M. Sc. Thesis

Bence Bakó

Quantum-enhanced and Quantum-inspired Machine Learning Models

Supervisor:
Zoltán Zimborás
Wigner RCP

University consultant:
István Csabai
ELTE

Eötvös Loránd University

2023

Acknowledgements

I would like to thank my supervisor, Zoltán Zimborás from Wigner RCP for introducing me to the world of quantum computing, and continuously supporting me throughout these past years, including the development of this work. I would also like to express my gratitude to my internship supervisor, Zsófia Kallus for her help and guidance. I am also grateful to István Csabai for accepting the role of being my university consultant. Finally, I would like to thank Péter Hága and Dániel Nagy for their insightful comments and valuable discussions.

Abstract

Probabilistic graphical models (PGMs) are powerful tools for modeling complex relationships between variables in a system. The quantization of PGMs provides a potential way of designing quantum machine learning models, that can exploit the problem structure. We propose a problem-informed quantum circuit Born machine for generative learning in Markov random fields, that outperforms previous designs. We also give possible ways of enhancing the learning framework, including tensor network methods.

Contents

1 Introduction

The goal of artificial intelligence is to create machines and systems that can learn about the world and understand it. The former task, referred to as machine learning (ML), aims to synthesize statistical characteristics of the world through perception in a data-driven way. In particular, generative ML models are built to create realistic replicas of observational data statistically matching its information content. Recent advances in generative ML, such as the public introduction of large language models [1], push the need for theoretical advancements even further. Quantum resources, due to their inherent probabilistic nature and their high expressive power, can be used to efficiently generate samples from probability distributions of high complexity.

Quantum machine learning (QML) has received significant attention in the past years as it emerges from the intersection of quantum computing and machine learning, arguably the two most transformative and impactful technologies of the 21st century. With the potential to speed up classically intractable calculations, it is a leading candidate to demonstrate useful quantum advantage in the near term [2, 3]. The most promising approach to QML in the noisy intermediate-scale quantum (NISQ) era is that of variational quantum algorithms, or quantum neural networks. While these models are highly expressive, they have many weaknesses connected to their trainability [4], including the presence of barren plateaus [5] and poor local minima [6].

Besides trying to demonstrate advantage over classical method for specific systems, the general theoretical aspects of these algorithms are also in the focus of continuous development [7]. A promising combination of these two perspectives drives the research towards frameworks that try to exploit the expressive power of quantum models, while maintaining good trainability, that is in general specific to classical ones. In this work, we focus on two model-classes, that fall under the umbrella of such techniques: the design of restricted QML models [8] and quantum-inspired machine learning models, such as tensor networks [9, 10], but also the combination thereof [11]. Probabilistic graphical models (PGMs) make good candidates for such constructions, since they can be directly implemented on a quantum computer [12, 13] and further enhanced with quantum correlations. Furthermore, their correspondence with tensor networks allows the implementation of quantum-inspired solutions as well [14, 15].

The focus of this thesis is the area of quantum generative learning models (QGLMs), since previous studies emphasize their potential to surpass purely classical solutions [16]. One of the most prominent classes of QGLMs is the quantum circuit Born machine (QCBM) [17], first introduced with a hardware-efficient Ansatz for benchmarking quantum processing units. However, such general purpose models (like the hardware-efficient Ansatz) have poor average performance, as also stated in the well-known no-free-lunch theorem of optimization [18].

In this thesis, we introduce a novel problem-informed QCBM for generative learning in Markov random fields (MRFs), and design numerical experiments to show that it outperforms previous constructions for the given task. Our model represents a significant advancement compared to problem-agnostic ones either in the number of trainable parameters, the expressivity, or both, depending on the MRF structure. Our investigations also reveal, that the reduction in the number of parameters, according to this method, can be achieved without sacrificing the expressivity of the model. This approach also eliminates the bottleneck of learning MRFs in a classical way, since it inherits the Born rule and does not rely on computing the partition function in each training step. Since our model shows numerous similarities to QAOA circuits, it can be argued, under reasonable complexity assumptions, that this framework could lead to demonstrating quantum advantage [19].

The remaining of the thesis is structured as follows: in Chapter 2 we give a theoretical introduction to probabilistic graphical models with an emphasis on Markov random fields; Chapters 3 describes the basics of quantum-enhanced and quantum-inspired generative learning; in Chapter 4 we introduce the proposed quantum circuit Markov random field framework for binary MRFs and show numerical results, that assess its expressivity; Chapter 5 shows the generalization to non-binary MRFs through the implementation of a toy problem inspired by language modeling and gives further ways of possible improvements; finally, Chapter 6 concludes the thesis.

2 Theory of Markov Random Fields

The pervasive presence of graphs unveils a unifying language of the fundamental principles that govern the intricate complexities of natural phenomena across scales and disciplines. Markov random fields, Markov networks or undirected graphical models form a subclass of probabilistic graphical models, that describe a system of random variables having the Markov property as described by an undirected graph. This graphical description makes them suitable for various low- and mid-level modeling tasks in various subfields of science, engineering and machine learning. The applications include medical image processing [20], natural language processing [21] and social network analysis [22]. We start describing the theoretical framework by first examining the foundations of the broader class of probabilistic graphical models based on Ref. [23]. This description aims to provide a deeper insight into the conditional relationships between variables in complex systems. Ultimately, this understanding can shed light on how Markov random fields are particularly suited to model complex systems with cyclic dependencies.

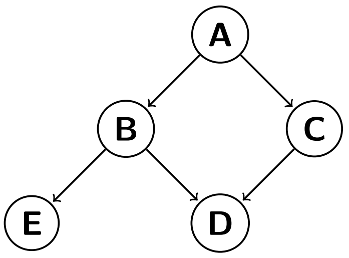
2.1 Probabilistic Graphical Models

Probabilistic graphical models (PGMs) use a graph representation to compactly encode a complex distribution of interacting random variables, that is, to represent a set of independence relationships that hold for these distributions. To explicitly encode the probabilities of each assignment in such a high-dimensional space is infeasible, as it scales exponentially with the number of variables. For example, in a space of only ten binary variables, we would need $2^{10} = 1024$ numbers to represent a probability distribution. Having independencies in the space enables to split the distribution into smaller *factors*, each over a smaller subspace.

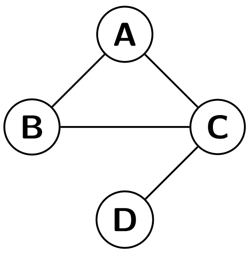
Definition 1 (Factor, scope) *Let* $\mathbf{D}$ *be a set of random variables. We define a factor* ϕ *to be a function from* $Val(\mathbf{D})$ *to* $\mathbb{R}$*, where* $Val(\mathbf{D})$ *denotes all possible joint states of the variables. The set of variables* $\mathbf{D}$ *is called the scope of the factor and denoted Scope*[ϕ].

We can get the joint distribution as the normalized product of these factors, that is, by combining the factors with the factor product.

Definition 2 (Factor product) *Let* $\mathbf{X}$, $\mathbf{Y}$, *and* $\mathbf{Z}$ *be three disjoint sets of variables, and let* $\phi_1(\mathbf{X}, \mathbf{Y})$ *and* $\phi_2(\mathbf{Y}, \mathbf{Z})$ *be two factors. We define the factor product* $\phi_1 \times \phi_2$ *to be a*



(a) Bayesian network



(b) Markov network

Figure 1: **Small examples of the two main classes of probabilistic graphical models.** The nodes represent random variables, the edges represent some dependence between them. Bayesian networks have directed edges, denoted by arrows, Markov networks have undirected edges.

factor : $\psi : Val(\mathbf{X}, \mathbf{Y}, \mathbf{Z}) \to \mathbb{R}$ *as follows:*

$$\psi(\mathbf{X}, \mathbf{Y}, \mathbf{Z}) = \phi_1(\mathbf{X}, \mathbf{Y}) \cdot \phi_2(\mathbf{Y}, \mathbf{Z}).$$

There are two main families of PGMs: Bayesian networks, that use directed acyclic graphs (Fig. 1a) and Markov networks (or Markov random fields due to historical reasons), which are defined with undirected graphs (Fig. 1b). We use the terms Markov networks (MNs) and Markov random fields (MRFs) interchangeably. In both Figs. 1a and 1b, nodes of the graphs denoted by letters A, B, C, D, E represent random variables.

These PGMs capture different types of independence relations. In Bayesian networks, a variable is independent of all other variables, given its parents. For the example depicted in Fig. 1a, the set of independencies can be written as: $(B \perp C \mid A)$, $(D \perp A \mid B, C)$, $(E \perp C, D \mid B)$. In Markov networks, the global Markov property applies, which states that any two subsets of variables are conditionally independent given a separating subset. This is equivalent to the local Markov property for positive distributions, according to which, a variable is conditionally independent of all other variables given its neighbors. In our example in Fig. 1b this translates to the independence relations: $(A \perp D \mid C)$, $(B \perp D \mid C)$.

Besides the graphical representation and the independence relations induced by the graph structure, there is a third perspective on PGMs, the induced factorization of the joint probability distribution. This factorization in the Bayesian example can be written as:

$$P(A,B,C,D,E) = P(A)P(B|A)P(C|A)P(D|B,C)P(E|B). \tag{1}$$

For the Markov network it can take different forms, the factorization according to the maximal cliques can be written as:

$$P(A,B,C,D) = \frac{1}{Z}\phi_1(A,B,C)\phi_2(C,D), \tag{2}$$

where Z is the partition function, ϕ_1 and ϕ_2 are the factors of the two maximal cliques of the graph. The role of all these will become clear in the next section.

It turns out, that the graphical representation and the conditional independence relations are equivalent, and they provide the skeleton for factorizing the distribution.

This framework, we described, has many advantages, the first one being that it allows the tractable description of probability distributions (*representation*). Second, we can answer queries using the distribution as our model of something in real life (*inference*). Third, it supports a data-driven approach, making it suitable for *learning* from data.

From now on, we will concentrate on Markov random fields, also restricting our attention to positive probability distributions over discrete state spaces.

2.2 Representation of Markov Random Fields

As opposed to Bayesian networks, where the factors are straightforward to comprehend, for Markov random fields, these cannot be interpreted directly. The concept of a factor introduced in Def. 1 describes a general-purpose function, which for BNs describes probabilities or conditional probabilities, but this is not the case for MRFs. However, we can view a factor as describing "compatibilities" between different values of the variables in the corresponding scope.

A possible factor description for the MRF in Fig. 1b is presented in Tab. 1, where we considered the maximal clique factorization of binary variables. Cliques in an undirected graph correspond to fully-connected subgraphs, the smallest cliques being pairs of nodes connected by an edge, thus we can consider different sets of cliques. By maximal clique factorization, we refer to a factorization according to the largest possible cliques. Here we also restricted the model to binary variables, but in general, they can have an arbitrary number of states.

$\phi_1(A,B,C)$				$\phi_2(C,D)$		
a^0	b^0	c^0	30	c^0	d^0	10
a^0	b^0	c^1	1	c^0	d^1	1
a^0	b^1	c^0	5	c^1	d^0	1
a^0	b^1	c^1	15	c^1	d^1	15
a^1	b^0	c^0	1			
a^1	b^0	c^1	20			
a^1	b^1	c^0	5			
a^1	b^1	c^1	10			

Table 1: Possible maximal clique factors for the MRF in Fig. 1b.

We can get the joint probability distribution by taking the product of these factors using Def. 2 and normalizing by the partition function. In this sense, the joint distribution can be thought of as a Gibbs distribution.

Definition 3 (Gibbs distribution) *A distribution P_Φ is a Gibbs distribution parametrized by a set of factors $\Phi = \{\phi_1(\mathbf{D}_1), \dots \phi_K(\mathbf{D}_K)\}$ if it is defined as follows:*

$$P_\Phi(X_1, \dots, X_N) = \frac{1}{Z}\tilde{P}_\Phi(X_1, \dots, X_N) = \frac{1}{Z}\phi_1(\mathbf{D}_1) \times \phi_2(\mathbf{D}_2) \times \cdots \times \phi_m(\mathbf{D}_m),$$

where $\tilde{P}_\Phi$ is the unnormalized measure and

$$Z = \sum_{X_1,\dots,X_n} \tilde{P}_\Phi(X_1, \dots X_n)$$

is the normalizing constant or partition function.

From the factors in Tab. 1 we can calculate the measures by multiplying certain rows of the two factor-tables according to the global assignments. From this, the corresponding probability distribution is obtained by normalization. This discrete probability distribution is presented in Fig. 2.

We can directly relate a graph structure to the parametrization of this Gibbs distribution. If there is a factor whose scope contains both X and Y, then the underlying graph structure contains an edge between the two nodes representing the variables X and Y.

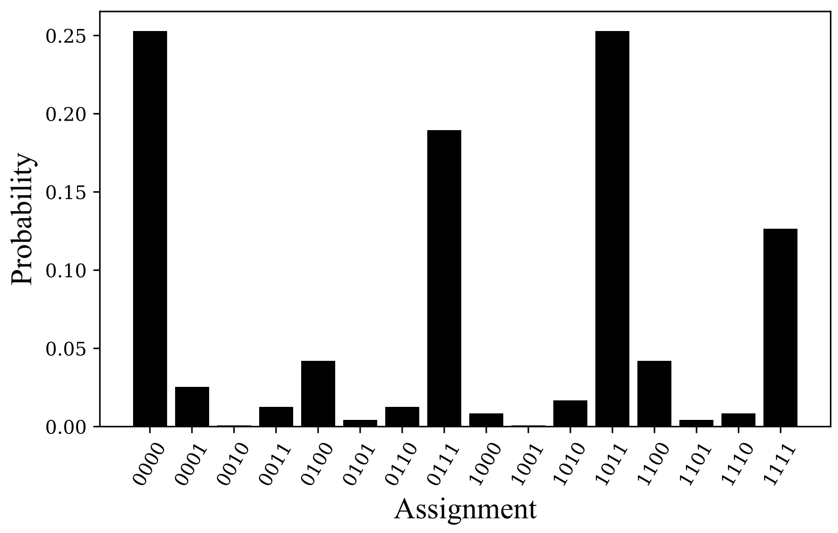


Figure 2: **Probability distribution generated by the factors in Tab. 1.** The assignments refer to the possible combinations of the binary variables A, B, C, D. The probabilities are calculated by multiplying the corresponding columns in the factor-tables, and normalizing them.

Definition 4 (Markov network factorization) *We say that a distribution* P_Φ *with* $\Phi = \{\phi_1(\mathbf{D}_1), \ldots, \phi_K(\mathbf{D}_K)\}$ *factorizes over a Markov network* $\mathcal{H}$ *if each* $\mathbf{D}_k$ $(k = 1, \ldots, K)$ *is a complete subgraph of* $\mathcal{H}$*.*

The factors that parametrize a Markov network are called clique potentials. We can reduce the number of factors by allowing only maximal cliques, and this parametrization can be used without loss of generality. However, it usually obscures structure that is present in the original set of factors, requiring possibly more (not necessarily independent) parameters to describe. In learning scenarios, assuming maximal cliques increases the number of trainable parameters at scale.

Besides encoding factors as complete tables over their scopes, there is an alternative parametrization, that converts these factors into log-space. This representation is extremely important for the quantized versions of MRFs.

Definition 5 (Log-linear model) *A distribution* P *is a log-linear model over a Markov network* $\mathcal{H}$ *if it is associated with:*

- *a set of features* $\mathcal{F} = \{f_1(\mathbf{D}_1), \ldots, f_k(\mathbf{D}_k)\}$*, where each* $\mathbf{D}_i$ *is a complete subgraph in* $\mathcal{H}$*,*

- *a set of weights* $w_1, \dots, w_k$*,*

such that

$$P(X_1, \dots, X_n) = \frac{1}{Z} \exp\left[-\sum_{i=1}^{k} w_i f_i(\mathbf{D}_i)\right].$$

In general, the log-linear model provides a more compact representation for many distributions.

Pairwise Markov Random Fields

A popular subclass of Markov random fields is that of pairwise MRFs, representing distributions where all the factors involve single variables or pairs of variables, that is to describe a system with only pairwise interactions.

Definition 6 (Pairwise Markov random fields) *A pairwise Markov random field over a graph* $\mathcal{G}$ *is associated with:*

- *a set of node potentials* $\{\phi(X_i) : i = 1, \dots, n\}$*;*
- *and a set of edge potentials* $\{\phi(X_i, X_j) : (X_i, X_j) \in \mathcal{G}\}$*.*

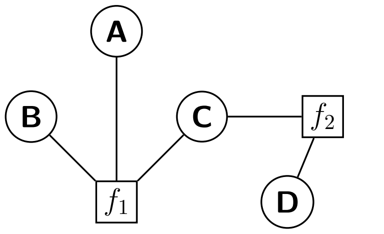
The log-linear model of a pairwise Markov network is the well-known Ising model with only one- and two-body terms.

Factor graphs

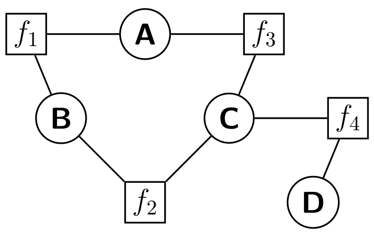
In general, the graph representation of an MRF does not reveal all the structure in the Gibbs parametrization. In particular, only looking at the graph, one does not know whether factors involve maximal cliques or their subsets. So far we considered our example to factorize according to the maximal cliques, but we can also consider the same structure with factors involving only two nodes, thus making it a pairwise MRF.

The graphical representation, that makes the clique factorization explicit, is called a *factor graph*. In a factor graph, there are two types of nodes: variable nodes and factor nodes. The graph contains edges only between variable nodes and factor nodes. This is parametrized by associating each factor node to a factor.

The two possible factor graphs for the example MRF are depicted in Fig. 3.



(a) Maximal clique factorization



(b) Pairwise MRF

Figure 3: **The two possible factor graphs for the same MRF graph in Fig. 1b.** Round nodes denote the variable nodes and the squares represent the factor nodes.

2.3 Learning in Markov Random Fields

MRFs have a wide range of applications in machine learning, involving different learning tasks, like density estimation, classification, or knowledge discovery. These can involve both generative and discriminative training. We focus our attention on generative learning tasks.

Let us assume some underlying distribution P^* that is induced by the MRF $\mathcal{H}$. We are given a dataset $\mathcal{D} = \{\mathbf{d}[1], \dots, \mathbf{d}[M]\}$ of M samples from P^*. A standard assumption is that the data instances are independently and identically distributed (IID). We consider learning the parameters of the model $\mathcal{M}$, such that the distribution it defines $P_{\mathcal{M}}$ is a good approximation of P^*.

To obtain an accurate representation of a high-dimensional distribution involving a large number of variables, we would need an exponential number of samples from P^*. In practice, this is not achievable. To this end, the goodness of $\mathcal{M}$ in approximating the target distribution can be defined in many ways.

Learning can have various different goals, but in the context of generative learning, the goal is to be able to produce new samples given the trained model.

We consider the task of learning the distribution of an MRF with known structure, which is a numerical optimization problem. With this task, we can assess the expressivity of different models.

Problem 1 (Distribution Learning in MRFs) *Given the graph structure and the clique factorization of a Markov random field $\mathcal{H}$ with an underlying distribution P^*, a dataset $\mathcal{D}$ sampled from P^* and $\varepsilon, \delta \in (0,1)$, output with probability at least $1-\delta$ a representation of a distribution $P_{\mathcal{M}}$ satisfying $TV(P^*, P_{\mathcal{M}}) \leq \varepsilon$.*

Here $TV(P,Q)$ denotes the total variational distance between $P,Q : \{0,1\}^n \to [0,1]$ defined as:

$$TV(P,Q) = \frac{1}{2} \sum_{x \in \{0,1\}^n} |P(x) - Q(x)|. \tag{3}$$

3 Quantum Generative Learning

Quantum generative learning models (QGLMs) based on quantum neural networks (QNNs) are expected to speed up the training of generative models, as they usually provide better expressivity [24]. The most important classes of such models include quantum circuit Born machines [17], quantum generative adversarial networks (QGANs) [25], quantum Boltzmann machines (QBM) [26] and quantum variational autoencoders (QVAE) [27].

While QNNs are highly expressive, they have other challenges connected to trainability. This drives the research towards restricted and structured QGLMs that try to find a balance between expressivity and trainability. These models also include the quantized versions of PGMs, like Bayesian quantum circuits [28] or unitary embedded MRFs [13].

3.1 Quantum Computing

Before we dive deeper into sophisticated QGLMs, we should review the basics of quantum computing that enables the understanding of these models. In quantum computing, information is represented in the form of quantum states $|\psi\rangle$, that live in the tensor product Hilbert-space $\mathcal{H}^{\otimes n}$ of n quantum bits or qubits [29]. Unlike classical bits that can represent either 0 or 1, qubits can exist in any *superposition* of the two basis states,

$$|\phi\rangle = \alpha|0\rangle + \beta|1\rangle, \tag{4}$$

where $\alpha, \beta \in \mathbb{C}$ such that $|\alpha|^2 + |\beta|^2 = 1$. Consequently, we can compose n qubits and get an n-qubit state as:

$$|\psi\rangle = \bigotimes_{i=0}^{n-1} (\alpha_i|0\rangle + \beta_i|1\rangle). \tag{5}$$

These, however, form just a small subset of n-qubit states, the pure product states. A generic n-qubit state can be expressed as a normalized linear combination of the computational basis states $|b_0 b_1 ... b_{n-1}\rangle := |b_0\rangle \otimes |b_2\rangle \otimes ... \otimes |b_{n-1}\rangle$,

$$|\psi\rangle = \sum_{\boldsymbol{b} \in \{0,1\}^{\times n}} \alpha_{\boldsymbol{b}} |b_0 b_1 ... b_{n-1}\rangle. \tag{6}$$

Superposition plays a crucial role in quantum computing, since it enables quantum algorithms to explore a vast number of possibilities, potentially leading to exponential speedup compared to classical computation for certain tasks.

These computations that take an initial state $|\psi_{in}\rangle$ to an output state $|\psi_{out}\rangle = C|\psi_{in}\rangle$ are usually achieved by the application of a unitary quantum circuits acting on n qubits. Formally, an operator is unitary if its adjoint is equal to its inverse: $U^\dagger U = UU^\dagger = I$, where I denotes the identity operator.

In practice, these n-qubit unitaries are formulated with one- and two-qubit gates, that are composed via matrix multiplication and tensor-product. In our context, the most important one qubit gates are the Pauli-matrices:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \quad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \quad I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{7}$$

Another fundamental operation in quantum computing is the Hadamard gate, that is, the linear combination of the Pauli-X and Pauli-Z matrices, $H = \frac{1}{\sqrt{2}}(X + Z)$.

The Pauli-matrices are not only unitary, but also Hermitian and traceless, thus they represent observable quantities, that we can measure, and they generate unitary evolutions in quantum systems. In particular, they generate the so-called rotation gates, which are very important in Ansatz construction for QML. A single qubit $R_Z(\theta)$ gate can be thought of as a θ angle rotation on the Bloch sphere around the Z axis, that can be written as

$$R_Z(\theta) = e^{-i\theta Z/2} = \cos(\theta/2)I - i\sin(\theta/2)Z = \begin{bmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{bmatrix}. \tag{8}$$

Another important property of quantum mechanics is *entanglement*, that describes quantum correlations between two or more systems, known to be stronger than any classical correlation. Entangled states can no longer be written as a tensor product of their subsystems, like Eq. (5), rather as a generic linear sum of the form of Eq. (6). In quantum circuits, these type of correlations are implemented with the help of two-qubit gates. The most important two-qubit gate is the controlled-not or $CNOT$ gate:

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{9}$$

A basic example of entangled states is a Bell state

$$|\phi^+\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}, \tag{10}$$

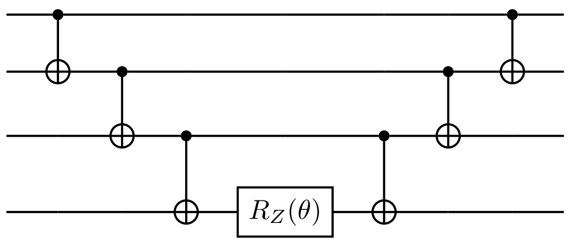


Figure 4: **Decomposition of the 4-body term** $\exp\left(-i\theta ZZZZ/2\right)$**.** The circuit decomposition contains 6 $CNOT$ gates and a one-qubit Z rotation gate.

that can be obtained by applying a Hadamard and a $CNOT$ gate on the product state $|0\rangle \otimes |0\rangle$, i.e., $|\phi^+\rangle = CNOT(H \otimes I)|0\rangle \otimes |0\rangle$. In this work, we will mostly deal with m-qubit Z interactions, that can be decomposed into $2(m-1)$ $CNOT$ gates and a single qubit R_Z rotation as shown in Fig. 4.

Quantum circuits do not grant access to the raw probabilities, instead we can perform *measurements* of a given observable, that collapses the quantum state to one of the eigenstates of this observable. To get a whole probability distribution, we have to run the circuit multiple times and perform the same measurement. These repetitions are usually referred to as shots. Most quantum algorithms use measurement in the computational basis, that is, they measure the Pauli-Z matrix on each qubit. Having a circuit $C = U_1 U_2 \dots U_k$, the probability of measuring the basis state corresponding to bitstring $|x\rangle$ is given by the Born rule:

$$P(x) = |\langle x|\psi_{out}\rangle|^2 = |\langle x|C|\psi_{in}\rangle|^2. \tag{11}$$

3.2 Quantum Circuit Born Machine

Born machines, first introduced in [17], are generative quantum machine learning models, that naturally inherit the Born rule and thus can be used to generate tunable and discrete probability distributions that approximate a target distribution. They have both quantum-inspired classical and quantum circuit implementations, the latter called quantum circuit Born machines (QCBMs). The general framework for data-driven quantum circuit learning in QCBMs is shown in Fig. 5. Instead of giving a general introduction to QML, we introduce the relevant concepts and techniques in the context of the QCBM.

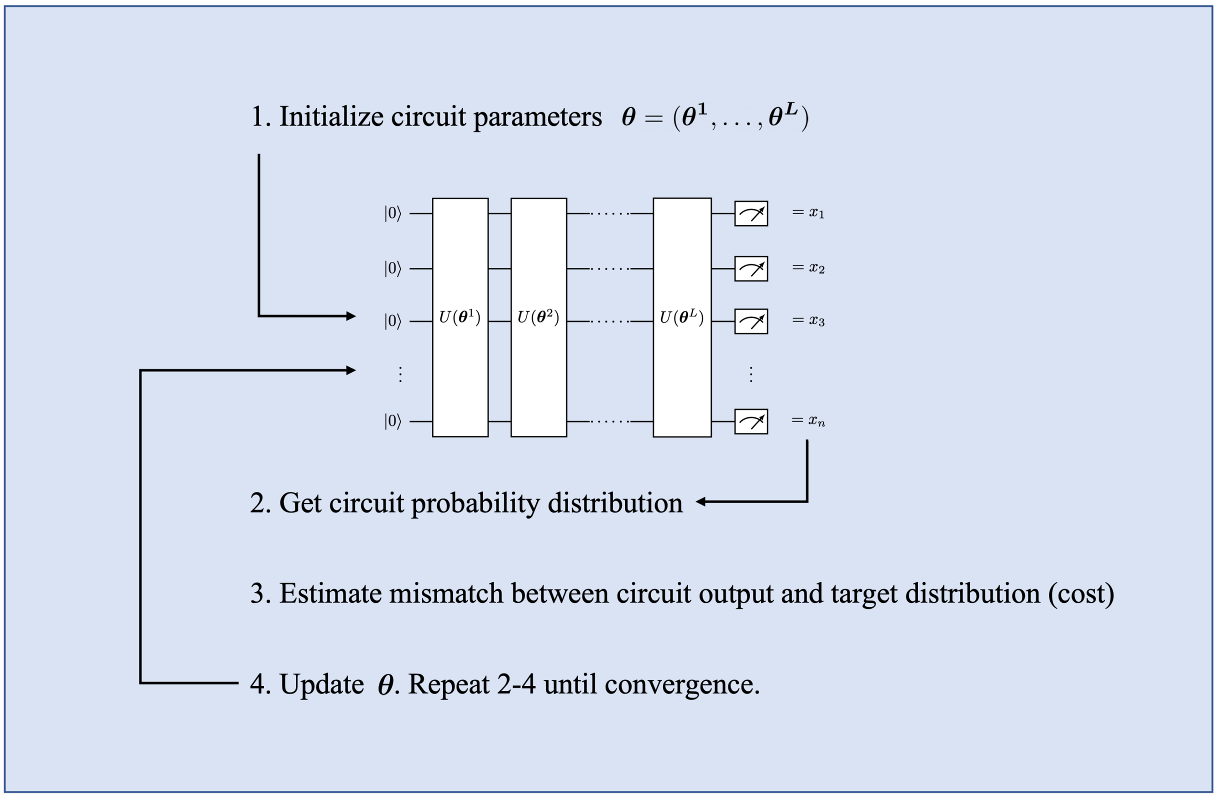

1. Initialize circuit parameters $\boldsymbol{\theta} = (\boldsymbol{\theta}^1, \ldots, \boldsymbol{\theta}^L)$
$|0\rangle$
$U(\boldsymbol{\theta}^1)$
$U(\boldsymbol{\theta}^2)$
$U(\boldsymbol{\theta}^L)$
$= x_1$
$= x_2$
$= x_3$
$= x_n$
2. Get circuit probability distribution
3. Estimate mismatch between circuit output and target distribution (cost)
4. Update $\boldsymbol{\theta}$. Repeat 2-4 until convergence.

Figure 5: **General QCBM learning framework adapted from [17].** The trainable parameters are adjusted to better approximate the target distribution.

3.2.1 Ansatz design in QML

Like in all QML models, one of the key building blocks of QCBMs is the *Ansatz*. The Ansatz refers to a parametrized family of quantum circuits used as a hypothesis or approximation for solving the problem in question. The Ansatz represents a trial solution, formulated as a variational quantum circuit (VQC) or quantum neural network (QNN), that is iteratively optimized to match the desired target behavior or to minimize a cost function.

Similarly to classical deep neural networks, the Ansatz $U(\boldsymbol{\theta})$ usually obeys a layered structure:

$$U(\boldsymbol{\theta}) = U_L(\boldsymbol{\theta}^L) \ldots U_2(\boldsymbol{\theta}^2) U_1(\boldsymbol{\theta}^1), \tag{12}$$

where L is the number of layers. These layers, represented by the parametrized n-qubit unitaries, are usually decomposed into a set of (parametrized) one- and two-qubit gates. These parameters are trainable, adjusted along the quantum circuit training. We can differentiate Hamiltonian-informed and Hamiltonian-agnostic Ansätze. For the former, the Hamiltonian generates the time evolution, that the Ansatz implements or approximates

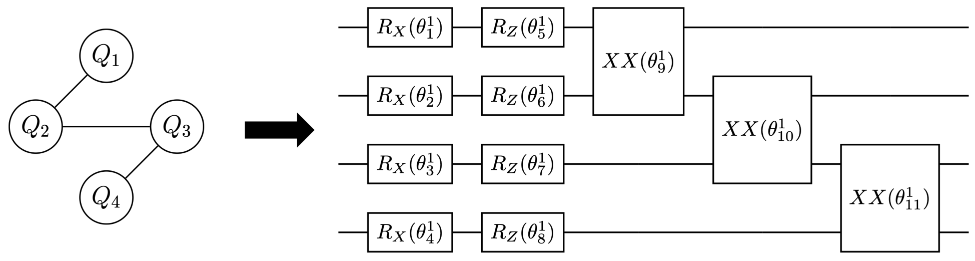


Figure 6: **Qubit connectivity and the first layer of the corresponding hardware-efficient QCBM Ansatz.** The 4 qubits have linear nearest neighbor topology, and the entangling pattern of the Ansatz respects this connectivity.

as $U(\boldsymbol{\theta}) = \exp(-iH(\boldsymbol{\theta}))$.

The hardware-efficient (thus Hamiltonian-agnostic) Ansatz introduced for QCBMs in the pioneering study of Benedetti et al. [17] is widely used in subsequent works. The layout of this model circuit is inspired by the gates readily available in ion-trap quantum computers composed of single qubit rotations and Mølmer-Sørensen XX entangling gates. It is also hardware-efficient in the sense, that the entangling pattern of the two-qubit gates respects the physical connectivity of the qubits in the underlying quantum processor. The first layer of a 4-qubit hardware-efficient Ansatz is shown in Fig. 6. Here the qubits have linear nearest neighbor topology and the two-qubit gates in the Ansatz are implemented accordingly. This original QCBM was introduced along with some specific learning problems for benchmarking the performance of quantum devices.

Quantum Circuit Ising Born Machine

In [30] Coyle et al. introduced a specific Hamiltonian-informed Ansatz coined quantum circuit Ising Born machine (QCIBM) with the general structure depicted in Fig. 7. The corresponding unitaries can be written as

$$U_z(\boldsymbol{\alpha}) = \prod_j U_z(\alpha_j, S_j) = \prod_j \exp\left(i\alpha_j \bigotimes_{k\in S_j} Z_k\right), \tag{13}$$

$$U_f(\boldsymbol{\Gamma}, \boldsymbol{\Delta}, \boldsymbol{\Sigma}) = \exp\left(i\sum_{k=1}^{n}(\Gamma_k X_k + \Delta_k Y_k + \Sigma_k Z_k)\right). \tag{14}$$

Here each S_j indicates a subset of qubits and the X, Y, Z operators are the Pauli-matrices. The U_f operators can also be thought of as a "parametrized measurement", or letting the measurements be in any local basis. The authors restrict the Hamiltonian, that generates

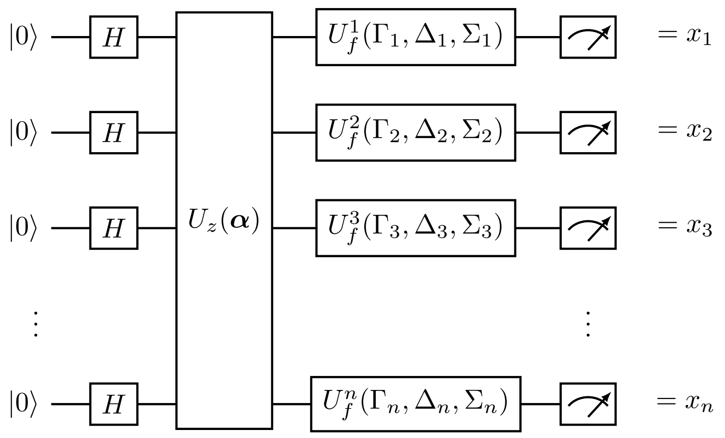

$|0\rangle$
H
$U_z(\boldsymbol{\alpha})$
$U_f^1(\Gamma_1, \Delta_1, \Sigma_1)$
$U_f^2(\Gamma_2, \Delta_2, \Sigma_2)$
$U_f^3(\Gamma_3, \Delta_3, \Sigma_3)$
$U_f^n(\Gamma_n, \Delta_n, \Sigma_n)$
$= x_1$
$= x_2$
$= x_3$
$= x_n$

Figure 7: **Quantum circuit Ising Born machine.** The local measurements are done in the computational basis, the output is a binary string of variables $x_i \in \{0, 1\}$.

U_z to only contain one- and two-body terms, since "only single and two-qubit gates are required for universal quantum computation" and they consider each pair.

They also make a connection to the $p = 1$ layered quantum approximate optimization algorithm (QAOA): by setting $\boldsymbol{\Delta} = 0$ and $\boldsymbol{\Sigma} = 0$, we get a QAOA circuit. Since simulating the output distribution of a QAOA with even just one layer is hard classically [19], this could lead to a potential way of demonstrating quantum advantage.

3.2.2 Possible cost functions

The goal of generative learning is to draw samples from a probability distribution $P_{\boldsymbol{\theta}}$, that is sufficiently close to the target distribution P^*, while only having access to a finite number of samples from P^*.

In the case of QCBMs, the final measurements are done in the computational basis on each qubit, producing a binary string of variables $x_i \in \{0, 1\}$ from the distribution

$$P_{\boldsymbol{\theta}}(\mathbf{x}) = |\langle \mathbf{x} | U(\boldsymbol{\theta}) | 0 \rangle^{\otimes n}|^2, \tag{15}$$

where the $\boldsymbol{\theta}$ vector contains all circuit parameters.

There are multiple ways to characterize the distance of two distributions. We already introduced the total variational distance (TV) in Eq. (3), which is a good benchmark, but it is not feasible as a cost function. One of the most common cost functions used in generative modelling is the Kullback-Leibler (KL) divergence:

$$D_{KL}(P_{\boldsymbol{\theta}}, P^*) = \sum_x P^*(x) \log \left(\frac{P^*(x)}{P(x)} \right). \tag{16}$$

This is still relatively strong as it upper bounds TV through Pinsker's inequality:

$$TV(P_{\boldsymbol{\theta}}, P^*) \leq \sqrt{\frac{1}{2} D_{KL}(P_{\boldsymbol{\theta}}||P^*)}. \tag{17}$$

In practice, however, this function blows up if there are $P_{\boldsymbol{\theta}}(x) = 0$ values, presenting a need to be modified in order to fit realistic training schemes. Consequently, most works use the negative log-likelihood (NLL) loss function, that is a practical version of the KL divergence:

$$\mathcal{L}(\boldsymbol{\theta}) = -\frac{1}{N} \sum_{i=1}^{N} \log\left(\max\left[\epsilon, P_{\boldsymbol{\theta}}(x_i)\right]\right), \tag{18}$$

where N denotes the number of samples in the training dataset and $\epsilon << 1$ is an infinitesimal constant.

Unfortunately, not even this function is optimal for gradient based training, as it requires a large number of training samples. In [31] an efficient cost function was introduced for gradient based training of QCBMs. The idea was to compare the distance of the samples drawn from the target and the model distribution in a kernel feature space. This loss function is called the squared maximum mean discrepancy (MMD):

$$\begin{aligned}\mathcal{L}_{\mathrm{MMD}}(P_{\boldsymbol{\theta}}, P^*) &= \left\| \sum_x P_{\boldsymbol{\theta}}(x)\phi(x) - \sum_x P^*(x)\phi(x) \right\|^2 \\ &= \underset{x \sim P_{\boldsymbol{\theta}}, y \sim P_{\boldsymbol{\theta}}}{\mathbb{E}}[K(x,y)] - 2 \underset{x \sim P_{\boldsymbol{\theta}}, y \sim P^*}{\mathbb{E}}[K(x,y)] + \underset{x \sim P^*, y \sim P^*}{\mathbb{E}}[K(x,y)],\end{aligned} \tag{19}$$

where the ϕ function maps x into a higher dimensional space and by definition $K(x, y) = \phi(x)^T \phi(x)$ is the kernel function, for which they used a Gaussian kernel:

$$K_{x,y} = \exp\left(-\frac{1}{2\sigma}|x - y|^2\right). \tag{20}$$

3.2.3 Optimizers and training

Having a family of parametrized circuits and a cost function to characterize the distance between the circuit output and the target distribution, the final building block is the optimizer, that adjusts the circuit parameters given the cost function. Both gradient-based and gradient-free optimizers have been used to train QCBMs. We will concentrate on gradient based optimization methods, since they are more capable of dealing with large-scale QNNs and are less sensitive to system noise. However, these methods require differentiating the output of the circuit with respect to the circuit parameters, that can present a challenging endeavor.

The parameter-shift rules introduced in [32] and extended in [33] provide a recipe for getting partial derivatives by evaluating parameter-shifted instances of a variational circuit. As an example, let us consider a quantum circuit with a single parametrized gate, that is a single qubit rotation. In this case, the derivative can be expressed in analytical form as

$$\nabla_\theta f(\theta) = \frac{1}{2}\left[f(\theta + \frac{\pi}{2}) - f(\theta - \frac{\pi}{2}) \right], \tag{21}$$

where $f(\theta)$ denotes the quantum function that describes the model. This is the only parameter shift rule we need for the gradient-based optimization of QCBMs, since all multi-qubit gates can be decomposed into parametrized single qubit rotations and $CNOT$ gates.

There also exist more sophisticated techniques, like quantum natural gradients, that take into account the quantum geometric structure of the parameter space by incorporating the quantum Fisher information matrix [34]. When we are classically simulating quantum circuits for benchmarking quantum models, both of these methods are unnecessarily intensive computationally. In these cases, the Pennylane software library [35] provides more efficient gradient methods, like adjoint differentiation and backpropagation.

The most popular gradient-based optimizers include the Adam [36] optimizer and the BFGS algorithm [37], that have also been used for training QCBMs. In this work, we use Adam, which is a stochastic gradient descent method that uses the adaptive estimates of first- and second-order moments.

3.3 Quantum Circuits for PGMs

In this section, we present the current state-of-the-art implementations of probabilistic graphical models on a quantum computer, focusing on the unitary embedding of Markov random fields.

3.3.1 Bayesian quantum circuits

Bayesian networks have an equivalent formulation in the computational basis measurements of a class of quantum circuits known as Bayesian quantum circuits (BQCs) [28].

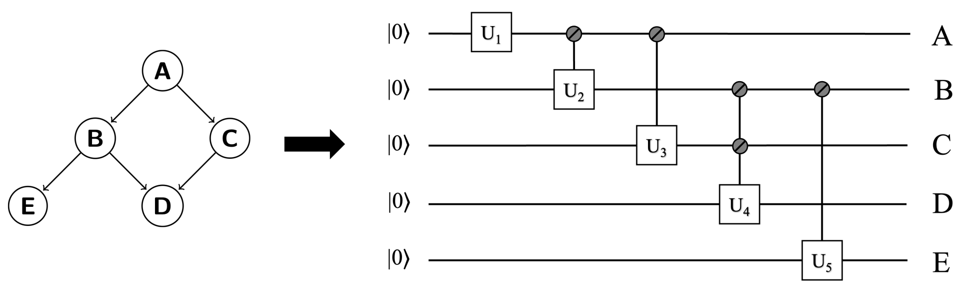


Figure 8: **Bayesian network and the corresponding Bayesian quantum circuit.** The building blocks of BQCs are uniformly controlled (parametrized) gates, for which control units and target units are labeled.

These are defined such that the probability distribution they sample from, by measuring the given qubits in the computational basis, corresponds to the distribution defined by the corresponding Bayesian network. BQCs are implemented with uniformly controlled gates, that can be decomposed into one-qubit rotations and *CNOT* gates [12, 38]. By definition, these gates obey some rules in accordance with the directed, acyclic nature of the underlying graph:

1. Uniformly controlled gates only target a single qubit, reflecting the directed edges;
2. After being used as a control, a qubit cannot be targeted anymore, which respects acyclicity.

As an example, we show in Fig. 8 the circuit corresponding to the Bayesian network depicted in Fig. 1a.

In [8] the authors introduced a minimal extension to BQCs and presented unconditional proof of separation in the expressive power of Bayesian networks and the corresponding basis-enhanced BQCs. They showed that by letting the final measurement be in any local basis, this separation appears, that can be associated with quantum nonlocality and contextuality. They also pointed out, that both BQCs and their basis-enhanced versions can be efficiently simulated with classical tensor network methods, when the graphs are not too dense.

3.3.2 Unitary embedding of MRFs

In Ref. [13] the authors identified a novel embedding technique of MRFs into unitary operators that relies on their log-linear representation. First they introduce the *Pauli-*

Markov sufficient statistics according to:

Definition 7 (Pauli-Markov Sufficient Statistics) *Let $\phi_{C,\mathbf{y}} : \mathcal{X} \to \mathbb{R}$ for $C \in \mathcal{C}$ and $\mathbf{y} \in \mathcal{X}_C$ denote the sufficient statistics of some overcomplete family of graphical models. The diagonal matrix $\Phi_{C,\mathbf{y}} \in \{0,1\}^{2^n \times 2^n}$, defined via $(\Phi_{C,\mathbf{y}})_{i,j} = \phi_{C,\mathbf{y}}(\mathbf{x}^j)$ iff $i = j$ (and 0 otherwise), denotes the Pauli-Markov sufficient statistics. Where $\mathbf{x}^j$ denotes the j-th full n-bit joint configuration w.r.t. some arbitrary but fixed order.*

The Pauli-Markov sufficient statistics can be calculated in linear time according to Algorithm 1.

Algorithm 1 Computing Pauli-Markov sufficient statistics in linear time.

```
Require: $C \subseteq V, \mathbf{y} \in \mathcal{X}_C$
Ensure: $\Phi_{C,\mathbf{y}} = \Phi$
  $\Phi \leftarrow 1$
  for $v \in V$ do
      if $v \notin C$ then
          $\Phi \leftarrow \Phi \otimes I$
      else if $v \in C$ and $\mathbf{y}_v = 1$ then
          $\Phi \leftarrow \otimes(I - Z)/2$
      else if $v \in C$ and $\mathbf{y}_v = 0$ then
          $\Phi \leftarrow \otimes(I + Z)/2$
      end if
  end for
  return $\Phi$
```

With this sufficient statistics, they constructed the following Hamiltonian:

Theorem 1 (Pauli-Markov Hamiltonian) *Assume an overcomplete binary graphical model specified by $(\boldsymbol{\theta}, \phi)$. When $H_{\boldsymbol{\theta}} = -\sum_{C \in \mathcal{C}} \sum_{\mathbf{y} \in \mathcal{X}_C} \boldsymbol{\theta}_{C,\mathbf{y}} \Phi_{C,\mathbf{y}}$, then $\mathbb{P}_{\boldsymbol{\theta}}(\mathbf{x}^j) = (\exp_M(-H_{\boldsymbol{\theta}})/Tr\ \exp_M(-H_{\boldsymbol{\theta}}))_{j,j}$, where $\exp_M$ is the matrix exponential and Tr is the trace.*

This is a commuting Hamiltonian, since it only contains many-body Pauli-Z terms.

They give a quantum algorithm, that implements the exponential of this Hamiltonian, meaning, that measuring the output qubits of the corresponding quantum circuit is equivalent to sampling the underlying MRF. The circuit, that implements this exponentiation, uses a special point-wise polynomial approximation and real part extraction, that might fail. For this reason, they have to measure ancillary qubits in order to determine whether this extraction was successful and start everything over if not. The success probability decreases exponentially with the number of maximal cliques. This can of course be amplified with quantum singular value transformation [39], but that makes it not very suitable for near-term quantum devices.

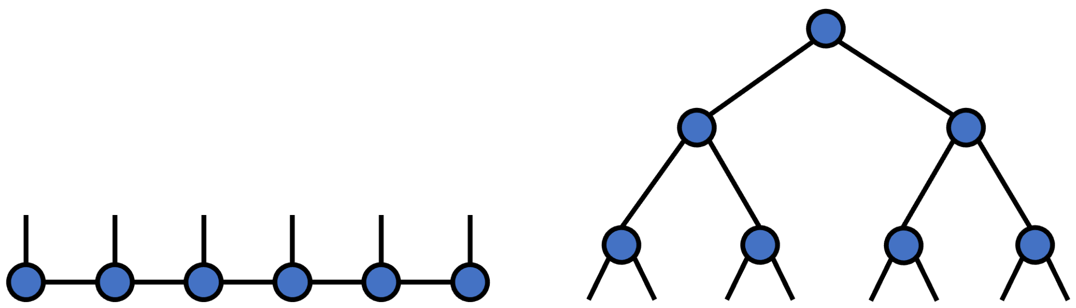
This model can be effectively used for all three tasks of Markov random fields, namely representation, inference, and learning as it gives a direct mapping between classical MRFs and unitary quantum circuits with post-selection. However, this also means, that the number of free parameters can be unnecessarily high in the context of learning.

Our approach, described in the next chapter, eliminates the weaknesses of this technique by combining it with the more powerful generic quantum circuit learning framework of QCBMs.

3.4 Quantum-inspired Techniques: Tensor Networks for QML

Initially developed for the numerical simulation of quantum many-body systems [40], tensor networks (TNs) also turned out to provide a useful paradigm for machine learning applications [41]. With the development of quantum computing, these architectures can now be implemented not only on a classical, but on quantum computers as well [42]. Recently, the interest for these methods has increased significantly, developing several new approaches. Here we focus on one approach, which is simulating quantum circuits and thus quantum machine learning models on a classical computer using tensor network methods, making these quantum-inspired classical models.

Tensor networks provide a mathematical formalism used to represent and manipulate high-dimensional arrays, or tensors. They exploit the underlying structure of tensors to efficiently compute with large amounts of data. By decomposing tensors into a network of smaller tensors and specifying contraction patterns, tensor networks enable scalable and efficient computations for tasks such as data compression, dimensionality reduction, feature extraction, and simulation of quantum systems. The rich variety of tensor network architectures, including matrix product states (MPS) and tree tensor networks (see Fig. 9) offer flexible tools for analyzing and processing complex data structures. Their appli-



(a) Matrix product state (b) Tree tensor network

Figure 9: **Examples of popular tensor network layouts.** A node with n lines denotes an n-dimensional tensor, while connected lines represent contraction of indices.

cation in classical machine learning usually allows the construction of more compact and sometimes even more scalable models than basic feed-forward neural networks.

MPS might be the most popular tensor network layout. As an example, the MPS factorization of a 6-dimensional tensor T can be written in the traditional notation as:

$$T^{s_1 s_2 s_3 s_4 s_5 s_6} = \sum_{\{\alpha\}} A^{s_1}_{\alpha_1} A^{s_2}_{\alpha_2} A^{s_3}_{\alpha_3} A^{s_4}_{\alpha_4} A^{s_5}_{\alpha_5} A^{s_6}_{\alpha_6}, \tag{22}$$

where the α values are called bond dimensions. This equation can also be formulated in the tensor network graphical notation, as shown in Fig. 10. In a tensor network diagram, tensors are denoted by vertices of arbitrary shape and edges represent summation over an index. Free indices are depicted by dangling edges, i.e. legs attached to a single vertex only. Both types of edges have some dimensionality associated to them. In an MPS diagram, the dimensionality of connected lines is called bond dimension, while the visible dimensions are represented by dangling legs.

Any tensor network can be represented exactly in MPS form if the bond dimensions are large enough. This technique can be effectively used in machine learning, since it breaks up large matrix multiplications into several multiplications of smaller matrices. By having parametrized tensors in the decomposition, these can also be trained.

There is a direct correspondence between tensor networks and quantum circuits. This correspondence makes it possible to express quantum circuits with their MPS factorization. When the bond dimensions are low enough, meaning, that there is limited entanglement in the corresponding quantum circuit, we can use MPS methods to efficiently simulate the quantum system classically [43].

In [42] and [11] the authors proposed frameworks in which they pre-train an MPS,

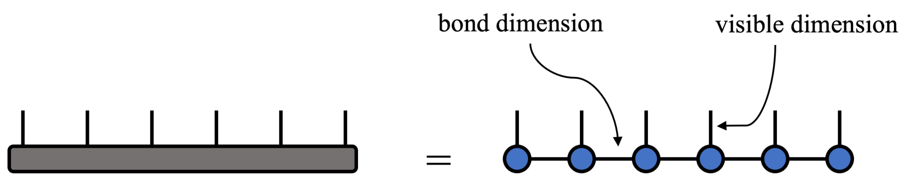


Figure 10: **Eq.** (22) **formulated using the tensor network notation.** A 6 dimensional tensor is represented by a matrix product state composed of 6 smaller tensors. The dimensionality of connected lines is called bond dimension, while the visible dimensions are represented by dangling lines.

then convert it to a parametrized quantum circuit, where the parameters are initialized according to the trained MPS. This circuit is then extended with additional parametrized gates, that introduce more entanglement to the system. This extended model is further trained on a quantum computer. These approaches effectively combine quantum-inspired and quantum-enhanced techniques to make new models that outperform both. We will refer to these techniques as hybrid methods.

4 Quantum Circuit Markov Random Fields

We propose a QCBM Ansatz for distribution learning in Markov random fields, a task described in Prob. 1. The basis of our model is a quantum circuit Ising Born machine, that implements the time evolution under the Pauli-Markov Hamiltonian similar to the one described in Thm. 1 as

$$U_Z(\boldsymbol{\alpha}) = e^{-iH(\boldsymbol{\alpha})}, \tag{23}$$

where $H(\boldsymbol{\alpha})$ denotes a simplified Pauli-Markov Hamiltonian described in the next section.

It is important to mention, that by implementing real time evolution under the simplified Hamiltonian, the direct mapping between factor values and circuit parameters is broken, but our numerical experiments suggest, that this heuristic model is expressive enough to learn the distribution of the corresponding MRFs.

We further generalize this Ansatz by initializing the circuit with a specific state preparation U_S instead of the Hadamard gates. This set of fixed gates prepares a superposition state, that can serve as an initial guess for the target probability distribution. In the most general case, when we have no prior knowledge, this simply prepares the equal superposition of all basis states $|+\rangle^{\otimes n}$ achieved by the application of Hadamard gates on each qubit as in the original QCIBM circuit.

In general, we also let the blocks composed of U_Z and U_f unitaries be repeated multiple times, but during the numerical investigations, we restrict ourselves to only one layer, as it proves to be powerful enough. The resulting general framework is depicted on Fig. 11. Since the gates in U_Z are defined by the clique factorization of the graph structure underlying a MRF, our model shows even more similarities to the Quantum Alternating Operator Ansatz [44] (for a theoretical overview of QAOA, see Appendix A). We call these models quantum circuit Markov random fields (QCMRFs).

4.1 The Hamiltonian

The Hamiltonian in Thm. 1 is composed of many-body Pauli-Z terms, however, these are repeated several times with different coefficients. Since all terms commute, we can reparamterize these coefficients such that each term only appears once. This way, we can end up with fewer parameters and a shallower circuit.

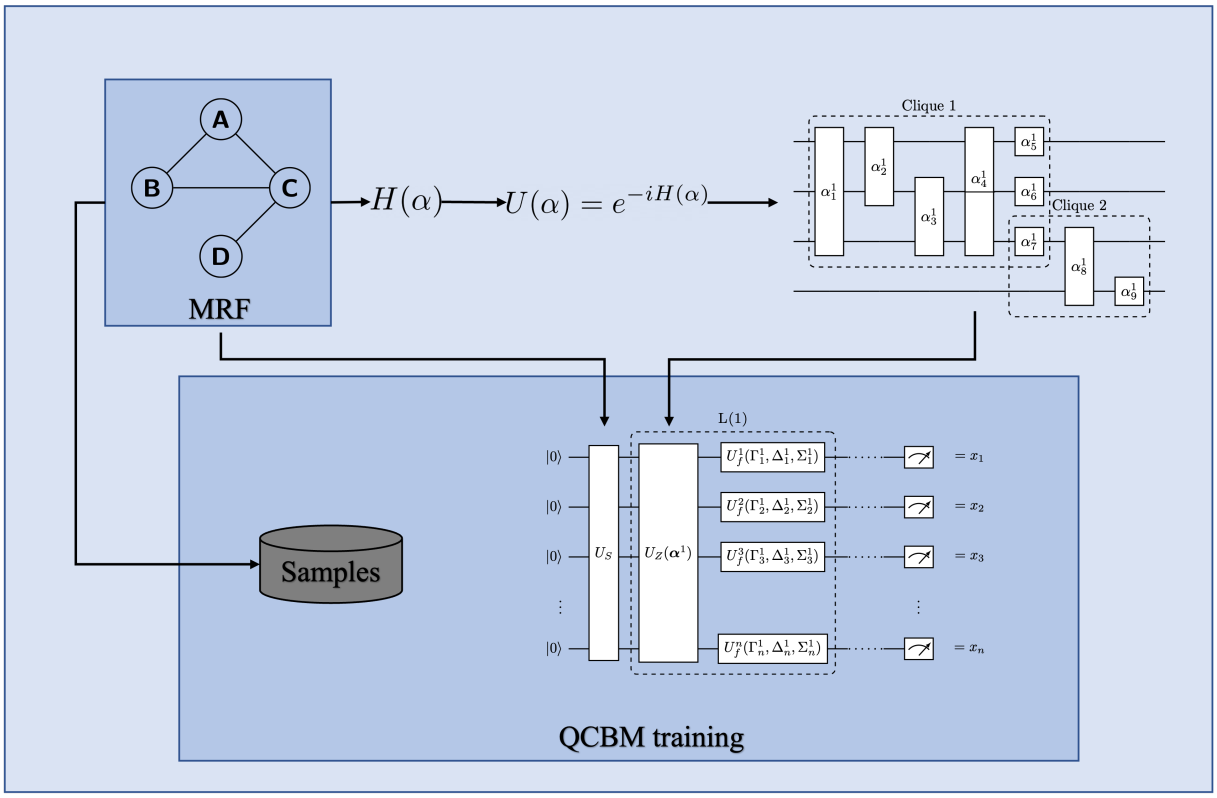

A
B
C
D
MRF
$H(\alpha)$
$U(\alpha) = e^{-iH(\alpha)}$
Clique 1
Clique 2
α_1^1
α_2^1
α_3^1
α_4^1
α_5^1
α_6^1
α_7^1
α_8^1
α_9^1
Samples
L(1)
$|0\rangle$
U_S
$U_Z(\boldsymbol{\alpha}^1)$
$U_f^1(\Gamma_1^1, \Delta_1^1, \Sigma_1^1)$
$U_f^2(\Gamma_2^1, \Delta_2^1, \Sigma_2^1)$
$U_f^3(\Gamma_3^1, \Delta_3^1, \Sigma_3^1)$
$U_f^n(\Gamma_n^1, \Delta_n^1, \Sigma_n^1)$
$= x_1$
$= x_2$
$= x_3$
$= x_n$
QCBM training

Figure 11: **The general quantum circuit Markov random field framework.** The QCBM Ansatz is constructed according to the MRF structure.

We can also think of this Hamiltonian as

$$H(\boldsymbol{\alpha}) = \boldsymbol{\alpha} \sum_{C \in \mathcal{C}} \bigotimes_{v \in C} (I + Z_v), \tag{24}$$

where Z_v is the Pauli-Z matrix acting on qubit v. In this case, however, we still have terms that are present multiple times, corresponding to subsets of nodes that appear in more than one clique, but these can be subsequently removed.

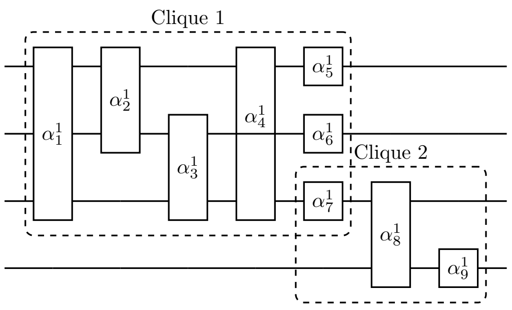
As an example, let us consider the MRF depicted in Fig. 1b with maximal clique factorization, that corresponds to the factor graph in Fig. 3a. This structure produces the following Hamiltonian:

$$H(\boldsymbol{\alpha}^1) = \alpha_1^1 Z_0 Z_1 Z_2 + \alpha_2^1 Z_0 Z_1 + \alpha_3^1 Z_1 Z_2 + \alpha_4^1 Z_1 Z_3 + \alpha_5^1 Z_2 Z_3 + \alpha_6^1 Z_0 + \alpha_7^1 Z_1 + \alpha_8^1 Z_2 + \alpha_9^1 Z_3. \tag{25}$$

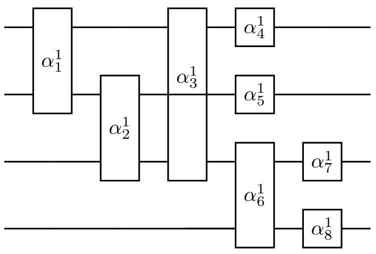
This Hamiltonian generates the circuit in Fig. 12a.

Similarly, considering a pairwise MRF with the same graph structure, as in Fig. 3b, the Hamiltonian can be written as

$$H(\boldsymbol{\alpha}^1) = \alpha_1^1 Z_0 Z_1 + \alpha_2^1 Z_1 Z_2 + \alpha_3^1 Z_1 Z_3 + \alpha_4^1 Z_2 Z_3 + \alpha_5^1 Z_0 + \alpha_6^1 Z_1 + \alpha_7^1 Z_2 + \alpha_8^1 Z_3. \tag{26}$$



(a) Maximal clique factorization



(b) Pairwise MRF

Figure 12: **The quantum circuits corresponding to the two possible factor graphs from Fig. 3.** The gates parametrized by α_j^i acting on n qubits denote $\exp\left(i\alpha_j^i \bigotimes_n Z_n\right)$.

The only difference in this case is, that this 2-local Hamiltonian does not consider the 3-body term, that comes from the first clique. The circuit that implements the time evolution under this Hamiltonian is depicted in Fig. 12b.

The most important difference from the original QCIBM is that in [30] the authors restricted the Hamiltonian to only one- and two-body terms, but our method also considers higher order interactions depending on the clique factorization of the graph. Through extensive numerical investigations, we show that this actually improves the model's performance for non-pairwise MRFs. In the case of pairwise binary MRFs, our model also implements only one- and two-body terms, but instead of all pairs, it only considers the qubit-pairs that correspond to edges in the graph. This way we get a shallower circuit with fewer parameters, that will still prove to be expressive enough, while having a higher chance of avoiding optimization bottlenecks, and providing a faster training.

4.2 Experimental Setup

To benchmark our method, we conducted numerical experiments with different graph structures. For each graph, with given number of nodes, we constructed a clique factorization (pairwise or according to maximal cliques) and assigned random factor values from a uniform distribution in an IID fashion. We calculated the joint probability distribution by multiplying the corresponding factors and normalizing the measures. The quantum models were trained with a negative log-likelihood cost function as in Eq. (18) using the exact distribution of the MRF, to benchmark their expressivity. We also conducted some experiments with the MMD loss function using a finite number of samples as a training set. To have more robust statistics, our benchmarks consider multiple factor values for each graph configuration and then average the optimization cycles. While not used directly for training, we tracked the TV throughout the optimization and benchmark the performance of different models based on this metric.

We benchmark our model against 2 kinds of previous designs, the first is the hardware-efficient Ansatz with different number of layers defined with a linear topology (as shown in Fig. 6), and the second being the original QCIBM. In these experiments, we do not have any prior knowledge about the target distributions, thus the U_S operator in the QCMRF circuits simply applies Hadamard gates on all qubits. All parametrized gates in all the circuits are initialized as identities, meaning, that all trainable parameters are set to 0. This leads to both the QCIBM and QCMRF models starting in the $|+\rangle^{\otimes n}$ state, but the hardware-efficient Ansatz starts the training with $|0\rangle^{\otimes n}$. This way, we do not get a completely fair comparison against this letter model, since the initialization of the parameters can heavily influence the performance.

We also make a simplification in the U_f operators for both the QCMRF and the QCIBM models. We consider a case, where $\mathbf{\Sigma} = 0$, since we can choose to end the U_z operator with the one-body Z terms, so these can be merged with the Z rotations in U_f. For QCMRFs, we also consider the case when $\mathbf{\Delta}$ is also 0, thus further reducing the parameters. Thus instead of general U_3 gates, that have three parameters, the circuits end with only an R_X or R_X and R_Y rotations. This way the correspondence between these models and the QAOA circuits becomes even more apparent.

The quantum circuit simulations, along with the automatic differentiation, were implemented with the Pennylane software package [35] in Python. We used gradient descent with the Adam optimizer [36] with the learning-rate set to 0.01 unless stated otherwise.

4.3 Pairwise MRFs

First, we consider pairwise MRFs. In this case, the Hamiltonian only consists of one- and two-body terms, so the circuit structure is very similar to the original QCIBM, but without considering all pairs, thus we get a more compact model with fewer parameters. The goal here is to show, that QCMRFs have the same performance as QCIBMs while having significantly fewer parameters.

The number of parameters in a QCIBM (setting $\mathbf{\Sigma} = 0$), that tries to solve a binary MRF with n variables is $k_1 = n(n-1)/2 + l \cdot n$, while this number for a QCMRF is $k_2 = m + l \cdot n$, where m is the number of edges in the graph and l is the number of parameters in the U_f operators. Since $m \leq n(n-1)/2$, it is generally true, that $k_2 \leq k_1$, where we have equality for full-graphs. This makes our method especially suitable for sparse graphs.

Note, that by using the log-linear representation, pairwise MRFs can be expressed with $m + n$ parameters, thus more compactly than either of the two quantum models. In [8], Gao et al. showed that by implementing Bayesian networks exactly on a quantum computer and letting the measurement be in any local basis, one can enhance the expressivity of the model with quantum(-inspired) correlations. For this reason, while having more parameters, than the log-linear model, we expect our model to be more expressive and have a potential to outperform classical learning algorithms for this task.

Inspired by the benchmark graphs in [45], we consider structured graphs with 9 nodes, including linear, circular, grid and 8-grid topology. The results are shown in Fig. 13, where we ran 10 iterations for each graph with different factor values and averaged over these. On the left side, we can see the average learning curves in TV for the first 200 training steps, while on the right side, the scatter plot of the performance is displayed. Here, we defined the final TV as the average TV of the last 50 training steps for each run, and the training speed denotes the number of iterations the TV first reaches this average value $\pm 10\%$. The results show that our method has the same average performance as the problem-agnostic QCIBM, while having less parameters and shallower circuits. The results of the original hardware-efficient QCBM with one and two layers are also displayed, but they rarely reach the performance of the other models.

4.4 Higher-order MRFs

In general, MRFs do not have pairwise factorization. Consequently, the Pauli-Markov Hamiltonian contains higher order interactions. In this case, the QCMRF circuit can be deeper and can have more parameters than the problem-agnostic QCIBM. We claim, that even in the cases, when our model has fewer parameters, its performance can be better in terms of both expressivity and trainability.

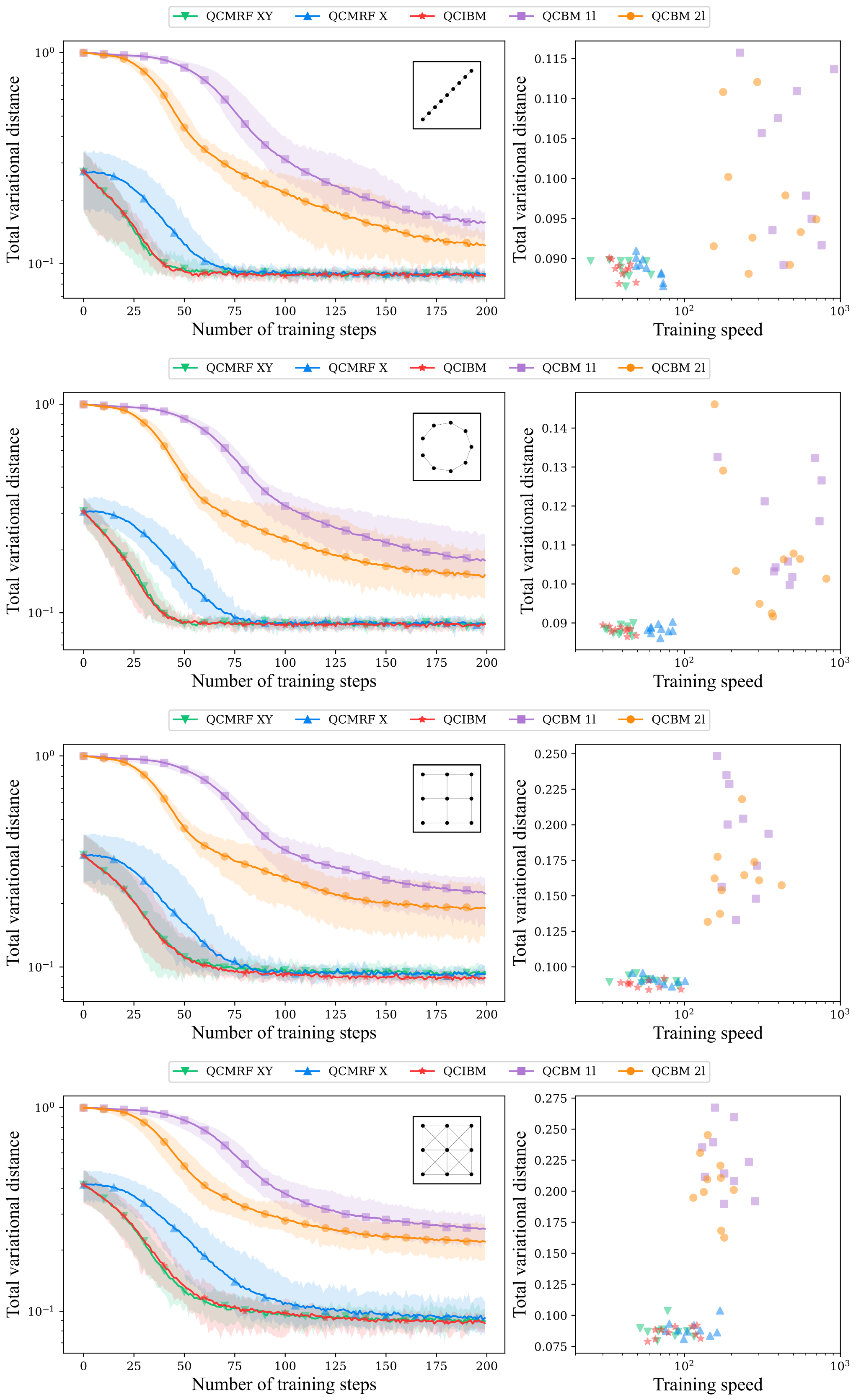


Figure 13: **Training performance of the QCMRF, QCIBM and hardware-efficient QCBM models on benchmark MRFs with given topology.** Average training curves (left), final individual performance (right).

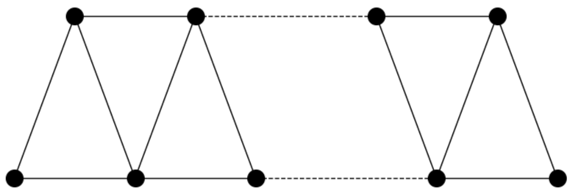


Figure 14: **Special graph topology, that has maximal cliques of size 3.** Every node is part of these maximal cliques.

The number of parameters in general is

$$\sum_{C\in\mathcal{C}}\sum_{k=1}^{l_C}\binom{l_C}{k}+3n=\sum_{C\in\mathcal{C}}(2^{l_C}-1)+3n, \tag{27}$$

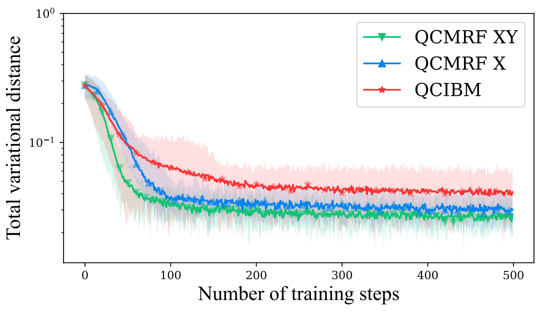
where l_C denotes the size of clique C. The worst case happens for MRFs that factorize over the maximal clique of a full graph. In this case, the number of parameters in the QCMRF circuit is exponential:

$$\sum_{k=1}^{n}\binom{n}{k}+3n=2^n+3n-1. \tag{28}$$

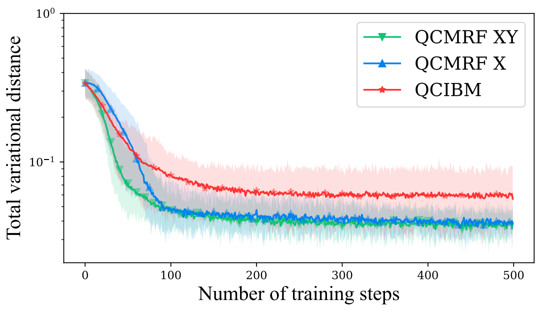
For this reason, our method is useful for sparse graphs, where the size of some maximal cliques is bigger than 2, but not too big.

Also note that for MRFs that are defined over a full graph, and do not factorize according to smaller cliques, to describe the joint probability distribution, we need 2^n parameters, one for each possible assignment. This is also exponential in the number of variables.

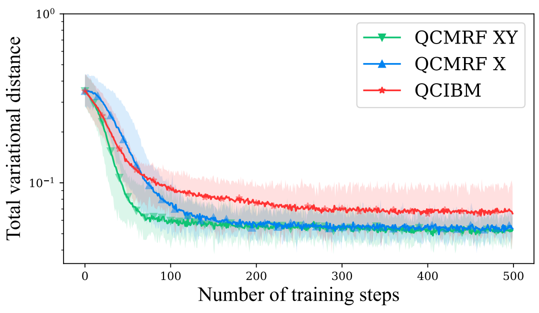
In general, it is true that the number of parameters in the QCMRF circuits does not depend directly on the number of random variables in the MRF, or nodes in the graph, only on the sizes of the cliques. Thus, we first tested the expressivity separation with graphs, that have a fixed clique size. We started the numerical investigations with a special kind of graph, that is depicted in Fig. 14. In this type of graphs, the size of the maximal cliques is exactly 3 for each clique. Consequently, the MRFs that factorize according to the maximal cliques of such a graph can be described by a Pauli-Markov Hamiltonian with 1-, 2- and 3-body terms. The number of parameters in the QCMRF circuits scales $\mathcal{O}(n)$, while for the QCIBM it is $\mathcal{O}(n^2)$, where n is the number of nodes, and we have equality for $n=5$.



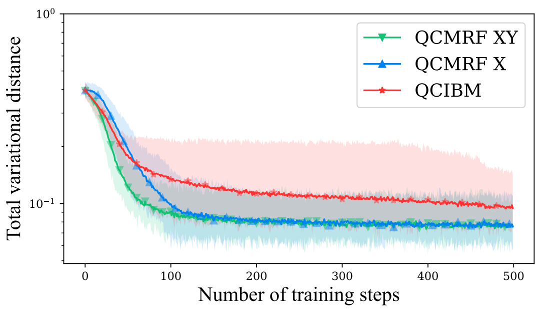
(a) 5 nodes



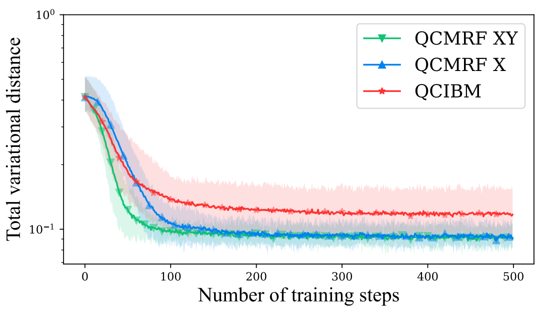
(b) 6 nodes



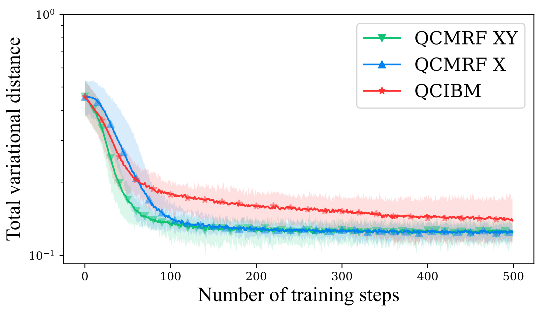
(c) 7 nodes



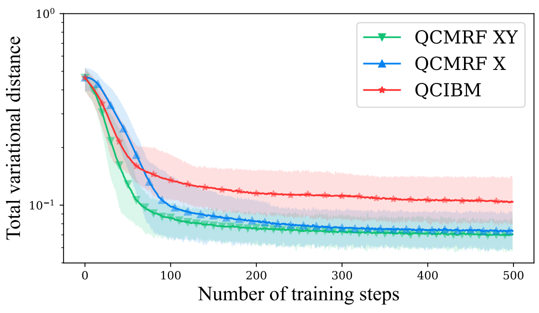
(d) 8 nodes



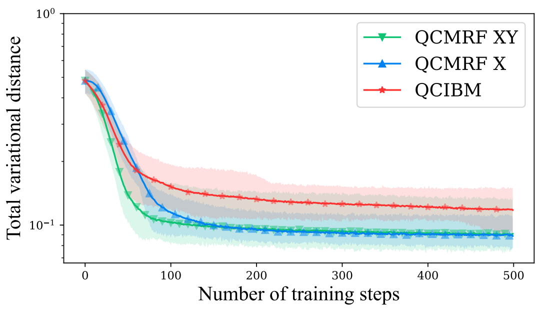
(e) 9 nodes



(f) 10 nodes



(g) 11 nodes



(h) 12 nodes

Figure 15: **Learning curves of the QCMRF and QCIBM models on MRFs defined over the graph topology in Fig. 14 with different numbers of nodes.** The MRFs factorize according to the maximal cliques of 3. The results are averaged over 10 runs with different factor values.

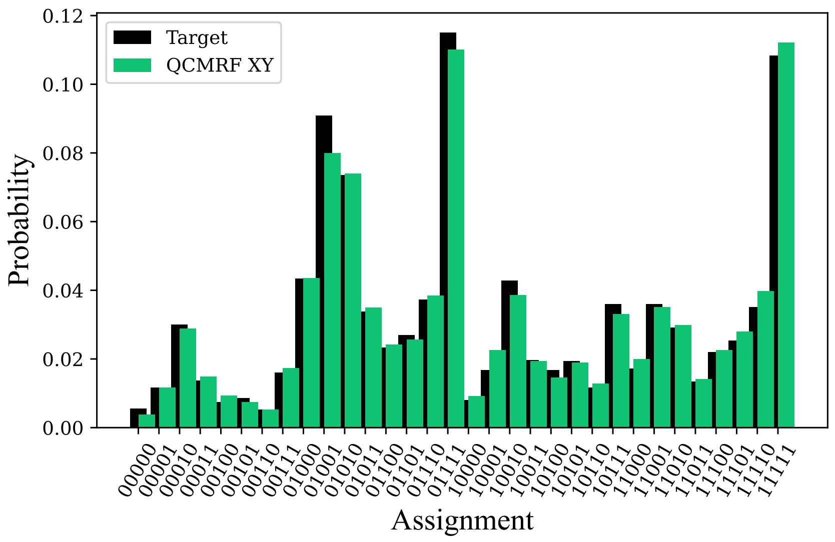


Figure 16: **Example of the final output distribution of the trained QCMRF model and the target distribution used for training.** This example is one of the benchmarks for the 5-node version of the graph in Fig. 14.

We ran similar experiments as in the previous section, testing the robustness of the expressivity separation between the QCMRF and QCIBM models with $n = 5$ to $n = 12$ nodes and with 10 different sets of random factor values in each case. The number of shots used to sample the quantum circuit for each graph was chosen from the set $\{10^3, 10^4, 10^5\}$ depending on the size of the graph. The learning curves similar to the previous results are shown in Fig. 15. We can see a clear separation between the QCMRF and QCIBM models that seems to be independent of the system size. As an example, a target probability distribution and the corresponding distribution produced by a QCMRF model is shown in Fig. 16.

Next, to showcase the performance for larger cliques, let us consider an Erdős-Rényi random graph with 10 nodes and cliques of 3 and 4 nodes; thus the corresponding Hamiltonian also has 3- and 4-local terms. The results are shown in Fig. 17, where the expressivity separation is more apparent, but it is also worth mentioning, that in this case, the QCMRF models have 26% and 33% more parameters than the QCIBM circuit.

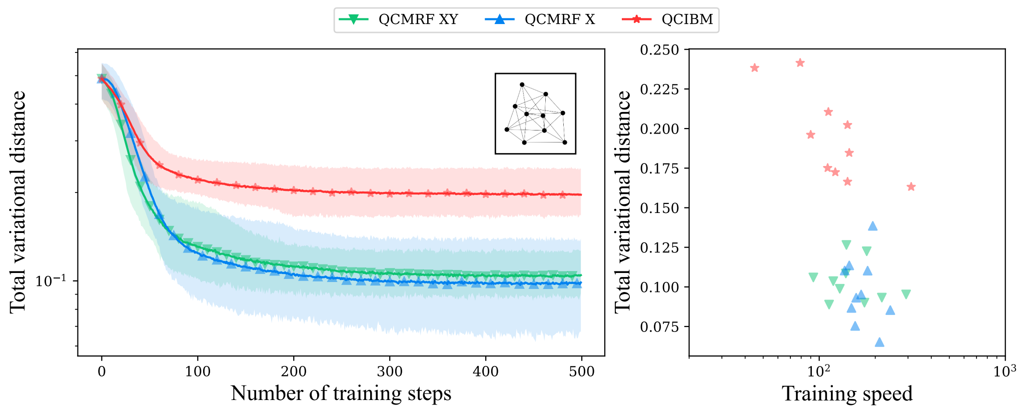
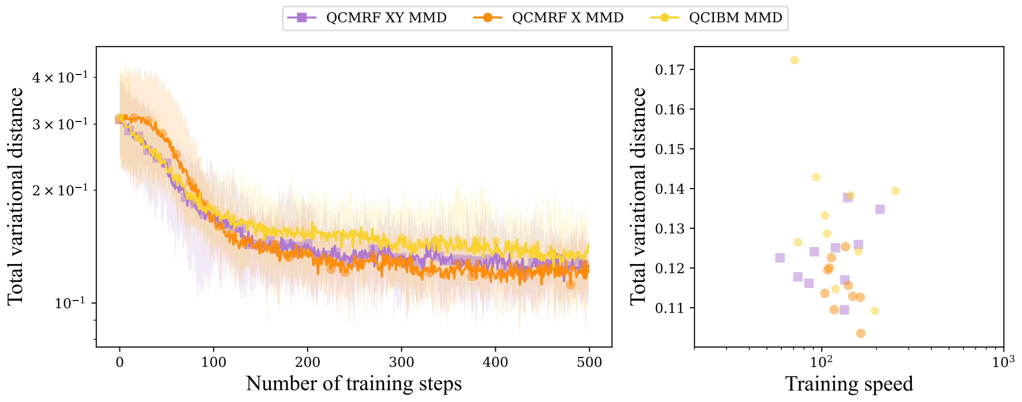


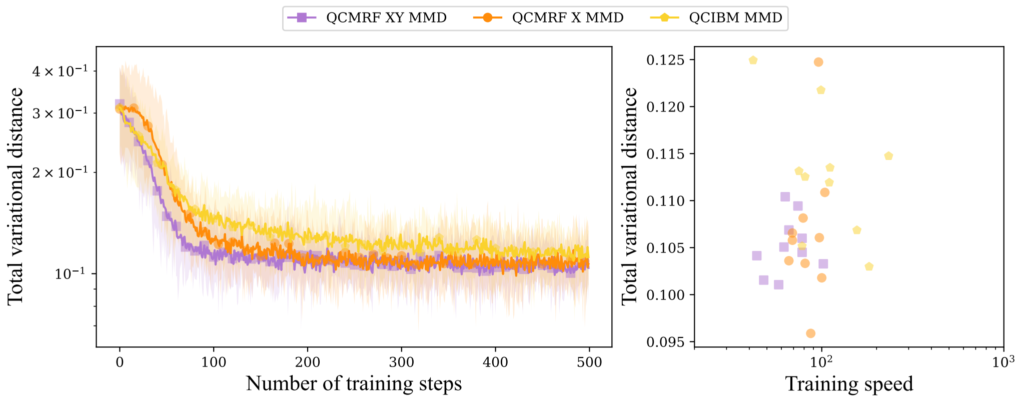
Figure 17: **Learning curves of the QCMRF and QCIBM models on MRFs defined over an Erdős-Rényi random graph.** The MRFs factorize over the maximal cliques of the graph, with 3 or 4 nodes in each clique. The results are averaged over 10 runs with different factor values.

Experiments with different loss functions

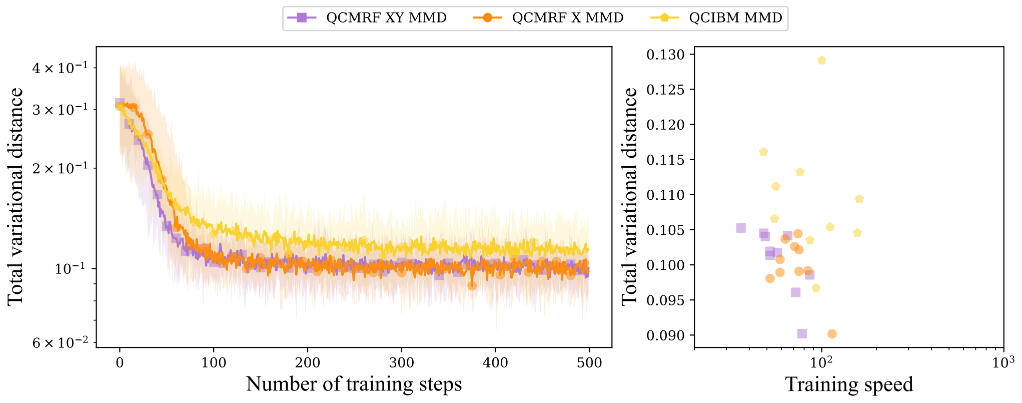
As a final addition to the numerical investigations regarding the expressivity separation between the two models, we should consider other training scenarios involving more realistic cost functions. For this, we implemented the training with the MMD loss function as expressed in Eq. (19) where the number of shots used to sample the quantum circuit is finite but fixed value and the number of training samples is adjustable. Fig. 18 shows the evolution of the total variational distance and the trained performance for the 3 models with different number of training samples for the MMD loss function. These results showcase how increasing this number increases the difference in the expressivity between the QCMRF and QCIBM models. In Fig. 19 we also investigate how the performance with the MMD loss function and finite number of training samples approaches the one achieved with the negative log-likelihood cost using the exact MRF distribution for training. The MMD loss functions reach the NLL performance already with 100 training samples.



(a) 10 training samples

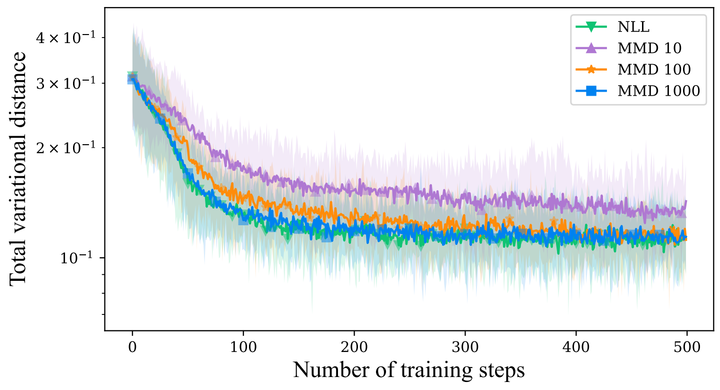


(b) 100 training samples

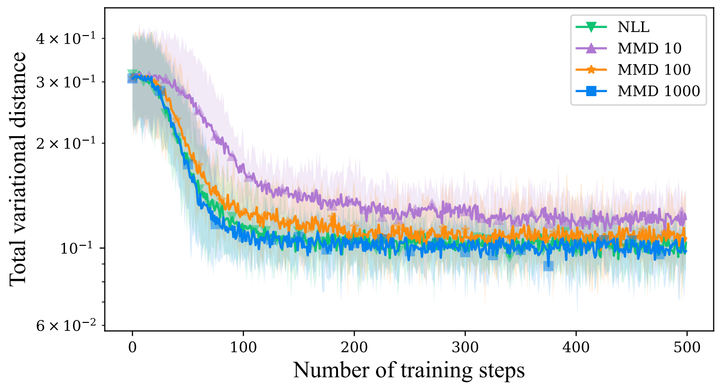


(c) 1000 training samples

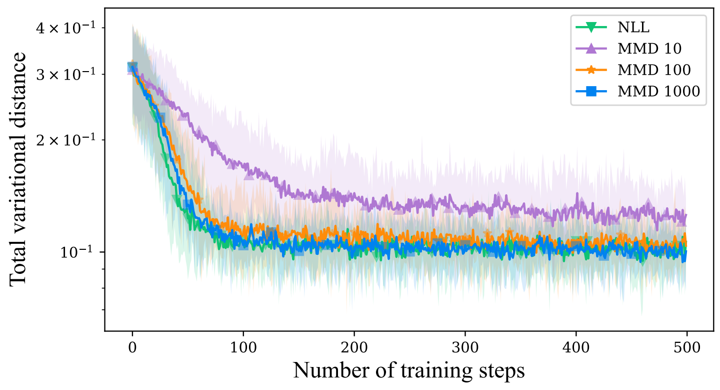
Figure 18: **The performance of the QCMRF and QCIBM models with the MMD loss function** We used different numbers of training samples for the MRF defined over the maximal clique factorization of the graph in Fig. 14 with 6 nodes. These results show how the increase in the number of samples increases the expressivity separation between models.



(a) QCIBM



(b) QCMRF X



(c) QCMRF XY

Figure 19: **The learning curves of the QCMRF and QCIBM models with the MMD and the NLL loss functions.** We used the MRF defined over the maximal clique factorization of the graph in Fig. 14 with 6 nodes. The MMD loss function reaches the NLL performance already with 100 training samples.

5 Generalization and Improvements

The Pauli-Markov sufficient statistics can be easily generalized to non-binary MRFs. Following the method used in [46] and [47] in the context of QAOA, we can use a binary representation and assign $\lceil \log(n_v) \rceil$ qubits to each vertex v, where n_v is the number of possible states in the MRF for the corresponding variable. Please note, that the number of possible states for a vertex n_v need not be equal for different vertices.

Having this representation, one can construct the Pauli-Markov sufficient statistics for general non-binary MRFs, as shown in Algorithm 2. In the algorithm, $\mathbf{y}_{v,i}$ denotes the i-th binary value of vertex v in assignment y.

Algorithm 2 Computing Pauli-Markov sufficient statistics for non-binary MRFs.

```
Require: $C \subseteq V, \mathbf{y} \in \mathcal{X}_C$
Ensure: $\Phi_{C,\mathbf{y}} = \Phi$
  $\Phi \leftarrow 1$
  for $v \in V$ do
      if $v \notin C$ then
          for $i \leq \lceil \log(n_v) \rceil$ do
              $\Phi \leftarrow \Phi \otimes I$
          end for
      else if $v \in C$ then
          for $i \leq \lceil \log(n_v) \rceil$ do
              if $\mathbf{y}_{v,i} = 1$ then
                  $\Phi \leftarrow \otimes (I - Z)/2$
              else if $\mathbf{y}_{v,i} = 0$ then
                  $\Phi \leftarrow \otimes (I + Z)/2$
              end if
          end for
      end if
  end for
  return $\Phi$
```

Thus, the Pauli-Markov Hamiltonian can be constructed using this Pauli-Markov sufficient statistics as in Thm. 1 and our QCMRF can be implemented according to the

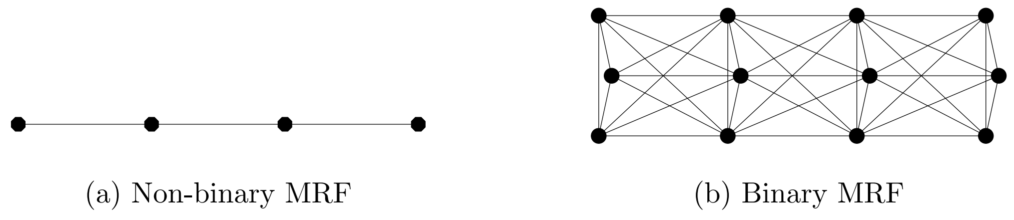


Figure 20: **Two possible MRFs for the language modeling problem.** The MRF with 4 variables and 8 possible states for each variable can be transformed to a binary MRF having 12 nodes and 3 cliques of 6 binary variables each.

simplified version of this Hamiltonian.

Ultimately, this is equivalent to mapping the problem defined over a non-binary MRF to one over binary variables. We demonstrate this process through a toy problem in the next section.

Simply using this Hamiltonian can result in measurements, that do not correspond to a possible assignment when the number of possible states n_v for any vertex v is not a power of 2, i.e. $\lceil \log(n_v) \rceil \neq \log(n_v)$. To work around this problem, instead of the initial Hadamard gates, we can use some clever initialization and instead of the final R_X and R_Y rotations, use something like the XY- or Grover-mixers introduced for QAOA [44, 48], to reduce the size of the effective space the quantum algorithm searches over. This can be implemented similarly to the 1-layer Func-QAOA circuit for the Max-K-Cut problem as described in [49]. In the supplementary materials of this previous work, we explained this approach and gave a circuit, that prepares the superposition state of the first K numbers in binary representation.

5.1 Language Modeling Experiment

In real-life use-cases, we can try to model certain tasks with Markov random fields, however, the distributions usually have a relatively small number of nonzero values, that might not factorize according to some MRF. In this section, we show how to use our non-binary QCMRF framework to learn the distribution of a toy problem inspired by generative language modeling.

A subtask of large language models is to estimate a distribution over words that best represents the text of a corpus. MRFs are rarely used for language modeling since computing the partition function is costly. Models based on QCBMs could potentially outperform these classical algorithms, since they inherit the Born rule and the calculation of the partition function can be avoided.

In [21] and [50] the authors have shown how Markov random fields can be used in language modeling, or rather how some language models can be thought of as MRFs. Based on this works, we define a simplified problem: given a set of sentences, train a model, that can generate similar sentences of the same length.

The steps of this process are as follows:

1. Choose (sample) the sentence length according to the training data;
2. Construct a non-binary MRF with this many nodes;
3. Knowing the number of distinct words in the corpus, construct another MRF with only binary variables (later qubits) having the clique factorization defined by the previous MRF;
4. Construct the corresponding quantum circuit Ansatz;
5. Choose cost function and optimizer;
6. Train the model;
7. Use the trained model for sampling.

We use sentences composed of 4 words and there are 8 distinct words in the corpus: “the”, “dog”, “brown”, “eats”, “yellow”, “drinks”, “cat”, “monkey”. The sentences in the training set are the following:

[

“The brown dog eats.”; “The brown dog drinks.”;

“The brown cat eats.”; “The brown cat drinks.”;

“The brown monkey eats.”; “The brown monkey drinks.”;

“The yellow dog eats.”; “The yellow dog drinks.”;

“The yellow cat eats.”; “The yellow cat drinks.”;

“The yellow monkey eats.”; “The yellow monkey drinks.”

].

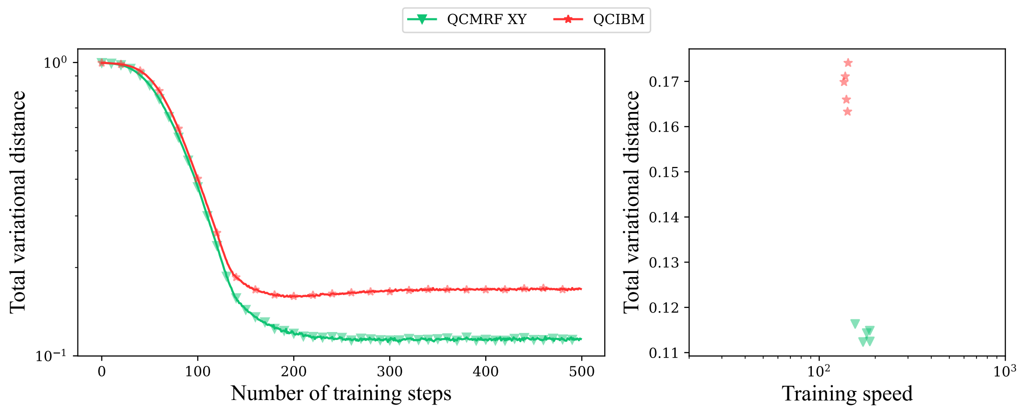


Figure 21: **Results of the language modeling experiment.** 5 training cycles were run, and their average is displayed. However, the difference between these is below 10^{-2} at each step.

We tried modeling this problem with an MRF having 4 variables, each with 8 possible values and linear nearest neighbors topology (Fig. 20a). Since $\lceil \log(8) \rceil = 3$, we assign 3 binary nodes to each higher order variable. The MRF graph transforms accordingly to a graph with 12 nodes and cliques of size 6 (Fig. 20b). This transformation is depicted in Fig. 20. The Pauli-Markov sufficient statistics of this binary MRF is equivalent to the one we get from running Algorithm 2 on the initial non-binary one, after simplifications. Thus, we can take this binary MRF and implement the corresponding QCMRF circuit according to the maximal cliques. The subsequent training is similar to those in the precious chapter.

To assess the performance of our method, we constructed the target distribution by sampling these sentences uniformly in an IID fashion. The words were tokenized by assigning a 3-bit number to each distinct word in binary form. This way, each possible sentence can be expressed by a 12 bit long bitstring. We trained the models similarly with the NLL loss function, and we also trained a general QCIBM for comparison. We ran 5 cycles with 10^5 shots to test the stability. The results are shown in Fig. 21.

After training, the models can be used to sample sentences. The average probabilities of getting the assignments that correspond to the sentences in the training set are shown in Tab. 2. By adding up these probabilities, we actually get that the QCMRF model samples sentences that are present in the training set 98.45% of the time, while for the QCIBM this is 99.98%. This means, that the QCIBM circuit learns better not to sample unwanted sentences, in this learning scenario, but since the total variational

Assignment	Target probability	QCMRF probability	QCIBM probability
000001011110	$8.15 \cdot 10^{-2}$	$10.75 \cdot 10^{-2}$	$12.59 \cdot 10^{-2}$
000001011111	$8.57 \cdot 10^{-2}$	$10.61 \cdot 10^{-2}$	$12.40 \cdot 10^{-2}$
000001100110	$8.39 \cdot 10^{-2}$	$7.12 \cdot 10^{-2}$	$6.30 \cdot 10^{-2}$
000001100111	$8.58 \cdot 10^{-2}$	$7.22 \cdot 10^{-2}$	$6.28 \cdot 10^{-2}$
000001101110	$8.29 \cdot 10^{-2}$	$6.72 \cdot 10^{-2}$	$6.38 \cdot 10^{-2}$
000010101111	$8.28 \cdot 10^{-2}$	$6.65 \cdot 10^{-2}$	$6.12 \cdot 10^{-2}$
000010011110	$8.35 \cdot 10^{-2}$	$10.80 \cdot 10^{-2}$	$12.49 \cdot 10^{-2}$
000010011111	$8.36 \cdot 10^{-2}$	$10.71 \cdot 10^{-2}$	$12.35 \cdot 10^{-2}$
000010100110	$8.21 \cdot 10^{-2}$	$7.21 \cdot 10^{-2}$	$6.26 \cdot 10^{-2}$
000010100111	$8.21 \cdot 10^{-2}$	$7.23 \cdot 10^{-2}$	$6.31 \cdot 10^{-2}$
000010101110	$8.38 \cdot 10^{-2}$	$6.73 \cdot 10^{-2}$	$6.27 \cdot 10^{-2}$
000010101111	$8.23 \cdot 10^{-2}$	$6.61 \cdot 10^{-2}$	$6.21 \cdot 10^{-2}$

Table 2: The assignments present in the training set, their target probabilities and the probabilities obtained by sampling the trained QCMRF and QCIBM models.

distance is higher, the samples are more biased towards certain assignments. We can see this by looking at the table as well: the probability of measuring the 1-st, 2-nd, 7-th and 8-th assignment is double than the 8 others, which can be an unwanted behavior.

5.2 Hybrid Training Schemes

We propose two hybrid training schemes similar to the one described in [11]. Having the simplified Pauli-Markov Hamiltonian, one can choose certain terms in such a way, that it can be efficiently simulated as an MPS tensor network on a classical computer. Then in the second stage, one can add, the remaining terms and continue the training on a quantum computer. For example, one can initially only consider a path in the graph of a pairwise MRF, which then can be simulated in a straightforward way with an MPS with bond dimension 2. In the case of higher order MRFs, we can consider its pairwise interactions first if they still admit efficient simulation, and then extend the set with higher order terms when continuing on the quantum computer. In this framework, the tensor network based pre-training can be seen as a good initialization of the gate parameters and thus can speed up the training. This method differs from the one described in [11] in the sense that since we have problem specific Ansatz, the way we choose the edges and cliques used for the MPS pre-training is not trivial and can vary from graph to graph, opening up an interesting path for future research.

The second method takes a slightly different approach. In this case, we start by training an arbitrary tensor network classically, then in the quantum training stage we construct the QCMRF circuit to start with some oracular initialization, that loads the classically obtained state. This way, the circuit does not start from the equal superposition over all global states, as we did in previous experiments, but from a distribution that is closer to the solution and thus can speed up the training process.

The numerical performance of both these methods still need to be investigated, but we leave this for future work.

6 Conclusion and Outlook

General purpose machine learning models have poor average performance, and this problem directly translates to QML models as well. In the context of quantum generative learning, problem-specific Ansätze, that exploit the structure of the underlying problem, are not common. By designing such models, one can enhance expressivity and trainability, while also opening the possibility of creating explainable models, that can be studied both theoretically and empirically.

In this work, we introduced a problem-specific Ansatz construction of quantum circuit Born machines for distribution learning in Markov random fields. This specific task has many applications in complex systems, where we can establish some prior knowledge about the independence relations of random variables. We call these models quantum circuit Markov random fields (QCMRFs).

Our numerical investigations showed, that our model has better expressivity than previous problem-agnostic QCBMs. We focused our analysis on the differences between our QCMRF and the 2-local QCIBM Ansatz from [30]. For pairwise MRFs, this novel construction leads to a more compact model, that proved to achieve the same performance as the QCIBM, while potentially having significantly fewer parameters. For higher-order MRFs, we have shown, that even in the cases, when our method leads to circuits with fewer parameters, it outperforms the previous model. These data-driven circuit trainings were designed with the NLL loss function to assess the expressivity of different models. However, our experiments also suggest, that the expressivity separation is consistent with other, more realistic cost functions as well.

Besides expressivity, QGLMs have other important aspects as well, like trainability and generalization performance. These also need to be studied, but we leave them for future work.

We extended our results by describing a method that allows to transform non-binary MRFs to binary ones, while also extending the Pauli-Markov sufficient statistics from [13] to non-binary MRFs, and we argued that these two methods produce the same Hamiltonian and thus the same QCMRF models. We also explicitly showed how one can model an arbitrary discrete distribution learning problem with higher order MRFs and use this method to train a QCMRF that solves it.

Just like the original QCIBM model, our QCMRF also corresponds to a QAOA circuit, meaning, that this approach could also lead to demonstrating quantum advantage [19] under reasonable complexity assumptions. The NP-hard problems solved by QAOA are often defined over some graph, thus our problem formulation of distribution learning in MRFs with fixed structure makes the similarities to a QAOA circuit even more apparent. This way, we can also see the proposed framework as the application of QAOA circuits to generative modeling. With the generalization of the state preparation at the beginning of the circuit, we also make connections to the generalized optimization models, like the XY-mixer and Grover-mixer QAOA. These extensions still need to be investigated along with increasing the number of layers, that our construction allows, but we did not consider during the numerical experiments.

As a final addition, we also proposed two hybrid approaches, that first train MPS tensor networks on a classical computer and use the results for initializing the extended model before training it on a quantum device. These approaches could potentially speed up the training or even achieve better performance. This also appears to be an interesting future direction for this work.

Abbreviations

PGM	probabilistic graphical model
MN	Markov network
MRF	Markov random field
NISQ	noisy intermediate-scale quantum
QML	quantum machine learning
QGLM	quantum generative learning models
QCBM	quantum circuit Born machine
QCIBM	quantum circuit Ising Born machine
QAOA	quantum approximate optimization algorithm
VQC	variational quantum circuit
QNN	quantum neural network
BQC	Bayesian quantum circuit
KL	Kullback-Leibler
MMD	maximum mean discrepancy
NLL	negative log-likelihood
IID	independent and identically distributed
TN	tensor network
MPS	matrix product state
TTN	tree tensor network
QCMRF	quantum circuit Markov random field

References

[1] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[2] D. Riste, M. P. Da Silva, C. A. Ryan, A. W. Cross, A. D. Córcoles, J. A. Smolin, J. M. Gambetta, J. M. Chow, and B. R. Johnson, "Demonstration of quantum advantage in machine learning," *npj Quantum Information*, vol. 3, no. 1, p. 16, 2017.

[3] H.-Y. Huang, M. Broughton, J. Cotler, S. Chen, J. Li, M. Mohseni, H. Neven, R. Babbush, R. Kueng, J. Preskill, *et al.*, "Quantum advantage in learning from experiments," *Science*, vol. 376, no. 6598, pp. 1182–1186, 2022.

[4] E. R. Anschuetz and B. T. Kiani, "Quantum variational algorithms are swamped with traps," *Nature Communications*, vol. 13, no. 1, p. 7760, 2022.

[5] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven, "Barren plateaus in quantum neural network training landscapes," *Nature communications*, vol. 9, no. 1, p. 4812, 2018.

[6] E. R. Anschuetz, "Critical points in quantum generative models," *arXiv preprint arXiv:2109.06957*, 2021.

[7] M. Schuld and N. Killoran, "Is quantum advantage the right goal for quantum machine learning?," *Prx Quantum*, vol. 3, no. 3, p. 030101, 2022.

[8] X. Gao, E. R. Anschuetz, S.-T. Wang, J. I. Cirac, and M. D. Lukin, "Enhancing generative models via quantum correlations," *Physical Review X*, vol. 12, no. 2, p. 021037, 2022.

[9] E. Stoudenmire and D. J. Schwab, "Supervised learning with tensor networks," *Advances in neural information processing systems*, vol. 29, 2016.

[10] J. Liu, S. Li, J. Zhang, and P. Zhang, "Tensor networks for unsupervised machine learning," *Physical Review E*, vol. 107, no. 1, p. L012103, 2023.

[11] M. S. Rudolph, J. Miller, J. Chen, A. Acharya, and A. Perdomo-Ortiz, “Synergy between quantum circuits and tensor networks: Short-cutting the race to practical quantum advantage,” *arXiv preprint arXiv:2208.13673*, 2022.

[12] S. E. Borujeni, S. Nannapaneni, N. H. Nguyen, E. C. Behrman, and J. E. Steck, “Quantum circuit representation of Bayesian networks,” *Expert Systems with Applications*, vol. 176, p. 114768, 2021.

[13] N. Piatkowski and C. Zoufal, “On quantum circuits for discrete graphical models,” *arXiv preprint arXiv:2206.00398*, 2022.

[14] J. Miller, G. Roeder, and T.-D. Bradley, “Probabilistic graphical models and tensor networks: A hybrid framework,” *arXiv preprint arXiv:2106.15666*, 2021.

[15] E. Robeva and A. Seigal, “Duality of graphical models and tensor networks,” *Information and Inference: A Journal of the IMA*, vol. 8, no. 2, pp. 273–288, 2019.

[16] M. Hibat-Allah, M. Mauri, J. Carrasquilla, and A. Perdomo-Ortiz, “A framework for demonstrating practical quantum advantage: Racing quantum against classical generative models,” *arXiv preprint arXiv:2303.15626*, 2023.

[17] M. Benedetti, D. Garcia-Pintos, O. Perdomo, V. Leyton-Ortega, Y. Nam, and A. Perdomo-Ortiz, “A generative modeling approach for benchmarking and training shallow quantum circuits,” *npj Quantum Information*, vol. 5, no. 1, p. 45, 2019.

[18] Y.-C. Ho and D. L. Pepyne, “Simple explanation of the no-free-lunch theorem and its implications,” *Journal of optimization theory and applications*, vol. 115, pp. 549–570, 2002.

[19] E. Farhi and A. W. Harrow, “Quantum supremacy through the quantum approximate optimization algorithm,” *arXiv preprint arXiv:1602.07674*, 2016.

[20] K. Held, E. R. Kops, B. J. Krause, W. M. Wells, R. Kikinis, and H.-W. Muller-Gartner, “Markov random field segmentation of brain MR images,” *IEEE transactions on medical imaging*, vol. 16, no. 6, pp. 878–886, 1997.

[21] Y. Jernite, A. Rush, and D. Sontag, “A fast variational approach for learning markov random field language models,” in *International Conference on Machine Learning*, pp. 2209–2217, PMLR, 2015.

[22] J. Jia, B. Wang, L. Zhang, and N. Z. Gong, "Attriinfer: Inferring user attributes in online social networks using markov random fields," in *Proceedings of the 26th International Conference on World Wide Web*, pp. 1561–1569, 2017.

[23] D. Koller and N. Friedman, *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[24] J. Tian, X. Sun, Y. Du, S. Zhao, Q. Liu, K. Zhang, W. Yi, W. Huang, C. Wang, X. Wu, *et al.*, "Recent advances for quantum neural networks in generative learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[25] S. Lloyd and C. Weedbrook, "Quantum generative adversarial learning," *Physical review letters*, vol. 121, no. 4, p. 040502, 2018.

[26] M. H. Amin, E. Andriyash, J. Rolfe, B. Kulchytskyy, and R. Melko, "Quantum Boltzmann machine," *Physical Review X*, vol. 8, no. 2, p. 021050, 2018.

[27] A. Khoshaman, W. Vinci, B. Denis, E. Andriyash, H. Sadeghi, and M. H. Amin, "Quantum variational autoencoder," *Quantum Science and Technology*, vol. 4, no. 1, p. 014001, 2018.

[28] G. H. Low, T. J. Yoder, and I. L. Chuang, "Quantum inference on bayesian networks," *Physical Review A*, vol. 89, no. 6, p. 062315, 2014.

[29] M. A. Nielsen and I. Chuang, "Quantum computation and quantum information," 2002.

[30] B. Coyle, D. Mills, V. Danos, and E. Kashefi, "The Born supremacy: quantum advantage and training of an Ising Born machine," *npj Quantum Information*, vol. 6, no. 1, p. 60, 2020.

[31] J.-G. Liu and L. Wang, "Differentiable learning of quantum circuit Born machines," *Physical Review A*, vol. 98, no. 6, p. 062324, 2018.

[32] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii, "Quantum circuit learning," *Physical Review A*, vol. 98, no. 3, p. 032309, 2018.

[33] M. Schuld, V. Bergholm, C. Gogolin, J. Izaac, and N. Killoran, "Evaluating analytic gradients on quantum hardware," *Physical Review A*, vol. 99, no. 3, p. 032331, 2019.

[34] J. Stokes, J. Izaac, N. Killoran, and G. Carleo, "Quantum natural gradient," *Quantum*, vol. 4, p. 269, 2020.

[35] V. Bergholm, J. Izaac, M. Schuld, C. Gogolin, S. Ahmed, V. Ajith, M. S. Alam, G. Alonso-Linaje, B. AkashNarayanan, A. Asadi, *et al.*, "Pennylane: Automatic differentiation of hybrid quantum-classical computations," *arXiv preprint arXiv:1811.04968*, 2018.

[36] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[37] R. Fletcher, *Practical methods of optimization.* John Wiley & Sons, 2013.

[38] V. Bergholm, J. J. Vartiainen, M. Möttönen, and M. M. Salomaa, "Quantum circuits with uniformly controlled one-qubit gates," *Phys. Rev. A*, vol. 71, p. 052330, May 2005.

[39] A. Gilyén, Y. Su, G. H. Low, and N. Wiebe, "Quantum singular value transformation and beyond: exponential improvements for quantum matrix arithmetics," in *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pp. 193–204, 2019.

[40] S. R. White, "Density matrix formulation for quantum renormalization groups," *Physical review letters*, vol. 69, no. 19, p. 2863, 1992.

[41] Y. Levine, O. Sharir, N. Cohen, and A. Shashua, "Bridging many-body quantum physics and deep learning via tensor networks," *Mathematical Aspects of Deep Learning*, p. 439, 2022.

[42] W. Huggins, P. Patil, B. Mitchell, K. B. Whaley, and E. M. Stoudenmire, "Towards quantum machine learning with tensor networks," *Quantum Science and technology*, vol. 4, no. 2, p. 024001, 2019.

[43] G. Vidal, "Efficient classical simulation of slightly entangled quantum computations," *Physical review letters*, vol. 91, no. 14, p. 147902, 2003.

[44] S. Hadfield, Z. Wang, B. O'gorman, E. G. Rieffel, D. Venturelli, and R. Biswas, "From the quantum approximate optimization algorithm to a quantum alternating operator ansatz," *Algorithms*, vol. 12, no. 2, p. 34, 2019.

[45] Y. Zheng, S. Jia, Z. Yu, T. Huang, J. K. Liu, and Y. Tian, "Probabilistic inference of binary Markov random fields in spiking neural networks through mean-field approximation," *Neural networks*, vol. 126, pp. 42–51, 2020.

[46] A. Glos, A. Krawiec, and Z. Zimborás, "Space-efficient binary optimization for variational quantum computing," *npj Quantum Information*, vol. 8, no. 1, p. 39, 2022.

[47] Z. Tabi, K. H. El-Safty, Z. Kallus, P. Hága, T. Kozsik, A. Glos, and Z. Zimborás, "Quantum optimization for the graph coloring problem with space-efficient embedding," in *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pp. 56–62, IEEE, 2020.

[48] A. Bärtschi and S. Eidenbenz, "Grover mixers for QAOA: Shifting complexity from mixer design to state preparation," in *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pp. 72–82, IEEE, 2020.

[49] B. Bakó, A. Glos, Ö. Salehi, and Z. Zimborás, "Near-optimal circuit design for variational quantum optimization," *arXiv preprint arXiv:2209.03386*, 2022.

[50] A. Wang and K. Cho, "BERT has a mouth, and it must speak: BERT as a Markov random field language model," *arXiv preprint arXiv:1902.04094*, 2019.

[51] E. Farhi, J. Goldstone, and S. Gutmann, "A quantum approximate optimization algorithm," *arXiv preprint arXiv:1411.4028*, 2014.

A Quantum Alternating Operator Ansatz

The Quantum Approximate Optimization Algorithm (QAOA) was proposed in [51] for approximately solving combinatorial optimization problems on a quantum computer. It is a variational quantum algorithm inspired by the adiabatic theorem. Here we describe its basic theoretical framework.

Let us consider a simple quantum system described by the Hamiltonian H_M and another one, that is more complicated with Hamiltonian H_P. We are interested in the ground state of H_P. Knowing these two operators, we can construct the following time-dependent Hamiltonian:

$$H(t) = (1 - s(t))\, H_M + s(t) H_P, \tag{29}$$

where the function $s(t = 0) = 0$ and $s(t = T) = 1$. The adiabatic theorem states, that if the change in $s(t)$ is slow enough, and we start in the ground state of H_M, then we stay in the ground state of $H(t)$ throughout the process, ending up in the lowest energy state of H_P (at $t = T$).

Let $H(t)$ describe the time evolution of the system. This way the time-dependent Schrödinger equation reads:

$$i\frac{\partial}{\partial t}|\psi(t)\rangle = H(t)\ |\psi(t)\rangle. \tag{30}$$

The solution is of this differential equation is

$$|\psi(t)\rangle = U(t, 0)|\psi(0)\rangle, \tag{31}$$

where $U(b, a)$ describes the propagator between times a and b:

$$U(b, a) = \mathcal{T} \exp\left\{ -i \int\limits_a^b H(t)dt \right\}. \tag{32}$$

This integral can be approximated as:

$$U(T, 0) = U(T, T - \Delta t)U(T - \Delta t, T - 2\Delta t)...U(\Delta t, 0) = \prod_{k=1}^{N} e^{-i\Delta t H(k\Delta t)}. \tag{33}$$

With the Trotter-Suzuki theorem, this can be written as

$$U(T, 0) \approx \prod_{k=0}^{N} e^{-i\Delta t(1-s(k\Delta t))H_M} e^{-i\Delta t s(k\Delta t)H_P}. \tag{34}$$

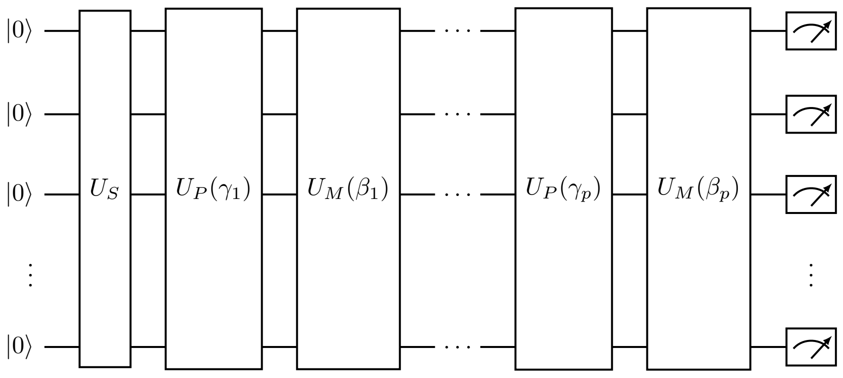


Figure 22: **General QAOA circuit with p layers.**

In the context of QAOA, let us reparametrize this trotterized evolution as:

$$\begin{aligned}\beta_k =& \Delta t\left(1 - s(k\Delta t)\right),\\ \gamma_k =& \Delta t \cdot s(k\Delta t).\end{aligned} \tag{35}$$

This way, we end up with the following unitary operator, that describes the QAOA circuit:

$$U(p, \boldsymbol{\beta}, \boldsymbol{\gamma}) = \prod_{k=0}^{p} U_M(\beta_k)U_P(\gamma_k) = \prod_{k=0}^{p} e^{-i\beta_k H_M} e^{-i\gamma_k H_P}, \tag{36}$$

where p is the number of layers. This quantum circuit can then be trained with some classical optimizer to minimize the cost function, that is by definition the expected value of H_P.

In general, H_P is called the problem Hamiltonian, since its ground state encodes the solution to a combinatorial optimization problem. H_M is the mixer Hamiltonian, whose ground state can be prepared easily. In the original QAOA circuit, this Hamiltonian is simply $H_M = \sum_i^n X_i$, thus its ground state is the $|+\rangle^{\otimes n}$ state. The research for more sophisticated mixers led to the extension of this algorithm to the Quantum Alternating Operator Ansatz [44] and the Grover-mixer QAOA [48]. These variations improve the basic algorithm by restricting the size of the effective space the algorithm searches in. The general structure of a p-layer QAOA circuit with arbitrary state preparation and mixer is shown in Fig. 22.