fields, the input exhibits strong local correlations. In such cases, fully connected layers become parameter-inefficient and may fail to capture relevant inductive biases due to the large number of free parameters (weights and biases). Convolutional layers address this limitation by introducing structured weight sharing through the convolution operation. Instead of learning an independent weight for each input–output connection, a convolutional layer applies a set of local kernels across the input domain, enforcing translational equivariance and significantly reducing the number of trainable parameters.

A defining characteristic of convolutional layers is the use of shared weights. In contrast to fully connected layers, where each connection between neurons is associated with an independent weight, convolutional layers employ the same kernel parameters across all spatial locations of the input. As a consequence, a single kernel captures repeated local patterns throughout the entire domain, and its parameters are optimized collectively based on their global contribution to the network output.

The convolution operation for continuous functions can be written as

$$s(t) = (x * \omega)(t) = \int x(a)\, \omega(t-a)\, \mathrm{d}a, \tag{3.7}$$

where the symbol $*$ denotes convolution, x represents the input function, and ω is a weight function.

In the context of neural networks, x is interpreted as the *input* of the layer, while the weight function ω is referred to as the *kernel*. The output of the convolutional layer is commonly called the *feature map*. Although convolution is naturally defined for continuous functions, machine learning applications typically employ a discrete formulation, given by

$$s(t) = (x * \omega)(t) = \sum_{a=-\infty}^{\infty} x(a)\, \omega(t-a), \tag{3.8}$$

where a is an integer-valued variable indexing the input domain.

In practical applications, the input is usually a multidimensional array and the kernel is a multidimensional tensor of trainable parameters. For example, considering a two-dimensional input such as an image, the

convolution operation takes the form

$$S(i,j) = (I * K)(i,j) = \sum_{m,n} I(m,n)\, K(i-m, j-n), \tag{3.9}$$

where I denotes the input array, K the kernel, and (i,j) label spatial coordinates.[1]

The forward propagation through a convolutional layer can therefore be expressed as

$$\mathbf{a}^{(l)} = \sigma^{(l)}\left(\mathbf{K}^{(l)} * \mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}\right), \tag{3.10}$$

where $\mathbf{K}^{(l)}$ denotes the convolutional kernel tensor, $\mathbf{b}^{(l)}$ is the bias term, $*$ represents the discrete convolution operator, and $\sigma^{(l)}$ is applied element-wise.

Beyond the kernel itself, the behavior of a convolutional layer is governed by several structural hyperparameters. The *stride* controls the displacement of the kernel between successive applications, with larger strides reducing the spatial resolution of the resulting feature maps. The *padding* operation extends the input domain, commonly by adding zeros, allowing control over output dimensions and mitigating boundary effects. The *dilation* parameter introduces spacing between kernel elements, effectively increasing the receptive field without increasing the number of parameters. Finally, the *groups* parameter partitions the input channels into independent subsets, enabling grouped or depth-wise convolutions and reducing computational complexity.

Figure 3.2 illustrates how a convolutional layer transforms an input matrix into a feature map; its spatial size depends on the padding and stride, and is reduced in the example shown, where no padding is applied, although a suitable padding can also be used to preserve it. A natural comparison between fully connected and convolutional layers can be drawn in terms of the number of internal connections, i.e. the number of free parameters. By restricting each neuron to interact only with a local neighborhood of the input, convolutional layers drastically reduce the number of trainable parameters. Consequently, the weights are

[1] Most deep learning frameworks, including the one used in this work, implement the closely related *cross-correlation* operation $S(i,j) = \sum_{m,n} I(i+m,\, j+n)\, K(m,n)$, which coincides with the convolution up to a reflection of the kernel. Since the kernel entries are learned, this distinction does not affect the expressive power of the layer, and the term *convolution* is retained by convention.

determined by local correlations in the data, yielding a significant reduction in computational cost while preserving the ability to extract relevant features. This same inductive bias is, however, also a limitation. If the relevant structure of the data is not governed by local (near-neighbor) correlations, a model relying solely on convolutional layers cannot be expected to generalize well.

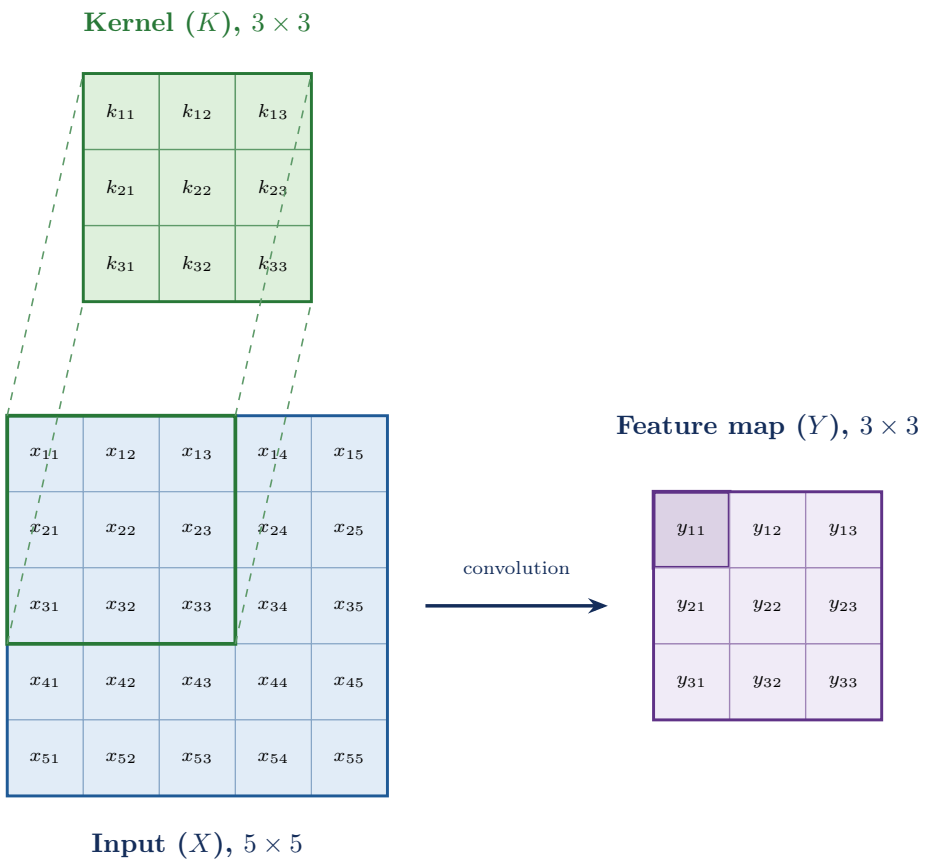


Figure 3.2: Schematic representation of a two-dimensional convolution. A 3×3 kernel is applied across the input array, and each output entry is obtained as the weighted sum of the corresponding local patch, producing a feature map of reduced spatial size.

Transposed convolutional layers Convolutional layers, particularly when combined with strides or pooling operations, progressively reduce the spatial resolution of the feature maps. While this contraction is desirable for tasks such as classification, many problems require the opposite operation, namely recovering or increasing the spatial resolution of a representation. Representative examples include the decoder stage

of autoencoders, generative models, semantic segmentation, and super-resolution. *Transposed convolutional layers* (also called fractionally-strided convolutions) provide a learnable upsampling operation that fulfils this role.

The name originates from the matrix formulation of the convolution. A discrete convolution can be expressed as a linear map $\mathbf{s} = \mathbf{C}\mathbf{x}$, where $\mathbf{x}$ and $\mathbf{s}$ are the flattened input and output, respectively, and $\mathbf{C}$ is a sparse matrix whose entries are determined by the kernel and whose structure encodes the locality and weight sharing of the operation. A transposed convolution applies the transpose of this operator, mapping a lower-dimensional representation back to a higher-dimensional one,

$$\mathbf{x} \longmapsto \mathbf{C}\mathbf{x} \quad \text{(convolution)}, \qquad \mathbf{z} \longmapsto \mathbf{C}^{\top}\mathbf{z} \quad \text{(transposed convolution)}. \tag{3.11}$$

Both operators are built from the same kernel parameters; the transposed convolution is therefore *not* the inverse of the convolution, since it does not recover the original input, but rather its adjoint in terms of connectivity. This is the operation that arises when backpropagating gradients through a convolutional layer. The gradient of the loss with respect to the input of a convolution is obtained by applying the transposed convolution to the gradient with respect to the output, and conversely.

In practice, a transposed convolution with stride s can be implemented as an ordinary, unit-stride convolution applied to a modified input. Two zero-insertion operations are involved. First, the input is *dilated* by inserting $s-1$ zeros between consecutive elements along each spatial dimension, which encodes the upsampling factor. Second, a zero-valued *border* of width $k-p-1$ is added around the dilated map, where k is the kernel size and p the padding of the corresponding direct convolution. A standard convolution with unit stride is then applied to the resulting array. Figure 3.3 shows this procedure for a 3×3 input and a 3×3 kernel. The inserted and border zeros are made explicit, and the final stride-one convolution yields a 7×7 output. These zeros are not inert; during the convolution they merely deactivate selected terms of the weighted sum, so that different output positions combine different subsets of input values and kernel weights, producing a genuine upsampling rather than a trivial zero-filling.

For a one-dimensional input of size i, kernel size k, stride s, and padding p, the spatial size of the transposed-convolution output is given

by

$$o = s\,(i-1) - 2p + k + a, \tag{3.12}$$

where a, with $0 \leq a < s$, is an optional *output padding* introduced to resolve the ambiguity that arises because several input sizes of a direct convolution may map to the same output size. In contrast to the direct convolution, whose output is generally smaller than its input, the transposed convolution typically produces a larger output, which is the desired upsampling behavior.

Analogously to the convolutional case, the forward propagation through a transposed convolutional layer can be written as

$$\mathbf{a}^{(l)} = \sigma^{(l)}\left(\mathbf{K}^{(l)} *^{\top} \mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}\right), \tag{3.13}$$

where $*^{\top}$ denotes the transposed-convolution operator, $\mathbf{K}^{(l)}$ is the trainable kernel tensor, $\mathbf{b}^{(l)}$ the bias term, and $\sigma^{(l)}$ is applied element-wise. The structural hyperparameters (stride, padding, dilation, and groups) play roles analogous to those of the direct convolution, but now act to expand rather than contract the spatial dimensions.

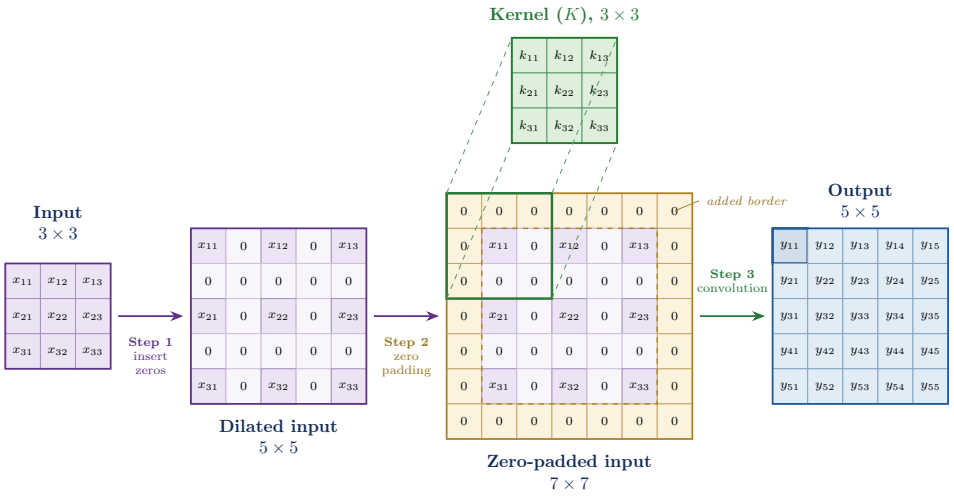


Figure 3.3: Schematic representation of a two-dimensional transposed convolution implemented as a direct convolution. The 3×3 input is dilated by inserting zeros (stride 2), padded with a zero border, and convolved with a 3×3 kernel using unit stride, yielding a 7×7 output.

Activation functions One of the key components of Artificial Neural Networks (ANNs) is the activation function. Its role is to introduce non-

linearity into the model, allowing the network to represent complex relationships beyond linear mappings. In practice, the activation function determines how the weighted input of a neuron is transformed before being propagated to subsequent layers, effectively controlling whether and how a neuron contributes to the overall network output.

Several activation functions have been proposed in the literature. Among the most commonly used ones is the hyperbolic tangent function (tanh), which maps the input to a bounded interval $(-1, 1)$ and is often preferred in physical and dynamical contexts due to its smoothness and symmetry around zero. Another widely used activation function is the sigmoid function, which produces outputs in the interval $(0, 1)$ and has historically been employed in early neural network models, particularly in probabilistic interpretations. More recently, the Rectified Linear Unit (ReLU) has become popular due to its computational simplicity and its ability to alleviate vanishing gradient effects in deeper architectures.

The choice of activation function directly impacts the expressive power, numerical stability, and trainability of the network, and therefore constitutes an important design consideration in neural network models. Figure 3.4 illustrates several commonly used activation functions.

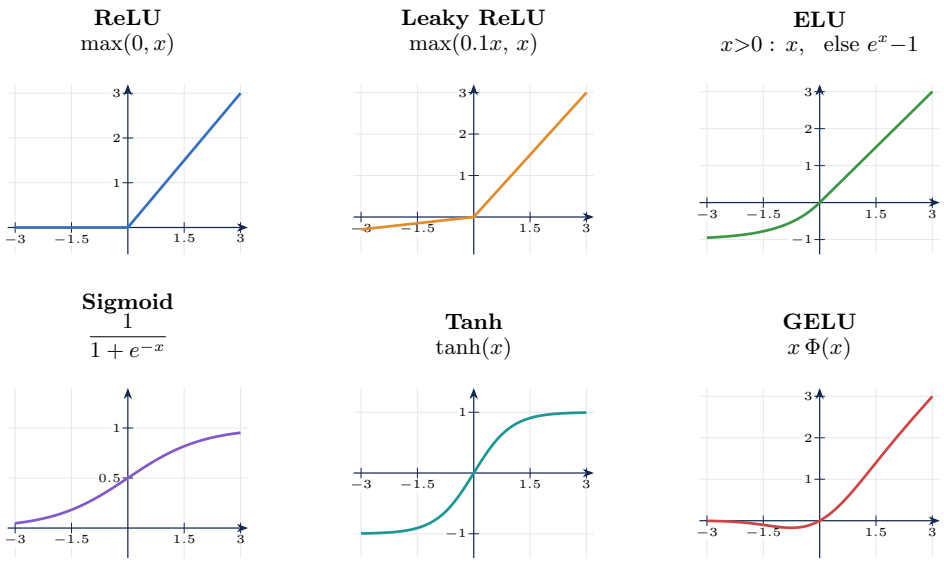


Figure 3.4: Examples of commonly used activation functions, including tanh, sigmoid, and ReLU.

Autoencoders An autoencoder is a type of artificial neural network trained, in a self-supervised manner, to reproduce its own input at the

output while passing the information through an internal representation of constrained capacity. Its purpose is therefore not to predict an external target but to learn a compact and informative encoding of the data. By forcing the signal through a bottleneck, the network is compelled to retain only the most salient structure of the input and to discard redundant or noisy components. Formally, an autoencoder is composed of two maps, an encoder f_θ and a decoder g_ϕ, parameterized by trainable weights θ and ϕ. The encoder transforms an input $\mathbf{x} \in \mathbb{R}^d$ into a latent code $\mathbf{z} = f_\theta(\mathbf{x}) \in \mathbb{R}^{d'}$, usually of lower dimension $d' < d$, and the decoder maps this code back to the input space, producing a reconstruction $\hat{\mathbf{x}} = g_\phi(\mathbf{z}) = g_\phi(f_\theta(\mathbf{x}))$.

Structurally, an autoencoder is organized symmetrically around the latent space, as illustrated schematically in Figure 3.5. The encoder consists of a sequence of layers that progressively reduce the dimensionality of the representation until reaching the latent (or bottleneck) layer, which holds the compressed code, while the decoder mirrors this structure, progressively expanding the latent representation back to the original dimensionality. When the data possess spatial structure, such as images or discretized physical fields, these layers are typically convolutional: the encoder employs convolutional layers followed by non-linear activations to contract the spatial resolution while increasing the number of feature channels, whereas the decoder employs transposed convolutional layers and activations to recover the original spatial resolution from the latent code. This configuration is known as a convolutional autoencoder, and it is the one adopted in this work.

The parameters of both maps are optimized jointly by minimizing a reconstruction loss that quantifies the discrepancy between the input and its reconstruction. For real-valued data a common choice is the mean squared error, so that the training objective takes the form

$$\mathscr{L}(\theta, \phi) = \frac{1}{N} \sum_{n=1}^{N} \left\| \mathbf{x}_n - g_\phi\big(f_\theta(\mathbf{x}_n)\big) \right\|^2, \tag{3.14}$$

where $\{\mathbf{x}_n\}_{n=1}^N$ denotes the training set. This objective is minimized through the same gradient-based optimization and backpropagation procedures used for supervised networks (introduced in Sections 3.4 and 3.5), with the crucial difference that the targets are the inputs themselves. When the latent dimension is smaller than the input dimension

(the so-called undercomplete regime), the bottleneck prevents the network from learning the trivial identity mapping and instead drives it to perform a non-linear dimensionality reduction. In fact, a linear autoencoder trained under the mean squared error recovers the same subspace as principal component analysis (PCA), so that autoencoders can be regarded as a non-linear generalization of classical linear dimensionality-reduction techniques. If, on the contrary, the latent space is overcomplete, with a dimension equal to or larger than that of the input, additional regularization must be imposed in order to obtain useful representations and to prevent the network from simply copying its input.

Several variants extend this basic scheme by modifying the architecture or the training objective. Denoising autoencoders are trained to reconstruct a clean input from a corrupted version of it, which encourages robustness and the extraction of stable features. Sparse and contractive autoencoders introduce penalty terms that promote, respectively, sparse activations or representations that are locally invariant to small perturbations of the input. Variational autoencoders, in turn, reinterpret the latent space in probabilistic terms and impose a prior distribution on the code, turning the autoencoder into a generative model from which new samples can be drawn. Despite these differences, all of these models share the common encoder–bottleneck–decoder structure described above.

Owing to their ability to learn compact representations without labeled data, autoencoders have found a wide range of applications. They are routinely used for dimensionality reduction and unsupervised representation learning, where the latent code serves as a low-dimensional feature vector for downstream tasks. Because the reconstruction error tends to be small for inputs similar to those seen during training and large for atypical ones, autoencoders are also effective for anomaly and novelty detection. The denoising variants are employed for data restoration and noise removal, while the generative variants are used for data synthesis and for modeling complex data distributions. In scientific and engineering contexts they are increasingly applied to compress high-dimensional physical fields, to construct reduced-order models of dynamical systems, and to learn informative representations of complex states, which makes them a natural tool within the data-driven modeling of physical and quantum systems considered in this work.

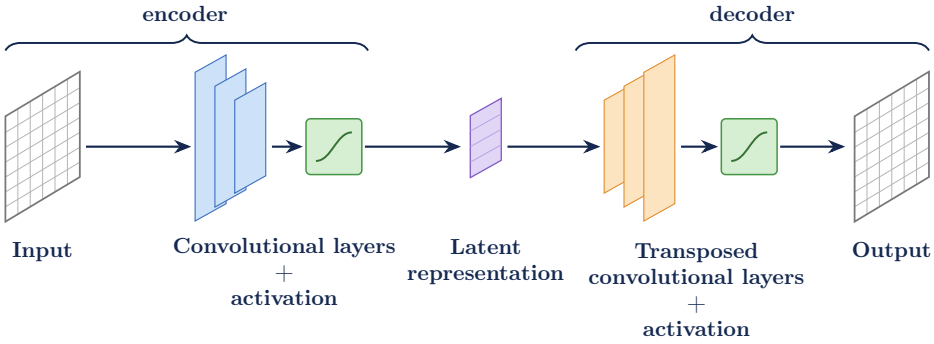


Figure 3.5: Schematic representation of a convolutional autoencoder. The encoder applies convolutional layers and activations to compress the input into a low-dimensional latent representation, while the decoder applies transposed convolutional layers and activations to reconstruct the input from that latent code.

3.4 Loss functions and training objective

The versatility of Deep Learning models stems from the wide range of tasks that can be addressed using them, some of which were outlined in the Introduction. However, one of the most important aspects of these models is not the architectures themselves, but rather the procedures employed to train them for a specific task. At a conceptual level, training is based on comparing the model output with a reference dataset and iteratively adjusting the internal parameters of the model in order to minimize this discrepancy.

More precisely, the training process relies on the definition of a scalar function that quantifies the disagreement between the model prediction and the target data. Let $\mathscr{L}$ denote such a function, defined here as the squared distance between the network output $\mathbf{y}_{\text{net}}$ and the reference data $\mathbf{d}$,

$$\mathscr{L} = \sum_i (y_i - d_i)^2. \tag{3.15}$$

This quantity is commonly referred to as the *loss function* or *cost function*. Here the index i runs over the components of the output vector for a single input sample; when a full dataset is considered, the total objective is obtained by averaging this per-sample loss over all training examples, as in the linear-regression example above. When the internal parameters of the model are appropriately chosen, the value of $\mathscr{L}$ approaches zero, indicating that the model successfully reproduces the target data $\mathbf{d}$ and, consequently, performs the desired task.

For illustrative purposes, consider a model composed of a single fully connected layer. In this case, the trainable parameters correspond to the weights and biases of the perceptrons. For an input vector $\mathbf{x} = (x_1, \ldots, x_n)$, the pre-activation variable associated with the i-th neuron is given by

$$z_i = \sum_j W_{ij} x_j + b_i, \tag{3.16}$$

and the corresponding network output is

$$y_i = \sigma(z_i), \tag{3.17}$$

where $\sigma(\cdot)$ denotes the activation function. The loss function in Eq. (3.15) can then be written as

$$\mathscr{L} = \sum_i \left(\sigma\left(\sum_j W_{ij} x_j + b_i \right) - d_i \right)^2. \tag{3.18}$$

The optimization of the model parameters is carried out by evaluating the partial derivatives of the loss function with respect to each trainable parameter. Since each neuron i is associated with a set of weights W_{ij}, the gradient of the loss function is naturally expressed using two indices. Applying the chain rule, the partial derivative of $\mathscr{L}$ with respect to a weight parameter W_{ij} reads

$$\frac{\partial \mathscr{L}}{\partial W_{ij}} = \frac{\partial \mathscr{L}}{\partial y_i} \frac{\partial y_i}{\partial z_i} \frac{\partial z_i}{\partial W_{ij}}. \tag{3.19}$$

This expression quantifies how variations in a specific weight W_{ij} affect the model performance and forms the basis of gradient-based optimization algorithms employed during the training process. The central idea is to iteratively update the weights in the direction of the negative gradient, thereby minimizing the value of the loss function $\mathscr{L}$. This procedure is commonly referred to as the *gradient descent* method.

When the model consists of more than a single layer, the computation of the gradients must be performed sequentially, starting from the output layer and propagating backwards through the network. This process, well established in the literature, is known as *backpropagation*. It constitutes one of the central pillars of Deep Learning and enables the efficient training of neural networks containing millions of parameters.

For a fully connected neural network composed of L layers, the forward propagation can be written in a compact form as shown in Eq. (3.6). Within this formulation, the gradient of the loss function with respect to the parameters of an intermediate layer can be expressed through successive applications of the chain rule. In particular, the gradient of $\mathscr{L}$ with respect to the weight matrix of the l-th layer, with $l \in \{1,\dots,L\}$, takes the general form

$$\frac{\partial \mathscr{L}}{\partial \mathbf{W}^{(l)}} = \frac{\partial \mathscr{L}}{\partial \mathbf{a}^{(L)}} \prod_{k=l+1}^{L} \frac{\partial \mathbf{a}^{(k)}}{\partial \mathbf{a}^{(k-1)}} \frac{\partial \mathbf{a}^{(l)}}{\partial \mathbf{W}^{(l)}}, \tag{3.20}$$

where the explicit structure of each term depends on the network architecture, including the number of layers, the number of neurons per layer, and the choice of activation functions. This expression should be understood schematically. Each factor $\partial \mathbf{a}^{(k)}/\partial \mathbf{a}^{(k-1)}$ is a Jacobian matrix, and the product must be evaluated in the appropriate order, from the output layer L down to layer l.

Therefore, in practical scenarios, the explicit analytical form of the gradients is not written out in full. Instead, gradient-based optimization relies on algorithmic implementations of backpropagation, which efficiently evaluate these derivatives without requiring their explicit expansion.

The explicit form of the gradients differs when the architecture incorporates convolutional layers instead of fully connected ones. In this case the trainable parameters are convolutional kernels rather than individual weights, and the resulting gradients reflect aggregated contributions over the corresponding receptive fields rather than local derivatives associated with single connections. The training objective, however, remains unchanged, and as before the explicit gradients are evaluated automatically by backpropagation. Understanding the general mechanism by which they are computed and propagated nonetheless remains important for the correct design and interpretation of convolutional architectures.

Examples of common loss functions While the squared loss introduced in Eq. (3.15) provides a simple and illustrative example, different tasks often require alternative choices of loss functions. The selection of an appropriate loss function reflects both the nature of the data and the objective of the learning problem, and may significantly influence the

training dynamics and the resulting model performance.

For regression tasks, the mean squared error (MSE) is among the most commonly used loss functions, as it directly penalizes deviations between predicted and target values. In classification problems, loss functions such as the binary or categorical cross-entropy are typically employed, as they are better suited to probabilistic interpretations of the network output. Other loss functions, including absolute error measures or task-specific objective functions, may be adopted depending on the desired robustness or structural constraints of the problem.

In physics-based applications, the loss function may further incorporate additional terms encoding prior knowledge or physical constraints, such as conservation laws or differential equations. In such cases, the training objective is no longer solely data-driven but reflects a combination of empirical accuracy and physical consistency. These extended formulations will be discussed in later sections.

3.5 Optimization and training

As discussed previously, neural networks are trained by optimizing a loss function in order to reproduce a set of known data samples. The underlying idea is that, given a sufficiently representative dataset, the model is able to identify relevant patterns such that, after training, it can successfully perform a specific task. From this perspective, the training of a neural network can be naturally interpreted as an optimization problem defined over a high-dimensional parameter space.

A fundamental connection therefore arises between neural networks and optimization theory. The adjustment of the trainable parameters is carried out through backpropagation, which provides the gradients of the loss function with respect to the model parameters. These gradients are then used by optimization algorithms to iteratively update the parameters. Among the most widely used optimization methods are Stochastic Gradient Descent (SGD) and adaptive schemes such as Adam, both of which are employed in the applications presented in this thesis. A comprehensive overview of optimization algorithms commonly used in deep learning can be found in Ref. [4].

In its general form, gradient-based optimization updates the param-

eters $\boldsymbol{\theta}$ of the model according to

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \, \nabla_{\boldsymbol{\theta}} \mathscr{L}(\boldsymbol{\theta}_t), \tag{3.21}$$

where $\boldsymbol{\theta}$ denotes the collection of all trainable parameters of the model and $\nabla_{\boldsymbol{\theta}} \mathscr{L}$ represents the vector of partial derivatives of the loss function with respect to each parameter. Explicitly, for a parameter θ_k, this update rule reads

$$\theta_k^{(t+1)} = \theta_k^{(t)} - \eta \left. \frac{\partial \mathscr{L}}{\partial \theta_k} \right|_{\boldsymbol{\theta}=\boldsymbol{\theta}_t}. \tag{3.22}$$

Here, $\eta > 0$ is the *learning rate*, a hyperparameter that controls the magnitude of the parameter updates. The learning rate plays a central role in the training dynamics: values that are too large may lead to unstable optimization, while excessively small values can result in slow convergence or stagnation. In Stochastic Gradient Descent, the gradient $\nabla_{\boldsymbol{\theta}} \mathscr{L}$ is not evaluated over the full dataset but estimated from a randomly sampled subset (a *mini-batch*) of the training data at each iteration. This reduces the computational cost per update and introduces a stochastic component that can help the optimizer escape shallow local minima.

In practical implementations, the learning rate is often adjusted during training through the use of *learning rate schedulers*, which progressively reduce its value as the optimization proceeds. This strategy allows for larger exploratory steps during the early stages of training, followed by finer adjustments as the loss function approaches a minimum.

Despite the effectiveness of gradient-based methods, the optimization landscape of deep neural networks is typically highly non-convex and may exhibit flat regions or multiple local minima. A well-known challenge in this context is the appearance of *vanishing gradients*, together with extended flat regions and saddle points of the loss landscape, where the gradients become extremely small and hinder efficient training.[2] Various techniques have been proposed to mitigate these issues, including the introduction of momentum terms, adaptive learning rate methods, and carefully designed initialization strategies. Such approaches aim to improve convergence by enabling the optimizer to escape shallow local minima or flat regions of the loss landscape.

[2] A closely related phenomenon, known as the *barren plateau*, arises in variational quantum circuits, where the gradients vanish exponentially with the number of qubits. The term is specific to that quantum-computing setting and is therefore avoided here in the classical context.

3.6 Physics-Informed Neural Networks

A central component in the development of deep learning models is the training stage. As previously discussed, training corresponds to the process by which the model learns to perform a specific task by adjusting its internal parameters through optimization. In standard data-driven approaches, this learning is achieved by supplying examples to the model and minimizing a loss function that quantifies the discrepancy between the model outputs and the target data. Through this process, neural networks are able to capture complex internal relationships between the variables involved in the task.

However, in physical sciences, the available information is not limited to observational or simulated data alone. Physical systems are also governed by well-established theoretical principles, typically expressed in the form of conservation laws, symmetry constraints, or differential equations. Ignoring this prior knowledge during training leads to models that are purely empirical, often requiring large datasets and offering limited extrapolation capabilities. Physics-informed learning addresses this limitation by explicitly incorporating known physical laws into the training process.

Physics-Informed Neural Networks (PINNs) constitute a class of models in which prior physical information is embedded directly into the learning procedure. This is commonly achieved by augmenting the loss function with additional terms that penalize violations of the governing equations or physical constraints, or by applying mathematical transformations to the network outputs in order to enforce such constraints by construction. As a consequence, the optimization process is guided not only by data fidelity but also by physical consistency.

This strategy offers several practical advantages. By constraining the hypothesis space of the model to physically admissible solutions, PINNs typically require fewer training examples, exhibit improved generalization, and converge faster during optimization. Moreover, the resulting models tend to be more robust to noise and better suited for interpolation and extrapolation tasks within the physically relevant regime.

A widely studied and natural application of PINNs is the solution of partial differential equations, where the neural network is trained to approximate the solution of a differential equation while minimizing the residual of the governing equation at a set of collocation points. This

framework has been successfully applied to a broad range of problems in classical mechanics, fluid dynamics, and electromagnetism. Comprehensive discussions and extensions of this methodology can be found in Refs. [8, 60, 62–66].

In this thesis, however, the physics-informed paradigm is employed beyond its conventional role as a differential equation solver. Two applications are presented in the context of quantum information where physics-informed learning is used to address complex physical and mathematical problems. In both cases, known structural properties of quantum systems are incorporated into the learning process, enabling the construction of models that are not only accurate but also consistent with the underlying quantum dynamics. These examples demonstrate that physics-informed neural networks provide a flexible framework for integrating deep learning techniques with the fundamental principles of quantum theory.

General formulation of Physics-Informed Neural Networks Let $\mathscr{N}_{\boldsymbol{\theta}}(\mathbf{x},t)$ be a neural network parametrized by $\boldsymbol{\theta}$, intended to approximate the solution $u(\mathbf{x},t)$ of a physical system governed by a differential equation of the form

$$\mathscr{D}[u(\mathbf{x},t)] = 0, \tag{3.23}$$

where $\mathscr{D}$ denotes a (possibly nonlinear) differential operator acting on the solution.

In the physics-informed framework, the neural network output is identified with the trial solution,

$$u(\mathbf{x},t) \approx \mathscr{N}_{\boldsymbol{\theta}}(\mathbf{x},t), \tag{3.24}$$

and automatic differentiation is employed to compute the required spatial and temporal derivatives. The physical consistency of the model is enforced by defining a residual function

$$\mathscr{R}(\mathbf{x},t;\boldsymbol{\theta}) := \mathscr{D}[\mathscr{N}_{\boldsymbol{\theta}}(\mathbf{x},t)]. \tag{3.25}$$

The training objective is constructed by minimizing a composite loss function of the form

$$\mathscr{L} = \mathscr{L}_{\text{data}} + \lambda_{\text{phys}}\mathscr{L}_{\text{phys}} + \lambda_{\text{bc}}\mathscr{L}_{\text{bc}}, \tag{3.26}$$

where $\mathscr{L}_{\text{data}}$ enforces agreement with known data (if available), $\mathscr{L}_{\text{phys}}$ penalizes violations of the governing equations,

$$\mathscr{L}_{\text{phys}} = \frac{1}{N_r} \sum_{i=1}^{N_r} \left| \mathscr{R}(\mathbf{x}_i, t_i; \boldsymbol{\theta}) \right|^2, \tag{3.27}$$

and $\mathscr{L}_{\text{bc}}$ incorporates boundary and initial conditions. The coefficients λ_{phys} and λ_{bc} control the relative importance of each contribution.

This formulation restricts the hypothesis space of the neural network to functions that are compatible with the known physical laws, effectively embedding prior knowledge into the optimization process.

Illustrative examples: solving differential equations with PINNs To illustrate the physics-informed approach, representative examples are now considered where PINNs are employed to solve differential equations of increasing complexity.

One-dimensional wave equation. A standard benchmark problem for PINN is the one-dimensional wave equation,

$$\frac{\partial^2 u(x,t)}{\partial t^2} = c^2 \frac{\partial^2 u(x,t)}{\partial x^2}, \tag{3.28}$$

defined on the spatial domain $x \in [0,1]$ and temporal domain $t \in [0,1]$. Fixed (Dirichlet) boundary conditions are considered,

$$u(0,t) = u(1,t) = 0, \tag{3.29}$$

together with the initial conditions

$$u(x,0) = \sin(\pi x), \qquad \partial_t u(x,0) = 0. \tag{3.30}$$

For the particular choice $c = 1$, the analytical solution is given by

$$u(x,t) = \sin(\pi x)\cos(\pi t). \tag{3.31}$$

Within the PINN framework, a neural network $\mathscr{N}_{\boldsymbol{\theta}}(x,t)$ is used to approximate the displacement field $u(x,t)$. The residual of the wave

equation is defined as

$$\mathscr{R}_{\boldsymbol{\theta}}(x,t) = \partial_t^2 \mathscr{N}_{\boldsymbol{\theta}}(x,t) - c^2 \partial_x^2 \mathscr{N}_{\boldsymbol{\theta}}(x,t). \tag{3.32}$$

Let $\{(x_i,t_i)\}_{i=1}^{N_f}$ denote the interior collocation points, $\{x_j^{(0)}\}_{j=1}^{N_0}$ the points used to enforce the initial conditions, and $\{(x_k^{(b)},t_k^{(b)})\}_{k=1}^{N_b}$ the boundary points. The discrete loss function minimized during training is given by

$$\begin{aligned}\mathscr{L}(\boldsymbol{\theta}) = \frac{1}{N_f}\sum_{i=1}^{N_f} |\mathscr{R}_{\boldsymbol{\theta}}(x_i,t_i)|^2 + \frac{1}{N_0}\sum_{j=1}^{N_0} \left|\mathscr{N}_{\boldsymbol{\theta}}(x_j^{(0)},0) - \sin\left(\pi x_j^{(0)}\right)\right|^2 \\ + \frac{1}{N_0}\sum_{j=1}^{N_0} \left|\partial_t \mathscr{N}_{\boldsymbol{\theta}}(x_j^{(0)},0)\right|^2 + \frac{1}{N_b}\sum_{k=1}^{N_b} \left|\mathscr{N}_{\boldsymbol{\theta}}(x_k^{(b)},t_k^{(b)})\right|^2. \end{aligned} \tag{3.33}$$

Figure 3.6 illustrates the standard evaluation procedure, where the predicted solution surface $\mathscr{N}_{\boldsymbol{\theta}}(x,t)$ is compared with the analytical solution, together with a visualization of the absolute error over the full space–time domain. This combined analysis provides both a quantitative and qualitative assessment of the network's accuracy in capturing the wave dynamics.

Nonlinear scalar field dynamics. As a more complex example, the nonlinear Klein–Gordon equation in $(2+1)$ dimensions is considered,

$$\frac{\partial^2 \phi(\mathbf{x},t)}{\partial t^2} - \nabla^2 \phi(\mathbf{x},t) + V'(\phi) = 0, \tag{3.34}$$

where $\mathbf{x} = (x,y)$, $\nabla^2 = \partial_x^2 + \partial_y^2$ denotes the spatial Laplacian, and $V(\phi)$ is a nonlinear potential. Such equations arise naturally in classical field theory, high-energy physics, and cosmology, and generally do not admit closed-form analytical solutions when nonlinear interactions are present.

In particular, a quartic self-interaction potential is considered, of the form

$$V(\phi) = \frac{1}{2}m^2\phi^2 + \frac{\lambda}{4}\phi^4, \tag{3.35}$$

so that

$$V'(\phi) = m^2\phi + \lambda\phi^3. \tag{3.36}$$

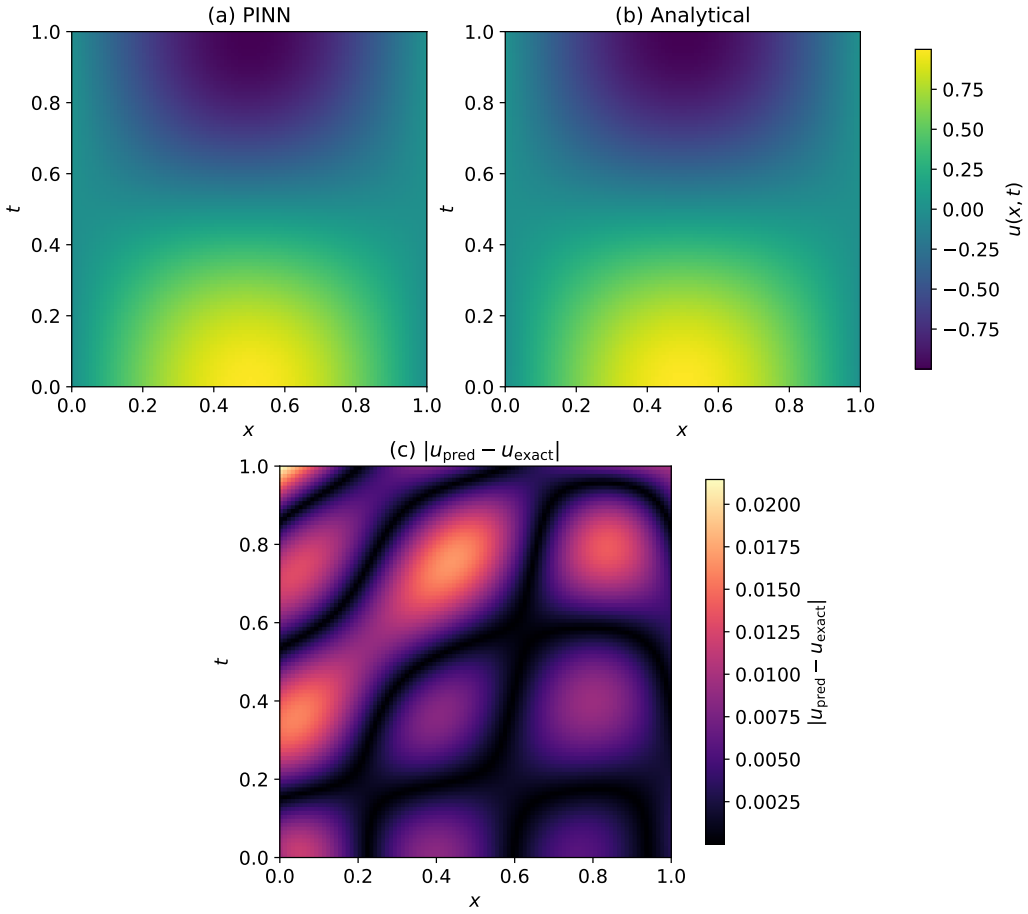


Figure 3.6: Comparison between the PINN prediction (left), the analytical solution (center), and the absolute error (right). The color scale of the error panel shows that the discrepancy remains uniformly small across the domain, indicating that the network accurately captures both the wave propagation and the imposed boundary conditions.

The resulting evolution equation becomes

$$\partial_t^2\phi - \partial_x^2\phi - \partial_y^2\phi + m^2\phi + \lambda\phi^3 = 0. \tag{3.37}$$

The field is defined on a spatial domain $(x,y) \in \Omega \subset \mathbb{R}^2$ and time interval $t \in [0,T]$, supplemented with initial conditions

$$\phi(x,y,0) = \phi_0(x,y), \qquad \partial_t\phi(x,y,0) = \pi_0(x,y), \tag{3.38}$$

and suitable boundary conditions (e.g., Dirichlet or periodic).

Within the PINN framework, a neural network $\mathscr{N}_{\boldsymbol{\theta}}(x,y,t)$ is employed to approximate the scalar field $\phi(x,y,t)$. A key advantage of PINNs in this context is that the nonlinear interaction term is handled directly through automatic differentiation, without requiring explicit discretization schemes for the spatial derivatives or nonlinear operator.

The corresponding physics residual is defined as

$$\mathscr{R}_{\boldsymbol{\theta}}(x,y,t) = \partial_t^2\mathscr{N}_{\boldsymbol{\theta}} - \partial_x^2\mathscr{N}_{\boldsymbol{\theta}} - \partial_y^2\mathscr{N}_{\boldsymbol{\theta}} + m^2\mathscr{N}_{\boldsymbol{\theta}} + \lambda\mathscr{N}_{\boldsymbol{\theta}}^3. \tag{3.39}$$

Let $\{(x_i,y_i,t_i)\}_{i=1}^{N_f}$ denote interior collocation points, $\{(x_j^{(0)},y_j^{(0)})\}_{j=1}^{N_0}$ the initial-condition points, and $\{(x_k^{(b)},y_k^{(b)},t_k^{(b)})\}_{k=1}^{N_b}$ boundary points. The discrete loss function minimized during training is given by

$$\begin{aligned}\mathscr{L}(\boldsymbol{\theta}) = \frac{1}{N_f}\sum_{i=1}^{N_f}|\mathscr{R}_{\boldsymbol{\theta}}(x_i,y_i,t_i)|^2 + \frac{1}{N_0}\sum_{j=1}^{N_0}\left|\mathscr{N}_{\boldsymbol{\theta}}(x_j^{(0)},y_j^{(0)},0) - \phi_0(x_j^{(0)},y_j^{(0)})\right|^2 \\ + \frac{1}{N_0}\sum_{j=1}^{N_0}\left|\partial_t\mathscr{N}_{\boldsymbol{\theta}}(x_j^{(0)},y_j^{(0)},0) - \pi_0(x_j^{(0)},y_j^{(0)})\right|^2 + \frac{1}{N_b}\sum_{k=1}^{N_b}\left|\mathscr{N}_{\boldsymbol{\theta}}(x_k^{(b)},y_k^{(b)},t_k^{(b)})\right|^2\end{aligned} \tag{3.40}$$

Due to the cubic nonlinear term, the dynamics exhibit amplitude-dependent dispersion and nontrivial mode coupling, rendering this problem significantly more challenging than the linear wave equation.

Figure 3.7 displays representative snapshots of the scalar field at different times, while the full dynamical evolution is provided in an accompanying video[3]. The results reveal a radially propagating structure

[3] https://youtu.be/bxEir5o0pdk

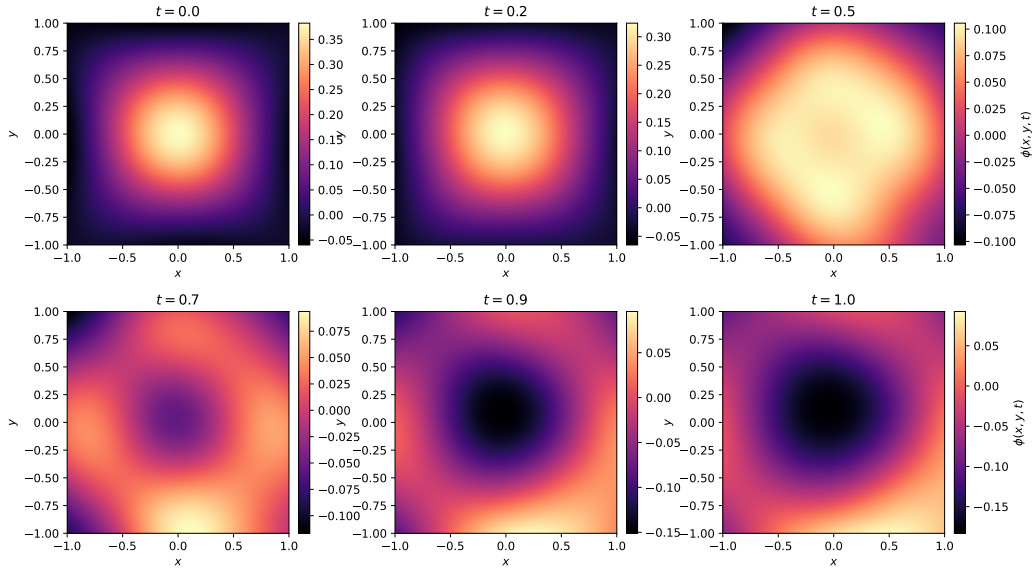


Figure 3.7: Snapshots of the scalar field $\phi(x, y, t)$ at different times. The nonlinear self-interaction generates ring-shaped wavefronts that propagate radially outward from the center, illustrating the nonlinear dynamics obtained with the PINN.

emerging from the central region. As the field evolves, nonlinear self-interaction induces deformation of the initial profile, generating ring-like wavefronts that expand outward from the origin. Unlike the wave-equation case, no closed-form solution is available for this configuration, so the assessment here is qualitative and the predicted field is not compared against an independent reference solution.

These examples illustrate how physics-informed neural networks provide a unified and flexible framework for solving both linear and nonlinear differential equations, serving as a foundation for more advanced applications in complex physical systems.

3.7 Summary

This chapter introduced the fundamental concepts required to understand and implement deep learning models within a rigorous mathematical framework. Beginning with the general formulation of learning problems in terms of tasks, experience, and performance measures, the conceptual foundations of machine learning and its specialization into deep learning architectures were established.

The structure of artificial neural networks was reviewed, including fully connected, convolutional, and transposed convolutional layers, along

with the role of activation functions in enabling nonlinear representation learning. Autoencoders were also introduced as a representative unsupervised architecture, built from these layers around a compressed latent representation. The training process was formulated as a high-dimensional optimization problem, where backpropagation and gradient-based methods allow efficient parameter updates. Particular emphasis was placed on the interpretation of loss functions, since they define the objective that guides the learning dynamics.

The chapter culminated with the introduction of Physics-Informed Neural Networks, which extend conventional data-driven models by incorporating physical constraints directly into the training objective. By embedding differential equations, boundary conditions, and structural properties into the loss function, PINNs restrict the hypothesis space to physically admissible solutions. This approach enables accurate modeling of complex dynamical systems while reducing the dependence on large datasets.

The examples of the one-dimensional wave equation and the nonlinear Klein–Gordon equation in $(2+1)$ dimensions demonstrated how deep learning architectures can approximate both linear and nonlinear field dynamics. These case studies illustrate the capacity of physics-informed models to capture intricate spatiotemporal behavior while maintaining consistency with governing physical laws.

The framework developed in this chapter serves as the methodological foundation for the subsequent applications presented in this thesis, where physics-informed deep learning is employed to address advanced problems in quantum information and quantum dynamics.

Parte II

Scientific contributions

Chapter 4

INTERPOLATION OF UNITARIES WITH TIME-DEPENDENT HAMILTONIANS VIA DEEP LEARNING

This chapter presents the first main application developed in this thesis, a physics-informed deep learning framework for estimating the unitary time-evolution operator of quantum systems governed by time-dependent Hamiltonians over a full time domain. The results reported here correspond to the work submitted for publication in *Machine Learning: Science and Technology* [40].

The theoretical background required for this chapter was developed in Chapter 2: the von Neumann equation (2.22) and its formal solution in terms of the time-ordered exponential (2.24), the Magnus expansion (2.27), the Trotter product formula (2.35), the Pauli-basis representation of N-qubit operators (Section 2.5), and the gate fidelity (2.72) between unitary channels obtained through the Choi–Jamiołkowski isomorphism (2.67). The deep learning machinery was introduced in Chapter 3: fully connected architectures (3.6), loss functions (Section 3.4), gradient-based optimization (Section 3.5), and the physics-informed paradigm (Section 3.6). In accordance with Chapter 2, natural units with $\hbar = 1$ are used throughout this chapter. The complete code for reproduction is available in a GitHub repository.[1]

4.1 Problem statement

A fundamental challenge in quantum theory is the accurate modeling of the dynamics of quantum systems driven by time-dependent Hamiltonians. Unlike the time-independent case, where the time-evolution operator reduces to a simple matrix exponential e^{-iHt}, time-dependent Hamiltonians give rise to the time-ordering operator, which precludes a closed-form solution and introduces substantial mathematical and nu-

[1] https://github.com/guerra-antonio/PINN_dynamics_estimation

merical complexity. The cost of any direct approach is further compounded by the exponential growth of the Hilbert space [10], together with the rapidly growing number of terms required in the series expansions used to represent the time-dependent evolution operator $U(t)$. For an N-qubit system the evolution operator is a $2^N \times 2^N$ matrix, and its dense construction or exponentiation scales as $\mathcal{O}(2^{3N})$. Expansions such as the Dyson or Magnus series, discussed in Chapter 2, involve nested time integrals whose number grows with the expansion order, and the associated numerical errors tend to accumulate over long evolution times [22, 23]. Even on quantum hardware, simulating time-dependent dynamics is generally more demanding than the time-independent case, particularly for rapidly varying or highly oscillatory Hamiltonians [24]. Despite these difficulties, time-dependent Hamiltonians play a central role in quantum technologies, including quantum control and gate optimization [25], as well as quantum simulation of complex many-body systems [26, 27].

Several approaches have been developed to address this challenge. Methods based on truncated Dyson series expansions have shown promising results in handling the time-ordering operator [28, 29]. Magnus-expansion-based algorithms have proven particularly effective in quantum control [30], and recent work has extended these to highly-oscillatory Hamiltonians relevant to near-term quantum devices [31, 32]. Modified Suzuki–Trotter decompositions have been proposed for interpolating unitary operators in time-dependent settings [33]. Tensor-network methods provide a complementary route. They enable efficient classical simulation of one-dimensional systems through time-evolving block decimation [34], and recent constructions optimize Suzuki–Trotter decompositions for Hamiltonian simulation and adiabatic quantum computing [35, 36]. On the machine learning side, PINNs have been explored to learn quantum dynamics under physical constraints [21], as well as to reconstruct time-dependent Hamiltonians using deep learning models [37, 38]. Other measurement-based protocols have also been proposed for this task [39].

The specific problem addressed in this chapter is the following. Starting from the knowledge of the time-evolution operator $U(t_i)$ on a finite set $\{t_i\}$ $(i = 1, \ldots, N)$ of times with $t_i \in [0, T]$, we seek to interpolate $U(t)$ over the entire time interval $[0, T]$. The model takes a single scalar, the time t, as input and returns the corresponding time-evolution operator. The time-dependent Hamiltonian itself is not provided to the model;

it learns the dynamics solely from the sampled unitaries $U(t_i)$ and the associated evolved states. Throughout this chapter, the training data are generated numerically, but are intended to play the role of experimental data that, in practice, would be obtained, for example, by quantum process tomography (QPT). Each model is trained to represent the dynamics of a single Hamiltonian realization; learning a mapping that generalizes across different Hamiltonians is beyond the scope of this work.

In the language of Chapter 2, the available information consists of a known set of discrete data, either evolved states $\{\rho_0, \rho(t_i)\}_{i=1}^{N_1}$ or sampled unitaries $\left\{U(t_j,t_0)\right\}_{j=1}^{N_2}$, where the states evolve according to *Postulate 2*, $\rho(t_i) = U(t_i,t_0)\,\rho_0\,U^\dagger(t_i,t_0)$, and the sampled unitaries are solutions of the von Neumann dynamics, see Eq. (2.22). From this discrete data the model is trained to reconstruct $U(t)$ over the full interval $[0,T]$.

Our method is based on the PINN framework introduced in Section 3.6, in which known physical constraints are explicitly incorporated into the training process to guide the learning of physically consistent solutions. Due to the exponential growth of the Hilbert space with the number of qubits, we train separate models per system size and test two distinct model architectures: (i) a model trained to directly predict the unitary operator $U(t)$, and (ii) a model that predicts the coefficients of an effective time-dependent Hamiltonian $H_{\text{eff}}(t)$, from which $U(t)$ is subsequently computed via the Magnus expansion or Trotterization. The motivation for the second architecture is scalability. By reducing the task from learning the full unitary to learning an effective Hamiltonian, the number of free parameters is drastically reduced, lowering the computational resources required for training.

The effective Hamiltonian $H_{\text{eff}}(t)$ learned by the second architecture should be understood as a generator that reproduces the observed dynamics, rather than as a unique reconstruction of the physical Hamiltonian governing the system. The map from a Hamiltonian to its time-evolution operator is not injective, since distinct Hamiltonians can produce the same unitary $U(t)$, so that different generators may account for identical dynamics under the conditions probed during training. This non-uniqueness is a known feature of Hamiltonian characterization, where a Hamiltonian is not always *identifiable* from the measured dynamics alone [67, 68]. Consequently, our method targets the faithful emulation of the evolution rather than the identification of a unique underlying Hamiltonian, an important distinction relative to genuine Hamiltonian

learning.

4.2 Physical systems

We consider three types of Hamiltonians across different many-body configurations, all expressed in the N-qubit Pauli basis introduced in Section 2.5. For systems with up to 6 qubits, we employ the general Hamiltonian containing all possible Pauli components; for systems with 7 or 8 qubits, we restrict the Hamiltonian to two reduced nearest-neighbor models: an Ising model and an XYZ Heisenberg model. These are explicitly given by

$$H_{\text{general}}(t) = \sum_{\alpha_1,\ldots,\alpha_N} c_{\alpha_1\cdots\alpha_N}(t)\,\sigma_{\alpha_1}\otimes\cdots\otimes\sigma_{\alpha_N}, \tag{4.1}$$

$$H_{\text{Ising}}(t) = \sum_{j=1}^{N-1} J_j(t)\,\sigma_j^z\sigma_{j+1}^z + \sum_{j=1}^{N} h_j(t)\,\sigma_j^z, \tag{4.2}$$

$$H_{XYZ}(t) = \sum_{j=1}^{N-1}\left[J_j^x(t)\,\sigma_j^x\sigma_{j+1}^x + J_j^y(t)\,\sigma_j^y\sigma_{j+1}^y + J_j^z(t)\,\sigma_j^z\sigma_{j+1}^z\right]. \tag{4.3}$$

Here each index $\alpha_j \in \{I,x,y,z\}$ labels the Pauli operator acting on the j-th qubit, the real-valued coefficients $c_{\alpha_1\cdots\alpha_N}(t)$ carry the full time dependence, and $\sigma_j^{x,y,z}$ denote the Pauli operators acting on the j-th qubit. In the Ising model, $J_j(t)$ is the time-dependent coupling strength between neighboring qubits j and $j+1$, while $h_j(t)$ is the time-dependent longitudinal field acting on qubit j. In the XYZ Heisenberg model, $J_j^x(t)$, $J_j^y(t)$, and $J_j^z(t)$ are the time-dependent nearest-neighbor exchange couplings along the x, y, and z directions, respectively. All coupling and field coefficients are real-valued functions of time. For brevity, we collect the indices into a multi-index $\vec{\alpha} \equiv (\alpha_1,\ldots,\alpha_N)$, so that $c_{\vec{\alpha}}(t) \equiv c_{\alpha_1\cdots\alpha_N}(t)$ and $\sigma_{\vec{\alpha}} \equiv \sigma_{\alpha_1}\otimes\cdots\otimes\sigma_{\alpha_N}$.

The choice of a fully non-zero Pauli Hamiltonian for systems up to 6 qubits is motivated by its generality, since any N-qubit Hamiltonian can be expressed as a linear combination of all Pauli tensor products (cf. Section 2.5), a model capable of reproducing the dynamics generated by such a Hamiltonian should, in principle, be capable of handling any sparser Hamiltonian as a special case. This choice serves as a proof of principle. An architecture expressive enough to represent the dynam-

ics of such a general Hamiltonian is, *a fortiori*, expressive enough to represent that of any sparser, physically motivated Hamiltonian, although each case requires training a dedicated model on its own data. For systems of 7 and 8 qubits, computational constraints required the use of more structured Hamiltonians that are directly relevant to physically realizable systems. The nearest-neighbor Ising model with longitudinal couplings and fields describes the effective spin dynamics in trapped-ion quantum simulators [69], superconducting qubit architectures [70], and quantum annealing devices such as D-Wave processors, where a time-dependent Hamiltonian drives the system toward the ground state of a programmable Ising problem [71]. The nearest-neighbor XYZ Heisenberg model describes anisotropic exchange interactions present in magnetic materials [72], has been experimentally realized in trapped-ion chains [73, 74], and serves as the effective low-energy Hamiltonian in silicon spin-qubit platforms [75] and nuclear magnetic resonance quantum processors [76].

For all cases, the time-dependent coefficients take the form

$$c_{\vec{\alpha}}(t) = \frac{A_{\vec{\alpha}} \sin(\omega_{\vec{\alpha}} t + \phi_{\vec{\alpha}})}{\mathcal{N}_N}, \tag{4.4}$$

where $A_{\vec{\alpha}}$, $\omega_{\vec{\alpha}}$, and $\phi_{\vec{\alpha}}$ are the amplitude, angular frequency, and phase shift of each component, respectively, and $\mathcal{N}_N$ is a system-size-dependent normalization factor introduced to control the operator norm of the Hamiltonian, ensuring the convergence of the Magnus expansion (see Section 4.3.3). The parameters are independently drawn from uniform distributions: $\omega_{\vec{\alpha}}, \phi_{\vec{\alpha}} \in [0, 2\pi)$ and $A_{\vec{\alpha}} \in [0, 1)$. Once fixed for a given realization, they remain constant throughout training. The same functional form is used for the coupling and field coefficients of the Ising and XYZ systems.

The number of independent coefficients depends on the Hamiltonian structure: $4^N - 1$ for the general Pauli case (the global identity component is fixed to zero, see below), $2N-1$ for the Ising model, and $3(N-1)$ for the XYZ model. The normalization factors are

$$\mathcal{N}_2 = 2, \quad \mathcal{N}_3 = 3, \quad \mathcal{N}_4 = 4, \quad \mathcal{N}_5 = 16, \quad \mathcal{N}_6 = 24, \tag{4.5}$$

for the general Pauli Hamiltonian, $\mathcal{N}_7 = \mathcal{N}_8 = 2$ for the Ising, and $\mathcal{N}_7 = \mathcal{N}_8 = 3$ for the XYZ systems, reflecting the smaller number of non-zero

terms in the latter cases. Additionally, the coefficient associated with the global identity operator is set to zero in all cases, ensuring that the Hamiltonian is traceless.

The use of randomly sampled parameters allows the generation of a diverse ensemble of time-dependent Hamiltonians. Since a dedicated model is trained for each realization, this ensemble does not probe generalization across Hamiltonians, but supports the systematic assessment of the robustness of the method over independent realizations reported in Section 4.4.3.

4.3 Model description

The rationale for employing PINNs lies in their capability to incorporate physical and mathematical constraints directly into the training process. As discussed in Chapter 3, PINNs have demonstrated strong performance in solving nonlinear differential equations and in modeling complex dynamical systems [8, 9]. The physics-informed formulation also makes it possible to impose boundary conditions, initial conditions, and other structural constraints either directly on the network output or through the loss function.

We employ a model that generates a unitary complex-valued matrix from a scalar input. The number of complex entries of these matrices increases exponentially with the number of qubits as 4^N. Figure 4.1 shows the number of trainable parameters for models up to 6 qubits, as well as the estimated parameter counts required for 7- and 8-qubit systems. Since the optimization process becomes computationally intractable at larger scales, an alternative strategy is adopted for these higher-dimensional cases. Rather than merely constraining the Hamiltonian to a reduced set of components, the model is constructed to estimate an effective time-dependent set of coefficients defining the effective Hamiltonian of the system $H_{\text{eff}}(t)$. Subsequently, with a model capable of generating $H_{\text{eff}}(t)$, we can use it to compute the time-evolution operator through (i) the Magnus expansion or (ii) the Trotterization. This approach reduces the number of trainable parameters to approximately 10^3.

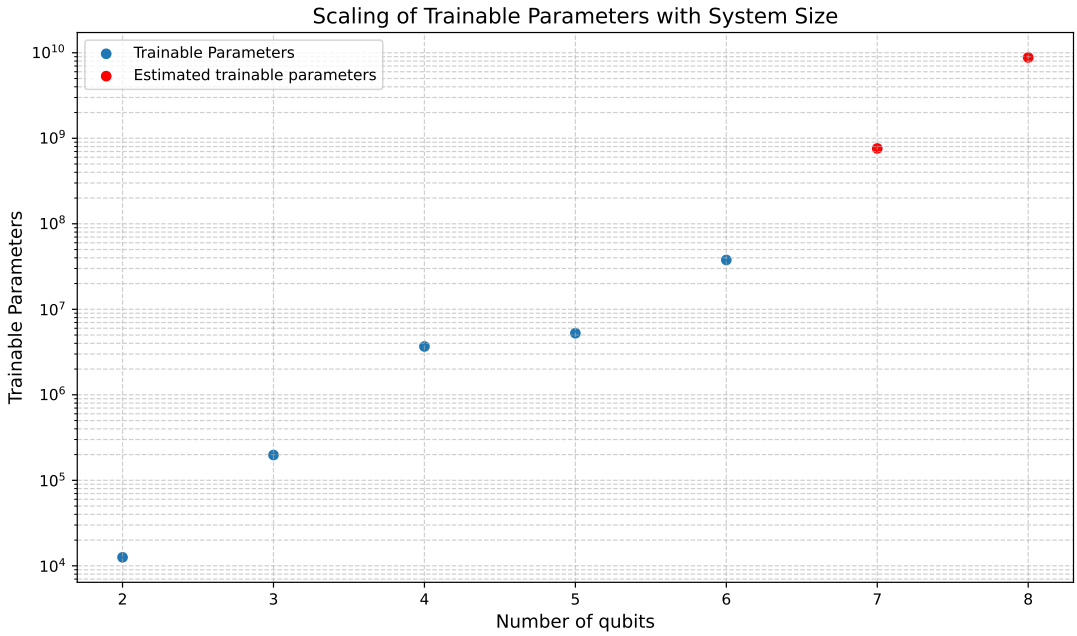
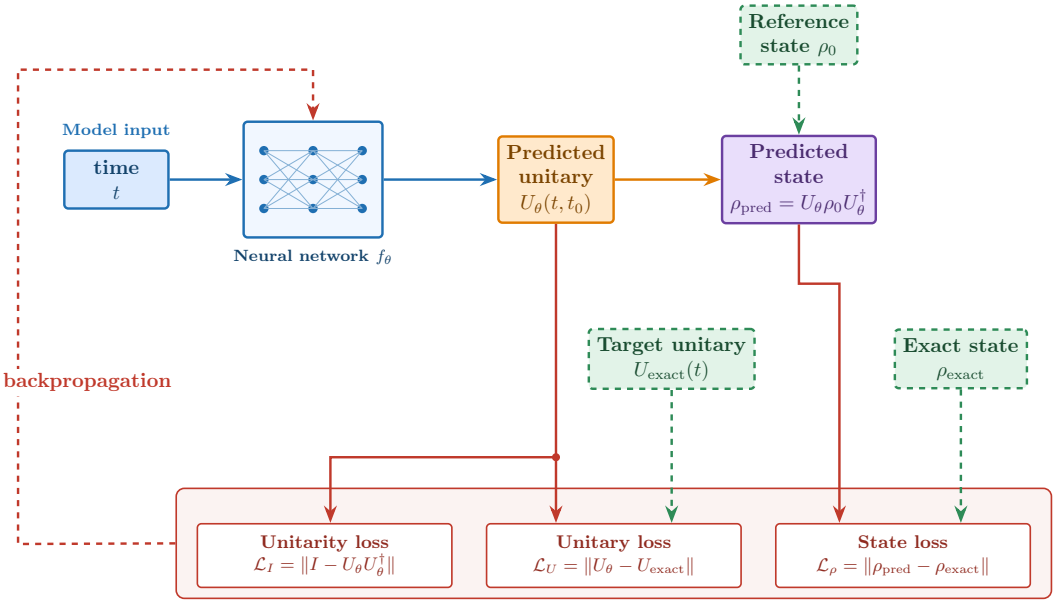


Figure 4.1: Amount of trainable parameters as a function of the number N of qubits. In cases with up to 6 qubits, this relationship is outlined. For systems containing 7 and 8 qubits, we have provided the estimated number of trainable parameters required to apply the same strategy to determine the unitary evolution operator for those systems.

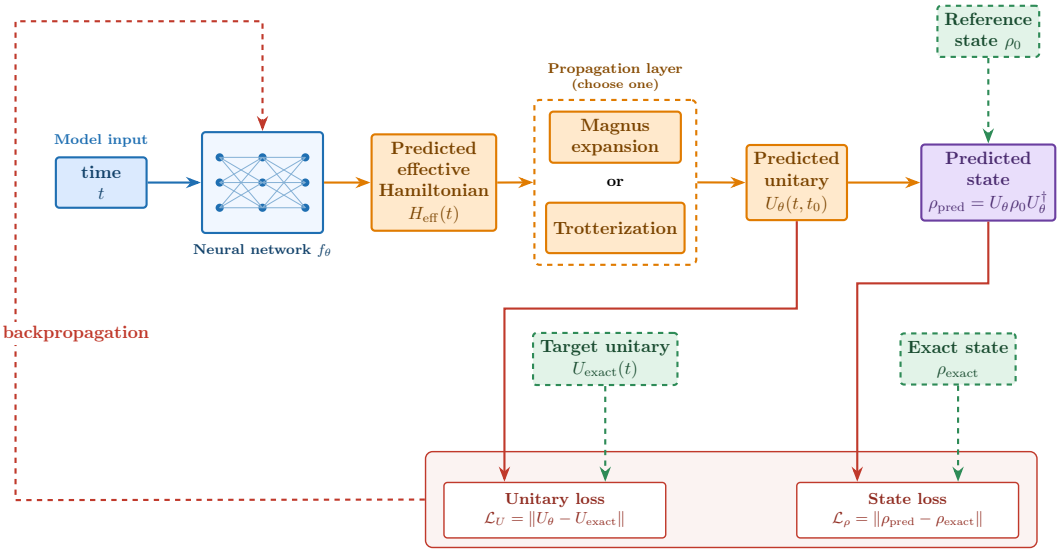
4.3.1 Architecture description

For the model architecture, we employ a straightforward approach consisting of a sequence of fully connected layers (cf. Section 3.3) designed to match the number of parameters required for the output matrix. The hyperbolic tangent function is used as the activation function. This architecture can generate arbitrarily complex-valued matrices, ensuring that the model can accurately represent any matrix form of the time-evolution operator. Figure 4.2 schematically illustrates the two modeling strategies adopted for the time-evolution operator.

Cases with 2 to 6 qubits. The model's internal data processing involves several fully connected layers. Initially, the input layer takes a scalar input corresponding to time. This input is processed through fully connected layers that progressively increase the dimensionality of the latent representation. After each hidden layer, a hyperbolic tangent activation function is applied. The resulting features are then reshaped into tensors of dimension $(2, 2^N, 2^N)$, where the first index is used to construct the real and imaginary parts of the resulting complex-valued



(a) PINN model for 2 to 6 qubits.



(b) PINN model for 7 and 8 qubits.

Figure 4.2: The two PINN architectures used to emulate the temporal evolution of isolated multi-qubit systems. The network f_θ takes time t as input and is trained via backpropagation. (a) For 2–6 qubits, it directly predicts the unitary $U_\theta(t,t_0)$, optimized using unitarity, unitary, and state losses. (b) For 7–8 qubits, it predicts an effective Hamiltonian $H_{\text{eff}}(t)$, which is converted into $U_\theta(t,t_0)$ through a propagation layer; unitarity is therefore guaranteed by construction. Colors: blue, inputs/outputs; orange, learned quantities; purple, predicted state; dashed green, reference and target quantities; red, losses and optimization path.

matrix. A detailed description of each architecture is presented in Table 4.1.

Cases with 7 and 8 qubits. For the models handling 7 and 8 qubits, the architecture follows a similar structure to the previous models but differs significantly in the output layer. Instead of generating a complex-valued matrix directly, the output layer produces time-dependent coefficients associated with the Pauli basis representation of the Hamiltonian $H(t)$ (cf. Section 2.5). The output dimension therefore equals the number of independent Hamiltonian coefficients, namely $2N-1$ for the Ising model (13 and 15 for $N=7,8$) and $3(N-1)$ for the XYZ model (18 and 21). A detailed description of these architectures is presented in Table 4.2, where the output sizes shown correspond to the Ising case.

model2 (2 qubits)

Layer	**Input**	**Output**
Linear 1	1	64
Tanh	-	-
Linear 2	64	128
Tanh	-	-
Linear 3	128	32
Reshape	32	$(2,4,4)$

model3 (3 qubits)

Layer	**Input**	**Output**
Linear 1	1	256
Tanh	-	-
Linear 2	256	512
Tanh	-	-
Linear 3	512	128
Reshape	128	$(2,8,8)$

model4 (4 qubits)

Layer	**Input**	**Output**
Linear 1	1	512
Tanh	-	-
Linear 2	512	1024
Tanh	-	-
Linear 3	1024	2048
Tanh	-	-
Linear 4	2048	512
Reshape	512	$(2,16,16)$

model5 (5 qubits)

Layer	**Input**	**Output**
Linear 1	1	512
Tanh	-	-
Linear 2	512	2048
Tanh	-	-
Linear 3	2048	2048
Reshape	2048	$(2,32,32)$

model6 (6 qubits)

Layer	**Input**	**Output**
Linear 1	1	1024
Tanh	-	-
Linear 2	1024	4096
Tanh	-	-
Linear 3	4096	8192
Reshape	8192	$(2,64,64)$

Table 4.1: Architectures of *model2* through *model6*, each representing the generator of a $2^N \times 2^N$ complex-valued unitary matrix from a scalar temporal input t.

model7 (7 qubits)		
Layer	**Input**	**Output**
Linear 1	1	50
Tanh	-	-
Linear 2	50	13

model8 (8 qubits)		
Layer	**Input**	**Output**
Linear 1	1	50
Tanh	-	-
Linear 2	50	15

Table 4.2: Architectures of *model7* and *model8*, designed to generate the real-valued coefficients of the Pauli-basis representation of the Hamiltonian from a scalar temporal input t. The output sizes shown (13 and 15) correspond to the Ising model $(2N-1)$; for the XYZ model the output size is $3(N-1)$, i.e. 18 and 21 for $N = 7, 8$.

4.3.2 Loss function and data generation

To train the model, a valid dataset is required. The dataset generation process is inspired by the idea that it can be interpreted as the outcome of multiple QPT experiments performed at different times [13]. If the goal is to perform a quantum control task, the dataset will consist of a set of quantum states forming a trajectory and target unitary operators applied at specific times t. However, for practical purposes, the dataset is generated numerically by computing the system dynamics. The states evolve according to Eq. (2.2), and the corresponding unitaries $U(t)$ are obtained from the Trotter product formula [Eq. (2.35)].

Employing the Magnus expansion or the Trotterization method increases the computational cost, as these calculations must be performed at each training iteration to update the model parameters. To mitigate this, the Magnus expansion is truncated at second order. For both the Magnus and Trotterization approaches, we use 50 integration steps per model query, i.e., each time the model produces an output.

Furthermore, a single QPT is insufficient to accurately characterize $U(t)$. In principle, achieving near-perfect fidelities over a continuous time domain would require an infinite number of QPTs due to its continuous nature. Nevertheless, models such as the one proposed here help to reduce the required measurement cost. The study focuses on the time interval $t \in [0,1]$, employing discrete time grids with step sizes of $\Delta t = 0.1, 0.25, 0.5$, and 1.0. This procedure yields a comprehensive dataset for the selected time values, essential for effective model training. A training dataset $\left\{\left(\rho_0^i, \tau_i, \rho_{\tau_i}^i, U_{\tau_i}\right)\right\}_{i=1}^{N_{\text{train}}}$, with $N_{\text{train}} = 11000$, was generated, where the target unitaries U_{τ_i} are known and the evolved states satisfy $\rho_{\tau_i}^i = U(\tau_i, 0)\,\rho_0^i\,U^\dagger(\tau_i, 0)$. The evolution times $\tau_i \in [0,1]$

are sampled from uniform time grids corresponding to different choices of the time step Δt. The initial states ρ_0^i are random mixed states sampled from the Hilbert–Schmidt measure [77]. A complex matrix G with independent Gaussian real and imaginary entries is drawn, and the state is obtained as $\rho_0 = GG^\dagger / \mathrm{Tr}\big(GG^\dagger\big)$, which is Hermitian, positive semi-definite, and normalized by construction.

The loss function is defined as an objective function whose minimization yields the optimal parameters of the deep learning model, such that the resulting model accurately describes the dynamics of the physical system under consideration. It is constructed to explicitly incorporate the known physical constraints governing the system dynamics, as well as the available supervision data. In particular, it enforces consistency with the unitary evolution of states (*Postulate 2*, Eq. (2.2)), agreement with known unitary operators when available, and the unitarity of the predicted time-evolution operator. The total loss function is defined as

$$\begin{aligned}\mathscr{L} = \frac{1}{N_{\text{train}}} \sum_{i=1}^{N_{\text{train}}} \Big(\Big\| \rho^i_{\tau_i} - U_\theta(\tau_i)\, \rho_0^i\, U_\theta^\dagger(\tau_i) \Big\| + \| U_\theta(\tau_i) - U_{\tau_i} \| \Big) \\ + \frac{1}{N_f} \sum_{k=1}^{N_f} \Big\| \mathbb{I} - U_\theta(t_k)\, U_\theta^\dagger(t_k) \Big\|, \end{aligned} \tag{4.6}$$

where $\|A\| = \sum_{ij} |a_{ij}|$ denotes the element-wise ℓ_1 matrix norm for any matrix $A = [a_{ij}]$, and $U_\theta(t)$ represents a neural network parametrization of the time-evolution operator with trainable weights $\theta = \{\theta_i\}$. As discussed in Section 3.6, Physics-Informed Neural Networks employ such neural representations to approximate unknown physical quantities while enforcing governing equations and constraints directly through the loss function [8].

Here, N_{train} denotes the number of available state–unitary training pairs, while N_f corresponds to the number of temporal collocation points used to enforce unitarity across the time domain. The first term of the loss enforces accurate quantum state evolution under the predicted unitary operator U_θ. The second term promotes consistency between the predicted unitaries and known target unitaries, when such information is available. The third term explicitly imposes the physical constraint of unitarity by penalizing deviations from $U_\theta(t)U_\theta^\dagger(t) = \mathbb{I}$ at arbitrary times.

For the 7- and 8-qubit systems, only the first two terms are employed. This choice is motivated by the fact that both the Magnus expansion and the Trotterization intrinsically preserve unitarity, rendering an explicit unitarity penalty unnecessary in these cases. During training, both methods use $r = 50$ integration steps. The results for 2 to 6 qubits are obtained with the full loss function of Eq. (4.6), except in Section 4.4.5, where the unitarity term is deliberately removed in order to quantify its contribution. That comparison shows that satisfactory performance is already achieved with a subset of the terms when the temporal sampling is dense. In this study, we focus on the most favorable training scenario, in which both the target unitary operators and the corresponding evolved quantum states are available during training.

4.3.3 Numerical integration of the effective Hamiltonian

For systems of 7 and 8 qubits, the network f_θ outputs the Pauli-basis coefficients of an effective Hamiltonian $H_{\text{eff}}(t)$, and a propagation layer maps it to the time-evolution operator. Two numerical methods are used for this purpose, both of which guarantee exact unitarity of the output by construction; their theoretical foundations were presented in Section 2.4, and here we describe their discrete numerical implementation.

Second-order Magnus expansion. Given a target time t and a number of integration steps r, a uniform time grid $\{t_k\}_{k=0}^{r-1}$ is constructed over $[0,t]$ with step size $\Delta t = t/r$. The neural network f_θ is evaluated at each grid point to obtain the sequence of Hamiltonians $\{H(t_k)\}_{k=0}^{r-1}$. The Magnus operator is truncated up to second order, $\Omega(t) \approx \Omega_1(t) + \Omega_2(t)$, with Ω_1 and Ω_2 as defined in Section 2.4, so that the factor $-i$ is already contained in Ω_1. The time-evolution operator is then recovered as

$$U(t) \approx \exp\left(\Omega_1(t) + \Omega_2(t)\right). \tag{4.7}$$

The full procedure is summarized in Algorithm 1.

Trotterization. Another path to compute the unitary from a given Hamiltonian is by using the Trotterization, described along the Eq. (2.35). Here, by discretizing the time interval into r subintervals of equal length

Algorithm 1 Second-order Magnus expansion

Require: Model f_θ, target time t, operator basis $\{\sigma_\mu\}$, number of steps r

Ensure: Unitary operator $U(t) \in \mathrm{SU}(d)$

$\Delta t \leftarrow t/r$
for $k = 0$ **to** $r-1$ **do**
  $t_k \leftarrow k\,\Delta t$
  $\boldsymbol{c}_k \leftarrow f_\theta(t_k)$ {model forward pass}
  $H(t_k) \leftarrow \sum_\mu c_k^\mu\, \sigma_\mu$ {reconstruct Hamiltonian}
end for
$\Omega_1 \leftarrow -i \sum_{k=0}^{r-1} H(t_k)\,\Delta t$
$\Omega_2 \leftarrow \mathbf{0}$
for $m = 0$ **to** $r-1$ **do**
  for $n = 0$ **to** $m-1$ **do**
    $\Omega_2 \leftarrow \Omega_2 - \frac{1}{2}[H(t_m), H(t_n)]\Delta t^2$
  end for
end for
$U(t) \leftarrow \exp(\Omega_1 + \Omega_2)$
return $U(t)$

$\Delta t = t/r$ and freezing the Hamiltonian within each slice, we are able to approximate a unitary matrix that, as $r \to \infty$, converges to the exact unitary that we are looking for. To improve accuracy over simple left-endpoint sampling, the Hamiltonian is evaluated at the midpoint of each subinterval,

$$t_k = \left(k + \frac{1}{2}\right)\frac{t}{r}, \quad k = 0, 1, \ldots, r-1. \tag{4.8}$$

Since each factor $\exp(-iH(t_k)\Delta t)$ is exactly unitary, their product is also exactly unitary regardless of the step size, which is the key practical advantage of this method. The full procedure is summarized in Algorithm 2.

Algorithm 2 Trotterization

Require: Model f_θ, target time t, operator basis $\{\sigma_\mu\}$, number of steps r

Ensure: Unitary operator $U(t) \in \mathrm{SU}(d)$

$\Delta t \leftarrow t/r$
$U \leftarrow \mathbf{I}_d$
for $k = 0$ **to** $r-1$ **do**
  $t_k \leftarrow \left(k + \frac{1}{2}\right)\Delta t$ {midpoint sampling}
  $\boldsymbol{c}_k \leftarrow f_\theta(t_k)$ {model forward pass}
  $H(t_k) \leftarrow \sum_\mu c_k^\mu \, \sigma_\mu$
  $U_k \leftarrow \exp\left(-iH(t_k)\,\Delta t\right)$
  $U \leftarrow U_k U$ {left-multiply to preserve time ordering}
end for
return U

In both methods, $r = 50$ integration steps are used during training and $r = 10$ steps at inference time, since the neural network outputs a smooth effective Hamiltonian for which high temporal resolution is not required at evaluation time.

4.3.4 Test and training

The AdamW optimizer is used for training, initialized with a learning rate of 10^{-3}. To enhance convergence stability, a learning-rate scheduler decreases the rate by a factor of 10^{-1} every 300 epochs. The training process spans 10^3 epochs, each consisting of 100 data batches with

a batch size of 10. Each dataset contains 11000 uniformly distributed tuples in the specified time intervals. In addition, the data batches used to train the models for 7- and 8-qubit systems are reduced to 10 data batches per epoch. All models presented in this chapter were trained using an NVIDIA Quadro P6000 GPU. For all model sizes, we observe that datasets with a larger Δt yield a more pronounced decrease in the training loss, since the data are less constrained and contain fewer distinct evolution times. We emphasize, however, that a lower training loss does not by itself imply better interpolation. Our goal is to predict the unitary at times not seen during training, and a more constrained dataset (smaller Δt) consistently yields better interpolation accuracy, as quantified by the test-set metrics reported in Section 4.4. The full training-loss curves for all system sizes and training configurations are available in the public repository accompanying this work.

4.4 Results

To assess the model's performance over the entire time domain we compute 100 target unitary matrices. These matrices are uniformly spaced in time and include values not present in the training set, thereby allowing us to rigorously evaluate the model's interpolation capability. To quantify accuracy, we adopt the gate fidelity [78] derived in Section 2.6 from the Choi–Jamiołkowski representation,

$$F(U(t), U_{\theta_{\text{opt}}}(t)) = \left| \frac{1}{d} \operatorname{Tr}\left[U^{\dagger}(t)\, U_{\theta_{\text{opt}}}(t) \right] \right|^2, \tag{4.9}$$

where θ_{opt} are the optimal weights of the neural network found during training, and $d = 2^N$ denotes the Hilbert-space dimension for an N-qubit system. The fidelity ranges from 0 to 1, with $F = 1$ only when the predicted operator $U_{\theta_{\text{opt}}}(t)$ coincides with the target $U(t)$ at time t, up to a global phase.

For evaluation, each trained model generates unitary matrices at the same 100 time points. These predicted matrices are then compared one-by-one with the target unitaries, yielding a fidelity function $F(t)$. Since multiple models for each system size are trained with different values of Δt, four fidelity curves $F(t)$ are obtained for each system size.

4.4.1 Gate fidelity across system sizes

The loss function additionally enforces unitarity across the time domain, ensuring that the model's predictions remain close to unitary transformations throughout training. To further minimize numerical deviations, we can compute the closest unitary matrix to each predicted output via Singular Value Decomposition (SVD), redefining the estimated unitary as

$$U_{\theta_{\text{opt}}}(t) = UV_{\text{h}}, \tag{4.10}$$

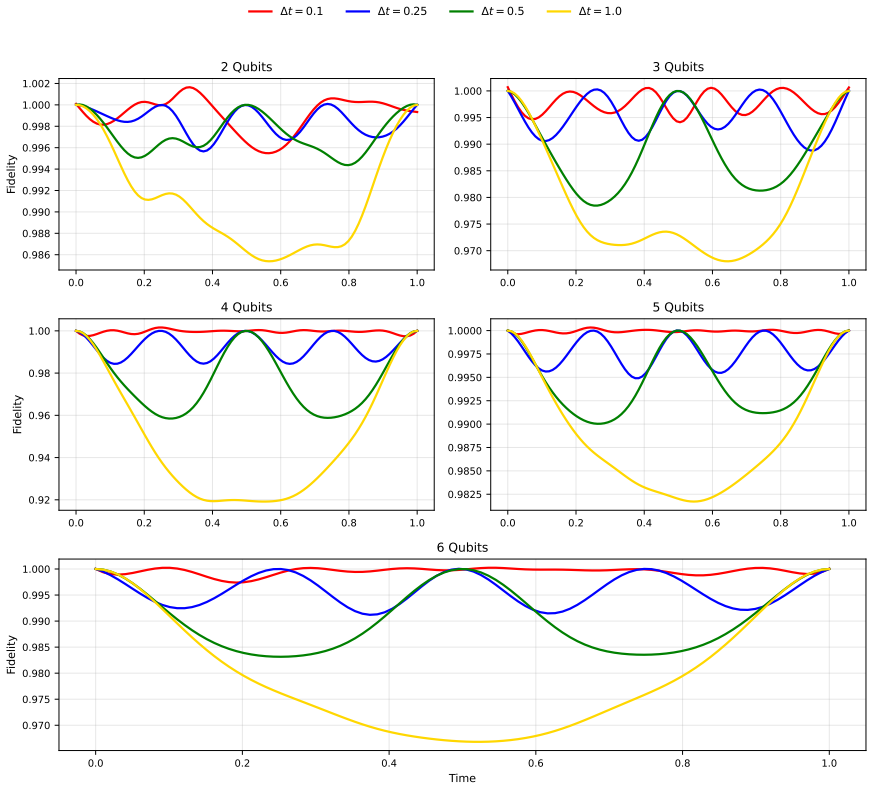
where U and V_{h} arise from the SVD of the model's predicted matrix. The following results compare the performance of the different models with and without post-processing, showing that high fidelity is preserved even in the absence of the closest-unitary correction.

The results in Fig. 4.3 summarize the fidelity performance for systems up to 6 qubits with and without the error correction via Eq. (4.10). As expected, datasets with smaller time steps (Δt) achieve the highest fidelities due to their finer temporal resolution. Nevertheless, even for larger Δt, the reduction in fidelity remains minimal. This reflects the PINN's strong interpolation capability, which maintains an accurate approximation of the quantum dynamics even with coarse temporal discretization.

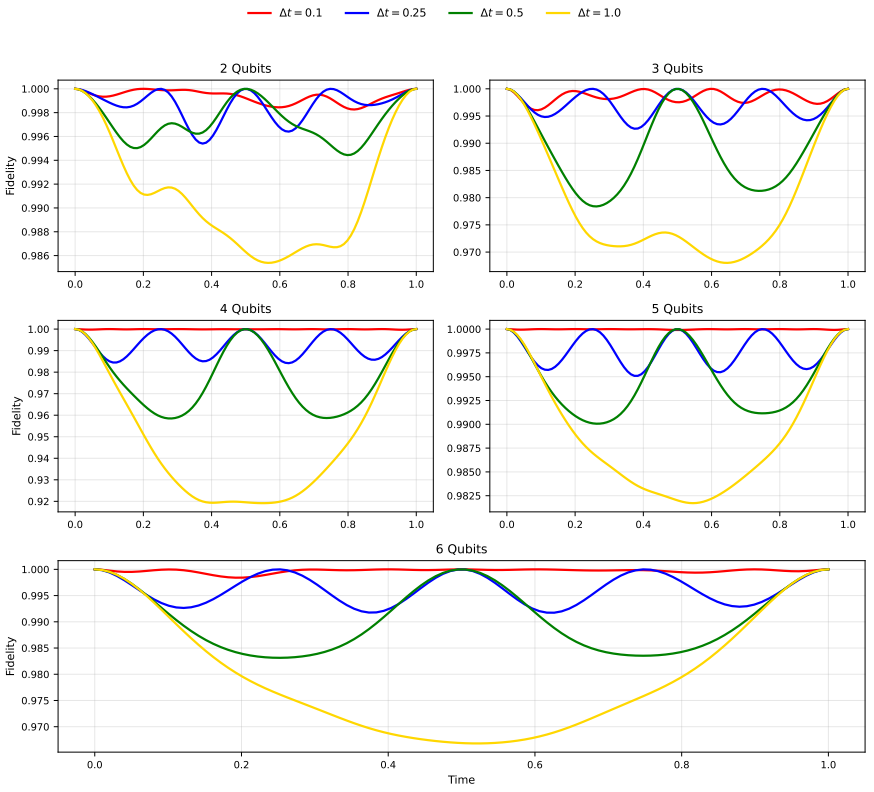
For the 7- and 8-qubit cases, the model follows a different strategy by estimating an effective Hamiltonian as an intermediate step to compute the corresponding unitary matrix. Since evolution can be achieved through Trotterization or Magnus expansion, both approaches were implemented and evaluated independently without SVD post-processing, as both methods guarantee unitarity. Figures 4.4 and 4.5 show the fidelity results for these higher-dimensional systems across different training datasets, for the Ising and XYZ Hamiltonians respectively. Consistent with the behavior observed in Fig. 4.3, even for large values of Δt, the degradation in fidelity is minimal, demonstrating that the interpolation remains highly effective in all tested scenarios.

4.4.2 Quantum state dynamics

To further validate the model's ability to reproduce quantum state dynamics, we evolve a randomly sampled initial state ρ_0 under two distinct time-evolution operators: (i) the exact unitary $U(t)$ computed via the second-order Magnus expansion, and (ii) the unitary $U_\theta(t)$ predicted by

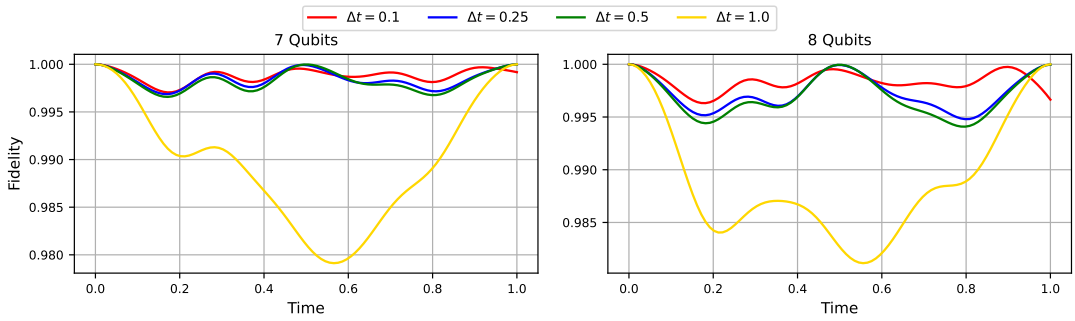


(a)

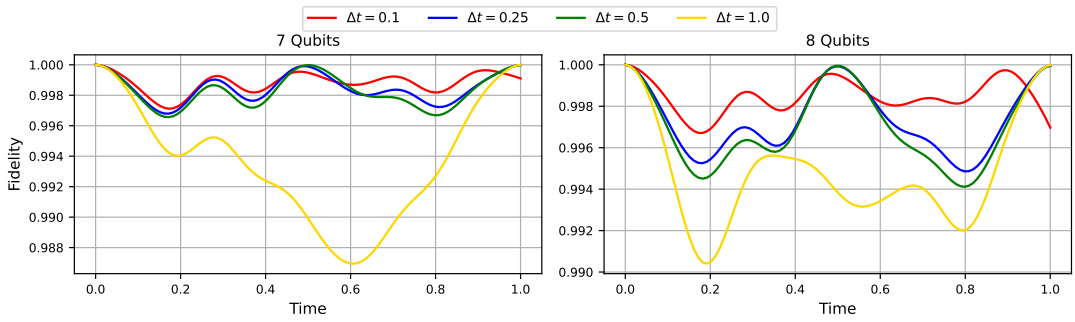


(b)

Figure 4.3: Fidelity of the predicted unitary evolution for *model2–model6*, shown (a) before and (b) after SVD-based post-processing error correction.

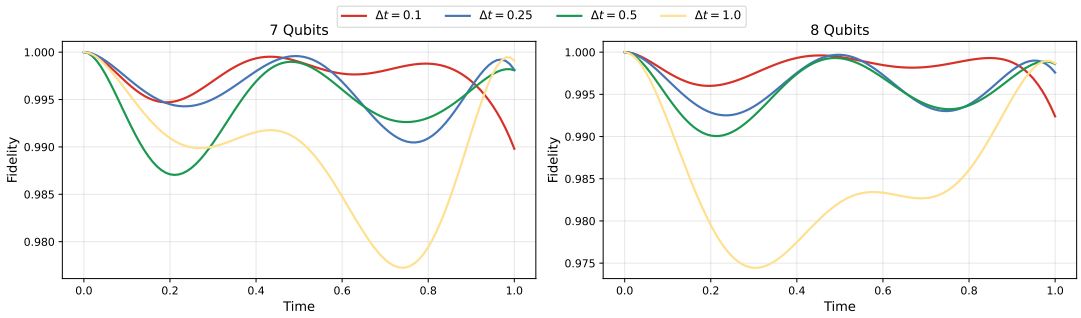


(a)

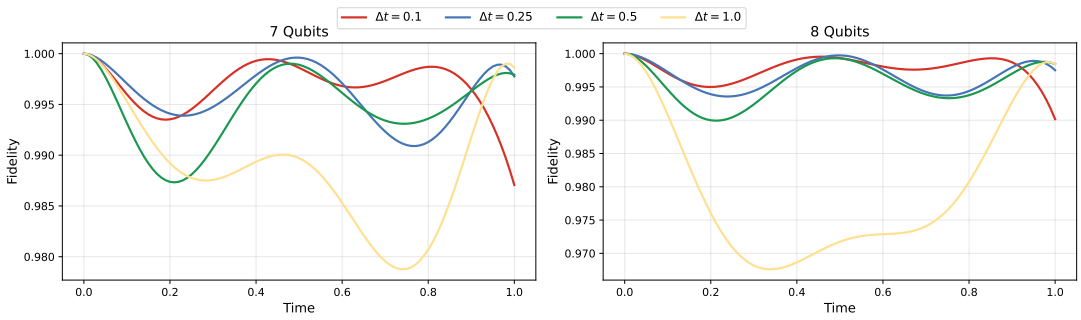


(b)

Figure 4.4: Fidelity performance for 7- and 8-qubit systems under the four training setups considered for the Ising Hamiltonian system. (a) Trotter-based model. (b) Magnus-based model.



(a)



(b)

Figure 4.5: Fidelity performance for 7- and 8-qubit systems under the four training setups considered for the XYZ Hamiltonian system. (a) Trotter-based model. (b) Magnus-based model.

the trained model without SVD correction. At each time $t \in [0,1]$, we compute the evolved density operator and track the diagonal elements $\rho_{ii}(t)$, which correspond to the occupation probabilities of the computational basis states. Figures 4.6 and 4.7 show these populations for systems ranging from 2 to 8 qubits, comparing the model predictions against the exact reference. In all cases, the trajectories are in close agreement, with only minor deviations consistent with the gate fidelity values reported in the preceding section.

4.4.3 Statistical analysis

To ensure that our evaluation did not rely on a single Hamiltonian realization per system size, we performed an additional statistical analysis to assess the model's robustness in interpolating unitary dynamics across diverse configurations. Three families of Hamiltonians were considered: the general Pauli Hamiltonian with all non-zero components (systems up to 6 qubits), the nearest-neighbor Ising model (7 and 8 qubits), and the nearest-neighbor XYZ Heisenberg model (7 and 8 qubits). For the general Pauli case, we carried out 100 independent training runs per system size. Due to computational resource constraints, this analysis was limited to 50 independent runs for the 7- and 8-qubit Ising Hamiltonian, 50 independent runs for the 7-qubit XYZ Hamiltonian, and 20 independent runs for the 8-qubit XYZ Hamiltonian. In each run, the amplitudes, frequencies, and phases defining the Hamiltonian were independently sampled from uniform distributions according to Section 4.2. This corresponds to $4^N - 1$ independent parameters for systems of up to 6 qubits, $2N - 1$ parameters for the Ising Hamiltonians, and $3(N-1)$ parameters for the XYZ Hamiltonians. These parameters were used to construct the target unitary evolutions, generate the corresponding training datasets, and train the model under identical conditions. Applying the same fidelity-evaluation pipeline yielded a collection of fidelity curves $F^{(k)}(t)$, with $k = 1, \ldots, N_r$, where N_r is the number of independent runs for each system size and Hamiltonian family. Each curve is obtained by evaluating the gate fidelity of Eq. (4.9) on a uniform grid of 100 points in $t \in [0,1]$, comparing the model prediction against the numerically computed target unitary. The mean fidelity $F_\mu(t)$ and the fidelity standard

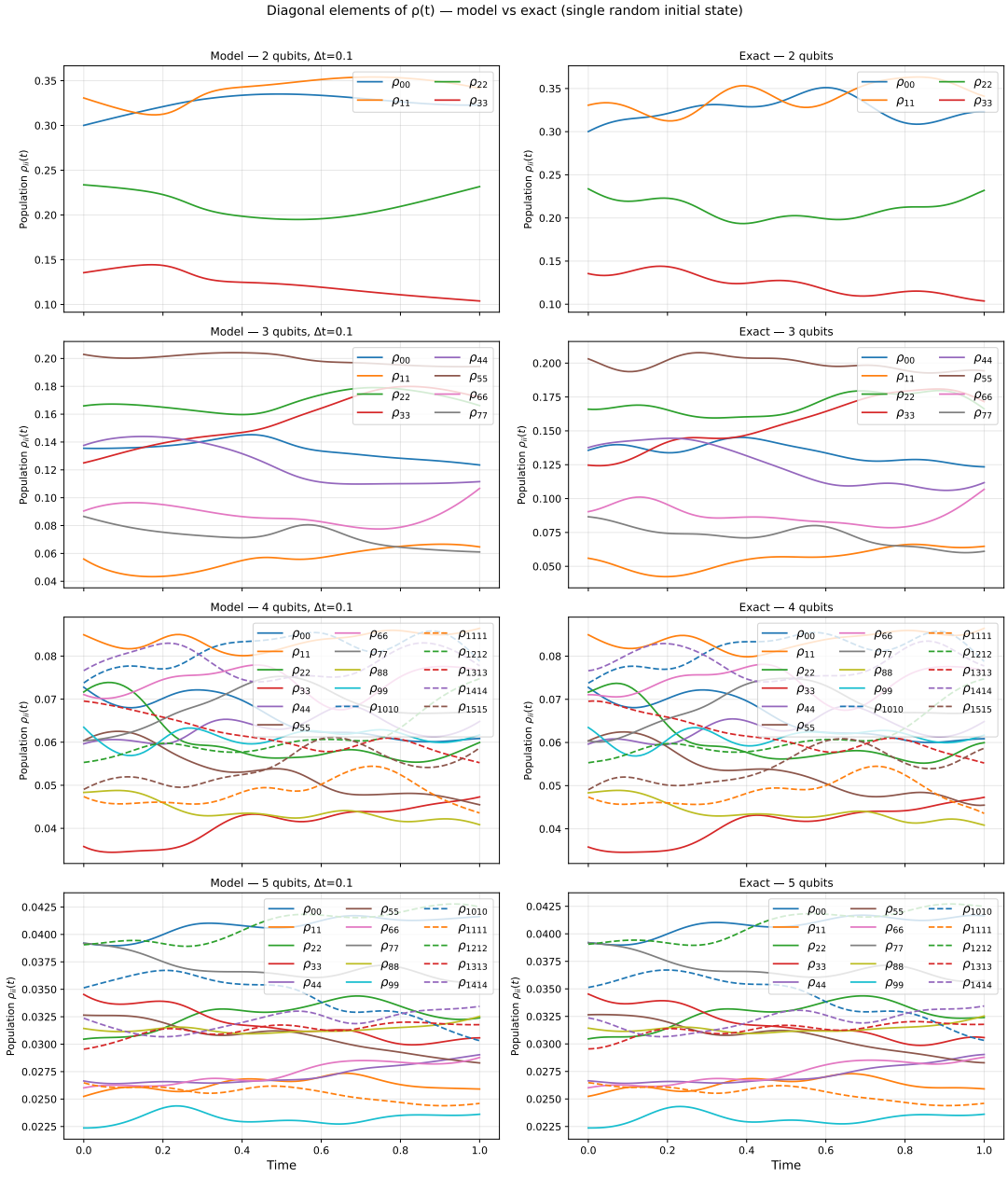


Figure 4.6: Diagonal elements $\rho_{ii}(t)$ of the evolved density operator for systems of 2 to 5 qubits, comparing the model prediction (left column) against the exact unitary evolution (right column). Each curve corresponds to a different basis state population, for a single randomly sampled initial state ρ_0 and training time step $\Delta t = 0.1$. The physical system considered is described by the general Pauli Hamiltonian $H_{\text{general}}(t)$, which includes all possible Pauli components in the N-qubit operator basis.

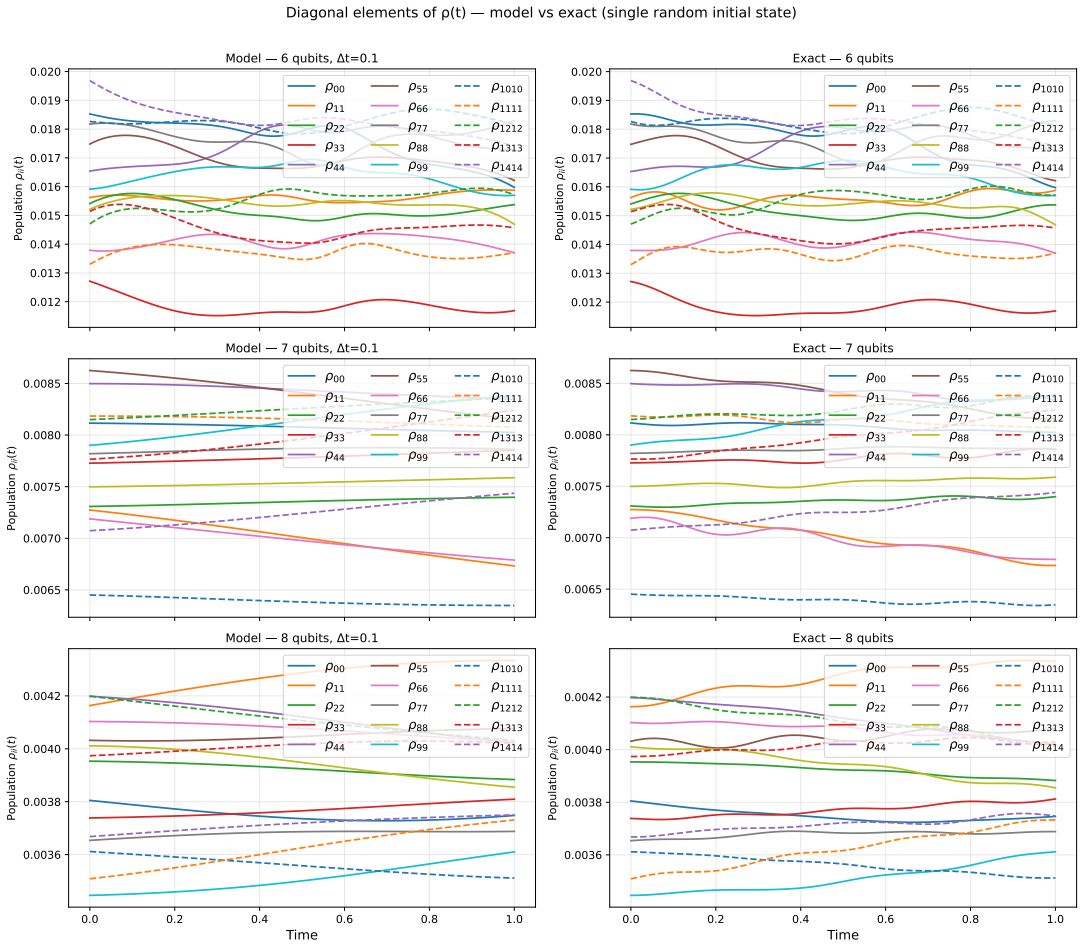


Figure 4.7: Same as Fig. 4.6 for systems of 6 to 8 qubits. For the 6-qubit case, the physical system is described by the general Pauli Hamiltonian $H_{\text{general}}(t)$. For the 7- and 8-qubit cases, the physical system is described by the nearest-neighbor Ising Hamiltonian $H_{\text{Ising}}(t)$, for which the model predicts the coefficients of an effective Hamiltonian $H_{\text{eff}}(t)$, from which the unitary is computed via the Magnus expansion.

deviation $F_\sigma(t)$ were computed as

$$F_\mu(t) = \frac{1}{N_r}\sum_{k=1}^{N_r} F^{(k)}(t), \tag{4.11}$$

$$F_\sigma(t) = \sqrt{\frac{1}{N_r - 1}\sum_{k=1}^{N_r}\left(F^{(k)}(t) - F_\mu(t)\right)^2}, \tag{4.12}$$

where $F^{(k)}(t)$ is the gate fidelity of the k-th run at time t.

Figures 4.8, 4.9, and 4.10 summarize the mean fidelity and standard deviation across all independent runs, providing a quantitative assessment of the variability in the model's performance. For each system size up to 6 qubits, two panels are shown: the first compares the raw model outputs with the target unitary evolutions, while the second reports the nearest-unitary matrices obtained through SVD-based post-processing. For 7- and 8-qubit systems, because unitarity is guaranteed by the Trotterization or Magnus expansion, SVD corrections are not required. Since these results closely match those obtained in the single-training analysis, we expect a similar statistical behavior for other values of Δt when applying the same procedure.

Across all analyses, the standard deviation remains consistently low. Even without enforcing unitarity via SVD, the variability is small, and after error correction the associated uncertainty becomes negligible to the point that it is not visually distinguishable in the figure. Moreover, no specific regions were identified where the model failed to interpolate successfully.

4.4.4 Trace-distance analysis

To extend our statistical analyses, we employ the trace distance as an additional figure of merit. For two density matrices ρ and σ, the trace distance is defined as

$$T(\rho,\sigma) = \frac{1}{2}\|\rho - \sigma\|_1 = \frac{1}{2}\sum_i |\lambda_i|, \tag{4.13}$$

where λ_i are the eigenvalues of the Hermitian operator $\rho - \sigma$, and $T \in [0,1]$. Following the same procedure as in the statistical analysis above, we evaluate the trace distance between the states evolved under the pre-

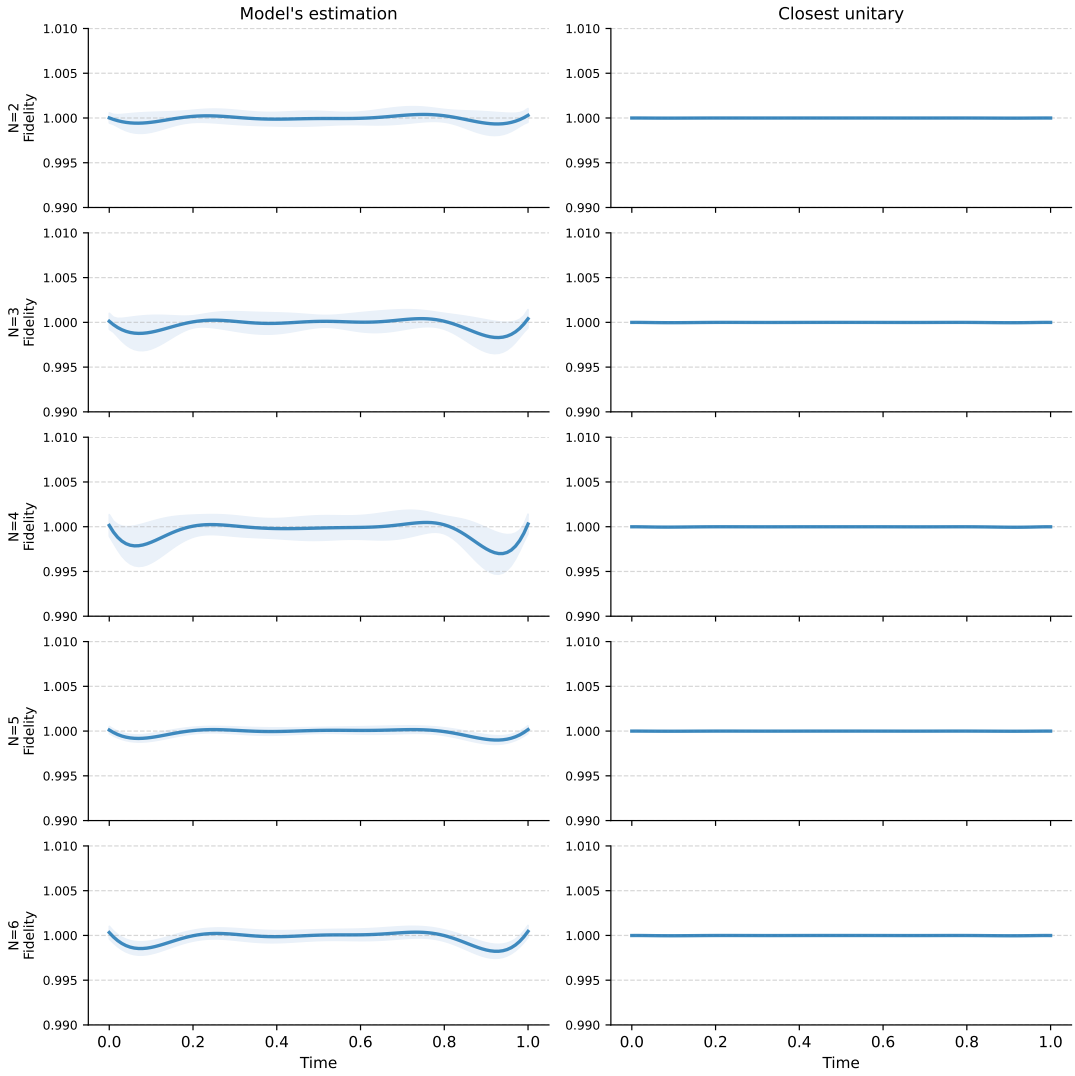


Figure 4.8: Mean fidelity and standard deviation across 100 independent training runs for systems up to 6 qubits. Left: raw model output vs. target unitary. Right: SVD-corrected output vs. target unitary.

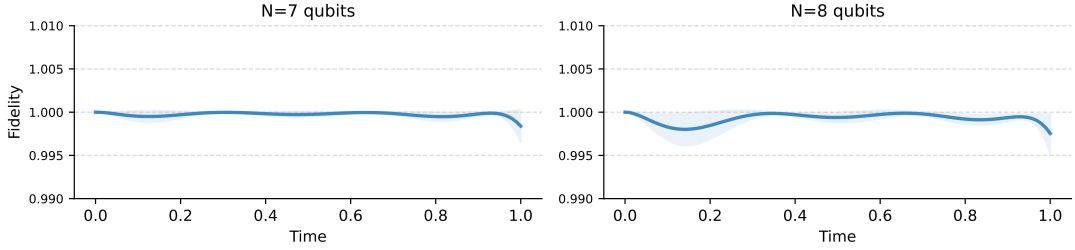


Figure 4.9: Same as Fig. 4.8 for 7- and 8-qubit systems under the nearest-neighbor Ising Hamiltonian (50 independent runs).

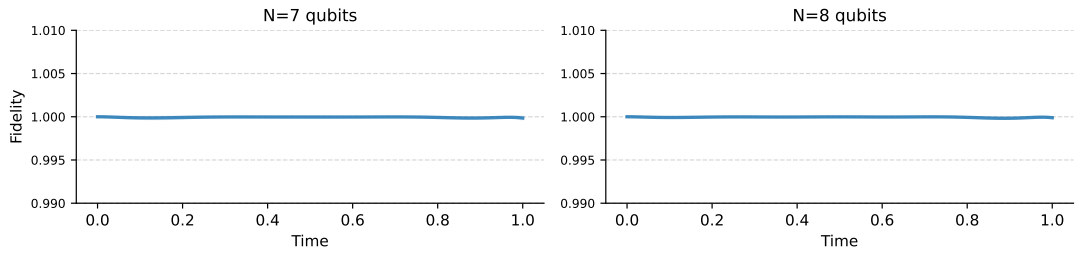


Figure 4.10: Same as Fig. 4.8 for 7- and 8-qubit systems under the nearest-neighbor XYZ Heisenberg Hamiltonian (50 runs for 7 qubits, 20 runs for 8 qubits).

dicted and exact unitaries over an ensemble of $N_s = 50$ random initial states, on a uniform grid of $N_t = 100$ points in $t \in [0,1]$. The initial states are sampled from the Hilbert–Schmidt measure, as described in Section 4.3.2.

For each qubit size, we compute the trace distances

$$T_i(t) = T(\rho_i(t), \sigma_i(t)), \tag{4.14}$$

where $\rho_i(t) = U_{\theta_{\text{opt}}}(t)\,\rho_0^i\,U_{\theta_{\text{opt}}}^\dagger(t)$ and $\sigma_i(t) = U(t)\,\rho_0^i\,U^\dagger(t)$ are the states evolved under the predicted and exact unitaries respectively, and $i = 1,\ldots,N_s$ labels the random initial states. The reported value corresponds to the mean trace distance,

$$T_\mu(t) = \frac{1}{N_s}\sum_{i=1}^{N_s} T_i(t), \tag{4.15}$$

while the error bars indicate one standard deviation,

$$T_\sigma(t) = \sqrt{\frac{1}{N_s-1}\sum_{i=1}^{N_s}\big[T_i(t) - T_\mu(t)\big]^2}. \tag{4.16}$$

Figure 4.11 summarizes the trace-distance statistics for models trained using a dataset with $\Delta t = 0.1$. The obtained values remain consistently of the order of $\mathcal{O}(10^{-2})$ across all qubit sizes, indicating a high degree of agreement between the states evolved under the predicted and exact dynamics. This is notable given that the trace distance is bounded in the interval $[0,1]$ and vanishes if and only if $\rho = \sigma$.

The ensemble variability is small across all sizes, becoming negligible at the scale of the plots for 6 qubits and larger. Taken together, the

results presented in this section show that the proposed models achieve both high average fidelity and low state-reconstruction error across the considered range of system sizes, highlighting their ability to accurately infer the underlying quantum dynamics.

4.4.5 Effect of the unitarity imposition term

In Section 4.3.2, we introduced the loss function employed during training together with a brief description of each contribution and its physical interpretation. The first two terms naturally incorporate the available knowledge of the system through supervised data constraints. In contrast, the third term introduces an additional soft constraint associated with the unitarity properties of the time-evolution operator, which may not always play a dominant role during the learning process.

Figures 4.12 and 4.13 present a statistical analysis of the impact of including or excluding the unitarity penalty term during training. The analysis was carried out through 25 independent training runs per system size, each considering a distinct realization of the fully non-zero Pauli Hamiltonian, with parameters sampled as described in Section 4.2. As in the preceding analyses, four values of the time grid were considered, $\Delta t \in \{0.1, 0.25, 0.5, 1.0\}$.

The figures compare the mean gate fidelity and its associated standard deviation across multiple independent realizations. The results show that the inclusion of the unitarity term becomes increasingly relevant as the amount of available training data decreases. In particular, for cases with sufficiently dense temporal sampling, such as $\Delta t = 0.1$, both approaches achieve fidelities close to unity over the entire time interval, indicating that the additional unitarity constraint does not provide improvements for the model. For $\Delta t = 0.25$, a small but still noticeable improvement can already be observed when the unitarity term is included.

However, for cases with more limited information, namely $\Delta t = 0.5$ and $\Delta t = 1.0$, the inclusion of the unitarity term significantly improves the overall performance of the model. In these regimes, the additional constraint not only increases the mean fidelity but also reduces the variance across independent realizations, leading to more stable and reliable predictions. This behavior is physically reasonable since, under reduced data availability, one of the few pieces of prior information that remains known with certainty is the unitary nature of the time-evolution opera-

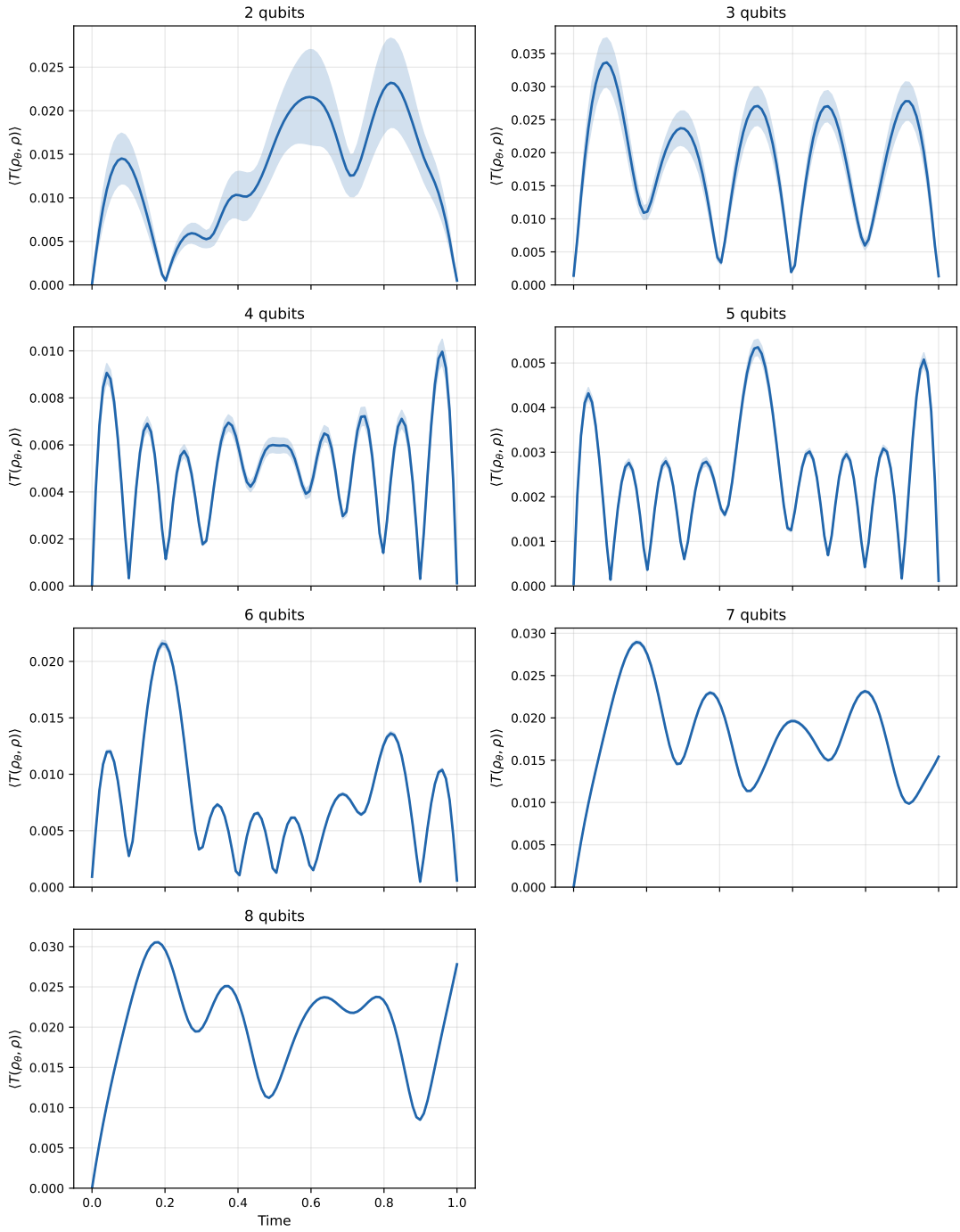
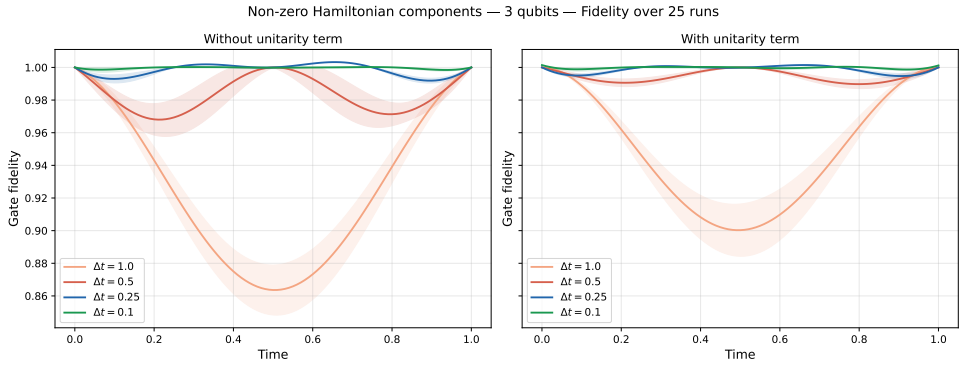


Figure 4.11: Mean trace distance between states evolved under the exact and predicted unitary operators for system sizes ranging from 2 to 8 qubits. The solid line shows the ensemble average over $N_s = 50$ random initial states at each time point, while the shaded region indicates one standard deviation. Trace distances remain of the order of 10^{-2} throughout the evolution, demonstrating the accuracy of the learned dynamics.

tor. Consequently, these results highlight the importance of incorporating physically motivated constraints into the loss function, particularly in low-data regimes.

(a)

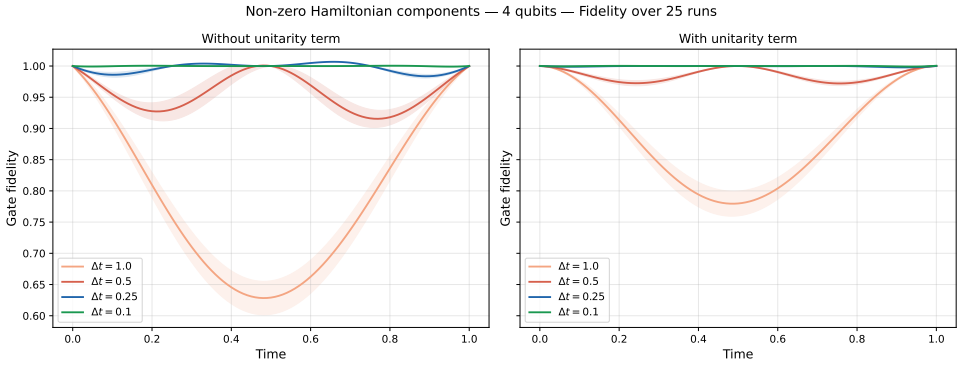
Statistical analysis for the $N = 2$ qubit case.



(b)

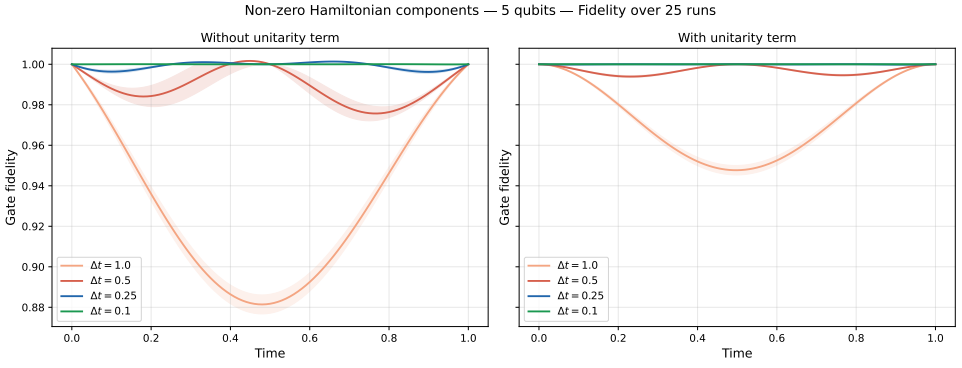
Statistical analysis for the $N = 3$ qubit case.

Figure 4.12: Statistical analysis of the effect of the unitarity imposition term in the loss function over 25 independent runs for different qubit system sizes. In each figure, the left panel shows the results obtained without the unitarity term, while the right panel includes the additional soft constraint associated with the unitary behavior of the time-evolution operator. Each curve represents the mean gate fidelity as a function of time for different temporal resolutions $\Delta t = 0.1, 0.25, 0.5$, and 1.0, while the shaded regions indicate the corresponding standard deviation across independent realizations with randomly generated Hamiltonian parameters.



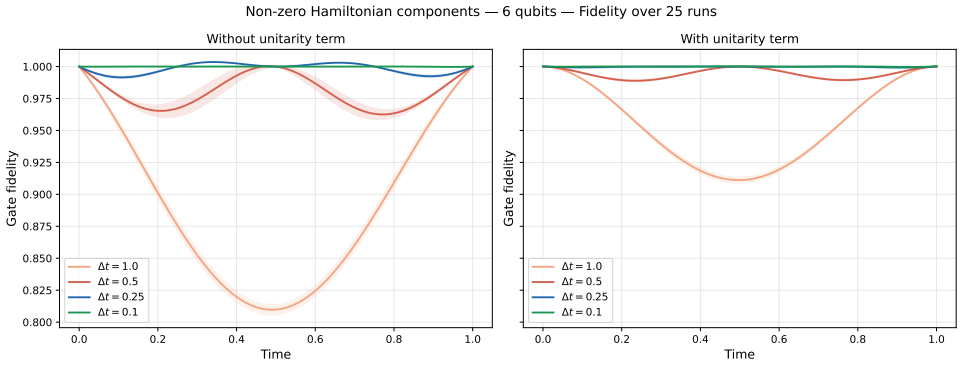
(c)

Statistical analysis for the $N = 4$ qubit case.



(d)

Statistical analysis for the $N = 5$ qubit case.



(e)

Statistical analysis for the $N = 6$ qubit case.

Figure 4.13: Statistical analysis of the effect of the unitarity imposition term in the loss function over 25 independent runs for different qubit system sizes (continuation of Fig. 4.12). In each figure, the left panel shows the results obtained without the unitarity term, while the right panel includes the additional soft constraint associated with the unitary behavior of the time-evolution operator.

4.5 Computational time benchmark

To assess the computational efficiency of the proposed framework, we compare the inference time of the trained PINN against the geodesic interpolation method introduced in Ref. [33], which serves as the numerical baseline throughout this work. Given two known unitary matrices $U_0 = U(t_0)$ and $U_1 = U(t_1)$ at consecutive time points, the geodesic interpolation constructs an intermediate unitary at time $t \in [t_0, t_1]$ as

$$U(t) = \left(U_1 U_0^\dagger\right)^\alpha U_0, \quad \alpha = \frac{t - t_0}{t_1 - t_0}, \tag{4.17}$$

where the matrix power is computed via `fractional_matrix_power` from `scipy.linalg`, which relies on a Schur decomposition and scales as $\mathcal{O}(d^3)$ with the matrix dimension $d = 2^N$.

The benchmark consists of measuring the wall-clock time per query for both methods over 500 independent repetitions per system size, with query times drawn uniformly from $[0, 1]$. For systems of 7 and 8 qubits, the PINN additionally requires a numerical integration step to compute $U(t)$ from the predicted effective Hamiltonian $H_{\text{eff}}(t)$, using either the Magnus expansion or the Trotterization. Importantly, while $r = 50$ integration steps are used during training, we find that reducing this to $r = 10$ steps at inference time does not lead to a significant loss of accuracy, since the network outputs a smooth effective Hamiltonian for which high temporal resolution is not required at evaluation time. This reduction further decreases the inference cost with respect to the training cost. In this section, all measurements for scenarios up to 5 qubits were performed on an AMD Ryzen 7 5800X, whereas those for cases of 6 qubits and above were performed on a conventional NVIDIA GeForce RTX 3060 GPU. We stress that this benchmark is not intended as a comparison of energy or hardware efficiency, but rather as a demonstration of the structural advantage of amortizing the training cost into a substantially lighter inference stage. This advantage is intrinsic to the method and does not depend critically on the specific hardware used, since the trained model and the reference methods, geodesic interpolation and SDP solvers, scale very differently with system size. In particular, the complexity gap between the $O(2^{3N})$ cost of the geodesic method and the direct evaluation of an already trained network is asymptotic in nature, and therefore remains qualitatively independent of the particular

hardware on which the empirical measurement is carried out. The result should thus be understood as a proof-of-principle regarding the relative scalability of the two approaches, rather than as a claim of absolute efficiency in a production environment.

The results are summarized in Figs. 4.14 and 4.15. For systems of 2 to 6 qubits, the PINN achieves speedups of 8.9×–19.3× over the geodesic interpolation across all system sizes, as shown in Fig. 4.14. The inference time of the PINN remains nearly constant as N increases, while the geodesic interpolation time grows with the matrix dimension, consistent with its $\mathcal{O}(d^3)$ scaling. For systems of 7 and 8 qubits, shown in Fig. 4.15, the Trotter-based variant achieves speedups of 6.8× and 17.8× respectively, while the Magnus-based variant achieves 2× and 7.8×, the latter being slower due to the additional cost of the nested commutator loop in the second-order correction. In all cases the speedup ratio exceeds unity, confirming that both PINN variants remain faster than the geodesic interpolation. This advantage is expected to grow further for larger systems where the $\mathcal{O}(d^3)$ cost of the geodesic interpolation becomes increasingly prohibitive as $d = 2^N$.

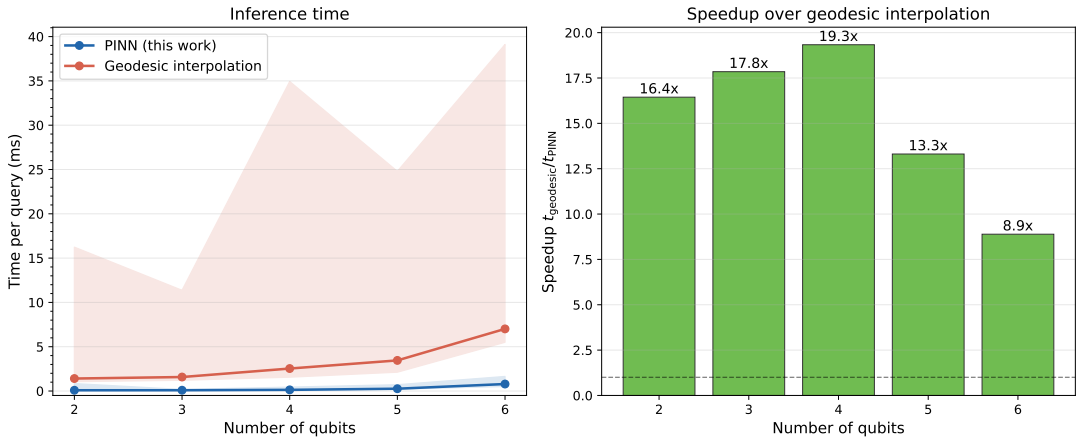


Figure 4.14: Timing benchmark for systems of 2 to 6 qubits. Left: wall-clock time per query for the PINN and the geodesic interpolation, averaged over 500 repetitions. The shaded region indicates the min/max case. Right: speedup ratio $t_{\text{geodesic}}/t_{\text{PINN}}$ for each system size. The dashed line indicates the break-even point at which both methods have equal cost.

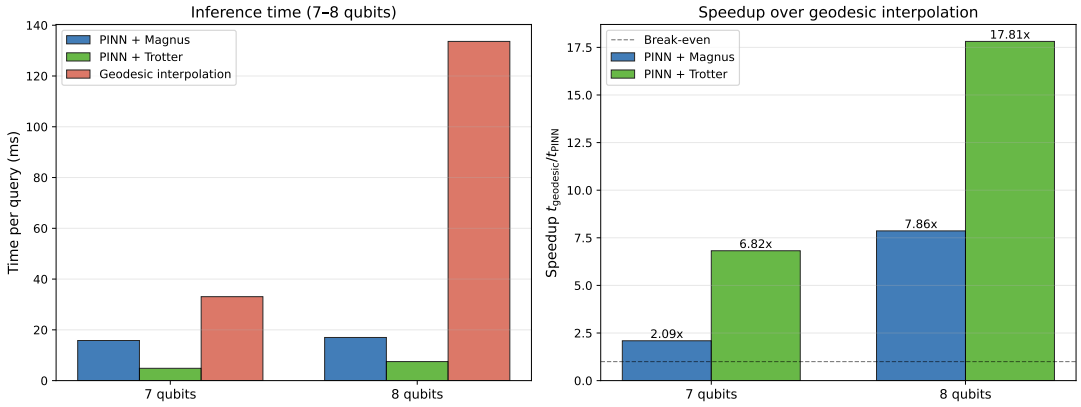


Figure 4.15: Timing benchmark for systems of 7 and 8 qubits. Left: wall-clock time per query for the PINN variants (Magnus and Trotter) and the geodesic interpolation, averaged over 500 repetitions with $r = 10$ integration steps at inference time. Right: speedup ratio $t_{\text{geodesic}}/t_{\text{PINN}}$ for each variant. The dashed line indicates the break-even point.

4.6 Summary

The results presented in this chapter show that PINNs can accurately interpolate the quantum dynamics encoded in the training data. Across all system sizes, the framework achieved mean gate fidelities close to 0.99 with low variability across independent Hamiltonian realizations, and models trained with coarse temporal sampling ($\Delta t = 1.0$) reproduced the dynamics almost as well as those trained with fine grids, confirming a strong interpolation capability that reduces the associated measurement cost. The trace-distance analysis confirmed accurate state reconstruction across the considered range of system sizes, and the timing benchmark showed inference times substantially below those of the geodesic-interpolation baseline [33], an advantage expected to grow with system size.

The framework also has well-defined limitations: a dedicated model must be trained for each Hamiltonian and system size, and the learned effective Hamiltonian for the 7- and 8-qubit cases captures the dynamics without coinciding with the true generator. These limitations, together with the broader implications of this work for large-scale quantum information and its connection to the marginal-reconstruction framework of Chapter 5, are discussed in the general conclusions of Chapter 6.

Chapter 5

MACHINE LEARNING APPROACH TO RECONSTRUCT DENSITY MATRICES FROM QUANTUM MARGINALS

This chapter presents the second main application developed in this thesis, a machine-learning-based approach to a central aspect of the Quantum Marginal Problem (QMP), namely the reconstruction of a global density operator compatible with a given set of quantum marginals. The results reported here correspond to the work published in *Machine Learning: Science and Technology* [51], developed in collaboration with Daniel Uzcategui-Contreras, Sebastian Niklitschek, and Aldo Delgado.

The method integrates a quantum marginal imposition technique with convolutional denoising autoencoders (CDAEs), with a loss function carefully designed to enforce essential physical constraints, including Hermiticity, positivity, and normalization. The theoretical ingredients were introduced in earlier chapters: the density-operator formalism, composite systems, the partial trace, and the quantum marginal problem in Chapter 2 (Sections 2.2 and 2.3), the fidelity between density matrices in Section 2.6, and convolutional neural networks and autoencoders (Section 3.3) and loss-function design (Section 3.4) in Chapter 3. The complete code for reproduction is available in a GitHub repository.[1]

5.1 Problem statement

As discussed in Section 2.3, the Quantum Marginal Problem arises as a fundamental challenge in quantum theory: given a set of reduced density matrices describing subsystems of a multipartite system, under what conditions are they consistent? Here, consistent means that there exists a global density operator, describing the entire system, whose reduced states coincide with the given marginals. This question naturally leads to several others: Can such a global state be uniquely determined? How

[1] https://github.com/guerra-antonio/Learning-QM

is entanglement reflected in the marginals? What is the minimal information required to reconstruct the global state?

The consistency problem of the QMP is a decision problem that was shown to be complete for the Quantum Merlin–Arthur (QMA) complexity class [41,42], highlighting its computational intractability in the general case. Solving the QMP would have important implications. For example, it could enable more efficient calculations of the ground states of local Hamiltonians [43]. The QMP also plays a central role in quantum chemistry, where it is referred to as the N-representability problem. Understanding the conditions under which reduced wave functions of electron pairs are compatible with an N-electron system could lead to more efficient calculations of binding energies [44]. However, the N-representability problem is also known to be QMA complete, reflecting its computational difficulty [43].

Previous works have addressed special cases of the QMP. For example, Klyachko provided the necessary and sufficient conditions for compatibility of 1-body marginals with a global pure state in the nonoverlapping case [45], while other studies have explored the uniqueness of pure global states determined by reduced marginals [46–48]. Additionally, symmetric instances of the QMP have been formulated as semidefinite programming (SDP) problems [49].

Machine learning approaches have been proposed for problems in quantum many-body physics [14–17]. Deep learning models have been shown to learn N-representability conditions from one- and two-electron systems and predict properties of larger systems [18–20]. In this chapter, we introduce a deep-learning-based approach to approximate solutions to the QMP. The method combines the Marginal Imposition Operator (MIO) [50], a technique designed to impose arbitrary quantum marginals on a matrix, with a CDAE architecture. This combination enables the construction of physically valid quantum states (i.e. Hermitian, positive semidefinite, and with trace one) that are compatible with a prescribed set of marginals, even in the presence of noise or initial inconsistency.

The architecture is capable of handling mixed quantum states for systems of 3 to 8 qubits. Additionally, we explore transfer learning between models trained on different numbers of qubits, showing that a model trained on 3-qubit systems can be adapted to systems with up to 8 qubits with limited retraining. This indicates that the network captures structural features of the compatibility conditions that are not specific to

a fixed Hilbert-space dimension.

5.2 The problem of compatibility

A quantum marginal $\sigma_{\mathscr{J}}$ is the reduced quantum state of a subsystem labeled by $\mathscr{J}$, derived from a larger system with global state $\rho_{\mathscr{J}}$. Let $\mathscr{I}$ denote a set of labels (e.g. $\mathscr{I} = 1,\ldots,N$) representing the components of an N-body physical system. The subset $\mathscr{J} \subset \mathscr{I}$ corresponds to the components of a reduced subsystem. When the global state $\rho_{\mathscr{J}}$ is known, the marginal state $\sigma_{\mathscr{J}}$ can be obtained via the partial trace operation introduced in Section 2.3 [cf. Eq. (2.15)] as

$$\sigma_{\mathscr{J}} = \mathrm{Tr}_{\mathscr{J}^c}[\rho_{\mathscr{I}}] = \sum_j \left(\langle j^c| \otimes \mathbb{I}_{\mathscr{J}}\right) \rho_{\mathscr{I}} \left(|j^c\rangle \otimes \mathbb{I}_{\mathscr{J}}\right), \tag{5.1}$$

where $\mathscr{J} \cup \mathscr{J}^c = \mathscr{I}$. In Eq. (5.1), the identity operator $\mathbb{I}_{\mathscr{J}}$ acts on the subsystem $\mathscr{J}$, while $\{|j^c\rangle\}$ is an orthonormal basis tracing the complement $\mathscr{J}^c$.

Let us consider now the opposite problem. That is, given a set of M quantum marginals $\{\sigma_{\mathscr{J}_\alpha}\}_{\alpha=1}^M$, we would like to know whether there is a global quantum state $\rho_{\mathscr{J}}$ such that $\sigma_{\mathscr{J}_\alpha} = \mathrm{Tr}_{\mathscr{J}_\alpha^c}[\rho_{\mathscr{J}}]$ for all $\alpha = 1,\ldots,M$. This problem can be stated as the SDP [79]

$$\begin{aligned} \text{maximize} \quad & \mathrm{Tr}(\rho_{\mathscr{J}}) \\ \text{subject to} \quad & \mathrm{Tr}_{\mathscr{J}_\alpha^c}[\rho_{\mathscr{J}}] = \sigma_{\mathscr{J}_\alpha}, \quad \alpha = 1,\ldots,M, \\ & \rho_{\mathscr{J}} = \rho_{\mathscr{J}}^\dagger, \\ & \rho_{\mathscr{J}} \succeq 0. \end{aligned} \tag{5.2}$$

There are algorithms, such as interior point methods, that are highly efficient for solving SDPs. However, the runtime and memory requirements of these methods scale with the size of the matrix and the number of constraints [80,81], making them impractical for large-scale instances of the problem. An iterative algorithm based on the MIO, introduced in the next subsection, was proposed in Ref. [50]. Nevertheless, determining the computational complexity of this approach remains a challenging task.

5.2.1 The Marginal Imposition Operator

The problem of determining a global quantum state from a given set of quantum marginals has been previously studied in Ref. [50], where the main tool introduced is the MIO. Let $\sigma_{\mathscr{J}}$ be a $|\mathscr{J}|$-party quantum state, with $\mathscr{J} \subset \mathscr{I}$. Then, the MIO is defined as follows:

$$\mathscr{Q}_{\mathscr{J}}(\tilde{\rho}_{\mathscr{J}}) := \tilde{\rho}_{\mathscr{J}} - \rho_{\mathscr{J}} \otimes \frac{\mathbb{I}_{\mathscr{J}^c}}{|\mathscr{J}^c|} + \sigma_{\mathscr{J}} \otimes \frac{\mathbb{I}_{\mathscr{J}^c}}{|\mathscr{J}^c|}, \tag{5.3}$$

where $\tilde{\rho}_{\mathscr{J}}$ is an $|\mathscr{I}|$-party quantum state, which can be chosen at random or in any convenient fashion, and $\rho_{\mathscr{J}} = \mathrm{Tr}_{\mathscr{J}^c}[\tilde{\rho}_{\mathscr{J}}]$. In Eq. (5.3) the symbol $|\mathscr{J}^c|$ denotes the dimension of the Hilbert space $\mathscr{H}_{\mathscr{J}^c}$, equal to 2^n for a complement of n qubits, so that $\mathbb{I}_{\mathscr{J}^c}/|\mathscr{J}^c|$ is the maximally mixed state on that space. Tensor factors are written in whichever order makes each expression readable, and since $\mathscr{J}$ need not consist of consecutive labels, the corresponding reordering of subsystems is left implicit.

It is easy to see that $\mathrm{Tr}_{\mathscr{J}^c}[\mathscr{Q}_{\mathscr{J}}(\tilde{\rho}_{\mathscr{J}})] = \sigma_{\mathscr{J}}$. In simple terms, the MIO enforces the marginal $\sigma_{\mathscr{J}}$ onto $\tilde{\rho}_{\mathscr{J}}$, although $\mathscr{Q}_{\mathscr{J}}(\tilde{\rho}_{\mathscr{J}})$ may not necessarily represent a valid quantum state. Despite this, the MIO exhibits several interesting properties:

(i) Hermiticity: $\mathscr{Q}_{\mathscr{J}}(\tilde{\rho}_{\mathscr{J}}) = \mathscr{Q}_{\mathscr{J}}(\tilde{\rho}_{\mathscr{J}})^{\dagger}$;

(ii) Trace preservation: $\mathrm{Tr}\big[\mathscr{Q}_{\mathscr{J}}(\tilde{\rho}_{\mathscr{J}})\big] = 1$.

More generally, for a given set $\{\sigma_{\mathscr{J}_\alpha}\}_{\alpha=1}^{M}$, the composition $\mathscr{Q}_{\mathscr{J}_M} \circ \cdots \circ \mathscr{Q}_{\mathscr{J}_1}(\tilde{\rho}_{\mathscr{J}})$ contains all the quantum marginals. That is,

$$\sigma_{\mathscr{J}_\alpha} = \mathrm{Tr}_{\mathscr{J}_\alpha^c}\left[\mathscr{Q}_{\mathscr{J}_M} \circ \cdots \circ \mathscr{Q}_{\mathscr{J}_1}(\tilde{\rho}_{\mathscr{J}})\right], \tag{5.4}$$

for all $\alpha = 1, \ldots, M$. An obvious condition for Eq. (5.4) to be satisfied is that for every pair $\sigma_{\mathscr{J}_\beta}, \sigma_{\mathscr{J}_\nu} \in \{\sigma_{\mathscr{J}_\alpha}\}_{\alpha=1}^{M}$, the quantum states $\sigma_{\mathscr{J}_\beta \cap \mathscr{J}_\nu}$ in the overlapping subsystems must coincide when $\mathscr{J}_\beta \cap \mathscr{J}_\nu \neq \emptyset$.

However, there are no guarantees that $\mathscr{Q}_{\mathscr{J}_M} \circ \cdots \circ \mathscr{Q}_{\mathscr{J}_1}(\tilde{\rho}_{\mathscr{J}}) \succeq 0$. Ensuring positivity may depend on the choice of $\tilde{\rho}_{\mathscr{J}}$, as numerically demonstrated in Ref. [50], among other factors. Although one could attempt to find the closest quantum state [82], this approach incurs the computational cost of an eigendecomposition. In the next section, we

introduce the CDAE, a machine learning model that offers a potential solution to this problem.

5.3 Model description

5.3.1 Density matrices as images

Machine learning has become a widely used tool for image processing [83], a field encompassing techniques designed to transform an image into another representation based on a specific objective. Examples of such transformations include resolution enhancement, noise reduction [84], image segmentation [85], and other feature-extraction methods. A central goal of image processing is to preserve essential information while enhancing the relevant features of the original image.

A color image can be represented as a set of three matrices corresponding to the RGB color channels, each with the same dimensions, containing a fixed number of pixels in both height and width. This establishes a direct and intuitive correspondence between images and matrices. Convolutional Neural Networks (CNNs), introduced in Section 3.3, are designed to process precisely this kind of multi-array data, and many state-of-the-art models for denoising, segmentation, and feature extraction are built upon CNN architectures, including autoencoders, U-Nets, variational autoencoders, and SegNet.

On the other hand, a fundamental object in quantum mechanics is the density operator (cf. Section 2.2), which encodes the information of a quantum system. These matrices consist of complex-valued elements and must satisfy specific mathematical properties: they must be Hermitian, positive semidefinite (PSD), and have unit trace. For a quantum system formed by N qubits, the dimension of the space of density matrices grows exponentially with N.

A density operator can be decomposed into its real and imaginary components, forming a multiarray, a set of two real-valued matrices. This suggests a natural analogy between density matrices and images, which in turn implies that image-processing techniques may be adapted to quantum data. If CNNs are effective at extracting spatial features from images, it is reasonable to explore their application in learning quantum states.

However, this relationship is not straightforward. The convolutional

layers of CNNs exhibit local connectivity, where each neuron is linked only to its neighboring units, in contrast with fully connected layers (cf. Section 3.3). This localized connectivity is particularly beneficial in image processing, as it allows the extraction of relevant features while preserving spatial relationships, without requiring a global representation of the entire image. Quantum density matrices do not inherently exhibit this locality-based structure. However, in the QMP we are primarily interested in global quantum states and their marginals, which are obtained through partial trace operations. This bears a conceptual similarity to the locality-based feature extraction performed by CNNs. While CNNs capture local structures within an image, partial traces extract relevant information from subsystems of a quantum state while maintaining a connection to the global system. This suggests that a CNN-based approach may be suitable for learning quantum states.

One of the main advantages of deep learning models for image processing is their scalability, as they are designed to handle images containing thousands or even millions of pixels. This scalability is particularly relevant in quantum mechanics, where the Hilbert space grows exponentially with the number of qubits. In this work, we present results for systems between 3 and 8 qubits. Training models for larger quantum systems requires significantly higher computational resources; nevertheless, all the necessary tools for the further development of such models are provided, facilitating their application to more complex quantum systems.

5.3.2 The convolutional denoising autoencoder

Autoencoders were introduced in Section 3.3 as neural networks that reconstruct their input through a compressed latent representation, using an encoder $\mathbf{h} = f(\mathbf{x})$ that maps the input $\mathbf{x}$ to a lower-dimensional latent code $\mathbf{h}$, and a decoder $\mathbf{r} = g(\mathbf{h})$ that reconstructs the data from it (cf. Fig. 3.5). *Denoising autoencoders* [86] are a variant trained to recover a clean signal $\mathbf{x}$ from a corrupted version $\tilde{\mathbf{x}}$ that still retains valuable information about $\mathbf{x}$; this is achieved by minimizing a loss that enforces the desired properties of $\mathbf{x}$ rather than exact reconstruction. In quantum mechanics, denoising autoencoders have been successfully applied to quantum state tomography [12, 87].

To address the quantum marginal compatibility problem introduced in Section 5.2, we propose a CDAE architecture. To represent density

matrices in a format suitable for the CDAE, we decompose them into two-channel tensors: one channel containing the real part and the other containing the imaginary part. This representation is analogous to how images are represented as multi-channel tensors, enabling the application of computer vision techniques such as CNNs [2] to address our problem.

For N-qubit systems, each channel corresponds to a $2^N \times 2^N$ matrix. The full architectural specifications of the implemented model are detailed in Section 5.7, and the proof that the architecture supports any N-qubit system with $N > 2$ is given in Section 5.7.1. Figure 5.1 illustrates the proposed CDAE architecture: the encoder, responsible for extracting the latent representation, comprises the layers to the left of the latent representation in the figure, while the decoder, tasked with reconstructing the original data from the latent code, consists of the layers to the right.

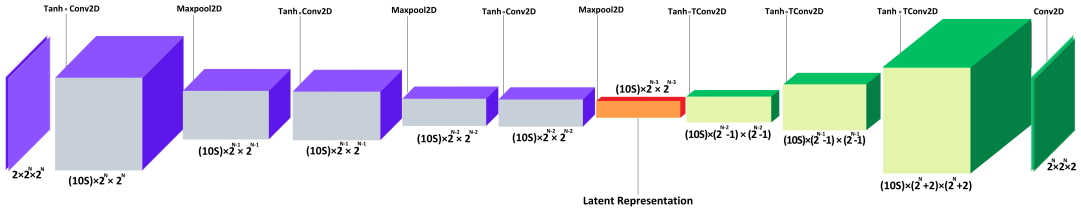


Figure 5.1: Illustration of the proposed CDAE architecture. The boxes represent tensors with dimensions specified as (number of channels) $\times$ height $\times$ width, indicated below each box. The operation applied to the tensor on the left is written above the arrow between two tensors, with the resulting tensor shown on the right. Since the Tanh activation function does not modify tensor dimensions, convolutional operations (Conv2D and TConv2D) are combined with the Tanh activation into single operations, denoted as $\text{Tanh} \circ \text{Conv2D}(\ldots)$ and $\text{Tanh} \circ \text{TConv2D}(\ldots)$. All boxes and operations to the left (right) of the latent representation correspond to the encoder (decoder). The leftmost box represents the input $\tilde{\mathbf{x}}$ of the CDAE, while the rightmost box represents its output $\mathbf{z}$.

5.4 Approach

Let $\tilde{\mathbf{x}}$ and $\mathbf{z}$ be the input and output of the CDAE, respectively, such that $\mathbf{z} = g(f(\tilde{\mathbf{x}}))$. The input $\tilde{\mathbf{x}}$ is the two-channel representation of the corrupted density operator $\tilde{X}$,

$$\tilde{X} = \mathcal{Q}_{\mathcal{J}_M} \circ \cdots \circ \mathcal{Q}_{\mathcal{J}_1}(\rho), \tag{5.5}$$

obtained by composing Eq. (5.3) for a given set $\{\sigma_{\mathscr{J}_\alpha}\}_{\alpha=1}^{M}$ of quantum marginals, with ρ serving as an initial seed state. The term "corrupted" refers to the presence of negative eigenvalues in the matrix $\tilde{X}$, which indicates that it is not a valid quantum state, although it still contains the quantum marginals, that is, $\sigma_{\mathscr{J}_\alpha} = \mathrm{Tr}_{\mathscr{J}_\alpha^c}[\tilde{X}]$ for all $\alpha = 1,\ldots,M$. The output $\mathbf{z}$ is also a two-channel tensor, and its matrix representation Z can be obtained by taking the first channel as the real part and the second channel as the imaginary part.

In addition, we explore an approach that involves using the output of the CDAE as an initial seed for the MIO. Since the MIO is capable of reconstructing global states that match the given marginals with perfect fidelities but often suffers from the issue of producing unphysical outputs, this hybrid strategy could mitigate the problem. By providing a well-structured initial guess from the trained autoencoder, the MIO may produce physically valid density matrices more reliably.

5.4.1 Data generation

The set $\{\tilde{\mathbf{x}}_\beta, \mathscr{M}_\beta\}_{\beta=1}^{N_S}$ represents the training set, with N_S denoting the number of samples, and $\mathscr{M}_\beta = \{\sigma_{\mathscr{J}_\alpha}\}_{\alpha=1}^{M}$. Each set $\mathscr{M}_\beta$ is obtained by calculating all k-body quantum marginals of an N-qubit density operator $\rho^{(\beta)}$, where $|\mathscr{M}_\beta| = \binom{N}{k}$. We choose $k = N-1$ to generate the data in all the cases presented in this work; however, the models trained in this way also work for $0 < k < N-1$, as demonstrated in the results section.

We consider rank-r density matrices $\rho^{(\beta)}$, generated according to the following procedure:

(i) Randomly generate an integer r from a uniform distribution in the interval $r_{\min} \leq r \leq 2^N$, where $r_{\min}$ is a pre-defined parameter.

(ii) Draw an $r \times r$ density matrix from the Hilbert–Schmidt (Ginibre) ensemble, rotate it with a Haar-random unitary, and take the diagonal of the result as the weight vector $\mathbf{p}^\beta = (p_1^\beta, \ldots, p_r^\beta)^{\mathrm{T}}$.

(iii) Sample r pure states $\left|\psi_i^\beta\right\rangle$ from the Haar measure.

(iv) Form the density operator $\rho^{(\beta)} = \sum_{i=1}^{r} p_i^\beta \left|\psi_i^\beta\right\rangle\!\left\langle\psi_i^\beta\right|$.

Two features of this sampling scheme are worth making explicit. Since the Hilbert–Schmidt ensemble is unitarily invariant, the rotation in step (ii) leaves the distribution unchanged, and the weights follow a Dirichlet distribution with all concentration parameters equal to r. They therefore concentrate around the flat vector $(1/r,\dots,1/r)$ as the rank grows, rather than covering the probability simplex uniformly. In addition, the states $\left|\psi_i^{\beta}\right\rangle$ are drawn independently and are not mutually orthogonal, so the weights p_i^{β} are not the eigenvalues of $\rho^{(\beta)}$; the operator nonetheless has rank r, since r Haar-random states are generically linearly independent.

Thus, we generate $\rho^{(\beta)}$ and then, using the partial trace $\sigma_{\mathscr{J}_\alpha} = \mathrm{Tr}_{\mathscr{J}_\alpha^c}\left[\rho^{(\beta)}\right]$, we obtain the elements of $\mathscr{M}_\beta$. With $\mathscr{M}_\beta$, we find $\tilde{X}_\beta$ using Eq. (5.5) and then its two-channel representation $\tilde{\mathbf{x}}_\beta$. This procedure is repeated N_S times to form the training set.

5.4.2 The loss function

To train the CDAE, we define a loss function that ensures that the output is a Hermitian PSD matrix and compatible with the given quantum marginals. Let Z be the matrix representation of the output $\mathbf{z}$ of the CDAE. We can decompose Z into its polar form as $Z = UP$, where U is a unitary matrix and P is a PSD matrix. The loss function is given by

$$\mathscr{L} = \varepsilon(\mathrm{Re}(U),\mathbb{I}) + \varepsilon(\mathrm{Im}(U),\mathbf{0}) + \sum_{i}^{|\mathscr{M}_\beta|} \varepsilon\left(\sigma_{\mathscr{J}_i}, z_{\mathscr{J}_i}\right), \tag{5.6}$$

where $\varepsilon(\cdot,\cdot)$ denotes the mean square error, $\mathbb{I}$ is the identity matrix, and $\mathbf{0}$ is the zero matrix. The first two terms in Eq. (5.6) drive U toward the identity, so that Z approaches the Hermitian PSD factor P, while the third term accounts for the quantum marginals, with $z_{\mathscr{J}_i} = \mathrm{Tr}_{\mathscr{J}_i^c}[Z]$ representing the output marginal. Although a term $\varepsilon(\mathrm{Tr}[Z],1)$ could be explicitly included in Eq. (5.6) to guarantee normalization, this constraint is approximately satisfied without the need to enforce it explicitly. The final output matrix can be renormalized as $Z/\mathrm{Tr}[Z]$ to ensure exact normalization. Additionally, in practice, Z is approximately Hermitian, which can be fixed by setting $Z = (Z+Z^\dagger)/2$, thus avoiding non-real eigenvalues.

5.4.3 Training and transfer learning

For training, the Adam optimizer was used with a learning rate of 10^{-4} and a batch size of 100 for all the cases presented. The model was implemented in PyTorch, and all numerical experiments were conducted on a workstation equipped with 125 GB of RAM, an NVIDIA GeForce RTX 4090 GPU, and an AMD Ryzen Threadripper 7980X 64-core CPU.

The architecture employed in this work demonstrates sufficient generalization capacity, successfully adapting to global quantum states ranging from 3 to 8 qubits without requiring significant structural modifications. As a result, there is no need to design new architectures or increase the model capacity for each case (e.g. by deepening the network or increasing the number of convolutional filters). This flexibility enables the use of transfer learning as a strategy for extending the model to systems with a higher number of qubits.

Transfer learning allows us to initialize training for a model targeting N qubits from a previously trained model with $N-1$ qubits, rather than training from scratch. Our results indicate that this approach substantially reduces both the training time and the amount of data required. During the transfer process, certain layers, typically the encoder, are frozen, i.e. their gradients are deactivated, and only selected layers (often in the decoder) are updated. We begin with a base model trained on 3-qubit global states. Once trained to an acceptable level of performance, achieving fidelities of approximately 0.98 for a random case, this model is used as a starting point for training a model for 4-qubit global states. In this case, it was sufficient to retrain only the final decoder layer, yielding fidelities of around 0.97 for a test case. This strategy is repeated to scale up the model. However, in some cases, retraining only the decoder was insufficient. For example, when transitioning from 4 to 5 qubits, retraining the full model was required to achieve average fidelities above 0.95 for randomly selected cases. Interestingly, for larger systems (6, 7, and 8 qubits), retraining only the decoder once again proved sufficient. In these cases, transfer learning from the immediately smaller model led to successful generalization, enabling high-fidelity global state reconstruction based on marginal inputs.

These findings suggest a notable property of the network. The encoder appears to capture the relevant marginal information in a way that is transferable across different system sizes. This behavior is consistent with the encoder having internalized structural features of the compati-

bility problem that do not depend on a fixed Hilbert-space dimension.

5.5 Results

In this section, we present the results of our experiments to evaluate the performance of the proposed CDAE in solving the task of finding a quantum state compatible with a given set of quantum marginals. We focus on two key metrics: the success rate and the accuracy. The success rate measures the fraction of outputs predicted by the model that are valid PSD matrices.

To assess the accuracy of the predicted quantum marginals, we use the mean fidelity, denoted F_{mean}, built upon the state fidelity introduced in Section 2.6. For each sample β in the test set, we compute the average fidelity F_β over all the marginals in that sample as

$$F_\beta = \frac{1}{|\mathscr{M}_\beta|} \sum_{i=1}^{|\mathscr{M}_\beta|} F\left(\sigma_{\mathscr{J}_i}, z_{\mathscr{J}_i}\right), \qquad F\left(\sigma_{\mathscr{J}_i}, z_{\mathscr{J}_i}\right) = \text{Tr}\left[\sqrt{\sqrt{\sigma_{\mathscr{J}_i}} z_{\mathscr{J}_i} \sqrt{\sigma_{\mathscr{J}_i}}}\right]^2, \tag{5.7}$$

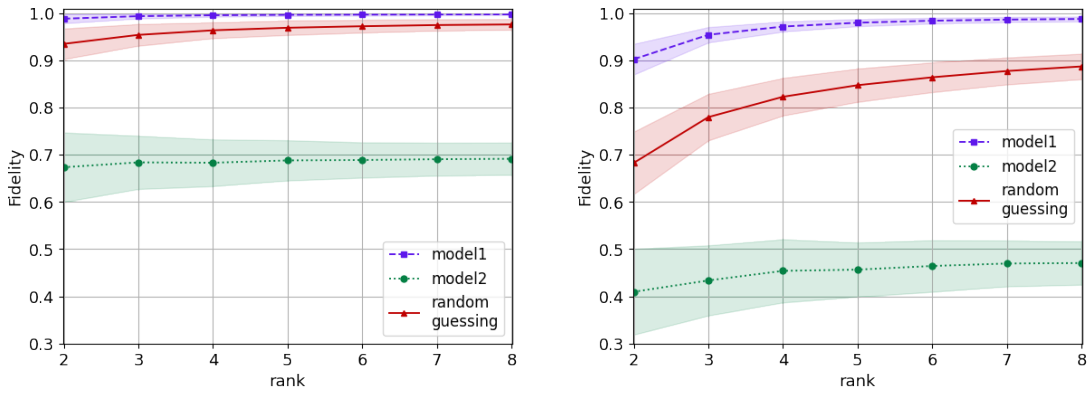
the fidelity between the target marginal $\sigma_{\mathscr{J}_i}$ and the predicted marginal $z_{\mathscr{J}_i}$, as given directly by Eq. (2.60), so that $F \in [0,1]$ and $F = 1$ if and only if the two marginals coincide, the same convention adopted for the gate fidelity in Chapter 2. The mean fidelity F_{mean} is obtained by averaging F_β over all samples in the test set, $F_{\text{mean}} = \sum_{\beta=1}^{N_{\text{test}}} F_\beta / N_{\text{test}}$, where N_{test} is the number of samples. To show variability in the data, we use the standard deviation (SD), calculated as

$$\text{SD} = \sqrt{\frac{1}{N_{\text{test}} - 1} \sum_{\beta=1}^{N_{\text{test}}} \left(F_{\text{mean}} - F_\beta\right)^2}. \tag{5.8}$$

We consider N-qubit global states with k-qubit marginals, denoted as $NXkY$ (e.g. $N3k2$ corresponds to 3-qubit global states with 2-qubit marginals). We label the model that uses the full loss function of Eq. (5.6) as *model1*, while *model2* uses Eq. (5.6) without the marginals term $\sum_i \varepsilon(\sigma_{\mathscr{J}_i}, z_{\mathscr{J}_i})$. The scaling parameter S of the architecture (see Section 5.7) is determined by experimentation, keeping the minimum value of S that produces satisfactory levels of accuracy and success rate.

To evaluate the performance of *model1* and *model2* in the case $N3k2$,

we set the scaling factor $S = 10$ and trained both models for 100 epochs using the same dataset of size 10^6 (see Section 5.4.1), with weights initialized from the same seed. We then evaluated the performance of these trained models in both the $N3k1$ and $N3k2$ cases. The values of $F_{\text{mean}} \pm \text{SD}$, calculated over 10^4 samples per rank, are shown in Fig. 5.2, for ranks up to 2^N. As expected, predictions from *model1* consistently achieve higher fidelity with target marginals compared to *model2* for all ranks, and both models achieve a high success rate. This demonstrates the importance of including the marginals term in the loss function. Moreover, predictions from *model1* outperform random guessing, which involves generating random N-qubit density matrices, calculating their k-qubit marginals, and computing the fidelity with the target marginals.



(a) $N3k1$ (b) $N3k2$

Figure 5.2: Mean and standard deviation (colored cloud) of the fidelity, obtained over 10^4 samples, versus rank for *model1* (purple squares), *model2* (green dots), and random guessing (red triangles) for the cases (a) $N3k1$ and (b) $N3k2$.

In Fig. 5.3, we compare *model1* to random guessing on systems with $N > 3$, up to $N = 8$ qubits, using the mean fidelity F_{mean} as a performance metric. The comparison is carried out for different values of k. We observe that *model1* consistently outperforms random guessing in all scenarios, with the performance gap increasing as k grows. This suggests that the model retains the information provided by the marginals with high accuracy. Furthermore, *model1* achieved high success rates for all cases shown in Fig. 5.3: above 95% for the case $N4k2$, and above 99% for the remaining cases, some of which reached 100%, as detailed in Fig. 5.4.

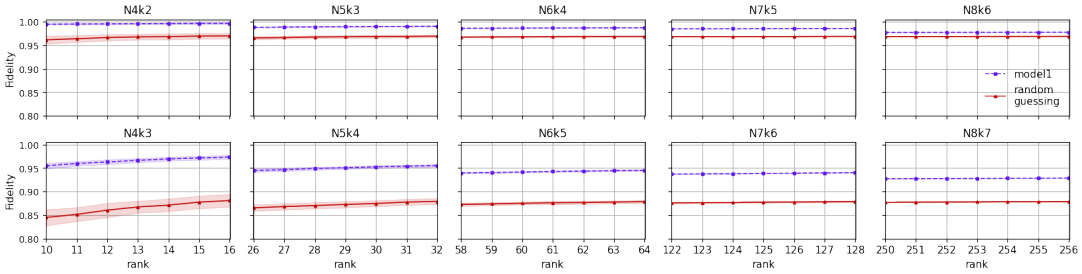


Figure 5.3: $F_{\text{mean}} \pm \text{SD}$, over 10^4 samples, as a function of rank for *model1* (purple dashed line) and for random guessing (solid red line) for the cases $N4k2$, $N4k3$, $N5k3$, $N5k4$, $N6k4$, $N6k5$, $N7k5$, $N7k6$, $N8k6$, and $N8k7$.

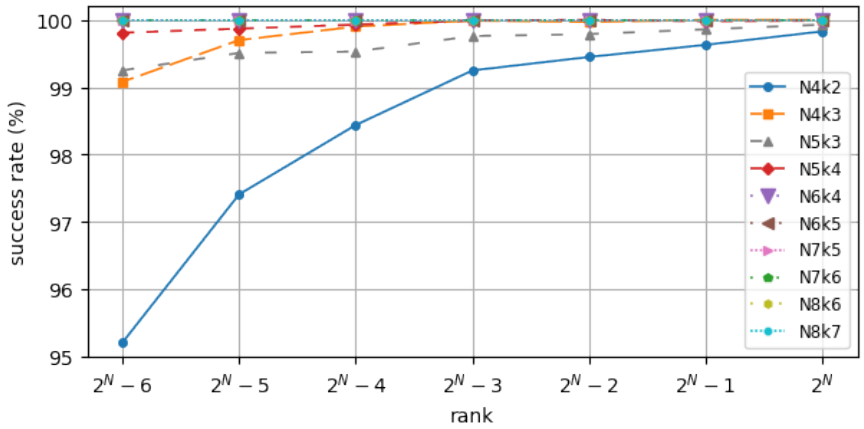


Figure 5.4: Success rates of *model1* for all the cases presented in Fig. 5.3. *model1* achieved a 100% success rate for all the cases with $N > 5$, with the lines overlapping at the 100% mark.

5.5.1 The hybrid *model1*+MIO scheme

Moreover, we tested a scheme in which the output of *model1* is passed through the MIO (*model1*+MIO). By first applying the MIO and then feeding its output into the CDAE, we obtain a global state that closely reproduces the target marginals with high fidelity. This resulting state can then serve as a new seed for a subsequent pass through the MIO, as given by Eq. (5.5), using the given target marginals $\{\sigma_{\mathscr{J}_\alpha}\}_{\alpha=1}^{M}$. Depending on the number k of qubits in the quantum marginals, some corrupted (i.e. negative) eigenvalues may still persist. However, we observed that, in many cases, the final output contains no corrupted eigenvalues, or at least fewer than those present after the initial MIO pass; see Fig. 5.5 for details. Additionally, the magnitude of these corrupted eigenvalues tends to decrease by approximately one order of magnitude.

In Fig. 5.6, we show results for cases in which the final output of this scheme (*model1*+MIO) is a valid quantum state whose marginals match the target marginals with perfect fidelity. Success rates of these cases are shown in Fig. 5.7. For systems with $N > 5$, all the instances were perfectly reconstructed, achieving a success rate of 100%, whereas for the cases $N4k2$ and $N5k2$ the success rate consistently increases with the rank. These results suggest that further improvements to the model, such as adding more layers, increasing filter sizes, or training on larger datasets, could lead the model to find exact solutions in all the cases.

We experimented with iterating the CDAE and MIO steps multiple times in cases where exact solutions were not initially found, but observed no significant improvement beyond the *model1*+MIO scheme. This suggests that a single iteration is sufficient to achieve this form of "purification". We hypothesize that this is because the remaining negative eigenvalues after the second MIO pass are too small to be identified by the CDAE as noise in subsequent iterations.

5.5.2 Computational time benchmark

In addition, we compare the performance of our model against state-of-the-art SDP solvers in terms of computation time. Figure 5.8 shows the average runtime computed over 10^3 samples for cases with $N < 6$, 100 samples for the case with $N = 6$, and 50 samples for cases with $N > 6$. In all cases, the target marginals were obtained from full-rank density matrices. As shown in Fig. 5.8, once trained, both *model1* and

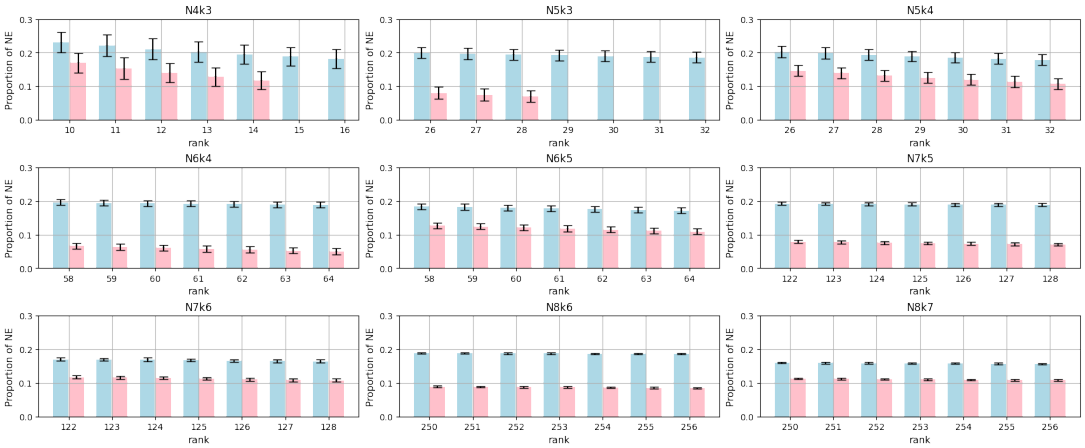


Figure 5.5: Comparison of the proportion of negative eigenvalues relative to the total number of eigenvalues. Each bar represents the average over a sample of 10^4 tests per rank, with error bars indicating the standard deviation. Light blue bars correspond to the first pass through the MIO, while pink bars correspond to the *model1*+MIO configuration.

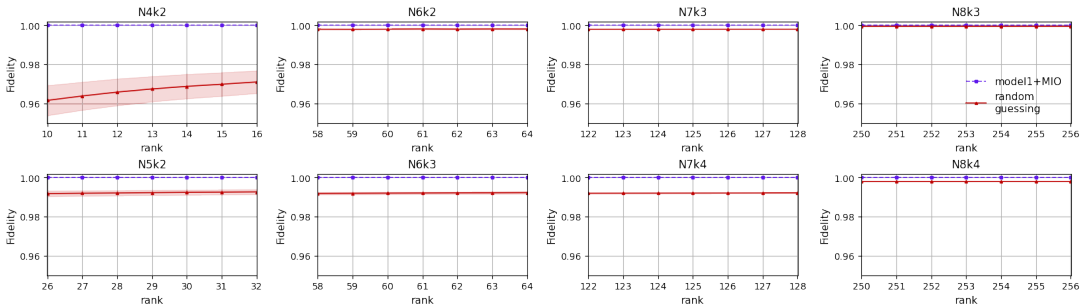


Figure 5.6: $F_{\text{mean}} \pm \text{SD}$, over 10^4 samples, as a function of rank for *model1* with an additional MIO pass (purple dashed line) and for random guessing (solid red line). Results are shown for the cases $N4k2$, $N5k2$, $N6k2$, $N6k3$, $N7k3$, $N7k4$, $N8k3$, and $N8k4$. All the states produced by the *model1*+MIO scheme match the target marginals with perfect fidelity.

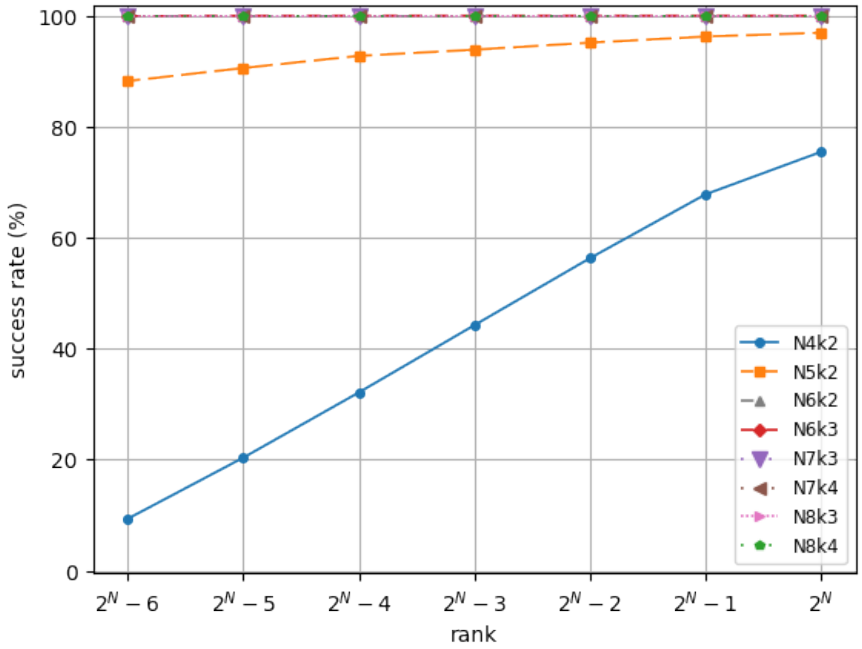


Figure 5.7: Success rates of the *model1*+MIO scheme for all the cases presented in Fig. 5.6. The scheme achieved a 100% success rate for all the cases with $N > 5$, with the lines overlapping at the 100% mark.

model1+MIO execute faster than the SDP solver provided by the CVXPY library in `Python`. While the SDP solver consistently found solutions matching the target marginals with perfect fidelity, it failed to run for the $N8k4$ case on our machine. In contrast, the *model1*+MIO scheme successfully found solutions with exact marginals for $N8k4$, with an average runtime of approximately 3.38 s.

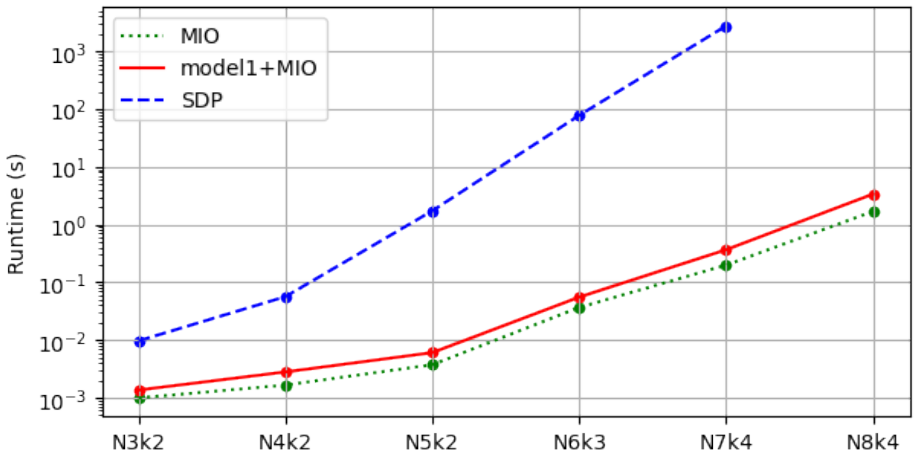


Figure 5.8: Average computation time of the CVXPY SDP solver (blue dashed line), *model1* (green dotted line), and *model1*+MIO (red solid line) across different system sizes. The vertical axis is in logarithmic scale. The SDP solver failed to run for the 8-qubit case.

5.6 Summary

We developed a CDAE and combined it with the MIO [50] to find a global N-qubit density operator compatible with a given set of quantum marginals. The model, which admits any N-qubit global system with more than 2 qubits, transforms, in most cases, the output of the MIO into a PSD matrix, preserving information about the quantum marginals with high accuracy. Transfer learning proved valuable, allowing the size of the training dataset to be reduced by leveraging knowledge learned in smaller-scale cases to improve performance on larger ones. The hybrid *model1*+MIO scheme reconstructed states whose marginals match the targets with perfect fidelity in many cases, and once trained, inference was significantly faster than state-of-the-art SDP solvers, even succeeding in regimes where the SDP solver failed to run.

A known limitation is that the model may struggle when the quantum marginals correspond to pure global states, whose global representation is unique; for mixed states, multiple representations exist, increasing the likelihood of finding an acceptable solution. The broader implications of this work for large-scale quantum information, its connection to the dynamics-interpolation framework of Chapter 4, and future directions, such as warm-starting SDP solvers, anomaly detection of incompatible marginals, and explainability of the learned representations, are discussed in the general conclusions of Chapter 6.

5.7 Model architecture

In this section, we describe the specifics of the proposed CDAE. The encoder consists of multiple layers, incorporating 2D convolution (Conv2D) operations, 2D max pooling (MaxPool2D) operations [88], and hyperbolic tangent (Tanh) activation functions. The decoder includes Conv2D operations and Tanh activation functions, along with layers utilizing 2D transposed convolutions (TConv2D).

Tables 5.1 and 5.2 provide detailed specifications of each layer in the encoder and decoder, respectively. All convolutional and transposed convolutional layers use a dilation factor of one. In the decoder, the output padding for transposed convolutions is fixed at zero. The intermediate number of channels in both encoder and decoder is controlled by a scaling factor S, which allows for flexible adjustment of the model's capacity. Lower values of S reduce the number of trainable parameters, offering a trade-off between model complexity and computational efficiency.

Layer	**Operation**	**Input ch.**	**Output ch.**	**Kernel**	**Stride**	**Padding**
(0)	Conv2D	2	$S \times 10$	3×3	1	1
(1)	Tanh	—	—	—	—	—
(2)	MaxPool2D	—	—	2×2	2	0
(3)	Conv2D	$S \times 10$	$S \times 10$	3×3	1	1
(4)	Tanh	—	—	—	—	—
(5)	MaxPool2D	—	—	2×2	2	0
(6)	Conv2D	$S \times 10$	$S \times 10$	3×3	1	1
(7)	Tanh	—	—	—	—	—
(8)	MaxPool2D	—	—	2×2	2	0

Table 5.1: Specifications for each encoder layer. Dilation is fixed to one throughout the network.

Layer	Operation	Input ch.	Output ch.	Kernel	Stride	Padding
(0)	TConv2D	$S \times 10$	$S \times 10$	3×3	2	1
(1)	Tanh	—	—	—	—	—
(2)	TConv2D	$S \times 10$	$S \times 10$	5×5	2	1
(3)	Tanh	—	—	—	—	—
(4)	TConv2D	$S \times 10$	$S \times 10$	6×6	2	0
(5)	Tanh	—	—	—	—	—
(6)	Conv2D	$S \times 10$	2	3×3	1	0

Table 5.2: Specifications for each decoder layer. Dilation is set to one, and output padding in transposed convolutions (TConv2D) is fixed at zero.

5.7.1 Compatibility of the CDAE with arbitrary system sizes

This subsection provides a detailed analysis of the channel sizes in the CDAE presented in Section 5.4. Each input to the CDAE is a two-channel tensor, where each channel is a $2^N \times 2^N$ matrix. We demonstrate that for $N > 2$, the channel size of the CDAE's output remains consistent at $2^N \times 2^N$. The size of a channel can change when Conv2D, MaxPool2D, and TConv2D operations are applied to tensors at different layers of the CDAE. The model considers a setting with square channels, square kernels, the same stride along both axes, and the same padding along both axes. Under these conditions, the channel size for Conv2D and MaxPool2D operations can be calculated using the following formula [88]:

$$C_{\text{out}(j)}^{E/D} = \left\lfloor \frac{C_{\text{in}(j)}^{E/D} + 2P - D(K-1) - 1}{S} + 1 \right\rfloor, \tag{5.9}$$

with $\lfloor \ldots \rfloor$ the floor function. Here, $C_{\text{in}(j)}^{E/D}$ and $C_{\text{out}(j)}^{E/D}$ are the input and output size, respectively, of a channel at layer (j) of the encoder (E) or decoder (D); P is the padding, D the dilation, K the kernel size, and S the stride. The dilation is set to $D = 1$ across the entire CDAE.

Note that $C_{\text{in}(l)}^{E/D} = C_{\text{out}(l-1)}^{E/D}$ for $l > 0$; that is, the input size of layer (l) is equal to the output size of the previous layer $(l-1)$. The input size

of the encoder is $C^E_{\text{in}(0)} = 2^N$. Using the values for P, D, K, and S given in Table 5.1, the convolutional layers (0), (3), and (6) leave the channel size unchanged, since

$$C^E_{\text{out}(0)} = \left\lfloor \frac{C^E_{\text{in}(0)} + 2\times 1 - 1\times(3-1) - 1}{1} + 1 \right\rfloor = \left\lfloor C^E_{\text{in}(0)} \right\rfloor = 2^N, \tag{5.10}$$

and the Tanh operations do not affect the channel size. Each MaxPool2D layer, namely layers (2), (5), and (8), halves the channel size,

$$C^E_{\text{out}(2)} = \left\lfloor \frac{C^E_{\text{out}(1)} + 2\times 0 - 1\times(2-1) - 1}{2} + 1 \right\rfloor = \left\lfloor 2^{-1} C^E_{\text{out}(1)} \right\rfloor = 2^{N-1}, \tag{5.11}$$

so that, applying the three convolution–pooling blocks in sequence, the output size of the encoder is

$$C^E_{\text{out}(8)} = 2^{N-3}. \tag{5.12}$$

Note that for $N < 3$ the encoder would render an invalid size, which establishes the requirement $N > 2$.

We now compute the channel sizes through the decoder using the values given in Table 5.2. For TConv2D operations, the channel size $D_{\text{out}(j)}$ changes according to [88]

$$D_{\text{out}(j)} = \Big(D_{\text{in}(j)} - 1\Big)S - 2P + D(K-1) + P_{\text{out}} + 1, \tag{5.13}$$

where P_{out} is the output padding, which is set to zero across the entire CDAE. As for the encoder, $D_{\text{in}(l)} = D_{\text{out}(l-1)}$ for $l > 0$, and the input size of the first decoder layer is $C^E_{\text{out}(8)} = 2^{N-3}$. Applying Eq. (5.13) successively with the parameters of Table 5.2 yields

$$D_{\text{out}(0)} = \Big(2^{N-3} - 1\Big)\times 2 - 2 + (3-1) + 1 = 2^{N-2} - 1, \tag{5.14}$$

$$D_{\text{out}(2)} = \Big(2^{N-2} - 1 - 1\Big)\times 2 - 2 + (5-1) + 1 = 2^{N-1} - 1, \tag{5.15}$$

$$D_{\text{out}(4)} = \Big(2^{N-1} - 1 - 1\Big)\times 2 - 0 + (6-1) + 1 = 2^{N} + 2, \tag{5.16}$$

where the Tanh layers in between leave the size unchanged. Finally, the

output size at layer (6) follows from Eq. (5.9):

$$C^{D}_{\text{out}(6)} = \left\lfloor \frac{D_{\text{out}(5)} + 2 \times 0 - (3-1) - 1}{1} + 1 \right\rfloor = \left\lfloor D_{\text{out}(5)} - 2 \right\rfloor = 2^N + 2 - 2 = 2^N, \tag{5.17}$$

where $D_{\text{out}(5)} = D_{\text{out}(4)}$. Thus, we have shown that the CDAE input and output sizes are equal for any $N > 2$, confirming that the same architecture supports arbitrary N-qubit systems beyond two qubits.

Chapter 6
CONCLUSIONS

This thesis explored the use of deep learning as a unifying methodological framework for the study of physical systems in quantum information. Two distinct but conceptually related problems were addressed: the interpolation of unitary time-evolution operators generated by time-dependent Hamiltonians (Chapter 4), and the reconstruction of global density matrices compatible with a prescribed set of quantum marginals (Chapter 5). In both cases, the central idea was the same. Rather than treating neural networks as opaque function approximators, we embedded the known physical structure of each problem directly into the architecture and the training objective, so that the learned solutions remained consistent with the underlying quantum mechanics. This physics-informed perspective, introduced in Chapter 3 and developed throughout the applications, is what gives both methods their accuracy, data efficiency, and physical interpretability.

6.1 Summary of the main results

The first application demonstrated that a Physics-Informed Neural Network can learn the full time evolution of a closed quantum system from a sparse set of sampled unitaries, interpolating the time-evolution operator $U(t)$ across the entire time domain without ever being given the Hamiltonian explicitly. By incorporating unitarity as a soft constraint and, for the larger systems, by predicting an effective Hamiltonian propagated through a second-order Magnus expansion or a Trotterization, the framework achieved mean gate fidelities close to 0.99 with low variability across independent Hamiltonian realizations, for systems ranging from 2 to 8 qubits. Models trained with coarse temporal sampling ($\Delta t = 1.0$) reproduced the dynamics almost as well as those trained with fine grids, confirming a strong interpolation capability that translates directly into a reduction of the measurement cost associated with quantum process tomography. Beyond accuracy, the trained models offered a computational

advantage at inference time, with speedups ranging from about $2\times$ to $19\times$ over a geodesic-interpolation baseline [33] that scales as $\mathcal{O}(2^{3N})$, depending on the system size and on the integration scheme used. This advantage is expected to grow with system size, making the approach particularly attractive in settings where the propagator must be evaluated repeatedly, such as dynamical quantum state tomography [89, 90].

The second application showed that the combination of a convolutional denoising autoencoder with the Marginal Imposition Operator [50] can construct physically valid global density matrices compatible with a given set of quantum marginals, a specific instance of the Quantum Marginal Problem, which is known to be QMA-complete in its general form [41, 42]. By encoding density matrices as two-channel images and designing a loss function that enforces Hermiticity, positivity, and normalization, the model preserved the target marginals with high fidelity for systems of 3 to 8 qubits. The hybrid *model1*+MIO scheme reconstructed states whose marginals matched the targets with perfect fidelity in many cases, and once trained, the inference was significantly faster than state-of-the-art semidefinite programming solvers, even succeeding in regimes where the SDP solver failed to run. The successful use of transfer learning across system sizes further suggested that the network internalizes structural features of the compatibility problem that are not tied to a fixed Hilbert-space dimension.

Taken together, these two results illustrate a common pattern. In both cases, a quantum-information task that is computationally demanding by conventional means, integrating a non-commuting time-dependent Hamiltonian or searching the exponentially large space of compatible global states, was recast as a learning problem in which physical constraints restrict the hypothesis space to physically admissible solutions. The neural networks did not merely fit data; they learned representations shaped by unitarity, positivity, and the partial-trace structure of composite systems, imposed exactly where the architecture allowed and as a penalty otherwise. This reflects the promise of physics-informed learning [8, 9], combining the flexibility and scalability of deep learning with the rigor of the governing physical laws.

6.2 Toward large-scale deep learning for quantum information

A recurring theme throughout this thesis is the exponential growth of the Hilbert space with the number of qubits, which renders exact classical methods intractable and motivates the search for scalable approximations [10]. Both frameworks developed here confront this challenge directly, and both point toward a broader role for deep learning in the study of large-scale quantum systems.

The interpolation framework addressed scalability by shifting, for the largest systems, from learning the full $2^N \times 2^N$ unitary to learning a compact effective Hamiltonian with only $\mathcal{O}(10^3)$ parameters, propagated through structure-preserving numerical integrators. This decoupling of the learned representation from the dimension of the operator it generates is a general design principle. The network learns the few physically relevant degrees of freedom, while the exponential structure is restored by a physics layer that guarantees unitarity by construction. The marginal-reconstruction framework, in turn, exploited the locality-based feature extraction of convolutional architectures and the transferability of learned representations across system sizes, suggesting that a single architecture can adapt to growing Hilbert spaces with limited retraining rather than full redesign.

These observations support a more general claim. Deep learning is not merely a faster numerical solver for isolated quantum-information tasks; it is a framework for learning the latent, low-dimensional structure that often underlies high-dimensional quantum problems. Once that structure is captured, predictions can be generated at arbitrary points, such as at any time in the evolution or for any rank of the global state, at a computational cost that grows far more slowly than that of the exact methods they approximate. As quantum platforms continue to scale, this capacity to amortize an expensive training stage into cheap, repeated inference is likely to become increasingly valuable, particularly in feedback-intensive settings such as quantum control [21, 25, 30] and adaptive measurement protocols.

6.3 Future work

The two frameworks presented here open several concrete directions for further research, both as direct extensions and as part of the broader program of applying deep learning to quantum information.

A natural extension of the interpolation framework is to move from reproducing the dynamics to explicitly reconstructing the Hamiltonian. In its current form, the effective Hamiltonian learned for the 7- and 8-qubit systems captures the dynamics without coinciding with the true generator. Incorporating differential constraints derived from the Schrödinger or von Neumann equation into the loss function, in the spirit of the original PINN formulation [8], could provide a pathway toward genuine Hamiltonian learning while retaining the flexibility of the data-driven approach. This would connect the present work more closely with existing Hamiltonian-reconstruction methods [37–39] and with operator-learning approaches that approximate the propagator directly [91, 92]. A second promising direction is the extension to open quantum systems [55], where the unitary constraint is replaced by complete positivity and trace preservation, and where the learned dynamical map could be combined with continuous-measurement protocols to further reduce experimental overhead. Finally, architectures capable of generalizing across multiple system sizes or across families of Hamiltonians, rather than requiring a dedicated model per configuration, would substantially broaden the practical applicability of the method, especially for systems with strong time dependence or chaotic behavior where larger architectures are expected to be necessary.

The marginal-reconstruction framework also admits several extensions. The transfer-learning behavior observed across system sizes invites a more systematic study of what the encoder learns, potentially through knowledge-extraction and explainability techniques [93], which could yield theoretical insight into the structure of the compatibility conditions themselves. The model's output could also serve as a warm-start initial guess for semidefinite programming solvers [79, 94], combining the speed of learned inference with the exactness of optimization-based methods. Furthermore, the autoencoder architecture is naturally suited to anomaly detection [95], suggesting that a related model could learn to distinguish compatible marginals from incompatible ones, addressing the decision version of the Quantum Marginal Problem rather than only

its constructive counterpart.

More broadly, the methods developed in this thesis fit within a rapidly expanding effort to apply machine learning across quantum information science. Neural networks have already been used for quantum state reconstruction [12, 87], for solving the quantum many-body problem [14–17], for learning N-representability conditions in quantum chemistry [18–20], and for quantum control through physics-informed networks [21]. The two contributions of this thesis, one operating on the level of quantum dynamics and the other on the level of quantum states, complement these efforts and reinforce a consistent conclusion. When the known physics of a problem is built into the learning process, deep learning becomes not a replacement for physical understanding but an extension of it, capable of tackling problems whose exact treatment lies beyond the reach of conventional computation. As both quantum hardware and machine-learning methods continue to mature, the synergy between them explored throughout this thesis is likely to remain a fertile ground for advancing our understanding and control of complex quantum systems.

BIBLIOGRAPHY

[1] David Silver, Aja Huang, Chris J. Maddison, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[2] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks, 2015.

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

[4] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv*, 2016.

[5] Amil Merchant, Simon Batzner, Samuel S. Schoenholz, et al. Scaling deep learning for materials discovery. *Nature*, 624:80–85, 2023.

[6] Remi Lam, Alvaro Sanchez-Gonzalez, Matthew Willson, et al. Learning skillful medium-range global weather forecasting. *Science*, 382(6677):1416–1421, 2023.

[7] Georgia Karagiorgi, Gregor Kasieczka, Scott Kravitz, Benjamin Nachman, and David Shih. Machine learning in the search for new fundamental physics. *Nature Reviews Physics*, 4:399–412, 2022.

[8] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[9] George Em Karniadakis, Ioannis G. Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3:422–440, 2021.

[10] Seth Lloyd. Universal quantum simulators. *Science*, 273(5278):1073–1078, 1996.

[11] Juan Carrasquilla, Giacomo Torlai, Roger G. Melko, and Leandro Aolita. Reconstructing quantum states with generative models. *Nature Machine Intelligence*, 1(3):155–161, 2019.

[12] Onur Danaci, Sanjaya Lohani, Brian T Kirby, and Ryan T Glasser. Machine learning pipeline for quantum state estimation with incomplete measurements. *Machine Learning: Science and Technology*, 2(3):035014, 2021.

[13] Tangyou Huang, Akshay Gaikwad, Ilya Moskalenko, et al. Quantum process tomography with digital twins of error matrices, 2025.

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

[15] Giuseppe Carleo, Ignacio Cirac, Kyle Cranmer, Laurent Daudet, Maria Schuld, Naftali Tishby, Leslie Vogt-Maranto, and Lenka Zdeborová. Machine learning and the physical sciences. *Rev. Mod. Phys.*, 91:045002, 2019.

[16] Hsin-Yuan Huang, Richard Kueng, Giacomo Torlai, Victor V. Albert, and John Preskill. Provably efficient machine learning for quantum many-body problems. *Science*, 377(6613):eabk3333, 2022.

[17] Borja Requena, Gorka Muñoz Gil, Maciej Lewenstein, Vedran Dunjko, and Jordi Tura. Certificates of quantum many-body properties assisted by machine learning. *Phys. Rev. Res.*, 5:013097, 2023.

[18] LeeAnn M. Sager-Smith and David A. Mazziotti. Reducing the quantum many-electron problem to two electrons with machine learning. *Journal of the American Chemical Society*, 144(41):18959–18966, 2022.

[19] Luis H. Delgado-Granados, LeeAnn M. Sager-Smith, Kristina Trifonova, and David A. Mazziotti. Machine learning of two-electron reduced density matrices for many-body problems. *The Journal of Physical Chemistry Letters*, 16(9):2231–2237, 2025.

[20] Xuecheng Shao, Lukas Paetow, Mark E. Tuckerman, and Michele Pavanello. Machine learning electronic structure methods based on the one-electron reduced density matrix. *Nature Communications*, 14:6281, 2023.

[21] Ariel Norambuena, Marios Mattheakis, Francisco J. González, and Raúl Coto. Physics-informed neural networks for quantum control. *Physical Review Letters*, 132(1):010801, 2024.

[22] S. Blanes, F. Casas, J.A. Oteo, and J. Ros. The magnus expansion and some of its applications. *Physics Reports*, 470(5):151–238, 2009.

[23] Nathan Wiebe, Dominic W. Berry, Peter Høyer, and Barry C. Sanders. Simulating quantum dynamics on a quantum computer. *Journal of Physics A: Mathematical and Theoretical*, 44(44):445308, 2011.

[24] Dong An, Di Fang, and Lin Lin. Time-dependent Hamiltonian simulation of highly oscillatory dynamics and superconvergence for Schrödinger equation. *Quantum*, 6:690, 2022.

[25] Q Ansel, E Dionis, F Arrouas, B Peaudecerf, S Guérin, D Guéry-Odelin, and D Sugny. Introduction to theoretical and experimental aspects of quantum optimal control. *Journal of Physics B: Atomic, Molecular and Optical Physics*, 57(13):133001, 2024.

[26] I. M. Georgescu, S. Ashhab, and Franco Nori. Quantum simulation. *Rev. Mod. Phys.*, 86:153–185, 2014.

[27] Yu Cao, Shi Jin, and Nana Liu. Quantum simulation for time-dependent hamiltonians—with applications to non-autonomous ordinary and partial differential equations. *Journal of Physics A: Mathematical and Theoretical*, 58(15):155304, 2025.

[28] Dominic W. Berry, Andrew M. Childs, Richard Cleve, Robin Kothari, and Rolando D. Somma. Simulating hamiltonian dynamics with a truncated taylor series. *Phys. Rev. Lett.*, 114:090502, 2015.

[29] Mária Kieferová, Artur Scherer, and Dominic W. Berry. Simulating the dynamics of time-dependent hamiltonians with a truncated dyson series. *Phys. Rev. A*, 99:042314, 2019.

[30] Mogens Dalgaard and Felix Motzoi. Fast, high precision dynamics in quantum optimal control theory. *Journal of Physics B: Atomic, Molecular and Optical Physics*, 55(8):085501, 2022.

[31] Guannan Chen, Mohammadali Foroozandeh, Chris Budd, and Pranav Singh. Quantum simulation of highly-oscillatory many-body Hamiltonians for near-term devices, 2023.

[32] Di Fang, Diyi Liu, and Rahul Sarkar. Time-dependent Hamiltonian simulation via Magnus expansion: Algorithm and superconvergence. *Communications in Mathematical Physics*, 2025.

[33] Michael Schilling, Francesco Preti, Matthias M. Müller, Tommaso Calarco, and Felix Motzoi. Exponentiation of parametric hamiltonians via unitary interpolation. *Phys. Rev. Res.*, 6:043278, 2024.

[34] Guifré Vidal. Efficient simulation of one-dimensional quantum many-body systems. *Physical Review Letters*, 93(4):040502, 2004.

[35] Conor Mc Keever and Michael Lubasch. Classically optimized Hamiltonian simulation. *Phys. Rev. Res.*, 5:023146, 2023.

[36] Conor Mc Keever and Michael Lubasch. Towards adiabatic quantum computing using compressed quantum circuits. *PRX Quantum*, 5:020362, 2024.

[37] Liangyu Che, Chao Wei, Yulei Huang, Dafa Zhao, Shunzhong Xue, Xinfang Nie, Jun Li, Dawei Lu, and Tao Xin. Learning quantum hamiltonians from single-qubit measurements. *Phys. Rev. Res.*, 3:023246, 2021.

[38] Chen-Di Han, Bryan Glaz, Mulugeta Haile, and Ying-Cheng Lai. Tomography of time-dependent quantum hamiltonians with machine learning. *Phys. Rev. A*, 104:062404, 2021.

[39] Giacomo Franceschetto et al. Hamiltonian learning via quantum Zeno effect, 2025.

[40] Antonio Guerra, Daniel Uzcategui-Contreras, Aldo Delgado, and Esteban S. Gómez. Interpolation of unitaries with time-dependent Hamiltonians via Deep Learning, 2026. Submitted to *Machine Learning: Science and Technology* (IOP Publishing).

[41] Yi-Kai Liu. Consistency of local density matrices is qma-complete. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 438–449. Springer Berlin Heidelberg, 2006.

[42] Anne Broadbent and Alex Bredariol Grilo. Qma-hardness of consistency of local density matrices with applications to quantum zero-knowledge. *SIAM Journal on Computing*, 51(4):1400–1450, 2022.

[43] Yi-Kai Liu, Matthias Christandl, and F. Verstraete. Quantum computational complexity of the *n*-representability problem: Qma complete. *Phys. Rev. Lett.*, 98:110503, 2007.

[44] Brian R. Judd. Reduced density matrices: Coulson's challenge. *Physics Today*, 54(2):58–59, 2001.

[45] Alexander Klyachko. Quantum marginal problem and representations of the symmetric group, 2004.

[46] N. Linden, S. Popescu, and W. K. Wootters. Almost every pure state of three qubits is completely determined by its two-particle reduced density matrices. *Phys. Rev. Lett.*, 89:207901, 2002.

[47] Lajos Diósi. Three-party pure quantum states are determined by two two-party reduced states. *Phys. Rev. A*, 70:010302, 2004.

[48] Nikolai Wyderka, Felix Huber, and Otfried Gühne. Almost all four-particle pure states are determined by their two-body marginals. *Phys. Rev. A*, 96:010102, 2017.

[49] Albert Aloy, Matteo Fadel, and Jordi Tura. The quantum marginal problem for symmetric states: applications to variational optimization, nonlocality and self-testing. *arXiv: Quantum Physics*, 2020.

[50] Daniel Uzcátegui-Contreras and Dardo Goyeneche. Reconstructing the whole from its parts, 2022.

[51] Daniel Uzcategui-Contreras, Antonio Guerra, Sebastian Nikl-itschek, and Aldo Delgado. Machine learning approach to reconstruct density matrices from quantum marginals. *Machine Learning: Science and Technology*, 6(2):025068, 2025.

[52] Gloria Devaud, María Teresa Erpelding, Lilian Kirsten, María Isabel Navarro, Myriam Ortega, and Myriam Vicente. *Álgebra Lineal*. Universidad de Concepción, Concepción, Chile, 2001. Facultad de Ciencias Físicas y Matemáticas.

[53] Sheldon Axler. *Linear Algebra Done Right*. Springer, 3 edition, 2015.

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

[55] Heinz-Peter Breuer and Francesco Petruccione. *The Theory of Open Quantum Systems*, pages 105–113. Oxford University Press, Oxford, UK, 2006.

[56] J. J. Sakurai. *Modern Quantum Mechanics*, pages 325–326. Addison-Wesley, Reading, USA, 1994.

[57] Hale F. Trotter. On the product of semi-groups of operators. *Proceedings of the American Mathematical Society*, 10(4):545–551, 1959.

[58] Richard Jozsa. Fidelity for mixed quantum states. *Journal of Modern Optics*, 41(12):2315–2323, 1994.

[59] John Watrous. *The Theory of Quantum Information*, pages 77–92. Cambridge University Press, Cambridge, UK, 2018.

[60] Marcus J. Neuer. *Machine Learning for Engineers: Introduction to Physics-Informed, Explainable Learning Methods for AI in Engineering Applications*. Springer, Cham, 2021.

[61] Jeffrey S. Bowers, Gaurav Malhotra, Marin Dujmović, Milton Llera Montero, Christian Tsvetkov, Valerio Biscione, Guillermo Puebla, Federico Adolfi, John E. Hummel, and Rachel F. Heaton.

Deep problems with neural network models of human vision. *Behavioral and Brain Sciences*, 46:e385, 2023.

[62] Lu Lu, Xuhui Meng, Zhiping Mao, and George Em Karniadakis. DeepXDE: A deep learning library for solving differential equations. *SIAM Review*, 63(1):208–228, 2021.

[63] Edward Small. An analysis of physics-informed neural networks. *arXiv*, 2023.

[64] Hwijae Son, Sung Woong Cho, and Hyung Ju Hwang. Enhanced physics-informed neural networks with augmented lagrangian relaxation method. *Neurocomputing*, 2023.

[65] Zhiyuan Ren, Shijie Zhou, Dong Liu, and Qihe Liu. Physics-informed neural networks: A review of methodological evolution, theoretical foundations, and interdisciplinary frontiers toward next-generation scientific computing. *Applied Sciences*, 15(14), 2025.

[66] Kini Bhanu Prakash and Karthika Nasir. Physics-guided machine learning is unlocking new capabilities in modeling complex systems. *arXiv*, 2025.

[67] Akira Sone and Paola Cappellaro. Hamiltonian identifiability assisted by a single-probe measurement. *Phys. Rev. A*, 95:022335, 2017.

[68] Jun Zhang and Mohan Sarovar. Quantum Hamiltonian identification from measurement time traces. *Phys. Rev. Lett.*, 113:080401, 2014.

[69] K. Kim, M.-S. Chang, S. Korenblit, R. Islam, E. E. Edwards, J. K. Freericks, G.-D. Lin, L.-M. Duan, and C. Monroe. Quantum simulation of the transverse Ising model with trapped ions. *New Journal of Physics*, 13:105003, 2011.

[70] Yang-Le Wu and S. Das Sarma. Understanding analog quantum simulation dynamics in coupled ion-trap qubits. *Physical Review A*, 93:022332, 2016.

[71] Tadashi Kadowaki and Hidetoshi Nishimori. Quantum annealing in the transverse Ising model. *Physical Review E*, 58:5355, 1998.

[72] R. Coldea et al. Spin waves and electronic interactions in La_2CuO_4. *Physical Review Letters*, 86:5377, 2001.

[73] K. Kim et al. Quantum simulation of spin models on an arbitrary lattice with trapped ions. *New Journal of Physics*, 14:095024, 2012.

[74] Tobias Graß and Maciej Lewenstein. Trapped-ion quantum simulation of tunable-range Heisenberg chains. *EPJ Quantum Technology*, 1:8, 2014.

[75] D. M. Zajac et al. Resonantly driven CNOT gate for electron spins. *Science*, 359:439–442, 2018.

[76] Jingfu Zhang, Gui Lu Long, Wei Zhang, Zhiwei Deng, Wenzhang Liu, and Zhiheng Lu. Simulation of Heisenberg XY interactions and realization of a perfect state transfer in molecular spin systems and liquid nuclear magnetic resonance. *Physical Review A*, 72:012331, 2005.

[77] Karol Życzkowski and Hans-Jürgen Sommers. Induced measures in the space of mixed quantum states. *Journal of Physics A: Mathematical and General*, 34(35):7111–7125, 2001.

[78] Renan Cabrera, Ofer M Shir, Rebing Wu, and Herschel Rabitz. Fidelity between unitary operators and the generation of robust gates against off-resonance perturbations. *Journal of Physics A: Mathematical and Theoretical*, 44(9):095302, feb 2011.

[79] Paul Skrzypczyk and Daniel Cavalcanti. *Semidefinite Programming in Quantum Information Science*. 2053-2563. IOP Publishing, 2023.

[80] Haotian Jiang, Tarun Kathuria, Yin Tat Lee, Swati Padmanabhan, and Zhao Song. A faster interior point method for semidefinite programming. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 910–918, 2020.

[81] Richard Y. Zhang and Javad Lavaei. Sparse semidefinite programs with near-linear time complexity. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 1624–1631, 2018.

[82] M Guţă, J Kahn, R Kueng, and J A Tropp. Fast state tomography with optimal error bounds. *Journal of Physics A: Mathematical and Theoretical*, 53(20):204001, 2020.

[83] Ge Wang, Yi Zhang, Xiaojing Ye, and Xuanqin Mou. *Machine Learning for Tomographic Imaging*. IOP Publishing, 2019.

[84] F. Scattarella, D. Diacono, et al. Deep learning approach for denoising low-snr correlation plenoptic images. *Sci Rep*, 13(19645), 2023.

[85] S. Suganyadevi, V. Seethalakshmi, and K. Balasamy. A review on deep learning in medical image analysis. *Int J Multimed Info Retr*, 11:19–38, 2022.

[86] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, pages 1096–1103, 2008.

[87] Lotfallahzadeh Barzili. *Quantum State Estimation and Tracking for Superconducting Processors Using Machine Learning*. PhD thesis, Chapman University, 2021.

[88] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning, 2018.

[89] Karthik Siva, Gerwin Koolstra, John Steinmetz, William P. Livingston, Debmalya Das, L. Chen, J.M. Kreikebaum, N.J. Stevenson, C. Jünger, D.I. Santiago, I. Siddiqi, and A.N. Jordan. Time-dependent hamiltonian reconstruction using continuous weak measurements. *PRX Quantum*, 4(4):040324, 2023.

[90] Marco Peruzzo, Tommaso Grigoletto, and Francesco Ticozzi. Reconstructing quantum states and expectations via dynamical tomography. *Physical Review A*, 113:032435, 2026.

[91] Karan Shah and Attila Cangi. Machine learning time propagators for time-dependent density functional theory simulations. *Machine Learning: Science and Technology*, 7(3):035019, 2026.

[92] Jun Yang and James D. Whitfield. Machine-learning Kohn–Sham potential from dynamics in time-dependent Kohn–Sham systems. *Machine Learning: Science and Technology*, 4(3):035022, 2023.

[93] Simon Odense and Artur d'Avila Garcez. Layerwise knowledge extraction from deep convolutional networks, 2020.

[94] Rico Angell and Andrew McCallum. Fast, scalable, warm-start semidefinite programming with spectral bundling and sketching, 2024.

[95] Mohammad Saiful Islam and Andriy Miranskyy. Anomaly detection in cloud components. In *2020 IEEE 13th International Conference on Cloud Computing (CLOUD)*, pages 1–3, 2020.