MSc Artificial Intelligence
Master Thesis

Physics-Informed Deep Learning Architectures for Quantum Many-Body System Analysis

by
Gerard Planella Fontanillas
14244950

July 31, 2024

48EC
Periods 2 - 6

Supervisor:
Prof. Max Welling
Second Supervisor:
Asst. Prof. Vikas Garg
Examiner:
Prof. Max Welling

Second reader:
Floor Eijkelboom

Universiteit van Amsterdam

Contents

1 Introduction **1**

2 Background Knowledge **3**
2.1 Quantum States . . . 3
2.1.1 Wavefunctions and the Hilbert Space . . . 3
2.1.2 Bloch Sphere . . . 4
2.1.3 Partial Trace . . . 4
2.1.4 Properties of Quantum States . . . 4
2.1.5 Density Operators . . . 4
2.1.6 Hamiltonian . . . 5
2.2 Ising Model . . . 6
2.3 Tensor Networks . . . 6
2.3.1 Tensor Network Diagrams . . . 7
2.3.2 Tensor Network Types . . . 7
2.3.3 Tensor Network Contraction . . . 8
2.3.4 Tensor Network Algorithms . . . 9
2.4 Graph Neural Networks . . . 10
2.5 Belief Propagation . . . 10
2.5.1 Vanilla Belief Propagation . . . 10
2.5.2 Tensor Network Belief Propagation . . . 11

3 Related Works **13**

4 Methodology **16**
4.1 Loss Function . . . 16
4.2 Architectures . . . 17
4.2.1 Physics-inspired GNNs . . . 17
4.2.2 TN-based Approaches . . . 22

5 Experimental Setup **28**
5.1 Datasets . . . 28
5.2 Baselines . . . 29
5.3 Training Procedure . . . 29
5.4 Experiments . . . 30

6 Results & Analysis **32**
6.1 Baseline Evaluation on PEPS and MPS Datasets . . . 32
6.2 Supervised Model depth in the 10×1 MPS Dataset . . . 32
6.3 Unsupervised QBP Evaluation . . . 34
6.4 Supervised Evaluation of QGNNs on Multiple Datasets . . . 35
6.5 Semi-supervised QBP-EBlochGNN Evaluation . . . 37

7 Conclusion **39**
7.1 Limitations 40
7.2 Future Research 40

A Code Base **41**

Acknowledgements

I want to extend my deepest gratitude to my supervisors, Prof. Max Welling and Asst. Prof. Vikas Garg, for their invaluable guidance and support throughout this project. Their expertise and encouragement have been significant in developing the necessary knowledge for this research. I also want to thank Adrian Müller and Floor Eijkelboom for their support and assistance during this thesis. This project would not have been possible without their contributions and encouragement.

"*La energía sigue al pensamiento*"
- J.C

Abstract

Quantum many-body systems pose significant computational challenges due to the exponential growth of their state spaces with system size. Traditional methods, such as Tensor Network (TNs) algorithms like Density Matrix Renormalisation Group (DMRG) and Simple Update (SU), have proven to be very effective in modelling these highly complex systems. However, they have been found to suffer scalability and complexity issues. This thesis explores the potential of imbuing Graph Neural Networks (GNNs) with quantum physics knowledge to address these challenges. We develop and evaluate several supervised learning models, including BlochGNN, EBlochGNN, RDMNet, and Neural Enhanced Quantum Belief Propagation (NEQBP), as well as an unsupervised Quantum Belief Propagation (QBP) method and a semi-supervised QBP-EBlochGNN model. These models are tested on diverse datasets, including Matrix Product States (MPS) and Projected Entangled Pair States (PEPS) of up to 20 qubits. Our research findings indicate that among the supervised learning methods, EBlochGNN performs the best but still does not surpass DMRG and only outperforms SU in one dataset. On the other hand, the unsupervised QBP method outperforms the baselines in all datasets. However, it is important to note that further improvements are needed in the implementation of QBP, as it currently faces issues related to efficiency. This work highlights the importance of integrating physics knowledge into AI models, setting the stage for future research to improve the accuracy and scalability of these approaches for quantum many-body systems.

Chapter 1

Introduction

The field of Artificial Intelligence (AI) for science is revolutionising how we tackle complex problems across various disciplines [1][2][3][4][5]. This study explores the potential of intersecting AI and quantum physics in the area of quantum many-body systems. These systems consist of multiple interacting particles, where their joint behaviour and long-range correlations give rise to complex behaviours that are challenging to model by only considering isolated particles.

Quantum many-body systems typically involve particles such as electrons, protons, neutrons, and photons, whose collective behaviour results in emergent properties. Their applications include high-temperature superconductivity, quantum magnetism, and quantum phase transitions, among many others. These applications are fundamental to understanding many phenomena in condensed matter physics, quantum chemistry, and materials science [6].

An important aspect of studying quantum many-body systems is determining their ground state, which is the lowest energy configuration of the system. The ground state is crucial because it provides the baseline for understanding the system's behaviour and stability. Accurately determining the ground state enables understanding and developing materials with specific properties, such as superconductors, which can conduct electricity without resistance, or magnetic materials used in data storage. During this study, we solely focus on the Ising model, which assumes that the way that particles are connected is very local. This assumption means that we can understand the entire system's behaviour through its smaller, manageable subsystems and thus gain insights into the system's global behaviour.

The properties of the ground state we approximate in this study are the ground state energy and Reduced Density Matrices (RDMs), which describe the statistical properties of subsystems within a larger quantum system [7][8]. This focus on RDMs aligns with the local nature of interactions in quantum many-body systems under the Ising model, allowing us to capture important features without needing to model the entire system simultaneously.

Traditional methods for tackling quantum many-body problems often become intractable as the number of particles increases. This rapid growth in complexity, known as the "curse of dimensionality" [9], has resulted in physics researchers developing specific techniques to model these highly complex systems.

One widely used approach in quantum physics to model quantum many-body systems is Tensor Networks (TNs). TNs are mathematical structures that efficiently represent quantum states by decomposing high-dimensional tensors into networks of lower-dimensional tensors connected by indices [10][11]. This representation captures the features and correlations in the system, which

traditional neural networks struggle with due to their inability to handle such multidimensional data structures effectively. Unlike traditional neural networks, TNs handle the exponential increase in dimensionality by utilising their intrinsic structure, making them a powerful tool for representing quantum states [12].

Given Graph Neural Network's (GNNs) success in fields such as chemistry, physics, bioinformatics, environmental science, neuroscience, and drug discovery [13][14], we propose exploring their potential to deepen our understanding of quantum many-body systems. GNNs excel at learning from data structured as graphs. While GNNs have previously been applied to quantum physics [15][16][17], this study aims to develop a set of architectures that infuse quantum physics knowledge into their design, setting the first steps towards creating more quantum-informed models without relying on quantum computers. We believe GNNs are a good candidate as they have proven to capture complex interactions [18], which are essential to model in quantum many-body systems. Moreover, how GNNs are designed aligns with the Ising model's assumption that a system's global behaviour can be modelled by considering its local interactions.

In this research, we have sequentially implemented and evaluated six different models for predicting ground state properties of quantum many-body systems. Each model integrates increasing levels of quantum physics knowledge. Among these models, BlochGNN has shown great performance, closely approaching the results of established baselines. RDMNet has also demonstrated promise, particularly in energy calculations. The unsupervised Quantum Belief Propagation (QBP) method also shows competitive results for datasets without cycles but struggles with more complex cyclic ones due to convergence issues, suggesting further refinement and optimisation. QBP and Neural Enhanced BP (NEQBP) also integrate TNs into their architecture, highlighting the potential for combining different methodologies.

Our findings suggest that integrating quantum physics knowledge into AI models can significantly enhance their performance. Moreover, by incorporating TNs into some models, we develop scalable and performing methods that are able to capture more intricate quantum correlations and entanglements, crucial for accurately modelling many-body systems. This hybrid approach bridges the gap between classical computational methods and quantum computing, providing a robust framework for future research.

Additionally, this research has the potential to influence other AI and quantum computing areas. By developing Quantum-Informed GNNs, we set a precedent for integrating domain-specific knowledge into AI models, enhancing their applicability across various scientific fields. The insights gained from this study could pave the way for the next generation of AI models, designed with a deeper understanding of the underlying physical principles governing the systems they aim to represent. However, there is still much work to be done, and future research should focus on refining these models and exploring new ways to embed quantum physics knowledge into AI systems.

Chapter 2

Background Knowledge

This section provides the necessary theoretical and technical background to understand the concepts and methods used in this study. We will discuss quantum states, the Ising model, TNs, GNNs, and Belief Propagation (BP), along with their relevance and application to quantum many-body systems.

2.1 Quantum States

This section will cover the fundamental concepts related to quantum states, including wavefunctions, the Bloch sphere, properties of quantum states, the partial trace, density operators, and reduced density matrices. These concepts allow an understanding of the methods used in this study to analyse quantum many-body systems.

2.1.1 Wavefunctions and the Hilbert Space

A wavefunction describes a quantum state, typically denoted as $|\Psi\rangle$, which encapsulates all the information about a quantum system. The wavefunction resides in a Hilbert space, a complex vector space equipped with an inner product, allowing for the mathematical manipulation of quantum states. For an N-particle system, its wavefunction can be expressed as:

$$|\Psi\rangle = \sum_{i_1,i_2,\ldots,i_N} c_{i_1 i_2 \ldots i_N} |i_1\rangle |i_2\rangle \cdots |i_N\rangle \tag{2.1}$$

where $c_{i_1 i_2 \ldots i_N}$ are complex coefficients, and $|i_k\rangle$ are the basis states of the k-th particle. The inner product in the Hilbert space between two quantum states Ψ and Φ, with coefficients $c_{i_1 i_2 \ldots i_N}$ and $d_{i_1 i_2 \ldots i_N}$ respectively, is defined as:

$$\langle \Phi | \Psi \rangle = \sum_{i_1,i_2,\ldots,i_N} c^*_{i_1 i_2 \ldots i_N} d_{i_1 i_2 \ldots i_N} \tag{2.2}$$

where $c^*_{i_1 i_2 \ldots i_N}$ is the complex conjugate of Ψ's coefficients. The result of the inner product $\langle \Phi | \Psi \rangle$ gives a measure of the overlap between the two states. If Ψ and Φ are orthogonal, the inner product is zero, indicating that they represent distinct states. If they are identical, the inner product is one, indicating complete overlap.

In quantum mechanics, states can be categorised as pure or mixed. A single wavefunction represents a pure state and lies on the surface of the Bloch sphere. A mixed state, on the other hand, is a statistical mixture of pure states and can be represented by a point inside the Bloch sphere.

2.1.2 Bloch Sphere

The Bloch sphere is a geometrical representation of the pure state space of a two-level quantum system (qubit). In addition, a qubit is the basic unit of quantum information, analogous to a classical bit but capable of existing in a superposition of states. A general pure state $|\psi\rangle$ of a qubit can be written as:

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\phi}\sin\left(\frac{\theta}{2}\right)|1\rangle \tag{2.3}$$

where θ and ϕ are the spherical coordinates on the Bloch sphere. The poles of the Bloch sphere represent the basis states $|0\rangle$ and $|1\rangle$.

2.1.3 Partial Trace

The partial trace operation describes subsystems of a larger quantum system. For a composite system with a wavefunction $|\Psi\rangle$ describing the entire system, the wavefunction for a subsystem can be obtained by summing (tracing out) over the degrees of freedom of the other subsystems. Mathematically, for a subsystem consisting of particles $1, 2, \ldots, k$, the reduced wavefunction $|\Psi^{(1,2,\ldots,k)}\rangle$ is given by:

$$|\Psi^{(1,2,\ldots,k)}\rangle = \sum_{m_{k+1},m_{k+2},\ldots,m_N} \langle m_{k+1}, m_{k+2}, \ldots, m_N|\Psi\rangle$$

This operation effectively sums over all possible states of the particles being traced out, leaving a wavefunction that describes the remaining particles.

2.1.4 Properties of Quantum States

The properties of quantum states ensure their physical validity and must be considered when designing approaches that approximate these. Key properties include:

- **Hermitian Operators**: Operators such as the Hamiltonian and density matrices are Hermitian, implying that they are equal to their conjugate transpose. This property ensures real eigenvalues corresponding to observable quantities in quantum mechanics.
- **Positive Semi-Definite (PSD) Matrices**: The eigenvalues of operators are non-negative, reflecting the probability distribution nature of quantum states.
- **Unit Trace**: The trace of a density matrix equals one, ensuring that the total probability distribution of all possible states sums to one.
- **Symmetry**: Quantum states and operators can exhibit symmetry properties based on the exchange of particles. We do not need to worry about this property as we deal with Spin-1/2 objects (protons, neutrons, electrons, neutrinos, quarks, etc.).

2.1.5 Density Operators

The density operator ρ describes a quantum system's state, it is especially useful when the system is in a mixed state. For a pure state $|\Psi\rangle$, the density operator is defined as:

$$\rho = |\Psi\rangle\langle\Psi| \tag{2.4}$$

For mixed states, the density operator is a weighted sum of pure state projectors, represented as:

$$\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i| \quad \text{where} \quad \sum_i p_i = 1 \tag{2.5}$$

Here, p_i are the probabilities associated with each pure state $|\psi_i\rangle$, and they satisfy the normalization condition $\sum_i p_i = 1$.

The density operator can be used to calculate expectation values of observables and to describe statistical mixtures of different quantum states. An added property of density operators is that their trace must sum to one. This property is directly linked to the probability distribution of mixed states, as the sum of the probabilities p_i must equal one. Moreover, the density operator's (ρ) eigenvalues reflect the quantum states' probability distribution. Therefore, ρ must also be PSD and Hermitian.

Reduced Density Matrices

Reduced Density Matrices (RDMs) are derived from the system's wavefunction and provide a way to describe subsystems of a larger quantum system. Using the N-particle wavefunction defined in (2.1) and constructing the density operator ρ as in Equation 2.4, we can obtain the RDM for a subsystem by performing a partial trace. For a subsystem consisting of particles $1, 2, \ldots, k$, the reduced density matrix $\rho^{(1,2,\ldots,k)}$ is given by:

$$\rho^{(1,2,\ldots,k)} = \mathrm{Tr}_{k+1,k+2,\ldots,N}(\rho) \tag{2.6}$$

This partial trace operation, as defined earlier, effectively sums over all possible states of the particles being traced out, leaving an RDM that describes the remaining particles.

RDMs are essential for understanding entanglement and correlations within quantum systems. For example, the 1-RDM $\rho^{(1)}$ provides information about the probability distribution and coherence of a single particle, while the 2-RDM $\rho^{(1,2)}$ describes the particle-particle correlations and entanglement between site one and two.

2.1.6 Hamiltonian

The Hamiltonian is the operator corresponding to the total energy of the system. It plays a central role in quantum mechanics, as it defines the system's time evolution and determines its energy spectrum. For a quantum system, the time-independent Schrödinger equation is given by:

$$H|\Psi\rangle = E|\Psi\rangle \tag{2.7}$$

where H is the Hamiltonian, $|\Psi\rangle$ is the wavefunction, and E is the energy eigenvalue.

Exact Diagonalisation

Exact diagonalisation is a numerical technique from linear algebra that can be used to find the eigenvalues and eigenvectors of the Hamiltonian matrix [19]. This method is exact but becomes impractical for large systems due to the exponential growth of the Hilbert space with the number of particles.

The Hamiltonian matrix H is diagonalised by solving the eigenvalue equation:

$$H|\psi_i\rangle = E_i|\psi_i\rangle \tag{2.8}$$

where $|\psi_i\rangle$ are the eigenvectors and E_i are the corresponding eigenvalues of the Hamiltonian. The ground state energy is the smallest eigenvalue E_0, and the corresponding eigenvector $|\psi_0\rangle$ represents the ground state wavefunction.

2.2 Ising Model

The Ising model[20] is a mathematical model used in statistical mechanics to describe ferromagnetism in statistical physics; it is also widely used in the study of critical phenomena, phase transitions, and properties of magnetic materials, as well as modelling liquid-gas transitions. It consists of discrete variables called spins, which, for spin-1/2 objects (our focus), can exist in either up or down states. The Hamiltonian of the Ising model in the presence of both transverse and longitudinal fields is given by:

$$H(\sigma) = \sum_{(n_i,n_j)\in\mathcal{N}} J_{n_i n_j} J_{n_i n_j} \sigma^z_{n_i} \sigma^z_{n_j} - \sum_{n_i} g_{n_i} \sigma^z_{n_i} - \sum_{n_j} u_{n_j} \sigma^x_{n_j} \tag{2.9}$$

where $J_{n_i n_j} \in \mathbb{R}$ represents the interaction strength between spins at sites n_i and n_j, $g_{n_i} \in \mathbb{R}$ is the longitudinal field at site n_i, and $u_{n_j} \in \mathbb{R}$ is the transverse field at site n_j. Moreover, $\mathcal{N}$ represents the set of nearest-neighbour pairs, and σ^z and σ^x are Pauli matrices. The Hamiltonian $H(\sigma)$ captures the energy of a given configuration of spins, considering both their interactions and the influence of external fields.

The energy expectation value E of the system can be expressed in terms of the RDMs as:

$$E = \sum_{n_i} g_{n_i} \operatorname{Tr}\left(\rho^{(n_i)} \sigma^z_{n_i}\right) + \sum_{n_j} u_{n_j} \operatorname{Tr}\left(\rho^{(n_j)} \sigma^x_{n_j}\right) + \sum_{(n_i,n_j)\in\mathcal{N}} J_{n_i n_j} \operatorname{Tr}\left(\rho^{(n_i,n_j)} (\sigma^z_{n_i} \otimes \sigma^z_{n_j})\right) \tag{2.10}$$

Here, $\rho^{(n_i)} \in \mathbb{C}^{2\times 2}$ and $\rho^{(n_i,n_j)} \in \mathbb{C}^{4\times 4}$ are the RDMs for single sites and pairs of sites respectively. These terms account for the contributions of the local fields and spin interactions to the system's total energy. In addition, the operator $\otimes$ refers to the *tensor product*, which is covered in Section 2.3.

If we analyse Equation 2.9, we can see that the system is described only considering local interactions under the Ising Model. This notion of explaining global behaviours by only considering local interactions is also the basis of Graph Neural Networks, which is why it makes sense to investigate ways in which GNNs can be applied to quantum many-body systems using the Ising Model.

2.3 Tensor Networks

Tensor networks (TNs) are a powerful framework for efficiently representing and manipulating high-dimensional tensors [10][11]. They are widely applied in quantum many-body physics to

describe complex quantum states and perform numerical simulations. A TN comprises tensors connected by indices, each representing a physical or bond dimension.

2.3.1 Tensor Network Diagrams

TN diagrams are a graphical representation of tensor contractions, where tensors are depicted as nodes and indices as edges connecting the nodes. Each tensor T^n has a physical leg i^n and a set of bond dimensions $\boldsymbol{\omega}^n$ connecting to other tensors. The physical leg represents the local Hilbert space dimension (often denoted as the physical dimension), while bond dimensions represent the entanglement between different parts of the system.

We write a TN as a collection of tensors $T^n_{i^n}(\boldsymbol{\omega}^n_1, \ldots, \boldsymbol{\omega}^n_{K_n})$, where n indexes the tensor, i^n is the physical leg for that tensor (one per tensor), and $\boldsymbol{\omega}^n$ is the K_n-dimensional vector of bond dimensions that connect to other tensors. All are discrete variables. As shown in Figure 2.1(left), a Matrix Product State (MPS) is a type of TN where each tensor is connected linearly by bond dimensions, with one physical leg per tensor.

Figure 2.1: Matrix Product State (MPS) diagram (left) and its dual representation (right)

We will be interested in the dual graph representation of the TN, formed by stacking the TN with its complex conjugate and contracting over its physical legs. This is illustrated in Figure 2.1 (right), which shows the dual representation of an MPS:

$$\mathcal{T}(\mathbf{z}^n) = \sum_{i^n} T^n_{i^n}(\boldsymbol{\omega}^n) T^{n*}_{i^n}(\boldsymbol{\omega}'^n) \tag{2.11}$$

with $\mathbf{z}^n = (\boldsymbol{\omega}^n, \boldsymbol{\omega}'^n)$ the combined index.

2.3.2 Tensor Network Types

Various types of TNs are used to represent different structures and dimensions of quantum systems. The most common types include Matrix Product States (MPS) [21] and Projected Entangled Pair States (PEPS) [22], which we focus on in this study.

Matrix Product States (MPS) are used to describe one-dimensional quantum systems. In an MPS, each tensor is connected linearly by bond dimensions, with one physical leg per tensor. This structure is particularly efficient for representing ground states of one-dimensional gapped systems. As shown in Figure 2.1 (left), the tensors are arranged in a chain, making it easy to visualise the linear connections.

Projected Entangled Pair States (PEPS) are used for two-dimensional systems. In PEPS, tensors are arranged in a grid, with bond dimensions connecting neighbouring tensors in the grid. This type of system allows for representing higher-dimensional systems with complex entanglement structures. Figure 2.2 shows a typical PEPS arrangement and its dual, highlighting the grid-like structure.

Other TN types include Tree Tensor Networks (TTN) [23] and Multi-scale Entanglement Renormalisation Ansatz (MERA) [24]. TTNs are hierarchical structures that efficiently represent

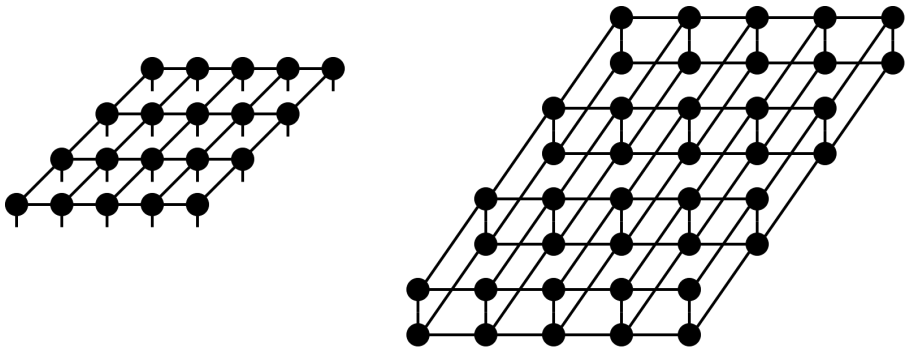

Figure 2.2: Projected Entangled Pair State (PEPS) diagram (left) and its dual representation (right)

states with a tree-like entanglement structure. They are often used in studying quantum circuits and hierarchical data. On the other hand, MERA networks are used to examine critical systems and scale-invariant structures, efficiently representing systems with long-range entanglement.

2.3.3 Tensor Network Contraction

TN contraction involves summing over the shared indices (bond dimensions) between tensors to reduce the network to a more straightforward form or to compute specific quantities[11].

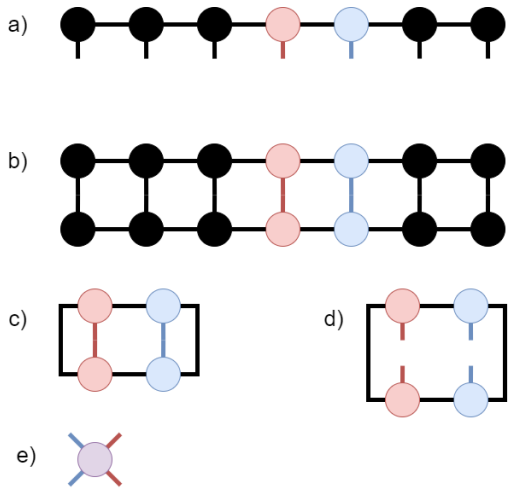

Figure 2.3: This figure shows the process of obtaining 2-RDMs from a) an initial MPS system, b) constructing the dual representation, c) performing tensor contraction, d) opening up the physical legs, and e) further contracting to obtain the (purple) 2-RDM for the blue and red particles. Realistically, you would fully contract the network and open the physical legs, but this way is more didactic.

As shown in Figure 2.3, we obtain a reduced TN after contracting the bond dimensions $\mathbf{z}^n$ with neighbouring bonds. For example, if we contract all indices for a Matrix Product State (MPS), we get the partition function:

$$Z = \sum_{z^1,\ldots,z^N} \mathcal{T}^1(z^1)\mathcal{T}^2(z^1,z^2)\ldots\mathcal{T}^{N-1}(z^{N-1},z^N)\mathcal{T}^N(z^N) \tag{2.12}$$

A reduced density matrix (RDM) ρ^n results by contracting all indices but opening the physical legs on variable n:

$$\rho^n_{i^n,i'^n} = \sum_{\boldsymbol{\omega},\boldsymbol{\omega}'} \mathcal{T}^1(\boldsymbol{\omega}^1,\boldsymbol{\omega}'^1)\dots T^n_{i^n}(\boldsymbol{\omega}^{n-1},\boldsymbol{\omega}^n)T^{n*}_{i'^n}(\boldsymbol{\omega}'^{n-1},\boldsymbol{\omega}'^n)\dots\mathcal{T}^N(\boldsymbol{\omega}^N,\boldsymbol{\omega}'^N) \tag{2.13}$$

$$= \sum_{\mathbf{z}} \mathcal{T}^1(\mathbf{z}^1)\dots T^n_{i^n}(\boldsymbol{\omega}^{n-1},\boldsymbol{\omega}^n)\mathcal{T}^{n*}_{i'^n}(\boldsymbol{\omega}'^{n-1},\boldsymbol{\omega}'^n)\dots\mathcal{T}^N(\mathbf{z}^N) \tag{2.14}$$

Here, the RDM ρ^n has the physical dimension corresponding to the local Hilbert space dimension at site n. The bond dimensions capture the entanglement between different parts of the system, making TNs an efficient representation of many-body quantum states. The physical dimension is typically small (e.g., 2 for spin-1/2 systems), while the bond dimension can grow significantly with system size, affecting the computational complexity of TN methods.

In the dual graph representation, the RDM can be visualised as a network where the physical legs are opened, representing the subsystem of interest while the other indices are contracted. This representation helps in understanding the structure and correlations within the quantum system. As shown in Figure 2.3, the dual graph provides an intuitive way to see how the tensors are interconnected through their physical and bond dimensions and allows us to obtain N-RDMs through tensor contraction.

2.3.4 Tensor Network Algorithms

Many TN algorithms exist that allow converging a TN to its ground state given a Hamiltonian [12][25][26][27]. These algorithms optimise the TN representation of the quantum state to minimise the energy converging towards the ground state. Once the ground state is found, TN contraction can compute the RDMs and thus determine the ground state energy. This study focuses on Simple Update and Density Matrix Renormalisation Group as baselines.

Simple Update

The Simple Update (SU) [25] algorithm is primarily used with PEPS. It optimises the TN by iteratively updating the tensors based on local operations. In each update step, the tensors are optimised to minimise the local energy, considering only the immediate neighbouring tensors. This method is computationally efficient and straightforward to implement, making it suitable for large systems due to its local nature. However, Simple Update may not capture long-range correlations accurately because it only considers local interactions. This can lead to slower convergence for systems with strong entanglement, as the algorithm does not effectively handle the global structure of the state.

Density Matrix Renormalisation Group

The Density Matrix Renormalisation Group (DMRG) [12] is a widely used algorithm for finding the ground state of MPS. DMRG optimises the MPS by iteratively sweeping through the chain and updating one or two sites simultaneously while keeping the rest of the system fixed. During each update, the algorithm constructs the reduced density matrix for the optimised subsystem and retains the most significant states by truncating the less significant ones. DMRG is highly accurate for one-dimensional systems and efficiently captures long-range correlations. However, it becomes computationally intensive for higher-dimensional systems and requires careful handling of truncation errors to maintain accuracy. As with SU, once the TN converges to the ground state, we can obtain the RDMs through contraction and thus calculate the ground state energy.

2.4 Graph Neural Networks

GNNs are a class of neural networks designed to work with data structured as graphs. In GNNs, nodes represent entities, and edges represent relationships between these entities.
One particular type of GNN that we are trying to extend in this study is the message-passing neural network (MPNN) [13]. In MPNNs, each node in the graph sends and receives messages to and from its neighbouring nodes, a concept which takes inspiration from TN. The state of each node is updated based on the messages it receives, which are aggregated and processed through neural network layers. The message-passing phase allows the network to learn representations that capture the local and global structure of the graph. Formally, message passing involves two steps: message aggregation and node update.

Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with nodes $v_i \in \mathcal{V}$ and edges $e_{uv} \in \mathcal{E}$. We define an MPNN as:

$$m_{u \to v}^{l} = \phi_e(h_u^l, h_v^l, e_{uv}) \tag{2.15}$$

$$m_v = \mathsf{AGG}(\{m_{u \to v} : u \in \mathcal{N}(v)\}) \tag{2.16}$$

$$h_v^{(l+1)} = \phi_h(h_v^{(l)}, m_v^l) \tag{2.17}$$

Where $\mathcal{N}(v)$ denote the set of neighbours of node v, h_u^l and h_v^l are the hidden states of nodes u and v at layer l, respectively, and e_{uv} represents the edge features between nodes u and v. An important remark in the design of MPNNs is that AGG must be a permutation invariant function. Both ϕ_e and ϕ_h represent edge and node operations, respectively, which are commonly implemented through Multi-Layer Perceptrons (MLP).

Other types of GNNs include Graph Convolutional Networks (GCNs) [28], Graph Attention Networks (GATs) [29], and Graph Recurrent Neural Networks (GRNNs) [30]. GCNs extend the concept of convolution to graph-structured data by applying convolutional operations to the nodes and their neighbours, capturing local graph structure like traditional convolutional neural networks. GATs introduce attention mechanisms to weigh the importance of neighbouring nodes differently, allowing the model to focus on more relevant nodes. Finally, GRNNs apply recurrent neural network (RNN) principles to graphs, capturing sequential dependencies in graph data.

2.5 Belief Propagation

Belief Propagation (BP) is an algorithm for performing probabilistic inference. It is widely used in various domains, including computer vision, error correction, and statistical physics because it can efficiently compute marginal distributions and perform probabilistic reasoning. This section will explore two main types of TN: Vanilla TN, which operates on Probabilistic Graphical Models (PGM), and TN TN, which extends BP to TNs.

2.5.1 Vanilla Belief Propagation

BP [20] is originally an algorithm that performs inference on PGMs, such as Bayesian networks and Markov random fields. The Sum-Product TN algorithm is a specific type of BP used to compute marginal distributions of variables. It operates by performing message passing along the edges of a graph, updating beliefs about the states of the nodes based on the messages received from neighbouring nodes.

In the Sum-Product BP algorithm, each node sends a message to its neighbouring nodes, representing the belief about the node's state given the evidence from its other neighbours. The message from node u to node v is defined as:

$$m_{u\to v}(x_v) = \sum_{x_u} \psi_{uv}(x_u, x_v)\psi_u(x_u) \prod_{w\in\mathcal{N}(u)\setminus v} m_{w\to u}(x_u), \tag{2.18}$$

where $\psi_{uv}(x_u, x_v)$ is the potential function between nodes u and v, $\psi_u(x_u)$ is the potential function at node u, and $\mathcal{N}(u)$ denotes the neighbours of node u. This process continues iteratively until the messages converge to stable values, representing the approximate marginal distributions of the nodes.

TN is equivalent to the message-passing framework used in GNNs, where messages are exchanged between nodes to update their states. However, BP is specifically designed for probabilistic inference and operates on probabilistic graphical models.

BP is exact when applied to tree-structured graphs, as there are no loops, and the algorithm can fully propagate the information through the graph without redundancy. However, the algorithm becomes approximate in graphs with loops (loopy BP). Loopy BP continues to pass messages even after cycles are formed, resulting in approximate marginal distributions. Despite its approximate nature in loopy graphs, BP has performed well in practice for many applications.

This study will explore integrating TN techniques into GNN architectures, leveraging their strengths in probabilistic inference to enhance the modelling of complex quantum many-body systems.

2.5.2 Tensor Network Belief Propagation

TN Belief Propagation extends the BP algorithm applied to TNs. This approach, as described by Alkabetz et al [31], involves forming the dual graph of the TN to facilitate the computation of approximately reduced density matrices (RDMs). The dual graph, as explained in Section 2.3.1, is used because it provides a quasi-canonical representation of the TN where its BP fixed points are identical to those obtained through the trivial Simple Update algorithm (modification of SU where the unitary transformations applied are the identity matrix).

We can define BP updates to compute the system's RDMs, given tensors $\{\mathcal{T}^n\}$. Define $\mathbf{z}_n$ as all the variables in the argument of tensor n and $\mathcal{N}(n)$ to be the tensors connected to $\mathcal{T}^n$ through a bond. The following will use a different indexing scheme for the variables. Because variables are always in the argument of precisely two tensors, we can index them by z_{mn} if they are in the boundary between tensor m and tensor n.

TN-BP's message-passing scheme can be written as:

$$M_{m\to n}(z_{mn}) \propto \sum_{\{\mathbf{z}_m\setminus z_{mn}\}} \mathcal{T}^m(\mathbf{z}_m) \prod_{k\in\mathcal{N}(m)\setminus n} M_{k\to m}(z_{km}), \quad M_{m\to n}(z_{mn}) \in \mathbb{R}^{B,B} \tag{2.19}$$

Where $M_{m\to n}(z_{mn})$ is the message sent from tensor m to tensor n about the variable z_{mn}, this message passing scheme iteratively updates the messages based on the current state of the neighbouring messages and the local tensor $\mathcal{T}^m$. The shape of each message will be $\mathbb{R}^{B,B}$, given

that BP is performed in a TN with real-valued tensors, with a homogeneous bond dimension of size B.

It has been proven that iterating these message updates maintains the PSD nature of the messages if we initialise them as PSD and if $\mathcal{T}^n$ is PSD, which they are in most cases. On a tree-like graph, these updates are guaranteed to produce exact contractions. However, the algorithm gives approximate results when run on a loopy graph such as a PEPS system.

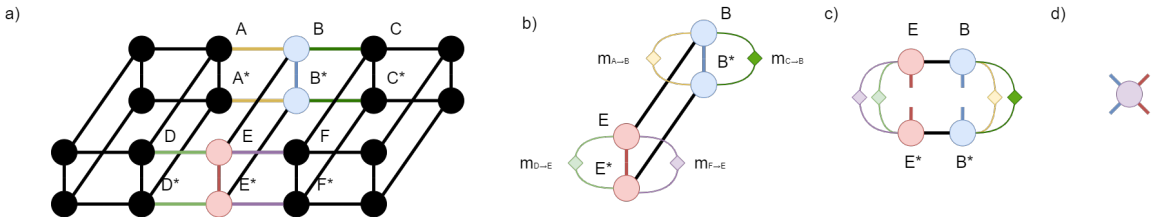

Figure 2.4: This figure shows the process of obtaining 2-RDMs using BP messages, specifically the 2-RDM for the particles E and B from a) a dual PEPS system, b) Collecting the messages sent to B and E excluding the ones sent between them, c) opening up the physical legs and e) tracing out the messages to obtain the (purple) 2-RDM for particles B and E

Figure 2.4 shows how we can calculate RDMs using the TN BP scheme, where it can be seen that BP and TN contraction are equivalent, as expressed by Alkabetz et al. [31]. A duality exists between QBP and BP on Factor Graphs, where one can convert a dual representation to a TN to a double-edge factor graph (DEFG) and run BP, arriving at the same marginals.

By employing QBP, we can efficiently approximate the RDMs in complex quantum many-body systems. This approach enables us to handle larger systems with intricate entanglement structures, providing a more scalable way of evaluating TNs than standard TN contraction.

Chapter 3

Related Works

Quantum many-body systems have been widely studied, employing various techniques to model their complex interactions. Besides TN-based approaches, many other methods have been developed. For instance, the variational Monte Carlo (VMC) and diffusion Monte Carlo (DMC) methods discussed by Toulouse et al. [32] offer robust techniques for addressing many-body problems. VMC is known for its simplicity and efficiency. In contrast, DMC is often regarded as more accurate due to its ability to sample directly from the exact wave function instead of using an approximation, as in VMC. However, both methods can face scalability issues and be computationally expensive for large systems.

Huang et al. [33] introduced a machine learning (ML) framework for quantum many-body problems using classical shadows to address scalability. This technique compresses quantum states into classical representations, allowing for efficient processing and analysis. Using data from classical simulations or quantum experiments, their ML algorithm can predict ground-state properties with a small error. This method differs from non-learning classical algorithms that require substantially more computation time to achieve similar tasks. Similarly, the quantum eigensolver proposed by Kandala et al. [34] uses quantum hardware to approximate ground states. Their approach optimises quantum circuits for specific problems with up to six qubits. Despite these advancements, the efficacy of this approach is limited by the current capabilities of quantum hardware, which we avoid in this study by focusing on classical computing.

In contrast, Neural Network (NN) based approaches have demonstrated significant potential in solving quantum problems. Carleo et al. [35] pioneered using NNs to solve the quantum many-body problem. They introduced a variational representation of quantum states based on NNs with variable numbers of hidden neurons. Their proposed method can either find the ground state or describe the unitary time evolution of complex interacting quantum systems. Their approach accurately describes the equilibrium and dynamical properties of 1D and 2D quantum many-body systems. Unlike TN States, Neural Quantum States (NQS) feature intrinsically non-local correlations, which can lead to substantially more compact representations of many-body quantum states.

In alignment with using NNs within quantum chemistry, Qiao et al. [36] developed OrbNet, which employs symmetry-adapted atomic-orbital features for deep learning in quantum chemistry. OrbNet predicts energies within chemical accuracy of density functional theory (DFT) at a computational cost of thousands of times lower. This method has shown exceptional learning efficiency and transferability, effectively predicting DFT results. For datasets of drug-like molecules, OrbNet demonstrates remarkable accuracy and efficiency.

Similarly, Pfau et al. [37] developed FermiNet, an NN-based solution for the many-electron Schrödinger equation, which resulted in high accuracy and computational efficiency in a self-supervised way. Their approach uses deep learning to approximate wavefunctions, achieving high accuracy in electronic structure calculations. FermiNet makes the VMC method competitive with DMC and auxiliary-field quantum Monte Carlo (AFQMC), among other methods. Importantly, Pfau et al. show how an NN with one set of training parameters can attain high accuracy across various systems. This method shows how NNs can significantly improve the accuracy of VMC to the point where it outperforms other quantum chemistry methods.

Building on this, von Glehn et al. [38] introduced the PsiFormer, a self-attention-based neural network architecture, for solving the many-electron Schrödinger equation. This model, inspired by the Transformer from Vaswani et al. [39], uses self-attention mechanisms to gate interactions between electrons. The PsiFormer can be used as a drop-in replacement for models like FermiNet, often dramatically improving the accuracy of calculations, especially for larger molecules where the ground state energy can be greatly improved. This demonstrates that self-attention networks can learn complex quantum mechanical correlations between electrons more effectively than previous methods.

GNNs have also been explored for applications in quantum mechanics, given their success in other scientific fields. Kochkov et al. [17] demonstrated the potential of GNNs to learn the ground states of quantum Hamiltonians, capturing intricate interactions within the system. Their approach uses the graph structure of many-body quantum systems to model interactions more effectively. Expanding on this, He et al. [16] developed a GNN-based predictor for quantum architecture search, showcasing the adaptability and power of GNNs in optimising quantum circuits. Furthermore, Ye et al. [15] proposed a symmetrical GNN for quantum chemistry that incorporates both real and momenta space, resulting in substantial improvements in computational efficiency and accuracy.

Efforts to combine TNs with neural networks have also shown promising results. Novikov et al. [40] explored tensorising neural networks to reduce the number of parameters while maintaining model capacity, providing a framework for efficiently handling high-dimensional data. Furthermore, Huggins et al. [41] aimed towards applying TNs within the field of quantum ML, using TN's structured representation to enhance GNNs. The whitepaper by Rieser et al. [42] provides an extensive overview of how TNs can be integrated within AI models, emphasising their potential to manage complex data structures effectively.

Integrating GNNs with TNs has been another area of interest. Hua et al. [43] introduced high-order pooling for GNNs with tensor decomposition, effectively managing the complexity of graph data. Building on this, Baghershahi et al. [44] proposed efficient relation-aware neighbourhood aggregation using tensor decomposition, further enhancing GNN scalability. Jia et al. [45] and Zhao et al. [46] explored dynamic and multi-view tensor graph neural networks, respectively, demonstrating how TN techniques can be embedded within GNN frameworks to improve performance on complex tasks. However, these approaches often do not incorporate the full spectrum of quantum information, which is crucial for accurate quantum many-body modelling.

Using BP with TNs is another intriguing approach. Leifer et al. [47] discussed quantum graphical models and BP, laying the groundwork for applying BP to quantum systems. Sahu et al. [48] proposed efficient TN simulations on sparse graphs, utilizing BP to enhance accuracy. Similarly, Guo et al. [49] developed the block TN algorithm for two-dimensional TNs, provid-

ing a scalable approach to managing the computational complexity of TNs. Tindall et al. [50] combined BP with TNs to gauge TNs, offering a novel method to handle computational challenges. Moreover, Alkabetz et al. [31] proved the duality between BP on TNs and the trivial SU algorithm. These studies illustrate the potential of BP in quantum systems, yet integrating BP with GNNs remains an under-explored area.

This study draws inspiration from these varied approaches, aiming to develop quantum-informed GNN architectures that effectively model quantum many-body systems. For instance, Batatia et al. [51] introduced equivariant matrix function neural networks, highlighting the need for messages in GNNs to be more than scalar values. Additionally, Arjovsky et al. [52] explored using unitaries throughout neural network layers, linking closely with quantum mechanics. Both concepts of matrix messages and the use of unitaries are further developed in Section 4 when explaining RDMNet. Moreover, Integrating BP into neural networks, as demonstrated by Satorras et al. [53], shows how probabilistic graphical models can enhance neural network performance, which is a concept also further developed in the NEQBP architecture.

This research considers these approaches and aims to develop novel GNN architectures incorporating quantum physics knowledge. It relies on the strengths of TNs and NNs while integrating techniques from probabilistic inference like BP for improved performance and scalability. This aims to address the gaps in existing methods, providing a more accurate and efficient framework for modelling quantum many-body systems.

Chapter 4

Methodology

This section outlines the methods and procedures used to conduct this study, explaining the loss function used and the developed architectures.

4.1 Loss Function

The multi-objective loss function used in this study combines the ground state energy and RDMs. The total loss function $\mathcal{L}_T$ is defined as:

$$\mathcal{L}_T = \mathcal{L}_E + \lambda(\mathcal{L}_{1-\text{RDM}} + \mathcal{L}_{2-\text{RDM}}), \tag{4.1}$$

where $\mathcal{L}_E$ is the loss associated with the ground state energy, $\mathcal{L}_{1-\text{RDM}}$ and $\mathcal{L}_{2-\text{RDM}}$ are the losses associated with the 1-RDMs and 2-RDMs, respectively, and λ is a weighting parameter that balances the contributions of the RDM losses.

The ground state energy loss $\mathcal{L}_E$ is measured using the L1 loss (absolute error) between the predicted ground state energy $\hat{E}_0$ and the true ground state energy E_0:

$$\mathcal{L}_E = |\hat{E}_0 - E_0|, \tag{4.2}$$

The RDM losses are measured using the trace distance, a PSD penalty, and a unit trace penalty loss. Trace distance is a suitable metric for comparing quantum states, whereas the PSD and unit trace penalty help to ensure the RDMs are physically valid. The trace distance between two density matrices ρ^i and ρ^j and the PSD and unit trace penalty for an RDM ρ where $(\rho, \rho^i, \rho^j) \in \mathbb{R}^{K,K}$ are defined as:

$$D_{\mathsf{trace}}(\rho^i, \rho^j) = \frac{1}{2}\mathsf{Tr}\left|\left|\rho^i - \rho^j\right|\right|_1 \tag{4.3}$$

$$\rho = U\Lambda U^\dagger, \quad \mathsf{where} \quad \Lambda = diag\left(\lambda_1, \lambda_2 ... \lambda_K\right) \tag{4.4}$$

$$L_{PSD}(\rho) = \sum_k^K \mathsf{min}(\lambda_k, 0)^2 \tag{4.5}$$

$$L_{tr}(\rho) = \max\left(|\|\rho_i\|_* - 1| - \epsilon, 0\right) \tag{4.6}$$

Where the trace norm $||A||_1 = Tr\sqrt{A^\dagger A}$ and the nuclear norm $||A||_* = \sum_i \lambda_i$, with λ_i being the singular values of A. For the 1-RDMs and 2-RDMs, the losses $\mathcal{L}_{1-\text{RDM}}$ and $\mathcal{L}_{2-\text{RDM}}$ are calculated as:

$$\mathcal{L}_{1-\text{RDM}} = \frac{1}{N}\sum_{n=1}^{N}\left(L_{tr}\left(\hat{\rho^n}\right) + L_{PSD}\left(\hat{\rho^n}\right) + D_{\mathsf{trace}}\left(\hat{\rho^n}, \rho^n\right)\right) \tag{4.7}$$

$$\mathcal{L}_{1-\text{RDM}} = \frac{1}{|E|}\sum_{i,j\in E}\left(L_{tr}\left(\hat{\rho}^{(i,j)}\right) + L_{PSD}\left(\hat{\rho}^{(i,j)}\right) + D_{\mathsf{trace}}\left(\hat{\rho}^{(i,j)}, \rho^{(i,j)}\right)\right) \tag{4.8}$$

The trace distance is a good candidate for measuring the difference between quantum states because it satisfies several desirable properties, such as being easily differentiable, bounded for physically plausible RDMs, and having a clear interpretation. Other metrics explored include quantum fidelity and the squared Hilbert-Schmidt distance. However, the trace distance was more stable and consistent in our experiments, making it the preferred choice for our loss function.

4.2 Architectures

This section presents the architectural details of our proposed models for predicting ground state properties in quantum many-body systems. We group as Quantum GNNs (QGNNs): BlochGNN, EBlochGNN, RDMNet and NEQBP. Moreover, the six proposed models are split into two categories: physics-inspired GNNs and TN-based approaches. The latter also includes QBP, NEQBP and QBP-BlochGNN. Each architecture is introduced in the subsections below.

4.2.1 Physics-inspired GNNs

This section introduces two novel architectures inspired by physical principles: BlochGNN (including EBlochGNN) and RDMNet.

BlochGNN & EBlochGNN

To develop the mathematics behind BlochGNN, our first goal is to develop a vector representation of RDMs where random values of these will still result in physically valid RDMs. We take inspiration from the Bloch Sphere representation of RDMs.

To achieve this, we rely on the idea that any 2×2 Hermitian matrix can be written as a linear combination:

$$\rho = a_1\sigma_x + a_2\sigma_y + a_3\sigma_z + a_0 I_2 \tag{4.9}$$

where $\mathbf{a} \in \mathbb{R}^4$. To ensure the matrix is PSD, we set $a_0 = \frac{1}{2}$, as the Pauli matrices are traceless. However, this does not always guarantee positive eigenvalues. The Bloch sphere representation helps:

$$\rho = \frac{1}{2}(I + \vec{a}\cdot\vec{\sigma}) \tag{4.10}$$

with $\|\vec{a}\|_2 \leq 1$. This allows us to specify 1-RDMs with 3 Degrees of Freedom (DOF).
Generalising this approach using the lie algebra $\mathbf{SU}(2^n)$ of traceless Hermitian $2^n \times 2^n$ matrices, we have for n-RDMs:

$$\rho = 2^{-n}\left(I_{2^n} + \sum_i b_i\Lambda_i\right) \tag{4.11}$$

where $\mathbf{b} \in \mathbb{R}^{2^{2n}-1}$ and $\Lambda_i \in \mathbf{SU}(2^n)$. This results in 15 DOF for 2-RDMs. For real RDMs, we exclude basis elements with imaginary numbers, reducing the requirement to 2 DOF for real 1-RDMs and 9 DOF for real 2-RDMs.

We define our specialised MPNN using this RDM representation. Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with nodes $v_i \in \mathcal{V}$ and edges $(i, j) \in \mathcal{E}$, we define an L-layer BlochGNN starting with the feature embeddings:

$$h_i^0 = \phi_{\mathsf{embed}_h}(g_i, u_i), \quad h_i^l \in \mathbb{R}^{128} \quad \forall l \in L, i \in \mathcal{V} \tag{4.12}$$

$$e_{ij}^l = \phi_{\mathsf{embed}_e}(J_{ij}), \quad e_{ij}^l \in \mathbb{R}^{128} \quad \forall (i,j) \in \mathcal{E} \tag{4.13}$$

After embedding the edge features and node features, we define a BlochGNN layer as:

$$m_{i\to j}^l = \phi_e(h_u^l, h_j^l, e_{ij}^l), \quad m_{i\to j}^l \in \mathbb{R}^{128} \quad \forall i,j \in \mathcal{V}, l \in L \tag{4.14}$$

$$m_j^l = \sum_{i \in \mathcal{N}(j)} m_{i\to j}, \quad m_{i\to j}^l \in \mathbb{R}^{128} \quad \forall j \in \mathcal{V}, l \in L \tag{4.15}$$

$$h_i^{(l+1)} = \phi_h(h_i^{(l)}, m_i^l) \quad \forall i \in \mathcal{V} \tag{4.16}$$

$$e_{ij}^{(l+1)} = m_{i\to j}^l \quad \forall i,j \in \mathcal{V}, l \in L \tag{4.17}$$

After propagating through the layers, we define the readout at the last layer l as:

$$h_{i_o} = \phi_{h_o}(h_i^l), \quad h_{i_o} \in \mathbb{R}^2 \quad \forall i \in \mathcal{V} \tag{4.18}$$

$$e_{ij_o} = \phi_{e_o}(e_{ij}^l), \quad e_{ij_o} \in \mathbb{R}^9 \quad \forall (i,j) \in \mathcal{E} \tag{4.19}$$

$$E_o = \phi_{E_o}(\sum_i h_{i_o}), \quad E_o \in \mathbb{R} \tag{4.20}$$

The functions ϕ_{embed_e}, ϕ_{embed_h}, ϕ_e, ϕ_h, ϕ_{h_o}, ϕ_{e_o}, and ϕ_{E_o} are implemented as two-layer MLPs with SiLU activation functions. Having obtained our node-level and edge-level readouts h_{i_o} and e_{ij_o} representing the RDMs' Bloch sphere representation, we can convert them back to the corresponding 1-RDMs and 2-RDMs:

$$\rho^i = 2^{-1}\left(I_2 + \sum_k^2 h_{i_o,k}\sigma_k\right) \quad \forall i \in \mathcal{V} \tag{4.21}$$

$$\rho^{(i,j)} = 2^{-2}\left(I_4 + \sum_k^9 e_{ij_o,k}\Lambda_k\right) \quad \forall (i,j) \in \mathcal{E} \tag{4.22}$$

A variant of this model exists where the global energy readout from Equation 4.20 is replaced by the energy calculation from Equation 2.10. Since the energy output depends on the generated 1-RDMs and 2-RDMs, it is logical to constrain the energy to represent the underlying physical properties of the system accurately, thus infusing our model with more physics-based constraints. We refer to this variant as EBlochGNN.

RDMNet

This architecture takes inspiration from Expectation Maximisation [54] (EM). We define two types of GNN layers meant to modify the hidden states through coarse and fine transformations, allowing us to achieve high output precision. It has been designed for generic complex-valued RDMs, but we can apply it to our dataset by assuming real-valued RDMs. Both layers are designed in the following way:

- **Hidden States** ρ: 1-RDMs $\in \mathbb{C}^{D\times D}$ and 2-RDMs $\in \mathbb{C}^{2D\times 2D}$.
- **Edge Attributes**: Interaction terms, $J \in \mathbb{R}$ (0 for non-existent edges in a non-fully-connected setting).
- **Node Attributes**: Field Effects, $h, u \in \mathbb{R}$.

The two message-passing layers are designed to bring the RDM node attributes closer to their targets. The two layers are:

1. **Rotation Layer**: This layer performs quantum state estimation by learning unitary matrix messages, which, when updating the RDM node attributes, apply a unitary transformation to align the RDMs with the target. It takes inspiration from the non-scalar messages used by Batatia et al. [51] and the unitary evolution explored by Arjovsky et al. [52].
2. **State Mixing Layer**: Unitary transformations do not modify the eigenvalues of the RDMs. For this reason, we design a layer capable of adjusting the eigenvalues of each RDM, enabling the learning of any RDM state and thereby modifying state mixing.

To instantiate this model, we can alternate the layers to achieve the desired updates to the RDMs. We will now introduce the intuition and mathematics behind these layers, starting with the Rotation Layer and State Mixing Layer, followed by an overview of a simplified Rotation Layer modification.

We will now cover each layer separately and introduce a simplified version of the rotation layer, which only works on 1-RDMs. Moreover, we will explain how we predict 2-RDMs using the RDMNet structure.

Rotation Layer for arbitrary $\mathbb{R}^{D\times D}$ The messages in this layer take the form of unitary matrices. This means that we must have a parameterised way of generating these matrices. We take inspiration from imaginary time evolution, where we create our unitary messages by performing the matrix exponential on a skew-hermitian matrix. The problem with generating this skew-hermitian matrix is that it is complex. A workaround to avoid dealing with complex model weights involves treating the imaginary and real parts separately. Of course, constraining a neural network's outputs to be skew-hermitian this way would be tedious, so we resort to the following property:

$$B = \frac{1}{2}\left(A - A^{\dagger}\right) \tag{4.23}$$

Equation 4.23 allows us to generate a skew-hermitian matrix $B \in \mathbb{C}^{D\times D}$ from an arbitrary complex matrix $A \in \mathbb{C}^{D\times D}$. We use this property to generate matrix B by parametrising matrix A through two different models, which is common practice in complex-valued neural networks.

We begin defining the input to the functions in layer l.

$$I_{ij}^{l} = \left\{\rho_i^l, \rho_j^l, J_{ij}, g_i, u_i\right\} \quad i,j \in V \wedge l \in L, \tag{4.24}$$

Where the node features for the first layer are initialised to random complex or real-valued RDMs. We then flatten the concatenated features, leading to a complex feature vector, which we split into real and imaginary parts, passing them through separate neural networks:

$$A_{ij}^{l}(I_{ij}^{l}) = \phi_{\mathbb{R}}(\mathbb{R}(I_{ij}^{l})) + i\phi_{\mathbb{C}}(\mathbb{C}(I_{ij}^{l})) \in \mathbb{C}^{D\times D} \tag{4.25}$$

where $\phi_{\mathbb{R}}$ and $\phi_{\mathbb{C}}$ are MLPs with three layers, each with SiLU activation functions. We aggregate the messages by taking the sum of the parameterised complex matrices A_{ij}^l coming to node i from all other nodes j:

$$A_i^l = \sum_{j \neq i} A_{ji}^l \in \mathbb{C}^{D \times D} \tag{4.26}$$

With our aggregated complex matrix A_i^l, we obtain our parameterised skew-Hermitian matrix by applying Equation 4.23:

$$B_i^l = \frac{1}{2}\left(A_i^l - (A_i^l)^\dagger\right) \in \mathbb{C}^{D \times D} \quad (skew - Hermitian) \tag{4.27}$$

We can now generate our unitary transformation message by performing a matrix exponential:

$$m_i^l = \exp\left\{B_i^l\right\} \in \mathbb{C}^{D \times D} \quad (Unitary) \tag{4.28}$$

Finally, with our aggregated messages, we can proceed to update our hidden state by applying the unitary transformation defined by m_i^l, which maintains the hidden state's properties, thus keeping it as a valid RDM:

$$\rho_i^{(l+1)} = m_i^l \rho_i^l (m_i^l)^\dagger \tag{4.29}$$

With the Rotation Layer defined. We need to define the second type of layer, which will perform state mixing to modify the eigenvalues of the different 1-RDMs.

State Mixing Layer This layer will focus on mixing the states of the RDM. We know that the previous layer doesn't modify their eigenvalues, so we must modify them while maintaining the RDM properties of unitary trace and PSD. This implementation relies on eigenvalue decomposition. The messages being sent represent a column vector for the eigenvalue update. We will now define the equations for this message-passing layer.

We begin defining the input to the functions in layer l, which are the same as previously.

$$I_{ij}^l = \left\{\rho_i^l \rho_j^l, J_{ij}, g_i, h_i\right\} \qquad i, j \in V \wedge l \in L \tag{4.30}$$

$$m_{ij} = \phi_{\mathsf{merge}}\left(\phi_{eig_{\mathbb{R}}}\left(\mathbb{R}\left(I_{ij}^l\right)\right) || \phi_{eig_{\mathbb{C}}}\left(\mathbb{C}\left(I_{ij}^l\right)\right)\right) \qquad m_{ij} \in \mathbb{R}^{2D} \tag{4.31}$$

where $\sum_d m_{ijd} = 1 \quad \forall i, j \in V$.

The functions $\phi_{eig_{\mathbb{R}}}$ and $\phi_{eig_{\mathbb{C}}}$ are MLPs with 3 layers each with SiLU activation functions, and ϕ_{merge} is an MLP with 2 layers with SiLU activation functions. We then aggregate the messages.

$$m_i = \sum_{i \neq j} m_{ij} \quad m_i \in \mathbb{R}^{2D} \tag{4.32}$$

After aggregation, we decompose the RDM.

$$\rho_i^l = Q_i^{(l)} \Lambda_i^l Q_i^{-1(l)} \quad Q_i \in \mathbb{C}^{D \times D}, \Lambda_i \in \mathbb{R}^D \tag{4.33}$$

We know that the eigenvalues will not be complex as the RDMs are Hermitian. We decompose the eigenvectors into their magnitude and phase to compute the updated eigenvalues.

$$\mu_i = |(Q_i)| \in \mathbb{R}^D \tag{4.34}$$

$$\theta_i = \mathsf{Arg}(Q_i) \in \mathbb{R}^{D^2 \times D^2} \tag{4.35}$$

We then pass this through a 2-layer MLP ϕ_Λ by flattening the features and concatenating them. The last layer of ϕ_Λ will be a softmax layer with output shape $\mathbb{R}^D$.

$$\Lambda_i^{l+1} = \phi_\Lambda(m_i||\Lambda_i^l||\mu_i||\theta_i) \qquad \sum_d \Lambda_{id}^{l+1} = 1 \tag{4.36}$$

With the updated eigenvalues, we can reconstruct the RDMs with the previous eigenvectors.

$$\rho_i^{l+1} = Q_i^l I \Lambda_i^{l+1} \left(Q_i^l\right)^\dagger \tag{4.37}$$

We will now define a simple rotation layer specific to the 1-RDM prediction scenario, which we can integrate into our model to reduce computational complexity.

Rotation Layer for $\mathbb{R}^{2x2}$ As the unitary operations are N-dimensional rotations, we can parameterise them with fewer degrees of freedom using Euler angles for 1-RDMs.
We begin defining the input to the functions in layer l.

$$I_{ij}^l = \left\{\rho_i^l \rho_j^l, J_{ij}, g_i, h_i\right\} \qquad i,j \in V \wedge l \in L \tag{4.38}$$

We obtain an embedding for the features, splitting the input feature's real and imaginary parts and passing them through two different networks with the same architectures as in the standard rotation layer.

$$A_{ij}^l = \phi_{\theta_\mathbb{R}}\left(\mathbb{R}\left(I_{ij}^l\right)\right) + \phi_{\theta_\mathbb{C}}\left(\mathbb{C}\left(I_{ij}^l\right)\right) \quad \in \mathbb{R}^{D^2} \tag{4.39}$$

Note that we don't multiply by I the imaginary part to avoid A_{ij}^l being complex; we also don't reshape the output to obtain a matrix. With the feature embedding, we can obtain the Euler angles representing the messages and then calculate the aggregate unitary message.

$$(\alpha||\beta||\gamma)_{ij}^l = \phi_a\left(A_{ij}^l\right), \qquad \alpha,\beta,\gamma \in [-\pi,\pi) \subseteq \mathbb{R} \tag{4.40}$$

Message aggregation is performed through a sum of the angles, scaling to ensure they remain in the $[-\pi,\pi)$ range.

$$\alpha_i^l = \sum_{j\neq i} \alpha_{ji}^l \quad \in [-\pi,\pi) \subseteq \mathbb{R} \tag{4.41}$$

$$\beta_i^l = \sum_{j\neq i} \beta_{ji}^l \quad \in [-\pi,\pi) \subseteq \mathbb{R} \tag{4.42}$$

$$\gamma_i^l = \sum_{j\neq i} \gamma_{ji}^l \quad \in [-\pi,\pi) \subseteq \mathbb{R} \tag{4.43}$$

We then calculate the aggregate unitary message.

$$m_i^l = e^{-i\alpha_i^l\sigma_z/2} e^{-i\beta_i^l\sigma_y/2} e^{-i\gamma_i^l\sigma_z/2} \quad \in \mathbb{C}^{D\times D}(Unitary) \tag{4.44}$$

Finally, with our aggregated messages, we can proceed to update our hidden state RDM by applying the transformation defined by m_i^l.

$$\rho_i^{(l+1)} = m_i^l \rho_i^l \left(m_i^l\right)^\dagger \tag{4.45}$$

Predicting 2-RDMs The previously defined layers assume that the RDMs are provided as node attributes, which easily applies to the 1-RDM case. Nevertheless, our model must predict 2-RDMs too. The current implementation performs these predictions in parallel, two stacked GNNs with rotation and mixing layers without communication between them. Nevertheless, there is much room for improvement. Communication between these layers may increase performance drastically and allow us to reduce the number of model parameters required, thus increasing efficiency.

To predict 2-RDMs, our approach involves generating a new graph from the input graph, designated as G. In this new graph, denoted as G', the number of nodes corresponds to the number of edges present in G. Specifically, for each edge e_{ij} in G, we create a new node ρ'_{ij} in G'. The attribute of ρ'_{ij} is defined by the concatenation of the edge attribute J_{ij} from G with the node attributes h_i and h_j of the connected nodes in G, formally given as $\rho'_{ij} = (J_{ij}, h_i, h_j)$. This transformation results in the new graph G' having empty edge features.

A critical point to consider in this architecture is its scalability, particularly for fully connected graphs. In a fully connected graph, the number of edges grows quadratically with the number of nodes, following an $O(N^2)$ pattern. Consequently, the number of nodes in the new graph G' also increases quadratically, posing potential scalability challenges for large graphs. This rapid increase in the size of G' can lead to computational inefficiencies, especially when dealing with dense graphs with high edge counts.

Despite this scalability challenge, the structure of G' allows the use of the same layers for predicting 2-RDMs without modifications. The concatenation process in the GNN for G' involves only the node attributes, as the edge features are non-existent. This method enables a consistent approach to prediction across 1 and 2 RDMs.

4.2.2 TN-based Approaches

This section presents three approaches, all relying on TNs to approximate the ground state energy and RDMs from the input Hamiltonian parameters. The first approach, Quantum Belief Propagation (QBP), is an unsupervised optimisation method that approximates the entire ground state of the system using TNs and BP. This allows us to obtain both the ground state energy and RDMs. The second model, Neural Enhanced QBP (NEQBP), integrates BP throughout the layers of a GNN using tensors as model weights, learning in a supervised manner. Finally, the third model, QBP-EBlochGNN, combines QBP and QBP-EBlochGNN to create a self-supervised model that uses the approximate ground state properties obtained through QBP to train BlochGNN in a supervised fashion, thereby predicting an improved version of the ground state parameters. We will now present each model's mathematical definition and intuition.

QBP

QBP is defined similarly to TN BP, as detailed in Section 2.5.2. We will start by covering the intuition behind the problem we want to solve and arriving at an update scheme that allows us to update a TN's tensors with respect to a TN's energy. Once we have this update scheme, we will further develop it to obtain a set of equations that allow us to define the entire model.

Starting with a set of Hamiltonian parameters and a randomly initialised TN, the goal is to converge the TN to the ground state corresponding to the given Hamiltonian. The TN's tensors

are treated as layer parameters with defined updates. The local belief at tensor n is:

$$B_n(\mathbf{z}_n) = \mathcal{T}^n(\mathbf{z}_n) \prod_{m \in \mathcal{N}(n)} M_{m \to n}(z_{mn}) \tag{4.46}$$

Viewing the BP update as GNN message updates, a non-linearity is required for optimising the TN:

$$f : [B_n(\mathbf{z}_n), H] \to T^n(\mathbf{z}_n) \tag{4.47}$$

The new tensors are input into the next BP round, iterating until the objective is met. To minimise energy and converge the TN to the ground state, we propose the following non-linearity:

$$\mathcal{T}_n(\mathbf{z}_n) \leftarrow \arg\min_{\mathcal{T}_n} E(\mathcal{T}_n) \tag{4.48}$$

This defines an update scheme to reduce the TN's energy. Gradient descent is used, with a learning rate η:

$$\mathcal{T}^n_{i_n, i'_n}(\mathbf{z}_n) \leftarrow \mathcal{T}^n_{i_n, i'_n}(\mathbf{z}_n) - \eta \frac{\partial E}{\partial \mathcal{T}^n_{i_n, i'_n}(\mathbf{z}_n)} \tag{4.49}$$

With an additional normalisation step to constrain the tensor's values using the Forbenius norm $|| \cdot ||_F$:

$$\mathcal{T}^n_{i_n, i'_n}(\mathbf{z}_n) \leftarrow \frac{\mathcal{T}^n_{i_n, i'_n}(\mathbf{z}_n)}{||\mathcal{T}^n_{i_n, i'_n}(\mathbf{z}_n)||_F} \tag{4.50}$$

Initially, this model was designed using a Mean Field (MF) approximation of the Hamiltonian, requiring only 1-RDMs and local Hamiltonian terms for energy calculation. This approach was too inaccurate, so we instead consider the Hamiltonian's interaction terms and 2-RDMs using Equation 2.10.

The following objects are defined:

$$\mathcal{T}^n_{i_n, i'_n}(\mathbf{z}_n) = T^n_{i_n}(\boldsymbol{\omega}_n) T_{i'_n}(\boldsymbol{\omega}'_n) \tag{4.51}$$

$$M_n(\mathbf{z}_n) = \prod_{m \in \mathcal{N}(n)} M_{m \to n}(z_{mn}) \tag{4.52}$$

$$M_{n \setminus n_k}(\mathbf{z}_{n_k n}) = \prod_{m \in \mathcal{N}(n) \setminus \{n_k\}} M_{m \to n}(\mathbf{z}_{mn}) \tag{4.53}$$

These definitions allow us to compute 1-RDMs and 2-RDMs from the converged BP messages:

$$\rho^n_{i_n,i'_n}(B_n) = \sum_{\boldsymbol{\omega}_n,\boldsymbol{\omega}'_n} T^n_{i_n}(\boldsymbol{\omega}_n) T_{i'_n}(\boldsymbol{\omega}'_n) \prod_{m\in\mathcal{N}(n)} M_{m\to n}(\omega_{mn},\omega'_{mn}) \tag{4.54}$$

$$\equiv \sum_{\mathbf{z}_n} \mathcal{T}^n_{i_n,i'_n}(\mathbf{z}_n) M_n(\mathbf{z}_n) \tag{4.55}$$

$$\begin{aligned}\rho^{(n_1,n_2)}_{i_{n_1},i'_{n_1},i_{n_2},i'_{n_2}}(B_{n_1},B_{n_2}) = &\sum_{\substack{\boldsymbol{\omega}_{n_1},\boldsymbol{\omega}'_{n_1},\\ \boldsymbol{\omega}_{n_2},\boldsymbol{\omega}'_{n_2}}} T^{n_1}_{i_{n_1}}(\boldsymbol{\omega}_{n_1}) T^{n_1}_{i'_{n_1}}(\boldsymbol{\omega}'_{n_1}) T^{n_2}_{i_{n_2}}(\boldsymbol{\omega}_{n_2}) T^{n_2}_{i'_{n_2}}(\boldsymbol{\omega}'_{n_2}) \\ &\prod_{m\in\mathcal{N}(n_1)\setminus\{n_2\}} M_{m\to n_1}(\omega_{mn_1},\omega'_{mn_1}) \\ &\prod_{m\in\mathcal{N}(n_2)\setminus\{n_1\}} M_{m\to n_2}(\omega_{mn_2},\omega'_{mn_2})\end{aligned} \tag{4.56}$$

$$\begin{aligned}\equiv &\sum_{\mathbf{z}_{n1},\mathbf{z}_{n2}} T^{n_1}_{i_{n_1}i'_{n_1}}(\mathbf{z}_{n_1}) T^{n_2}_{i'_{n_2}i'_{n_2}}(\mathbf{z}_{n_2}) \\ &\prod_{m\in\mathcal{N}(n_1)\setminus\{n_2\}} M_{m\to n_1}(\mathbf{z}_{mn_1}) \\ &\prod_{m\in\mathcal{N}(n_2)\setminus\{n_1\}} M_{m\to n_2}(\mathbf{z}_{mn_2})\end{aligned} \tag{4.57}$$

$$\equiv \sum_{\mathbf{z}_{n1}\setminus\mathbf{z}_{n2n1}} T^{n_1}_{i_{n_1}i'_{n_1}}(\mathbf{z}_{n_1}) M_{n_1\setminus n_2}(\mathbf{z}_{n_1n_2}) \tag{4.58}$$

$$\sum_{\mathbf{z}_{n2}\setminus\mathbf{z}_{n_1n_2}} T^{n_2}_{i_{n_2}i'_{n_2}}(\mathbf{z}_{n_2}) M_{n_2\setminus n_1}(\mathbf{z}_{n_2n_1}) \tag{4.59}$$

Our definition of 2-RDMs only considers RDMs for neighbouring particles in the TN. This is valid because in our datasets, for non-neighbouring particles $J_{ij} = 0$, so their RDMs do not contribute to the system's energy. Given the one and two-body RDMs, the system's energy can be calculated using Equation 2.10.

After calculating the ground state energy for the TN, we can compute the partial derivatives with respect to the TN's tensors:

$$\begin{aligned}\frac{\partial E}{\partial \mathcal{T}^{n_k}_{i_n,i'_n}(\mathbf{z}_{n_k})} &= \frac{\partial E}{\partial \rho^n}\frac{\partial \rho^n}{\partial \mathcal{T}^{n_k}_{i_n,i'_n}(\mathbf{z}_{n_k})} + \frac{\partial E}{\partial \rho^{(n_i,n_j)}}\frac{\partial \rho^{(n_i,n_j)}}{\partial \mathcal{T}^{n_k}_{i_n,i'_n}(\mathbf{z}_{n_k})} \\ &= \left(g_{n_k}\sigma^z + h_{n_k}\sigma^x\right)\frac{\partial \rho^n}{\partial \mathcal{T}^{n_k}_{i_n,i'_n}(\mathbf{z}_{n_k})} \\ &\quad + \sum_{(n_i,n_j)\in\mathcal{N}} J_{n_in_j}\,\mathrm{Tr}\left(\sigma^z\otimes\sigma^z\right)\left(\delta_{n_in_k}+\delta_{n_jn_k}-\delta_{n_in_k}\delta_{n_jn_k}\right)\frac{\partial \rho^{(n_i,n_j)}}{\partial \mathcal{T}^{n_k}_{i_n,i'_n}(\mathbf{z}_{n_k})}\end{aligned} \tag{4.60, 4.61}$$

As the BP messages have converged, we can assume:

$$\frac{\partial \rho}{\partial M_{m\to n}} = 0 \quad \forall m\to n \tag{4.62}$$

Allowing us to compute the partial derivative with respect to the 1-RDMs as:

$$\frac{\partial \rho^n}{\partial \mathcal{T}^{n_k}_{i_n,i'_n}(\mathbf{z}_{n_k})} = M_n(z_{n_k}) \tag{4.63}$$

And for the 2-RDMs:

$$\frac{\partial \rho^{(n_i,n_j)}_{z_{n1},z_{n2}}}{\partial \mathcal{T}^{n_k}_{i_n,i'_n}(\mathbf{z}_{n_k})} = \begin{cases} \sum_{\mathbf{z}_{n1},\mathbf{z}_{n2}} M_{n2\setminus n1}(\mathbf{z}_{mn_2})^T M_{n1\setminus n2}(\mathbf{z}_{mn_1})^T T^{n_2}_{i'_{n_2} i'_{n_2}}(\mathbf{z}_{n_2})^T & \text{if} n_1 = n_k \\ \sum_{\mathbf{z}_{n1},\mathbf{z}_{n2}} T^{n_1}_{i'_{n_1} i'_{n_1}}(\mathbf{z}_{n_1})^T M_{n2\setminus n1}(\mathbf{z}_{mn_2})^T M_{n1\setminus n2}(\mathbf{z}_{mn_1})^T & \text{if} n_2 = n_k \\ 0I & \text{if} n_1, n_2 \neq n_k \end{cases} \tag{4.64}$$

Arriving at the result:

$$\begin{aligned} \frac{\partial E}{\partial \mathcal{T}^{n_k}_{i_n,i'_n}(\mathbf{z}_{n_k})} &= \left(g_{n_k}\sigma^z + h_{n_k}\sigma^x\right) M_n(\mathbf{z}_{n_k}) \\ &\quad + \sum_{(n_i,n_j)\in\mathcal{N}} J_{n_i n_j} \operatorname{Tr}\left(\sigma^z \otimes \sigma^z\right)\left(\delta_{n_i n_k} + \delta_{n_j n_k} - \delta_{n_i n_k}\delta_{n_j n_k}\right) \frac{\partial \rho^{(n_i,n_j)}}{\partial \mathcal{T}^{n_k}_{i_n,i'_n}(\mathbf{z}_{n_k})} \end{aligned} \tag{4.65}$$

While the update Equation 4.49 uses first-order gradients to converge the TN to the ground state, we found it more effective to use second-order methods in practice, as they allow for faster convergence with fewer iterations. This will be covered in Section 5.3. Additionally, we rely on PyTorch's auto-differentiation engine to simplify the implementation of our method.

In order to converge the TN to the ground state, we use the change in TN energy as our convergence criteria. This is implemented through a tolerance convergence parameter, which is compared to the difference between the TN's evaluated energy and consecutive optimisation steps, allowing us to tune the desired accuracy of the method.

Given the Hamiltonian's parameters and having defined the QBP model, one can initialise a TN with random tensors and optimise it to the ground state. As discussed in Section 2.5, BP is exact in graphs without loops. Therefore, this method enables the exact ground state to be calculated in any MPS.

NEQBP

The Neural Enhanced QBP (NEQBP) model integrates BP into GNN layers, utilising tensors in a TN as model weights to enhance the model's capability to capture the interactions in quantum many-body systems. This model is inspired by Neural Enhanced TN from Satorras et al. [53], where GNNs are combined with BP, resulting in a hybrid method that leverages the prior knowledge embedded in the graphical model for improved performance and generalisation capabilities.

Our approach takes a different route. For each layer, an MLP predicts the initial PSD messages for BP, and we run a configurable number of iterations. The unconverged messages are then used to calculate the marginals and energy, which are in turn used to update the hidden states propagated throughout the model.

Figure 4.1 shows the architecture of a 2-layer NEQBP, which we will now describe mathematically for a generic L-Layer model.

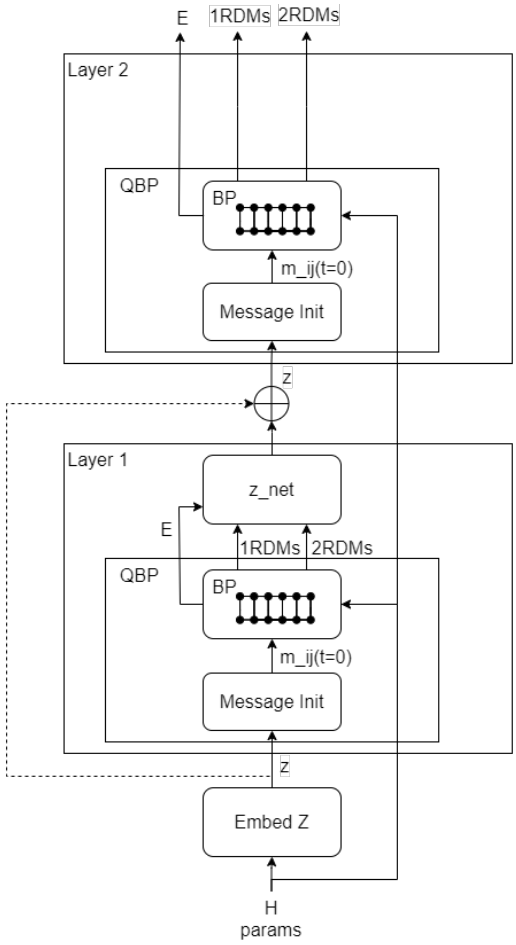

Figure 4.1: NEQBP Architecture for a 2-layer model.

Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with nodes $v_i \in \mathcal{V}$ and edges $(i, j) \in \mathcal{E}$, we define an L-layer NEQBP by first embedding the input features using a 2-layer MLP with ReLU activation functions:

$$z_i^0 = \phi_{\mathsf{embed}}(g_i, u_i, J_{ij} \forall j \in \mathcal{N}(i)), \quad z_i^0 \in \mathbb{R}^{16} \quad \forall i \in \mathcal{V} \tag{4.66}$$

$$\tag{4.67}$$

After embedding the input features, we initialize the BP messages to be PSD using the Cholesky decomposition, which states that for any symmetric, positive definite matrix B, there exists a unique lower triangular matrix A such that $B = AA^\top$. Supposing a TN with bond dimension B has been randomly initialised in each BP layer:

$$I_{ij}^l = \{z_i^l, z_j^l\}, \quad I_{ij}^l \in \mathbb{R}^{32} \quad \forall (i,j) \in \mathcal{E},\ l \in L \tag{4.68}$$

$$A_{\mathsf{flat}_{ij}}^l = \phi_m(I_{ij}^l), \quad A_{\mathsf{flat}_{ij}}^l \in \mathbb{R}^{\frac{B\cdot(B+1)}{2}} \quad \forall (i,j) \in \mathcal{E},\ l \in L \tag{4.69}$$

$$m_{ij}^{(l,t=0)} = A_{ij}^l {A_{ij}^l}^\top, \quad m_{ij}^{(l,t=0)} \in \mathbb{R}^{B\times B} \quad \forall (i,j) \in \mathcal{E},\ l \in L \tag{4.70}$$

Here, ϕ_m is a 2-layer MLP with SiLU activation functions. Moreover, the output $A_{\mathsf{flat}_{ij}}^l$ is reshaped to form a lower triangular matrix A_{ij}^l, with which we calculate our PSD initial messages.

Having calculated our initial messages for layer l, we run the configured number of BP iterations, after which we obtain intermediate values for the 1-RDMs $\rho_i^l \quad \forall i \in \mathcal{V}$, 2-RDMs $\rho_{ij}^l \quad \forall (i,j) \in \mathcal{V}$ and energy E^l. With these intermediate values, we perform aggregation and update the hidden state with an optional residual connection for improved gradient flow:

$$\mathsf{agg}_i^l = \sum_j \rho_{ij}^l \quad \forall i \in \mathcal{V},\ l \in L \tag{4.71}$$

$$z_i^{(l+1)} = \phi_z(z_i^l, \rho_i^l, \mathsf{agg}_i^l, E^l) \tag{4.72}$$

$$z_i^{(l+1)} = z_i^{(l+1)} + z_i^l \tag{4.73}$$

The function ϕ_z is implemented as a 2-layer MLP with SiLU activation functions. The last layer does not calculate z; instead, it directly uses the generated RDMs and energy from the BP layer as outputs for the model.

This model allows us to use the previously defined QBP model as a layer in a GNN, allowing us to define a supervised learning method that integrates TNs as model weights.

QBP-EBlochGNN

QBP-EBlochGNN is a semi-supervised method that combines the unsupervised QBP with the supervised EBlochGNN to create a more robust model for predicting ground state properties. Initially, the unsupervised QBP method generates an approximate dataset of ground state energy, 1-RDMs, and 2-RDMs. Then this approximate dataset is used to train the supervised EBlochGNN model, which refines and improves the predictions, resulting in a more accurate representation of the ground state parameters.

This approach is conceptually similar to Variational Autoencoders (VAEs)[55] and variational inference[56]. In VAEs, an encoder network generates an approximate latent representation of the data, which is then refined by a decoder network to produce more accurate reconstructions. Similarly, an approximate posterior distribution is iteratively refined in variational inference to better match the true posterior. QBP-EBlochGNN follows a comparable process: QBP acts as an initial approximation step (analogous to the encoder in VAEs or the initial posterior in variational inference), and BlochGNN refines this approximation to produce more accurate results (similar to the decoder in VAEs or the refined posterior in variational inference).

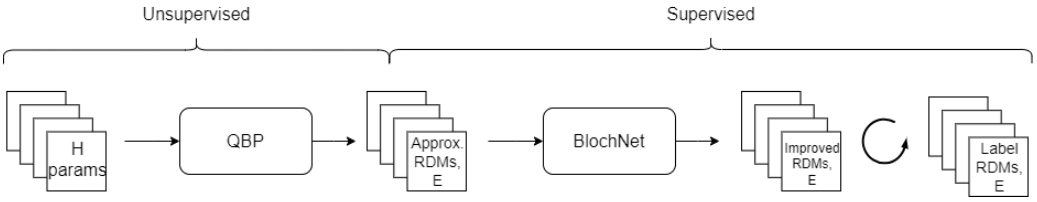

Figure 4.2: QBP-EBlochGNN architecture, highlighting the semi-supervised nature of the model.

As shown in Figure 4.2, the model starts with Hamiltonian parameters as inputs. QBP first approximates the ground state properties, generating rough estimates of 1-RDMs, 2-RDMs, and ground state energy. These approximations serve as intermediate targets for EBlochGNN. EBlochGNN then refines these targets through a supervised learning scheme, improving QBP's approximations. The QBP-EBlochGNN architecture can be adapted to use any supervised learning method instead of EBlochGNN.

Chapter 5

Experimental Setup

This chapter describes the experimental procedures used to evaluate the performance of the previously described models and approaches for predicting the ground state properties of quantum many-body systems. We will cover the dataset creation process, the baseline setup, the training procedure, and the experiments we have performed.

5.1 Datasets

We will begin explaining the different datasets we use throughout the study, outlining the data generation process. Four datasets were obtained from quantum simulations, specifically two Matrix Product States (MPS) and two Projected Entangled Pair States (PEPS). The configurations for these datasets are as follows:

1. MPS 10×1
2. MPS 20×1
3. PEPS 3×3
4. PEPS 5×4

Each dataset contains 2000 data points. The simulations used the Ising model Hamiltonian from Equation 2.9 under open boundary conditions. These specific system sizes were chosen because larger sizes would significantly increase the computational complexity of the label extraction method, making the simulations intractable within reasonable time frames and computational resources.

Each data point is defined as a graph $G = (V, E)$, where V represents the set of vertices (sites) and E represents the set of edges (interactions between sites). An edge $e_{ij} \in E$ connects vertices n_i and n_j if there is a non-zero interaction $J_{n_i n_j}$ between them. The edge index is a list of pairs of vertex indices corresponding to these edges.

The inputs for the models include Hamiltonian parameters $(J_{n_i n_j}, g_{n_i}, u_{n_i}) \in \mathbb{R}$, edge indices, and optionally estimated outputs, which are only used for the architecture explained in Section 4.2.2. The outputs consist of 1-RDMs ($\rho^{(n_i)}_{z^{n_i}} \in \mathbb{R}^{2\times 2}$), 2-RDMs ($\rho^{(n_i,n_j)}_{z^{n_i},z^{n_j}} \in \mathbb{R}^{2\times 2}$), and the ground state energy $E_0 \in \mathbb{R}$.

The dataset generation process involves several steps. First, Hamiltonian parameters, including interaction strength $J_{n_i n_j}$, transverse field g_{n_i}, and longitudinal field u_{n_j}, are sampled from a uniform distribution $[-1, 1]$. Using NetKet [57], the full Hamiltonian matrix is created with

dimensions $2^{N_x} \times 2^{N_y}$, where N_x and N_y represent the number of particles in the x and y directions, respectively. The Hamiltonian is used to find the system's ground state via exact diagonalisation to obtain the labels, as detailed in Section 2.1.6. Once the ground state is determined, the 1-RDMs and 2-RDMs are obtained by performing a partial trace, as explained in Section 2.1.5.

The data generation process ensures that the datasets are suitable for training and evaluating the models developed in this study. The chosen system sizes capture the complex interactions inherent in quantum many-body systems while also being computationally feasible to generate.

5.2 Baselines

Having explained the datasets used, we will briefly cover the baselines we use, with their set hyperparameters and usage. Specifically, two baseline methods are used to compare the performance of our proposed models: SU and DMRG.

SU is explained in Section 2.3.4 and implemented using the QUIMB framework by Gray et al. [58]. The hyperparameters used in this study include the bond dimension χ, which controls the maximum number of states kept during the TN update and is set to 15. The parameter *bond_dim* is the maximum bond dimension allowed during updates, set to 4. The number of iterations *num_iters* specifies how often the update procedure is repeated, with a value of 100. Finally, the step size τ used in the update process varies across three values: 0.1, 0.01, and 0.001.

The DMRG method explained in Section 2.3.4, is implemented using the TenPy framework by Hauschild et al. [59]. In this study, we employ the two-site DMRG approach, which optimises over two sites simultaneously, improving accuracy compared to the one-site variant. The hyperparameters include max_E_err, which defines the maximum allowable error in the energy calculation and is set to 1×10^{-10}. The maximum bond dimension χ_{max} is limited to 30, ensuring manageable computational complexity while maintaining sufficient accuracy. The last parameter, svd_min, sets the minimum singular value threshold for truncating the bond dimension during sweeping, with a value of 1×10^{-10}.

To use these baseline models, we first create a TN of the given structure for each input data point with random initial tensor values. The SU or DMRG algorithm is then run until convergence to generate a TN representation of the ground state. The RDMs are then obtained through TN contraction as explained in Section 2.3.3. Both implementations of DMRG and SU calculate the ground state energy internally.

It is important to note that these implementations are not GPU-accelerated, so all computations are performed on a CPU. This limitation impacts the computational efficiency but provides a reliable baseline for comparison with our proposed models.

5.3 Training Procedure

We will now cover the training procedure that has been followed for the upcoming experiments, covering the optimisers used, data splits, and specific hyperparameters.

In our experiments, all models except QBP are trained using the Adam optimiser [60] with a Cosine Annealing scheduler [61]. For BlochGNN, EBlochGNN and RDMNet, we set the learning

rate to 5×10^{-4} and apply a weight decay of 1×10^{-16}. On the other hand, NEQBP uses two learning rates: a higher learning rate for the tensors, with a value of 1×10^{-2} and a learning rate of 5×10^{-4} for the MLPs, both with the same weight decay of 1×10^{-16}. In addition, the TN-based methods are trained using a batch size of 1 as their implementation was not vectorised. The physics-inspired GNNs are trained using a batch size of 16.

Unlike the other models, QBP optimises the TN using the L-BFGS optimiser. L-BFGS [62], or Limited-memory Broyden–Fletcher–Goldfarb–Shanno, adapts the BFGS [63] algorithm to reduce memory usage. BFGS is a quasi-Newton method designed to solve unconstrained non-linear optimisation problems by building up an approximation to the inverse Hessian matrix. This matrix helps guide the search for optimal parameters using the curvature of the loss landscape. However, BFGS requires storing a dense matrix, which can be impractical for large-scale problems due to its memory demands.

L-BFGS addresses this limitation by maintaining only a limited amount of past information, or history, to approximate the inverse Hessian matrix, significantly reducing memory usage. In our optimisation, we set the history size to 50, which we found to provide reasonable memory usage without sacrificing convergence speed.

Additionally, L-BFGS uses a line search function to determine the optimal step size along the search direction in each iteration. We use the strong Wolfe conditions [64] for line search, which results in improved convergence rate and optimisation stability.

Finally, the dataset is divided into 70% for training, 15% for validation, and 15% held out for testing. QBP and the baseline models are evaluated on the test set, consisting of 300 data points per dataset (0.15×2000). This standardised evaluation allows for a fair comparison of model performance. The QBP-EBlochGNN method is also evaluated on the held-out test set. The approximate RDM and energy values generated by QBP are then split into training, validation and testing, with a split of 60%,20% and 20%, respectively, allowing us to validate whether the QBP approximations can be improved through supervised approaches.

5.4 Experiments

This section outlines the different experiments designed to evaluate different parts of our proposed models, helping to analyse their strengths and weaknesses across the different datasets.

Baseline Evaluation on PEPS and MPS Datasets We first evaluate the performance of the SU and DMRG algorithms on both PEPS and MPS datasets described in Section 5.1. We employ a bond dimension of 4 for SU, and for DMRG, we use a maximum χ of 30, using the same hyperparameters as highlighted in Section 5.2. This experiment provides a benchmark for comparing the performance of our proposed models.

Supervised Model depth in the 10×1 MPS Dataset In this experiment, we train and evaluate BlochGNN, EBlochGNN, RDMNet, and NEQBP (with a maximum of 1 BP iteration in each BP layer) on the 10×1 MPS dataset. We use 3, 5, 7, and 9-layer models, training each model for 60 epochs. This experiment aims to assess the performance of our models with varying depths and the impact of including energy calculation for EBlochGNN.

Unsupervised QBP Evaluation We evaluate QBP on all datasets' test sets, perform a statistical analysis of the results to assess the method's behaviour and theorise how it would scale to larger systems. We set the convergence tolerance for the energy to 1×10^{-6}, which sets the maximum precision of our results. This compromise results in a decreased execution time at the cost of precision. We also test QBP using a convergence tolerance of 1×10^{-10} to prove that the method is exact for MPS systems of any size, given enough iterations.

Supervised Evaluation of QGNNs on multiple datasets We train and evaluate BlochGNN, EBlochGNN, RDMNet, and NEQBP using 7-layer models. The models are evaluated on the 10×1, 20×1, 3×3, and 5×4 datasets, each trained for 100 epochs. This evaluation provides information about our models' scalability and generalisation capabilities across different dataset configurations. The results are then compared with the unsupervised QBP approach.

Semi-supervised QBP-EBlochGNN Evaluation The last experiment involves training our QBP-EBlochGNN approach on the 10×1, 20×1, 3×3, and 5×4 datasets, using QBP with a convergence tolerance of 1×10^{-6}. We will train 7-layer EBlochGNN for 150 epochs on the approximated RDMs and Energy and analyse any improvements compared to training the model without QBP. This experiment aims to demonstrate the effectiveness of the semi-supervised QBP-EBlochGNN method in enhancing the accuracy of ground-state property predictions. We will also replace the EBlochGNN supervised learning part with a 7-layer RDMNet to verify if the previous findings regarding their relative performances still apply.

These experiments evaluate the proposed models, allowing us to analyse their performance against the baseline methods and across varying dataset configurations. The following chapter will present the results and analysis of these experiments.

Chapter 6

Results & Analysis

This chapter presents the results of the experiments to evaluate the performance of the previously described models. Each set of results is immediately followed by its corresponding analysis.

6.1 Baseline Evaluation on PEPS and MPS Datasets

The performance of the SU and DMRG algorithms on the PEPS and MPS datasets is summarised in Table 6.1.

System	**Model**	$\mathcal{L}_{1-\text{RDM}} \downarrow$	$\mathcal{L}_{2-\text{RDM}} \downarrow$	$\mathcal{L}_E \downarrow$
3×3	SU	2.62×10^{-2}	4.18×10^{-2}	6.16×10^{-2}
5×4	SU	5.44×10^{-2}	9.02×10^{-2}	0.729
10×1	DMRG	2.77×10^{-6}	3.49×10^{-6}	7.79×10^{-8}
20×1	DMRG	2.74×10^{-6}	3.43×10^{-6}	6.68×10^{-8}

Table 6.1: Baseline results for SU and DMRG on PEPS and MPS datasets.

The results show a large difference in performance between the SU and DMRG models. The DMRG algorithm consistently outperforms SU across all metrics for the MPS datasets (10×1 and 20×1). Specifically, DMRG achieves significantly lower $\mathcal{L}_{1-\text{RDM}}$, $\mathcal{L}_{2-\text{RDM}}$ and $\mathcal{L}_E$. The poor performance of SU, especially on the 5×4 PEPS dataset, can be attributed to the limitations of the SU algorithm when dealing with larger and more complex systems. SU, with a bond dimension of 4, may not adequately capture the complex interactions present in these systems, leading to higher errors.

Overall, these results highlight the **difficulty of predicting ground state properties correctly for large systems with cycles**, which we will consider when analysing our proposed methods.

6.2 Supervised Model depth in the 10×1 MPS Dataset

The performance of various models on the 10×1 MPS dataset is summarised in Table 6.2. The models evaluated are BlochGNN, EBlochGNN, RDMNet, and NEQBP with varying layers (3, 5, 7, and 9).

Model	Layers	$\mathcal{L}_{1-\mathrm{RDM}} \downarrow$	$\mathcal{L}_{2-\mathrm{RDM}} \downarrow$	$\mathcal{L}_E \downarrow$
BlochGNN	3	**0.0334**	**0.0426**	**0.530**
	5	0.0350	0.0468	0.538
	7	0.0323	0.0468	0.550
	9	0.0323	0.0468	0.639
EBlochGNN	3	0.0104	0.0202	0.107
	5	**0.0103**	**0.0185**	**0.104**
	7	0.0136	0.0289	0.274
	9	0.4240	0.0362	0.719
RDMNet	3	0.0181	0.0452	0.0433
	5	0.0442	0.0500	0.115
	7	0.0177	0.0410	**0.0343**
	9	**0.0173**	0.0396	0.0384
NEQBP	3	0.297	0.672	0.785
	5	0.303	0.684	0.775
	7	**0.284**	**0.656**	**0.736**
	9	0.305	0.674	0.760

Table 6.2: Test metrics for various models on the 10×1 MPS dataset with different layers. Bold values highlight the lowest loss for a specific model.

Several observations can be made from the results in Table 6.2. Firstly, EBlochGNN **achieves the lowest $\mathcal{L}_{1-\mathrm{RDM}}$ and $\mathcal{L}_{2-\mathrm{RDM}}$** among the QGNN models. Despite being a simpler model, BlochGNN does not perform as well as EBlochGNN. Recall that the only difference between BlochGNN and EBlochGNN is the energy calculation step instead of a global readout. These results show how integrating physics knowledge into our model can drastically improve its performance.

Moreover, RDMNet **demonstrates the best performance in $\mathcal{L}_E$**, particularly with the 7-layer model achieving 3.43×10^{-2}, the lowest among all models. However, it is still less accurate than EBlochGNN in $\mathcal{L}_{1-\mathrm{RDM}}$ and $\mathcal{L}_{2-\mathrm{RDM}}$. We can also observe that increasing model depth in RDMNet does lead to improved performance across all objectives, whereas with BlochGNN and EBlochGNN, that is not the case.

NEQBP shows higher errors compared to BlochGNN, EBlochGNN, and RDMNet, suggesting that **the inclusion BP iterations in NEQBP does not necessarily improve performance** and might even degrade it due to the complexity added by multiple BP iterations. However, it is difficult to conclude whether this is the case as the low performance could also be attributed to implementation errors in the inclusion of TNs and TN BP in the model.

None of the QGNN models surpass the baseline DMRG results. However, EBlochGNN shows the most promise, especially with the 5-layer model, which achieves the best overall results among the QGNN models in $\mathcal{L}_{1-\mathrm{RDM}}$ and $\mathcal{L}_{2-\mathrm{RDM}}$ with RDMNet achieving the best $\mathcal{L}_E$.

In summary, while the QGNN models provide reasonable approximations, they fall short of the precision the DMRG baseline achieves. **The energy calculation in EBlochGNN enhances its accuracy**, but there is still a significant gap compared to DMRG, indicating the need for further refinement and optimisation of the GNN-based approaches.

6.3 Unsupervised QBP Evaluation

We now evaluate the performance of the unsupervised QBP model on all of the datasets, including MPS 10×1, MPS 20×1, PEPS 3×3, and PEPS 5×4. The evaluation metrics used are $\mathcal{L}_{1-\text{RDM}}$, $\mathcal{L}_{2-\text{RDM}}$, and $\mathcal{L}_E$. Figures 6.1 to 6.4 show box plots of the evaluation results, and Table 6.3 summaries the median (**M**) and mean (μ) values for each metric.

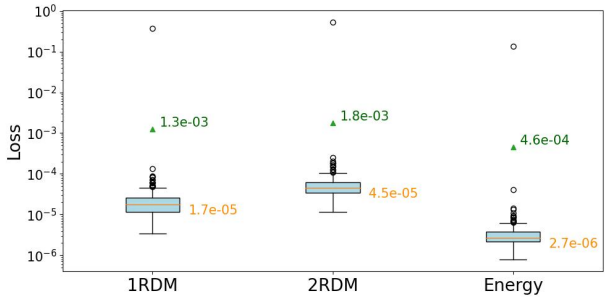

Figure 6.1: QBP Evaluation on the MPS 10×1 dataset.

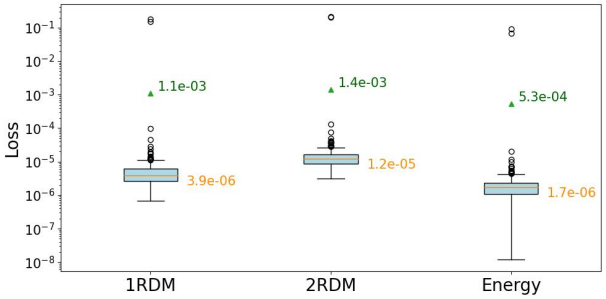

Figure 6.2: QBP Evaluation on the MPS 20×1 dataset.

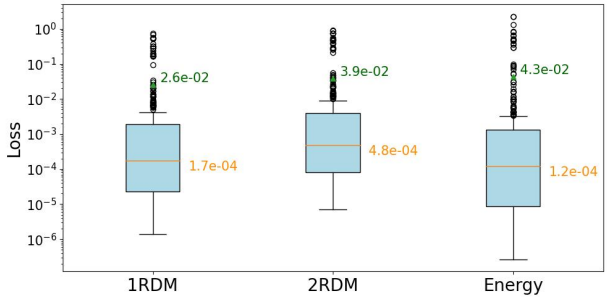

Figure 6.3: QBP Evaluation on the PEPS 3×3 dataset.

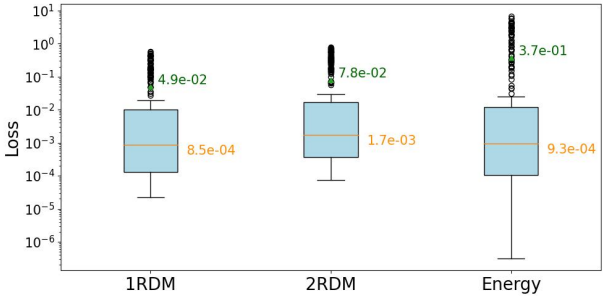

Figure 6.4: QBP Evaluation on the PEPS 5×4 dataset.

System	Metric	$\mathcal{L}_{1-\text{RDM}} \downarrow$	$\mathcal{L}_{2-\text{RDM}} \downarrow$	$\mathcal{L}_E \downarrow$
10×1	**M**	1.7×10^{-5}	4.5×10^{-5}	2.7×10^{-6}
	μ	1.3×10^{-3}	1.8×10^{-3}	4.6×10^{-4}
20×1	**M**	3.9×10^{-6}	1.2×10^{-5}	1.7×10^{-6}
	μ	1.1×10^{-3}	1.4×10^{-3}	5.3×10^{-4}
3×3	**M**	1.7×10^{-4}	4.8×10^{-4}	1.2×10^{-4}
	μ	2.6×10^{-2}	3.9×10^{-2}	4.3×10^{-2}
5×4	**M**	8.5×10^{-4}	1.3×10^{-3}	9.3×10^{-4}
	μ	4.9×10^{-2}	7.8×10^{-2}	1.7×10^{-3}

Table 6.3: Loss metrics for different system sizes with a set convergence tolerance of 1×10^{-6}. **M** represents the median and μ represents the mean.

From Table 6.3, we can see how, for the MPS datasets (10×1 and 20×1), the QBP model reaches the configured tolerance. **The median values for $\mathcal{L}_{1-\text{RDM}}$, $\mathcal{L}_{2-\text{RDM}}$, and $\mathcal{L}_E$ are very low, outperforming our supervised models**. This indicates that the QBP model is effective in these scenarios. The mean values are higher due to a few outliers, suggesting that while the model performs well on average, there are instances where the loss is significantly higher. These results could be improved by configuring the convergence tolerance to reach

machine precision.

If we analyse the box plots from Figure 6.3 and Figure 6.4, we can see how there is a noticeable spread in the loss values for the PEPS datasets compared to the MPS datasets in Figure 6.1 and Figure6.2. This spread is due to BP convergence issues in the loopy TNs. The median values for $\mathcal{L}_{1-\text{RDM}}$, $\mathcal{L}_{2-\text{RDM}}$, and $\mathcal{L}_E$ are higher than those for the MPS datasets, indicating that the QBP **model struggles to converge with machine precision in cyclic settings**. These convergence issues could potentially be reduced through dampening strategies [65][66][67], which would help to stabilise the BP updates and improve convergence.

System	Metric	$\mathcal{L}_{1-\text{RDM}} \downarrow$	$\mathcal{L}_{2-\text{RDM}} \downarrow$	$\mathcal{L}_E \downarrow$
10×1	**M**	2.5×10^{-8}	3.5×10^{-9}	4.2×10^{-10}
	μ	1.2×10^{-5}	1.7×10^{-5}	3.9×10^{-6}
20×1	**M**	3.1×10^{-8}	2.7×10^{-9}	2.1×10^{-10}
	μ	9.8×10^{-6}	1.3×10^{-5}	4.4×10^{-6}

Table 6.4: Loss metrics for MPS systems with a set convergence tolerance of 1×10^{-10}. **M** represents the median and μ represents the mean.

Compared to the baselines from Table 6.1, the QBP model does not reach DMRG's performance for MPS systems due to the configured convergence tolerance parameter. However, from the results in Table 6.4, we can see that when setting a tolerance of 1×10^{-10}, **the achieved performance is superior to DMRG**, reaching the exact values given by ED. Moreover, with a convergence tolerance of 1×10^{-6}, **we outperform SU in** 3×3 **and** 5×4 **datasets**, showing this method's potential if we resolve its convergence issues for PEPS.

In summary, the QBP model can surpass DMRG in most MPS settings when a low enough tolerance is configured at the cost of execution time. Moreover, it also surpasses the performance of the SU algorithm. This means QBP **can also be used on the baseline method's output TNs to improve their accuracy further**. However, BP's convergence criteria should be improved to make the model scalable for larger systems, reducing convergence time and enhancing performance for structures with cycles, making it a practical replacement for DMRG and SU.

6.4 Supervised Evaluation of QGNNs on Multiple Datasets

This section presents the performance evaluation of different QGNN models on multiple datasets, including MPS (10×1, 20×1) and PEPS (3×3, 5×4) systems. The evaluation metrics used are $\mathcal{L}_{1-\text{RDM}}$, $\mathcal{L}_{2-\text{RDM}}$, and $\mathcal{L}_E$. The models compared are BlochGNN, EBlochGNN, RDMNet, and NEQBP. The results are presented in Table 6.5.

Analysing the results in Table 6.5, we can see that EBlochGNN consistently achieves the lowest $\mathcal{L}_{1-\text{RDM}}$, $\mathcal{L}_{2-\text{RDM}}$, and $\mathcal{L}_E$ across all datasets and models. In comparison, BlochGNN shows higher errors across most metrics, indicating that the energy calculation step in EBlochGNN significantly improves performance. RDMNet performs better than BlochGNN but still falls short of EBlochGNN, especially in $\mathcal{L}_{1-\text{RDM}}$ and $\mathcal{L}_{2-\text{RDM}}$ metrics. Moreover, NEQBP generally shows the highest errors across all metrics and datasets, suggesting that including TN BP iterations does not necessarily improve performance and may add unnecessary complexity.

Model	System Size	$\mathcal{L}_{1-\mathrm{RDM}} \downarrow$	$\mathcal{L}_{2-\mathrm{RDM}} \downarrow$	$\mathcal{L}_E \downarrow$
BlochGNN	10×1	0.0193	0.0220	0.0182
	3×3	0.0235	0.0319	0.0229
	20×1	0.0300	0.0396	0.0528
	5×4	0.0193	0.0220	**0.0182**
EBlochGNN	10×1	**0.00954**	**0.0129**	**0.00619**
	3×3	**0.0126**	**0.0186**	**0.0212**
	20×1	**0.0107**	**0.0150**	**0.0168**
	5×4	**0.0172**	**0.0285**	0.0529
RDMNet	10×1	0.0167	0.0414	0.0352
	3×3	0.0178	0.0397	0.0365
	20×1	0.0159	0.0424	0.0441
	5×4	0.0205	0.0426	0.0600
NEQBP	10×1	0.276	0.714	0.720
	3×3	0.340	0.698	0.813
	20×1	0.309	0.686	1.130
	5×4	0.314	0.716	1.260

Table 6.5: Test metrics for QGNNs on different system sizes, with the lowest loss values for each dataset highlighted in bold.

If we recall the baseline results from Table 6.1, it can be observed how **DMRG outperforms all QGNN models** for the 10×1 and 20×1 datasets. For instance, the $\mathcal{L}_{1-\mathrm{RDM}}$ for DMRG in the 10×1 dataset is significantly lower than the best EBlochGNN result. However, for the PEPS datasets, EBlochGNN **performs better in the** 3×3 dataset but struggles with the 5×4 dataset. For the 5×4 dataset, EBlochGNN's $\mathcal{L}_{1-\mathrm{RDM}}$ is 0.0172, which is better than SU's 0.0544, but interestingly, BlochGNN has a lower $\mathcal{L}_E$ in the 5×4 dataset compared to EBlochGNN. This unexpected result for $\mathcal{L}_E$ might be due to the complexity and structure of the 5×4 dataset, where the inclusion of energy calculation in EBlochGNN might constrain the model too much, leading to overfitting or convergence issues in the presence of cycles. In contrast, the simpler approach of BlochGNN might provide a more stable estimation in such cases.

As the system size increases from 10×1 to 20×1, all models show an increase in $\mathcal{L}_{1-\mathrm{RDM}}$, $\mathcal{L}_{2-\mathrm{RDM}}$, and $\mathcal{L}_E$, indicating that predicting ground state properties becomes more challenging with larger systems. The models struggle more for the PEPS datasets (3×3 and 5×4), especially with the 5×4 dataset, likely due to the increased complexity and presence of cycles. The 5×4 dataset results show higher errors across all metrics, indicating convergence issues and the need for improved methods to handle such configurations.

In conclusion, while the QGNN models show promise, particularly EBlochGNN and RDMNet, they fall short of the precision the DMRG baseline achieves. The increased errors with larger system sizes and PEPS datasets highlight areas for further refinement and optimisation. Improving convergence strategies and incorporating more physics knowledge could enhance the performance of these models, making them more competitive with traditional methods.

6.5 Semi-supervised QBP-EBlochGNN Evaluation

The evaluation of QBP-EBlochGNN and QBP-RDMNet models on all system sizes are summarised in Table 6.6, which also presents the percentage change in all loss components compared to the supervised models trained directly without QBP, which were shown in Table 6.5.

Model	**System**	$\mathcal{L}_{1-\mathrm{RDM}}$ $\Delta\mathcal{L}_{1-\mathrm{RDM}}(\%)$ ↓	$\mathcal{L}_{2-\mathrm{RDM}}$ $\Delta\mathcal{L}_{2-\mathrm{RDM}}(\%)$ ↓	$\mathcal{L}_{E}$ $\Delta\mathcal{L}_{E}(\%)$ ↓
QBP-EBlochGNN	10×1	0.0158 66%	0.0339 163%	0.0635 926%
	3×3	0.0171 36%	0.0262 41%	0.113 433%
	20×1	0.0201 88%	0.0410 173%	0.0694 313%
	5×4	0.0191 11%	0.0391 37%	0.240 354%
QBP-RDMNet	10×1	0.0187 12%	0.0435 5%	0.0457 30%
	3×3	0.00832 **-53%**	0.0417 5%	0.0481 32%
	20×1	0.0166 4%	0.0478 13%	0.0758 72%
	5×4	0.0258 34%	0.0510 132%	0.0934 413%

Table 6.6: Experiments for the semi-supervised QBP-EBlochGNN and QBP-RDMNet for all system sizes, including the change in loss with respect to the fully supervised model results from Table 6.5. $\Delta\{\mathcal{L}_{1-\mathrm{RDM}}, \mathcal{L}_{2-\mathrm{RDM}}, \mathcal{L}_{E}\}(\%)$ represents the percentage change in loss metrics compared to evaluating the supervised models without QBP as in Section 6.4.

QBP-EBlochGNN's results show a **significant increase in errors** across all metrics and datasets compared to the supervised EBlochGNN model. Similar trends are observed for the other datasets, indicating that the supervised model did not improve the approximations obtained from the QBP approach.

QBP-RDMNet shows mixed results. While there is a notable improvement in $\mathcal{L}_{1-\mathrm{RDM}}$ for the 3×3 dataset with a decrease of 53.3%, the other metrics and datasets show an increase in errors, smaller in magnitude compared to QBP-EBlochGNN. Surprisingly, QBP-RDMNet results in a lower performance drop than QBP-EBlochGNN, which does not follow the same pattern as the results obtained in the fully supervised evaluation from Section 6.4.

When comparing the obtained results to the baseline performance in Table 6.1, we observe that both QBP-EBlochGNN and QBP-RDMNet still do not match the precision of DMRG for MPS systems. For instance, DMRG achieves a lower $\mathcal{L}_{1-\mathrm{RDM}}$ in the 10×1 dataset than any of the results from the QBP-enhanced models. The results from the QBP approach in Table 6.3 show that even without supervised fine-tuning, the QBP model achieves competitive results for MPS datasets. In addition, the performance gap between QBP-enhanced models and SU is

still present for PEPS datasets, although QBP approaches offer some improvements for smaller systems.

The results suggest that **supervised methods may not improve the approximations obtained from** QBP. The supervised fine-tuning did not lead to better performance and, in most cases, introduced additional errors. This could be due to the supervised models overfitting the approximations obtained from QBP or not generalising well due to the small size of the datasets. Therefore, further refinement and optimisation of the QBP method itself might be a more effective strategy for enhancing the accuracy of ground-state property predictions in quantum many-body systems. This would result in an improved unsupervised method and allow us to generate larger approximate RDM datasets, which could be more suitable for semi-supervised methods.

Chapter 7

Conclusion

In this research, we have sequentially implemented and evaluated six different models for predicting ground state properties of quantum many-body systems. Among these models, EBlochGNN has shown great performance and potential, closely approaching the results of established baselines. RDMNet has also demonstrated considerable promise, particularly in energy calculations. Conversely, NEQBP has resulted in poor performance, which, if further refined, could be resolved. This study highlights the significant progress made in integrating physics-inspired models with AI techniques. However, there remains a substantial gap between our results and the precision achieved by traditional methods.

Our findings, detailed in Section 6, indicate that EBlochGNN consistently achieves the lowest $\mathcal{L}_{1-\mathrm{RDM}}$, $\mathcal{L}_{2-\mathrm{RDM}}$, and $\mathcal{L}_E$ across all datasets and models. This performance underscores the effectiveness of incorporating energy calculation steps into the model. On the other hand, BlochGNN, which lacks this energy calculation step, shows higher errors across most metrics, which shows the significant impact that can be achieved by integrating more physics knowledge into the models.

Comparing the QGNN models to the baseline methods, it is evident that the DMRG algorithm outperforms all QGNN models for the MPS datasets (10×1 and 20×1). However, for the PEPS datasets, particularly in the 3×3 system, EBlochGNN performs better than the SU baseline. The increased errors observed in larger and more complex systems, like the 5×4 dataset, highlight the challenges these models face with cycles and scalability. Additionally, the unsupervised QBP evaluation shows that while QBP can achieve competitive results for MPS datasets, it struggles with PEPS due to BP convergence issues, suggesting that further refinement and optimisation of BP are necessary.

The ambitious nature of this study has provided many valuable learning opportunities across various domains, including Quantum Physics, TNs, Probabilistic Graphical Models, and Graph Neural Networks. In addition, creating a new architecture is a challenging task, and this thesis details the process of working towards innovation in a very competitive and complete field. While the reported results are not groundbreaking, they represent a step in the right direction toward creating physics-inspired models capable of performing well on both quantum-specific and generic datasets.

Our findings suggest that integrating physics knowledge into AI models can be achieved in many unexplored ways, significantly enhancing their performance. However, there is still much work to be done. We will now discuss some limitations of our study and propose directions for future research.

7.1 Limitations

One of the primary limitations of this research is the extensive background knowledge required in all the previously mentioned fields. Most of these fields were new to the researcher and required much time to understand correctly. Additionally, the data generation process using ED is not scalable to larger systems, limiting the number of qubits we could study. This restriction has implications for the generalisability of our findings to larger and more complex quantum systems.

7.2 Future Research

Future research should explore different TN arrangements, such as MERA and TTS, and investigate their performance on various system sizes, potentially even real molecules. Analysing specific Hamiltonian parameter symmetries and phase transitions could provide deeper insights into the models' capabilities. Moreover, implementing BP in Factor Graph form instead of TN form may enhance performance through computational optimisations available in frameworks like Torch Geometric. This approach would use the duality between dual TNs and Factor Graphs, as discussed by Alkabetz et al [31]. Additionally, improving the 2-RDM prediction method for RDMNet could improve its performance and time complexity, possibly outperforming BlochGNN in RDM Losses.

Further investigation into integrating BP into GNNs could yield more robust models. Additionally, testing QGNNs on generic datasets not specific to quantum systems could broaden their applicability and impact. The potential for physics-inspired models to perform well across various domains remains an exciting prospect for future exploration.

Appendix A

Code Base

The implementation of all architectures, dataset creation and evaluation pipelines can be found in the GitHub repository: `https://github.com/gerardPlanella/QGNN`.

Bibliography

[1] Silviu-Marian Udrescu and Max Tegmark. Ai feynman: A physics-inspired method for symbolic regression. *Science Advances*, 6(16):eaay2631, 2020.

[2] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *nature*, 596(7873):583–589, 2021.

[3] Keith T Butler, Daniel W Davies, Hugh Cartwright, Olexandr Isayev, and Aron Walsh. Machine learning for molecular and materials science. *Nature*, 559(7715):547–555, 2018.

[4] Tian Xie and Jeffrey C Grossman. Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties. *Physical review letters*, 120 (14):145301, 2018.

[5] Jonas Degrave, Federico Felici, Jonas Buchli, Michael Neunert, Brendan Tracey, Francesco Carpanese, Timo Ewalds, Roland Hafner, Abbas Abdolmaleki, Diego de Las Casas, et al. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602 (7897):414–419, 2022.

[6] Alexandre M Zagoskin. *Quantum theory of many-body systems*, volume 174. Springer, 1998.

[7] Per-Olov Löwdin. Quantum theory of many-particle systems. iii. extension of the hartree-fock scheme to include degenerate systems and correlation effects. *Physical review*, 97(6): 1509, 1955.

[8] Rev McWeeny. Some recent advances in density matrix theory. *Reviews of Modern Physics*, 32(2):335, 1960.

[9] Richard Bellman. Dynamic programming. *science*, 153(3731):34–37, 1966.

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

[11] Jacob Biamonte and Ville Bergholm. Tensor networks in a nutshell. *arXiv preprint arXiv:1708.00006*, 2017.

[12] Ulrich Schollwöck. The density-matrix renormalization group. *Reviews of modern physics*, 77(1):259–315, 2005.

[13] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

[14] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

[15] Shuqian Ye, Jiechun Liang, Rulin Liu, and Xi Zhu. Symmetrical graph neural network for quantum chemistry with dual real and momenta space. *The Journal of Physical Chemistry A*, 124(34):6945–6953, 2020.

[16] Zhimin He, Xuefen Zhang, Chuangtao Chen, Zhiming Huang, Yan Zhou, and Haozhen Situ. A gnn-based predictor for quantum architecture search. *Quantum Information Processing*, 22(2):128, 2023.

[17] Dmitrii Kochkov, Tobias Pfaff, Alvaro Sanchez-Gonzalez, Peter Battaglia, and Bryan K Clark. Learning ground states of quantum hamiltonians with graph networks. *arXiv preprint arXiv:2110.06390*, 2021.

[18] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[19] JM Zhang and RX Dong. Exact diagonalization: the bose–hubbard model as an example. *European Journal of Physics*, 31(3):591, 2010.

[20] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[21] Steven R White. Density matrix formulation for quantum renormalization groups. *Physical review letters*, 69(19):2863, 1992.

[22] Jacob Jordan, Roman Orús, Guifre Vidal, Frank Verstraete, and J Ignacio Cirac. Classical simulation of infinite-size quantum lattice systems in two spatial dimensions. *Physical review letters*, 101(25):250602, 2008.

[23] Glen Evenbly and Guifré Vidal. Tensor network states and geometry. *Journal of Statistical Physics*, 145:891–918, 2011.

[24] Glen Evenbly and Guifré Vidal. Algorithms for entanglement renormalization. *Physical Review B—Condensed Matter and Materials Physics*, 79(14):144108, 2009.

[25] Saeed S Jahromi and Román Orús. Universal tensor-network algorithm for any infinite lattice. *Physical Review B*, 99(19):195105, 2019.

[26] Ho N Phien, Johann A Bengua, Hoang D Tuan, Philippe Corboz, and Román Orús. Infinite projected entangled pair states algorithm improved: Fast full update and gauge fixing. *Physical Review B*, 92(3):035142, 2015.

[27] Andrew John Daley, Corinna Kollath, Ulrich Schollwöck, and Guifré Vidal. Time-dependent density-matrix renormalization-group using adaptive effective hilbert spaces. *Journal of Statistical Mechanics: Theory and Experiment*, 2004(04):P04005, 2004.

[28] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[29] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[30] Luana Ruiz, Fernando Gama, and Alejandro Ribeiro. Gated graph recurrent neural networks. *IEEE Transactions on Signal Processing*, 68:6303–6318, 2020.

[31] Roy Alkabetz and Itai Arad. Tensor networks contraction and the belief propagation algorithm. *Physical Review Research*, 3(2):023073, 2021.

[32] Julien Toulouse, Roland Assaraf, and Cyrus J Umrigar. Introduction to the variational and diffusion monte carlo methods. In *Advances in Quantum Chemistry*, volume 73, pages 285–314. Elsevier, 2016.

[33] 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.

[34] Abhinav Kandala, Antonio Mezzacapo, Kristan Temme, Maika Takita, Markus Brink, Jerry M Chow, and Jay M Gambetta. Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. *nature*, 549(7671):242–246, 2017.

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

[36] Zhuoran Qiao, Matthew Welborn, Animashree Anandkumar, Frederick R Manby, and Thomas F Miller. Orbnet: Deep learning for quantum chemistry using symmetry-adapted atomic-orbital features. *The Journal of chemical physics*, 153(12), 2020.

[37] David Pfau, James S Spencer, Alexander GDG Matthews, and W Matthew C Foulkes. Ab initio solution of the many-electron schrödinger equation with deep neural networks. *Physical review research*, 2(3):033429, 2020.

[38] Ingrid von Glehn, James S Spencer, and David Pfau. A self-attention ansatz for ab-initio quantum chemistry. *arXiv preprint arXiv:2211.13672*, 2022.

[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[40] Alexander Novikov, Dmitrii Podoprikhin, Anton Osokin, and Dmitry P Vetrov. Tensorizing neural networks. *Advances in neural information processing systems*, 28, 2015.

[41] William Huggins, Piyush Patil, Bradley Mitchell, K Birgitta Whaley, and E Miles Stoudenmire. Towards quantum machine learning with tensor networks. *Quantum Science and technology*, 4(2):024001, 2019.

[42] Hans-Martin Rieser, Frank Köster, and Arne Peter Raulf. Tensor networks for quantum machine learning. *Proceedings of the Royal Society A*, 479(2275):20230218, 2023.

[43] Chenqing Hua, Guillaume Rabusseau, and Jian Tang. High-order pooling for graph neural networks with tensor decomposition. *Advances in Neural Information Processing Systems*, 35:6021–6033, 2022.

[44] Peyman Baghershahi, Reshad Hosseini, and Hadi Moradi. Efficient relation-aware neighborhood aggregation in graph neural networks via tensor decomposition. *arXiv preprint arXiv:2212.05581*, 2022.

[45] Chengcheng Jia, Bo Wu, and Xiao-Ping Zhang. Dynamic spatiotemporal graph neural network with tensor network. *arXiv preprint arXiv:2003.08729*, 2020.

[46] Xusheng Zhao, Qiong Dai, Jia Wu, Hao Peng, Mingsheng Liu, Xu Bai, Jianlong Tan, Senzhang Wang, and S Yu Philip. Multi-view tensor graph neural networks through reinforced aggregation. *IEEE Transactions on Knowledge and Data Engineering*, 35(4):4077–4091, 2022.

[47] Matthew S Leifer and David Poulin. Quantum graphical models and belief propagation. *Annals of Physics*, 323(8):1899–1946, 2008.

[48] Subhayan Sahu and Brian Swingle. Efficient tensor network simulation of quantum many-body physics on sparse graphs. *arXiv preprint arXiv:2206.04701*, 2022.

[49] Chu Guo, Dario Poletti, and Itai Arad. Block belief propagation algorithm for two-dimensional tensor networks. *Physical Review B*, 108(12):125111, 2023.

[50] Joseph Tindall and Matthew Fishman. Gauging tensor networks with belief propagation. *SciPost Physics*, 15(6):222, 2023.

[51] Ilyes Batatia, Lars L Schaaf, Huajie Chen, Gábor Csányi, Christoph Ortner, and Felix A Faber. Equivariant matrix function neural networks. *arXiv preprint arXiv:2310.10434*, 2023.

[52] Martin Arjovsky, Amar Shah, and Yoshua Bengio. Unitary evolution recurrent neural networks. In *International conference on machine learning*, pages 1120–1128. PMLR, 2016.

[53] Victor Garcia Satorras and Max Welling. Neural enhanced belief propagation on factor graphs. In *International Conference on Artificial Intelligence and Statistics*, pages 685–693. PMLR, 2021.

[54] Todd K Moon. The expectation-maximization algorithm. *IEEE Signal processing magazine*, 13(6):47–60, 1996.

[55] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[56] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

[57] Giuseppe Carleo, Kenny Choo, Damian Hofmann, James ET Smith, Tom Westerhout, Fabien Alet, Emily J Davis, Stavros Efthymiou, Ivan Glasser, Sheng-Hsuan Lin, et al. Netket: A machine learning toolkit for many-body quantum systems. *SoftwareX*, 10: 100311, 2019.

[58] Johnnie Gray. quimb: A python package for quantum information and many-body calculations. *Journal of Open Source Software*, 3(29):819, 2018.

[59] Johannes Hauschild and Frank Pollmann. Efficient numerical simulations with tensor networks: Tensor network python (tenpy). *SciPost Physics Lecture Notes*, page 005, 2018.

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

[61] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[62] Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.

[63] Roger Fletcher. *Practical methods of optimization*. John Wiley & Sons, 2000.

[64] Philip Wolfe. Convergence conditions for ascent methods. *SIAM review*, 11(2):226–235, 1969.

[65] Pritam Som and Ananthanarayanan Chockalingam. Damped belief propagation based near-optimal equalization of severely delay-spread uwb mimo-isi channels. In *2010 IEEE International Conference on Communications*, pages 1–5. IEEE, 2010.

[66] Nevena Lazic, Brendan Frey, and Parham Aarabi. Solving the uncapacitated facility location problem using message passing algorithms. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 429–436. JMLR Workshop and Conference Proceedings, 2010.

[67] Liel Cohen, Rotem Galiki, and Roie Zivan. Governing convergence of max-sum on dcops through damping and splitting. *Artificial Intelligence*, 279:103212, 2020.