UNIVERSITY OF BRITISH COLUMBIA

CPEN 400Q - Topics in Computer Engineering Gate-model quantum computing
Spring 2023

Synergy Between Quantum Circuits and Tensor Networks: Short-cutting the Race to Practical Quantum Advantage

Abhishek Abhishek, Daniel Kong, Harmeeta Dahiya, Mushahid Khan

Synergy Between Quantum Circuits and Tensor Networks: Short-cutting the Race to Practical Quantum Advantage

Abhishek Abhishek, Daniel Kong, Harmeeta Dahiya, Mushahid Khan

April 16, 2023

Abstract

Parameterized Quantum Circuits (PQCs) are the building blocks for near-term quantum computing in the Noisy Intermediate Scale Quantum (NISQ) era. Their relatively low circuit depth and gate count, in addition to their differentiability, makes them amenable for hybrid quantum-classical algorithms. However, the challenges of optimizing PQCs have been widely noted in practice primarily due to the phenomena of barren plateaus. In this project, we explored a recent work that aims to address one of the key factors believed to cause barren plateaus i.e. random initialization of PQCs. The work proposes a combined classical-quantum framework, where a tensor network model is first optimized using classical resources to find good task-specific initialization, and is then mapped onto a Parameterized Quantum Circuit, which is then extended and further optimized using quantum resources.

1 Introduction

Quantum algorithms struggle to compete with classical ones due to many issues. One of which is the existence of barren plateaus in the optimization of PQCs (McClean et al. 2018). In this paper, the idea of task-specific initialization for the PQC is explored and applied to Quantum Machine Learning (QML) in order to overcome barren plateaus due to poor initialization.

Another problem found in QML is the increasing efficiency of classical solutions through Tensor Networks (TNs). TNs provide an incredible advantage by having the ability to be trained and run on classical hardware such as GPUs.

The solution proposed here combines both classical and quantum machine learning methods. Our solution uses TNs to initialize the parameters for a PQC before further extending this PQC and training it on quantum hardware. To initialize the parameters, we use the methods found in Han et al. 2018 to encode a probability distribution into the MPS, to represent entangled quantum states. We then transform the resulting trained MPS into unitaries for use in a PQC (Barratt et al. 2021, Dborin et al. 2022). Finally, we can extend these unitaries with additional gates for better performance (Benedetti et al. 2019). This results in a better performance relative to traditional PQCs. An example of this structure can be seen in Figure 1.

The expressivity of an MPS is determined by its bond dimension χ. Increasing χ implies the use of more classical computing resources. This project compares the loss function for PQC initializations with classically trained MPS, randomly initialized PQCs, and near unitary initialized PQCs. Initializations with MPS with higher χ should give better performance. This is intuitively visualized in Figure 2.

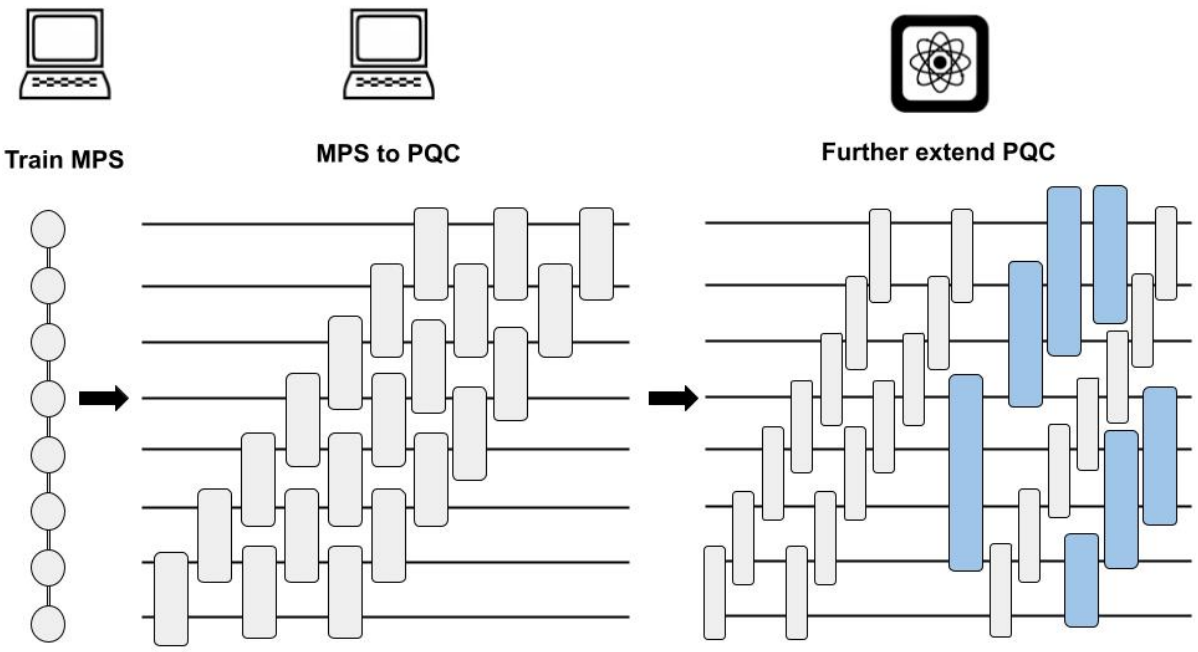

Figure 1: Structure of the training solution utilizing both TNs and PQCs. We begin with a classically trained model involving tensor networks and matrix product states. This is then transformed into unitaries for use in a quantum circuit, which is further extended and trained.

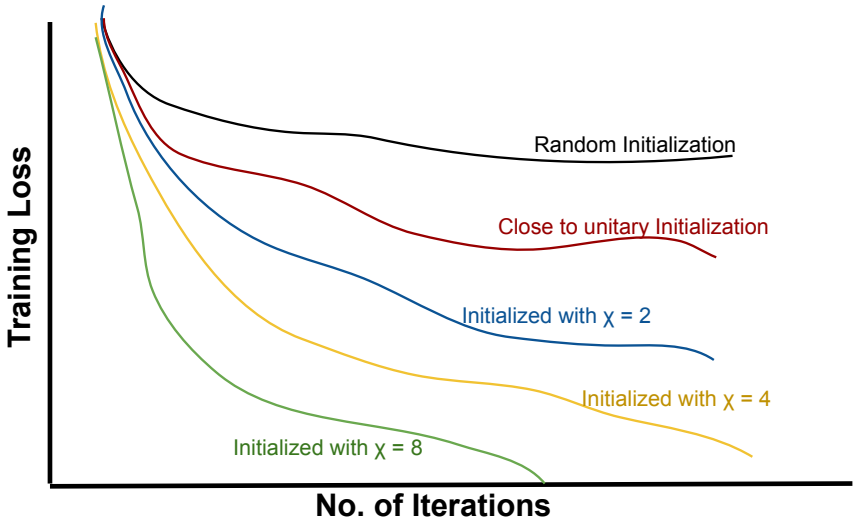

Figure 2: Behaviour of training loss for different types of PQC initializations observed in (M. S. Rudolph, Miller, et al. 2022)

2 Theory

2.1 Tensor networks and matrix product states

A tensor is a linear algebraic object where the order of the tensor corresponds to the number of indices (i.e. the dimensions of the tensor). For example, a scalar x is an order-0 tensor, a vector ν_α is an order-1 tensor, and a matrix $A_{\alpha,\beta}$ is an order-2 tensor. A tensor network is a graph where the individual nodes are tensors of arbitrary order, edges between the nodes correspond to "virtual" indices and open/dangling edges correspond to "real" indices. For e.g. the tensor networks in Fig 3 (with the virtual indices contracted) correspond to a matrix-matrix product $C_{\alpha\gamma} = \Sigma_\beta A^\beta_\alpha B^\beta_\gamma$, a vector

dot product $C = \Sigma_\beta A^\beta B^\beta$, and a trace of the product of two matrices $C = \mathrm{Tr}(AB)$ respectively. Note that in the notation we use, the superscripts correspond to "virtual" indices and the subscripts correspond to "real" indices respectively.

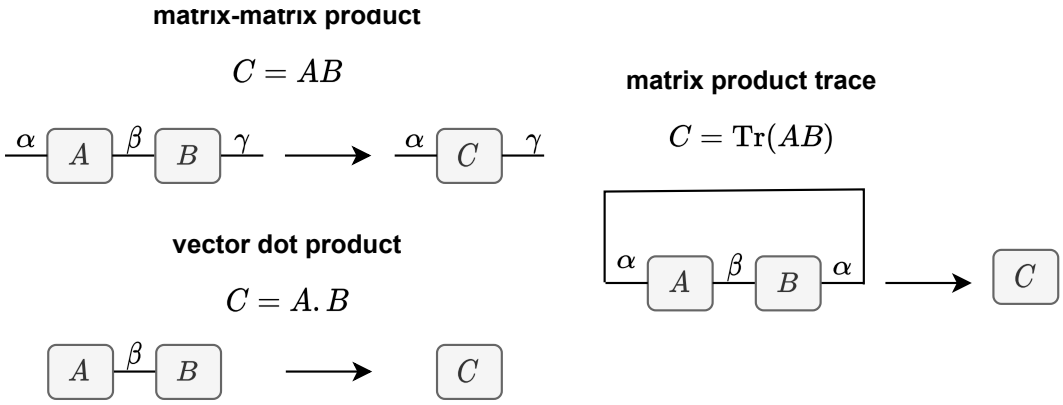

Figure 3: Examples of tensors network diagrams illustrating commonly used linear algebraic operations such as matrix-matrix products and matrix trace.

Tensor networks were initially proposed in condensed matter physics to classically simulate complex quantum many-body systems. In order to see why, let's consider an N qubit wavefunction,

$$|\Psi\rangle = \sum_{i_1,i_2,\ldots,i_N} \psi_{i_1,i_2,\ldots,i_N} |i_1\rangle \otimes |i_1\rangle \otimes \ldots |i_N\rangle \tag{1}$$

We note that the tensor of the complex coefficients $\psi_{i_1,i_2,\ldots,i_N}$ can be described as an order-N tensor with $O(2^N)$ entries and N indices that can take up to two values (i.e. $\forall j, i_j \in \{0,1\}$) (see Fig.4a.). However, if there exists some "structure" in the wavefunction e.g. limited entanglement and local nearest-neighbor interactions, it can be efficiently approximated using low-order tensor networks such as matrix product states (MPS) or projected entangled pair states (PEPS). Matrix product states are computationally tractable tensor network models whose core tensors $M_{[i]}$ are connected along a line graph (see Fig.4). The MPS wavefunction,

$$|\tilde{\Psi}\rangle = \sum_{i_1,i_2,\ldots,i_N} m_{i_1,i_2,\ldots,i_N} |i_1\rangle \otimes |i_1\rangle \otimes \ldots |i_N\rangle \tag{2}$$

is an approximation to $|\Psi\rangle$ where the coefficients are now stored in the core tensors $M_{[i]}$, and are given by :

$$m_{i_1,i_2,\ldots,i_N} = \sum_{\alpha_1,\alpha_2,\ldots,\alpha_{N-1}} M^{\alpha_1}_{[1],i_1} M^{\alpha_1,\alpha_2}_{[2],i_2} \ldots M^{\alpha_{N-1}}_{[N],i_N} \tag{3}$$

which corresponds to fixing the values of all open indices i_j to either $\{0,1\}$ and contracting all virtual indices $\alpha_j, j \in \{1,\ldots,N-1\}$. The summation over each virtual index α_j goes from 1 to χ_j where χ_j is what's referred to as the bond dimension at bond j i.e.

$$\sum_{\alpha_1,\alpha_2,\ldots,\alpha_{N-1}} \rightarrow \sum_{\alpha_1=1}^{\chi_1} \sum_{\alpha_2=1}^{\chi_2} \ldots \sum_{\alpha_{N-1}=1}^{\chi_{N-1}} \tag{4}$$

It can be observed that an N-qubit matrix product state (MPS) has $O(2N\chi^2)$ parameters where $\chi = \max_j\{\chi_j\}$ is the maximum bond dimension in the MPS. In general, practical tensor network approximations of higher order tensors have parameters $O(poly(N)poly(\chi))$ (Orús 2014) which is significantly more efficient compared to $O(exp(N))$ in the full state representation. This is the primary reason why tensor networks are so popular in the condensed matter physics community since they allow us to approximately simulate quantum systems far beyond what is possible with full-state simulators. More recently, due to their ability to capture correlations (i.e. entanglement) in the quantum states, tensor network models are being explored for performing machine learning tasks such as unsupervised generative modeling (Han et al. 2018) and supervised classification (Stoudenmire and Schwab 2016).

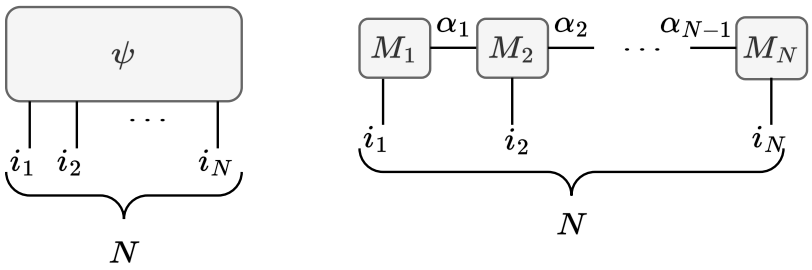

Figure 4: An order-N tensor (left) and a Matrix Product State (MPS) tensor network (right) corresponding to an N-site quantum state. If each site is an m-level quantum system, the open indices i_j can take upto m different values, i.e. $i_j = 1, \ldots, m$ ($m = 2$ for qubits)

2.2 Mapping MPS tensor networks to Parameterized Quantum Circuits

One of the main focuses of our project was to understand and implement the mapping from a trained matrix product state tensor network to a parameterized quantum circuit. There are several ways to go about doing this mapping ranging from "simple" (Barratt et al. 2021, Dborin et al. 2022) to "complex" (Ran 2020) to "even more complex" (Shirakawa et al. 2021, M. S. Rudolph, Chen, et al. 2022). The key difference between them is that the "simple" techniques map the MPS tensor network to a parameterized quantum circuit consisting of multi-qubit unitaries while the "complex" and "even more complex" techniques map to a parameterized quantum circuit consisting of only two-qubit unitaries.

2.2.1 Mapping to multi-qubit unitaries

In this section, we describe the "simple" technique which maps an MPS tensor network to a quantum circuit consisting of multi-qubit unitaries. In order to fully understand how this works, we need to first review the concept of QR-factorization and see how that applies to MPS tensor networks.

QR-factorization Given any $D \times D'$ complex rectangular matrix M (without loss of generality, we can assume $D > D'$), M can be decomposed as $M = QR$ where Q is a $D \times D'$ rectangular matrix, and R is an upper-triangular $D' \times D'$ matrix (see Fig. 5). The key thing to note here is that the columns of the matrix Q are orthonormal, therefore it satisfies $Q^\dagger Q = I_{D' \times D'}$ and is a left isometry. Such a factorization can be obtained using the Gram-Schmidt process or Givens rotations.

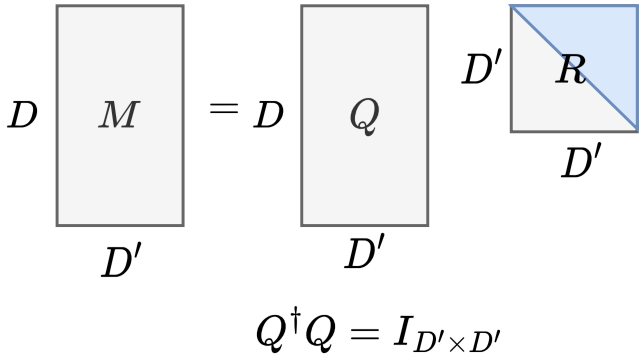

Figure 5: QR-factorization of a $D \times D'$ matrix M into a $D \times D'$ left isometry Q and an $D' \times D'$ upper triangular matrix R

QR-factorization and the Matrix Product State A key idea in the framework of MPS tensor networks is that of the gauge freedom in its representation. It can be simply observed that for any

two adjacent MPS core tensors $M^{\alpha_{m-1},\alpha_m}_{[m],i_m}$ and $M^{\alpha_m,\alpha_{m+1}}_{[m+1],i_m+1}$, there is an equivalent representation:

$$M^{\alpha_{m-1},\alpha_m}_{[m],i_m} M^{\alpha_m,\alpha_{m+1}}_{[m+1],i_m+1} = (M^{\alpha_{m-1},\alpha_m}_{[m],i_m} X)(X^{-1} M^{\alpha_m,\alpha_{m+1}}_{[m+1],i_m+1}) \tag{5}$$

This leads naturally to the derivation of the so-called **canonical** or **normalized** forms of the MPS where certain sites in the MPS are fixed to be the center of orthogonality. For an in-depth discussion on this, we refer to (Perez-Garcia et al. 2006, Ballarin 2021) but note that in this project, we primarily focused on, and used the left canonical form of the MPS described below.

An MPS is said to be in left-canonical form if all its rightmost tensor is its **center of orthogonality**. In other words, this simply means that all core tensors in the MPS are isometries between their *(left virtual + open)* and *(right virtual)* indices i.e. they are left-isometries. Any arbitrary MPS $|\tilde{\Psi}\rangle$ can be converted to its left-canonical form through iterated QR-factorizations. In Fig. 6, we illustrate a simple example of left-canonicalization of a 3-site MPS through iterated QR-factorizations.

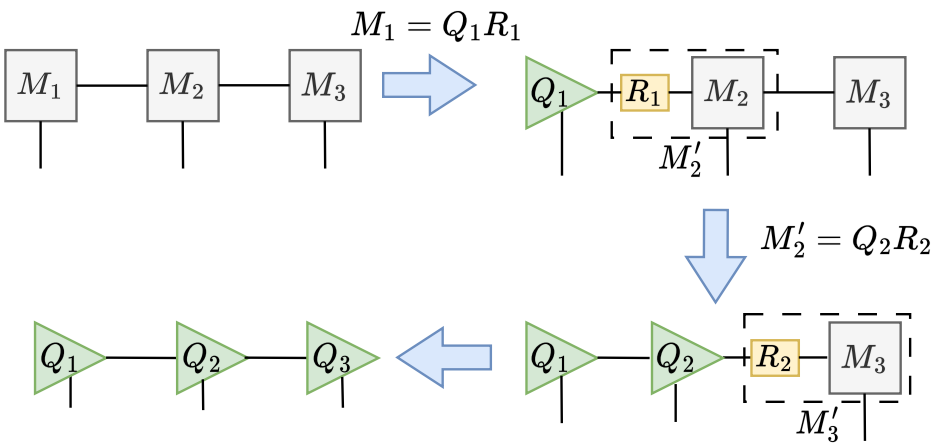

Figure 6: Left canonicalization (also known as left-normalization) of a three-site MPS tensor network via iterated QR factorizations. We start from the left-most site, and end at the right-most site. Once the procedure is completed, the MPS is said to be in its left-canonical form, and the core tensor at each site is a left isometry.

MPS isometries to unitaries Once the MPS is in the left-canonical form via the above procedure, the next step is to expand the left isometries Q_i (green triangles in Fig. 6) into unitary matrices which can be applied as quantum gates in a parameterized quantum circuit. To reiterate, each MPS core tensor Q_i in the left-canonical form satisfies :

$$Q_i^\dagger Q_i = I, \quad Q_i Q_i^\dagger \neq I \tag{6}$$

Given a left isometry Q_i of size $D \times D'$ (assume $D > D'$ without loss of generality), in order to construct the unitary U_i of size $D \times D$, we simply extend the D' columns of Q_i with $\kappa := D - D'$ vectors from the kernel space (i.e. null space) of $Q^\dagger$. These additional basis vectors can be arranged in a $D \times \kappa$ matrix X_i satisfying:

$$Q_i^\dagger X_i = 0, \quad X_i^\dagger X_i = I_{\kappa \times \kappa} \tag{7}$$

The resulting matrix $U_i = [Q_i \;\; X_i]$ is unitary satisfying :

$$U_i^\dagger U_i = U_i U_i^\dagger = I_{D \times D} \tag{8}$$

We illustrate this construction in Fig. 7. The last thing to note here is that since the sizes of the left-isometries i.e. (D, D') are related to the bond-dimensions of the original MPS (see Section

2.1), they're not necessarily factors of 2 which is needed for n-qubit gates i.e. $D = 2^n$. In order to address this, the bond dimensions for each MPS core tensor, M_i (see Fig. 4) χ_{left} and χ_{right} are padded to powers of two (M. S. Rudolph, Chen, et al. 2022, Section 2B.) before performing left-canonicalization. The resulting unitaries can then be made into a quantum circuit (see Figure 13).

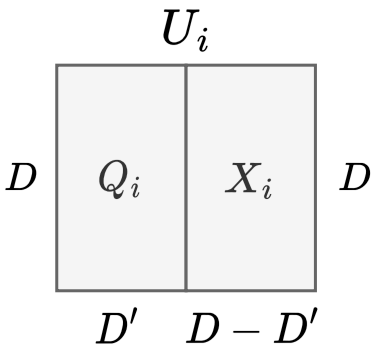

Figure 7: Converting a $D \times D'$ left-isometry Q_i to a $D \times D$ unitary U_i by stacking another $D \times (D - D')$ matrix X_i next to it. The columns of X_i are vectors from the kernel space of $Q_i^\dagger$.

2.3 Quantum Circuit Born Machine

A Quantum Circuit Born Machine (QCBM) (Benedetti et al. 2019) is a near-term hybrid quantum-classical algorithm that uses quantum circuit learning for solving generative tasks. QCBMs use Born's rule to parameterize classical probabilities and are capable of representing complicated probability distributions (Coyle et al. 2020). If we consider an n-bit binary string $\mathbf{x} = [x_1, x_2, \ldots, x_n]$, its probability under the model distribution is given using Born's rule :

$$q_\theta(\mathbf{x}) = |\langle \mathbf{x} | \psi_\theta \rangle|^2 \tag{9}$$

where θ are the model parameters, and ψ_θ is the model wave function, $\psi_\theta = U_\theta |0\rangle^{\otimes n}$.

In order to train the model parameters θ, we often use the Kullback-Libeler (KL) divergence between the QCBM distribution $q_\theta(\mathbf{x})$ and the target classical probability distribution $p(\mathbf{x})$. Given a dataset $\mathcal{D}$ with an associated empirical probability distribution $p_\mathcal{D}(\mathbf{x})$, the QCBM aims to minimize the KL divergence:

$$\mathcal{L}(\theta) = KL(p_\mathcal{D}(\mathbf{x}) || q_\theta(\mathbf{x})) \underset{\mathbf{x}\ \text{i.i.d.} \in \mathcal{D}}{=} -log(|\mathcal{D}|) - \frac{1}{|\mathcal{D}|} \sum_{x \in |\mathcal{D}|} \log(q_\theta(\mathbf{x})) \tag{10}$$

where $|\mathcal{D}|$ is the size of the dataset $\mathcal{D}$, and the last part of the equation holds if all $\mathbf{x} \in \mathcal{D}$ are i.i.d. (independently and identically distributed). We note that for datasets of moderate size, the KL divergence can be computed analytically using the form above, however, the above approach (of summing over all examples) does not scale well to very large real-world datasets. An overview of the learning procedure for the QCBM is illustrated in Fig. 8.

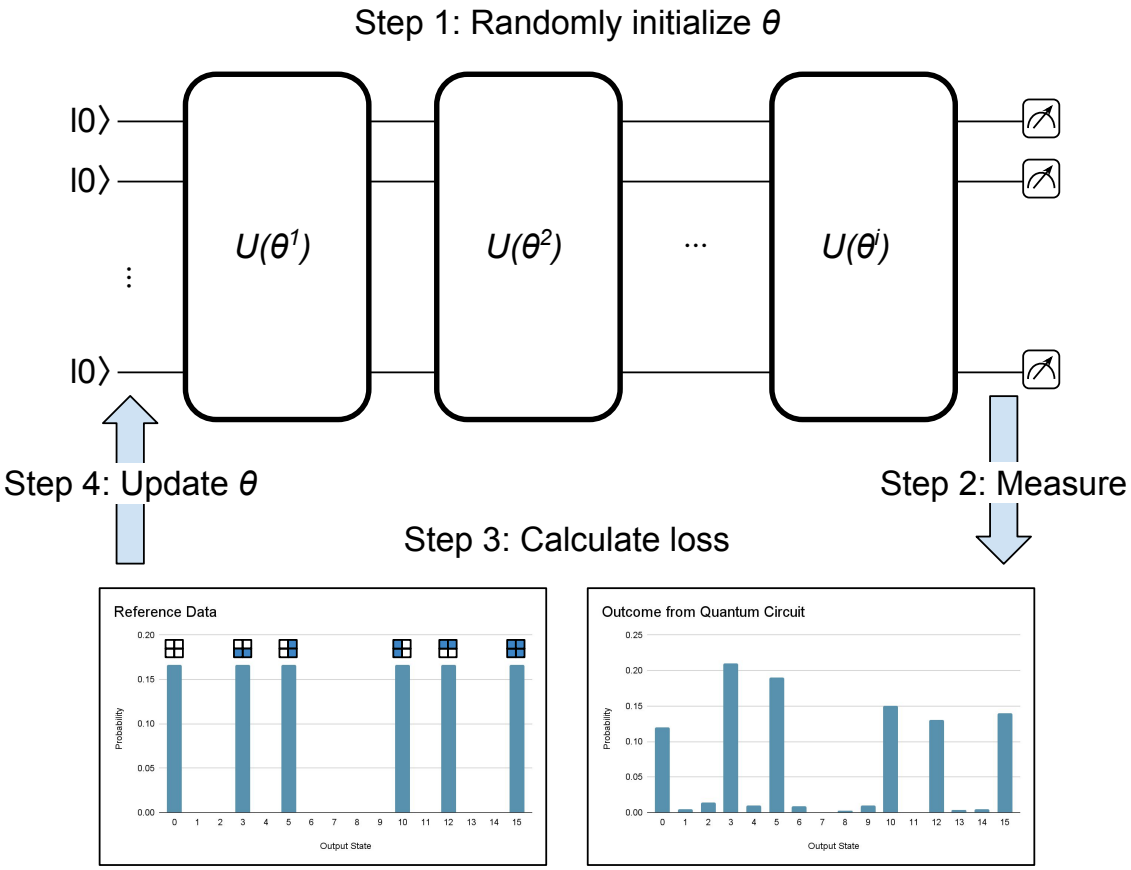

Figure 8: The original QCBM training method proposed in (Benedetti et al. 2019). The parameters are originally randomly initialized. We then measure the probabilities assigned to different computational basis states by the quantum circuit, and compare then against the basis states corresponding to bitstrings **x** in our data. KL divergence is commonly used as the loss function. Finally, the model parameters are updated and we iterate until the loss is minimized or the model has converged.

3 Framework

3.1 Choice of quantum programming framework

We used Python3 for the implementation. From the wide array of quantum programming frameworks available, such as Cirq, Qiskit, Q#, ForestSDK, we decided on utilizing Pennylane (Bergholm et al. 2018) for the following reasons:

1. Firstly, its functional programming nature led to the easy development of maintainable modules for different sections of the project.

2. Secondly, its core principle of maintaining differentiability of both classical and quantum computations and workflows allowed for straightforward mapping of the trained parameters of a trained MPS to parameters of a differentiable quantum circuit.

3. And finally, its ease of interface with automatic differentiation frameworks such as autograd and JAX helped in the implementation of the extended circuits and their further training

The above features are either not well supported or completely missing from the other frameworks we looked at which justified the choice of using Pennylane.

3.2 MPS tensor network implementation and training

We built our software framework on top of the repository developed by authors of (Han et al. 2018) which can be found at github.com/congzlwag/UnsupGenModbyMPS. The existing codebase had an implementation of the MPS tensor network, and the learning algorithm for the Tensor Network Born Machine (TNBM) described in the paper which forms the **"train MPS"** part of the solution proposed in (M. S. Rudolph, Miller, et al. 2022) (see Fig 1).

For the tensor network training, we made some minor modifications to the existing codebase to allow for the following things:

1. training and saving of MPS with maximum bond dimension restricted to $\chi_{max} = \{2, 4, 8\}$
2. training of MPS on Bars-and-Stripes (BAS) data with dimensions (4×3) which was used in the results and training of M. S. Rudolph, Miller, et al. 2022) rather than (4×4) BAS data which formed the results of (Han et al. 2018)

3.3 MPS to PQC mapping

In this project, we managed to fully implement the "simple" technique (Barratt et al. 2021, Dborin et al. 2022), and attempted to implement the "complex" method (Ran 2020), but were unsuccessful due to the level of technical details regarding tensor network operations and manipulations involved in that method (see mps-analytic-decomposition.ipynb and mpd-decomposition.ipynb for our not-so-successful attempts on this).

3.3.1 Implementation of the "simple" technique

We implemented new functions to perform **"MPS to PQC"** step in Fig. 1. In summary, these functions implement the theory described in (Section 2.2) and take as input a trained MPS tensor network, and convert it into a parameterized quantum circuit consisting of multi-qubit unitaries for general MPS.

To do this, we primarily relied on *qml.QubitUnitary* from PennyLane in order to place arbitrary unitaries extracted from the MPS as quantum gates in a PQC. However, we ran into issues during training since *qml.QubitUnitary* is not a differentiable operation. We note we relied on a quantum transform from *qml.transforms.unitary_to_rot* in order to first decompose our arbitrary 1- and 2-qubit *qml.QubitUnitary* gates (for e.g. see circuit in fig. 14) into differentiable parameterized gates such as *qml.RZ* and *qml.Rot* (e.g. see circuit in fig. 15). Since the decomposition transform only applies to single- and two-qubit unitary gates, this severely limited the different χ_{max} MPS that we could extend and train to reproduce the results such as those shown in Fig. 2 of (M. S. Rudolph, Miller, et al. 2022) where the authors show that the PQCs initialized with classically trained MPS were able to outperform the randomly initialized PQCs.

3.4 PQC training

To train our quantum circuits, we use the Adam Optimizer from Optax[1]. Adam optimizer is a gradient-based optimizer that can be used with Pennylane. We used the Adam optimizer because we created our quantum circuits in Pennylane and the Adam optimizer has fewer parameters for tuning. The original work uses Covariance Matrix Adaptation Evolution Strategy (CMAES) instead to optimize the circuits. To allow our code to run faster than generic code execution, we used Google's JAX with its Just-In-Time (JIT) compilation.

[1] https://optax.readthedocs.io/en/latest/api.html#optax.adam

4 Results

Here, we will look at the KL divergence to compare the performances of the PQC initialized with random SU(4) gates, near identity initialization, and those initialized with the solution found by the classically trained MPS. The quantum circuit of PQCs initialized with random and near identity SU(4) unitaries has three layers as seen in Figure 17 in Appendix 6. The first two layers follow a staircase topology and the last layer follows an all-to-all topology. As mentioned above, we were able to generate a quantum circuit with trainable gates and parameters for $\chi = 2$ MPS (this can be seen in 16 in Appendix 6).

We trained each model for 15000 iterations and used the following learning rates to generate the KL Divergence results: 1e-6, 1e-3, and 0.01, which are shown in figures 9, 10 and 11 below (for all the figures below, we fixed the y-scale to be the same as the Fig 2. in M. S. Rudolph, Miller, et al. 2022 to allow for direct comparison). In Figure 9, the KL divergence for the random initialization and near identity initialization does approach to a smaller value but does not converge very nicely after 2000 iterations potentially due to the very high learning rate. By lowering the learning rate, we were able to achieve more stable KL Divergence changes as shown in Figure 10. However, lowering the learning rate too low to 1e-6, the KL divergence for all three models does not even approach 1 as shown in 10.

In our implementations, the KL Divergence of the extended MPS circuit of $\chi = 2$ performs worse than the random initialization and near identity initialization. This is in direct contrast to the paper, where the KL Divergence of the extended MPS circuit of $\chi = 2$ (and also for $\chi > 2$) performs significantly better than the random initialization and the near identity initializations. We believe this deviation from the original paper could be attributed to us using a different optimizer, since automatic differentiation likely works better than CMA-ES for PQCs (as can be seen by the lower KL-divergence values for the random and near-identity initializations for our results compared to the original paper). Anyways, in all scenarios, we also observed that the near identity initializations converged significantly faster than the random initializations for all learning rates.

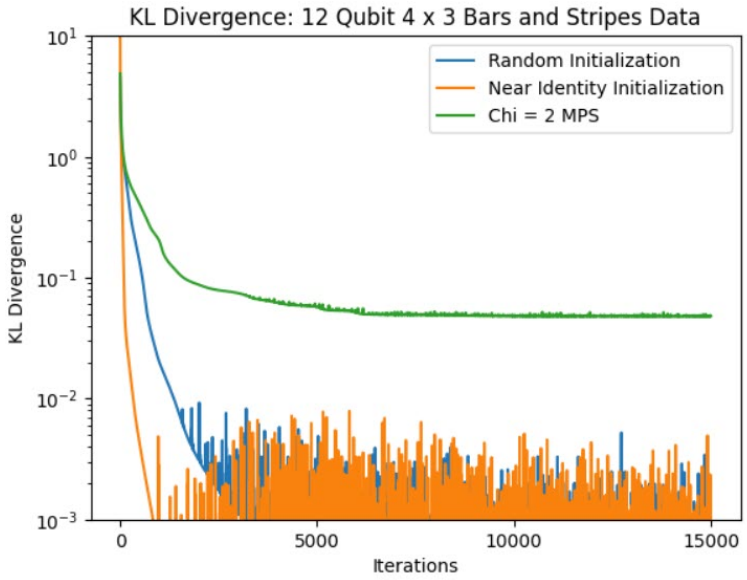

Figure 9: KL divergence plot for different PQC initializations for learning rate = 0.01

The bars and stripes dataset generated by our training models using learning rate 1e-3 for different PQC initializations can be seen in Figure 12.

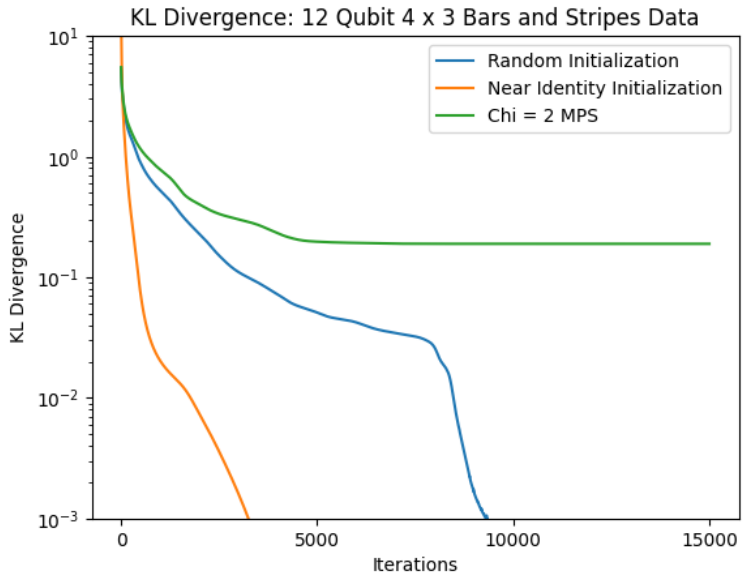

Figure 10: KL divergence plot for different PQC initializations for learning rate = 1e-3

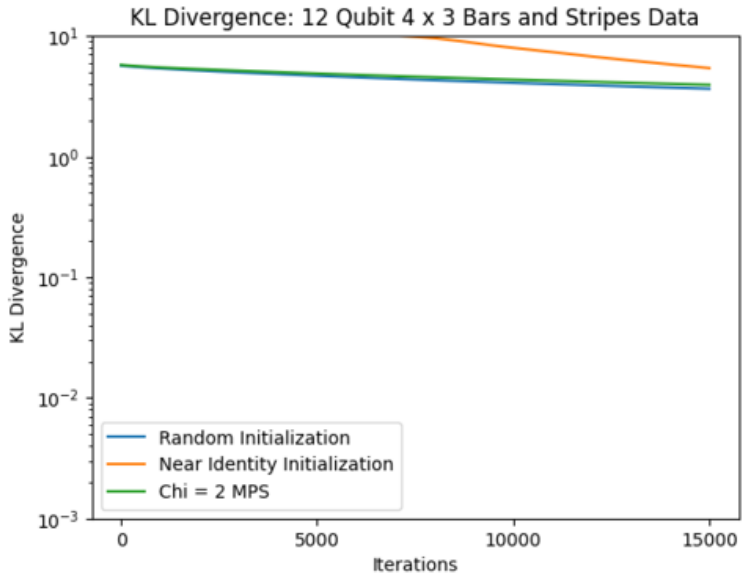

Figure 11: KL divergence plot for different PQC initializations for learning rate = 1e-6

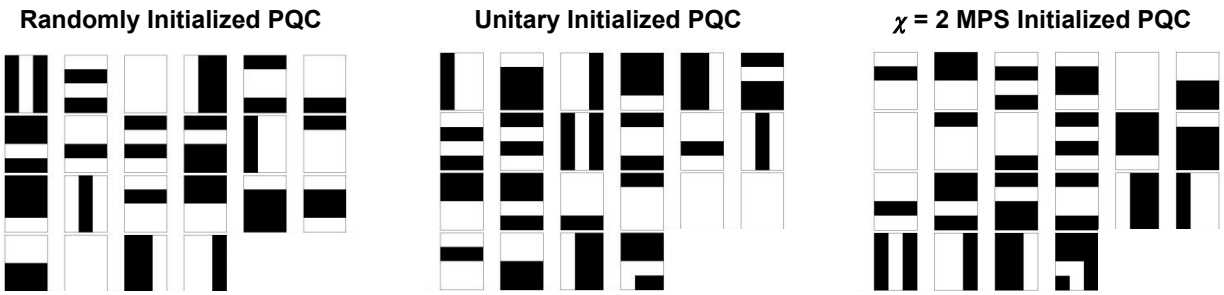

Figure 12: Bars and stripes data generated by different PQCs using a learning rate of 1e-3

5 Issues with Reproducibility

For this project, we faced several issues in reproducing the results from the original paper M. S. Rudolph, Miller, et al. 2022.

Firstly, the framework proposed in the paper comprised of many components from different papers such as the Tensor Network Born Machine (TNBM) from (Han et al. 2018), Quantum Circuit Born Machine (QCBM) from (Benedetti et al. 2019), MPS to PQC mapping from (M. S. Rudolph, Chen, et al. 2022), and several more. This required us to dive into numerous papers both from the quantum computing and tensor network literature, which made it quite technically challenging to fully understand and implement the framework.

Secondly, in order to training our models, we needed to ensure numerical stability when calculating the KL divergence. For calculating the KL divergence, we needed to specify a clipping value for $q_\theta(x)$ when $q_\theta(x)$ is 0. This clipping value has to be close to 0. The choice of the clipping value vastly affects the total magnitude of the loss and convergence of the models. However, the exact choice of this hyperparameter was not specified in the original paper.

Thirdly, we believe this is to one of the main reasons why the results above differ from that of the original paper. In (M. S. Rudolph, Miller, et al. 2022), the authors use the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) optimizer for optimizing their quantum circuits, which is a gradient-free optimizer that is based on an adaptive evolutionary strategy. Since, we used PennyLane which has better integration with automatic differentiation than gradient-free methods, in our work we used the Adam optimizer which led to significantly different results.

Additionally, for all the models we trained, we had to initialize parameters for the SU(4) gates. These initializations were done randomly and the authors of the paper did not mention what these initializations were.

And finally, We were not able to decompose the unitary matrices generated by MPS for $\chi > 2$ and therefore could not train PQCs using MPS with $\chi > 2$ and produce comparisons for those results in the or.

6 Conclusion

Overall, the work (M. Rudolph et al. 2022) is creative and tries to utilize classical resources to improve initialization strategies for PQCs. It attempts to solve the problem of barren plateaus and uncertainties in the optimization of local minima.

Our training models were able to generate some bars and stripes datasets. The KL divergence plots, however, did not match what the authors of the paper had. The extended circuit of $\chi = 2$ MPS should have performed better than the randomly and near unitary initialized PQCs. Reproducibility issues including different notations being used in papers referenced in (M. Rudolph et al. 2022), random initialization of parameters, unitary matrices not being decomposable for $\chi > 2$, and a different optimizer used for training all contributed to this.

Contributions

Abhishek Abhishek, 29706215

- **Report:** Wrote the Abstract, Section 2.1, (majority of Sections 2.2 and 2.3), Section 3.1, 3.2, 3.3 and parts of section 3.4
- **Code:** Wrote and implemented all the functions in the following files: metrics.py, mps_circuit_helpers.py, mps_circuits.py, mps_helpers.py to implement the functionality discussed in Section 3.3, and several of the development/testing notebooks in sandbox
- **Presentation:** Slides on MPS to PQC, and Code overview
- **Theory:** Worked through foundational tensor network theory (Ballarin 2021; Perez-Garcia et al. 2006) and through the MPS to PQC mapping theory in (Barratt et al. 2021; Dborin et al. 2022; Ran 2020; M. S. Rudolph, Chen, et al. 2022)

Daniel Kong, 68133677

- **Code:** Explored QCBMs and worked on training expanded PQCs and tried different circuits to try to get a better loss (in sandbox) and tried ways to optimize unitaries in QCBM
- **Theory:** Researched QCBMs (Benedetti et al. 2019)
- **Presentation:** Worked on introduction, problem description parts of the presentation
- **Report:** Wrote Abstract, Introduction, QCBM (2.3), helped write 3, 4, edited overall report, developed figures for report

Harmeeta Dahiya, 78598497

- **Theory:** Explored Tensor Networks and Matrix Product States
- **Theory:** Researched MPS to PQC mapping by looking into isometries, isometry to unitary conversion, and multi-qubit unitary to two-qubit unitary conversion mentioned in (M. S. Rudolph, Chen, et al. 2022)
- **Presentation:** Worked on Overview and MPS training parts of Presentation
- **Report:** Created unique figures and wrote and edited several sections for the final report (Introduction, Isometry to Unitary, QCBM, Issues with Reproducibility, Conclusion)

Mushahid Khan, 501068444

- **Report:** Wrote sections: Results, Issues with Reproducibility, and Conclusion
- **Code:** Wrote and implemented code necessary for creating and training the randomly initialized and near identity initialized QCBM models, extending the circuits generated by MPS with different topologies and training them using the following files: random_and_near_identity_circuit.py, random_near_identity_extended_circuit.py, random_near_identity_extended_circuit.ipynb. Incorporated JAX, created some of the development/testing notebooks in sandbox, worked on the repository sandbox folder ReadMe.md
- **Presentation:** Slides on results and future work, as well as demo presentation

Appendix A: MPS to PQC

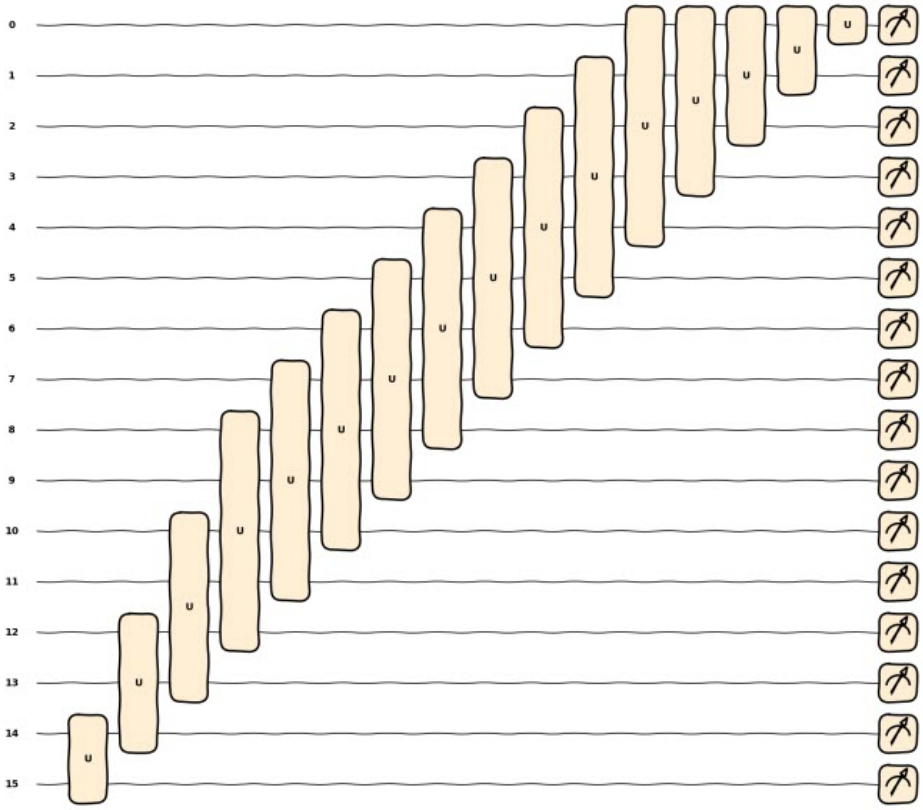

Figure 13: MPS to quantum circuit with multi-qubit unitaries

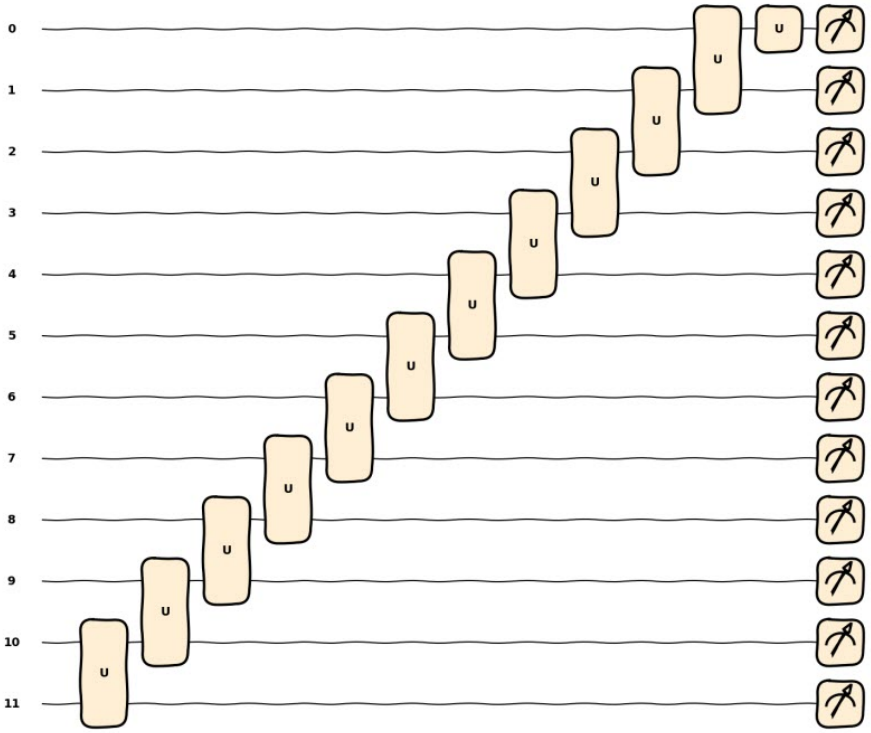

Figure 14: MPS to quantum circuit with two-qubit unitaries

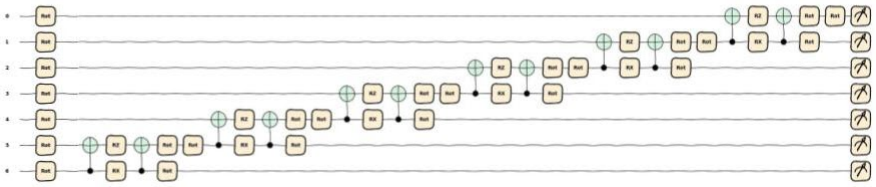

Figure 15: KAK Decomposition of Two-Qubit Unitaries

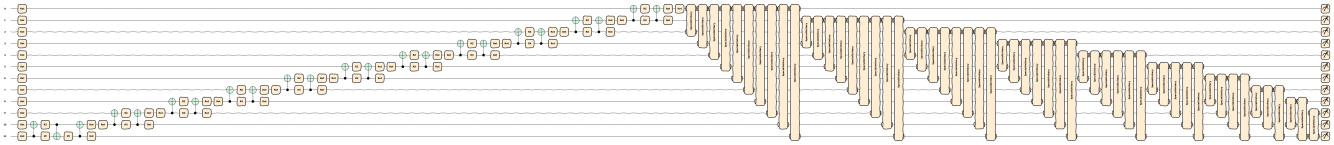

Figure 16: MPS Chi = 2 Extended Circuit

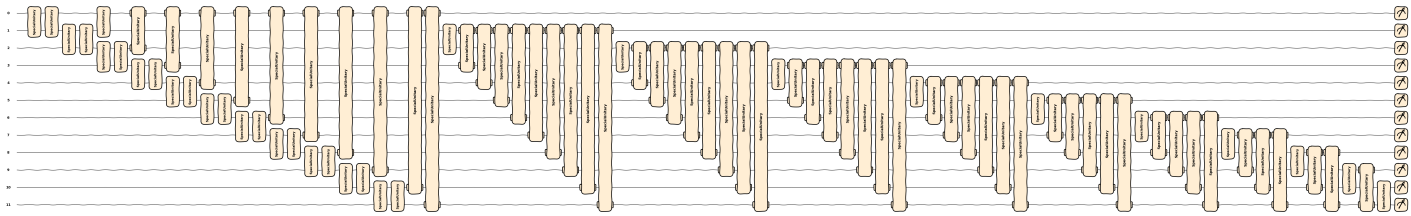

Figure 17: Random Initialization and Near Identity Initialization Circuit Without MPS Extension

References

Ballarin, Marco (2021). "Quantum Computer Simulation via Tensor Networks." In.

Barratt, Fergus, James Dborin, Matthias Bal, Vid Stojevic, Frank Pollmann, and Andrew G Green (2021). "Parallel quantum simulation of large systems on small NISQ computers." In: *npj Quantum Information* 7.1, p. 79.

Benedetti, Marcello, Delfina Garcia-Pintos, Oscar Perdomo, Vicente Leyton-Ortega, Yunseong Nam, and Alejandro Perdomo-Ortiz (2019). "A generative modeling approach for benchmarking and training shallow quantum circuits." In: *npj Quantum Information* 5.1, p. 45.

Bergholm, Ville, Josh Izaac, Maria Schuld, Christian Gogolin, Shahnawaz Ahmed, Vishnu Ajith, M Sohaib Alam, Guillermo Alonso-Linaje, B AkashNarayanan, Ali Asadi, et al. (2018). "Pennylane: Automatic differentiation of hybrid quantum-classical computations." In: *arXiv preprint arXiv:1811.04968*.

Coyle, Brian, Daniel Mills, Vincent Danos, and Elham Kashefi (2020). "The Born supremacy: quantum advantage and training of an Ising Born machine." In: *npj Quantum Information* 6.1, p. 60.

Dborin, James, Fergus Barratt, Vinul Wimalaweera, Lewis Wright, and Andrew G Green (2022). "Matrix product state pre-training for quantum machine learning." In: *Quantum Science and Technology* 7.3, p. 035014.

Han, Zhao-Yu, Jun Wang, Heng Fan, Lei Wang, and Pan Zhang (2018). "Unsupervised generative modeling using matrix product states." In: *Physical Review X* 8.3, p. 031012.

McClean, Jarrod R, Sergio Boixo, Vadim N Smelyanskiy, Ryan Babbush, and Hartmut Neven (2018). "Barren plateaus in quantum neural network training landscapes." In: *Nature communications* 9.1, p. 4812.

Orús, Román (2014). "A practical introduction to tensor networks: Matrix product states and projected entangled pair states." In: *Annals of physics* 349, pp. 117–158.

Perez-Garcia, David, Frank Verstraete, Michael M Wolf, and J Ignacio Cirac (2006). "Matrix product state representations." In: *arXiv preprint quant-ph/0608197*.

Ran, Shi-Ju (2020). "Encoding of matrix product states into quantum circuits of one-and two-qubit gates." In: *Physical Review A* 101.3, p. 032310.

Rudolph, Manuel, Jacob Miller, Jing Chen, Atithi Acharya, and Alejandro Perdomo-Ortiz (2022). "Synergy Between Quantum Circuits and Tensor Networks:Short-cutting the Race to Practical Quantum Advantage." In: *arXiv preprint arXiv:2208.13673*.

Rudolph, Manuel S, Jing Chen, Jacob Miller, Atithi Acharya, and Alejandro Perdomo-Ortiz (2022). "Decomposition of matrix product states into shallow quantum circuits." In: *arXiv preprint arXiv:2209.00595*.

Rudolph, Manuel S, Jacob Miller, Jing Chen, Atithi Acharya, and Alejandro Perdomo-Ortiz (2022). "Synergy between quantum circuits and tensor networks: Short-cutting the race to practical quantum advantage." In: *arXiv preprint arXiv:2208.13673*.

Shirakawa, Tomonori, Hiroshi Ueda, and Seiji Yunoki (2021). "Automatic quantum circuit encoding of a given arbitrary quantum state." In: *arXiv preprint arXiv:2112.14524*.

Stoudenmire, Edwin and David J Schwab (2016). "Supervised learning with tensor networks." In: *Advances in neural information processing systems* 29.