2020

Tensor Network States: Optimizations and Applications in Quantum Many-Body Physics and Machine Learning

Justin Reyes
University of Central Florida

TENSOR NETWORK STATES: OPTIMIZATIONS AND APPLICATIONS IN QUANTUM MANY BODY PHYSICS AND MACHINE LEARNING

by

JUSTIN REYES
B.S. in Physics, University of Central Florida, 2015

A dissertation submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy
in the Department of Physics
in the College of Sciences
at the University of Central Florida
Orlando, Florida

Summer Term
2020

Major Professor: Eduardo Mucciolo

ABSTRACT

Tensor network states are ubiquitous in the investigation of quantum many-body (QMB) physics. Their advantage over other state representations is evident from their reduction in the computational complexity required to obtain various quantities of interest, namely observables. Additionally, they provide a natural platform for investigating entanglement properties within a system. In this dissertation, we develop various novel algorithms and optimizations to tensor networks for the investigation of QMB systems, including classical and quantum circuits. Specifically, we study optimizations for the two-dimensional Ising model in a transverse field, we create an algorithm for the k-SAT problem, and we study the entanglement properties of random unitary circuits. In addition to these applications, we reinterpret renormalization group principles from QMB physics in the context of machine learning to develop a novel algorithm for the tasks of classification and regression, and then utilize machine learning architectures for the time evolution of operators in QMB systems.

This dissertation is dedicated to my Lord and Savior Jesus Christ, "in whom are hidden all the treasures of wisdom and knowledge" – Colossians 2:3

ACKNOWLEDGMENTS

The completion of this dissertation has been made possible by the determination and dedication of a few special people in my life. I am thankful first to my Lord Jesus Christ, who has blessed me with the opportunity to pursue this degree, and also supported me through the process.

Secondly, I am grateful to my wife Danielle Reyes for her consistent motivation and support. I would have given up on this endeavor long ago without her. She is my biggest supporter in all of my academic adventures.

I would like to thank my family for encouraging me and remaining by my side even if my research made little to no sense to any of them.

Finally, I would like to thank my advisor Dr. Eduardo Mucciolo, for all of his hard work in guiding me through the process of becoming a doctor of physics. He has been a consistent example to me of the dedication, endurance, patience, and humility necessary to appropriately perform scientific research. Without him, I would have no idea what I am doing.

A special thank you to Dr. Andrei Ruckenstein, Dr. Claudio Chamon, Dr. Stefanos Kourtis, and Lei Zhang for collaborating with me on many projects, specfically the work detailed in chapters 5, 6, and 7. In each of these cases, the team was instrumental in both proposing novel ideas and in modifying the implementations of code throughout each project. Similarly, thank you to Dr. Dan Marinescu for working patiently with me on my very first project, for which I am sure my inexperience showed greatly. Thank you to Dr. Miles Stoudenmire for your collaboration on the project found in chapter 8, providing me access to your iTensor library and coming up with the ideas behind the network architecture and its applications. This truly allowed me to pursue my interests in machine learning. Also, thank you to Sayandip Dhara for providing the necessary

input data for the machine learning tasks performed in chapter 9, and also for being an office mate, a friend, and someone to discuss research ideas with.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION TO TENSOR NETWORK STATES

Introduction

The work in this dissertation is interdisciplinary in nature, drawing from the disciplines of physics, mathematics, and computer science. As such, it is necessary to provide a review of a few key concepts from these disciplines which are relevant to the projects contained herein. In chapter 1, we will first introduce tensor networks in the context of quantum many-body physics. In chapter 2, we will review important features of machine learning, particularly focusing on the use of neural networks. We will finish the review material in chapter 3 by highlighting a few key elements in quantum circuit modelling. All of these areas of study will come together in the projects discussed in chapters 4 – 9. Finally, summary conclusions and future work will be presented in chapter 10.

A Limitation in Quantum Many-Body Systems

We begin our review by discussing some fundamental notions from quantum many-body (QMB) physics. The study of QMB physics is primarily concerned with understanding how emergent (macroscopic) properties arise from microscopic descriptions of quantum systems involving many particles [1]. Examples of such investigations within the context of condensed matter physics include understanding the mechanisms behind high-T_c superconductivity [2], determining the bounds on the transition between many-body localized (MBL) and eigenstate thermalization hypothesis (ETH) phases [3], and understanding topologically order phases [4], among others. Often, in order to find solutions to these problems, simplified models are constructed as a representation of the system in question. Though they are not full descriptions of the system, these models are defined in such a way that they implement all the relevant interactions being considered in the problem.

Some particular models of interest which are ubiquitous in condensed matter physics are the Hubbard model, the Heisenberg model, and the Anderson model [5].

When specifying a model, it is necessary to define a Hamiltonian operator $\mathscr{H}$. This operator acts on a system state vector $|\psi\rangle$ according to the Schrodinger wave equation as

$$\mathscr{H}|\psi\rangle = i\frac{d|\psi\rangle}{dt}, \tag{1.1}$$

with $i^2 = -1$ (we work with units such that the rationalized Planck constant $\hbar = 1$). This Hamiltonian operator measures the total energy within the system. When it is associated with a basis set $\{|\phi_i\rangle\}$, the Hamiltonian can be represented by a matrix with elements given by

$$\mathscr{H}_{ij} = \langle\phi_i|\mathscr{H}|\phi_j\rangle. \tag{1.2}$$

The choice of basis determines the vector space that the Hamiltonian acts on. This vector space over complex numbers is called a Hilbert space [6]. A basis of particular interest is the one in which the basis states satisfy the time independent Schrodinger wave equation

$$\mathscr{H}|\phi_j\rangle = E_j|\phi_j\rangle. \tag{1.3}$$

The states contained within this basis are said to be *eigenstates* of the Hamiltonian, with eigenvalues E_j corresponding to their associated energies [6]. In this basis, the Hamiltonian matrix is

diagonal and the time dependence of each basis state can be given as

$$|\phi_j(t)\rangle = e^{-iE_j t} |\phi_j\rangle . \tag{1.4}$$

It is clear from Eq. 1.4 that the time dependence for eigenstates is trivial, producing changes only in the phase of the state. For this reason, eigenstates are also called *stationary states* [6]. If this basis set is known, the full state of the system $|\Psi\rangle$ at a time t can be expanded as

$$|\Psi(t)\rangle = \sum_j C_j |\phi_j(t)\rangle, \tag{1.5}$$

with C_j being a complex-valued probability amplitude such that the probability $P(\phi_j)$ of measuring energy E_j for state $|\phi_j\rangle$ is

$$P(\phi_j) = |C_j|^2. \tag{1.6}$$

From this expansion, the expectation value of any relevant quantity given by the action of an operator $\mathscr{O}$ can be calculated directly as

$$\langle \mathscr{O} \rangle = \langle \Psi(t)| \mathscr{O} |\Psi(t)\rangle = \sum_j |C_j|^2 \langle \phi_j(t)| \mathscr{O} |\phi_j(t)\rangle . \tag{1.7}$$

However, it is often the case (particularly when considering systems involving many particles), that the Hamiltonian cannot be exactly diagonalized, and approximate or numerical methods must be employed [5, 7]. The reason for this is two-fold. Firstly, the Hamiltonian may be composed of

operators $\hat{h}_i$, e.g., $\mathscr{H} = \sum_i \hat{h}_i$, which do not commute, namely,

$$\left[\hat{h}_i, \hat{h}_j\right] = \hat{h}_i\hat{h}_j - \hat{h}_j\hat{h}_i \neq 0 \text{ for } i \neq j. \tag{1.8}$$

The non-commutability of these operators guarantees that a complete set of simultaneous eigenstates cannot be found [5, 7], making it difficult to determine the eigenstates of $\mathscr{H}$. The second reason is that the size of the Hilbert space is known to grow exponentially with the number of particles under consideration [5, 7]. This feature prevents the numerical studies of larger system sizes. While the former limitation is an analytic consequence determined by the definition of a given model, the latter is a computational limitation that can be overcome in part or minimized by reformulating the problem statement in the language of tensor networks.

Tensor Network States

To understand tensor network states and their use in QMB physics, it is first necessary to introduce the mathematical structure of tensors. A tensor is an algebraic structure which serves as high-order generalization of a vector [8]. By this it is meant that a tensor is a collection of scalar quantities (elements) which are tabulated by a set of indices $\vec{v} = \{v_1, v_2, v_3, ...\}$. The *order* of a tensor (some times called rank) is given by the number of indices necessary to specify a unique element within the tensor [8]. Under this definition, a scalar, a vector and a matrix are equivalent to a zeroth-order, first-order, and second-order tensor, respectively. While the indices of a tensor may have specific designations (e.g., space, time, momentum, spin, etc) and be related by specific transformation properties, here we do not make any assumption in both regards. Formally, a tensor of arbitrary order can be constructed by successively taking the Cartesian products of index sets defined over vector spaces. Various operations can be applied to tensors. Take, for example, a third-order tensor

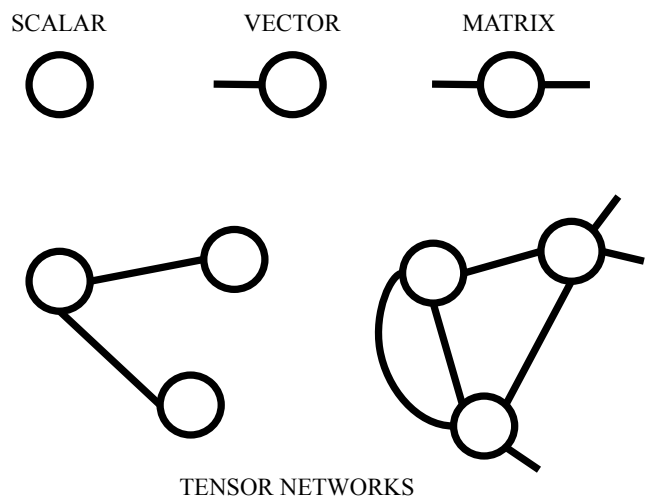


Figure 1.1: Top: Tensor diagrams for a scalar, a matrix, and an order-3 tensor. Bottom: Tensor network diagrams involving three tensors generated from the decomposition of a scalar (left) and an order-3 tensor (right).

T_{ijk}. This tensor can me multiplied by a scalar factor α as

$$T'_{ijk} = \alpha T_{ijk}, \tag{1.9}$$

or added with another tensor of equal order, S_{ijk}, as

$$(T+S)_{ijk} = T_{ijk} + S_{ijk}, \tag{1.10}$$

or contracted with another tensor of arbitrary order. This final operation is performed as a summation over all possible values taken by common indices between the two tensors [8]. As a specific

example, contracting T_{ijk} with a second-order tensor, S_{kl}, is given as

$$(TS)_{ijl} = \sum_k T_{ijk} S_{kl}. \tag{1.11}$$

Notice that the result is itself another tensor, composed of indices which were not repeated between tensors T and S. In light of this, we can say that any tensor can be rewritten (decomposed) as a set of tensors which are to have a specific pattern of contraction (geometry). This set of tensors is called a *tensor network*. Before continuing further, it is beneficial to introduce a graphical notation for tensors and tensor networks. We first note that a tensor network (TN) is specified by the common set of indices between various tensors in the network. In the example above, k is the common index in a network composed of two tensors T and S. This set of common indices defines a geometry in the network. In the bottom panel of Fig. 1.1, an order-0 and an order-3 tensor are decomposed into arbitrary TN geometries. In these diagrams, the circles represent tensors, while the lines represent indices. Lines which connect two tensors are indicative of an internal (common) bond, while lines which are disconnected represent open bonds. When two tensors are drawn with a common bond, then a contraction operation is taken along that bond.

Having conceptually introduced tensors and tensor networks, we will discuss their use in QMB systems. In general, a TN can be used to model a quantum state expressed as the tensor product of N particles as

$$|\Psi\rangle = \sum_{i_1,i_2,\ldots,i_N} C_{i_1,i_2,\ldots,i_N} |i_1\rangle \otimes |i_2\rangle \otimes \ldots \otimes |i_N\rangle \tag{1.12}$$

by performing a TN decomposition on the multi-indexed coeffecient $C_{i_1,i_2,\ldots,i_N}$ [8]. Each site, i_k, in the system becomes associated to a single site in the tensor network. The internal bonds forming the network itself are imposed by the interactions defined in the Hamiltonian. These interactions induce a specific geometry on the network, such that open bonds, $\{i_1, i_2, \ldots, i_N\}$, represent the orig-

inal physical indices of the system (i.e., local degrees of freedom), while internal bonds represent inter-particle entanglement [8]. This representation is very advantageous for systems involving only local interactions. In what follows, we will discuss ubiquitous TN geometries, describing their definitions and properties. The particular TNs we will focus on are the Matrix Product States (MPS) and the Projected Entangled Pair States (PEPS).

Matrix Product States

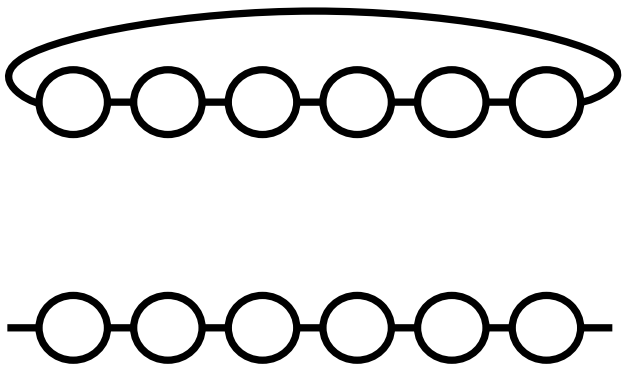


Figure 1.2: A tensor diagram for a six-site MPS with periodic boundary conditions (top), and a tensor diagram for a six-site MPS with open boundary conditions (bottom).

Matrix Product States (MPS) are TNs that represent a one-dimensional array of tensors [8]. In Fig. 1.2, diagrams for MPS with *periodic* and *open* boundary conditions are shown. This TN has some interesting properties that make it particularly useful for QMB physics.

The first property of note is that the MPS decomposition of a one-dimensional quantum system can

be accomplished by successive Schmidt decompositions. The decomposition of a quantum state $|\Psi\rangle$ into two orthonormal Schmidt vectors $|\Phi^L_\alpha\rangle$ and $|\Phi^R_\alpha\rangle$ is given by

$$|\Psi\rangle = \sum_{\alpha=1}^{D} \lambda_\alpha \, |\Phi^L_\alpha\rangle \otimes |\Phi^R_\alpha\rangle \,, \tag{1.13}$$

where λ_α are Schmidt coefficients ordered in descending order [8]. For an N particle system, where each particle has physical dimension p, Schmidt decompositions are performed successively over each particle. Beginning with the full state $|\Psi\rangle$, the first particle i_1 is separated from the remaining $N-1$ particles as

$$|\Psi\rangle = \sum_{\alpha_1}^{\min(p,D)} \lambda_{\alpha_1} \, |\Phi^{(1)}_{\alpha_1}\rangle \otimes |\Phi^{(2,\ldots,N)}_{\alpha_1}\rangle \,. \tag{1.14}$$

The Schmidt vector $|\Phi^{(1)}_{\alpha_1}\rangle$ is then written in the original physical basis set as

$$|\Phi^{(1)}_{\alpha_1}\rangle = \sum_{i_1} \Gamma^{i_1}_{\alpha_1} \, |i_1\rangle \,. \tag{1.15}$$

Following this, particle i_2 is decomposed from the $N-2$ remaining particles

$$|\Phi^{(2,\ldots,N)}_{\alpha_1}\rangle = \sum_{\alpha 2}^{\min(p,D)} \lambda_{\alpha_2} \, |\Phi^{(2)}_{\alpha_1\alpha_2}\rangle \otimes |\Phi^{(3,\ldots,N)}_{\alpha_2}\rangle \,, \tag{1.16}$$

with a subsequent change of basis in the same manner as i_1,

$$|\Phi^{(2)}_{\alpha_1\alpha_2}\rangle = \sum_{i_2} \Gamma^{i_2}_{\alpha_1,\alpha_2} \, |i_2\rangle \,. \tag{1.17}$$

By iterating this procedure over every site on the lattice, one obtains a final decomposition as

$$|\Psi\rangle = \sum_{\{i\}} \sum_{\{\alpha\}} \left(\Gamma^{i_1}_{\alpha_1} \lambda_{\alpha_1} \Gamma^{i_2}_{\alpha_1,\alpha_2} \lambda_{\alpha_2} \ldots \lambda_{\alpha_{N-1}} \Gamma^{i_{N-1}}_{N-1} \right) |i_1\rangle \otimes |i_2\rangle \otimes \ldots \otimes |i_N\rangle \tag{1.18}$$

Such a procedure introduces what is called the *canonical form* of the MPS [8]. In practice, the Schmidt decomposition can be accomplished by a singular value decomposition (SVD),

$$A_{ij} = \sum_{\alpha} U_{i\alpha} \lambda_{\alpha} V^{\dagger}_{\alpha j} \tag{1.19}$$

where A_{ij} is the original matrix with singular values λ_{α}. By multiplying these singular values to U or $V^{\dagger}$ at each iteration of the full Schimdt decomposition procedure, the MPS can be brought to left or right canonical form, respectively [8]. This procedure is shown in Fig. 8.4. This form is useful in that eigenvalues of partitions in the lattice are taken as the squares of Schmidt coefficients and truncation (approximation) processes over the system are reduced to a selection of the largest D values of Schmidt coefficients.

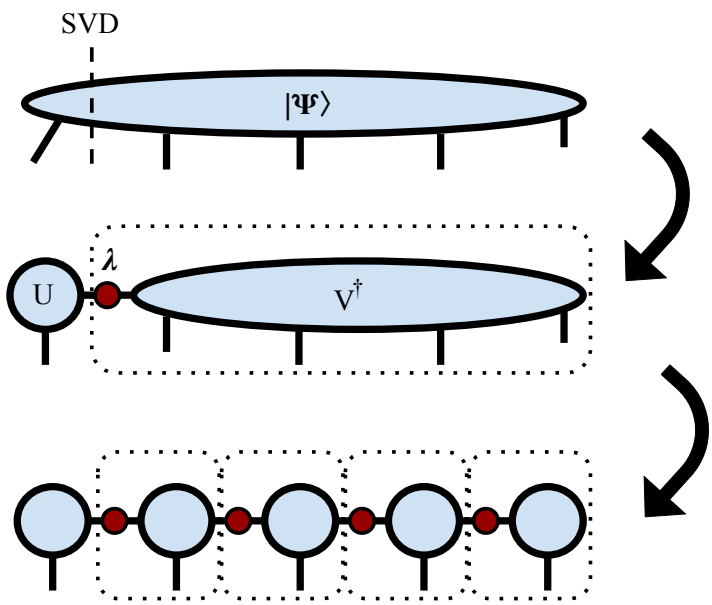


Figure 1.3: A diagrammatic representation of the successive Schmidt decomposition steps which transform an initial state $|\Psi\rangle$ into a right canonical MPS. First, an SVD is performed between site one and the rest of the system. Then, the singular values λ are multiplied to the right with $V^{\dagger}$. These steps are repeated over every site until the decomposition is complete.

Another property that makes MPS useful is that they have the ability to represent any quantum system accurately so long as the internal bond dimensions D are made sufficiently large [8]. The MPS is particularly suitable for one-dimensional systems with local interactions. To represent all of the states available in the Hilbert space of the system, it would be necessary to have an exponentially large D. Fortunately, for one-dimensional gapped Hamiltonians (when there is a finite, constant energy separation between ground and excited states), MPS can efficiently represent the ground state of the system with a finitely sized D, which is only polynomial in the system size [8]. This is a significant advantage for representation capability. In addition to this, consider the density matrix ρ of a QMB system, i.e., the matrix which contains the probabilities p_i of obtaining each global quantum state Φ_i. For a pure state, it is explicitly calculated as

$$\rho = \sum_i p_i \left|\Phi_i\right\rangle \left\langle\Phi_i\right|. \tag{1.20}$$

If this system is partitioned into two subsystems, L and $\neg L$, then the reduced density matrix ρ_L is obtained by taking the partial trace of ρ over $\neg L$,

$$\rho_L = \sum_{\phi_j \in \neg L} \left\langle\phi_j\right| \rho \left|\phi_j\right\rangle. \tag{1.21}$$

From this, the entanglement entropy, $S(L)$, can be calculated as

$$S(L) = -\mathrm{Tr}(\rho_L \log \rho_L). \tag{1.22}$$

This quantity has particular significance in that it provides a means of quantifying the amount of state information which is shared between complementary blocks in the Hilbert space of a given system. For blocks which are independent, $S(L)$ will go to zero. For MPS, the scaling of the entanglement entropy follows the ”area law“ (i.e., it is bounded by a constant) [8].

Continuing with the properties of MPS, another relevant property is that correlation functions between sites on MPS with finite bond dimensions decay exponentially with their separation distance. While this property prevents MPS from efficiently representing critical systems (where correlations diverge and therefore bond dimensions would become exponentially large), it does make them suitable for approximating ground states of gapped Hamiltonians.

One property of particular note is that the scalar product between two MPS having N sites can be calculated exactly in polynomial time [8]. This means that expectation values of operators acting on MPS can be efficiently and exactly calculated. Expectation values for MPS (and other TNs) are calculated by inserting the operator $\mathscr{O}$ between an MPS and its Hermitian conjugate or adjoint. Schematically, this is shown in Fig. 1.4. It is worth noting that the operator is defined only over the support of the vector space it operates on, and therefore, all that is necessary for computation is the reduced density matrix of the MPS over that support. Having introduced the important features of MPS, we now look at the properties of the higher-dimensional TN generalization, the Projected Entangled Pair States.

Projected Entangled Pair States

Projected Entangled Pair States (PEPS) are TNs that generalize the MPS to higher dimensional geometries [8]. Two-dimensional PEPS with open boundary conditions are shown in Fig. 1.5. Just as in the case of MPS, two-dimensional PEPS have a few interesting properties worth considering for the analysis of QMB systems.

The first few properties that PEPS possess are reminiscent of MPS. PEPS can represent any quantum system so long as the internal bond dimension D is allowed to increase to any value, though of course, just as with MPS, this bond dimension would have to be exponentially large to represent any arbitrary state in the Hilbert space. Also, for local gapped Hamiltonians, it is conjectured (and

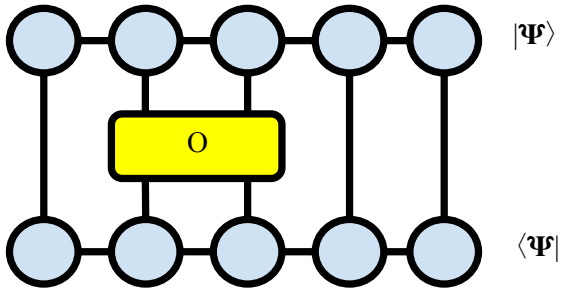


Figure 1.4: An example tensor diagram for the calculation of the expectation value $\langle O \rangle$. The result is computed by contracting all connected bonds. Here, O is a two-body operator.

empirically verified) that the entanglement entropy for ground and low-energy PEPS obeys the two-dimensional area law [8].

A notable difference arises between PEPS and MPS representations when considering two-point corelation functions. Namely, in the case of PEPS, correlation functions between sites with finite bond dimensions do not need to decay exponentially. In fact, PEPS can efficiently capture power-law decaying correlations, which is a characteristic found in critical systems (i.e., where the correlation length becomes infinite) [8, 9]. This allows PEPS to be useful in approximating critical systems as well as gapped systems.

There are two disadvantages to PEPS as compared to MPS. The first is that that the exact computation of the scalar product between two PEPS having N sites is #P-hard [8]. This complexity class essentially means that the computation involves an exponential number of steps. Therefore,

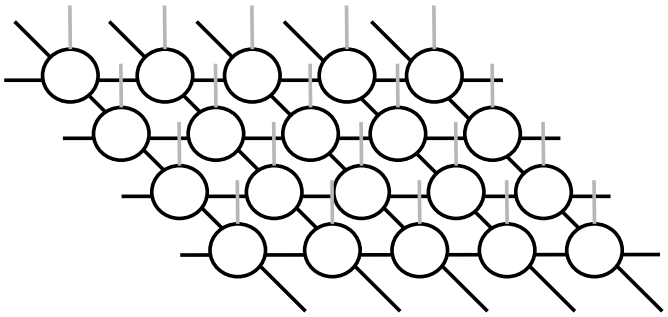


Figure 1.5: A tensor diagram for a two-dimensional, twenty-site, rectangular PEPS with open boundary conditions. Black lines indicate internal bonds (indices), while grey lines indicate physical bonds (indices).

calculating expectation values and correlation functions for PEPS is computationally inefficient. The second disadvantage is that there is no decomposition into a canonical form for PEPS. This means that ground state approximations cannot be achieved by successive Schmidt decompositions. This limitation is overcome with either the use of iterative projection algorithms [10] or variational methods [11].

While we have only covered the basics of TNs, it is enough to demonstrate their use in QMB physics. By utilizing system interactions for the creation of specific TN geometries, the TN representation is shown to be an adaptive structure which naturally captures entanglement properties, being useful for the determination of correlations, expectation values, and distributions.

CHAPTER 2: MACHINE LEARNING PRINCIPLES

Machine learning (ML) is an area of study in computer science that focuses on developing numerical architectures and algorithms which "teach" a machine how to do a specific task. Common applications of ML include image identification, speech recognition, classification, and text translation [12]. Though the range of applications is broad, the mechanisms behind each task are quite similar. In each case, one simply seeks to have the machine "learn" approximations for functions, distributions, or correlations from a given set of input data. Here, we review the relevant topics pertaining to the use of neural networks (NN), which are ubiquitous models in ML.

Random Variables, Classification, and Optimization

Before introducing NNs, it is beneficial to highlight some of the fundamental properties and basic techniques used in ML. The first and perhaps most important property is that input data received by the ML model is general. It is given by a random variable X, which can take on any number of values [12]. If the random variable is discrete (taking on a finite number of values), then the probability of obtaining a specific value x is given by the *probability mass function* $Pr(x)$. If the random variable is continuous (taking on an infinite number of values), then the probability of obtaining a specific value x is given by the *probability distribution function* $p(x)$. $Pr(x)$ and $p(x)$ are both non-negative and sum or integrate to 1, respectively [12].

Most data involves more than a single set X. When given two random sets of variables X and Y, it is useful to introduce the joint probability $p(x,y)$, which measures the probability of obtaining the

measurements $x \in X$ and $y \in Y$ simultaneously, and the conditional probability

$$p(x|y) = \frac{p(x,y)}{p(y)} \tag{2.1}$$

which measures the probability of obtaining the value $x \in X$, given a previous measurement of the value $y \in Y$ [12]. Eq. 2.1 can be brought to a more practical form using Bayes Rule

$$p(x|y) = \frac{p(y|x)\ p(x)}{p(y)} \tag{2.2}$$

Expressed in this way, the conditional probability in Eq. 2.1 can be determined by measuring the reverse condition [12]. As a simple demonstration of ML in practice, if one seeks to approximate Eq. 2.2 for a measurement x containing sub-categories $w \in x$ which are treated as independent (i.e., $p(w^i, w^j) = p(w^i)p(w^j)$), then it is possible to determine the conditional probability as

$$p(x|y) = \prod_j p(w^j|y). \tag{2.3}$$

Here the conditional probability of obtaining x has been reduced to the product of conditionals on w, presumably which are easier to measure. The machine has therefore "learned" a distribution for x based upon input measurements of w. This simple ML method is known as *naive Bayes* [12].

For more complicated data, where variables may not be statistically independent, it is often useful to define a distance measure between measurements on X. Consider an initial example set of data X' having a corresponding label variable Y'. We will define *classification* as the task of labelling measurements x. In comparison to the previous ML method, the classification of a measurement x with a label y can be thought of as an approximation of the conditional probability $p(y|x)$. One particularly convenient way of accomplishing this task is to utilize the predetermined distance measure between measurements x. Unlabelled variables are assigned labels by considering their

k nearest neighbors. This assignment is accomplished by defining the k nearest neighbors of a coordinate x, and labeling x with the majority label belonging to these neighbors. The success of this classification method, termed *k nearest neighbors* [12], is obviously dependent on the validity of the distance measure selected. We say that in this case, the utilization of the distance measure for the determination of the nearest neighbor provided a *classification rule* for the model.

If a classification rule cannot be necessarily determined *a priori* (as it was for the k-nearest-neighbor algorithm), then approximating a suitable classification rule becomes the primary goal of machine learning. The first step in accomplishing this is done by introducing the concept of the *perceptron* [13]. The perceptron is a model for use in classification. It is defined by a weighting vector, W, which maps input data X to labels Y at sequential time intervals t as

$$Y_t = f(W \cdot X_t). \tag{2.4}$$

Here, f can be any linear function of X. For each time interval, W is updated until f properly labels all input X_t. This update procedure is called the *perceptron rule* and is given by

$$W_{t+1} = W_t + Y_t X_t. \tag{2.5}$$

The perceptron and its corresponding rule introduce the idea of iteratively updating weights on input data sets until convergence is achieved [13]. This concept provides the foundation for the introduction of NNs. In what follows, we will define NNs and explore their applications.

Neural Networks

Neural Networks (NNs) are abstract models of computation taken from an understanding of the neural processing structure of the brain. The fundamental component of neural networks in bio-

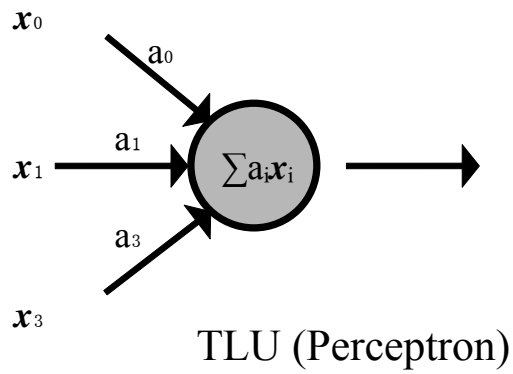


Figure 2.1: A graphical representation of a TLU (perceptron).

logical systems is the neuron, and consequently the most basic fundamental component of a NN is the threshold logic unit (TLU) [13]. The TLU is defined as a structure having n weighted binary inputs $\{a_i x_i | x_i = \{0,1\} a_i \in [0,1]\}$ which determine its activation f by a simple summation

$$f = 1 \;\text{ if }\; \gamma = \sum_{i=0}^{n} a_i x_i > \gamma_*; \;\text{ else }\; f = 0, \tag{2.6}$$

where γ_* is some threshold value [13]. A graphical model of the TLU is given in Fig. 2.1. In Eq. 2.6, the activation is a step function. While useful, an improvement to this can be accomplished by replacing the step function with the sigmoid logistic function,

$$f = \frac{1}{1 + e^{-(\gamma - \gamma_*)/\rho}}. \tag{2.7}$$

Here ρ is a parameter that adjusts how smoothly f goes from 0 to 1 is. Whatever the activation, the TLU can be updated following a slightly adjusted perceptron rule as

$$W_{t+1} = W_t + \alpha\,(Y' - Y_t)\,X_t. \tag{2.8}$$

Here, W_t is a vector of all the weights contributing to the activation, and α is the *learning rate*, a parameter which controls the size of each update step. Generically, the difference between the target Y' and output Y_t can be replaced by a differential $\frac{dC(W)}{dW}$, where $C(W)$ is a cost function parameterized by the weights of the NN [13]. With this generalization, the learning procedure has been transformed into a minimization problem (the goal being the minimization of $C(W)$).

With all of these pieces defined, we have formed the basis for the construction of NNs. All NNs use TLUs (we will generically refer to them as perceptrons from now on) with various kinds of activation functions, and follow iterative update ("learning") procedures to optimize their weights until convergence to a desired output is achieved [13]. As a note, the learning procedure in Eq. 2.8 is defined with access to a target output value y'. When the learning procedure has access to the target output, it is called *supervised learning* [13]. If there is no knowledge of the target output, then it is called *unsupervised learning* [13]. In the following, we introduce various common NN architectures.

Multi-Layer Perceptrons

The Multi-Layer Perceptron (MLP) is a feed-forward NN, meaning that the graphical network structure is directed from the input to the output. This model is a ubiquitous tool for classification, particularly for classification of linearly separable variables [14]. The model itself consists of sequential layers of perceptrons which are inter-connected between layers, but have no connections

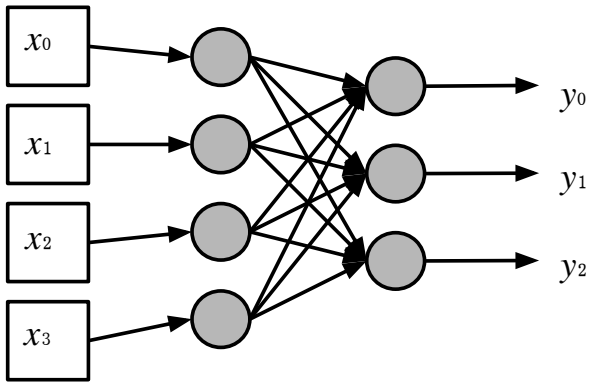


Figure 2.2: A graphical representation of an MLP consisting of an input layer, and one fully connected layer, followed by the output

within a layer. This is shown in Fig. 2.2. Under this architecture, the output of one layer becomes the input of the next, until the final layer is the output of the model. If there is more than one layer of perceptrons present in the model, then it is said to be a *deep neural network* (DNN) [14].

Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are NNs which take input data X and apply pre-processing filters to it before sending the data to an MLP [14]. There are two typical pre-processing layers present in a CNN. The first is called the *convolution layer* and the second is called the *pooling* layer.

The convolution layer is a filter applied to each input data set [14]. For a one-dimensional input set

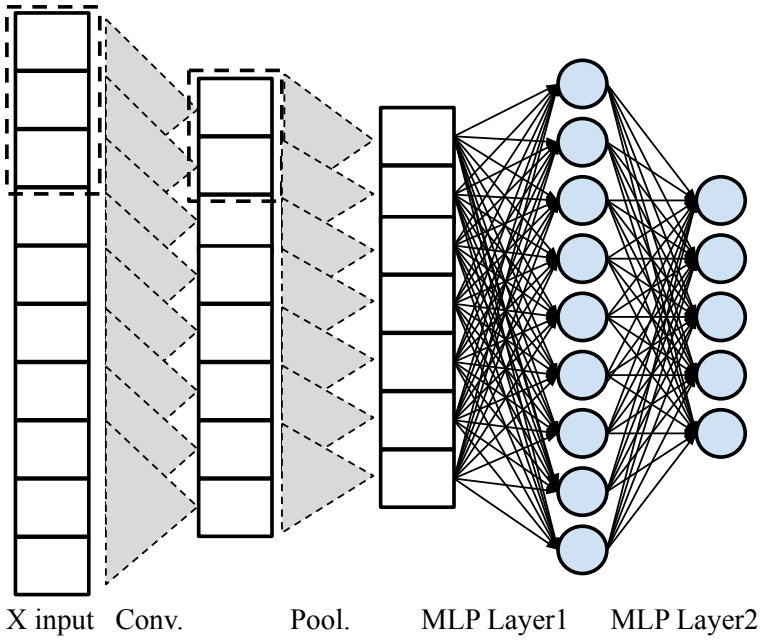


Figure 2.3: A graphical representation of a CNN taking in one-dimensional input, applying a convolution filter of size three, and a pooling filter of size two. The output of these filters is passed into a two-layer MLP.

X having N, and weighting function h, defined on a vector of size M, the filter applies a convolution operation,

$$(X \star h)[n] = \sum_{m=0}^{M} X[n-m]\, h[m]. \tag{2.9}$$

This operation operation produces an output vector of convoluted data which is then used as input for the next layer of the network. This output can be of equal, greater, or smaller size than the original input depending on the stride (i.e., step length of the convolution operation over the input) and padding (i.e., number of zero elements added as buffers to the boundaries of the input) [14].

The pooling layer, which follows the convolution, is a filter which reduces the dimension of the input [14]. This is accomplished by defining a sliding window of size m over the input vector X, which pools the data together into a new vector Y. The two most common pooling operations are

the Max Pooling and the Average Pooling. Max Pooling is given by

$$Y[n] = \max(X[n*m], X[n*m+1], ..., X[n*m+m-1]). \tag{2.10}$$

Average Pooling is given by

$$Y[n] = \frac{1}{M}\sum_{i=0}^{m} X[i]. \tag{2.11}$$

After the application of these two filter, the input is either passed through more layers of convolution and pooling, or sent to an MLP, where the training procedure is applied. Figure 2.3 shows a schematic of the CNN for a one-dimensional input.

Restricted Boltzmann Machines

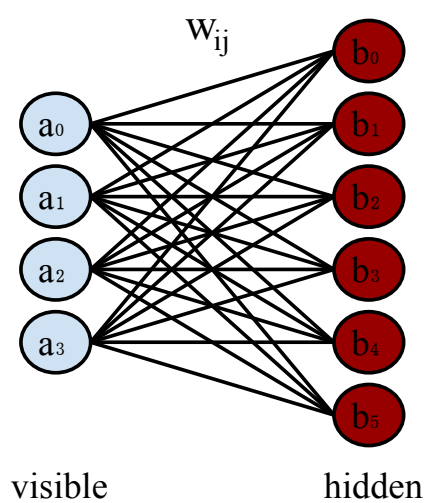


Figure 2.4: A graphical representation of an RBM having four visible units with biases a_i and six hidden units with biases b_i.

The Restricted Boltzmann Machine (RBM) is a NN which differs from the MLP and CNN in that it does not have a directed graphical structure. Rather than being an input-output model, the RBM is a generative model used to represent probability distributions [15]. The RBM is defined by two layers of binary neural units: the visible layer, v, and the hidden layer, h. The joint configuration of these two layers is used to define an energy term

$$E(v,h) = -\sum_{i\in\{v\}} a_i v_i - \sum_{j\in\{h\}} b_j h_j - \sum_{i,j} v_i w_{ij} h_j, \tag{2.12}$$

where a_i and b_i are visible and hidden layer biases and w_{ij} are inter-layer weights [15]. Using this energy, a probability distribution, $p(v,h)$, is given by the Boltzmann distribution

$$p(v,h) = \frac{1}{Z}\, e^{-E(v,h)} = \frac{e^{-E(v,h)}}{\sum_{v,h} e^{-E(v,h)}}. \tag{2.13}$$

To find the specific probability of an input configuration v, a summation is taken over the hidden variables

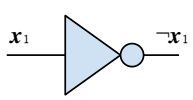
$$p(v) = \frac{1}{Z} \sum_{h} e^{-E(v,h)}. \tag{2.14}$$

Training of this model is done by minimizing the energy in Eq. 2.12 with respect to the weights and biases, so that the RBM distribution converges to the actual distribution [15]. A schematic representation of an RBM having four visible neurons and six hidden neurons is given in Fig. 2.4.

CHAPTER 3: FUNDAMENTALS OF QUANTUM COMPUTATION

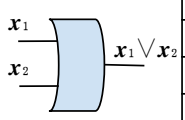
Quantum computation is a model of controlled manipulation of information contained within the state of a quantum system [16]. This is an extension of classical computational models, which model manipulations of information contained within binary strings. Both computational models include forms of information storage, transformation, and measurement. In what follows, we introduce the fundamental concepts and structures used in both classical and quantum computational circuit models, developing the tools necessary to appropriately compare each to the other.

Classical Circuits



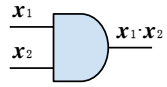
x_1	$\neg x_1$
0	1
1	0

(a)



x_1	x_2	$x_1 \vee x_2$
0	0	0
0	1	1
1	0	1
1	1	1

(b)



x_1	x_2	$x_1 \cdot x_2$
0	0	0
0	1	0
1	0	0
1	1	1

(c)

Figure 3.1: Circuit diagrams and truth tables for the NOT gate (a), the OR gate (b), and the AND gate (c). Input to output is read from left to right.

Classical computation is a vast area of study focused on information control and manipulation. Computations can be modelled in a number of ways, including finite-state machines [17] and Turing machines [18]. However, one of the most common models used is the logic circuit model.

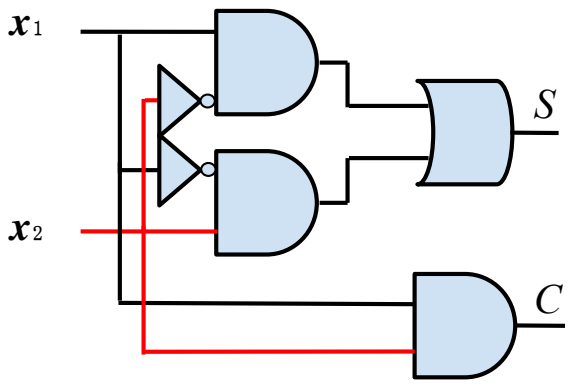
A classical logic circuit is defined by a string of n binary variables x_i, called *bits*, which are stored in a *register*, X. This register can take on any of the 2^n configurations that are formed by the

set $\{x_i\}$. The current state of the register is simply represented by the concatenation of each bit x_i as $X = x_0 x_1 x_2 \ldots x_n$. This state can be manipulated by a function f, which operates on each of the bits in a controlled fashion. Directly encoding large transformations over many bits can be cumbersome, but large operations can be decomposed into sequences of smaller operations over sub-strings of the register. These smaller operations are called *gates* [18]. Typically, these operations act on either 1 or 2 bits performing basic operations. As a particularly important note, if a set of gates being applied to a register has the capacity to model any function f, then that gate set is said to be *universal* [18].

A common universal gate set is the combination of AND, OR, and NOT gates. The AND and OR gates operate on 2-bit states $x_1 x_2$, while the NOT gate transforms a single bit x_1. The transformations for each of these gates is shown in Fig. 3.1a–c. As an example, this gate set can be used to construct a half-adder circuit, which is an instructive fundamental arithmetic transformation which calculates the sum S and carry-over value C of two bits x_1 and x_2. This operation is shown in Fig. 3.2.

A final set of gates which are worth mentioning are those which are said to be *reversible*. A reversible gate is one in which there is a one-to-one mapping between the inputs and outputs. As opposed to the previously defined gates, these gates have just as many output bits as input bits. Two particularly important reversible gates are the controlled-NOT (CNOT) gate and the Toffoli gate. The CNOT operates on two bits, with one bit being defined as the control and the other defined as the target. The CNOT gate is basically equivalent to the classical XOR gate, with the addition of a second output line. The Toffoli is an extension of the CNOT to three bits, where two bits act as the control and the third bit is the target. Interestingly, the Toffoli gate alone is universal. The transformations given by each of these gates is shown in Fig. 3.3. Control bits are those where a black dot is placed on the circuit line.

Having defined these gates for this model of computation, classifications and comparisons of algorithmic computational complexity can easily be done by counting the number of gates involved in a computation. This is an important feature when making comparisons to quantum algorithms computed on quantum circuits.

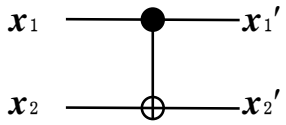


x_1	x_2	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Figure 3.2: Circuit diagram and truth table for a half-adder circuit acting on two bits x_1 and x_2. The summation of the two inputs is contained in S, while the carry value is contained in C.

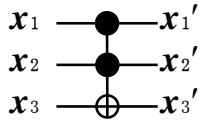
Quantum Circuits

Quantum computation primarily differs from classical computation in that the information being manipulated is contained within a system which is fundamentally quantum (i.e., the scale of the system is such that quantum mechanical effects cannot be ignored or in fact are being explored).



x_1	x_2	x_1'	x_2'
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

(a)



x_1	x_2	x_3	x_1'	x_2'	x_2'
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	1
1	1	1	1	1	0

(b)

Figure 3.3: Circuit diagrams and truth tables for the CNOT gate (a), and the Toffoli gate (b). Input to output is read from left to right.

In order for full computational ability to be realized in these systems, a few properties must be present: 1) the particles used for computation must have two clearly distinguishable states and be scalable to large numbers of particles (this allows the circuit to be treated as bits, or the quantum analog *qubits*). 2) the system must be able to be initialized and sufficiently isolated from the environment (this mitigates noise and decoherence effects which are primary causes for information loss). 3) Ancillary qubits must often be included in computational algorithms (this inclusion prevents direct measurement of the QMB state, which would cause quantum state collapse and destroy any information in the circuit). 4) The circuit must only be transformed by a universal set of unitary operators (this property is inherent in the evolution of any quantum system) [19].

If these properties are all present within a given system, then quantum computation can successfully be carried out. These properties can be achieved for a variety of systems including superconducting Josephson junctions [20], ion traps [21, 22], quantum optical systems [23, 24], nuclear magnetic resonance (NMR) systems [25, 26], and quantum dots [27, 28]. Whatever the system,

modeling the computational aspects of the system is done using the quantum circuit model.

Analogous to the classical logic circuit model, the quantum circuit consists of a few key components. First, there is the quantum register $|\Psi_N\rangle$ which houses N qubits $|\phi_i\rangle$. Using braket notation, these qubits can be in the state $|0\rangle$, $|1\rangle$, or the superposition state

$$|\phi_i\rangle = \beta_i |1\rangle + \alpha_i |0\rangle , \tag{3.1}$$

with α_i and β_i being complex probability amplitudes. The state of the register is given by

$$|\Psi_N\rangle = \sum_{\phi_1,\phi_2\ldots\phi_N} C_{\phi_1,\phi_2,\ldots,\phi_N} |\phi_1, \phi_2, \ldots, \phi_N\rangle . \tag{3.2}$$

where $C_{\phi_1,\phi_2,\ldots,\phi_N}$ is the joint-probability amplitude for a configuration $\phi_1, \phi_2, \ldots, \phi_N$. Also included in the register are ancillary qubits which help facilitate computations efficiently.

Second, there are operators which are applied to this register through the sequential application of gates. This is analogous to use of gate sets in classical computation. However, the gates used in quantum circuits have constraints on them that are not present classically. Primarily, these gates are required to be unitary, and therefore also reversible [16]. As such, not every gate definition previously introduced is applicable to the quantum circuit model. An example of gates which do satisfy this condition are the Pauli gates X, Y, and Z. These are single qubit operators operating on qubits $|\phi_i\rangle$ as

$$X |\phi_i\rangle = \alpha_i |1\rangle + \beta_i |0\rangle , \tag{3.3}$$

$$Y |\phi_i\rangle = -i\beta_i |0\rangle + i\alpha_i |1\rangle , \tag{3.4}$$

and

$$Z\left|\phi_i\right\rangle = \alpha_i\left|0\right\rangle - \beta_i\left|1\right\rangle. \tag{3.5}$$

While these are useful operators, they cannot perform universal computation. For universal computation, it is necessary to introduce 2-qubit operators. We introduce a common universal gate set called the Clifford + T set. This gate set is composed of the CNOT, Hadamard, S, and T gates [29]. The Hadamard (H), S, and T gates are single-qubit operators defined as follows:

$$H\left|\phi_i\right\rangle = (\alpha_i + \beta_i)\left|0\right\rangle + (\alpha_i - \beta_i)\left|1\right\rangle, \tag{3.6}$$

$$S\left|\phi_i\right\rangle = \alpha_i\left|0\right\rangle + i\beta_i\left|1\right\rangle, \tag{3.7}$$

and

$$T\left|\phi_i\right\rangle = \alpha_i\left|0\right\rangle + e^{\frac{i\pi}{4}}\beta_i\left|1\right\rangle. \tag{3.8}$$

The CNOT gate is a two-qubit operator which transforms the state $\left|\phi_1,\phi_2\right\rangle$. Before giving its definition, note that the 2-qubit state is the tensor product of $\left|\phi_1\right\rangle$ and $\left|\phi_2\right\rangle$, given as

$$\left|\phi_1\phi_2\right\rangle = \alpha_1\alpha_2\left|00\right\rangle + \alpha_1\beta_2\left|01\right\rangle + \beta_1\alpha_2\left|10\right\rangle + \beta_1\beta_2\left|11\right\rangle. \tag{3.9}$$

The operation of CNOT on this state is

$$\text{CNOT}\left|\phi_1\phi_2\right\rangle = \alpha_1\alpha_2\left|00\right\rangle + \alpha_1\beta_2\left|01\right\rangle + \beta_1\beta_2\left|10\right\rangle + \beta_1\alpha_2\left|11\right\rangle. \tag{3.10}$$

The final component necessary to complete the quantum circuit model is a measurement protocol. Measurement for quantum systems is not handled in the same way as classical. For quantum circuits, measurements must be made by applying projection operators $P^{(i)}$ to the qubits in the

register. For many quantum algorithms, this is done over only a subset of the computational qubits, or over the ancillary qubits in the system, where the solution is encoded into either the $|0\rangle$ or the $|1\rangle$ state of the qubits [19]. The construction of quantum circuits in this manner is given for four qubits in Fig. 3.4. With all of these components defined for quantum circuits, there is a clear framework for the comparison of quantum algorithms to classical.

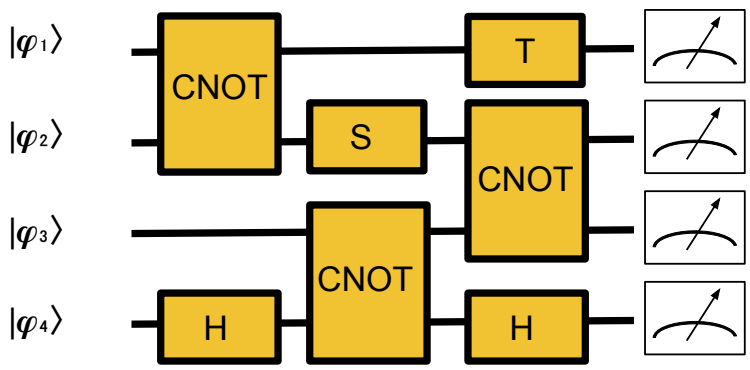


Figure 3.4: A quantum circuit diagram with four qubits ϕ_i initiliazed on the left, transformed by a random sequence of gates selected from the universal gate set $\{CNOT, H, S, T\}$, and measured on the right by projectors.

Outlook

Having reviewed the necessary disciplines, we are now in a place to discuss the projects contained within this dissertation. Each project discussed is provided with the following: 1) An introduction and motivation for the problem being modelled, 2) A detailed explanation of the methodologies

and algorithms used in the model, 3) The results of the work performed, and 4) The conclusions taken from these results.

CHAPTER 4: TN APPLICATIONS: QMB USING TN STATES ON THE CLOUD

Introduction and Motivation

The work contained within this chapter was done in collaboration with Dr. Dan Marinescu and is currently under review for publication in Computer Physics Communications.

As an initial optimization of current tensor network (TN) methods, we first investigate the feasibility of moving tensor operations to the cloud computing infrastructure. This infrastructure, shown in Fig. 4.1, is designed to perform optimally for Big Data, online transaction processing, and data streaming applications. Such applications exploit data-level, task-level, and thread-level parallelism. Warehouse scale computers (WSCs), which serve as the backbone of the cloud infrastructure, host tens to hundreds of thousands processors communicating through networks with sufficient bandwidth and with a relatively high latency. This proves advantageous for enterprise cloud applications, where procedures such as Remote Procedure Calls (RPCs), serialization, deserialization, and compression of buffers use only 22–27% of CPU cycles [30, 31]. However, when considering applications in science and engineering, the communication latency can produce a significant effect.

Specifically considering QMB applications, simulations typically exhibit fine-grained parallelism, deploying many parallel threads which communicate frequently with each other, and which use barrier synchronization to transit from one stage of computation to the next. While there are methods such as the parallel Monte Carlo which have been optimized for such communication costs [32], many other methods simply cannot avoid this level of communication. This is particularly the case for parallel TN algorithms, such as the parallel-DMRG [33], where partial compu-

tations must be synchronized frequently. Because of the communication latency of current cloud services, QMB applications are preferably performed on supercomputers with fast interconnection networks such as Mirinet, Infiniband, or some custom designed network. For instance, a group from ETH Zurich was able to perform a 45-qubit simulation using a supercomputer at the Lawrence Berkeley National Laboratory [34].

However, recent advances in Cloud Service Provider (CSP) technologies have narrowed the performance gap vis-a-vis supercomputers. Clusters of compute nodes with faster interconnects are now offered. Instances with physical memory on the order of hundreds of GiB are now provided. Faster processors and coprocessors are even in development for new instance types. For example, the Amazon Elastic Compute Cluster (EC2) has provided access to Graphics Processing Unit (GPU) instance types, which are optimal for linear algebra operations, and Google has pioneered Domain Specific Architectures (DSAs) with the creation and provision of Tensor Processing Units (TPUs), which are optimal for the many small matrix-matrix operations needed in deep learning applications. While these advances are significant, there is still a challenge in managing high-order tensor contractions in QMB applications. This is because both the number of operations and the memory footprint involved in the computation grow exponentially with the system size.

While several TN algorithms have been formulated to implement computer simulations of these systems [35, 11, 36, 37, 38, 39, 40], their success is often found in the one dimensional case, where simulations can scale up to large system sizes because exact TN contractions require only a polynomial effort. For systems of higher dimension, this is not the case. Such simulations have been limited to systems of modest size and particular geometries due to the overwhelming number of computational resources necessary to perform TN contractions in more than one dimension [41, 42]. Approximate methods to overcome these limitations have been proposed, for instance, by simplifying the tensor environment and thus, avoiding a full contraction [43, 44, 45, 46]. However, the primary focus of the literature has been on infinite, translation-invariant systems and

no particular attention has been paid to adapt the methodology to distributed computing. In this work, we investigate two-dimensional QMB spin systems and explore the limitations of a cloud computing environment using a heuristic for parallel TN contractions without approximations. We do so by selecting the instances from the AWS EC2 which provide the largest RAM storage. Then we allow tensor contractions to be carried out concurrently while reducing the amount of communication between parallel threads. We facilitate this last part by avoiding distributing a single tensor across multiple threads, choosing instead to distribute groups of tensors according to the geometry of the system under consideration.

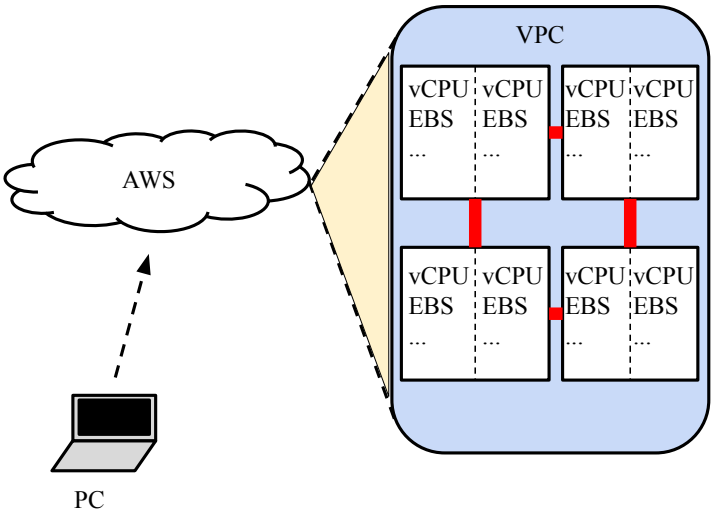


Figure 4.1: A schematic for the AWS cloud. Depicted is a local computer which connects and submits computational scripts to a personally assigned virtual private cloud (VPC), consisting of inter-connected instances (white blocks) each having their own set of virtual processors (vCPUs) and access to an elastic block storage (EBS) system.

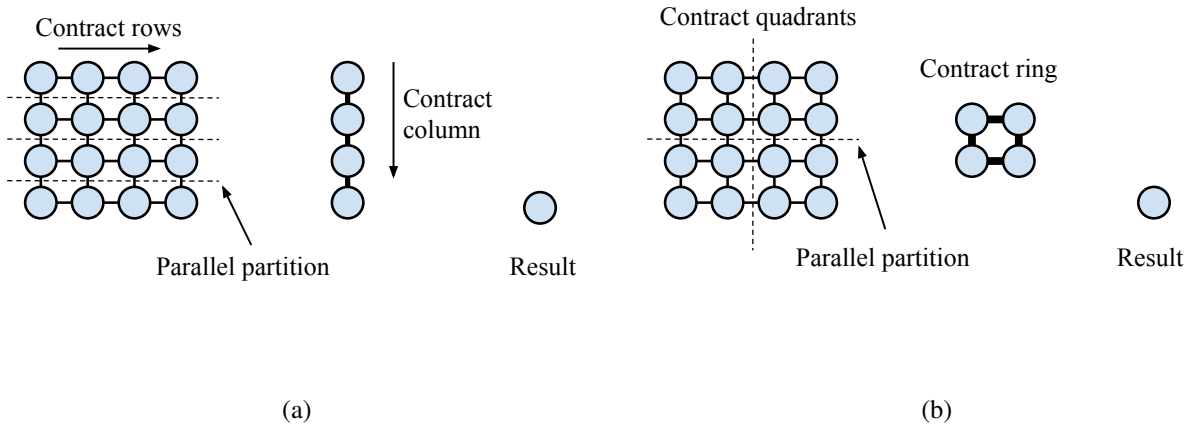


Figure 4.2: Schematics for two square TN contraction orderings and parallel partitionings. In a) the row contraction and parallel partition are shown, resulting in a final column of tensors each with bond dimension D^L. In b) the quadrant contraction and parallel partition are shown, resulting in a ring of tensors with bond dimension $D^{L/2}$.

Methods

Because our method distributes groups of tensors according to the geometry of the TN, our method is model dependent (i.e., specific to the QMB model under consideration). In this study, we specifically consider two-dimensional PEPS, which are planar, rectangular TN quantum states. We focus on the computation of expectation values over these networks, such that physical indices are always contracted first, reducing the computation to a summation over all the internal indices in a planar TN.

To recognize the difficulty in this computation, consider a square lattice of tensors with L tensors along each dimension. The full and exact contraction of a TN is practically accomplished by an iterative sequence of tensor pair contractions. The number of tensor elements contained in the network, which provides an accurate estimate of computational memory requirements, evolves as

follows. Consider two tensors, A_1 and A_2, with dimensions $d(A_1)$ and $d(A_2)$. As this tensor pair is contracted into a single tensor B, the dimension $d(B)$ of satisfies

$$d(B) = \frac{d(A_1) * d(A_2)}{d(x)}, \tag{4.1}$$

where x is the set of shared indices between A_1 and A_2. If the product in the numerator is significantly larger than $d(x)$, then the tensor pair contraction significantly increases the memory requirements for the full network contraction. Each contraction of tensor pairs threatens to make the full contraction practically intractable. To mitigate this threat, it is necessary to optimize the order of tensor pair contractions.

Optimizing the contraction order first involves defining a unique tensor pair within the sequence of contractions over the network. For every full TN contraction there exists a *bottleneck contraction*, after which every tensor pair contraction ceases to increase the memory footprint. It is defined as the kth pair contraction, involving the largest tensors generated from a given sequence of n pair contractions $\{c_0, c_1, ..., c_k, ..., c_n\}$. It is our desire to pick a sequence of contractions which minimizes the size of the tensors involved in this bottleneck.

In addition to selecting a contraction ordering which minimizes the bottleneck contraction, we implement a parallelization scheme which takes advantage of the cloud infrastructure by using only a minimal amount of inter-process communication. We call this parallelization scheme TN *geometric partitioning*.

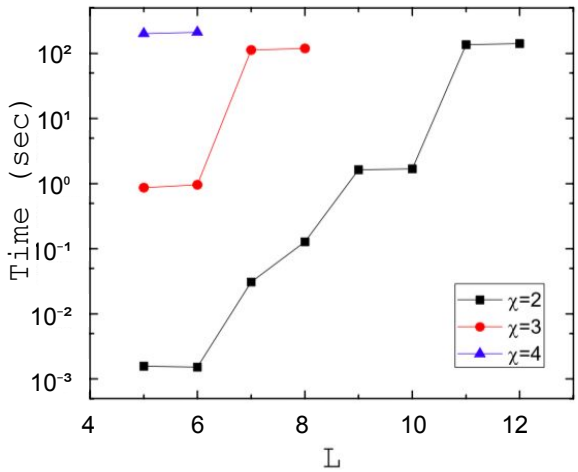


Figure 4.3: Results for the computational time needed to compute the contraction of TN square lattices sized $L = [5, 12]$. Different lines correspond to different values of the fixed maximum bond dimension $\chi = \sqrt{D}$. Memory limitations prevent simulation of larger sized systems for larger sized bond dimensions.

Results

Contraction Ordering

In order to understand the capabilities of the cloud for QMB applications, we considered two extremes for the contraction ordering on a square $L \times L$ TN lattice . The first is termed *row contraction*, and is shown in Fig. 4.2a. For this contraction sequence, if every tensor in the lattice had internal bond dimension χ, then the bottleneck occurs when there is one column of tensors left. The tensors at this point of the contraction each have χ^{2L} elements. The second ordering is termed *quadrant contraction*, and is shown in Fig. 4.2b. In this contraction sequence, the bottleneck occurs with a ring of four tensors each having χ^{L} elements. Other contraction orderings fall

somewhere between these two extremes because they inevitably introduce a tensor having more than χ^L elements along the way, but not necessarily having χ^{2L} elements. In Fig. 4.3, we show the results for the CPU clock time needed to contract square networks with varying sizes L and varying uniform bond dimension χ.

Parallel Partitioning

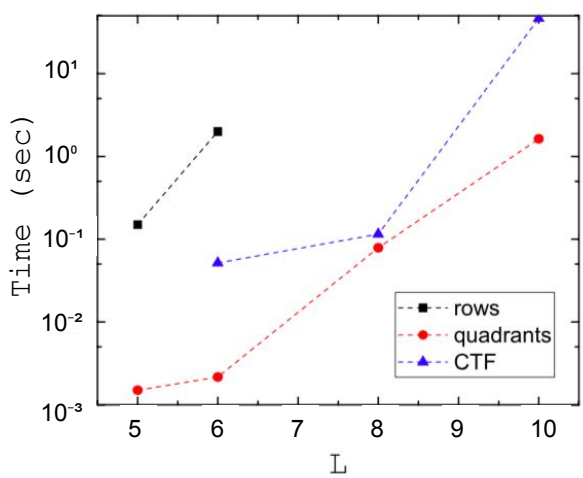


Figure 4.4: Results for the time to complete the contraction of square TN lattices sized $L = [5, 10]$. Differing lines represent different contraction orderings and schemes: row contraction (black), quadrant contraction (red), and CTF contraction (blue). The quadrant contraction shows a more favorable scaling for larger lattice sizes.

Additionally, we studied the effects of partitioning TNs over various processors. Naturally, this helps to reduce the computational time for any contraction order. It allows various subsets of the network to be contracted concurrently. A comparison of sending entire rows of the network to different processes, and sending quadrants of the network to different processes is given in Fig. 4.4. In

addition, both of these partitions are compared with the direct parallelization of individual tensors, a process implemented using the Cyclops Tensor Framework (CTF). This latter parallel scheme is not geometry specific. It is evident from Fig. 4.4 that the quadrant partitioning has a computational advantage over both the row partition and the CTF as the system size grows.

The 2D Transverse Field Ising Model

Finally, having selected the quadrant contraction order and the quadrant parallel partition for generic rectangular TNs, we selected the AWS EC2 x1.32x large instance type (composed of Intel Xeon E78880v3 processors having the largest DRAM capacity at 1952 GB) for application to the QMB simulation of the two-dimensional transverse field Ising model.

The Ising model is used in statistical mechanics to describe magnetic lattice systems with strong anisotropy [47], and also serves as a paradigmatic model for the study of quantum phase transitions [48]. The model consists of discrete binary variables S_i^z that represent magnetic dipole moments of atomic spins that can be in one of two states $(+1)$ or (-1). The spins are arranged in a lattice, allowing each spin to interact with its nearest neighbors (spin-spin interaction). For spatial dimensions larger than one, the model has a finite-temperature phase transition and exhibits critical behavior.

In the presence of a transverse magnetic field, the model yields a zero-temperature quantum phase transition driven by the competition between the spin-spin interaction (favoring the ferromagnetic state when $J > 0$ or the anti-ferromagnetic state if $J < 0$) and the external field (favoring the paramagnetic state). Mathematically, the model is defined by the Hamiltonian

$$\mathscr{H} = -J \sum_{\langle i,j \rangle} \hat{S}_i^z \hat{S}_j^z - \Gamma \sum_i \hat{S}_i^x, \tag{4.2}$$

where J characterizes the spin-spin interaction and Γ characterizes the coupling to the external field. The quantum critical behavior at $\Gamma/J \approx 3$ [49] arises from the non-commutability of the on-site spin operators $\hat{S}_i^z$ and $\hat{S}_i^x$. Finding the ground state $|\Psi_0\rangle$ of this system is an important non-trivial task. If this state is known, then important physical quantities can be determined, such as the energy,

$$E = \frac{\langle\Psi_0|\,\mathscr{H}\,|\Psi_0\rangle}{\langle\Psi_0|\Psi_0\rangle}, \tag{4.3}$$

the transverse magnetization,

$$m_k^x = \frac{\langle\Psi_0|\hat{S}_k^x|\Psi_0\rangle}{\langle\Psi_0|\Psi_0\rangle}, \tag{4.4}$$

or the longitudinal magnetization,

$$m_k^z = \frac{\langle\Psi_0|\hat{S}_k^z|\Psi_0\rangle}{\langle\Psi_0|\Psi_0\rangle}. \tag{4.5}$$

To obtain this ground state, we implement an imaginary time evolution (ITE) algorithm. This algorithm is defined by the iterative application of an incremental ITE operator $\hat{U}_{\delta\tau} = e^{-i\hat{H}(-i\delta\tau)} = e^{-\hat{H}\,\delta\tau}$. This operator is applied to an initial quantum state $|\Psi_{\text{initial}}\rangle$, over m steps, evolving the system in incremental steps $\delta\tau = \tau/m$ through imaginary time until the total time τ is reached. For $\tau \gg \max\{J^{-1}, \Gamma^{-1}\}$, the state vector $|\Psi\rangle = \left[\prod_{i=1}^{m} \hat{U}_{\delta\tau}\right] |\Psi_{\text{initial}}\rangle$ becomes exponentially close to the ground state $|\Psi_0\rangle$, provided that the initial state had a nonzero overlap with the ground state, namely, $\langle\Psi_{\text{initial}}|\Psi_0\rangle \neq 0$. Typically, one chooses as the initial state $|\Psi_{\text{initial}}\rangle$ a product state that is a random superposition of individual spin states. For our purposes, we initialize the TN as a uniform superposition of product states,

$$|\Psi_{\text{initial}}\rangle = \frac{1}{2^{N/2}} \prod_{k=1}^{N} \left[\sum_{\sigma_k=\pm 1} |\sigma_k\rangle \right] \tag{4.6}$$

(i.e., the uniform bond dimension is initially $\chi = 1$).

The ITE was first performed for the Ising model with a transverse field of $\Gamma = 1$ and spin-spin in-

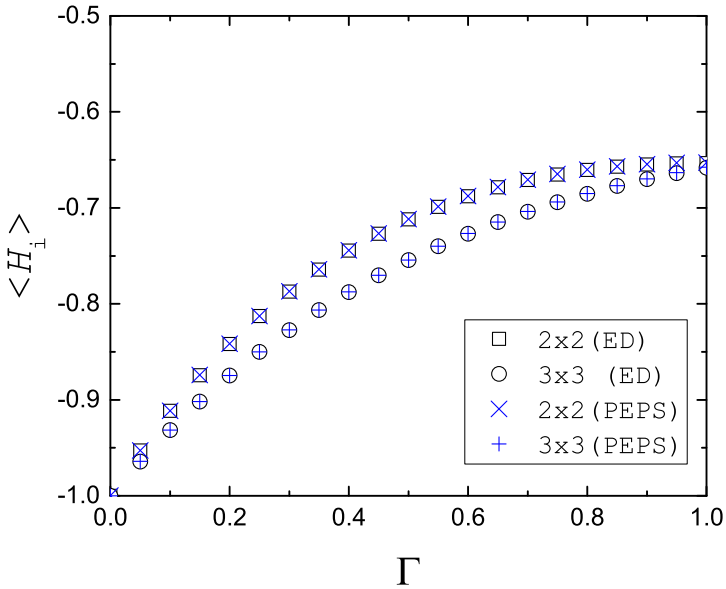
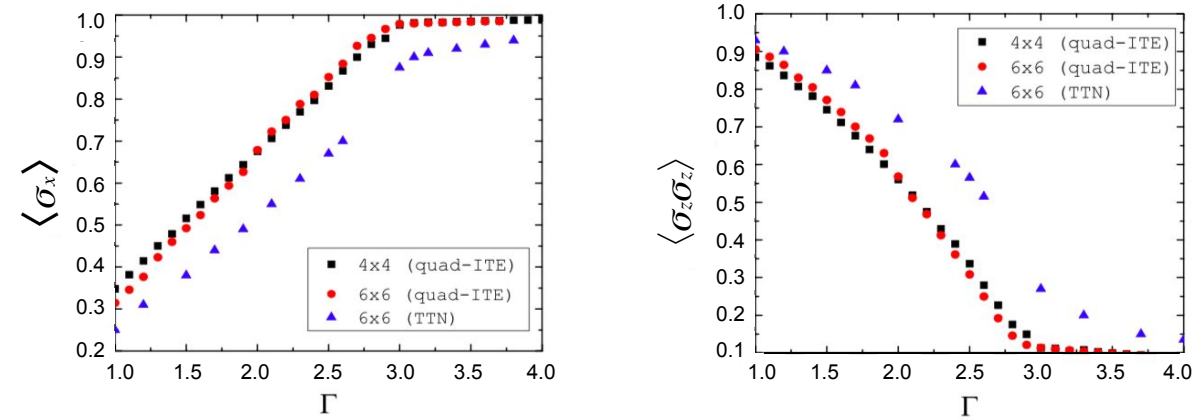


Figure 4.5: Results comparing exact diagonalization calculations to the final energies calculated for the PEPS ITE algorithm applied to the transverse field Ising model. Here lattice sizes are $L = \{2,3\}$, spin coupling $J = 1$, transverse field strength Γ varies from 0 to 1, and time steps are taken as $\delta\tau = 3/100$.

teraction $J = 1$ (far from the critical point) over varying lattice sizes L. For each size, we measured the time for the energy to converge. Imaginary time steps were fixed to $\delta\tau = 3/100$. The singular value cutoff parameter for Schmidt decompositions was fixed to $\varepsilon = 0.01$, where $\lambda_k/\lambda_1 \geq \varepsilon$, to temper the growth of the bond dimensions in the system. The time for completion of the ITE scales exponentially with the system size L, as shown in Fig. 4.5.

Moving towards the critical point (i.e., to regimes of high entanglement), we also performed the ITE for the Ising model with a transverse field of $\Gamma = 3$ and spin-spin interaction $J = 1$ (i.e., near the critical point). The lattice lengths tested were $L = 6$ and $L = 8$ (i.e., 36 and 64 spins, respectively). For $L = 6$ and $L = 8$, each time step δ was set to $\delta\tau = 3/75$ and $\delta\tau = 4/250$, respectively. The ground states were reached with a computational runtime of 12.5 hours and 293.3

hours. Because bond dimensions grew quickly in this regime, a maximal cutoff of $\chi = 4$ was set to allow convergence to be reached without overwhelming the RAM capacity of the processors.



(a) (b)

Figure 4.6: A comparison between the PEPS ITE alogrithm and a TTN algorithm for calculating the ground state expectation values (a) $\langle\sigma_x\rangle$ and (b) $\langle\sigma_{zz}\rangle$ for the two-dimensional Ising model with spin coupling $J = 1$ and transverse field strengths $\Gamma = [0,4]$. Comparisons are made between square lattices sized $L = 6$. Also depicted are the values obtained for lattice size $L = 4$. ITE calculations were done with time step $\delta\tau = 3/75$ and maximal bond dimension $\chi = 4$.

As a final experiment, for transverse field values of Γ in the range $[0,4]$, we compared the convergence of the observables M^x and M^z with previous work established by Tagliacozzo *et al* [50], where the TN environment is accurately approximated by coarse-graining the $L \times L$ lattice to a smaller size in real-space by applying optimized isometries over groups of sites. The results are shown in Figs. 4.6a and 4.6b. Because we forcibly reduced our bond dimensions to $\chi = 4$ and avoided using any tensor environment approximation methods, there is a notable difference between our computations and the those calculated with the TTN. They do, however, qualitatively exhibit the same behavior indicative of a phase transition near $\Gamma = 3$. At the paramagnetic phase ($\Gamma > 3J$), the on-site transverse magnetization $\langle\sigma_x\rangle$ is maximum, while in the ferromagnetic phase ($\Gamma < 3J$) the on-site longitudinal spin-spin local correlator $\langle\sigma_z\sigma_z\rangle$ is maximum.

Conclusions

Profiling the ITE algorithm shows that the computational time is dominated by two in-house procedures, `contract tVtl` and `contract lattice mpi`. The first procedure, which is responsible for contracting two tensors is called 31,347 times and uses 97.18% of the execution time during a 6×6 ITE run. The second procedure, which is a subroutine calling `contract tVtl` repeatedly as it performs the parallel contraction of the TN, is called 3,843 times. Within this subroutine, the time of execution is 1.1 ms/call. This time is mostly spent within `contract tVtl` at 1.07 ms/call. Because of this, we infer that the procedure for contracting two tensors, particularly bottleneck tensors, is indeed the limiting factor in the ITE algorithm, justifying the use of our TN contraction and partition protocol.

Looking at how our TN algorithm performs on the AWS EC2, it is clear that supercomputing clusters being built around low-latency interconnection networks still provide the most suitable means for simulation applications in engineering and physics. However, the cloud infrastructure is far less costly, and we have shown that for certain problems of interest, there are cost-effective alternatives to supercomputing clusters in the AWS EC2 instance types. Specifically, the x1.32xlarge instance types are well suited for the ground state simulation of QMB systems such as the transverse field Ising model. As improvements to the L3 caches of instances are developed and network interconnection latency is decreased in the AWS EC2, further implementations of QMB TN algorithms can be utilized on the cloud.

CHAPTER 5: TN APPLICATIONS: CLASSICAL COMPUTATION WITH TN STATES

Introduction and Motivation

Tensor networks are not only relevant within the context of QMB systems. They also have proven useful as practical tools for data-driven computational tasks [51] and as formal concepts of study in theoretical computer science [52]. This broad range of TN applicability drives us to consider TN algorithms for the solution of generic computer science problems which are known to have a high computational complexity. Specifically, the problem of counting solutions of Boolean satisfiability problems (#kSAT), has been proven to admit no meaningful approximation scheme [53], effectively guaranteeing that no approximate tensor network contraction is likely to be accurate. The #kSAT problem is defined as counting the number of configurations in a string of n Boolean variables $X = \{x_0 x_1 x_2 \ldots x_n\}$ which satisfy the conditions set by m constraints (clauses). Each clause constrains the assignments of k variables in X. We show that by 1) encoding classical solutions into the ground states of reversible circuits embedded onto rectangular tensor networks and 2) compressing the information in the network through iterative Schmidt decompositions over every tensor pair, it is possible to obtain exact solutions for the #2SAT, #3SAT, and #3XORSAT problems. Benchmarking this method reveals that the compression protocol detects the complexity dichotomy between #3XORSAT (in P) and #3SAT (#P-complete), and that there is a sub-exponential scaling of the computational time for our method to reach solution for typical instances, which stands in contrast to the exponential scaling found in many of the generic algorithmic tools used for solving SAT problems [54, 55, 56, 57]. All the work contained in this chapter was done in collaboration with Lei Zhang, Stefanos Kourtis, Claudio Chamon, and Andrei Ruckenstein.

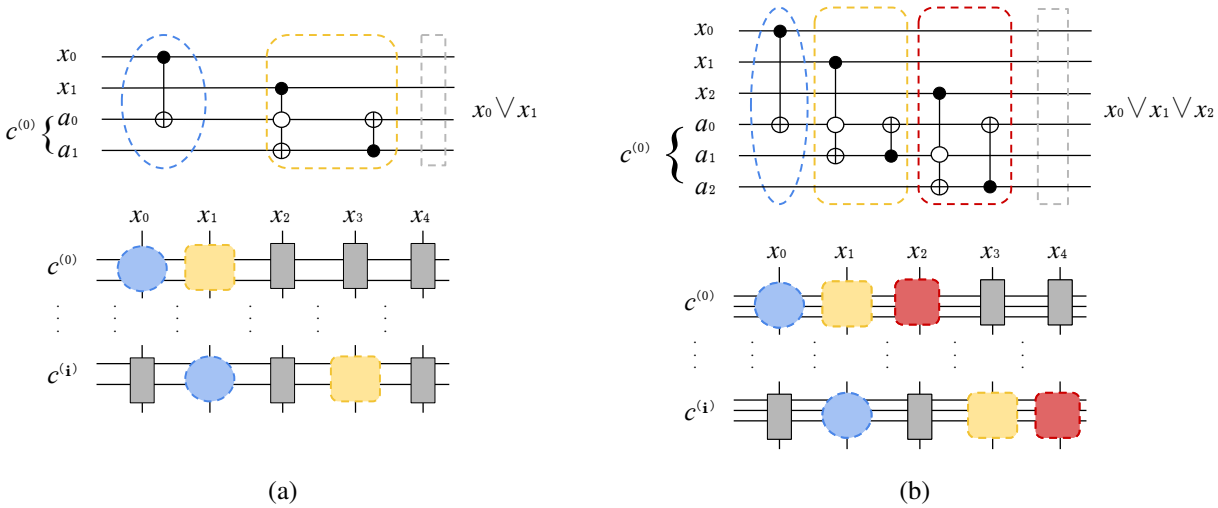


Figure 5.1: A schematic detailing the tensor network embedding of boolean circuits. In (a) a single #2SAT clause is shown above, with gates grouped by dashed lines. The blue group becomes the CIRCLE gate, the yellow group becomes the BLOCK gate, and the grey group is an ID gate used to create uniformity throughout the tensor network. Below, each clause ancillary bus $c^{(i)}$ forms the horizontal bonds of the network, while bits x_i form the vertical bonds. In (b) a single #3SAT clause is shown above, with gates also grouped by dashed lines. The blue group becomes the CIRCLE gate, the group becomes the BLOCK1 gate, the red group becomes the BLOCK2 gate, and again the grey group becomes the ID gate used to create uniformity in the tensor network. Horizontal bonds are given by each ancillary bus $c^{(i)}$ and vertical bonds are given by bits x_i

Boolean Circuit Embedding

Our method begins with representing reversible classical computations as Boolean circuits. A reversible Boolean circuit is defined by a set of reversible gates $\{g_{\sigma_i} : i = 1, 2, .., m\}$ which are applied to an input vector of N Boolean variables, $\boldsymbol{v}$, yielding an output vector $\boldsymbol{v}'$. The reversibility of this gate set enforces a one-to-one mapping of the input to the output. By defining the input to be propagating from the left along N wires, where each wire corresponds to a variable, each

gate can be said to be connected to a subset σ_i of the wires. If we take a universal set of gates, $\{g_{\sigma_i}\}$, and impose fixed boundary conditions on the input variables, then this reversible Boolean circuit model can be used to model any arbitrary computation. For our purposes, we select the gates CNOT and TOFFOLI, which encode 2-bit and 3-bit interactions, respectively, and which form universal gate sets. The CNOT gate transforms the 2-bit state $|x,a\rangle \mapsto |x,x\oplus a\rangle$, while the TOFFOLI gate transforms the 3-bit state $|x,y,a\rangle \mapsto |x,y,x\cdot y\oplus a\rangle$.

Specifically considering a single instance of the 2SAT problem involving N Boolean variables $\{\boldsymbol{x}\} = \{x_1, x_2, \ldots, x_N\}$, we note that the 2SAT problem is formed by the conjunction ($\wedge$) of pairwise disjunctions ($\vee$) of *literals*. Each literal is a variable in $\{\boldsymbol{x}\}$ or its negation. A 2SAT formula is illustrated in Fig. 5.1(a). We encode an elementary representation of a 2SAT clause over variables x_1 and x_2, requiring an ancillary "bus" of two Boolean variables a and b, and two types of gates, CNOT and TOFFOLI, arranged as shown in Fig. 5.1(a). By fixing $a = b = 0$ on the left boundary and $a = 1$ on the right boundary, allowable configurations are constrained to satisfy $x_1 \vee x_2 = 1$. As a note concerning the gate definitions, we sequentially group the gates which comprise a single clause, the first group simply being a CNOT gate, referred to as a CIRCLE gate, and the second group containing a TOFFOLI followed by and a CNOT, referred to as a BLOCK gate.

Under this construction of Boolean circuits, we embed 2SAT instances onto a square lattice, with vertical edges corresponding to variables, horizontal edges corresponding to ancillary buses, and vertices defining local interactions between variables and buses, as shown in Fig. 5.1(a).

This embedding is easily extended to represent 3SAT instances by requiring three variable-bus interactions and including an additional BLOCK gate, as shown in Fig. 5.1(b). The 3XORSAT is even simpler, requiring only 3 CIRCLE gates (i.e., no 3-bit interactions) and an ancillary bus having only one bit.

An immediate observation about the lattice structure is that the order of the variables is incon-

sequential to the determination of the solution. All that matters is that contradictions between variables within a shared clause are removed. By reducing the distance between CIRCLE and BLOCK gates (i.e., reordering the variables), the rate of information exchange between these variables can be increased, improving the likelihood of finding a satisfying configuration. We reorder the variables by implementing an MSRO algorithm [58].

Tensor Network Boolean Circuit

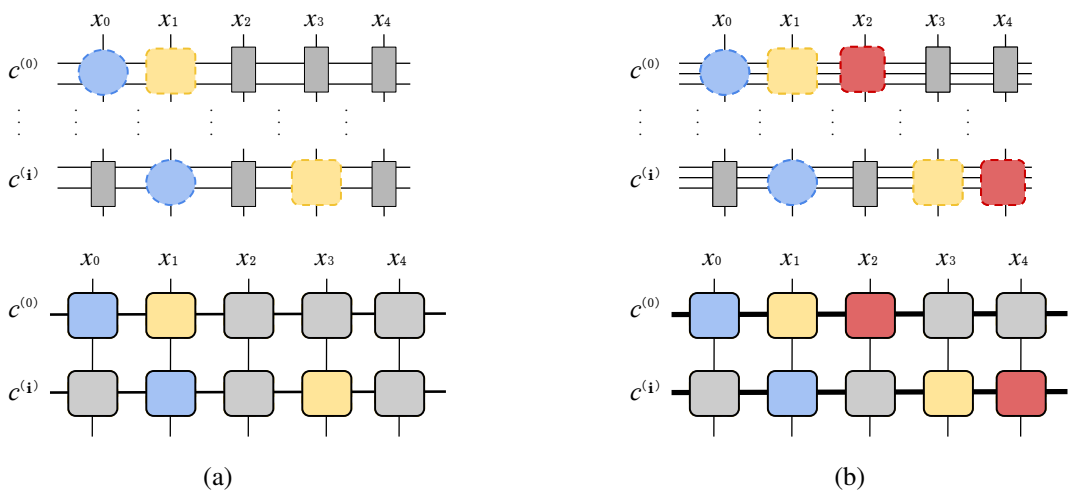


Figure 5.2: A schematic of Boolean circuit gates for (a) the #2SAT and (b) the #3SAT circuits being converted into tensor networks with a uniform distribution of tensors throughout. Horizontal bonds represent bus auxiliary bits and vertical bonds represent bits.

To carry out the TN contraction, the square lattice must be converted into a TN. This is accomplished by representing the truth tables for n-variable functions F_n as matrices $\{M_{jk}^{(F_n)}\}$, with indices j and k tabulating the space $\{0,1\}^n$ for the input and output variables, respectively. Specifically, we map the CIRCLE and BLOCK truth tables to order-4 tensors, with horizontal bonds corresponding to to ancillary buses and vertical bonds corresponding to variables. Tables 5.1 and 5.2, show this mapping for the CIRCLE and BLOCK gates, respectively.

Table 5.1: Truth table and the corresponding tensor components for the CIRCLE gate. On the input side, $\alpha = x$, $\beta \equiv (ab) = 2^1 a + 2^0 b$; on the output side, $\gamma = x'$, $\delta \equiv (a'b') = 2^1 a' + 2^0 b'$. All unspecified components are zero.

Input			Output			Tensor component
x	a	b	x'	a'	b'	$T_{\alpha\beta\gamma\delta} \equiv T_{x(ab)x(a'b')}$
0	0	0	0	0	0	$T_{0000} = 1$
0	0	1	0	0	1	$T_{0101} = 1$
0	1	0	0	1	0	$T_{0202} = 1$
0	1	1	0	1	1	$T_{0303} = 1$
1	0	0	1	1	0	$T_{1012} = 1$
1	0	1	1	1	1	$T_{1113} = 1$
1	1	0	1	0	0	$T_{1210} = 1$
1	1	1	1	0	1	$T_{1311} = 1$

Table 5.2: Truth table and the corresponding tensor components for the BLOCK gate. On the input side, $\alpha = x$, $\beta \equiv (ab) = 2^1 a + 2^0 b$; on the output side, $\gamma = x'$, $\delta \equiv (a'b') = 2^1 a' + 2^0 b'$. All unspecified components are zero.

Input			Output			Tensor component
x	a	b	x'	a'	b'	$T_{\alpha\beta\gamma\delta} \equiv T_{x(ab)x(a'b')}$
0	0	0	0	0	0	$T_{0000} = 1$
0	0	1	0	1	1	$T_{0103} = 1$
0	1	0	0	1	0	$T_{0202} = 1$
0	1	1	0	0	1	$T_{0301} = 1$
1	0	0	1	1	1	$T_{1013} = 1$
1	0	1	1	0	0	$T_{1110} = 1$
1	1	0	1	1	0	$T_{1212} = 1$
1	1	1	1	0	1	$T_{1311} = 1$

Because the vertical bonds in the tensor network involve only one variable, these bonds are fixed to initially have a dimension of 2. The horizontal bonds, however, are dependent upon the number of ancillary variables contained in a bus. For m ancillary variables, the bond dimension is equal to the size of the tensor product space $2^{\otimes m}$. For the 2SAT problem, only two ancillary variables are needed, giving a bond dimension of 4. For the 3SAT problem, three ancillary variables are needed, giving a bond dimension of 8. Finally, for the XORSAT, only one ancillary variable is needed,

giving a bond dimension of 2.

To set these tensors in a rectangular network structure, we associate a single clause with a horizontally tiled set of tensors. If a variable is not included in the clause corresponding to the current horizontal ancillary bus "line", then an identity gate (transformed to a tensor in the same fashion as the CIRCLE and BLOCK tensors) is laid at the bus-variable intersection. If a variable is included in the clause, then either a CIRCLE or BLOCK tensor is laid. After doing this for all variables, the process is repeated for a new clause. Variables are optimally reordered according to the MSRO algorithm mentioned above and clauses are stacked in the vertical direction until all the clauses in a given satisfiability instance are tiled. The final tensor networks are given in Fig. 5.2 for both the #2SAT and #3SAT circuits.

Contraction

Contracting this network gives the total number of satisfying assignments for the original Boolean circuit. To compute this, we use a variant of the iterative compression-decimation (ICD) algorithm [59]. The TN is contracted from the left and right boundaries, iteratively alternating between a step of contraction of the first two tensor columns on the boundaries (decimation) and a step of SVD decompositions over every tensor pair in the lattice (compression). The former step renormalizes the length scale of the system, propagating initial boundary conditions into the bulk of the lattice. The latter step is reminiscent of the DMRG algorithm [60], removing short range "entanglement" in the system. By doing this step, contradicting configurations can be removed (because they produce singular values of zero), and low bond dimensions can be maintained. To accurately count the number of solutions through this process, all singular values above machine precision are kept ($\approx 10^{-15}$).

Results

 # 3XORSAT

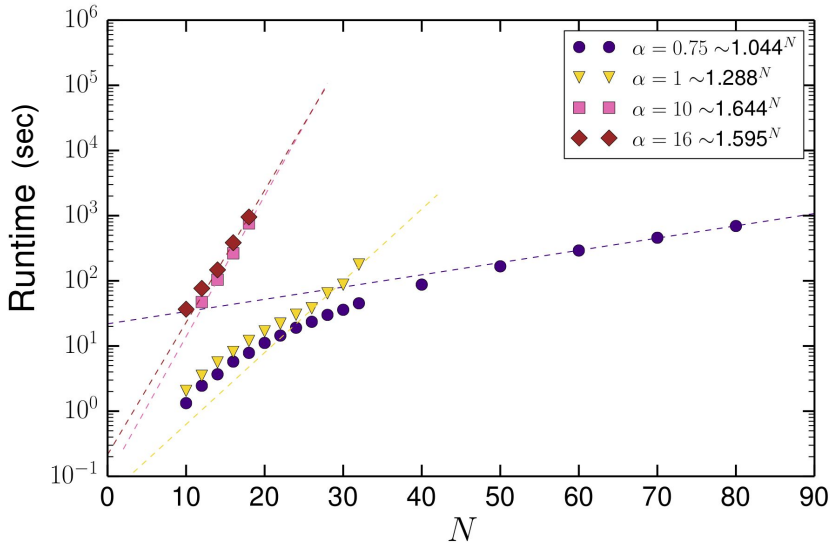


Figure 5.3: Results of the average runtime $\langle\tau\rangle$ to solution for the #3-XORSAT problem with varying number of bits $N = [10, 150]$ and differing clause-to-bit ratios α. Dashed lines fit exponential scales to the final four points of each α curve.

As a first test of our Boolean circuit embedding and tensor contraction, we considered #3XORSAT instances with 3-regular connected graphs (meaning every variable is involved in exactly three clauses). Such instances have a clause-to-variable ratio of $\alpha = 1$, setting them close to the known satisfiability threshold of $\alpha \sim 0.92$. Though the complexity class of satisfiability is known to be solvable in polynomial time by Gaussian elimination, many practical solutions do not achieve this performance. DPLL solvers [54], survey propagation [55], and annealing [56, 57] all fail to find solutions efficiently. We generated random instances of N variables by constructing a random $N \times N$ bi-adjacency matrix as the sum of three random row permutations of the identity matrix,

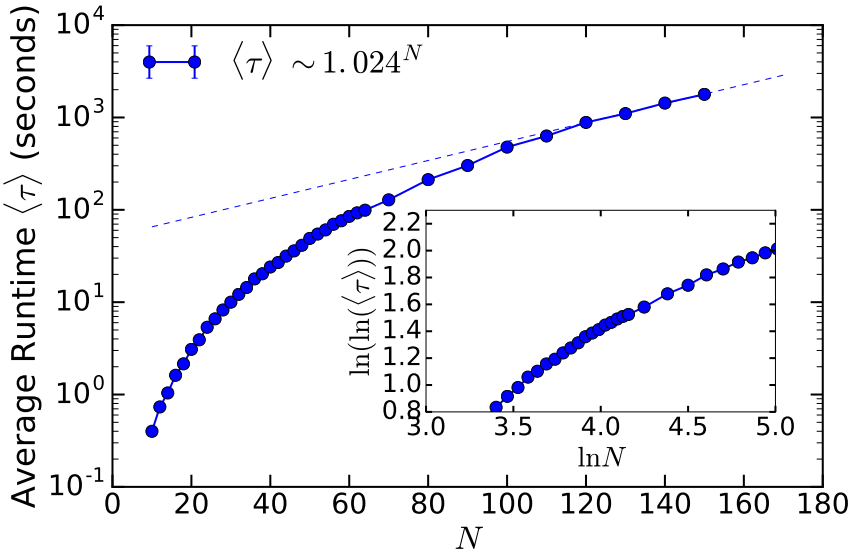


Figure 5.4: Scaling of the average runtime $\langle\tau\rangle$ for clause-to-bit ratio $\alpha = 0.75$. This regime is below the satisfiability transition $\alpha = 0.92$. The inset displays a linear trend in the plot of $\ln(\ln\tau)$ vs $\ln N$, indicating a polynomial scaling of the runtime.

accepting only instances which represent connected graphs. After converting this circuit to the TN formulation previously discussed, we contract this network for varying numbers of variables N, measuring both the average bond dimension at each iteration of the ICD, and the total time to solution. The averages for the time to solution are taken over 200 instances of the #3XORSAT, with results shown in Fig. 5.3. As shown in Fig. 5.4, below the $\alpha \sim 0.92$ satisfiability threshold, the time to solution appears to be sub-exponential, while above it the exponential dependence seems to dominate. As shown in Fig. 5.5, by tracking $\langle\chi_{\max}\rangle$, we observe that for XORSAT, the compression step of the ICD is highly effective in removing contradicting configurations from the network (i.e., reducing the bond dimension).

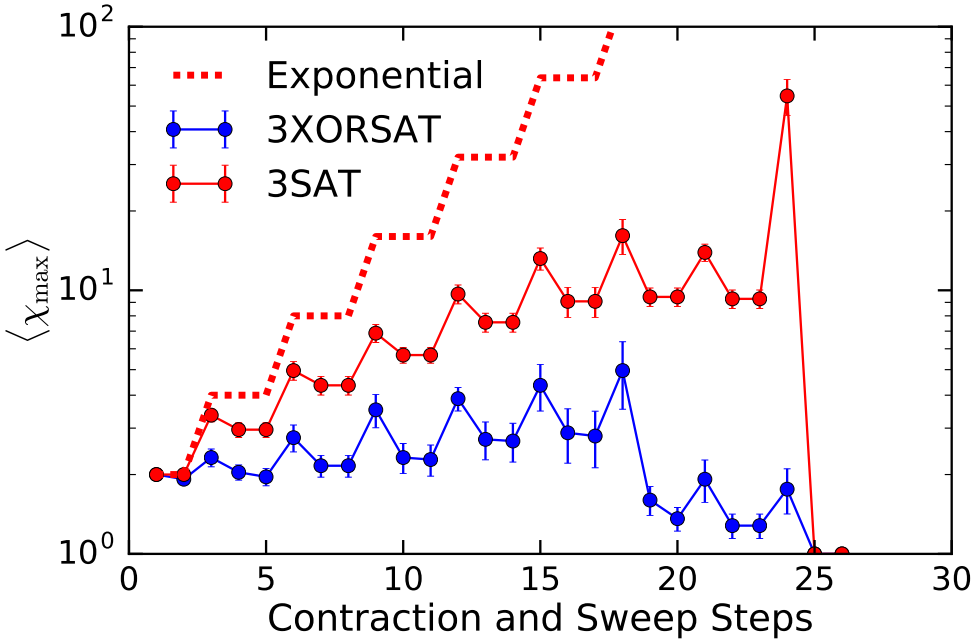


Figure 5.5: Calculations of the average maximum bond dimension $\langle \chi_{\max} \rangle$ occurring during each step of the ICD. When a decimation (contraction) step occurs, the bond dimensions are seen to increase, while during the compression (SVD sweeps) steps, bond dimensions decrease. Measurement were made for an clause-to-bit ratio $\alpha = 1$ for both the #3-SAT and #3-XORSAT instances.

#3SAT

Comparing these results to measurements on #3SAT 3-regular connected graphs (having #P-complete complexity), we see that for $\alpha = 1$, the ICD is ineffective at removing the contradictory configurations in the network, since bond dimension are seen to increase. This is indicative of a discriminatory behavior of the ICD. The compression step seems to allow the ICD to distinguish between classes of complexity. Furthermore, when measuring $\langle \chi_{\max} \rangle$ for a varying α, we see that the ICD can also discriminate between hard instances within a complexity class, based upon their graph structure. In Fig. 5.6, we see that the bond dimension for low α does not grow exponentially, indicating a quicker time to solution, whereas the bond dimension for high α does. Because α is a ratio of clauses to variables, we infer that α encodes information about the original Boolean

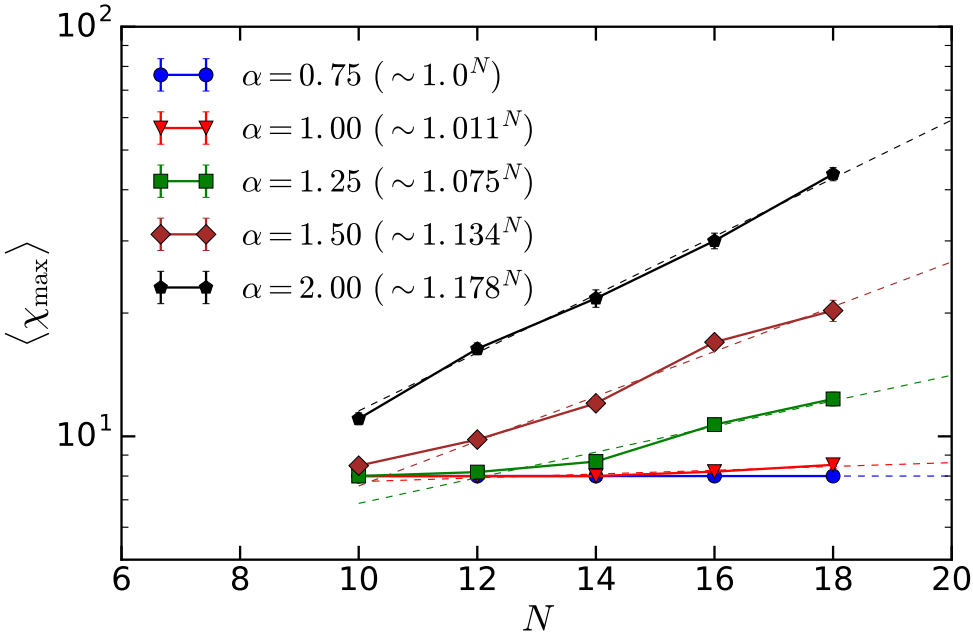


Figure 5.6: Calculations of the average maximal bond dimension $\langle \chi_{\max} \rangle$ occurring throughout the computation of solutions for the #3-SAT problem. Solutions are gathered over varying clause-to-bit ratios $\alpha \in [0.75, 2]$. Below $\alpha = 1$, the scaling of $\langle \chi_{\max} \rangle$ appears to be sub-exponential.

graph structure (low α signifying a tree-like graph structure with no loops). Therefore the α dependence of the bond dimension reduction through the compression step of the ICD is indicative of a discrimination property, which differentiates between graph structures of Boolean problems belonging to the same complexity class.

Conclusions

In this work, we have established a systematic procedure for converting SAT formulas to logic circuits embedded on two-dimensional TN lattices, with only local compatibility constraints on the gates. We have demonstrated the capability of the modified ICD TN contraction algorithm for various instances of the #3SAT and #3XORSAT problems. Our primary result is that the method dis-

criminates between the complexity classes defining XORSAT and SAT, leading to sub-exponential time scalings for the XORSAT. Secondarily, we notice that the method can infer the original underlying Boolean graph structure for SAT instances, despite the fact that the TN is constructed as a uniform rectangular lattice, again leading to an improved time scaling. The improvement in scaling is a consequence due in large part to the compression phase of the ICD algorithm, because bond dimensions are seen to decrease. In light of this, the SVD decomposition sweeps can be viewed as a parallel message passing method. Relevant information from the boundary of the network is passed into the bulk during each iteration, resolving compatibility constraints. We do note, however, that the runtime scaling of the #3SAT instances is not sub-exponential, suggesting that this method is not amenable to all #P-complete problems, but only a subset of them. It will therefore be necessary to establish further theoretical tools which can determine which complex problems are best suited for the ICD, particularly those where information is propagated throughout the network efficiently, leading to a decrease in the average maximal bond dimension.

CHAPTER 6: TN APPLICATIONS: QUANTUM CIRCUIT MEASUREMENTS ENHANCED BY ENTANGLEMENT DILUTION

The work contained in this chapter is currently being prepared for publication.

Introduction and Motivation

Moving from classical computations to quantum computations, we observe that when coupled with a novel TN tiling protocol, which we term *entanglement dilution*, the compression step in the ICD algorithm previously presented can also reduce the number of computational resources necessary to perform TN simulations of random quantum circuits, particularly quantum circuits which benchmark quantum supremacy. The primary goal of quantum supremacy is the demonstration that quantum computers provide a paradigm shift in computation, either accomplishing tasks that are classically impossible, or providing an orders of magnitude speed up on computations which are known to be difficult for classical computers [61]. Most recently, Arute *et al* developed the first experimental realization of quantum supremacy by using a 54 transmon qubit quantum computer to produce a quantum chaotic state [62]. This state has particular significance because it possesses qualities that automatically make it difficult for classical computers to simulate. Most notably, the system is in a highly entangled state and measurements over the computational basis are known to produce a Porter-Thomas distribution [63]. We will focus our attentions on the simulation of quantum circuits that lead to this quantum chaotic state.

Typically, classical simulations of these systems incorporate two elements: 1) tracking either the partial or complete information of the quantum state throughout the computation (as in the case of stochastic Monte-Carlo simulations of quantum computation [64]), and 2) aggregating the com-

putational gates into "super gates" in order to generate the fastest entanglement growth. As a specific example, both of these conventions were used by Pednault *et al* in the development of an algorithm utilizing novel secondary storage and memory management techniques to generate all the amplitudes of a random quantum circuit up to arbitrary depth [65]. In contrast to this, we choose to take a counter-intuitive approach to chaotic state preparation by noting that simulations of measurements on quantum circuits do not necessarily mandate the preservation of state information throughout the computation. Additionally, as will be seen in this paper, while aggregating gates does certainly expedite entanglement growth, it may ultimately prove counterproductive for simulations involving larger numbers of qubits.

Here, we develop a tensor network algorithm for the simulation of quantum circuit measurements, such that, 1) we circumvent the preservation of state information and 2) we intentionally "slow down" our approach to the highly entangled state. We do this by modelling our quantum circuit measurements in a manner analogous to the Keldysh formalism used in the study of out-of-equilibrium many-body systems, and by tiling our circuits in a way that prevents quick entanglement growth. We term this tiling process as *entanglement dilution*.

Methods

Analogy to the Keldysh Formalism

Before discussing entanglement dilution, we discuss the foundations of the Keldysh formulation for interacting many-body systems as a precursor to the construction of our quantum circuits. We then follow this by explaining the utilization of the Keldysh time contour when performing measurements on quantum circuits built on a tensor network model.

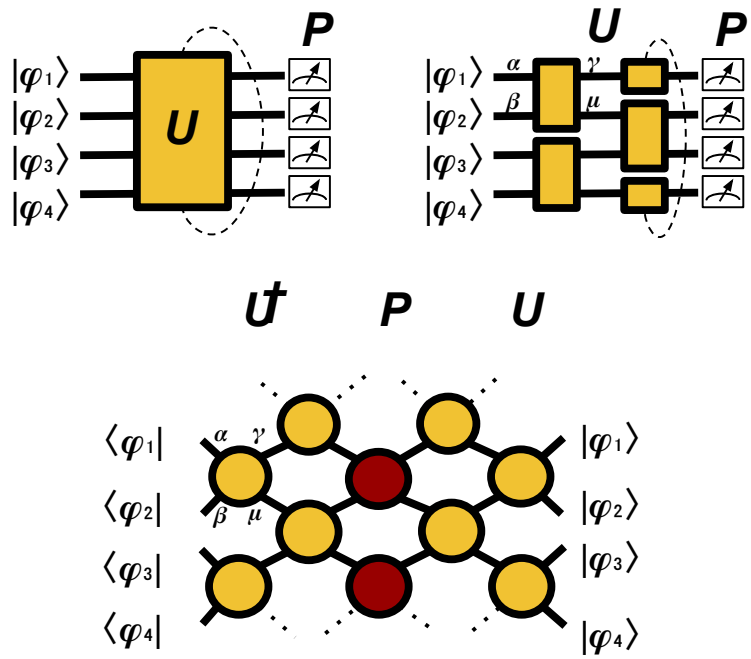


Figure 6.1: A schematic for the transformation of a quantum circuit to a $\pi/4$ rotated rectangular tensor network. Beginning with a circuit diagram, with qubits initially in the state $|\phi_i\rangle$, the transformation U is decomposed into 2-qubit operators which are reshaped into tensors. Dashed lines indicate periodic boundary conditions on the vertical bonds.

The expectation value of an operator $\mathscr{O}$ acting on a quantum state $|\Psi\rangle$ is given by

$$\langle \mathscr{O} \rangle = \langle \Psi | \, \mathscr{O} \, | \Psi \rangle \, . \tag{6.1}$$

For a time-dependent Hamiltonian $\mathscr{H}(t)$, the time evolution of this quantity can be evaluated by applying the time-ordered operator $\mathscr{U}_{t,0} = T\left[\int_0^t \exp\{-i\mathscr{H}(t')\}\, dt'\right]$ to an initial state $|\Psi_0\rangle$ so that the expectation value becomes

$$\langle \mathscr{O} \rangle = \langle \Psi_0 | \, \mathscr{U}_{0,t} \, \mathscr{O} \, \mathscr{U}_{t,0} \, | \Psi_0 \rangle \, . \tag{6.2}$$

If the evolution leads to a ground state of the Hamiltonian, it is assumed that adiabatic switching of the interacting terms in the evolution operator yield an extra phase factor, e^{iL} [66], where L is an arbitrary phase. This prefactor can present challenges when obtaining the disorder averages of expectation values, but when a system is driven out of equilibrium, the direct determination of this factor can be avoided by taking the evolution over a closed time contour (called the Keldysh contour) from 0 out to t and back [66].

To utilize this forward-backward propagation within the framework of quantum circuits, we first note that a quantum circuit model for N qubits is characterized by a few key elements: 1) an initial state vector $|\Psi_0\rangle$, which is in a product state over the N qubits, 2) a global unitary transformation which is decomposed into the sequential application of local unitary transformations drawn from a universal gate set $\mathbf{U}$, and 3) a measurement protocol which is performed after the evolution, often accomplished by the application of a global projector $\mathscr{P}$. These necessary features provide a straightforward path for carrying out an analogy between the expectation values of measurements on a quantum circuit and the expectation values obtained for any generic observable in an out-of-equilibrium QMB problem.

The analogy is developed as follows: first, we equate the unitary transformations obtained from some universal gate set with the time evolution operators from 0 to t. We then define the state of the system at $t = 0$ to be an initial product state, and set the state at $t = t$ to be the value obtained by the measurement operation. Time ordering the local unitary transformations amounts to specifying a particular instance of sequencing for the gates. In this way, the analogy is complete.

To practically implement this circuit construction, we utilize a two-dimensional $\pi/4$ rotated rectangular tensor network with periodic boundary conditions along the vertical direction, where each tensor $T^{(i)}_{\alpha,\beta,\gamma,\mu}$ in a column i implements a unitary transformation over a two-qubit space, with α,β being indices over the computational basis for two input qubit states and γ,μ being indices

associated with the output qubit states. Columns of tensors are stacked horizontally, alternating between even and odd pairs of tensors in the qubit chain. Each column propagates the initial product state through a single computational time step. Each tensor is laid within a column following the time ordering of the circuit until the forward evolution is complete. After this forward propagation, a global projector is laid. Finally, conjugate tensors are placed in a reverse time order to implement the backwards evolution. The state of the system is contained within the edges of the network. Tensors residing on the first and last columns are contracted along their open indices with a tensor product state of qubits such that each qubit $|\phi_j\rangle$ is specified by an angle θ_j as $|\phi_j\rangle = \cos\theta_j |0\rangle + \sin\theta_j |1\rangle$. Fig 6.1 displays this construction over a four qubit system evolving one step in time and being measured by projector $\mathscr{P}$.

Circuit Tiling

When selecting a protocol to tile these tensor gates for a quantum circuit, one can either aggregate one- and two-body gates into a single transformation at a given time step, or one can choose to place the one- and two-body gates into separate steps of computational time. The former approach is often chosen when the goal is to demonstrate quantum supremacy against classical algorithms. In this case, one intuits that by aggregating the transformations into one "super transformation", entanglement will spread throughout the system quickly. This rapid spread ensures that high entanglement regimes can be reached with a low circuit depth. While this intuition is valid, it is not necessarily advantageous, particularly if the desire is simply to simulate the measurement of large quantum circuits as opposed to obtaining information about the full state of the system.

For the specific case of measuring global projector expectation values, we choose to tile our gates in contrast to the conventional schemes involving aggregation. We name our tiling protocol *entanglement dilution*. Rather than approaching the highly entangled chaotic state quickly, we take an

approach reminiscent with adiabatic evolution. Foregoing aggregation, we lay one- and two-body gates independently at a given computational time slice. Furthermore, we allow identity gates to be laid throughout the network. An immediate consequence of this is that any given pair of two-body gates are not necessarily guaranteed to have direct interaction. They are spread throughout the system, separated by single-body operations. Since we associate the two-body gates as being the primary source of entanglement across the system, this construction guarantees a slow growth of entanglement. Naively, we could assume this separation introduces a disadvantage to simulations for quantum supremacy because it inherently increases the depth of the circuit. But as will be shown below, the inherent increase in depth introduced by entanglement dilution is not so costly that it makes measurements on circuits intractable. Instead, it allows us to make measurements over the quantum chaotic state for models involving a far greater number of qubits than those constructed with aggregated gates.

TN Contraction

Circuit measurements are computed by the ICD contraction of our TN. Recall from the previous chapter that in the ICD algorithm, the TN is contracted from the left and right boundaries, iteratively alternating between a step of contraction of the first two tensor columns on the boundaries (decimation) and a step of SVD decompositions over every tensor pair in the lattice (compression). The former step renormalizes the length scale of the system, while the latter step is reminiscent of the DMRG alogrithm [60], removing short range "entanglement." During the compression step, singular values, $\lambda_i^{(j)}$ of the jth tensor pair satisfying the condition $\lambda_i^{(j)}/\lambda_0^{(j)} > \varepsilon$ are kept. For our purposes, we set $\varepsilon = 10^{-9}$.

Before the ICD was performed, a pre-processing step was taken to decimate the initial diagonal lattice into a horizontal rectangular lattice, as shown in Fig 6.2. From this point, the ICD is per-

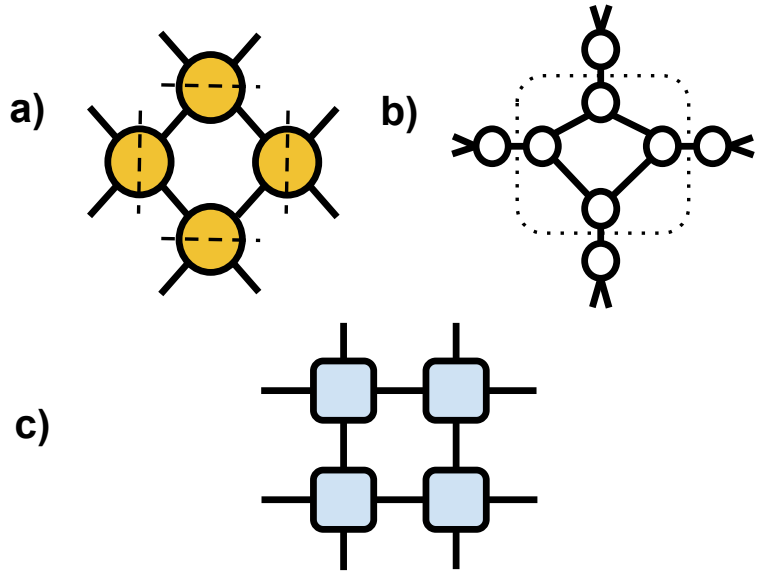


Figure 6.2: A tensor diagram for the decomposition rotation of the initial $\pi/4$ rectangular lattice. a) An SVD decomposition is performed over a group of 4 neighboring tensors. b) Tensors generated in the bulk of the neighbors are grouped and contracted together. c) Steps (a) and (b) are performed over every group of 4 tensors until the network is rotated.

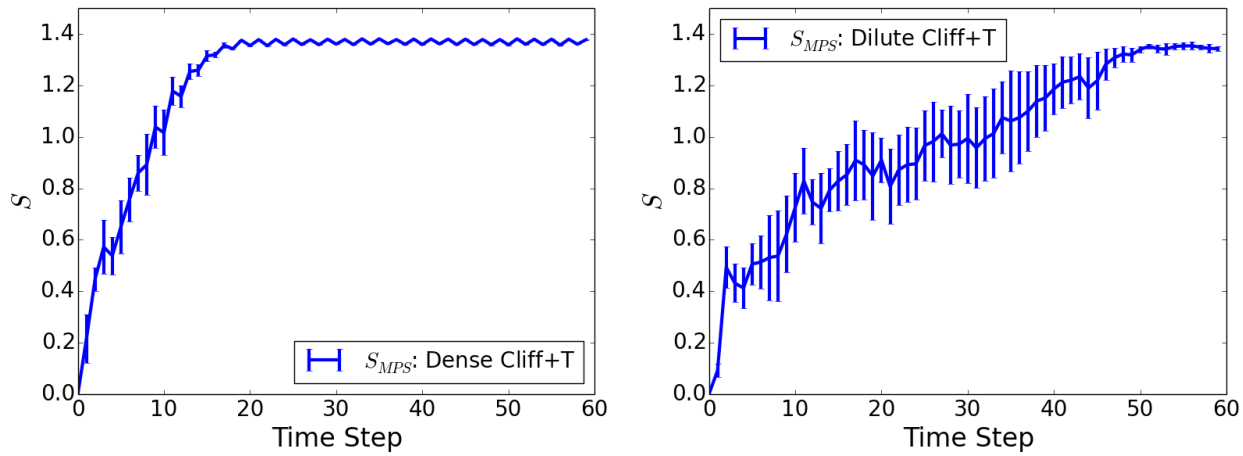
formed with decimation steps involving the neighboring tensors on the right and left boundaries. The alternating sequence of decimation and compression steps is carried out until a single tensor train with periodic boundary conditions is left. This is finally contracted to obtain the expectation value of the given operator on the circuit.

Results

We test our model and algorithm on random circuits drawn from two different gate sets. First we utilize the sycamore gate set composed of the gates $\mathbf{U}_{\text{syc}} = \{\sqrt{\text{X}}, \sqrt{\text{Y}}, \sqrt{\text{W}}, f\text{Sim}\}$ [62], generating random circuits for up to $N = 36$ qubits. Subsequently, we perform the same tests using the Clifford + T gate set $\mathbf{U}_{\text{Cliff+T}} = \{\text{Hadamard}, \text{T}, \text{CNOT}\}$ [67] for up to $N = 44$ qubits. All computations were

performed using an i3-4010U processor having a 3MB L3 cache and operating at 1.7 GHz.

Comparison of Circuit Tilings



(a) (b)

Figure 6.3: Entanglement Entropy S for MPS simulations of $N = 10$ qubits evolved to the quantum chaotic state by application of gates taken from the Clifford + T gate set. In (a), the circuit is constructed using dense gates. In (b), the circuit is constructed using dilute gates. The entropy is averaged over every two-site partitition in the qubit chain and over every realization. For both cases, the chaotic state is achieved when the entanglement has saturated to its maximal value. Only 5 realizations were used for each circuit type.

Before investigating the differences between dilute and dense circuits in our TN algorithm, we verify that the chaotic state can be achieved for a dilute tiling of gates if the number of computational steps is increased sufficiently. We do this by simulating each circuit tiling with a TEBD algorithm [68] applied over an initial MPS. At each time step, the entanglement entropy given in Eq. 1.22 is measured for two-site partitions in the MPS. In Fig. 6, the entropy is shown for $N = 10$ qubits evolving under the action of gates drawn from the Cliff+T gate set. Notice that the dense tiling requires $\approx 2N$ steps for the entanglement to saturate, while the dilute tiling requires $\approx 5N$ steps. Having verified the possibility of reaching the chaotic state, we note that our

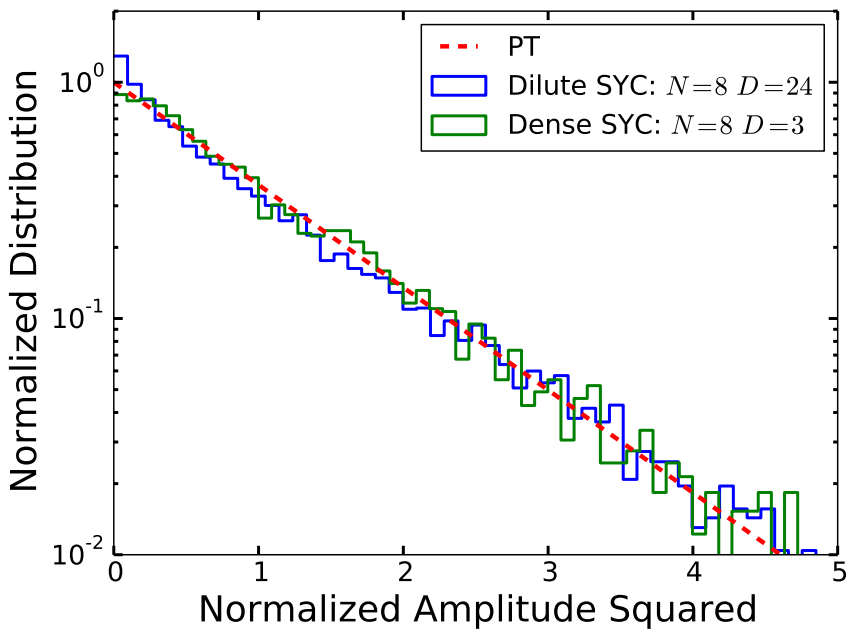


Figure 6.4: A comparison of the distribution of amplitudes taken over the projector $\mathscr{P}_0$ for both the dense and dilute tilings of sycamore circuits with $N = 8$ qubits. The distributions are similar and in agreement with the quantum chaotic Porter-Thomas (PT) distribution. Averages were taken over 5000 realizations.

TN algorithm does not preserve the state information and therefore cannot be used to measure the entropy throughout computation. For this reason, we choose instead to measure the distribution of amplitudes gathered from measurements of expectation values on circuits which differ in their tiling protocols, and compare them to the Porter-Thomas distribution.

Expectation values are gathered over many instances of random circuits with measurements done over the projector, $\mathscr{P}_0 = |000...00\rangle$. Because our TN algorithm can only generate the amplitudes of measurements taken on the circuits, we evolve the circuits until the distribution of amplitudes for $\mathscr{P}_0$ reaches the quantum chaotic Porter-Thomas distribution. As expected and shown in Fig. 6.4, this regime occurs for a differing number of time steps for circuit tilings which differ in the concentration of their primary entangling gates. The "dense" circuit is defined as having all gate

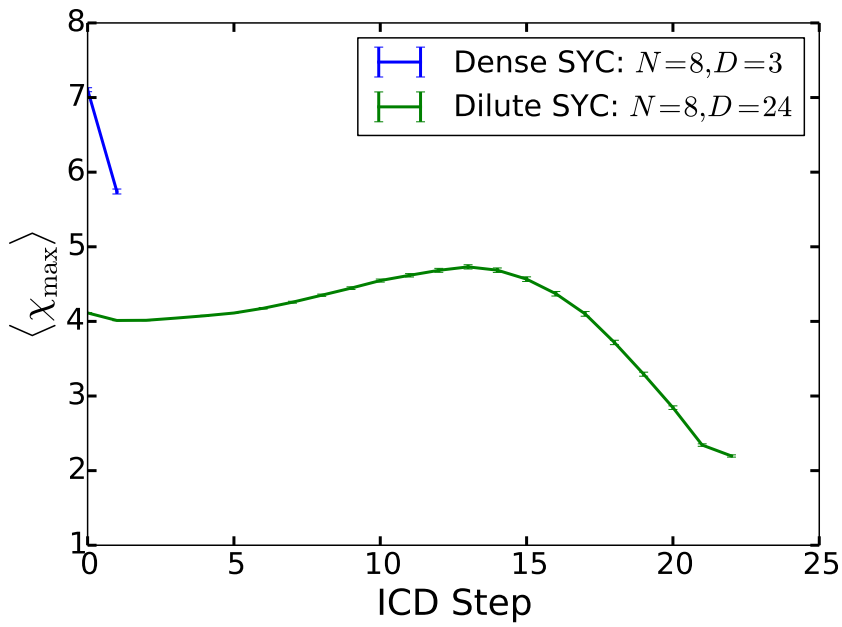


Figure 6.5: A comparison of the average maximum bond dimension, $\langle \chi_{\max} \rangle$, encountered at a given ICD step for dense and dilute tiling of random sycamore circuits with $N = 8$ qubits. Each ICD step consists of ten decomposition sweeps, followed by a single step of decimation from the right and left edges.

operations aggregated into a "super-gate", so that the concentration of primary entangling gates is 1. The "dilute" circuit separates the one- and two-body gates, assigning them to the circuit with equal probability. For the dilute circuit in Fig. 6.4, the concentration of entangling gates in the dilute circuit is 0.25.

Confident that the chaotic state has been reached within both circuit tiling schemes, we move on to understanding the difference in computational time steps that arises between the two entanglement regimes. To do this, we monitor the average maximal bond dimension $\langle \chi_{\max} \rangle$ in the tensor network through each iteration of the ICD. Again comparing dense and dilute circuits, Fig. 6.5 indicates that the separation of two-body gates in the circuit reduces the concentration of entangling gates and produces a regime where the $\langle \chi_{\max} \rangle$ is maintained at a low value. The eight-qubit sycamore

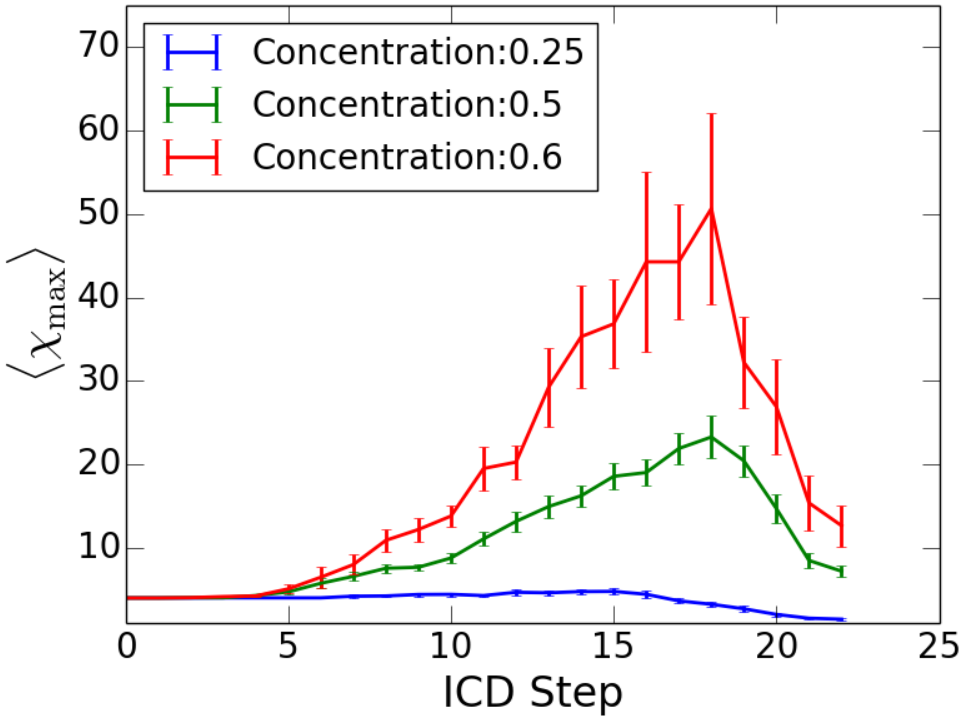


Figure 6.6: Plot of the average maximum bond dimension $\langle \chi_{\max} \rangle$ occurring for each ICD iteration for varying entangling gate concentrations. Random circuits are constructed with 12 qubits and a circuit depth of 24. Gates are selected from the $\mathbf{U}_{\text{Cliff+T}}$ gate set, with the concentration determined by the number of CNOT gates present in the system.

circuit constructed with aggregated gates only (blue) takes three computational steps to reach the chaotic state. However, $\langle \chi_{\max} \rangle$ for this circuit begins at a high value and is not decreased until the final time step. In contrast, the eight-qubit sycamore circuit constructed under the entanglement dilution protocol (green) only reaches a maximal $\langle \chi_{\max} \rangle$ after ≈ 14 computational steps, and even then this value is not significantly larger than the initial $\langle \chi_{\max} \rangle = 4$.

To qualitatively observe the transition between these regimes, we constructed random circuits with fixed number of qubits, $N = 12$, and fixed computational depth 24. Again, $\langle \chi_{\max} \rangle$ was measured over each ICD step. In Fig. 6.6, we can see that as the concentration of entangling gates increases,

the rate of increase of $\langle \chi_{\max} \rangle$ increases. This observation reinforces the intuition that aggregated gates (i.e., dense concentrations of entangling gates) speed up the entanglement process.

Figure 6.7 provides a more complete understanding of what occurs during each iteration when there is no entanglement dilution present. During each step of the ICD, the color coded bond dimensions in Fig. 6.7 indicate the size of bond dimensions throughout the tensor network. Decomposition sweeps are not displayed after the first iteration of decomposition because bond dimensions are no longer reduced after this step. Instead, the bond dimensions maintain their maximal value. Using the size of these bond dimensions as an indicator of the entanglement present in the system, it is clear that the dense system immediately becomes highly entangled, as is expected. However, this quick approach makes finding low-rank decompositions practically impossible. In the opposite case, where entanglement has been made dilute, the Schmidt decompositions do continue to succeed in finding low rank approximations to tensor pairs after the first iteration. Therefore, entanglement dilution is shown to enhance the maintenance of a low $\langle \chi_{\max} \rangle$ throughout the ICD. This comes at the cost of increasing the number of time steps necessary for reaching the quantum chaotic state.

Large Scale Implementation

What we find most surprising about this dilution result is that the inherent increase in the depth of the circuit does not prevent the simulation of circuits involving large numbers of qubits. In fact, the entanglement dilution approach enhances our simulation capabilities to the point that we can generate quantum chaotic states for circuits involving large numbers of qubits. Analyzing the small system size case in Fig. 6.5, it can be seen that though there is a factor of 8 increase in the number of computational steps, there is also a factor of 2 decrease in $\langle \chi_{\max} \rangle$. Since the scaling of tensor network contractions over 2-D networks with uniform bond dimension $D = \langle \chi_{\max} \rangle$ and length L

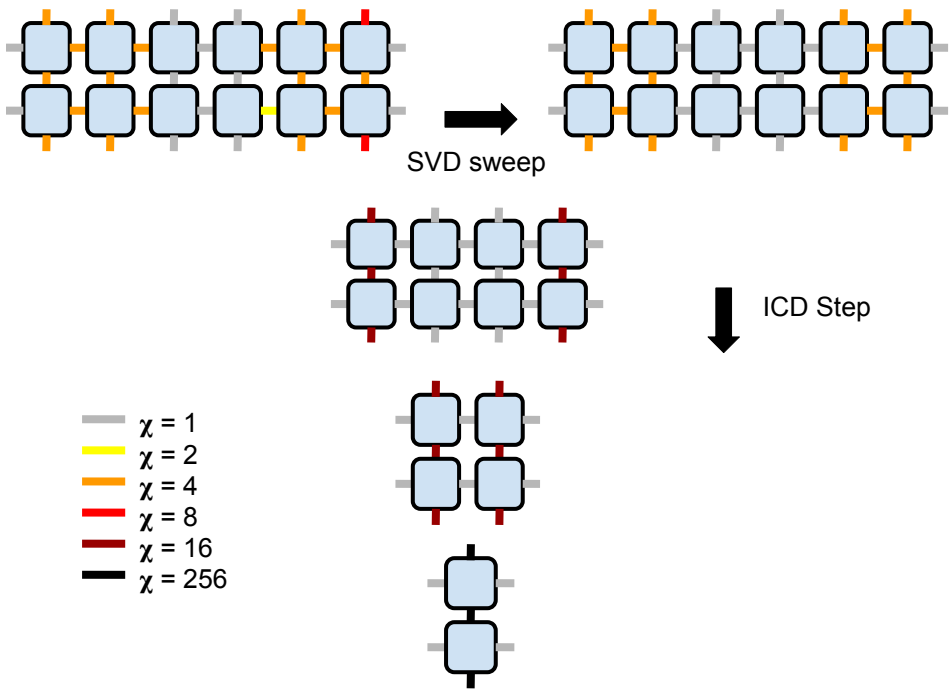


Figure 6.7: A tensor diagram for a 4-qubit random quantum circuit with gates taken from $\mathbf{U}_{\text{syc}}$. Bonds dimensions are color-coded. The horizontal arrow indicates a single pass of Schmidt decompositions over every tensor pair. The vertical arrow indicates progression to the next decimation step in the ICD algorithm. It is clear that as the bond dimensions continue to grow through the decimation process up to the final step.

is $\approx O(D^L)$, the reduction of D by entanglement dilution introduces a more significant effect upon the computational complexity than the extension of L. The counterbalance of the reduction in D on the extension in L is what allows us to simulate circuits with increasingly larger numbers of qubits. To investigate the limits of this extension, we simulate circuits with the largest number of qubits possible for our given method and for our available processing power. We show the results of these large circuit simulations in Fig. 6.8. The distributions shown are taken over 1000 measurements on 44 qubit circuits drawn from $\mathbf{U}_{\text{Cliff+T}}$ and 36 qubit circuits drawn from $\mathbf{U}_{\text{syc}}$.

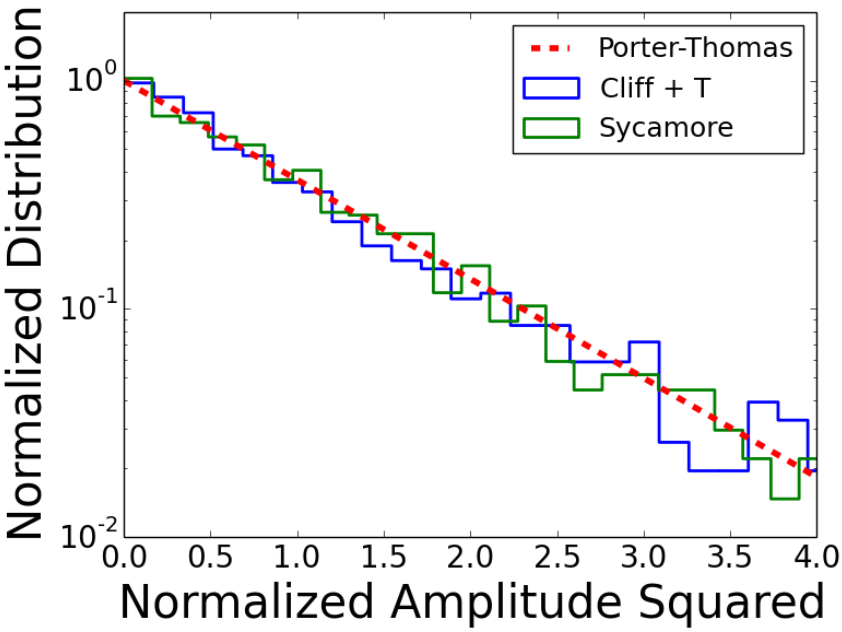


Figure 6.8: Distribution of the squared amplitudes for random circuits generated from the $\mathbf{U}_{\text{Cliff+T}}$ and $\mathbf{U}_{\text{syc}}$ gate sets. For the former, circuits were simulated with $N = 44$ qubits, and for the latter, circuits were simulated with $N = 36$ qubits. Distributions are compared to the Porter-Thomas distribution. Statistics were gathered over 1000 instances each.

Conclusions

Entanglement dilution is a protocol which limits the memory requirements for any given random circuit. By "diluting" the entangling gates in a given calculation, we have shown that difficult computational tasks, such as the state preparation of the quantum chaotic state, can be made tractable. Specifically, we have uncovered the existence of two regimes of computational efficiency for the simulation of quantum circuit measurements. These regimes transition between each other as a function of the concentration of entangling gates present in the circuit.

It has been shown that entanglement dilution protocol itself provides a novel means of simulating larger circuits without involving significantly more computational resources. The advantage

gained by limiting the entanglement growth (i.e., bond dimension increase) outweighs the disadvantage of requiring more computational steps. As a method of measuring the amplitudes of measurements on quantum circuits, entanglement dilution is demonstrably sufficient for the task. Further improvements to this protocol can be explored. Immediately obvious is that improvements in processing power can enhance simulation capabilities. Other less obvious but still promising avenues of improvement lie in exploring the effects of using different decimation protocols in the ICD and in implementing multi-threaded processes.

As a secondary consideration, we note that this method is useful for applications beyond random circuit models. Quantum algorithms for addition [69], prime factorization, and discrete logarithms [70] naturally have circuit definitions that are dilute in the concentration of their entangling gates. This property is present in these algorithms because of the interdependence the qubits have on previously calculated information throughout the computation. As such, the simulation of systems of this type would be well suited for our entanglement dilution ICD algorithm. Here the decomposition steps in our procedure would clearly work to mitigate the growth of the bond dimensions within the network.

CHAPTER 7: TN APPLICATIONS: EFFECT OF PROJECTIVE MEASUREMENTS ON QUANTUM CIRCUIT MODELS

Most of the work contained in this chapter is taken from a paper which has been published in Physical Review B in collaboration with Lei Zhang, Stefanos Kourtis, Claudio Chamon, and Andrei Ruckenstein.

Introduction and Motivation

Continuing with the study of quantum circuits, we note that quantum circuits can be generically viewed as quantum systems undergoing a unitary evolution. Such an evolution generally takes the quantum system to a thermalized regime, where it exhibits a volume-law entanglement [71, 72, 73, 74, 75]. Exceptions to this scenario have drawn considerable attention, due to their relevance in experimentally controllable quantum systems. For example, many-body localization, which precludes thermalization and leads instead to area-law entanglement, has been the subject of extensive theoretical and experimental work [76, 77, 78, 79, 80, 81, 82, 83, 84, 85]. Recently, a failure to thermalize has also been reported in simulations of quantum circuits subjected to random local measurements. These local measurements are used to model the system coupling to a classical environment [86, 87, 88, 89, 90, 91, 92, 93]. Investigations with this circuit model have shown that by evolving an initial product state with a volume-law entangling gate set (e.g., Clifford + T) and subsequently applying random local projective measurements (i.e., Z-gates) with probability p, one can disentangle the system, eventually leading to the localization of the system in the Hilbert space [86, 87, 88, 89, 90, 91, 92, 93]. Continued interest in these models is fueled by ongoing efforts to exploit noisy intermediate-scale quantum devices for tasks beyond the reach of classical computers.

In this work, we study quantum circuits subjected to projection operations randomly inserted at a finite rate throughout the time evolution of the circuit. We employ the level spacing statistics of the entanglement spectrum [94], referred to hereafter as “the entanglement spectrum statistics" (ESS), as a measure of thermalization and entanglement [95, 96, 97]. Our computations are carried utilizing the iterative tensor network contraction method introduced in Ref. [59]. As a function of projection rate p, we study the transition from a Wigner-Dyson distribution of the ESS (characteristic of the volume-law entanglement) to the Poisson distribution (characteristic of the area-law entanglement).

Methods

Entanglement Spectrum for Random Circuits

To compute the entanglement spectrum, we consider n qubits evolving in time t from an initial product state of the form

$$|\Psi(t=0)\rangle = |\psi_1\rangle \otimes |\psi_2\rangle \otimes \cdots \otimes |\psi_n\rangle \ , \tag{7.1}$$

where the single-qubit state for the j-th qubit is defined as $|\psi_j\rangle = \cos(\theta_j/2)\,|0\rangle + \sin(\theta_j/2)e^{i\phi_j}\,|1\rangle$ with arbitrary angles θ_j and ϕ_j. This initial state evolves under the action of (i) random unitary gates, and (ii) single-qubit projection operators, randomly inserted after each gate with a finite probability p. The state at time t is

$$|\Psi(t)\rangle = M\,|\Psi(t=0)\rangle = \sum_x \Psi_x(t)\,|x\rangle \ , \tag{7.2}$$

where $|x\rangle = |x_1 x_2 \ldots x_n\rangle$ is a configuration in the computational basis with $x_j = 0, 1$ for $j = 1, \ldots, n$, and M is a $2^n \times 2^n$ non-unitary matrix describing both the unitary evolution and the projection operations. The resulting circuit is illustrated in Fig. 7.1(b), where two-qubit gates are represented as blocks and projection operators as circles.

We consider a projection operator which acts on the j-th qubit to be of the form $M_0 = I_1 \otimes I_2 \otimes \cdots \otimes |0_j\rangle \langle 0_j| \otimes \cdots \otimes I_n$, where I_j is the identity operator on a single qubit. Such projection operators are not norm-preserving. We therefore normalize final states for consistency. Projection operators can be physically interpreted as randomly picking a qubit and resetting it to the computational basis.

After undergoing the evolution described above, a pure state $|\Psi\rangle$ can be expanded in the computational basis as

$$|\Psi\rangle = \sum_x \Psi_x |x\rangle \tag{7.3}$$

with x being a configuration of qubits, and Ψ_x being a corresponding coefficient. Ψ_x can be reshaped into a matrix $\Psi_{A,B}$ by partitioning the state into subsystems A and B, so that $|\Psi\rangle$ becomes

$$|\Psi\rangle = \sum_{x_A, x_B} \Psi_{A,B} \, |x_A\rangle \otimes |x_B\rangle \ , \tag{7.4}$$

where x_A and x_B are the configurations for subsystems A and B, respectively. The entanglement spectrum can be obtained by Schmidt decomposition [94]

$$|\Psi\rangle = \sum_k \lambda_k \, |x_A^k\rangle \otimes |x_B^k\rangle \ . \tag{7.5}$$

The set of entanglement levels λ_k defines the entanglement spectrum (ES). The entanglement entropy is given by

$$S = -\sum_k \lambda_k^2 \ln \lambda_k^2 \, . \tag{7.6}$$

By ordering the ES in descending order, $\lambda_k > \lambda_{k+1}$, the ratio of adjacent gaps in the spectrum can

be defined as

$$r_k = \frac{\lambda_{k-1} - \lambda_k}{\lambda_k - \lambda_{k+1}} \,. \tag{7.7}$$

It is this distribution which we of level spacing ratios that we focus our attention on. For Haar-random states, the probability distribution follows Wigner-Dyson statistics from random matrix theory [98, 99] and fits well the surmise [100]

$$P_{\mathrm{WD}}(r) = \frac{1}{Z} \frac{(r+r^2)^{\beta}}{(1+r+r^2)^{1+3\beta/2}} \tag{7.8}$$

with $Z = 4\pi/81\sqrt{3}$ and $\beta = 2$ for the Gaussian Unitary Ensemble (GUE) distribution. In contrast, the ESS for integrable systems takes the Poisson form

$$P_{\mathrm{Poisson}}(r) = \frac{1}{(1+r)^2} \,. \tag{7.9}$$

As a note on the two distributions, the most marked difference between the GUE and the Poisson is the level repulsion ($P_{\mathrm{WD}} \to 0$ for $r \to 0$) in the former and its absence ($P_{\mathrm{Poisson}} > 0$ at $r = 0$) in the latter.

Quantum Circuits as TNs

To calculate the ESS, we map the random quantum circuits described above into a rectangular tensor network and detail the contraction. Each two-qubit gate g is expressed as a 4×4 unitary matrix $T^{g}_{(i_1 i_2)(o_1 o_2)}$, where $(i_1 i_2)$ and $(o_1 o_2)$ are combined indices corresponding to gate input and output qubit states, respectively. Each 4×4 matrix is reshaped into a $2 \times 2 \times 2 \times 2$ tensor $T^{g}_{i_1 i_2 o_1 o_2}$. To transform the circuit into a rectangular lattice geometry, we regroup the indices of each tensor

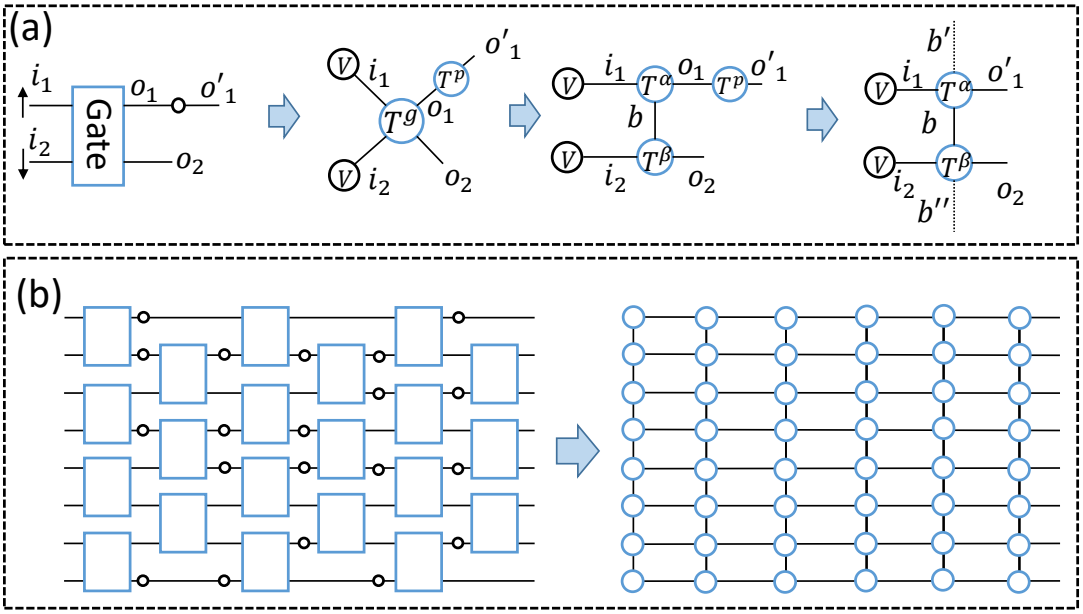


Figure 7.1: (a) A gate and and projector are reshaped into tensor T^g, which is then decomposed via SVD, and the tensor projector aggregated together with the gate. (b) The procedure for reshaping a single tensor and projector is applied over a full circuit until a rectangular tensor network is formed.

to reshape it to a matrix $T^g_{(i_1o_1)(i_2o_2)}$, using an SVD to produce

$$\begin{aligned} T^g_{(i_1o_1)(i_2o_2)} &= \sum_{m,m'} U_{(i_1o_1)m}\,\Sigma_{mm'}\,V^\star_{(i_2o_2)m'} \\ &= \sum_{m,b,m'} U_{(i_1o_1)m}\sqrt{\Lambda}_{mb}\sqrt{\Lambda}_{bm'}\,V^\star_{(i_2o_2)m'} \\ &= \sum_b T^\alpha_{(i_1o_1)b}\,T^\beta_{(i_2o_2)b}\,, \end{aligned} \tag{7.10}$$

where $U_{(i_1o_1)m}$ and $V_{(i_2o_2)m'}$ are unitary matrices and $\Lambda_{mm'}$ is a semi-positive diagonal matrix containing the singular values. The two new matrices $T^\alpha_{(i_1o_1)b} = \sum_m U_{(i_1o_1)m}\sqrt{\Lambda}_{mb}$ and $T^\beta_{(i_2o_2)b} = \sum_{m'}\sqrt{\Lambda}_{bm'}V^\star_{(i_2o_2)m'}$, with b an index running over singular values, are then reshaped to tensors $T^\alpha_{i_1o_1b}$ and $T^\beta_{i_2o_2b}$. To complete the rectangular geometry, we introduce a fourth "dummy" index

with dimension 1 to each tensor, connecting it with a neighboring tensor in the space dimension, as indicated by the faint vertical lines in Fig. 7.1(a).

Each single-qubit projector can also be expressed as a tensor $T^{p}_{o_1 o'_1}$, where o'_1 has dimension 1. These can be contracted with gate tensors as

$$T^{\alpha}_{i_1 o'_1 b b'} = \sum_{o_1} T^{\alpha}_{i_1 o_1 b b'} T^{p}_{o_1 o'_1} \,. \tag{7.11}$$

Initial states are taken to be product states, being written simply as the tensor product of single-qubit vector states.

Finally, wherever no gates are applied to qubits at the top and bottom boundaries, a rank-3 identity tensor $\delta_{i_1 o_1 b}$ is added to complete the square lattice. The final tensor network is shown in Fig. 7.1(b).

TN Contraction

Contraction of this lattice is carried out following the ICD algorithm along the width of the lattice, until a single column of tensors representing the final state of the system is obtained. Decimation steps are carried out by contracting tensor pairs from the right and left boundaries, and compression steps are carried out by SVDs over every tensor pair in the lattice.

Results

Intermediate Entanglement Spectrum Regime

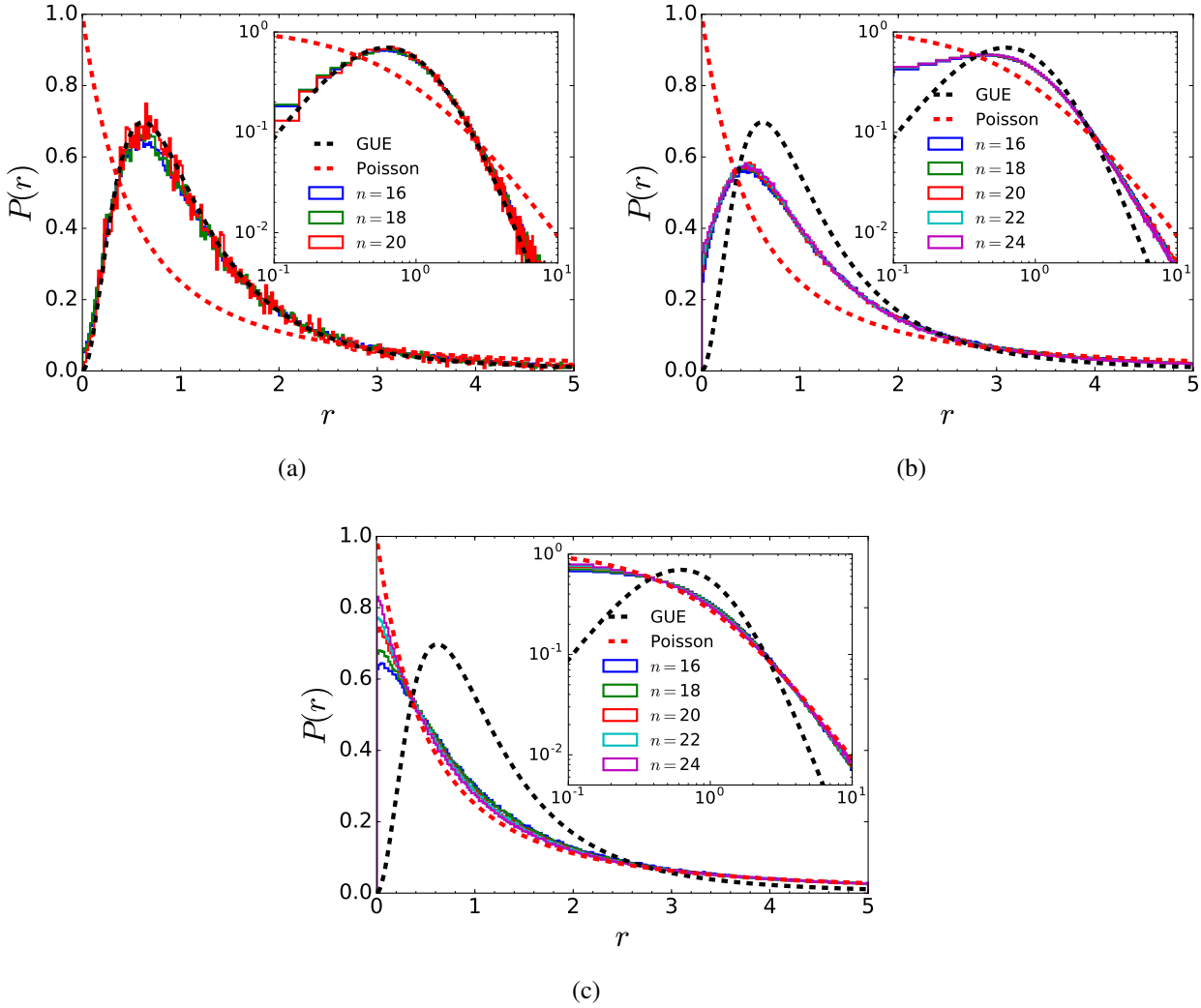


Figure 7.2: Level spacing ratio distributions for the entanglement spectrum in the three phases of entanglement. Distributions are taken from circuits constructed from the random Haar-measure with N bits and with depth $d = N$. (a) Volume-law phase at $p = 0.2$, as indicated by the GUE distribution with level repulsion in the limits $r \to 0$ and a Gaussian tail at $r \to \infty$. (b) Residual repulsion phase at $p = 0.35$. Level repulsion disappears at $r \to 0$ while a majority of levels show level repulsion as indicated by the presence of a peak at finite r. (c) Poisson phase at $p = 0.5$. The spectrum displays an absence of level repulsion in similarity to the Poisson distribution. The insets show the distributions in log-log scales in order to capture their behavior at the tails. In (a), results are obtained from 500 realizations for $n = 20$ and from 1000 realizations for $n < 20$. For (b) and (c), , results are obtained from 1000 realizations for up to $n = 24$.

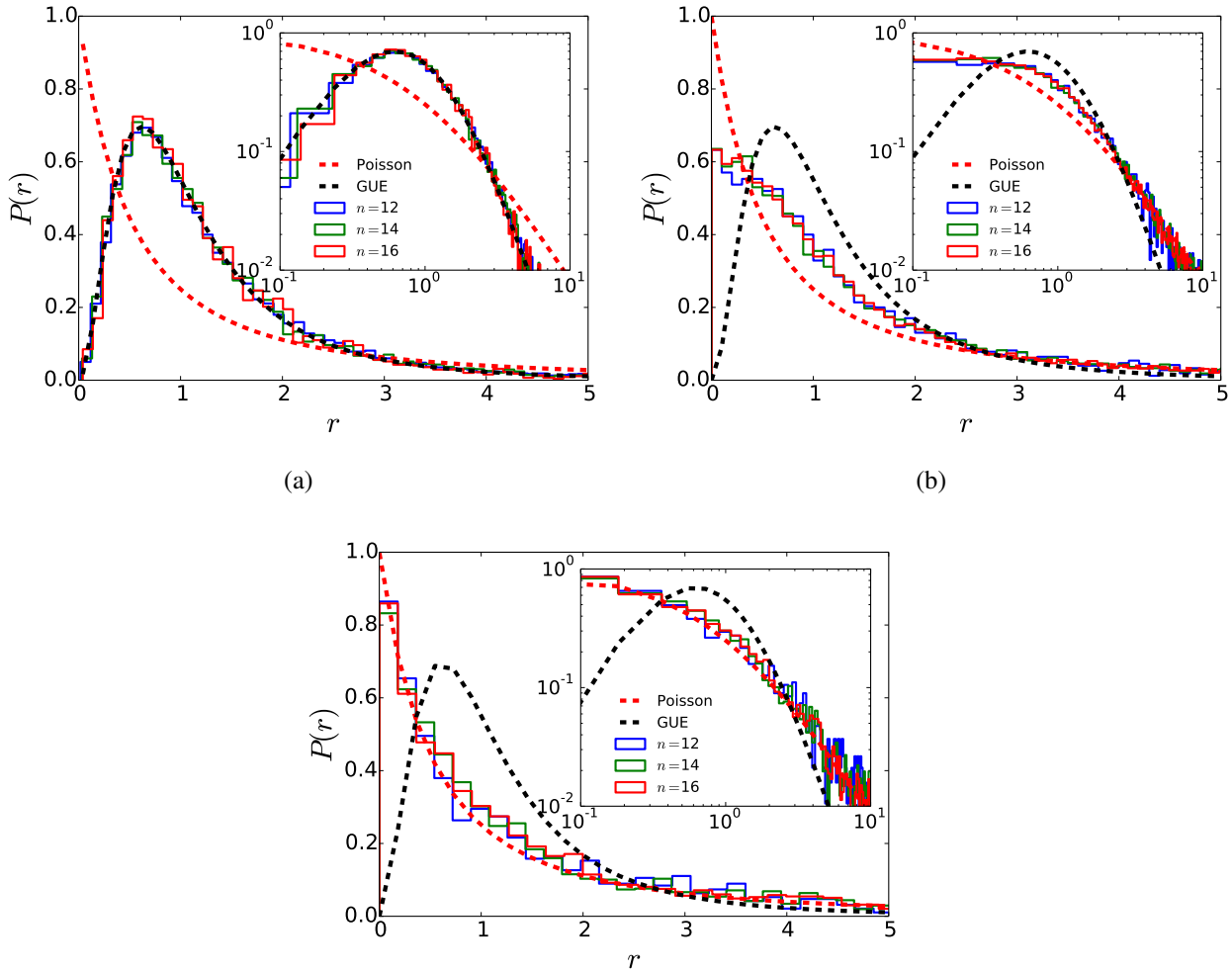


Figure 7.3: Level spacing ratio distributions for the entanglement spectrum in the three phases of entanglement. Distributions are taken from circuits constructed from the universal gate set $\mathbf{U}_{\text{syc}} = \sqrt{X}, \sqrt{Y}, \sqrt{W}$, fSim with N bits and depth $d = 2N$. (a) Volume-law phase at $p = 0$, again indicated by the GUE statistics in the level spacing. (b) Residual repulsion phase at $p = 0.2 > p_s = 0.15$, where the shifted peak has morphed into a plateau extending from zero. (c) Poisson phase at $p = 0.4 > p_c = 0.35$. The insets show the distributions in log-log scales in order to capture their behavior at the tails. Results are obtained from 500 realizations up to $n = 16$.

We numerically investigate the ESS as a function of projector density, p, in random quantum circuits satisfying the geometry given in Fig. 7.1. For each circuit realization, the initial state is

of the form of Eq. (7.1), where all θ_j and ϕ_j are selected uniformly at random. For comparative purposes, two separate gate sets were used to construct the random circuits. The first case is composed of two-qubit gates selected from the Haar-random measure, while the second consists of gates uniformly selected from the universal gate set $\mathbf{U}_{\text{syc}} = \{\sqrt{X}, \sqrt{Y}, \sqrt{W}, \text{fSim}\}$ [62]. A single-qubit projector is applied with probability p to each qubit after every gate and before the final time step. We calculate the ES of many random realizations, binning the spectra for the same p to obtain the ESS. Our results are summarized in Figs. 7.2 and 7.

We characterize the quantum chaotic and integrable regimes for these circuits with Wigner-Dyson and Poisson distribution ESS, respectively, and then proceed to describe a previously unforeseen intermediate regime. For $p < p_S$, the ESS follows the GUE distribution over the entire range of r, as seen in Figs. 7.2a and 7a. This corresponds to a highly entangled final state (i.e., volume-law state), indicating that the system has settled into the quantum chaotic regime. On the other hand, for $p > p_c$, with $p_c \simeq 0.41$, the ESS follows the Poisson distribution, which indicates that the system is integrable — see Figs. 7.2c and 7c. The intuition for this change in the ESS as a function of p is that as the frequency of projectors is increased, the system becomes frozen in local states, therefore failing to entangle [86, 87, 88, 89, 90, 91, 92, 93]. The small deviations from the expected exact Poisson distribution are due to our choice to avoid placing projectors in the final time step of simulations. This choice prevents us from potentially reducing the number of qubits in the system at the final time step. This effect disappears in the thermodynamic limit.

The primary result of this work is the discovery of an intermediate regime between $p_S < p < p_c$, where the ESS smoothly transitions from the GUE distribution to the Poisson. We call this phase the *residual repulsion phase*. As shown in Figs. 7.2b and 7b, the strict level repulsion emblematic of the chaotic regime disappears, i.e., $P(r)$ becomes nonzero for $r \to 0$, and is replaced by a distribution that is between the two regimes, having a maximum at a non-zero value of r. For the quantum circuits taken from the set $\mathbf{U}_{\text{syc}}$, this transitionary phase exists within a shifted window

of p values, specifically $0.15 < p < 0.3$. We infer that this quicker transition occurs as a result of the particular universal gate set which we have selected. The argument is as follows: from $\mathbf{U}_{\text{syc}}$, the gate responsible for introducing entanglement into the system is the fSim gate, which is an iSWAP gate concatenated with a controlled Z. This gate has an internal block structure as $1\times 1, 2\times 2, 1\times 1$. Because of this structure, the entanglement created by this gate is not as robust as that introduced by a random Haar unitary gate, which has no predefined symmetries or structure. The entanglement arising from this structured gate is therefore more susceptible to the presence of projective measurements. Similar evidence for this argument is also found when considering a separate universal set $\mathbf{U} = \text{CNOT}, \text{T}, \text{Hadamard}$. For this gate set, the entangling gate (CNOT) has an internal structure of $2\times 2, 2\times 2$, and only encodes a bit flip operation. In this case, we found that the residual repulsion phase exists only within an even narrower window $0.01 < p < 0.03$. We will see that the local structure of these gates works to affect a global percolation of bond dimensions throughout the lattice. We have marked the points of transition from the residual repulsion to the Poisson by measuring the Kullback-Leibler (KL) divergence between the measured distribution and the Poisson distribution. The KL divergence is given as

$$D_{KL}(P(x)||Q(x)) = \sum_x P(x) \ln\left(\frac{P(x)}{Q(x)}\right), \tag{7.12}$$

where $P(x)$ and $Q(x)$ are two statistical distributions. Transition points are estimated where the distributions differ by less than ten percent.

Percolation Transition

To further understand the transitional phase between the GUE and Poisson statistics, we analyzed the effects of imposing a global structure on GUE matrices. This is done by generating block diagonal matrices composed of two square GUE sub-matrices. By modifying the relative sizes of

these two sub-matrices, we investigate the effects of breaking the larger system into independent subsystems. As the results in Fig. 7.4 indicate, the ESS of these matrices reveal the same intermediate phase as those given in Figs. 7. This indicates that the projectors induce a percolation of the bonds throughout the space-time of the circuit, breaking the system into smaller subsystems as more projectors are added, each subsystem still describable by Wigner-Dyson statistics, until finally the percolation threshold is reached and the system becomes Poisson.

Conclusions

By exploring projection-driven quantum circuits from the perspective of the ESS of the output state, our results uncover three distinct behaviors of the entanglement spectrum as a function of the projection rate, p, of qubits onto the computational basis. The first regime, $0 < p < p_S$, displays volume-law entanglement entropy and Wigner-Dyson statistics of the EES. At $p = p_S$ the systems undergoes a volume-to-area-law transition similar to that studied in Refs. [86, 87, 88, 89, 90, 91, 92, 93]. The principal result of this paper is that the ESS of the area law phase emerging at $p = p_S$ is non-universal and interpolates between Wigner-Dyson and Poisson statistics, with the region of residual level repulsion extending up to a second transition, $p = p_c > p_S$, beyond which the ESS of the system is Poisson. Our TN algorithm allows us to identify the transition from the intermediate phase to the Possion as a percolation transition of the bonds across the 1 + 1D space-time of the circuits.

This work leaves open the question of the origin of the intermediate regime with non-universal statistics of the entanglement spectrum. The nature of level statistics is determined by the details of the interactions between eigenvalues which can induce complex non-universal level statistics, including a Griffiths-like phase observed in studies of MBL [101]. More detailed work is needed to elucidate the non-universal regime in the ESS.

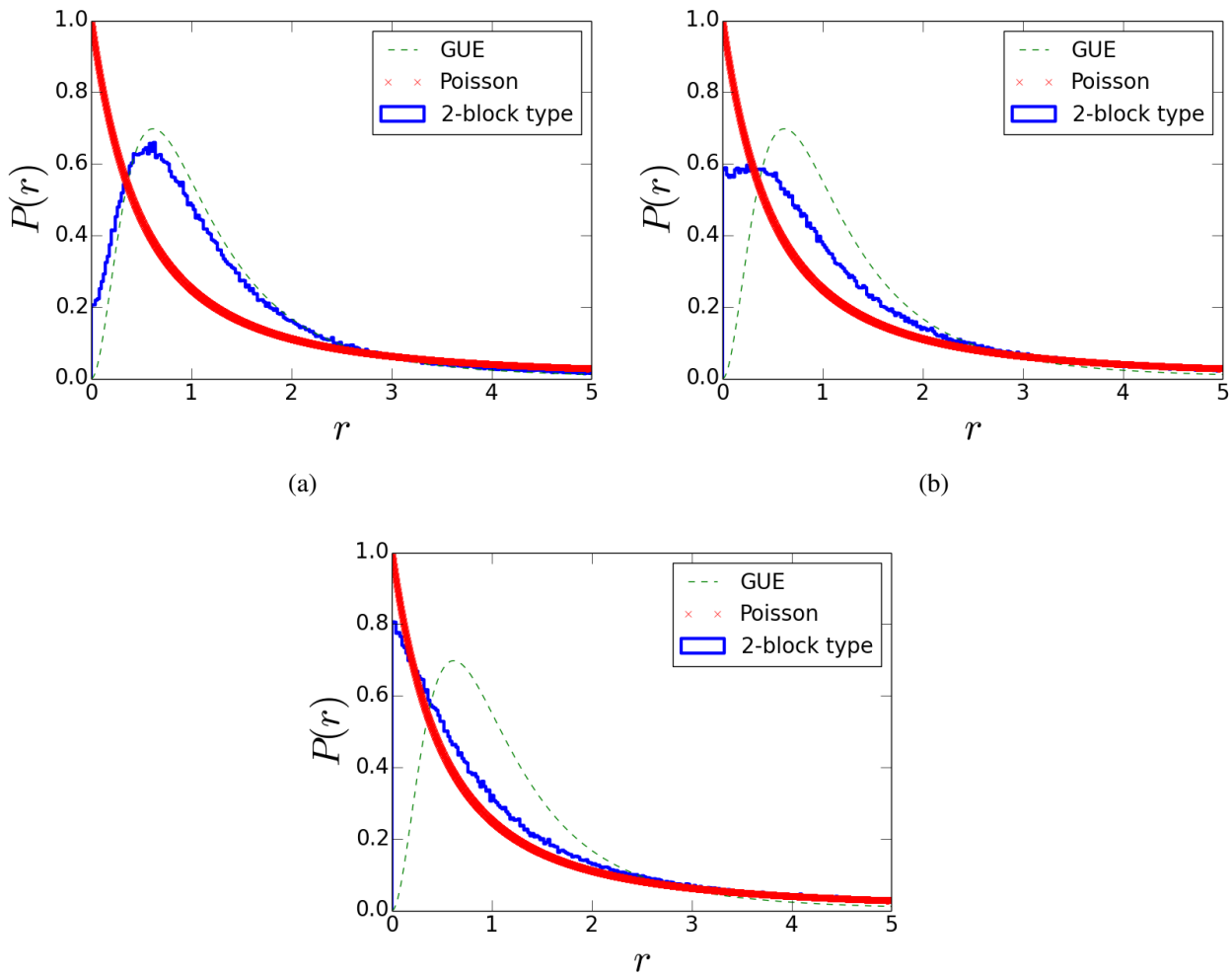


Figure 7.4: Level spacing ratio distributions for block-diagonal GUE matrices composed of two blocks A and B. The overall size of the composite matrix AB is 100×100, while the relative sizes of blocks A and B vary. In (a) block A is 10×10, while block B is 90×90. In (b) block A is 30×30, while block B is 70×70. In (c), block A and B are both 50×50. Distributions are gathers over 5000 samples for each block diagonal structured GUE matrix.

CHAPTER 8: TN APPLICATIONS: MULTI-SCALE TENSOR ARCHITECTURE FOR MACHINE LEARNING

The work contained in this chapter is taken from a paper which has been submitted for publication to the Journal of Machine Learning Research in collaboration with Miles Stoudenmire.

Introduction and Motivation

In this study, we recognize that there exists a certain overlap between TN QMB studies and machine learning. Tensor networks serve as a foundational tools in the simulation of QMB systems because the full system state vector—which encodes the joint probability of many variables—cannot be explicitly stored or manipulated due to the exponential growth of its dimensionality with the number of variables. Certain machine learning approaches, such as kernel learning, encounter this same problem when a model involves a large number of data features [102]. The similarity in the growth of the dimensionality of vector spaces in the two fields of study therefore motivates the use of tensor network algorithms for enhancing the performance of machine learning tasks.

Because tensors are multi-linear objects, controlled transformations of one tensor network into another are possible through techniques such as matrix factorization and tensor contraction [103, 104, 105]. We will take advantage of this capability for the initialization of our machine learning model. This is accomplished by initially training a model which is defined through a larger number of coarse-graining steps, and subsequently fine graining the top trained tensor of adjustable model weights until a desired architecture is reached. This architecture is utilized in two machine learning contexts: binary classification and regression.

The algorithm we present for training this model is built around three main steps. The first step

involves coarse graining to reduce the feature space representing the data, and consequently the number of parameters of the model. This is done through a series of wavelet transformations approximated by the layers of a MERA tensor network. The second step is the training of a weight tensor represented as an MPS and optimized by an alternating least squares algorithm analogous to the density matrix renormalization group (DMRG) algorithm. In the third step, we show how the weight tensor can be controllably fine grained into a model over a larger set of features, which initially has the same performance as the coarse-grained model but can then be further optimized.

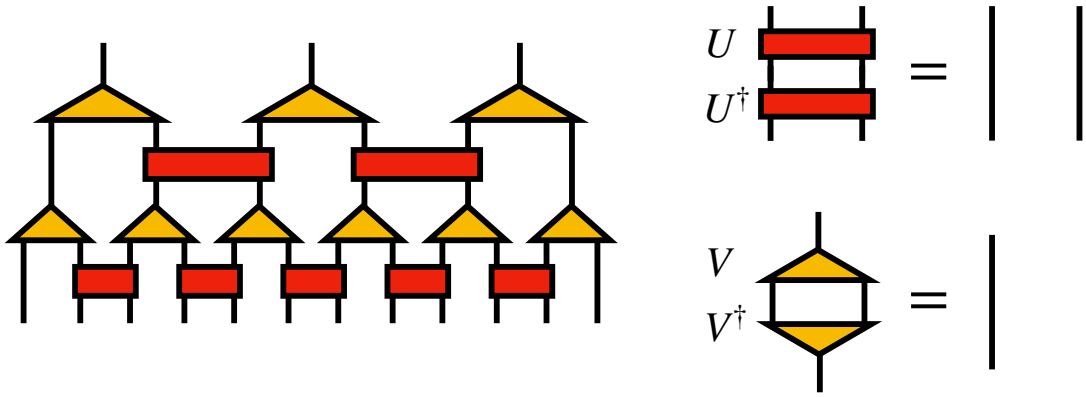


Figure 8.1: A tensor diagram depicting two layers of a MERA tensor network, showing the unitary condition obeyed by the disentanglers U and isometric condition obeyed by the isometries V.

Methods

The model we implement combines concepts from a number of disciplines. As such, we detail how each discipline influences the construction of the model.

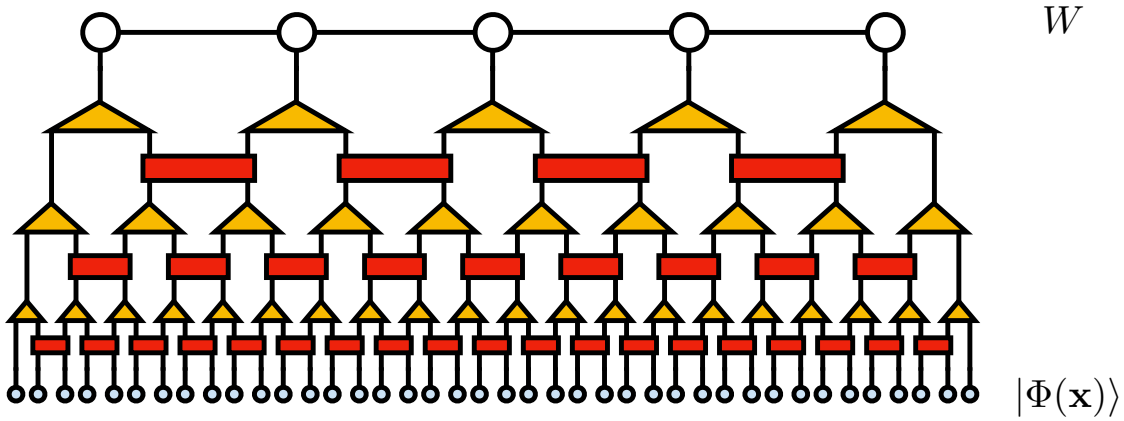


Figure 8.2: The tensor diagram for the model we use for classification and regression, showing the case of three MERA layers. Each input data $\mathbf{x}$ is first mapped into a MPS $|\Phi(\mathbf{x})\rangle$, then acted on by MERA layers approximating Daub4 wavelet transformations. At the top layer, the trainable parameters of the model W are decomposed as an MPS. Because all tensor indices are contracted, the output of the model is a scalar.

Classifiers As Tensor Networks

First, from a machine-learning perspective, we recognize that the class of model functions we seek to optimize have the general form

$$f_W(\mathbf{x}) = W \cdot \Phi(\mathbf{x}), \tag{8.1}$$

where W is the weight vector which enters linearly. The function Φ is the feature map. This map is generally nonlinear and maps the input $\mathbf{x}$ to a feature vector $\Phi(\mathbf{x})$. Model functions of this class are the same starting point for tasks which use kernel learning and support vector machine approaches for machine learning. Unlike these approaches, we will optimize over the weights W without introducing any dual parameters as in the kernel trick. To do this kind of optimization, we represent the weights W in a compressed form as a tensor network.

The feature map we implement takes the form

$$\Phi^{s_1 s_2 \cdots s_N}(\mathbf{x}) = \phi^{s_1}(x_1)\,\phi^{s_2}(x_2)\cdots\phi^{s_N}(x_N), \tag{8.2}$$

where N is the dimension of the input vectors $\mathbf{x}$, and each index s_i runs over d values. The *local feature maps* ϕ thus map each input component x_i to a d-dimensional vector. The resulting feature map Φ is a map from $\mathbb{R}^N$ to a rank-1 tensor of order N, whose indices are all of dimension d. Informally, one may say that Φ is a product of vector-valued functions of each of the input components.

As is the case in most machine learning tasks, the goal is to find a suitable weight vector W^* which minimizes an appropriate cost function between the model output $f_{W^*}(\mathbf{x}_j)$ and the expected output $\mathbf{y}_j$ over the observed data. The cost function we will use is the quadratic cost

$$C(W) = \frac{1}{2n}\sum_{j=1}^{n} ||f_W(\mathbf{x}_j) - \mathbf{y}_j||^2 + \lambda ||W||^2, \tag{8.3}$$

where the summation is over n training examples and λ is an empirical regularization parameter. Because of the quadratic form of this function, optimization can be conveniently carried out by efficient methods such as the conjugate gradient algorithm.

Putting these machine learning elements into the language of tensor networks, we map each input data element x_i to the vector $|\phi(x_i)\rangle = |0\rangle + x_i|1\rangle$, where utilize the physics braket notation such that $|v\rangle$ is a vector labeled v. We have also defined

$$|0\rangle = (1\ 0)^T \tag{8.4}$$

$$|1\rangle = (0\ 1)^T \tag{8.5}$$

The feature map applied to each data sample is taken to be the tensor product

$$|\Phi(\mathbf{x})\rangle = |\phi(x_1)\rangle \otimes |\phi(x_2)\rangle \otimes \ldots \otimes |\phi(x_N)\rangle\,, \tag{8.6}$$

where the $\otimes$ symbol is often omitted in practice when using ket $|\ \rangle$ notation. This input tensor can be thought of as an MPS of bond dimension 1.

Coarse Graining with Wavelet Transformations

With the machine learning task now being well defined and encoded into an MPS, we turn our attention to the implementation of a coarse-graining procedure on the vector space of input data. The first important task in accomplishing this is defining a coarse-graining transformation function. In signal processing, the Fourier transform is a ubiquitous transformation which can be used to take an input signal from the time domain to the frequency domain. However, in the cases where an input signal is not stationary, a more appropriate analysis can be done in both the time and the frequency domain, by utilizing wavelet transformations [106].

The continuous wavelet transformation of an input signal $x(t)$ is given by

$$X(\tau,s) = \frac{1}{\sqrt{|s|}} \int x(t)\, \bar{\psi}\left(\frac{t-\tau}{s}\right) dt, \tag{8.7}$$

where ψ is the *mother wavelet* which characterizes the transformation, τ is the translation parameter, and s is the scale parameter.

For discrete signals taken over constant time intervals, the mother wavelet becomes a vector whose inner product is taken with a subset of the signal x. The integral becomes a summation over the elements within that subset; neighboring subsets can generally overlap.

The two types of discrete wavelet transformations we will use below are Haar transformation

$$h_i^{(2)} = \frac{x_{2i} + x_{2i+1}}{\sqrt{2}} \tag{8.8}$$

for $i = 0$ to $i = N/2 - 1$ and the Daubechies-4 (Daub4) transformation

$$d_i^{(4)} = \sum_{j=0}^{3} x_{2i+j} D_j, \tag{8.9}$$

where the coefficients D_j are the elements of the vector

$$D = \left(\frac{1+\sqrt{3}}{4\sqrt{2}}, \frac{3+\sqrt{3}}{4\sqrt{2}}, \frac{3-\sqrt{3}}{4\sqrt{2}}, \frac{1-\sqrt{3}}{4\sqrt{2}} \right) . \tag{8.10}$$

For both of these discrete transformations the translation parameter is $\tau = 2$. The summation is over two elements for the Haar transformation and four elements for the Daub4 transformation. These discrete wavelet transformations of an input signal x can be used to average and rescale the initial signal from size N to size $N/2$, while preserving local information in both the time and frequency domains. The information that is preserved is that associated with the part of the signal which varies more smoothly with the time index i.

Wavelets as a MERA

These re-scaling transformations are embedded in the multi-scale entanglement renormalization ansatz (MERA). This is a tensor network whose geometry and structure implement multiple coarse graining transformations on a chain of input variables. In physics terminology, this process is known as renormalization [107]. The MERA architecture includes *disentangler* tensors which span across subtrees of the network [108, 109] —these are the four-index tensors U shown in

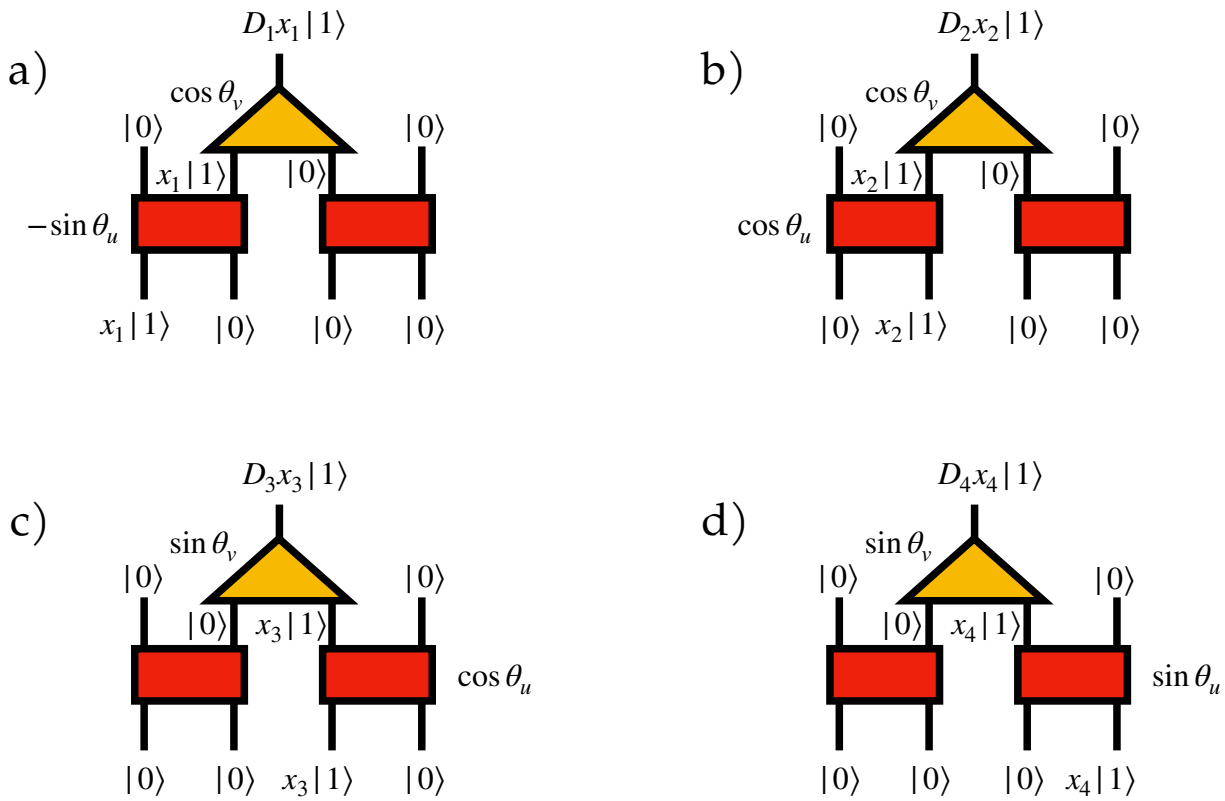


Figure 8.3: (a–d) the decomposition of each transformation element D_{1-4} into the unitaries and isometries of the wavelet MERA. The complete Daub4 wavelet transformation over data elements x_{1-4} is given by the contraction over connected indices. Each transformation is parameterized by θ_U and θ_V.

Fig. 8.1. These tensors are constrained to always be unitary, such that $U^\dagger U = UU^\dagger = 1$. Likewise, the MERA includes tree tensors, or *isometry* tensors V, which are constrained to obey an isometric condition $V^\dagger V = 1$ (yet $VV^\dagger \neq 1$). These conditions are depicted in diagrammatic form in Fig. 8.1. Because of these constraints, computations of the scalar product between two MERAs with N sites (i.e., the norm, expectation values, and correlation functions) can be accomplished within a polynomial number of steps. Here, we constrain the disentanglers and isometries further by

parameterizing them as

$$U = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_U & \sin\theta_U & 0 \\ 0 & -\sin\theta_U & \cos\theta_U & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$V = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \sin\theta_V & \cos\theta_V & 0 \end{pmatrix}.$$

This is a sufficient parameterization of MERA layers to approximately compute wavelet coarse graining transformations of input data.

To explicitly encode the wavelet transformations into the disentanglers and isometries of a MERA, we first decompose each of the wavelet coefficients in the set $\{D_i\}$ given in Eq. 8.10 into two sequentially applied transformations. If we consider the term $x_i|1\rangle$ in each local feature vector $|\phi(x_i)\rangle$ of Eq. 8.6 as a "particle" whose state has a coefficient given by the input component x_i, we can trace the path of this particle through the MERA as shown in Fig. 8.2, assigning appropriate transformations to x_i as it propagates through the tensors. Following this construction, one can work out the result of applying the MERA layer to a patch of four adjacent input tensors, whose

dependence on input components (x_1, x_2, x_3, x_4) is to leading order:

$$
\begin{aligned}
&|\phi(x_1)\rangle\,|\phi(x_2)\rangle\,|\phi(x_3)\rangle\,|\phi(x_4)\rangle = \\
&\quad = (|0\rangle + x_1\,|1\rangle)\,(|0\rangle + x_2\,|1\rangle)\,\cdots \\
&\quad = |0\rangle\,|0\rangle\,|0\rangle\,|0\rangle \\
&\qquad + x_1\,|1\rangle\,|0\rangle\,|0\rangle\,|0\rangle \\
&\qquad + x_2\,|0\rangle\,|1\rangle\,|0\rangle\,|0\rangle \\
&\qquad + x_3\,|0\rangle\,|0\rangle\,|1\rangle\,|0\rangle \\
&\qquad + x_4\,|0\rangle\,|0\rangle\,|0\rangle\,|1\rangle + \ldots,
\end{aligned}
\tag{8.11}
$$

where the omitted terms are higher order in components of $\mathbf{x}$, such as $x_2 x_4\,|0\rangle\,|1\rangle\,|0\rangle\,|1\rangle$.

From the conditions shown in Fig. 8.2, it follows that the result of acting with the MERA layer on this patch of inputs is, to linear order in the components of $\mathbf{x}$, an output vector

$$
(D_1 x_1 + D_2 x_2 + D_3 x_3 + D_4 x_4)\,|1\rangle\,, \tag{8.12}
$$

where now $|1\rangle$ labels the second basis vector of the vector space defined by the tensor indices along the top of the MERA layer, and the coefficients D_i are related to the parameters in the MERA factor tensors as

$$D_1 = -\sin\theta_U \cos\theta_V \tag{8.13}$$

$$D_2 = \cos\theta_U \cos\theta_V \tag{8.14}$$

$$D_3 = \cos\theta_U \sin\theta_V \tag{8.15}$$

$$D_4 = \sin\theta_U \sin\theta_V \ . \tag{8.16}$$

With the D_i chosen to be the Daub4 wavelet coefficients Eq. (8.10), this system of equations is easily solved by setting $\theta_U = \pi/6$ and $\theta_V = \pi/12$. With this choice, the wavelets are successfully encoded into the unitaries and isometries of the MERA. The full TN construction of this model is given in Fig. 8.2.

Training

Before applying this model to various machine learning tasks, we describe the details of the training procedure for our model. After each input training data sample $|\Phi(\mathbf{x}_i)\rangle$ has been coarse grained through the wavelet-MERA from size N to $N' = N/2^L$ where L is the number of layers used, we compute the (scalar) output of the model by an inner product with the tensor of weights W at the topmost scale. We choose this weight tensor to be represented by an MPS,

$$W^{s_1 s_2 s_3 \ldots s_N} = \sum_{\{a\}} A^{s_1}_{a_1} A^{s_2}_{a_1 a_2} A^{s_3}_{a_2 a_3} \cdots A^{s_N}_{a_{N-1}}. \tag{8.17}$$

We optimize the cost function in Eq. 9.8 by sweeping back and forth through the tensors of the above MPS, updating each tensor in a DMRG-like fashion [110].

The prominent feature of the optimization is a local update to W by optimizing the MPS tensors at

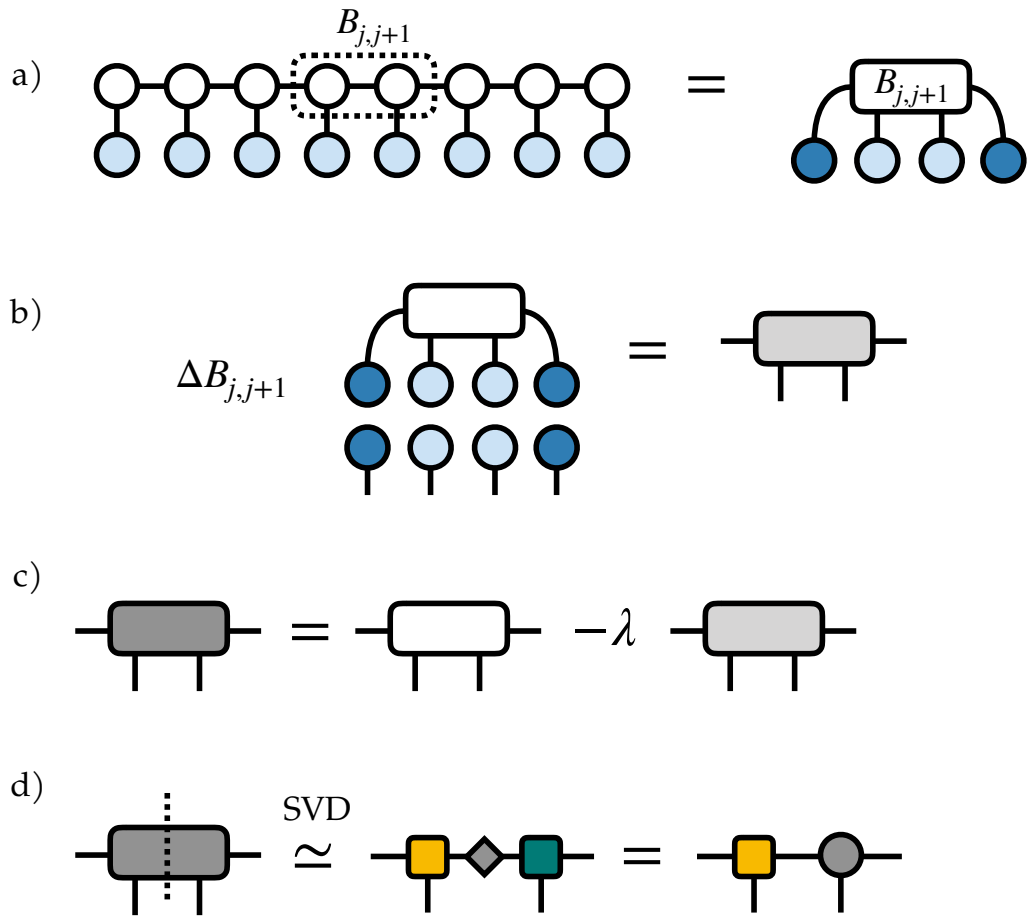


Figure 8.4: A tensor digram depicting the local update of $W^{(MPS)}$ is as follows: a) select a pair of neighboring $W^{(MPS)}$ tensors as B_{jj+1}, b) compute the gradient of the cost over all training examples, c) update B_{jj+1}, and d) decompose back into two MPS tensors using a truncated SVD. This is done for each pair in the MPS, sweeping back and forth until the cost is minimized.

sites j and $j+1$ together. This is accomplished by constructing

$$B^{s_j s_{j+1}}_{\alpha_{j-1} l \alpha_{j+1}} = \sum_{\alpha_j} A^{s_j}_{\alpha_{j-1}\alpha_j} A^{s_{j+1}}_{\alpha_j \alpha_{j+1}} \tag{8.18}$$

Figure 8.4(a) shows how the output of the model can be computed by first contracting all of the coarse-grained data tensors with the MPS tensors not currently being optimized, then with the

bond tensor B. The gradient of the cost function is proportional to the tensor ΔB which is shown in Fig. 8.4(b). The ΔB tensor can be used to improve B as shown in Fig. 8.4(c). In practice, we actually use the conjugate gradient algorithm to perform the optimization, but the first step of this algorithm is identical to Fig. 8.4(b–c) for the proper choice of λ. Finally, to proceed to the next step the MPS form of W must be restored to ensure efficiency. This can be done by treating B as a matrix and computing its SVD as shown in Fig. 8.4(d). Truncating the smallest singular values of the SVD gives an adaptive way to automatically adjust the bond dimension of the weight MPS during training.

Fine Scale Projection

As a final and unique feature of the model, we introduce an additional step which provides the possibility of further optimizing the weight parameters for our classifier. Once training over the weight MPS has been completed at the topmost scale, the optimized weights can be projected back through the MERA consisting of N_{d4} layers to the previous scale defined by $N_{d4} - 1$ layers.

This is done by first applying the conjugate of the isometries of the topmost MERA layer to each MPS tensor representing W, thereby doubling the number of sites. Next, we apply the conjugate of the unitary disentangler transformations to return to the basis defining the previous scale. Finally, to restore the MPS form of the transformed weights, we use an SVD factorization to split each tensor so that the factors each carry one site or 'feature' index. All of these steps are shown in Fig. 8.5(b).

The projection of these trained weights onto to the new finer scale serves as an initialization W' for layer $N_{d4} - 1$. At this step, the coarse-grained data previously stored at this finer scale are retrieved (having been previously saved in memory), and a new round of training is performed for W'. The intent is for this projected MPS W' to provide a better initialization for the optimization

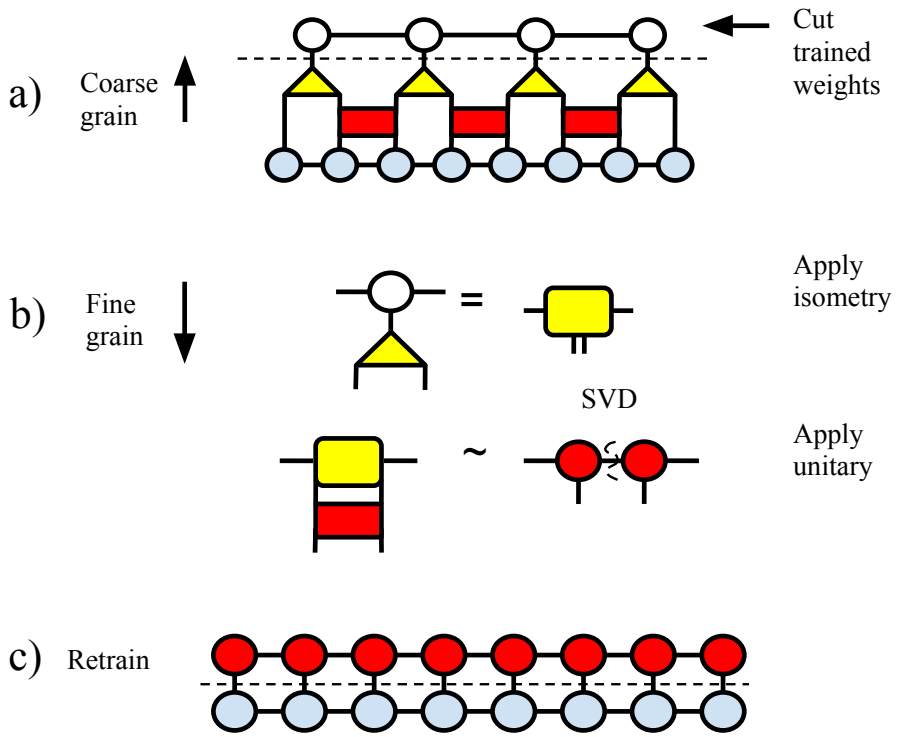


Figure 8.5: Tensor diagrams detailing the steps of our machine learning wavelet MERA alorithm. In a) the training sets are embedded into an MPS (light blue) and coarse grained through the wavelet MERA for training of the weight MPS (white) at the top most layer. b) Once training is complete, the weight MPS is fine grained through the wavelet MERA by applying isometries and unitaries. c) Training is carried out on the training set at the finer scale

than could have otherwise been obtained directly at finer scales. Table 8 enumerates each step in our algorithm from initial coarse-graining to training and fine scale projection.

Table 8.1: Wavelet MERA training algorithm

step 1	Map each training sample $\mathbf{x}_i$ into $\lvert\Phi(\mathbf{x}_i)\rangle$, Eq. 8.6
step 2	Coarse grain each $\lvert\Phi(\mathbf{x}_i)\rangle$ through the wavelet MERA
step 3	Train W at the current layer
step 4	Project the trained W to the finer scale, Fig. 8.5
step 5	Repeat steps 3 and 4 as desired

Results

Audio File Binary Classification

Applying our model to the task of binary classification, we used the DCASE audio classification set which consists of 15 batches (one batch per label), each with 234 ten second audio clips for training [111]. Each audio clip is a vector of 441,000 samples, which we embedded into a vector of 2^{19} elements by padding with zeros. We constrained the problem to focus on binary classification by specifically distinguishing between "bus" and "beach" environment audio clips.

Each data set for the selected labels was coarse grained through N_{h2} Haar transformation before encoding the data into our Daub4 wavelet MERA.

We trained of the classifier at the top layer of the MERA for a varying number of $N_{d4}+1$ Daub4 transformations, and subsequently projected the weights to one previous finer scale (scale N_{d4}). The percentage of correctly labeled examples after training at each scale is shown in Fig. 8.6. Each two point segment corresponds to training at the coarse scale (the right-most point in a segment), followed by training at the finer scale (the left-most point in a segment). The accuracy of the model decreases with the number of wavelet transformations. But it is also apparent that training the weights after $N_{d4}+1$ layers produces a better initialization and optimization for training at

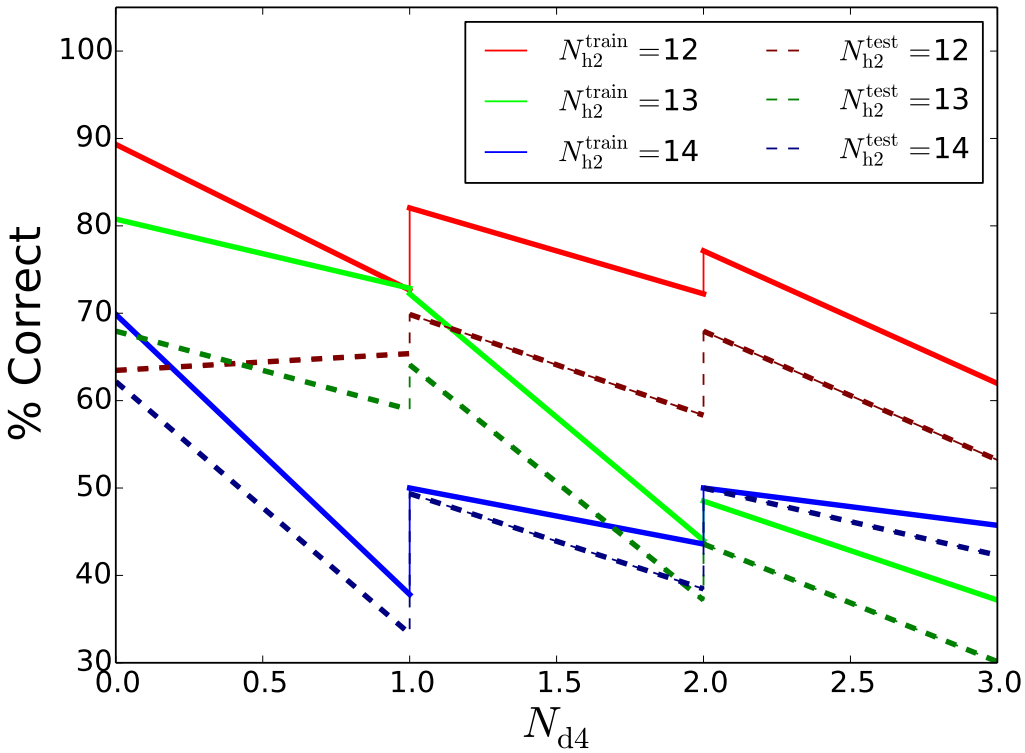


Figure 8.6: Results showing the percentage of correctly labeled samples (training set as solid lines and test set as dashed lines) for the audio experiment. The results were collected after training W over input data coarse grained through $N_{d4}+1$ Daub4 wavelet transformations and then again after being projected back and trained at N_{d4} layers. Each data set was initially embedded into a vector of 2^{19} elements and coarse grained N_{h2} Haar wavelet transformations before being embedded in the Daub4 wavelet MERA. The optimization was carried over 5 sweeps, keeping singular values above $\Delta = 1 \times 10^{-14}$.

the N_{d4} scale. Additionally, as can be noted by the closeness in accuracy between the training and testing sets for $N_{h2} = 14$, versus for $N_{h2} = 12$, the generalization of the model improves with the number of wavelet layers applied. Optimization was carried over five sweeps, with the bond dimension of the weight MPS adaptively selected by keeping singular values above the threshold $\Delta = 1 \times 10^{-14}$.

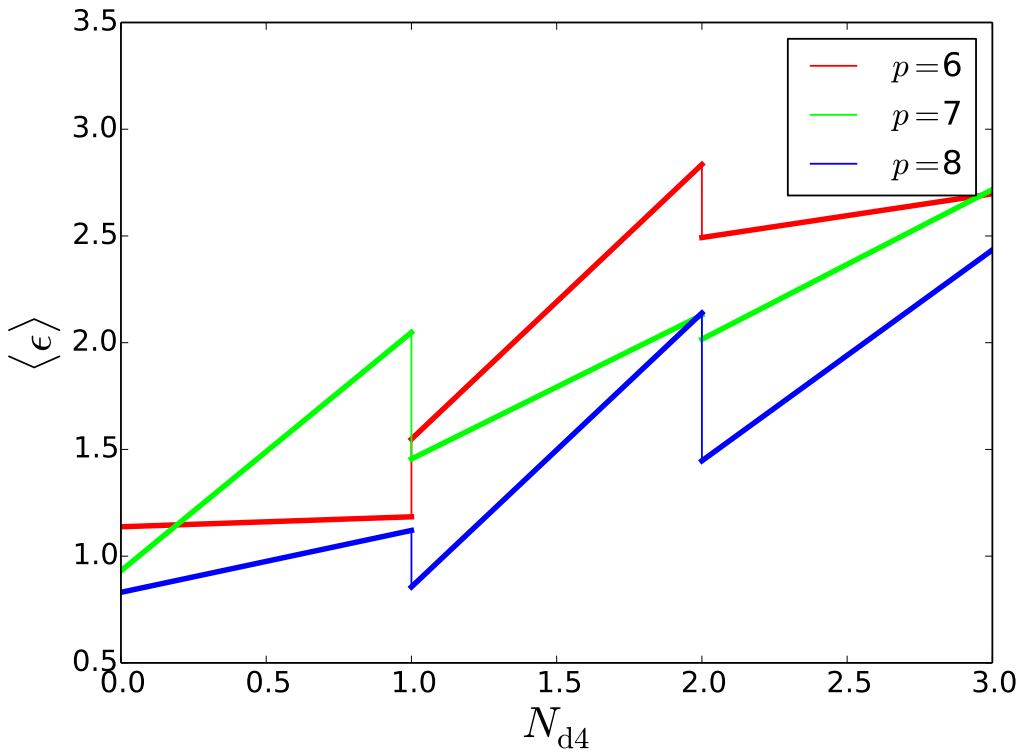


Figure 8.7: Results for the average absolute deviation $\langle \varepsilon \rangle$ (lower is better) of the predicted output and target output for the temperature experiment. Restuls were gathered after training $W^{(MPS)}$ over input data coarse grained through $N_{d4}+1$ wavelet layers and projected back to N_{d4} layers. Each data set was initially constructed by selecting p data points within the set N_{fit}. The optimization was carried over 40 sweeps, keeping singular values above $\Delta = 1 \times 10^{-9}$.

Applying our model to regression, we took a single data file of the average daily temperatures of the Fisher River recorded from January 1, 1988 to December 31, 1991. The input data sets were constructed from this file in the following manner. By labeling each temperature x_i for $0 < i < 1462$, the fitting interval N_{fit} was taken as all the temperatures $\{x_i | i \in [731, 1461]\}$. A single training example was constructed by selecting a contiguous block of p temperatures from N_{fit} as input for the MERA. The temperature immediately following this p block was assigned as the label

for that training example. By shifting the starting index of the p block of temperatures, multiple examples could be constructed. In this way, the regression task was recast as a classification task over a continuous label. Because nearly every example contained a unique label, we used the average absolute difference of the interpolated label and the actual label, $\langle \varepsilon \rangle$, as a measure of the accuracy of the model. The training phase of this data was carried out for forty sweeps, but with a singular value threshold set to $\Delta = 1 \times 10^{-9}$. In Fig. 8.7, similar to the audio classification task, $\langle \varepsilon \rangle$ is given for input data coarse-grained through $N_{d4} + 1$, with the weights trained at this scale, and then projected onto the finer scale N_{d4} for further training. Each two-segment section is representative of training at $N_{d4} + 1$ layers on the right-most point and training at N_{d4} layers on the left-most point. Again, we note a general decrease in accuracy with the number of wavelet transformations, but an improvement in accuracy when the weights are initialized by optimization through $N_{d4} + 1$ wavelet layers, as compared to directly randomly initializing at N_{d4} layers.

Conclusions

MERA are a family of tensor network factorizations which process information in a hierarchical way and preserve computational advantages associated with tree-tensor networks, such as efficient marginalization, yet can be more expressive and powerful because of extra "disentangler" layers which mix branches of the tree. Here we have proposed an architecture for machine learning which uses MERA layers which approximate wavelet scaling functions as a preprocessing step, and explored the advantages of this choice. Because of the presence of disentanglers, MERA are able to approximate non-trivial families of wavelets with overlapping support, such as Daubechies-4 wavelets.

Because our ML model is encoded into a MERA, we find that it exhibits an interesting reversibility property: the trainable parameters of the model can be projected to a finer scale of resolution, while

preserving the output of the model. This allowed our procedure to initialize more expressive models by initially training the model with fewer parameters. This initialization step showed notable improvement in the accuracy of the model as compared to the direct initialization of the weights at any given scale, for a fixed number of optimization sweeps at the finer scale.

By noting the dependence of the accuracy of our model on the number of wavelet layers, we deduce that the input data can exhibit correlations at length scales that can be lost through the wavelet transformations. It is therefore important to tailor the number of wavelets and initial size to the specific data set being analyzed in order to maintain a desirable accuracy. Our setup gives an affordable and adaptive way to strike a balance between the efficiency and generalization gains obtained by coarse-graining versus model expressivity by trading off one for the other through the fine-graining procedure.

Among techniques widely used in machine learning, our model architecture most closely resembles a CNN [112]. In both architectures, data is initially processed through a set of layers which mix information in a locality-preserving way. Pooling layers commonly used in CNNs closely resemble the isometry maps in a MERA, which act as a coarse-graining transformation. The connection to CNNs suggests one future direction of investigation for our model being the inclusion of adjustable MERA layer parameters (i.e., training the wavelet MERA transformation angles as well as the weight MPS). More ambitiously, instead of just training the parameters in a given MERA layer, one can envision computing this layer from a set of weights at a given scale through a controlled factorization procedure.

We conclude that our algorithm provides an interesting platform for classification and regression, with unique capabilities. Further work is needed to improve the model accuracy for the classification of continuous labels (i.e., regression). It is also worth investigating the effect that different wavelet transformations may have on the model; determining how much is gained by introduc-

ing optimizable parameters into the coarse-graining layers; and investigating adaptive learning schemes for the sizes of indices in the MERA layers.

CHAPTER 9: MACHINE LEARNING REGRESSION FOR OPERATOR DYNAMICS

The work contained in this chapter is currently being prepared for publication.

Introduction and Motivation

The final project included in this dissertation is separate from the previous projects in that its primary focus is the application of a machine learning architecture to a QMB problem. Specifically, we consider the determination of the long-time dynamics of observables in QMB systems. There has been much progress made in studying these dynamics for various specific systems of interest, such as the Ising chain with a quenched transverse field [113], or the Ohmic spin-boson model coupled to a harmonic non-Markovian environment [114]. However, these developments have focused on systems where symmetries and approximations can be exploited, analytic or exact diagonalization methods can be used, or MPS algorithms can be employed. Such approaches are either limited in their scope or quickly become computationally demanding, particularly for systems in more than one dimension. In modeling QMB systems, current advances in the construction and implementation of machine learning models and algorithms have offered new insights where traditional techniques have failed [115, 116, 117]. Motivated by these successes, we investigate the advantages machine learning can provide when modeling operator dynamics for QMB systems.

Previous work utilizing machine learning techniques in QMB systems focuses on the use of Restricted Boltzmann Machines (RBMs) as generative models of quantum states [118, 119, 120].

These models are energy-based, with a cost energy functional given by

$$E(v,h) = -\sum_i a_i v_i - \sum_j b_j h_j - \sum_{i,j} v_i W_{ij} h_j. \tag{9.1}$$

Here $v = \{v_i\}$ and $h = \{h_j\}$ are the visible and hidden layers of neurons in the RBM network, respectively. Each visible (hidden) neuron has an associated bias $a_i(b_j)$ and is fully connected to the hidden (visible) layer by the weight matrix W [121]. With this model, the distribution of states of for a quantum system $\Psi_{\mathrm{RBM}}(v)$ is given by

$$\Psi_{\mathrm{RBM}}(v) = \sum_h e^{-E(v,h)} = \prod_i e^{a_i v_i} \prod_j 1 + e^{b_j + \sum_i v_i W_{ij}}. \tag{9.2}$$

.

By sampling this distribution, dynamical properties can be (and have been) determined by following various variational approaches over time [122, 123, 124]. However, in spite of much success, this model still faces similar challenges to MPS models in that accurately representing the system state becomes computationally demanding as the system sizes grow. This fault in the representation of the distribution eventually results in an inability to properly evaluate operator expectation values at sufficiently long times.

However, full knowledge of the system state at every instant in time is unnecessary when calculating expectation values. This can be seen by considering the Heisenberg picture of quantum mechanics. Here, when considering a system governed by Hamiltonian $\mathscr{H}$, it is mandated that the time dependence originally contained within the system state $|\Psi\rangle$ be transferred to the operators acting on that system. Rather than propagating the state forward in time, we propagate operators forward in time. Within this framework, the focus shifts from determining the distribution of states to the determining the specific dynamics of an applied operator $\mathscr{O}(t)$. The dynamics of the operator

is governed by the following equation:

$$\frac{d\mathscr{O}(t)}{dt} = i[\mathscr{O}(t), \mathscr{H}] + \frac{\partial \mathscr{O}(t)}{\partial t}. \tag{9.3}$$

Assuming this picture of quantum mechanics, rather than iteratively generate the RBM state over each time step of the evolution, we implement a direct time evolution for the expectation values of operators by using a multi-layer perceptron (MLP) as a tool for performing a regression over a small training set of initially exact expectation values. A schematic for this machine learning is given in Fig. 2.2. We will see that this model can extend the operator dynamics to sufficiently long times with few computational resources.

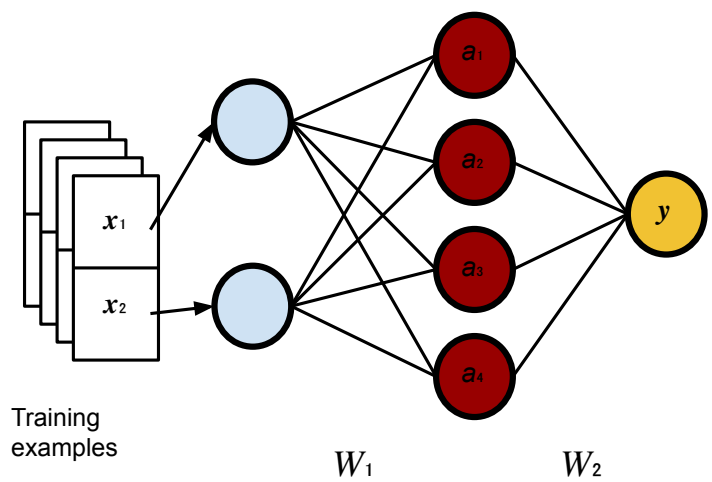


Figure 9.1: A schematic of MLP receiving input vectors having two elements x_i, applying a single layer of four perceptrons with weights a_i, and outputting a single value y.

Methods

Time Evolving Block Decimation

Before introducing the MLP regression algorithm, we provide a short review of the TEBD algorithm. The TEBD algorithm facilitates the time evolution (real or imaginary) of one-dimensional quantum systems under local Hamiltonians [68]. As such, it is naturally expressed within the framework of MPS. The time evolution is accomplished by generating and repeatedly applying matrix product operator (MPO) Suzuki-Trotter expansions of the time evolution operator $\exp(-i\mathscr{H}T)$ [125] to an initial MPS $|\Psi_{\text{MPS}}\rangle$. These expansions can be done up to arbitrary order, though in practice this can be difficult. For a local Hamiltonian

$$\mathscr{H} = \sum_{i=1}^{N-1} H_{i,i+1}, \tag{9.4}$$

the second order Suzuki-Trotter expansion of $\exp(-i\mathscr{H}T)$ for small time-step $\delta > 0$ is given as

$$e^{-iT\mathscr{H}} \approx \left[\left(e^{-\frac{i\delta}{2}H_{1,2}}e^{-\frac{i\delta}{2}H_{3,4}}\cdots\right)\left(e^{-i\delta H_{2,3}}e^{-i\delta H_{4,5}}\cdots\right)\left(e^{-\frac{i\delta}{2}H_{1,2}}e^{-\frac{i\delta}{2}H_{3,4}}\cdots\right)\right]^{\frac{T}{\delta}}. \tag{9.5}$$

After applying a single operator for a time step δ, the TEBD algorithm mandates that the MPS be brought to canonical form (i.e., orthonormalizing the indices) [126]. This process involves $O(poly(N)poly(D))$ parameters (where N is the number of sites and D the maximum internal bond dimension of the MPS). Because D grows exponentially with both the system size and the evolution time, this computational cost quickly becomes intractable for large systems and long times.

When all of the operators have been applied, the final state of the system is obtained as

$$|\Psi(t=T)\rangle = e^{-iT\mathscr{H}}\,|\Psi(t=0)\rangle\,. \tag{9.6}$$

Typically, there are two sources of error in the TEBD framework. The first comes from the truncation of the MPS bond dimensions during the orthonormalization process. The other source of error arises during the Suzuki-Trotter expansion, which for our purposes is taken to second-order. In this case, the error ε for an evolution time T is given as

$$\varepsilon = (T\delta^2)^2. \tag{9.7}$$

.

In this paper, we choose not to truncate the MPS in order maintain the accuracy of the approximation. This comes at the price of significantly increasing the computational cost. It is this cost which has proved to be a significant barrier in the determination of long-time operator dynamics.

MLP for Regression

In order to effectively model the evolution of operator expectation values, we construct the MLP in a manner conducive to regression rather than classification. Accomplishing this involves a few specifications about the input-output pairs $\{\mathbf{x}, y\}$. We treat an input vector $\mathbf{X}$ as being parameterized by time t, over a time interval $[0,\tau]$ so that each element $X_i \in \mathbf{X}$ is labeled by a coordinate t_i. The total time τ, is partitioned into m discrete time intervals $\{t_i | 0 < i < m\}$. From $\mathbf{X}$, each input-output pair is constructed as follows. Starting from the first element in X corresponding to time $t_0 = 0$, we select a contiguous block of p elements from X to form an input vector $\mathbf{x} = \{X_0, X_1, ..., X_p\}$. We call this block our *training window*. The corresponding label for this window is selected as the

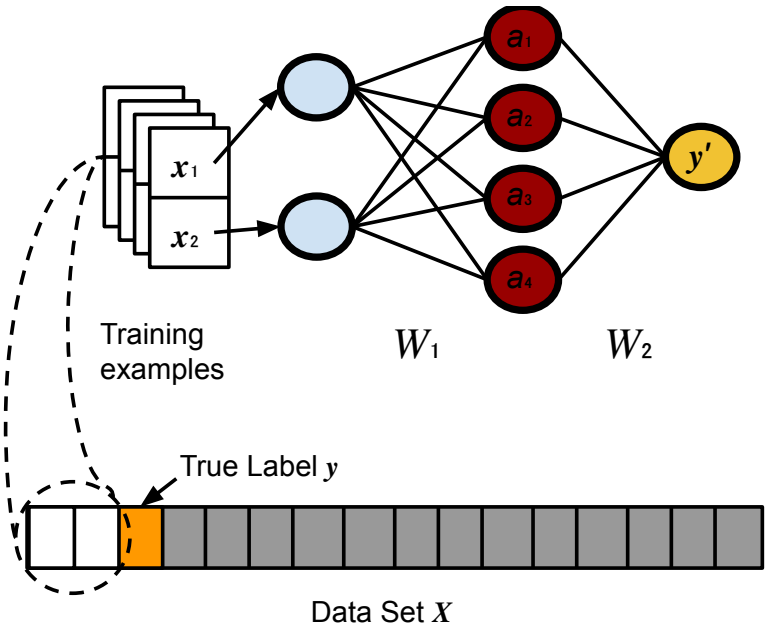


Figure 9.2: A schematic representation of a MLP used for regression. First the data set X is partitioned into examples for training, with the true label being y being the data point x_{p+1}, where p is the size of a training window. These training sets are passed to a MLP, where the cost of output y' is measured against y and optimized the weights a_i.

element X_{p+1}. To construct multiple input examples for training, we shift the starting position of the training window throughout $\mathbf{X}$ until the desired number of examples is achieved. A diagram of this initialization procedure is given in Fig. 9.2.

In addition to constructing the input-output pairs in the aforementioned manner, we choose to define our activation functions by the linear unit, $f = x$. This activation allows us to effectively propagate all of the input information through the network. This is in contrast to the more commonly used rectified linear unit (ReLU), $f = \max(0, x)$, which, depending on the values selected for the weights, can suppress some information propagation through the network by eliminating all negative values. The ReLU activation is useful when the MLP is used for classification over a discrete set of positive valued labels. However, we select the linear activation because our output

values are continuous and include values less than zero. We anticipate that the structure of this input data will introduce a greater dependence of our output on the given input values.

Training

As a final note, to train our MLP, we utilize the cost function

$$C(y'_n, y_n) = \frac{1}{N}\sum_{n}^{N} |y'_n - y_n|, \tag{9.8}$$

where y'_n is the current guessed output of the MLP model, y_n is the known correct output, and N is the number of training examples. This cost is optimized by a stochastic gradient descent.

Results

We test our MLP regression by evaluating operator expectation values over two model systems, comparing them to second-order Trotter-Suzuki time-evolved MPS calculations. First, we determine values at the critical point for the one-dimensional Ising model in a transverse field with an evolution time of $\tau = 30$. Then we apply the MLP regression to the one-dimensional XXZ model for $\tau = 20$. All machine learning simulations were implemented using the Tensorflow Keras library [127].

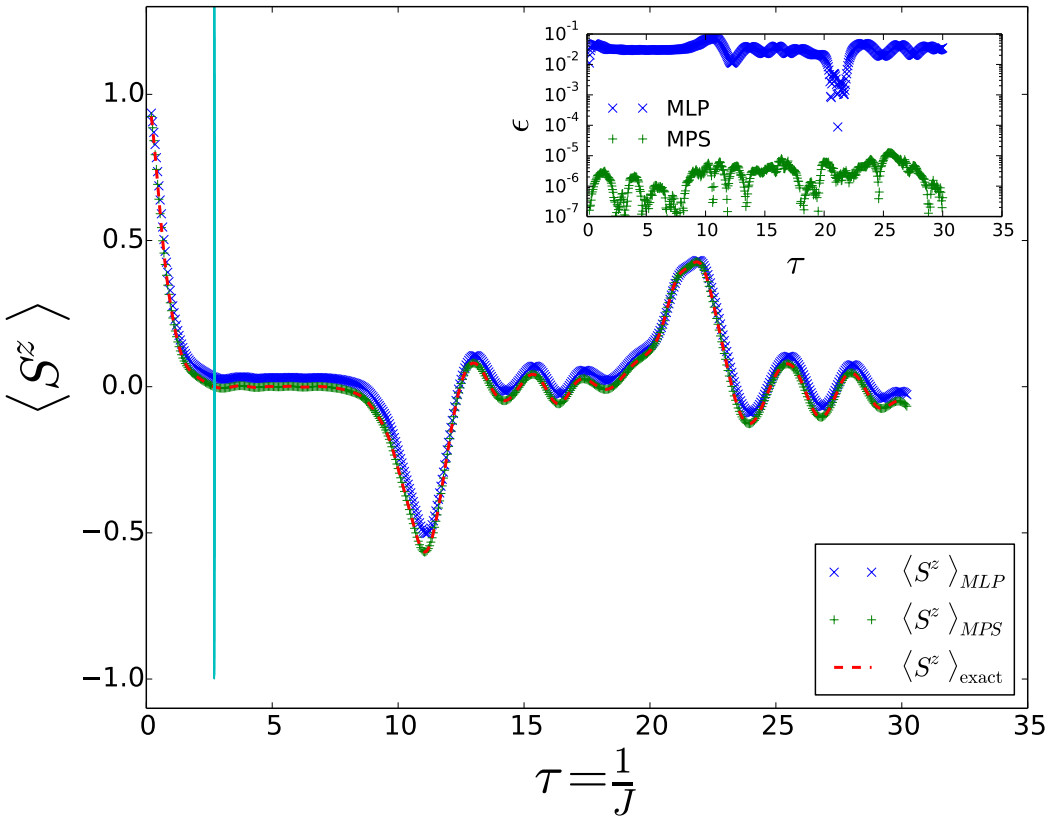


Figure 9.3: Results for the dynamics of the $\langle S^z \rangle$ operator measured for the ten-site one-dimensional Ising model with spin coupling $J = 1$ and transverse field strength $\Gamma = 1$. MLP calculations are compared with TEBD calculations and exact diagonalization results. The inset shows the absolute deviations of both the MLP and MPS calculations as compared to the exact results.

Ising Model

For n spins in a one-dimensional chain, the Ising model in the presence of a transverse field is given by the Hamiltonian

$$\mathscr{H} = -J \sum_{\langle i,j \rangle} S_i^z S_j^z - \Gamma \sum_i S_i^x, \tag{9.9}$$

where S^z is the longitudinal spin operator, Γ characterizes the coupling strength to the transverse field. The first sum is taken over neighboring pairs $\langle i, j \rangle$, while the second sum is over every site. Due to the non-commutability of terms in the Hamiltonian, in one dimension this model is known to have a quantum phase transition near $\frac{\Gamma}{J} = 1$. This phase transition takes the system from the ordered ferromagnetic state to the paramagnetic state [128].

We explore the dynamics of the average local spin operator $\langle S^z \rangle$ near this phase transition for a spin chain with $n = 10$ spins. To accomplish this, we first use MPS calculations to generate expectation values for time steps of $t_i = 0.05$. We split these time-ordered expectation values into subsets for training and testing the MLP. For our model, we select training windows of $p = 4$, giving us access to 995 input-output pairs. Of this, we only use 50 pairs for training. Our MLP architecture is optimized with the following parameters: one layer of 128 linear activated neurons, followed by a single layer with one linear activated neuron for output. Training is carried out by a stochastic gradient descent over the cost function given in Eq. 9.8. Figure 9.3 shows the results with $\Gamma = 1$ and $J = 1$. Within the training region, the MLP is trained until it has significant overlap with the MPS calculations. This overlap is seen to continue far past the region of training. Comparison with exact diagonalization results, as shown in the inset of Fig. 9.3, reveals that the MLP accuracy is marginally worse than the accuracy of the MPS calculations, but still on average within $|y' - y| < 10^{-3}$. The time to train the MLP was 51.312 seconds, while the time to predict the rest of the dynamics was 5.824 seconds. Comparatively, for this system size, exact diagonalization calculations take ≈ 30 seconds to complete.

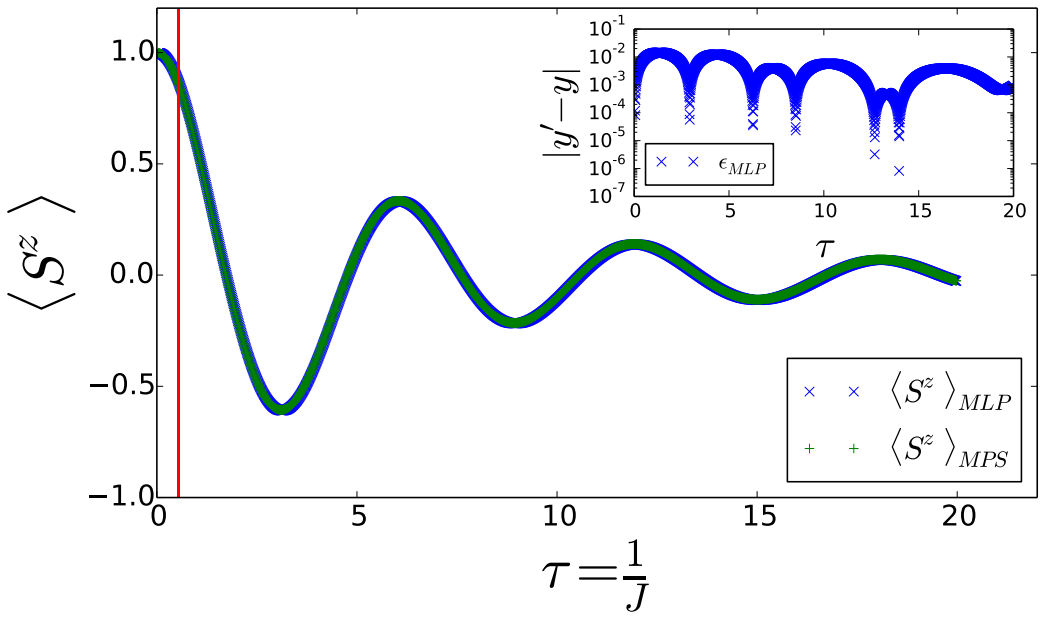


Figure 9.4: Results for the dynamics of the $\langle S^x \rangle$ operator measured for the twelve-site one-dimensional XXZ model with $J_\perp = 2$, $J_z = 1$, and $\Gamma = 1$. MLP calculations are compared with TEBD calculations. The inset shows the absolute deviation of between the MLP and TEBD.

XXZ Model

We test another ubiquitous spin system with our MLP regression; The XXZ model. The Hamiltonian governing the evolution of this system is given by

$$\mathscr{H} = -\sum_{\langle i,j \rangle} [J_\perp (S_i^x S_j^x + S_i^y S_j^y) + J_z S_i^z S_j^z] - \Gamma \sum_i S_i^x, \tag{9.10}$$

with $J_\perp$ and J_z being spin-spin couplings, and Γ being the strength of coupling to a transverse field. Similar to the Ising model above, the XXZ model exhibits a transition between the ferromagnetic and the paramagnetic phases. The order parameter governing this transition is given by the quantity

$\frac{J_z}{|J_\perp|}$ [129]. We choose to once again remain near the transition point, and set $J_\perp = 2$, $J_z = 1$, and $\Gamma = 1$, exploring the dynamics of the spin operator $\langle S^z \rangle$ for $n = 12$ spins.

We again select a training window of $p = 4$, producing 1995 input-output pairs. From these, we train over 100 pairs. The MLP is composed of a single layer of 256 linearly activated neurons, followed by an output layer with a single linearly activated neuron. This model is again trained used stochastic gradient descent. Comparing with the results taken from MPS calculations over time intervals $t_i = 0.01$, we see in Fig. 9.4 that the MLP regression agrees with the MPS and continues to do so deep into the testing regime. As shown in the inset of Fig. 9.4, the MLP on average differs consistently from the TEBD calculations by $|y' - y| < 10^{-3}$. The time to sufficiently train to the desired accuracy was 94.93 seconds, while the time to predict the rest of the dynamics was 10.92 seconds. Comparatively, at this system size, exact diagonalization calculations take ≈ 1800 seconds to complete. As the system size grows, it is evident that the MLP regression becomes a more useful tool.

Conclusions

By examining Eq. 9.3, we conclude that the learning procedure for our MLP regression optimizes an internal approximation of the commutator between any given operator $\mathscr{O}$ and the Hamiltonian $\mathscr{H}$. This internal approximation is generated within in a small parameter space and uses very few training examples. Additionally, this approximation demonstrates a robustness displayed in the way that MLP and MPS calculations remain in close agreement for simulations both close to and far away from the critical point.

Of the unique properties of this MLP regression, the most significant one is the decrease in computational resources used to generate long time dynamics. For TEBD calculations, long time

dynamics are obtained by determining the state of the system $|\Psi_{\text{MPS}}\rangle$ at every time step [68]. For N sites with maximal bond dimension D, this results in using $O(poly(N)poly(D))$ parameters for the optimization of $|\Psi_{\text{MPS}}\rangle$ [8]. Computations with this complexity can quickly become cumbersome for long times, particularly near critical points, when D begins to grow exponentially. However, in contrast to this, we have demonstrated that the MLP regression accurately extends MPS calculations with few computational resources, independent of the system size. Consider a training set having N_{train} examples constructed over training windows of size p. For a given value of p elements, the training phase of the MLP regression only has a computational cost $O(N_{\text{train}}p^2)$. After the training, prediction for later times only has a computational cost of $O(1)$. Clearly, the computational advantage is quite significant. By generating the first few expectation values with the MPS, the MLP regression is shown to be able to predict long time operator dynamics with only the addition of a relatively small number of compute cycles.

Though this computational advantage is significant, it is worth noting that the MLP regression can only extend the operator expectation values generated by the MPS. It cannot calculate operator dynamics without the presence of some initial expectation values. Further work must be done to explore machine learning architectures which can directly generate operator dynamics.

CHAPTER 10: SUMMARY

List of Publications

1. L. Zhang, J. Reyes, S. Kourtis, C. Chamon, E. R. Mucciolo, and A. E. Ruckenstein, *Non-Universal Entanglement Level Statistics in Projection-Driven Quantum Circuits*. Phys. Rev. B **101** 235104 (2020)

2. J. Reyes, D. Marinescu, and E. R. Mucciolo, *Simulation of Quantum Many-Body Systems on Amazon Cloud*. Comp. Phys. Comm. (under review)

3. J. Reyes, and E. M. Stoudenmire, *A Multi-Scale Tensor Network Architecture for Classification and Regression*. JMLR (under review)

4. J. Reyes and E. R. Mucciolo, *Quantum Circuit Measurements Enhanced by Entanglement Dilution* (in preparation)

5. J. Reyes, S. Dhara, and E. R. Mucciolo, *Machine Learning Regression for Operator Dynamics* (in preparation)

Conclusions and Outlook

Tensor networks are ubiquitous computational tools used to capture entanglement properties, correlations, and phase transitions in QMB systems. Their applicability extends far beyond this, however, spanning many disciplines including quantum information science (QIS) and ML. In this dissertation, I have developed or modified tensor network methods within the disciplines of QMB physics, QIS, and ML.

In my initial study, I optimized current tensor algorithms for use on the AWS EC2 cloud infrastructure. This was done by introducing a novel tensor contraction ordering protocol and a parallel partitioning scheme which minimized inter-node communication. This work demonstrated the feasibility of moving QMB calculations to the cloud.

After this, I utilized modified versions of the iterative-compression-decimation (ICD) algorithm to perform analysis on various classical and quantum computational models. Looking at classical boolean satisfiability problems, I discovered that the ICD possessed capabilities of computational complexity discrimination, arising during the compression steps of the algorithm. Moving to quantum circuits, I also discovered that quantum state preparation of the chaotic state can be achieved more efficiently by utilizing a novel tensor tiling protocol which I termed *entanglement dilution*.

Finishing studies of the ICD algorithm, I then modelled quantum circuits perturbed by random measurements (noise) using MPS and analyzed the final state of the system by measuring entanglement spectrum statistics. This analysis revealed a new intermediate phase as a function of the random measurement concentration p, existing between the area-law (Poisson) and volume-law phases (GUE).

After this, I utilized tensor network algorithms to develop novel machine learning architectures. Taking the MERA tensor network, and embedding it with wavelet transformations, I developed a unique ML model which was applied to binary classification and regression, demonstrating a high accuracy of classification and having built in back-propagation properties.

Finally, I improved the computational requirements necessary to determine long-time operator dynamics on MPS. This was accomplished by using a multi-layer perceptron (MLP) to perform a regression on the short-time operator dynamics initially generated by an MPS.

With all of these optimizations, applications, and investigations done over a diverse set of research

areas, it is evident that the path forward for research with tensor networks is extensive. Much work still must be done to enhance tensor network contraction protocols and algorithms, especially as new technologies are developed for super-computing clusters or cloud services. Investigations into the properties of quantum circuits must be continued with the goal of determining the limitations on quantum supremacy and elucidating the nature of transitions between regimes of computation. This endeavor is becoming particularly important as experimental realizations of quantum computation continue to become more of a reality. Finally, as machine learning continues to permeate every discipline of study, it will become increasingly necessary to utilize insights and principles gather from many disciplines, such as QMB physics, to provide both unique and optimal solutions to various problems of interest.

LIST OF REFERENCES

[1] P. Coleman. *Introduction to Many-Body Physics*. Cambridge University Press, 2015.

[2] P.W. Anderson. *The Theory of Superconductivity in the High T_c Cuprates*. Princeton University Press, 1995.

[3] D.A. Abanin, E. Altman, I. Bloch, and M. Serbyn. Many-body localization, thermalization, and entanglement, 2019. arXiv:1804.11065v2.

[4] X-G. Wen. Colloquium:zoo of quantum topological phases of matter. *Rev. Mod. Phys.*, 89:041004, 2017.

[5] H. Bruus, and K. Flensberg. *Many-body Quantum Theory in Condensed Matter Physics*. Oxford Graduate Texts, Copenhagen, 2002.

[6] J.J. Sakurai. *Modern Quantum Mechanics*. Addison-Wesley, 1994.

[7] J.W. Negele, and H. Orland. *Quantum Many Particle Systems*. CRC Press, 1988.

[8] R. Orus. A practical introduction to tensor network states: Matrix product states and projected entangled pair states. *Annals of Physics*, **349**:117–158, 2014.

[9] F. Verstraete, M.M. Wolf, D. Perez-Garcia, and J.I. Cirac. Criticality, the area-law, and the computational power of peps. *Phys. Rev. Lett.*, 96:220601, 2006.

[10] H.C. Jiang, Z.Y. Weng, and T. Xiang. Accurate determination of tensor network state of quantum lattice models in two dimensions. *Phys. Rev. Lett.*, 101:090603, 2008.

[11] F. Verstraete, and J. I. Cirac. Renormalization algorithms for quantum many-body systems in two and higher dimensions, 2004. arXiv:cond-mat/0407066.

[12] A. Smola, and S.V.N. Vishwanathan. *Introduction to Machine Learning*. Cambridge University Press, 2008.

[13] K. Gurney. *Introduction to Neural Networks*. UCL Press, 1997.

[14] T. Epelbaum. Deep learning:technical introduction, 2017. arXiv:1709.01412.

[15] G. Hinton. *A Practical Guide to Training Restricted Boltzmann Machines*. Springer, 2010.

[16] A.Y. Kitaev, A. Shen, M.N. Vyalyi. *Classical and Quantum Computation*. American Mathematical Society, 2002.

[17] F.C. Hennie. *Finite-State Models for Logical Machines*. Wiley, New York, 1968.

[18] R. Sedgewick, and K. Wayne. *Computer Science: An Interdisciplinary Approach*. Addison-Wesley, 2017.

[19] M.A. Nielsen, and I.L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2010.

[20] M. Kjaergaard, M.E. Schwartz, J. Braumuller, P. Krantz, J.I-J. Wang, S. Gustavsson, and W.D. Oliver. Superconducting qubits: Current state of play. *Annual Review of Condensed Matter Physics*, 11:369–395, 2020.

[21] A.M. Steane. The ion trap quantum information processor. *Appl. Phys. B*, 64:623, 1997.

[22] C.D. Buzewicz, J. Chiaverini, R. McConnell, and J.M. Sage. Trapped ion quantum computing: Progress and challenges. *Appl. Phys. Rev.*, 6:021314, 2019.

[23] Q.A. Turchette, C.J. Hood, W. Lange, H. Mabuchi, and H.J. Kimble. Measurement of conditional phase shifts for quantum logic. *Phys. Rev. Lett.*, 75:4710–4713, 1995.

[24] P. Kok, W.J. Munro, K. Nemoto, T.C. Ralph, J.P. Dowling, and G.J. Milburn. Linear optical quantum computing. *Rev. Mod. Phys.*, 79:135, 2007.

[25] D.G. Cory, M.D. Price, and T.F. Havel. Nuclear magnetic resonance spectroscopy: An experimentally accessible paradigm for quantum computing, 1997. arXiv:quant-ph/9709001.

[26] T. Xin, B-X. Wang, K-R. Li, X-Y. Kong, S-J. Wei, T. Wang, D. Ruan, and G-L. Long. Nuclear magnetic resonance for quantum computing: Techniques and recent achievements. *Chinese Physics B*, 27(2):020308, 2018.

[27] D. Loss, and D.P. DiVincenzo. Quantum computation with quantum dots. *Phys. Rev. A*, 57:120, 1998.

[28] C. Kloeffel, and D. Loss. Prospects for spin based quantum computing in quantum dots. *Annual Review of Condensed Matter Physics*, 4:51–81, 2013.

[29] R. Cleve D.P. DiVincenzo N. Margolus P. Shor T. Sleator J. Smolin A. Barenco, C.H. Bennett and H. Weinfurter. Elementary gates for quantum computation. *Phys. Rev. A*, 52:3457–3467, 1995.

[30] S. Kanev, J. P. Darago, K. Hazelwood, P. Ranganathan, T. Moseley, G-Y.Wei, and D. Brooks. Profiling a warehouse–scale computer. In *Proceedings of the 42nd Annual Int. Sym. Computer Architecture, ISCA*, pages 158–169, 2015.

[31] D. C. Marinescu. *Cloud Computing; Theory and Practice*. Morgan Kaufmann, San Francisco, CA, 2 edition, 2017.

[32] D.R. Kent. *New Quantum Monte Carlo Algorithms to Efficiently Utilize Massively Parallel Computers*. PhD thesis, California Institute of Technology, Pasadena, CA, 2004.

[33] E.M. Stoudenmire, and S.R. White. Real-space parallel density matrix renormalization group. *Phys. Rev. B.*, 87:155137, 2013.

[34] T. Häner, and D. S. Steiger. 0.5 petabyte simulation of a 45-qubit quantum circuit, 2017. arXiv:1704.01127.

[35] A. A. Auer, G. Baumgartner, D. E. Bernholdt, A. Bibireata, V. Choppella, D. Cociorva, X. Gao, R. Harrison, S. Krishnamoorthy, S. Krishnan, C.-C. Lam, Q. Lu, M. Nooijen, R. Pitzer, J. Ramanujam, P. Sadayappan, and A. Sibiryakov. Automatic code generation for many-body electronic structure methods: the tensor contraction engine. *Mol. Phys.*, 104:211–228, 2005.

[36] G. Baumgartner, D. E. Bernholdt, D. Cociorva, C.-C. Lam, J. Ramanujam, R. Harrison, M. Noolijen, and P. Sadayappan. A performance optimization framework for compilation of tensor contraction expressions into parallel programs. In *Proceedings of 16th International Parallel and Distributed Processing Symposium*, page 33, 2002.

[37] G. Baumgartner, D. E. Bernholdt, D. Cociorva, R. Harrison, S. Hirata, C.-C. Lam, M. Nooijen, R. Pitzer, J. Ramanujam, and P. Sadayappan. A high-level approach to synthesis of high-performance codes for quantum chemistry. In *Proceedings of the 2002 ACM/IEEE Conference on Supercomputing*, page 5, 2002.

[38] R. N. C. Pfeifer, P. Corboz, O. Buerschaper, M. Aguado, M. Troyer, and G. Vidal. Simulation of anyons with tensor network algorithms. *Phys. Rev. B*, 82:115126, 2010.

[39] F. Verstraete, J. I. Cirac, and V. Murg. Matrix product states, projected entangled pair states, and variational renormalization group methods for quantum spin systems. *Adv. Phys.*, 57:143, 2008.

[40] G. Vidal. A class of quantum many-body states can be efficiently simulated. *Phys. Rev. Lett.*, 101:110501, 2008.

[41] C.-C. Lam, P. Sadayappan, and R. Wenger. On optimizing a class of multi-dimensional loops with reduction for parallel execution. *Parallel Process. Lett.*, 7:157, 1997.

[42] N. Schuch, M. M. Wolf, F. Verstraete, and J. I. Cirac. Computational complexity of projected entangled pair states. *Phys. Rev. Lett.*, 80:094403, 2007.

[43] R. Orus, and G. Vidal. Simulation of two-dimensional quantum systems on an infinite lattice revisited: Corner transfer matrix for tensor contraction. *Phys. Rev. B*, 80:094403, 2009.

[44] Z.-C. Gu, M. Levin, and X.-G. Wen. Tensor-entanglement renormalization group approach as a unified method for symmetry breaking and topological phase transitions. *Phys. Rev. B*, 78:205116, 2008.

[45] M. Levin, and C. P. Nave. Tensor renormalization group approach to two-dimensional classical lattice models. *Phys. Rev. Lett.*, 99:120601, 2007.

[46] Z. Y. Xie, H. C. Jiang, Q. N. Chen, Z. Y. Weng, and T. Xiang. Second renormalization of tensor-network states. *Phys. Rev. Lett.*, 103:160601, 2009.

[47] R. B. Stinchcombe. Ising model in a transverse field. i. basic theory. *J. Phys. C: Solid State Phys.*, 6:2459, 1973.

[48] S. Sachdev. *Quantum Phase Transitions*. Cambridge University Press, Cambridge, U.K., 2011.

[49] H.W.J. Blote, and Youjin Deng. Cluster monte carlo simulation of the transverse ising model. *Phys. Rev. E*, 66:066110, 2002.

[50] L. Tagliacozzo, G. Evenbly and G. Vidal. Simulation of two-dimensional quantum systems using a tree tensor network that exploits the entropic area-law. *Phys. Rev. B.*, 80:235127, 2009.

[51] A. Cichocki, N. Lee, I. Oseledets, A.-H. Phan, Q. Zhao, M. Sugiyama, and D.P. Mandic. Tensor Networks for Dimensionality Reduction and Large-scale Optimization: Part 2 Applications and Future Perspectives. *Foundations and Trends in Machine Learning*, 9(6):431, 2017.

[52] L.G. Valiant. The Complexity of Enumeration and Reliability Problems. *SIAM J. Comput.*, 8(3):410–421, Aug 1979.

[53] D. Roth. On the hardness of approximate reasoning. *Artif. Intell.*, 82(1-2):273–302, Apr 1996.

[54] H. Haanpää, M. Järvisalo, P. Kaski, and I. Niemelä. Hard Satisfiable Clause Sets for Benchmarking Equivalence Reasoning Techniques. *Journal on Satisfiability, Boolean Modeling and Computation*, 2:27–46, 2006.

[55] H. Jia, C. Moore, and B. Selman. From Spin Glasses to Hard Satisfiable Formulas. In *Theory and Applications of Satisfiability Testing, 7th International Conference, SAT 2004*, pages 199–210, 2005.

[56] L. Zdeborová, and F. Krzakala. Generalization of the cavity method for adiabatic evolution of Gibbs states. *Phys. Rev. B*, 81(22):224205, Jun 2010.

[57] E. Farhi, D. Gosset, I. Hen, A.W. Sandvik, P. Shor, A.P. Young, and F. Zamponi. Performance of the quantum adiabatic algorithm on random instances of two optimization problems on regular hypergraphs. *Phys. Rev. A*, 86(5):052334, Nov 2012.

[58] J.A. Scott. A New Row Ordering Strategy for Frontal Solvers. *Numerical Linear Algebra with Applications*, 6(3):189–211, 1999.

[59] Z.-C. Yang, S. Kourtis, C. Chamon, E.R. Mucciolo, and A.E. Ruckenstein. Tensor network method for reversible classical computation. *Phys. Rev. E*, 97(3):033303, Mar 2017.

[60] U. Schollwöck. The density-matrix renormalization group in the age of matrix product states. *Ann. Phys. (N. Y).*, 326(1):96–192, Jan 2011.

[61] J. Preskill. Quantum computing and the entanglement frontier, 2012. arXiv:1203.5813.

[62] F. Arute, K. Arya, R. Babbush *et al*. Quantum supremacy using a programmable superconducting processor. *Nature*, 574:505–510, 2019.

[63] S. Boixo, S. Isakov, V.N. Smelyanskiy, R. Babbush, N. Ding, Z. Jiang, M.J. Bremner, J.M. Martinis, H. Neven. Characterizing quantum supremacy in near term devices. *Nature Physics*, 14:590–600, 2018.

[64] N.J. Cerf, and S.E. Koonin. Monte carlo simulation of quantum computation. *Math and Comp. in Simulation*, 47:143–151, 1998.

[65] E. Pednault, J.A. Gunnels, G. Nannicini, L. Horesh, and R. Wisnieff. Leveraging secondary storage to simulate deep 54-qubit sycamore circuits, 2019. arXiv:1910.09534.

[66] F.M. Haehl, R. Loganayagam, and M. Rangamani. Schwinger-keldysh formalism part 1:brst symmetries and superspace. *J. High Energ. Phys.*, 69, 2017.

[67] V. Kliuchnikov, D. Maslov, and M.Mosca. Fast and efficient exact synthesis of single qubit unitaries generated by clifford and t gates. *Quantum Information and Computation*, 13:607–630, 2013.

[68] S. Paeckel, T. Köhler, A. Swoboda, S.R. Manmana, U. Schollwöck, and C. Hubig. Time-evolution methods for matrix-product states. *Annals of Physics*, 411:167998, 2019.

[69] T.G. Draper. Addition on a quantum computer, 2000. arXiv:quant-ph:0008033.

[70] P.W. Shor. Polynomial time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Sci. Statist. Comput.*, 26:1484, 1997.

[71] J. M. Deutsch. Quantum statistical mechanics in a closed system. *Phys. Rev. A.*, 43:2046–2049, 1991.

[72] P. Calabrese, and J. Cardy. Evolution of entanglement entropy in one-dimensional systems. *Journal of Statistical Mechanics: Theory and Experiment*, 2005:04010, 2005.

[73] M. Rigol, V. Dunjko, and M. Olshanii. Thermalization and its mechanism for generic isolated quantum systems. *Nature*, 452:854–858, 2008.

[74] H. Kim, and D. A. Huse. Ballistic spreading of entanglement in a diusive nonintegrable system. *Phys. Rev. Lett.*, 111:127205, 2013.

[75] M. Mezei, and D. Stanford. On entanglement spreading in chaotic systems. *Journal of High Energy Physics*, 2017:65, 2017.

[76] B.L. Altshuler, Y. Gefen, A. Kamenev, and L.S. Levitov. Quasiparticle lifetime in a finite system: A nonperturbative approach. *Phys. Rev. Lett.*, 78:2803, 1997.

[77] V. Oganesyan, and D.A. Huse. Localization of interacting fermions at high temperature. *Phys. Rev. B*, 75:155111, 2007.

[78] A. Pal, and D.A. Huse. Many-body localization phase transition. *Phys. Rev. B*, 82:174411, 2010.

[79] J.H. Bardarson, F. Pollmann, and J.E. Moore. Unbounded growth of entanglement in models of many-body localization. *Phys. Rev. Lett.*, 109:017202, 2012.

[80] T. Grover. Certain General Constraints on the Many-Body Localization Transition. *arXiv:1405.1471*, 2014.

[81] S.S. Kondov, W.R. McGehee, W. Xu, and B. DeMarco. Disorder-induced localization in a strongly correlated atomic hubbard gas. *Phys. Rev. Lett.*, 114:083002, Feb 2015.

[82] M. Schreiber, S.S. Hodgman, P. Bordia, H.P. Lüschen, M.H. Fischer, R. Vosk, E. Altman, U. Schneider, and I. Bloch. Observation of many-body localization of interacting fermions in a quasirandom optical lattice. *Science*, 349(6250):842–845, 2015.

[83] J.-Y. Choi, S. Hild, J. Zeiher, P. Schauß, A. Rubio-Abadal, T. Yefsah, V. Khemani, D.A. Huse, I. Bloch, and C. Gross. Exploring the many-body localization transition in two dimensions. *Science*, 352(6293):1547–1552, 2016.

[84] K.X. Wei, C. Ramanathan, and P. Cappellaro. Exploring localization in nuclear spin chains. *Phys. Rev. Lett.*, 120:070501, Feb 2018.

[85] J. Smith, A. Lee, P. Richerme, B. Neyenhuis, P.W. Hess, P. Hauke, M. Heyl, D.A. Huse, and C. Monroe. Many-body localization in a quantum simulator with programmable random disorder. *Nature Physics*, 12(10):907, 2016.

[86] X. Cao, A. Tilloy, and A.D. Luca. Entanglement and transport of a fermion chain under continuous monitoring. *arXiv:1804.04638*, 2018.

[87] A. Chan, R.M. Nandkishore, M. Pretko, and G. Smith. Weak measurements limit entanglement to area law (with possible log corrections). *arXiv:1808.05949*, 2018.

[88] B. Skinner, J. Ruhman, and A. Nahum. Measurement-Induced Phase Transitions in the Dynamics of Entanglement. *arXiv:1808.05953*, 2018.

[89] Y. Li, X. Chen, and M.P.A. Fisher. Measurement-driven entanglement transition in hybrid quantum circuits. *arXiv:1901.08092*, 2019.

[90] Y. Li, X. Chen, and M.P.A. Fisher. Quantum zeno effect and the many-body entanglement transition. *Phys. Rev. B*, 98:205136, 2018.

[91] Y. Bao, S. Choi, and E. Altman. Theory of the Phase Transition in Random Unitary Circuits with Measurements. *arXiv:1908.04305*, 2019.

[92] C.-M. Jian, Y.-Z. You, and A.W.W. Ludwig. Measurement-induced criticality in random quantum circuits. *arXiv:1908.08051*, 2019.

[93] M.J. Gullans, and D.A. Huse. Scalable probes of measurement-induced criticality. *arXiv:1910.00020*, 2019.

[94] H. Li, and F.D.M. Haldane. Entanglement spectrum as a generalization of entanglement entropy: Identification of topological order in non-abelian fractional quantum hall effect states. *Phys. Rev. Lett.*, 101:010504, 2007.

[95] C.Chamon, A. Hamma, and E.R. Mucciolo. Emergent irreversibility and entanglement spectrum statistics. *Phys. Rev. Lett.*, 112:240501, 2014.

[96] S.D. Geraedts, R. Nandkishore, and N. Regnault. Many-body localization and thermalization: Insights from the entanglement spectrum. *Phys. Rev. B*, 93:174202, 2016.

[97] Z.-C.Yang, A. Hamma, S.M. Giampaolo, E.R. Mucciolo, and C. Chamon. Entanglement complexity in quantum many-body dynamics, thermalization, and localization. *Phys. Rev. B*, 96:020408, 2017.

[98] Z.-C. Yang, C. Chamon, A. Hamma, and E.R. Mucciolo. Two-component structure in the entanglement spectrum of highly excited states. *Phys. Rev. Lett.*, 115:267206, 2015.

[99] V.A. Marcenko, and L.A. Pastur. Distribution of eigenvalues for some sets of random matrices. *Mathematics of the USSR-Sbornik*, 1:457, 1967.

[100] Y.Y. Atas, E. Bogomolny, O. Giraud, and G. Roux. Distribution of the ratio of consecutive level spacings in random matrix ensembles. *Phys. Rev. Lett.*, 110:084101, 2013.

[101] E.A. Demler, S. Gopalakrishnan, K. Agarwal, and M. Knap. Griffiths effects and slow dynamics in nearly many-body localized systems. *Phys. Rev. B*, 93:134206, 2016.

[102] A. Cichocki. Tensor networks for big data analytics and large-scale optimization problems. *arxiv:1407.3124*, 2014.

[103] G. Vidal. Algorithms for entanglement renormalization (arxiv version 2). *arxiv:0707.1454v2*, 2007.

[104] M. Dolfi, B. Bauer, M. Troyer, and Z. Ristivojevic. Multigrid algorithms for tensor network states. *Phys. Rev. Lett.*, 109:020604, 2012.

[105] K. Batselier, A. Cichocki, and N. Wong. MERACLE: Constructive layer-wise conversion of a tensor train into a MERA. *arxiv:1912.09775*, 2019.

[106] J. Walker. *A Primer of Wavelets and Their Scientific Applications*. CRC Press, 1999.

[107] K.G. Wilson. Problems in physics with many scales of length. *Scientific American*, pages 158–179, 1979.

[108] G. Vidal. Entanglement renormalization. *Physical Review Letters*, **99**(220405), 2007.

[109] G. Evenbly, and G. Vidal. Algorithms for entanglement renormalization. *Phys. Rev. B*, **79**(144108), 2009.

[110] E. M. Stoudenmire and D. J. Schwab. Supervised learning with quantum inspired tensor networks. *Advances in Neural Information Processing Systems*, **29**(4799), 2017.

[111] IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events. `http://www.cs.tut.fi/sgn/arg/dcase2017/challenge/`, 2017.

[112] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, **61**:85–117, 2015.

[113] P. Calabrese, F.H.L. Essler, and M. Fagotti. Quantum quench in the transverse field ising chain. *Phys. Rev. Lett.*, 106:227203, 2011.

[114] A. Strathearn, P. Kirton, D. Kilda, J. Keeling, B.W. Lovett. Efficient non-markovian quantum dynamics using time-evolving matrix product operators. *Nature Communications*, 9:3322, 2018.

[115] D. C. Marcello, M. Caccin, P. Baireuther, T. Hyart, and M. Fruchart. Machine learning assisted measurement of local topological invariants. *arxiv:1906.03346*, 2019.

[116] A. Melkinov, L. Fedichkin, A. Alodjants. Detecting quantum speedup by quantum walk with convolutional neural networks. *arxiv:1901.10632*, 2019.

[117] K. Chinjo, S. Sota, S. Yunoki, and T. Tohyama. Characterization of photoexcited state in the half-filled one-dimensional extended hubbard model assisted by machine learning. *arxiv:1901.07900*, 2019.

[118] Y. Nomura, A.S. Darmawan, Y. Yamaji, and M. Imada. Restricted boltzmann machine learning for solving strongly correlated quantum systems. *Phys. Rev. B*, 96:205152, 2017.

[119] X. Gao, and L.-M. Duong. Efficient representation of quantum many body states with deep neural networks, 2019. arXiv:1701.05039.

[120] I. Glasser, N. Pancotti, M. August, I.D. Rodriguez, and J.I. Cirac. Neural-network quantum states, string-bond states, and chiral topological states. *Phys. Rev. X*, 8:011006, 2018.

[121] G. Montufar. Restricted boltzmann machines: Introduction and review, 2018. arXiv:1806.07066.

[122] M.J. Hartmann, and G. Carleo. Neural network approach to dissipative quantum many-body dynamics. *Phys. Rev. Lett.*, 122:250502, 2019.

[123] G. Carleo, and M. Troyer. Solving the quantum many-body problem with artificial neural networks. *Science*, 355:602, 2017.

[124] D. Hendry, and A.E. Feiguin. A machine learning approach to dynamical properties of quantum many-body systems. *Phys. Rev. B*, 100:245123, 2019.

[125] M. Suzuki. Generalized trotter's formula and systematic approximants of exponential operators and inner derivations with applications to many-body problems. *Commun. Math. Phys.*, 51:183–190, 1976.

[126] G. Vidal. Efficient simulation of one-dimensional quantum many-body systems. *Phys. Rev. Lett.*, 93:040502, 2004.

[127] F. Chollet, and others. Keras. `https://keras.io`, 2015.

[128] J. Strecka, and M. Jascur. A brief account of the ising and ising-like models: Mean-field, effective-field and exact results. *Acta Physics Slovaka*, 65:235–367, 2015.

[129] F. Franchini. An introduction to integrable techniques for one-dimensional quantum systems, 2017. arXiv:1609.02100v3.