# UNIVERSITY OF BRITISH COLUMBIA

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


*Synergy between Quantum Circuits and Tensor Networks*

Abhishek Abhishek, Daniel Kong, Harmeeta Dahiya, Mushahid Khan

# Synergy between Quantum Circuits and Tensor Networks

Abhishek Abhishek, Daniel Kong, Harmeeta Dahiya, Mushahid Khan

April 15, 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 which 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. In this paper, the idea of task specific initialization for the PQC is explored and applied to Quantum Machine Learning (QML).

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. 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 $\chi$. Icreasing $\chi$ implies use of more classical computing resources. This project compares the loss function for PQC initializations with classically trained MPS, randomly initialized PQCs, and unitary initialized PQCs. MPS with higher $\chi$ should give better performance. This is intuitively visualized in Figure 2.

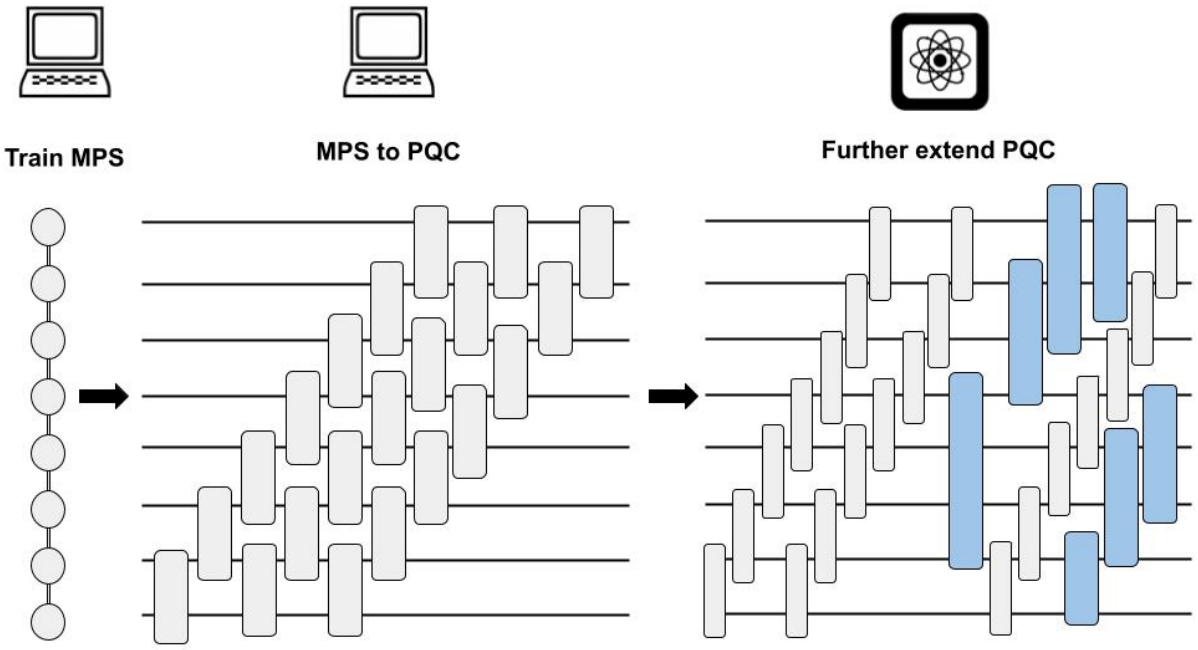


Figure 1: Structure of the training solution utilizing both TNs and PQCs

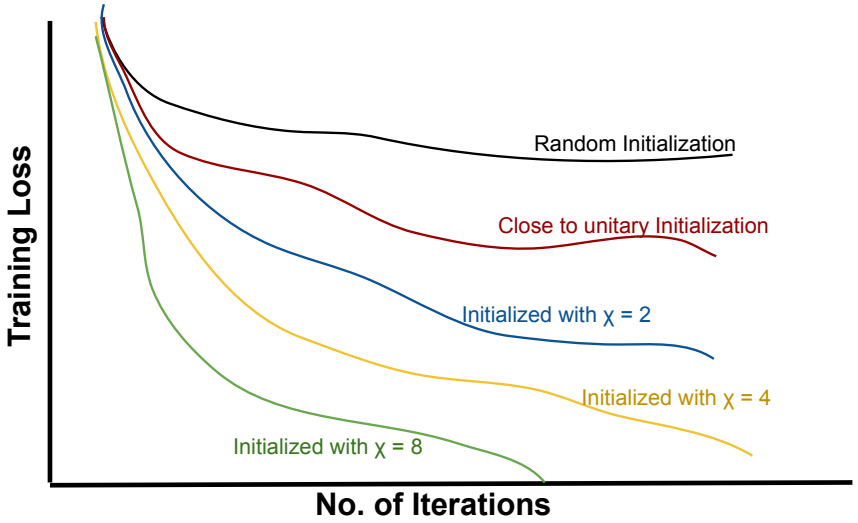


Figure 2: Correlation between different types of PQC initializations and Loss

# 2 Theory

## 2.1 Introduction to 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 $\nu_\alpha$ 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 product of two matrices $C = \mathrm{Tr}(AB)$ respectively. Note that in the notation we use, the superscripts correspond to a "virtual" indices and the subscripts correspond to "real" indices respectively.

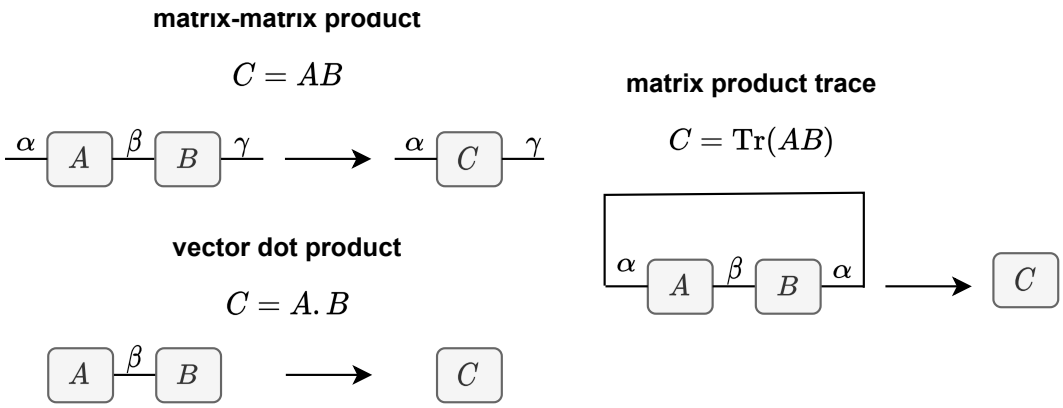


Figure 3: Intro to Tensor Networks

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-neighbour 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 $\alpha_j$ goes from 1 to $\chi_j$ where $\chi_j$ 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 to approximately simulate quantum systems far beyond what are 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).

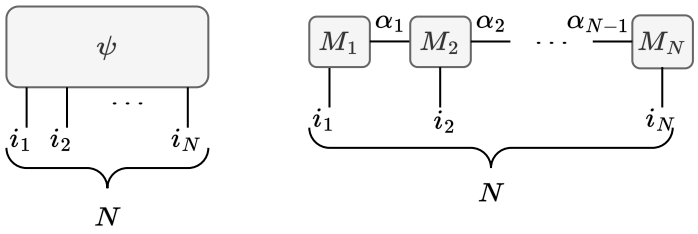


Figure 4: MPS Representation

## 2.2 Mapping MPS tensor networks to Parameterized Quantum Circuits

One of the main focus 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 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 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 circuits 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.

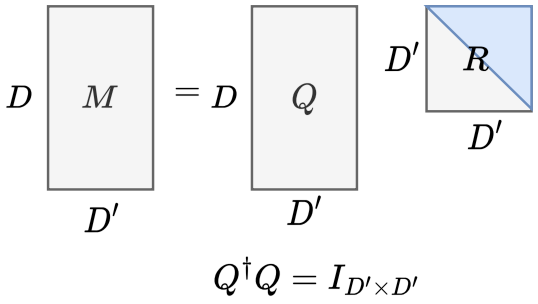


Figure 5: QR-factorization of matrix M

**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.

A MPS is said to be in left-canonical form if all it's 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.

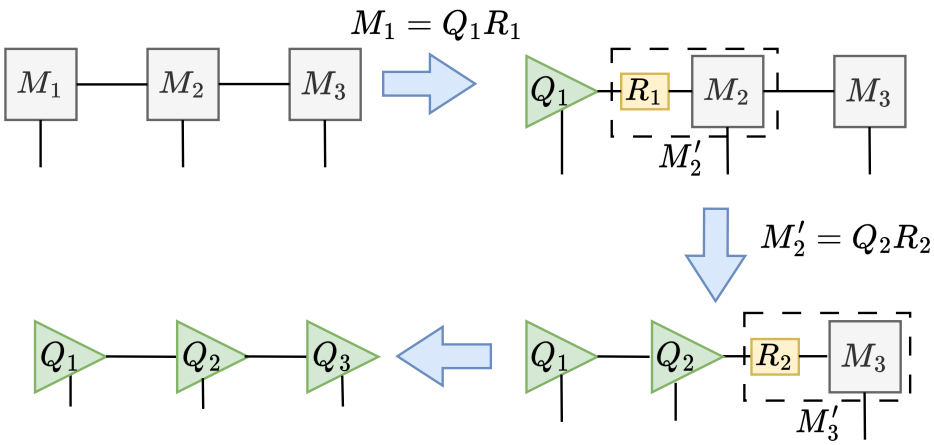


Figure 6: Left Cannonicalization of MPS tensor network

**MPS isometries to unitaries** Once the MPS is in the left-canonical form via the above procedure, the next step is 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) $\chi_{left}$ and $\chi_{right}$ are padded to powers of two (M. S. Rudolph et al. 2022, Section 2B.) before performing left-canonicalization.

## 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

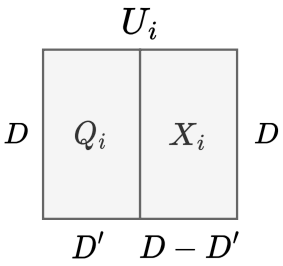


Figure 7: Left-isometry $Q_i$ to unitary $U_i$

the born rule to parameterize classical probabilities and are capable of representing complicated probability distributions (Coyle et al. 2020). If we consider a binary string $\mathbf{x}$, its probability under the model distribution using Born's rule is given by

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

where $\theta$ is the quantum parameter, and $\psi_\theta$ is the associated parameterized wave function. We use the Kullback-Leibler (KL) divergence as the loss function between the QCBM distribution and the target probability distribution. The QCBM aims to minimize the loss function of a problem based on the dataset $D$. The loss function is defined as:

$$L(\theta) = KL\ divergence = -log(|D|) - \frac{1}{|D|} \sum_{x \in |D|} log(q_\theta(x)) \tag{10}$$

The QCBM represents the probability distributions through the following steps which can also be seen in Figure 8. The weights ($\theta$) are randomly initialized. The quantum circuit output is then compared to the reference data using the KL divergence. The learning algorithm then iteratively adjusts the parameters in order to minimize the loss function.

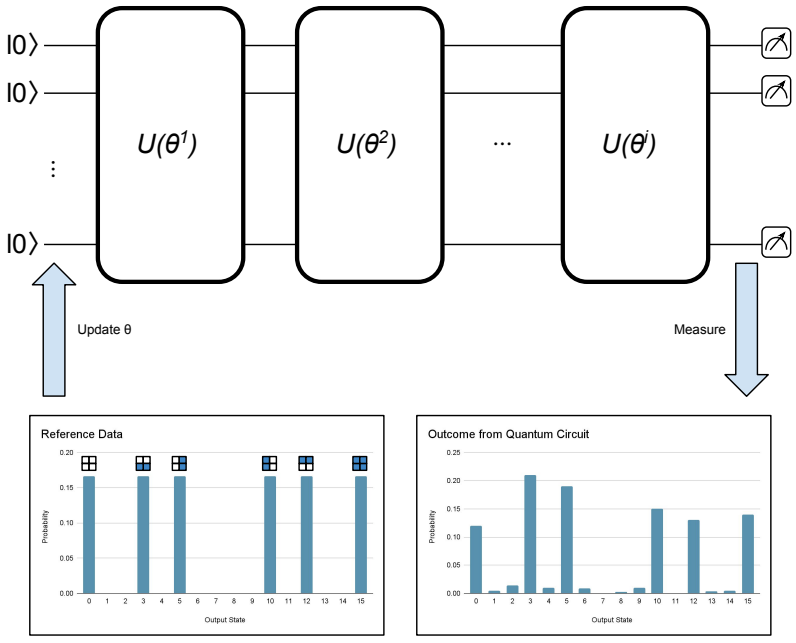


Figure 8: The QCBM training method.

# 3 Framework

The software implementation of the work is done in Python. We utilize Pennylane to create our circuits since these circuits are differentiable and hence their parameters are trainable. To train our quantum circuits, we use Adam Optimizer. Adam optimizer is a gradient-based optimizer which 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 excution, we used Google's JAX is a Just-In-Time (JIT).This acts as a machine learning framework.

**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**

- **Multi-qubit Unitary to 2-qubit Unitary** : Now the MPS consists of multi-qubit unitaries. We can use qml.QubitUnitary to convert these unitaries into a quantum circuit (see Figure 9).

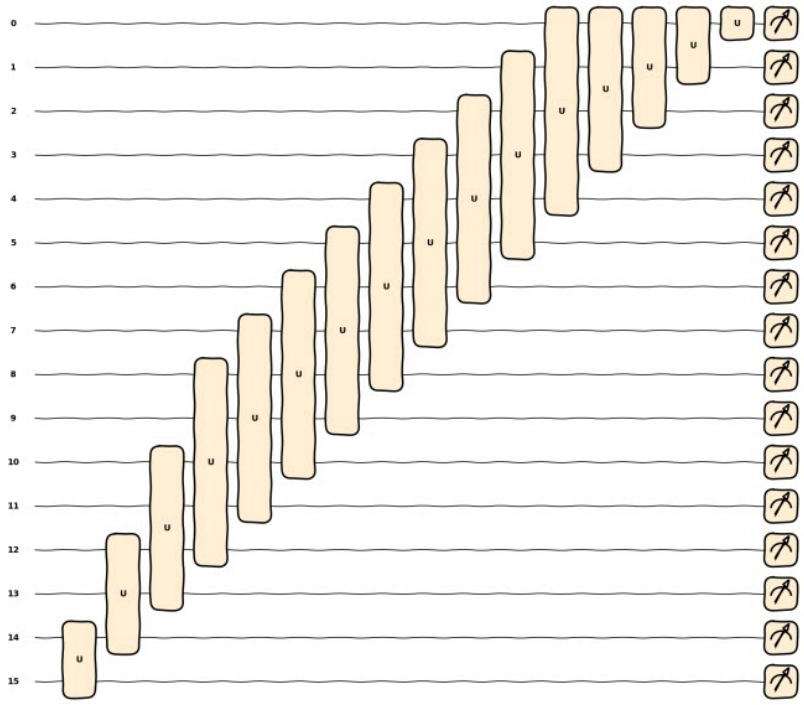


Figure 9: Multi-qubit unitaries to Quantum Circuit

However, this circuit will not be differentiable and hence trainable for our purposes. We can try to decompose the MPS into multiple layers of 2 qubit unitaries. This can be accomplished classically using the approach identified in (M. S. Rudolph et al. 2022), however, this step was not attainable due to the level of technical details regarding tensor networks involved.

- **Our Approach** : Fortunately, $\chi = 2$ MPS produces 2 qubit unitaries (see Figure 10), so we can use PennyLane's qml.QubitUnitary and a transform function to convert the unitaries into differentiable rotations and CNOT gates similiar to what is shown in Figure 11.

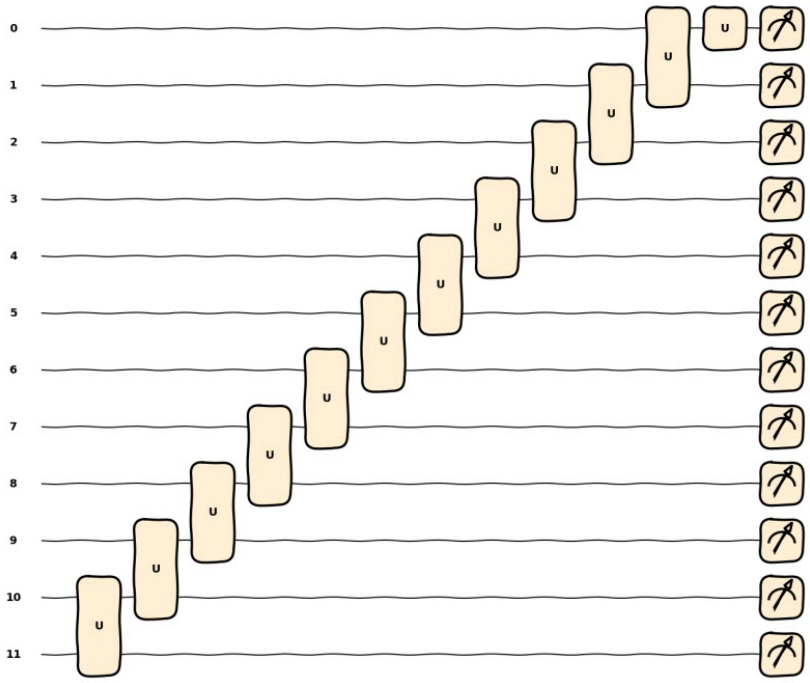


Figure 10: Two-qubit unitaries to Quantum Circuit

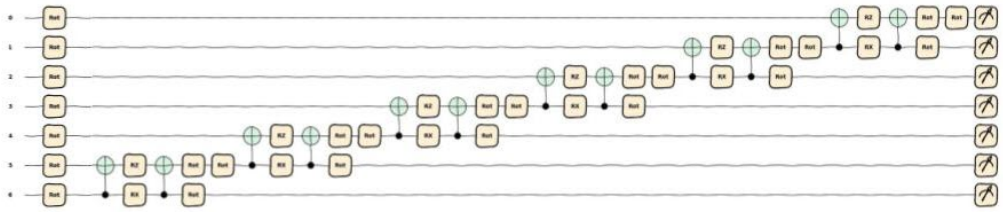

Figure 11: Two-qubit unitaries converted into quantum gates

# 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 the solution found by using MPS. The quantum circuit of PQCs initialized with random SU(4) and near identity SU(4)'s have three layers. The first two follow a staircase topology and the last one follows an all-to-all topology (as seen in Figure **??**). As mentioned above, we were able to generate a quantum circuit with trainable gates and parameters for $\chi = 2$ MPS.

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. These can be seen in Figure **??**, **??** and **??**.

The bars and stripes dataset generated by our training models for different PQC initializations can be seen in Figure 12. The respective KL divergence (loss function) for the three PQC initializations is shown in Figure 13 and Figure 14. The KL diveregence plots in the two figures (13, 14) are based on different learning rates.

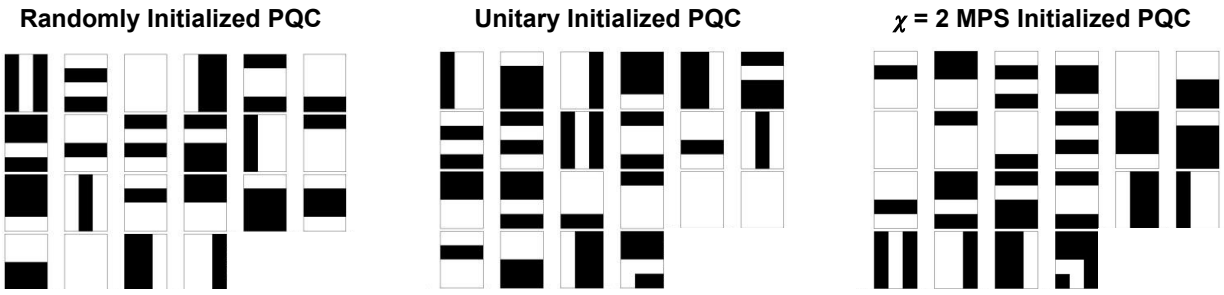


Figure 12: Bars and Stripes data generated by different PQCs

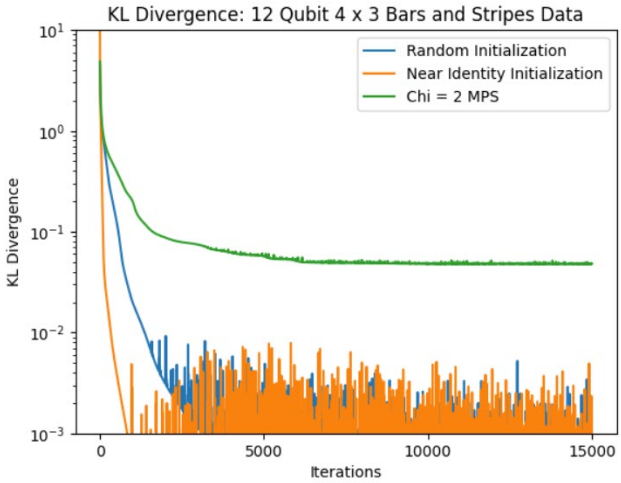


Figure 13: KL - divergence plot for different PQC initializations for learning rate = 0.01

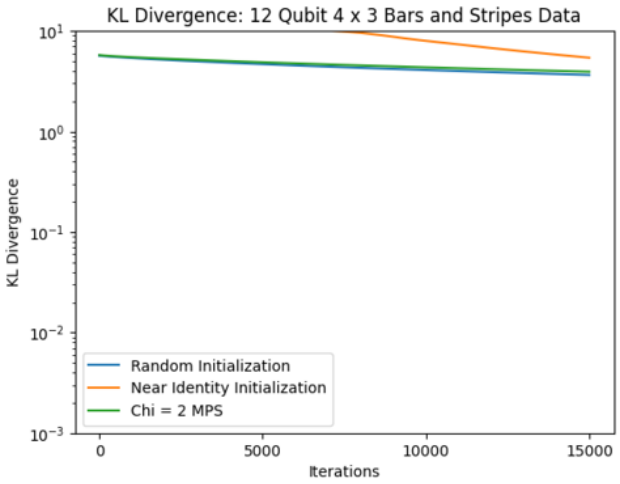


Figure 14: KL - divergence plot for different PQC initializations for learning rate = 1e-6

# 5 Issues with Reproducibility

For this project, we faced some issues in reproducing the results from the original paper. Here, we will go over those issues.

The framework proposed in the paper was comprised of parts from other papers. This required us to dive into more papers, each with its own notations. This made it difficult to implement the framework.

For training our models, we need to ensure numerical stability when calculating the KL diver-

gence. For calculating the KL divergence, we need 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.

The authors of the paper use covariance matrix adaptation evolution strategy (CMA-ES) optimizer for optimizing their quantum circuits. CMA-ES is a gradient-free optimizer which is based on an adaptive evolutionary strategy. We used the adam optimizer. Adam optimizer is a gradient-based optimizer which 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.

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.

We were not able to decompose the unitary matrices generated by MPS for $\chi > 2$ and so we could not train our model using $\chi > 2$ . Instead we worked on $\chi = 2$, which is mapped to two-qubit unitary matrices. This can be decomposed in Pennylane and turned into trainable gates and parameters.

# 6 Conclusion

Our training model generated some very useful datasets for bars and stripes problem. The KL divergence plot maintains the relation between random initialization and near identity initialization, where KL divergence for near identity PQC initialization converges faster than randomized PQC initializations. However, due the issues identified in section 5, the KL divergence plot for $\chi = 2$ MPS does not give the expected results we hoped for in comparison to Figure 2. Ideally, with specific values used for clipping value for $q_\theta(x)$ would have given us lower KL divergence plot for $\chi = 2$ MPS as compared to both nearly identity initialization and random initialization.

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 plateus and uncertainities in the optimization of local minima.

# Contributions

| Name | Student No. | Contributions |
|---|---|---|
| Abhishek Abhishek | item 11 | item 12 |
| Daniel Kong | 68133677 | - Explored QCBMs and worked on expanded PQC training<br>- Researched (Benedetti et al. 2019)<br>- Worked on introduction, problem description parts of presentation<br>- Created unique figures and wrote and edited several sections for the final report |
| Harmeeta Dahiya | 78598497 | - Explored Tensor Networks and Matrix Product States<br>- 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 et al. 2022)<br>- Worked on Overview and MPS training parts of Presentation<br>- Created unique figures and wrote and edited several sections for the final report |
| Mushahid Khan | item 41 | item 42 |

# 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.

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.

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*.

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.