Università degli studi di Torino

SCUOLA DI SCIENZE DELLA NATURA

Corso di Laurea Magistrale in Fisica dei Sistemi Complessi

Tesi di Laurea Magistrale

Tensor Network Based Generative Modeling

Relatore:
Prof. Carlini Alberto

Controrelatore:
Prof. Fariselli Piero

Candidato:
Robbiano Mattia

Anno Accademico 2024/2025

The amazing quote
that I chose as inspiration
for this work

Author, *Title*

Abstract

Generative models based on tensor networks offer a promising framework for capturing complex correlations in high-dimensional data. In this work, we propose a novel approach to generative modeling through global training of tensor network architectures, where every tensor element is optimized as a trainable parameter. Our method is evaluated on multiple datasets, and we consider two complementary loss functions: one rooted in explicit formulations and another in implicit ones. In particular, we reformulate the Maximum Mean Discrepancy (MMD) loss, typically an implicit method, into an explicit version via our tensor network representation—a strategy that aligns well with recent findings [**?**]. Additionally, we incorporate advanced measurement techniques using Positive Operator-Valued Measures (POVMs) to further enhance the model's capacity to represent intricate probability distributions. All experiments are implemented using Quimb and JAX. Our study provides insights into the trade-offs between implicit and explicit training approaches in quantum generative modeling and highlights the potential benefits of a unified tensor network framework.

Italian abstract

I modelli generativi basati su tensor network offrono un quadro promettente per la cattura di correlazioni complesse in dati ad alta dimensionalità. In questo lavoro, proponiamo un approccio innovativo alla generazione di dati attraverso l'addestramento globale di tensor network, in cui ogni elemento della rete è trattato come parametro ottimizzabile. Il nostro metodo viene testato su diversi dataset e analizziamo due funzioni di costo complementari: una formulata esplicitamente e una di natura implicita. In particolare, riformuliamo la funzione di costo Maximum Mean Discrepancy (MMD), solitamente implicita, in una versione esplicita sfruttando la rappresentazione mediante tensor network, in accordo con risultati recenti [**?**]. Inoltre, utilizziamo misurazioni avanzate basate su Positive Operator-Valued Measures (POVMs) per migliorare la capacità del modello di rappresentare distribuzioni di probabilità complesse. Tutti gli esperimenti sono implementati con Quimb e JAX. Questo studio offre un'analisi del compromesso tra approcci espliciti e impliciti nell'addestramento di modelli generativi quantistici e evidenzia i vantaggi di una formulazione unificata basata su reti tensoriali.

Contents

Chapter 1

Introduction

1.1 Introduction to Quantum Computing

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris. [?] [?] [?] [?] [?] [?]

1.2 Introduction to Tensor Network

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi.

Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

1.3 Introduction to Machine Learning and Generative Models

Generative models are a class of machine learning models designed to learn the underlying probability distribution of a given dataset and generate new samples that are statistically similar to those in the training data. A fundamental distinction in generative modeling is between explicit and implicit models, which differ in their approach to defining and optimizing the learned probability distribution.

Explicit models define an explicit probability distribution that can be directly evaluated, allowing for gradient-based optimization techniques that leverage likelihood-based loss functions such as the Kullback-Leibler (KL) divergence. Implicit models, on the other hand, do not require an explicit formulation of the probability distribution. Instead, they generate samples and optimize the model by comparing distributions through statistical measures like the Maximum Mean Discrepancy (MMD). In this chapter, we explore these two paradigms in the context of generative models for quantum states, highlighting their theoretical properties and implications for training efficiency.

Chapter 2

Theoretical Background

2.1 Description of generative model

2.1.1 Quantum Circuit Born Machine

Quantum Circuit Born machines are unsupervised generative models that aim to learn and represent the probability distributions of the dataset through pure quantum states. Through a circuit, we build a parametric ansatz ψ_θ, whose wavefunction represents the candidate probability distribution. We will call $D = \{x\}$ our sample dataset, and π the target distribution. We measure the projector on the computational base to get the trained probability distribution (through bitstring statistics).

$$p_{\boldsymbol{\theta}}(x) = |\langle x \mid \psi_{\boldsymbol{\theta}} \rangle|^2$$

The objective is to align the model probability distribution $p_{\boldsymbol{\theta}}$ with the target distribution π.

2.1.2 Tensor-Network Born Machine

Building on the connection between quantum circuits and tensor networks, the canonical form of a Matrix Product State (MPS) provides a simple and natural framework for constructing generative models analogous to Quantum Circuit Born Machines. This correspondence arises from the fact that isometric tensors in an MPS can be mapped to unitary operations in a quantum circuit. Specifically, the virtual bonds of the MPS, which have dimension D, can be interpreted as collections of q qubits, where $D = 2q$. The isometric tensors are then viewed as columns of a block unitary matrix with a fixed output qubit, assuming a local dimension $d = 2$. This mapping allows the MPS to be expressed through a block unitary circuit, where each block unitary is a matrix of size $dD \times dD$. We can generalized this framework to other tensor network architectures such as the Tree Tensor Network (TTN) and the Projected Entangled Pair State (PEPS) [**?**].

By leveraging this structural analogy, Tensor-Network Born Machines offer an alternative approach to generative modeling, rooted in the well-established tensor network formalism. In this representation every entry in the tensors composing the network is a possible parameter to be optimized. This "global" approach allows for the design of more expressive models.

We can divide training strategies in two main categories: local and global optimization. In a local optimization each tensor, or couple of tensors, is optimized independently, through a sweeping process that iterates over the network, similarly as in DMRG [?] [?]. In a global optimization, all the parameters are optimized simultaneously.

2.2 Explicit and Implicit Loss Functions

An important yet subtle distinction in generative modeling is between explicit and implicit models [?]. Explicit generative models allow for efficient access to the model probability distribution $Q_\theta(x)$ for any data sample x. Here, "efficient" means that the probability values can be computed in polynomial time and memory with respect to the size of the data. Once both model's and dataset's probability distributions are knows, a likelihood-based objective function is optimized. A common choice is the minimization of the Kullback-Leibler (KL) divergence:

$$D_{KL}(P\|Q) = \sum_x P(x) \log \frac{P(x)}{Q_\theta(x)},$$

where $P(x)$ is the target distribution, and $Q_\theta(x)$ is the model's distribution parameterized by θ.
In contrast, implicit generative models do not provide direct access to $Q_\theta(x)$ but instead allow for efficient sampling from the model distribution. Training is done by comparing the model's samples to the data distribution through a distance measure. One widely used statistical measure for training implicit models is the Maximum Mean Discrepancy (MMD), which quantifies the difference between two distributions based on their embeddings in a reproducing kernel Hilbert space (RKHS). The MMD is defined as:

$$\text{MMD}(P, Q_\theta) = \mathbb{E}_{x,x'\sim P}[k(x, x')] - 2\mathbb{E}_{x\sim P, y\sim Q_\theta}[k(x, y)] + \mathbb{E}_{y,y'\sim Q_\theta}[k(y, y')],$$

where $k(x, y)$ is a positive-definite kernel function.
A prominent example of an implicit generative model is Generative Adversarial Networks (GANs), whose training makes use of the samples produced by the generator to foul the discriminator. Quantum Circuit Born Machines, when implemented on quantum device, cannot grant explicit access to $Q_\theta(x)$ but only to samples drawn from the model distribution, measuring the final state of the circuit.

2.2.1 Focus on Maximum Mean Discrepancy

To train the QCBM, we use the squared maximum mean discrepancy (MMD) as the loss function.

$$\mathcal{L}(\boldsymbol{\theta}) = \left\| \sum_x p_{\boldsymbol{\theta}}(x)\phi(x) - \sum_x \pi(x)\phi(x) \right\|^2$$

where $\phi(x)$ maps x to a larger feature space. Using a kernel $K(x, y) = \phi(x)^T\phi(y)$ allows us to work in a lower-dimensional space. We use the Radial basis function (RBF) kernel for this purpose, which is defined as:

$$K(x, y) = \frac{1}{c}\sum_{i=1}^{c} \exp\left(\frac{|x - y|^2}{2\sigma_i^2}\right)$$

Here, σ_i is the bandwidth parameter controlling the width of the Gaussian kernel. $\mathcal{L}$ approaches to zero if and only if $p_{\boldsymbol{\theta}}$ approaches π. We can now write the loss function in terms of $K(x, y)$ as

$$\mathcal{L} = \mathop{\mathbb{E}}_{x,y\sim p_\theta}[K(x,y)] - 2\mathop{\mathbb{E}}_{x\sim p_\theta, y\sim\pi}[K(x,y)] + \mathop{\mathbb{E}}_{x,y\sim\pi}[K(x,y)]$$

For discrete distributions, where p and q are represented as vectors (or histograms) and the kernel matrix K is computed over a discrete "space" [1], [2], the above expression reduces to:

$$(p_x - p_y)^\top K (p_x - p_y)$$

This is the expression used to calculate MMD in case of a QCBM, and involves summing over the set of all possible bitstrings that the model can generate, an exponential number of terms. This is computationally infeasible for large systems. In our "dequantized" tensor network representation we wanto to find a way to calculate the MMD loss function efficiently, leveraging tensor network contractions.

2.3 Efficient loss calculation with tensor network

The Maximum Mean Discrepancy (MMD) is a distance measure between two probability distributions based on the expectation values of a positive semi-definite kernel function $K(x, y)$. In our context, it is used to compare the distribution $q_\theta(x)$, generated by a quantum model, with a target distribution $p(x)$ obtained from classical data. The data samples y are bitstrings, i.e., elements of the computational basis.

$$\mathrm{MMD}(\theta) = \sum_{x\in\Omega}\sum_{y\in\Omega} q_\theta(x)K(x,y)q_\theta(y) + \sum_{x\in T}\sum_{y\in T} p(x)p(y)K(x,y) - 2\sum_{x\in\Omega}\sum_{y\in T} q_\theta(x)K(x,y)p(y), \tag{2.1}$$

where the Born rule is applied as $q_\theta(x) = |\langle x|\psi\rangle|^2$. The optimization problem can then be expressed as:

$$\min_\theta \mathrm{MMD}(\theta) = \min_\theta \left[\sum_{x,y\in\Omega} |\langle x|\psi\rangle|^2 K(x,y) |\langle y|\psi\rangle|^2 - 2\sum_{x\in\Omega}\sum_{y\in T} |\langle x|\psi\rangle|^2 K(x,y)p(y) \right]. \tag{2.2}$$

We can manually contruct the TN that represents the sampling probabilities of the state $|\psi\rangle$. For example, for a single biststring $b = b_1 \dots b_n$, the probability of sampling it is given by

$$p(b) = |\langle\psi|b\rangle|^2 = \langle\psi|b\rangle\langle b|\psi\rangle = \langle\psi|\Pi_b|\psi\rangle$$

where in the last equation we defined the projector $\Pi_b = |b\rangle\langle b|$ corresponding to bitstring b. Now, focus on a single qubit and imagine we put together in a list the projectors on state $|0\rangle$ and $|1\rangle$, thus obtaining

$$\Pi = [\Pi_0, \Pi_1], \quad \Pi_0 = |0\rangle\langle 0|, \Pi_1 = |1\rangle\langle 1|$$

This object Π is a tensor of dimensions (2, 2, 2), and it is essentially a representation of the POVM corresponding to measuring in the computational basis Z. One can then

construct a TN that encodes the sampling probabilities from state $|\psi\rangle$ by using the formula above

$$p(b_1 b_2 \dots b_n) = \langle\psi|\Pi_{b_1} \otimes \Pi_{b_2} \dots \otimes \Pi_{b_n}|\psi\rangle$$

where each of the local Π_b is the (2,2,2) tensors introduced above. Imporantly, this is a tensor networks where the open legs are the outcomes one b_i. For a single-qubit space, measuring only in Z basis, this corresponds to

$$\left[\begin{array}{cc|cc} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{array}\right]$$

For each node of the tensor network. We can create the single node and put them together in an MPO

The expression $p(b_1 b_2 \dots b_n) = \langle\psi|\Pi_{b_1} \otimes \Pi_{b_2} \dots \otimes \Pi_{b_n}|\psi\rangle$ can be built as:

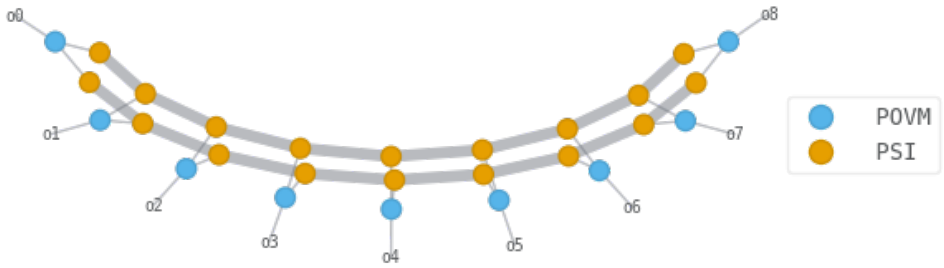


Figure 2.1: Tensor network representation of the sampling probabilities $p(b_1 b_2 \dots b_n)$. Each node corresponds to a local projector Π_{b_i}, and the open legs represent the outcomes b_i.

The expression above become:

$$= \min \left[\sum_{x,y} \langle\psi|\Pi|\psi\rangle \, K(x,y) \, \langle\psi|\Pi|\psi\rangle - 2 \sum_x \sum_y \langle\psi|\Pi|\psi\rangle \, K(x,y) p(y) \right]$$

The kernel matrix K is defined as

$$K = \begin{pmatrix} 1 & e^{-\frac{1}{2\sigma^2}} \\ e^{-\frac{1}{2\sigma^2}} & 1 \end{pmatrix}$$

which encodes similarity between bitstrings using a Gaussian function. This matrix acts on the output space indexed by x and y, and can be realized as a kernel MPO.
In full tensor index notation, the MMD becomes:

$$= \psi_b \, \Pi_k^{b,o} \, \psi^k \, K_o^{o'} \, \psi_b \, \Pi_k^{b,o'} \, \psi^k - 2 \sum_B \left[\psi^b \, \Pi_b^{ko} \, \psi_k \, K_o^{o'} \, B^o \right]$$

Here, ψ^k and ψ_b are the components of the quantum state $|\psi\rangle$ and its dual $\langle\psi|$, respectively. Indices are placed according to the usual tensor convention: kets carry upper indices and bras lower indices. The operator Π is a tensor with indices $\Pi_k^{b,o}$, encoding its action on both input and measurement bases. The sum over B denotes averaging over the target data bitstrings y, with B^o representing the empirical frequency vector from data.

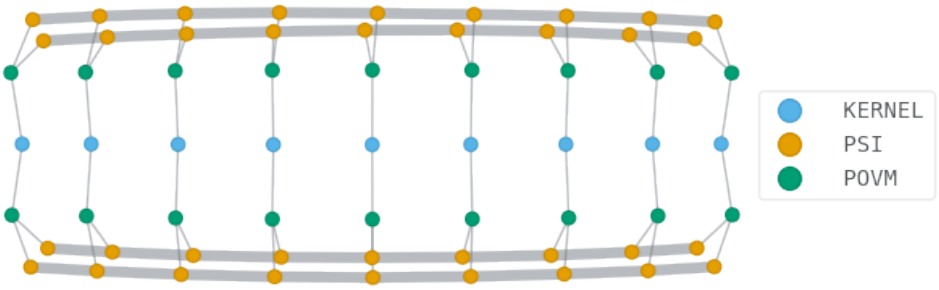


Figure 2.2: Tensor network representation of the first term in the MMD expression. The kernel matrix K acts on the output space indexed by x and y, and can be realized as a kernel MPO.

The first term can be built as:
The second term needs the MPS representation of data. We are going to build a "computational MPS" for each bitstring in the dataset, which can be thought as the eigenstate that gives every time we sample the original bitstring. To compute efficiently the sum over the all possible bitstrings, we can group all the computational MPS in an hyperindexed tensor network, quimb will then sum over the hyperindex efficently.

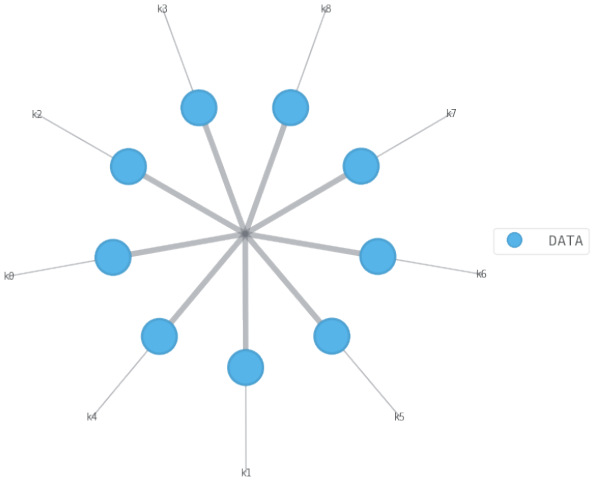


Figure 2.3: The data bitstrings are represented as computational MPS, and the hyperindex allows efficient summation over all possible bitstrings.

The mixed term of the MMD can be built as:

2.3.1 Appendix: other measures

MMD as MPO as sum of operators of different bodyness

Let's consider the MMD loss function defined as:

$$\mathcal{L}_{\text{MMD}}(\boldsymbol{\theta}) = \mathop{\mathbb{E}}_{x,y\sim p_\theta}[K(x,y)] - 2\mathop{\mathbb{E}}_{x\sim p_\theta, y\sim\pi}[K(x,y)] + \mathop{\mathbb{E}}_{x,y\sim\pi}[K(x,y)] = \sum_{\boldsymbol{x},\boldsymbol{y}\in\mathcal{X}} q_{\boldsymbol{\theta}}(\boldsymbol{x}) q_{\boldsymbol{\theta}}(\boldsymbol{y}) K(\boldsymbol{x},\boldsymbol{y}) - 2\sum_{\boldsymbol{x},\boldsymbol{y}\in\mathcal{X}} q_{\boldsymbol{\theta}}(\boldsymbol{x}) p$$

where

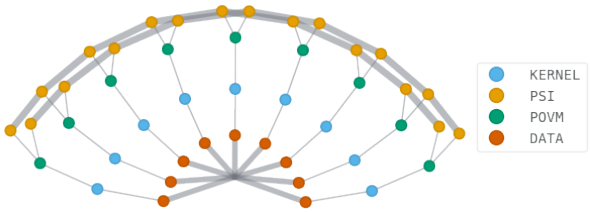


Figure 2.4: Tensor network representation of the mixed term in the MMD expression. The kernel matrix K acts on the output space indexed by x and y, and can be realized as a kernel MPO.

$$K_\sigma(\boldsymbol{x}, \boldsymbol{y}) = e^{-\frac{\|\boldsymbol{x}-\boldsymbol{y}\|_2^2}{2\sigma}} = \prod_{i=1}^{n} e^{-\frac{(x_i-y_i)^2}{2\sigma}},$$

We can rewrite each term in MMD loss as an expectation value of an observable:

$$\mathcal{M}(\rho, \rho') = \mathrm{Tr}\left[O_{\mathrm{MMD}}^{(\sigma)}(\rho \otimes \rho')\right]$$

where

$$O_{\mathrm{MMD}}^{(\sigma)} := \sum_{\boldsymbol{x},\boldsymbol{y}} K_\sigma(\boldsymbol{x}, \boldsymbol{y}) |\boldsymbol{x}\rangle\langle\boldsymbol{x}| \otimes |\boldsymbol{y}\rangle\langle\boldsymbol{y}|$$

We can also rewrite the observable as the sum of Puli strings, each one containing

$$O_{\mathrm{MMD}}^{(\sigma)} = \sum_{l=0}^{n} \binom{n}{l} (1-p_\sigma)^{n-l} p_\sigma^l D_{2l}$$

where

$$D_{2l} = \frac{1}{\binom{n}{l}} \sum_{\substack{A \subseteq \mathcal{N} \\ |A|=l}} \bigotimes_{i \in A} (Z_i \otimes Z_{n+i})$$

where the summation runs over all possible combinations of strings of length l $A \subseteq \mathcal{N} = \{1, 2, \ldots, n\}$ and $p_\sigma = \left(1 - e^{-1/(2\sigma)}\right)/2$.

Figure 2.5: We contract the model (in blue) and the data (green) with the MPO representing the loss function (organge). - figure to bi fixed, I didn't noticed the legend was not rendered

Local quantum fidelity

$L_{aF}^{(e)} = 1 - \langle N_O| H_L |\varphi\rangle$, where $H_L = \frac{1}{m}\sum_{i=1}^{m} |0\rangle\langle 0| \otimes 1\big|_{\bar{i}}$, whare $\bar{i}$ undicale all qubits except $i =$

$$= \frac{1}{m}\left[\begin{pmatrix} |0\rangle\langle 0| \\ \mathbb{I} \\ \mathbb{I} \\ \vdots \end{pmatrix} + \begin{pmatrix} \mathbb{I} \\ |0\rangle\langle 0| \\ \mathbb{I} \\ \mathbb{I} \\ \vdots \end{pmatrix} + \ldots + \begin{pmatrix} \mathbb{I} \\ \mathbb{I} \\ \cdots \\ |0\rangle\langle 0| \end{pmatrix}\right] =$$

$$\begin{bmatrix} \begin{pmatrix} 1 & 0 \\ 0 & \frac{n-1}{m} \end{pmatrix} \\ \begin{pmatrix} 1 & 0 \\ 0 & \frac{n-1}{m} \end{pmatrix} \\ \vdots \end{bmatrix} = MPO$$

Chapter 3

Training

3.1 Training

METTERE SOPRA DATASET DEL PAPER E SOTTO BS

The training process optimizes the model parameters to align the generated probability distribution with the target distribution of a dataset. We perform training on two toy datasets—*bars and stripes* and *cardinality*. The *bars and stripes* dataset consists of bitstrings that represent distinct horizontal or vertical bar and stripe patterns, while the *cardinality* dataset contains bitstrings constrained to have a fixed number of nonzero entris. Each bitstring is mapped to a Matrix Product State (MPS) that deterministically outputs the corresponding bitstring upon sampling.

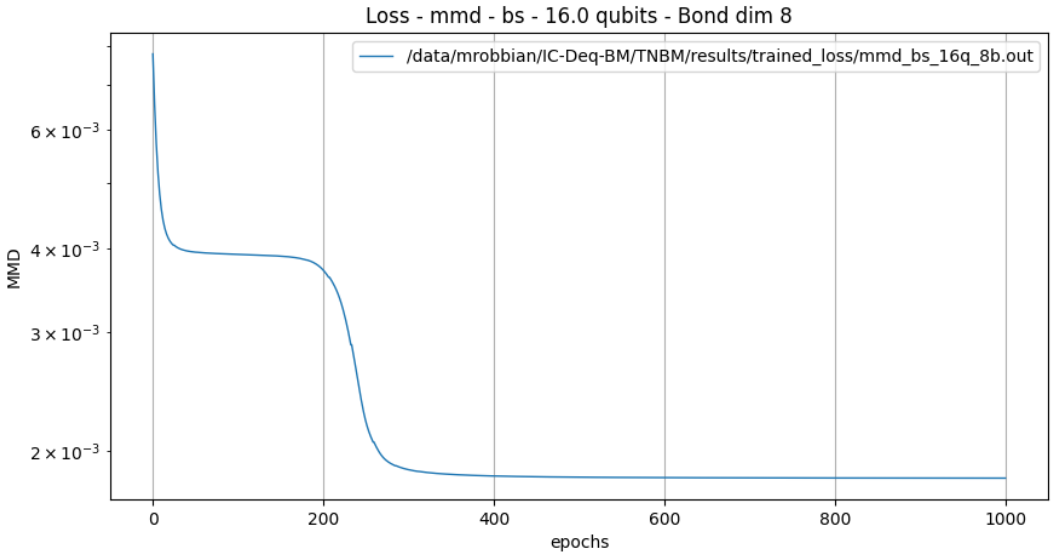
Training is performed globally, meaning that every element in every tensor of the network is treated as a trainable parameter. We optimize the model using two different loss functions: the Maximum Mean Discrepancy (MMD) and the Kullback-Leibler divergence (DKL).

For MMD-based training, we construct a hyper-indexed tensor network over the training samples and compute contractions as detailed in the previous chapter. Figure 3.1 illustrates the training process using the MMD loss, showing the loss evolution and corresponding generated samples.

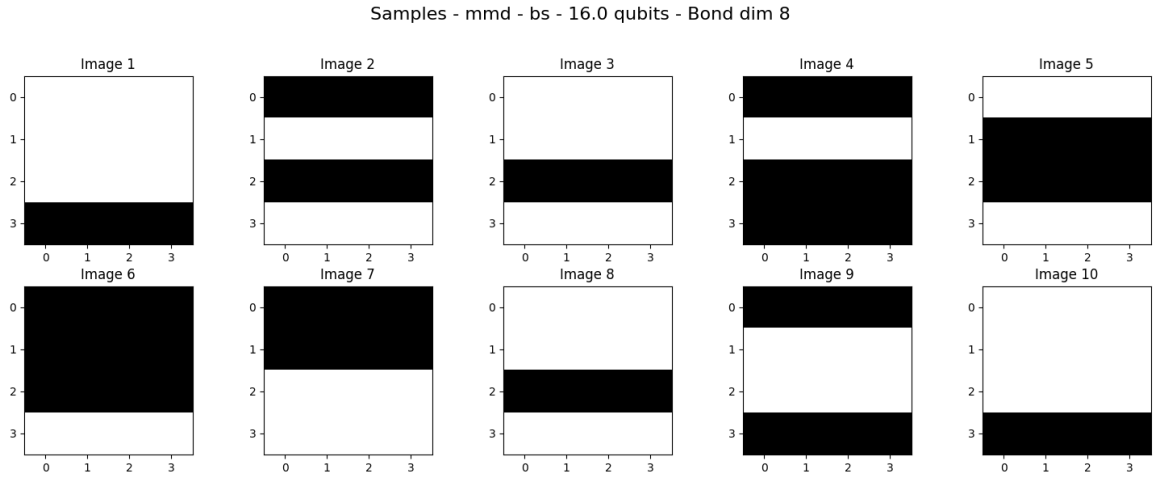
For DKL-based training, we compute the divergence for each bitstring, sum over all samples, and normalize by the total number of samples. Figure **??** shows the training evolution and generated samples when optimizing with the DKL loss.

From the training results, we observe that the model trained with the MMD loss exhibits superior performance compared to the one trained with DKL, given the same number of iterations. This improvement is evident both in the loss trajectory, which shows more stable convergence, and in the generated samples, which more accurately reflect the structure of the target distribution. As discussed in the previous chapter, MMD is generally an implicit method. However, in our case, due to the nature of tensor networks, we have formulated it in an explicit manner.

Previous work [**?**] has indicated that it can yield improved performance in certain contexts. Our results are in line with these findings, suggesting that the MMD-based training may offer advantages even when reinterpreted in an explicit framework. In our case, the tensor network representation allows us to recast the method explicitly, which appears to further benefit the model's performance.



(a) Training progress using MMD loss.



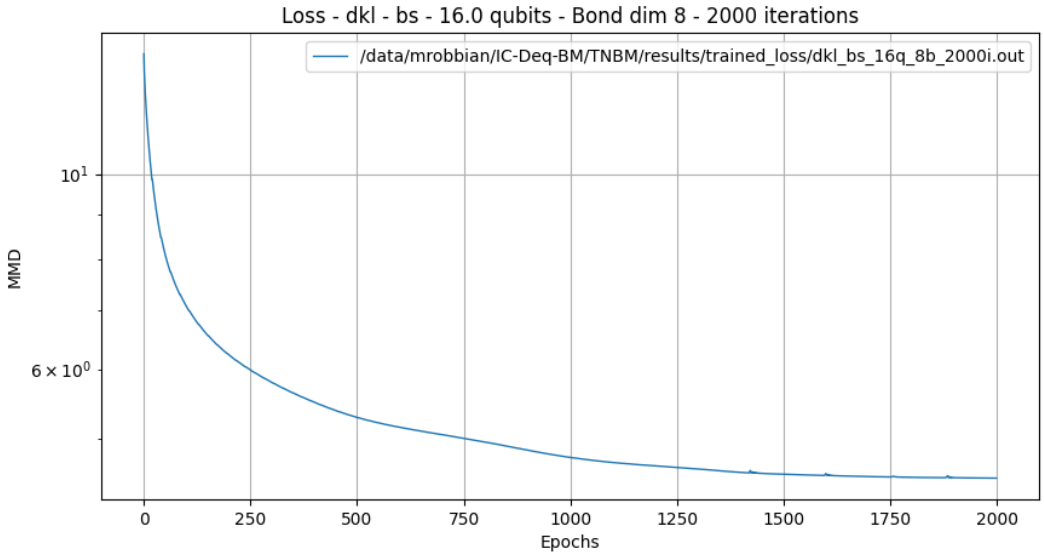
(b) Generated samples using MMD loss.

Figure 3.1: Training results and corresponding generated samples for MMD-based training.

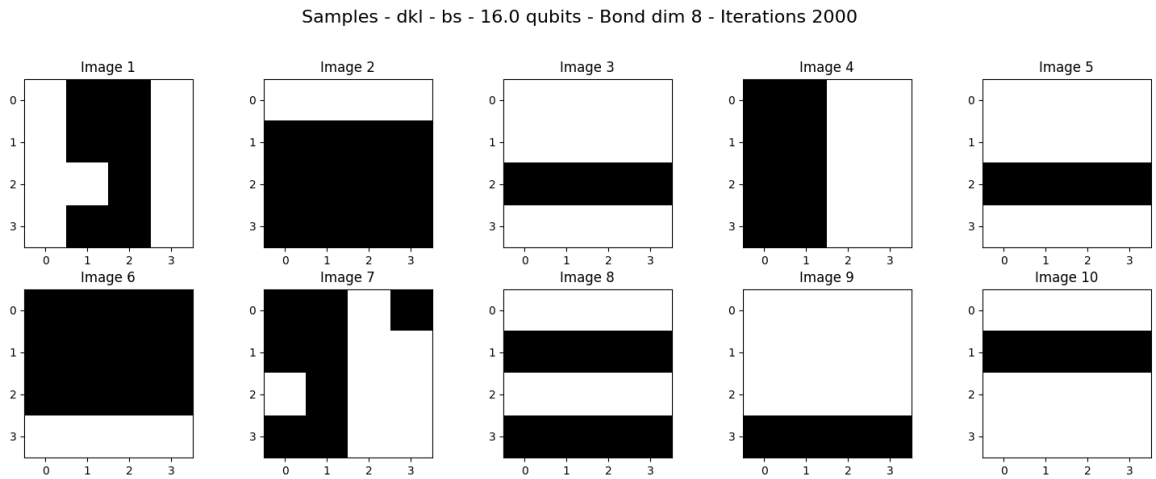
3.2 Barren Plateau Analysis

One of the main challenges in optimizing parametrized quantum models is the presence of barren plateaus, where the gradients of the cost function vanish exponentially with system size, making training intractable. To investigate whether our approach suffers from this phenomenon, we analyze the variance of the loss function under random initialization.

For a fixed bond dimension, we compute the variance of the loss function across different random initializations while increasing the number of qubits. This process is repeated for multiple bond dimensions to assess whether the effect depends on the expressivity of the tensor network. The results, shown in Fig. 3.3, indicate that the variance decreases exponentially as the system size grows, suggesting the presence of a barren plateau. Previous work [**?**] has shown that while barren plateaus are a known issue for Matrix Product States (MPS), the phenomenon is mitigated when using tensor network architectures such



(a) Training progress using DKL loss.



(b) Generated samples using DKL loss.

Figure 3.2: Training results and corresponding generated samples for DKL-based training.

as Tree Tensor Networks (TTN) and Multi-scale Entanglement Renormalization Ansatz (MERA). Our results align with these findings, demonstrating a similar exponential decay in variance for MPS, but with potential for better performance with other tensor network structures.

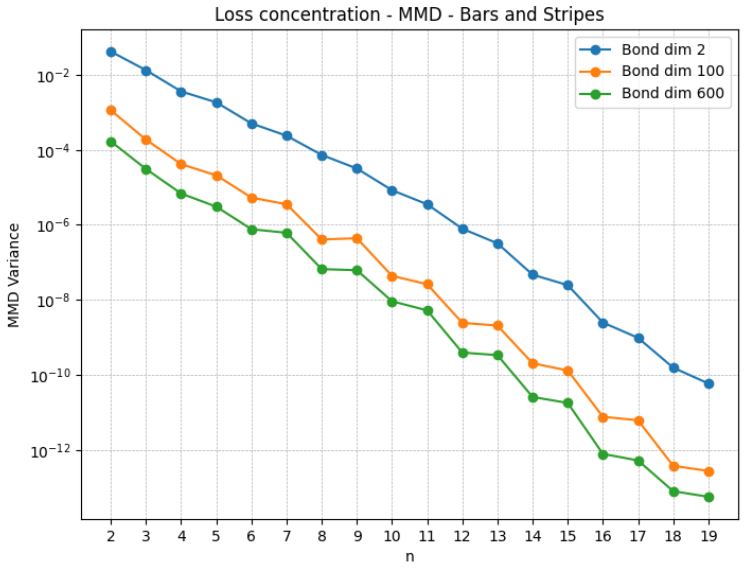


Figure 3.3: Variance of the loss function at random initialization as a function of the number of qubits, for different bond dimensions. The exponential decay suggests the presence of a barren plateau.

List of Figures

List of Tables