QNT SCI 412: Final Presentation

June 14th, 2023

Tensor Networks and the QFT-MPO Algorithm

By: Ryan Dougherty, Payal Kaushik, Will Yzaguirre, and Arvind Gangadhar

Master of Quantum Science and Technology Program at UCLA

Outline

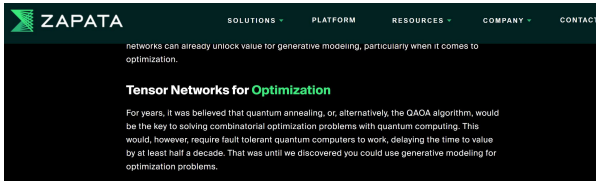
- Introduction and Summary of our Results (~8 mins)
  - Motivation - Why study Tensor Network Methods?
  - Summary of work conducted
  - Introducing the QFT-MPO Algorithm
- Methodology (10-15 mins)
  - Why the QFT has small entanglement (Arvind)
  - Bond Dimension and Tensor Network Fundamentals (Will)
  - Matrix Product States (MPS) for the QFT-MPO (Payal)
  - Matrix Product Operators (MPOs) for the QFT-MPO (Ryan)
- Results and Demo (~25 mins)
  - Implementations in Python: Ryan and Payal
  - Implementations in Julia: Will and Arvind

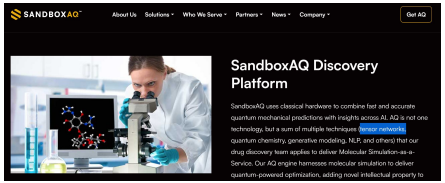
Motivation - Why study Tensor Network Methods?

- Tensor Network Methods form the basis for a large number of Quantum simulators
- All Quantum Circuits can be represented as Tensor Networks
- Tensor Network Methods have demonstrated new quantum-inspired classical algorithms (focus of our talk today)
- Tensor Network Methods represent a way to conduct linear algebraic operations in a high-dimensional space

"Tensor Network methods are currently the best known algorithms for simulating Random Circuit Sampling"

Sergio Boixo, Google Quantum AI talk at Simons Colloquium, April 25th 2023.





Motivation - Main Study

- The core study for our talk today utilizes Tensor Network Methods to make a few key arguments:
  - The core QFT Quantum circuit has low entanglement
  - The QFT-MPO, a quantum-inspired classical algorithm, can offer speedups over a classical FFT

Dec 2022

The Quantum Fourier Transform Has Small Entanglement

Jielun Chen,[1,2,*] E.M. Stoudenmire,[3] and Steven R. White[1]

[1] *Department of Physics and Astronomy, University of California, Irvine, CA 92697-4575, USA*
[2] *Department of Physics, California Institute of Technology, Pasadena, CA 91125, USA*
[3] *Center for Computational Quantum Physics, Flatiron Institute, 162 5th Avenue, New York, NY 10010, USA*

Summary of Work Conducted

Project Objectives

Build:
- QFT-MPO
- Bitwise-encoding of functions as MPS
- Apply MPO to MPS

Compare:
- Runtimes for FFT and QFT-MPO
- Demo real-life use case for FFT vs QFT-MPO

Understand:
- QFT-MPO
- 1D Tensor Networks (MPS and MPOs)
- SVD methods

Work Product Created

Models:
- QFT-MPO in Python; State prep as MPS in Python (Quimb)
- Above models in Julia (ITensor)

Results:
- Reproduced runtime results from paper up to 26 qubits
- Image blurring and deblurring demo using FFT and QFT-MPO

Project Milestones

Common Core:
- Penrose Notation, Tensor Contractions, MPS, MPO, Zip-up Algo

Individual focus areas:
- 4-qubit QFT-MPO and MPS recreation; FFT comparison; scale to higher number of qubits; Image as MPS; Blur and Deblur Models

9 meetings held with:
- Dr. Palsberg and Dr. Ross (UCLA), and Miles Stoudenmire (Flatiron Institute)

Skills acquired

Methods:
- QFT-MPO
- Bitwise-encoding of functions as MPS
- FFT, SVD

Languages:
- Quimb (Python)
- ITensor (Julia)

Other applications:
- TN methods for Quantum Simulation, Machine Learning
- FFT applications for Image Processing

The QFT Circuit Diagram

The QFT circuit is composed of two parts, a core recursive part Q_n, and a bit reversal part R_n.

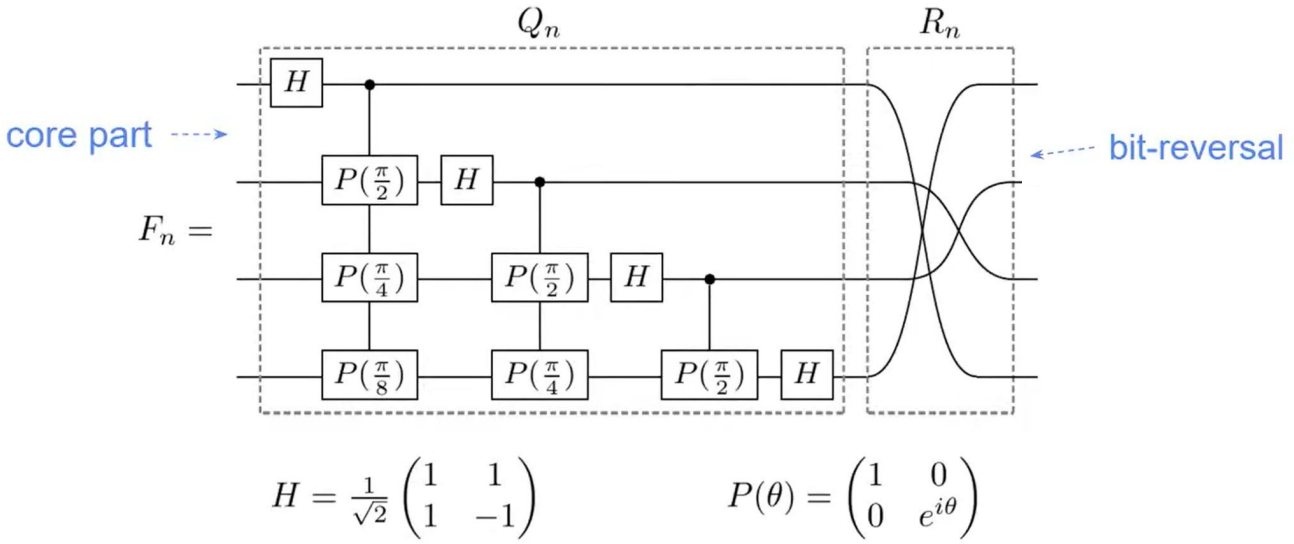


Image credits: Chris Chen, Caltech

Paper's Result: QFT-MPO vs FFT Runtime Comparison

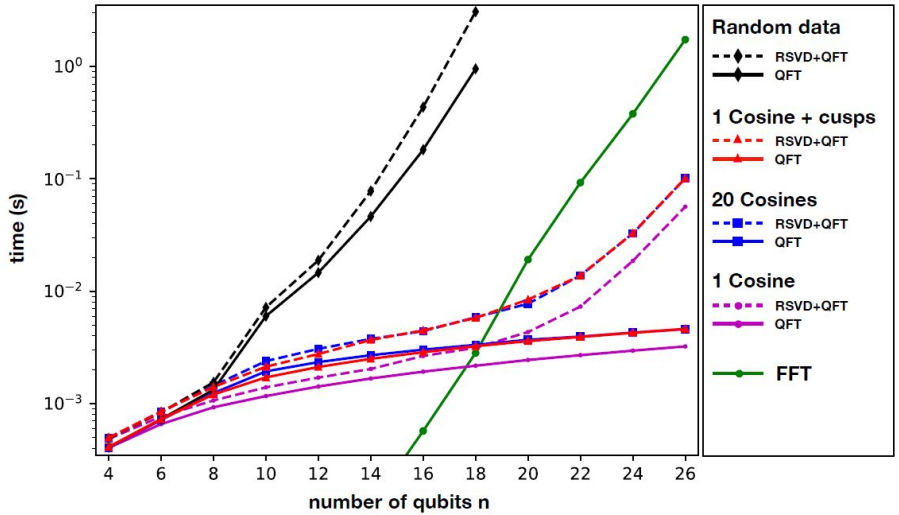


FIG. 4. Time in seconds to compute the discrete Fourier transform using either the fast Fourier transform (FFT) algorithm for data of size 2^n or the quantum Fourier transform (QFT) MPO acting on various functions represented as MPS with n sites (qubits). Dashed curves include the time to convert the data to the MPS format using a randomized SVD (RSVD) algorithm plus the application of the QFT, while solid curves are the time of the QFT step only. Timings were performed on a 2021 10-core Macbook Pro with an M1 Max processor, using the FFTW and ITensor software [36, 37].

Why the authors claim the QFT has low entanglement

The Schmidt Decomposition of the core QFT circuit Q_n has exponentially decaying coefficients and can be efficiently represented as a Tensor Network.

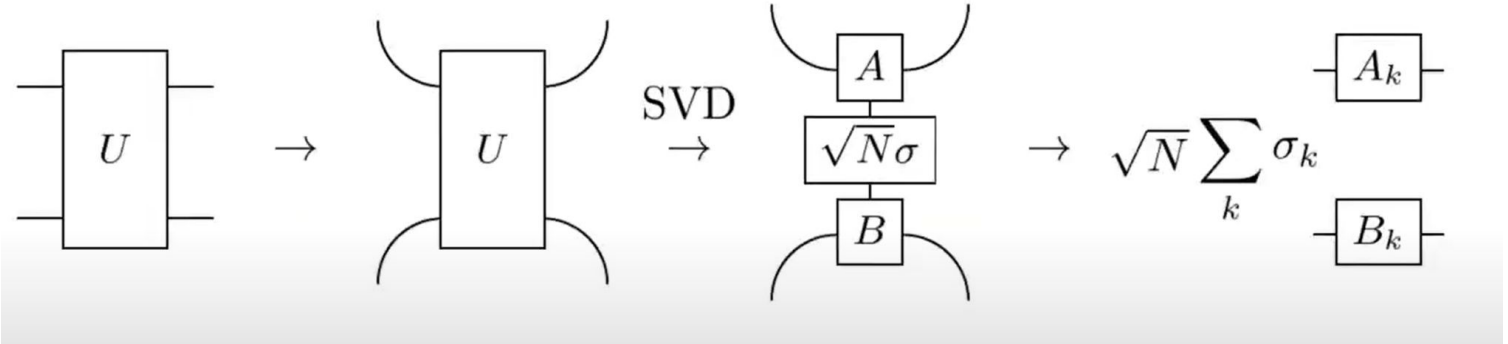


Image credits: Chris Chen, Caltech

Why the authors claim the QFT has low entanglement

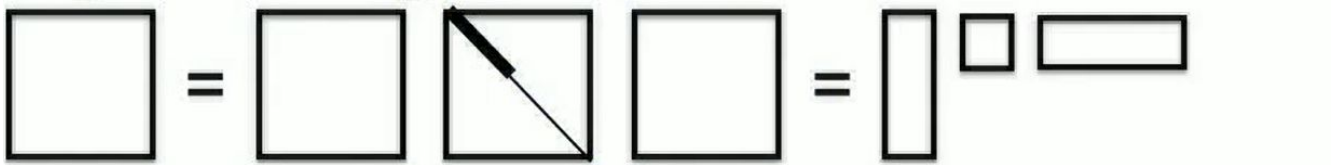
Most of the operator entanglement comes from the bit reversal part R_n. Similar to a SWAP gate, it has uniform Schmidt coefficients and therefore high operator entanglement.

$$\mathrm{SWAP} = \frac{1}{2} I \otimes I + \frac{1}{2} X \otimes X + \frac{1}{2} Y \otimes Y + \frac{1}{2} Z \otimes Z$$

Image credits: Chris Chen, Caltech

Simulating the core part of the QFT Circuit Classically

- Tensor Networks offer a way to compress a quantum circuit using multiple low rank approximations
- Singular Value Decomposition (SVD)
  - Given exponentially decaying Schmidt coefficients, it's possible to discard most singular values (lowest in value) and still retain a good approximation to the original circuit

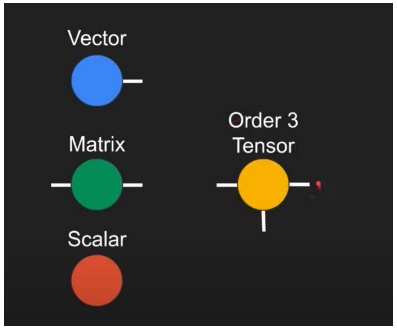


Visual depiction of a low rank approximation using a truncated SVD approach

Image credit: Steven R. White, UC Irvine.

What are Tensor Networks?

- Tensors = Generalized representations of vectors and matrices
- For our purposes - a multidimensional array
- Order of a tensor
  - Order 0 - Scalar
  - Order 1 - Vector
  - Order 2 - Matrix
  - Order 3 - Tensor
- Tensors Networks = representation for depicting a network of tensors

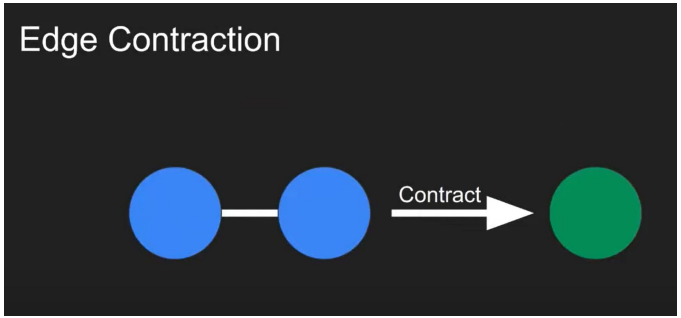


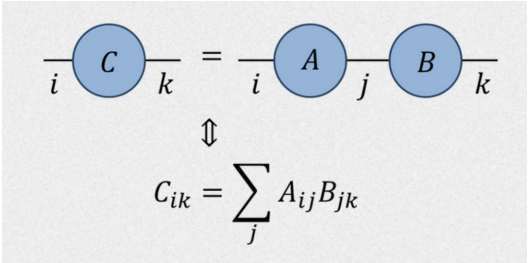
$$T = \begin{bmatrix} \begin{bmatrix} 5 & 4 \\ 3 & 2 \end{bmatrix} & \begin{bmatrix} 7 \\ 5 \end{bmatrix} \\ \begin{bmatrix} 3 & 1 \end{bmatrix} & \end{bmatrix} \qquad \begin{aligned} T_{111} &= 3 \\ T_{112} &= 5 \end{aligned}$$

Image Credits: Chase Roberts, Tensor Network Library in Python

Tensor Contractions

- A connected edge implies a contraction along an index (or indices) shared by two tensors





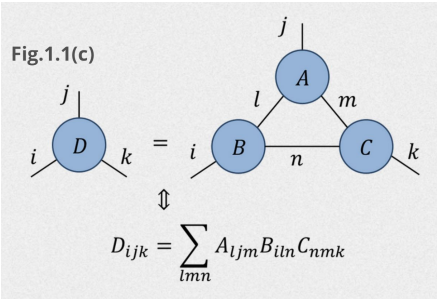
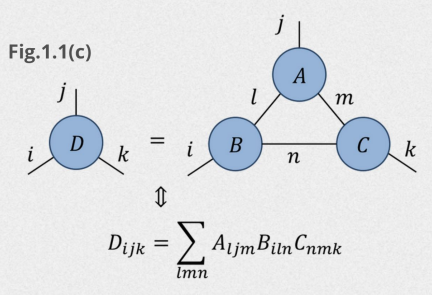


Image Credits: Tensors.Net

Contractions Pt.2



$$D_{ijk} = \sum_{l=0,m=0,n=0}^{3} A_j^{lm} B_{il}^{n} C_{nmk}$$

$$D_{ijk} = A_{0j0}B_{i00}C_{00k} + A_{1j1}B_{i11}C_{11k} + A_{2j2}B_{i22}C_{22k}$$

$$D_{000} = A_{000}B_{000}C_{000} + A_{111}B_{111}C_{111} + A_{222}B_{222}C_{222}$$

$$D_{010} = A_{010}B_{000}C_{000} + A_{111}B_{011}C_{110} + A_{212}B_{022}C_{220}$$

$$D_{101} = A_{000}B_{100}C_{001} + A_{101}B_{111}C_{111} + A_{202}B_{122}C_{221}$$

$$\vdots$$

Tensor Network Diagrams

- Quick Recap

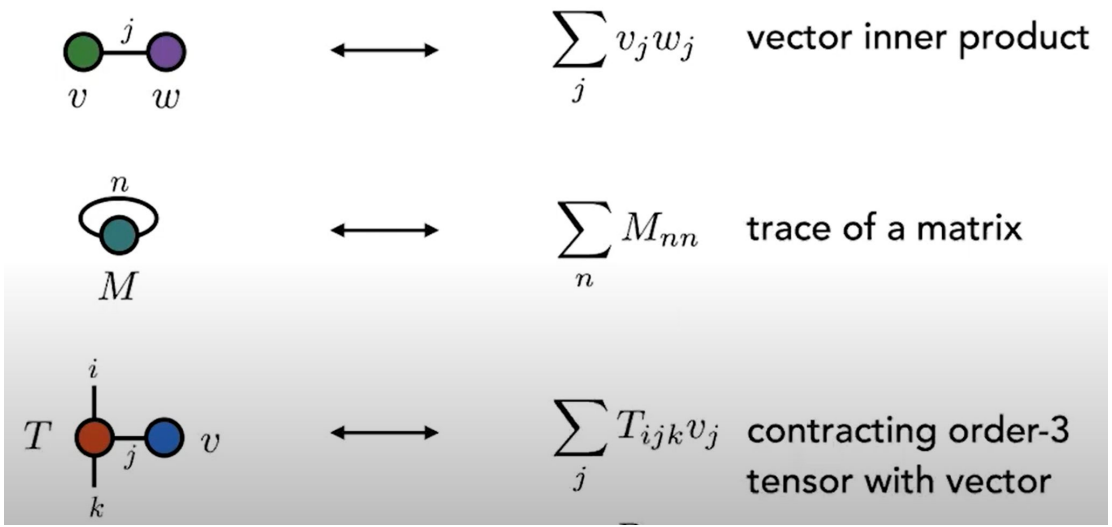
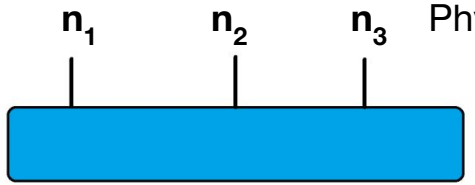


Image Credits: Tensors.Net

Tensor Networks in Quantum Information Science

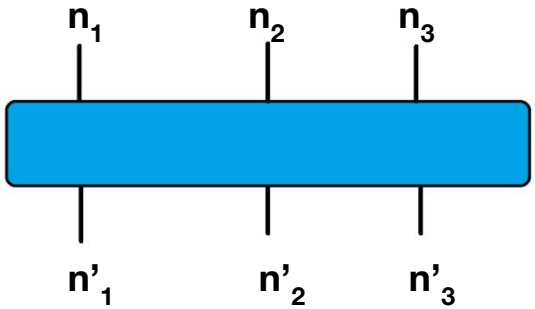
$$|\Psi\rangle = \sum_{n_1 n_2 n_3} \Psi^{n_1 n_2 n_3} |n_1 n_2 n_3\rangle = \Psi^{000}|000\rangle + \Psi^{001}|001\rangle + \Psi^{010}|010\rangle + \dots$$

where $|n\rangle = \{|\uparrow\rangle, |\downarrow\rangle\}$

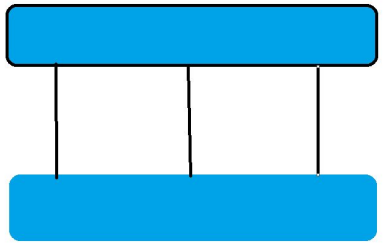


Physical Indices

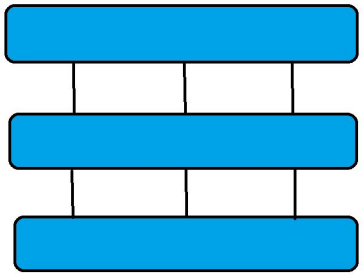
$$\hat{O} = \sum_{n,n'} O^{n_1 n_2 n_2}_{n'_1 n'_2 n'_3} |n_1 n_2 n_3\rangle\langle n'_1 n'_2 n'_3|$$



Penrose Notation Simplification



$$\langle \Psi | \Phi \rangle = \sum_{n} \Psi_{n_1 n_2 n_3} \Phi^{n_1 n_2 n_3}$$



$$\langle \Psi | O | \Phi \rangle = \sum_{n,n'} \Psi_{n_1 n_2 n_3} O^{n_1 n_2 n_3}_{n'_1 n'_2 n'_3} \Phi^{n_1 n_2 n_3}$$

Tensor Networks are the language of low-entanglement states

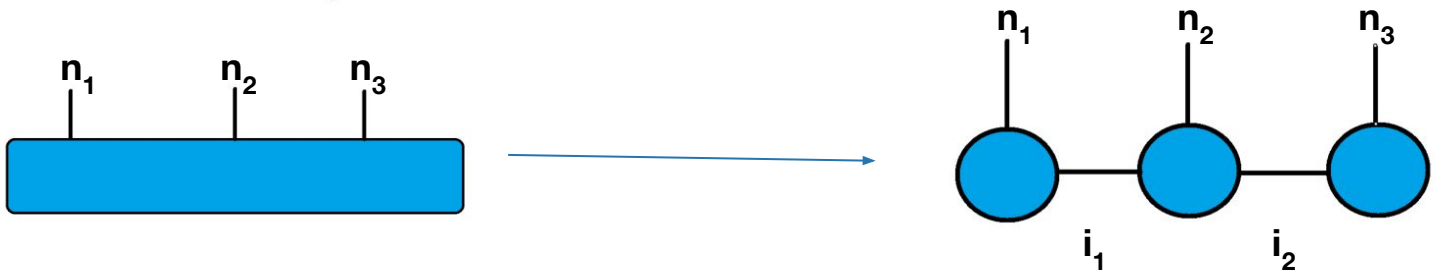
$$\Psi^{n_1 n_2} = A^{n_1} A^{n_2}$$

No entanglement state

$$\Psi^{n_1 n_2} = \sum_i A_i^{n_1} A_i^{n_2}$$

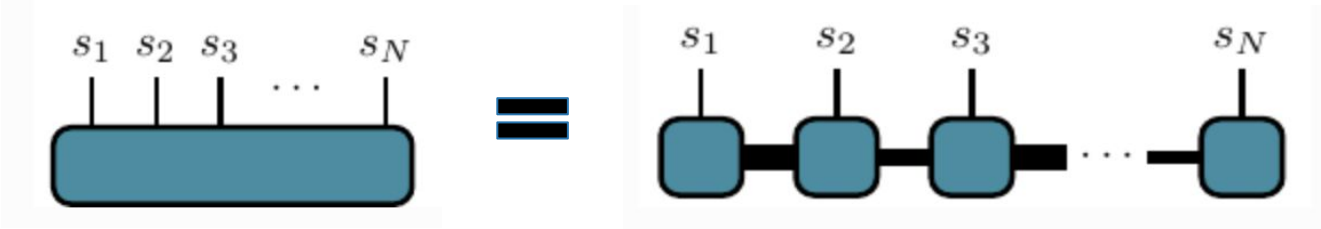
Entangled state

$$\Psi^{n_1 n_2 n_3 \ldots n_l} = \sum_i A_{i_1}^{n_1} A_{i_1 i_2}^{n_2} A_{i_2 i_3}^{n_3} \ldots A_{i_l}^{n_l} = A^{n_1} A^{n_2} A^{n_3} \ldots A^{n_l}$$



Matrix Product State (MPS)/ Tensor Train (TT)

A format of representing a tensor with **N** indices into a chain-like product of three-index tensors



N- dimensional tensor

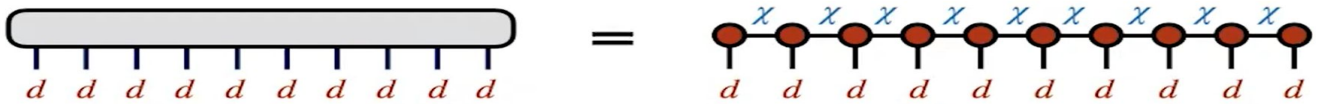
MPS with N tensors

$$T^{s_1 \dots s_N} = \sum_{\{\alpha\}} A^{s_1}_{\alpha_1} A^{s_2}_{\alpha_1 \alpha_2} A^{s_3}_{\alpha_2 \alpha_3} \cdots A^{s_{N-1}}_{\alpha_{N-2}\,\alpha_{N-1}} A^{s_N}_{\alpha_{N-1}}$$

Image credits: https://tensornetwork.readthedocs.io/en/latest/basic_mps.html

Why do we need an MPS ??

- Any arbitrary tensor can be represented in an MPS given a large enough bond dimension
- If bond dimension is small (or grows logarithmically), significant compression of original circuit is possible



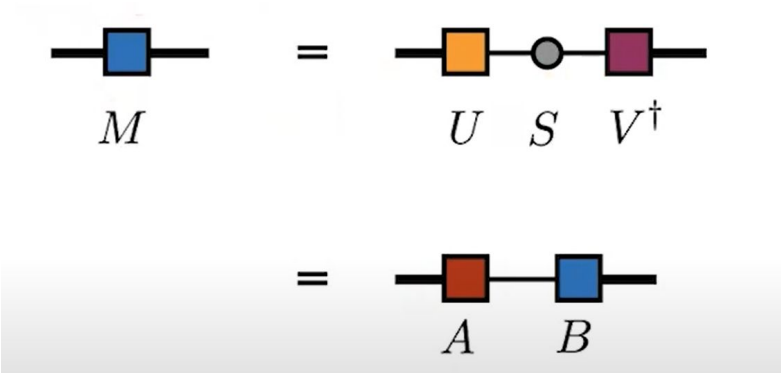
If modest χ yields good approximation,
obtain massive compression:

$$d^N \longrightarrow N d \chi^2$$

Image Credits: iTensor

How to prepare Matrix Product State (MPS) ?

- The essence of MPS resides in the **Singular Value Decomposition (SVD)**



M: Tensor/Matrix to be decomposed

U: Left Singular matrix

V: Right Singular matrix

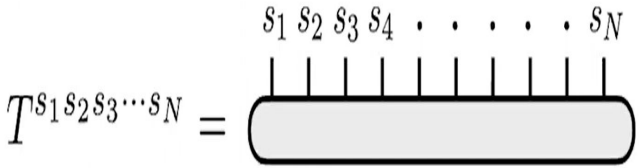
S: Singular value matrix

$$M = USV^{\dagger}$$

Image Credits: Tensors.Net

How to prepare Matrix Product State (MPS) ?

A tensor with **N** indices can be decomposed into two tensors using **SVD**



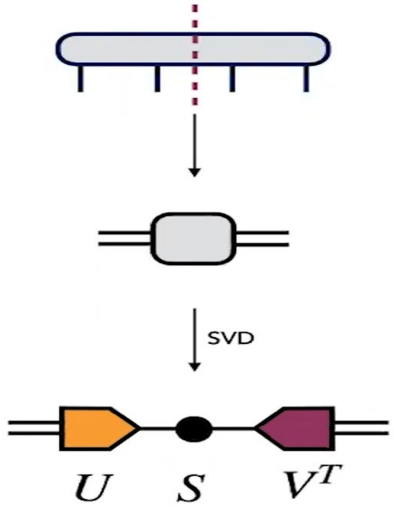


Image Credits: Tensors.Net

Preparing MPS from a 4 dimensional Tensor

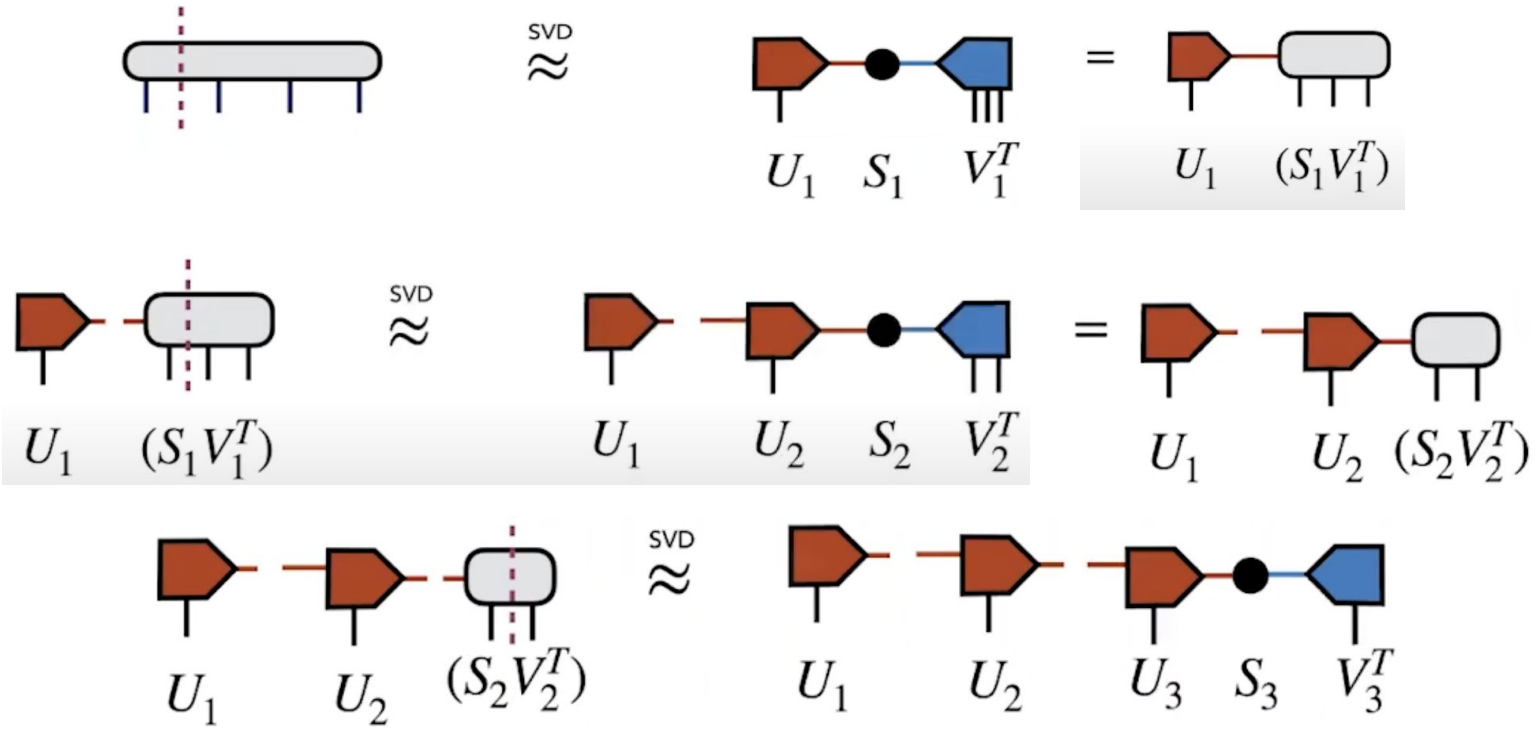


Image Credits: Tensors.Net

Preparing MPS from a 4 dimensional Tensor

- Converting a tensor representation of of n qubits as a Matrix Product State (MPS)

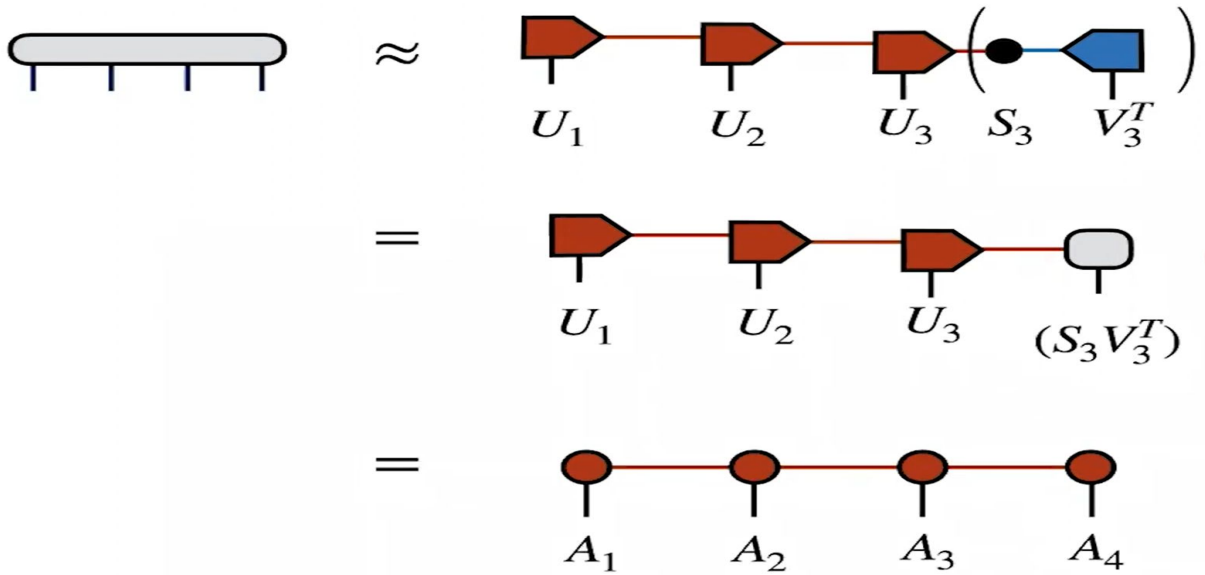
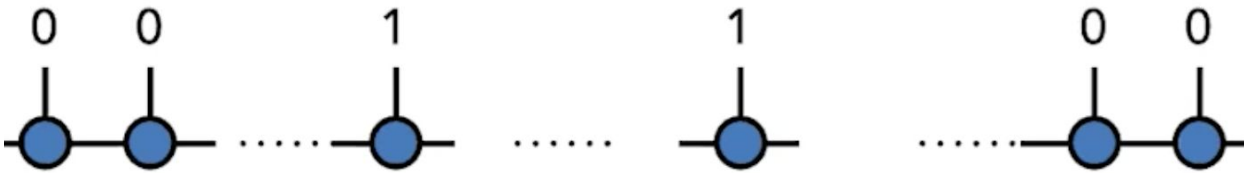


Image Credits: Tensors.Net

Encoding of Functions as MPS

- We bitwise encode the data to prepare the MPS.



- We create a finely spaced grid of spacing $1/2^n$
- Pass the data points to the corresponding function and get the final tensor which serves as an input to the DFT.
- Perform iterative svd on the n-dimensional tensor and get MPS

Image Credits: Tensors.Net

Code Implementation Results

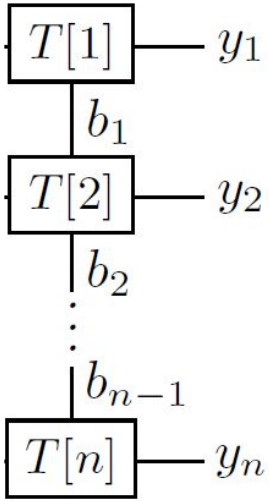
Code Demo 1:

Preparing MPS representations for functions in Python

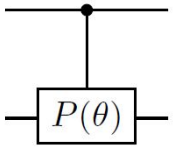
Matrix Product Operators

MPS -> MPO

- Analogous to MPS except that each tensor has one more input index to represent an operator instead of a state
- Can think of an MPO as an operator that acts on a state vector
- **Goal:** Transform the QFT into a MPO

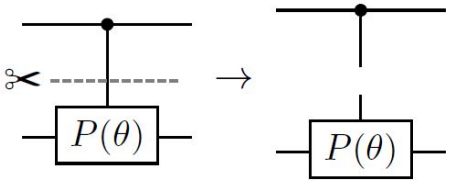


Quantum Circuits are Just Tensor Networks!

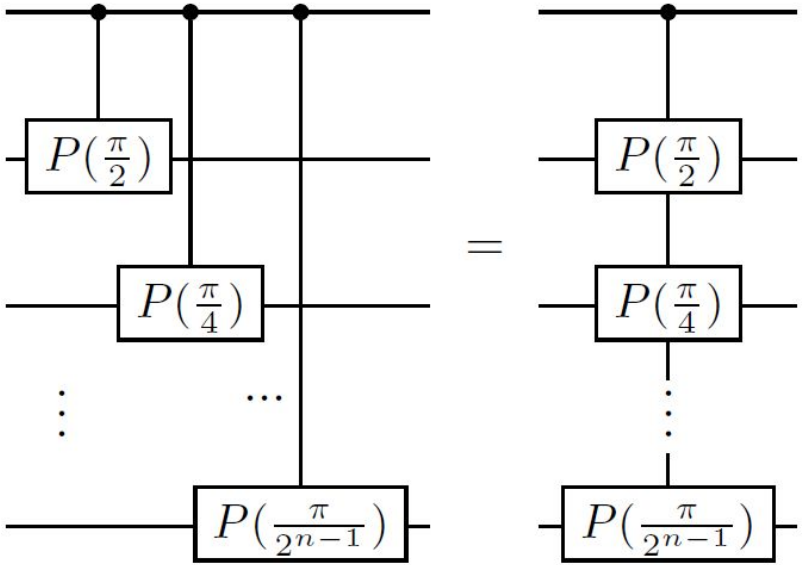


$$= |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes P(\theta)$$

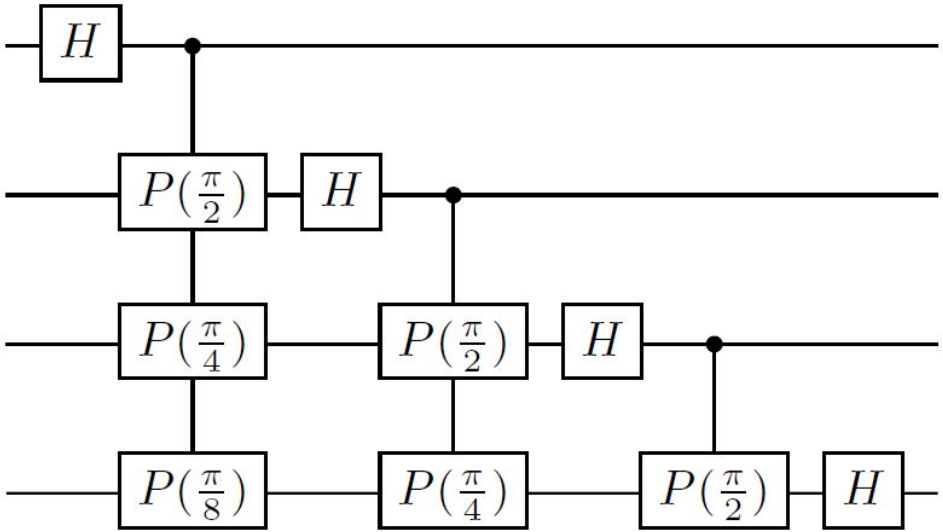
$$\begin{pmatrix} |0\rangle\langle 0| & |1\rangle\langle 1| \end{pmatrix} \begin{pmatrix} I \\ P(\theta) \end{pmatrix}$$



Creating a Tensor Network for QFT Circuit

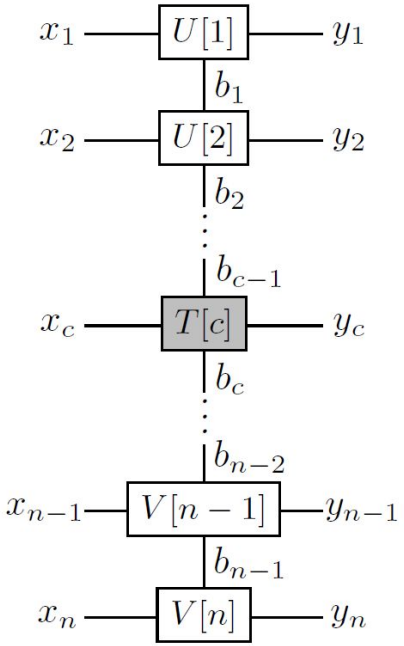


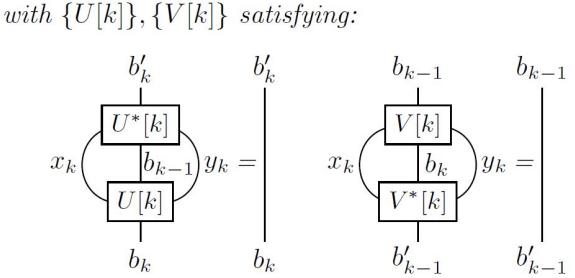
Creating a Tensor Network for QFT Circuit



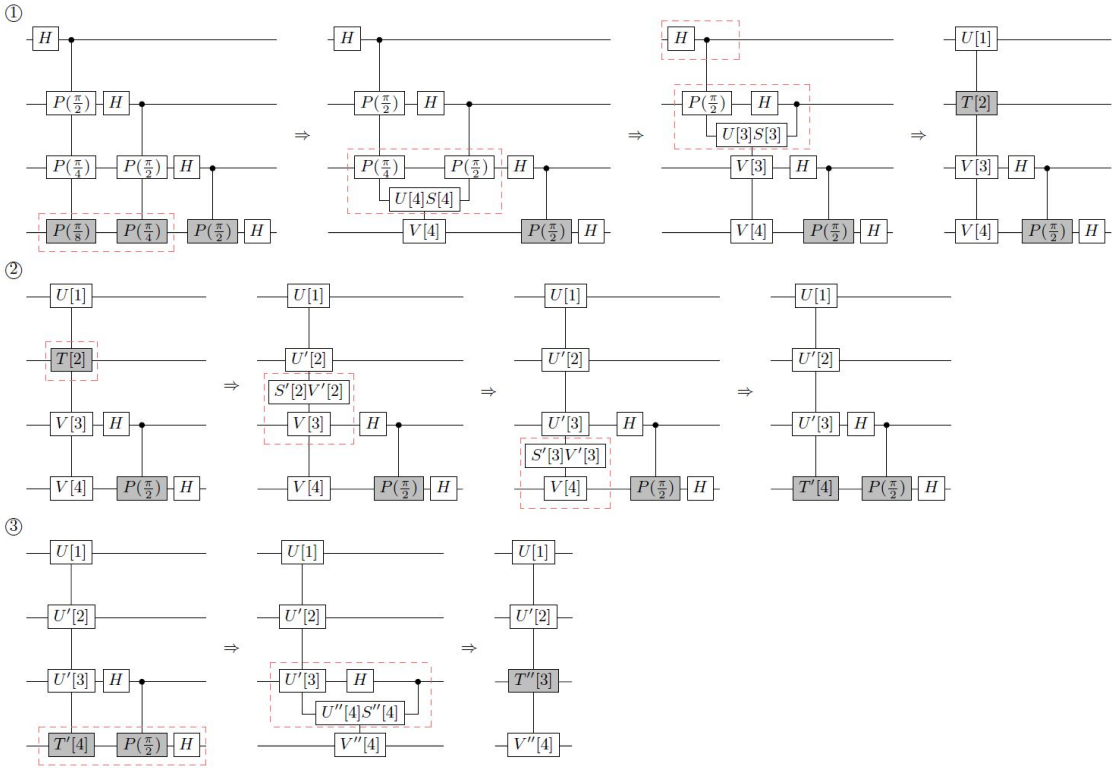
Building MPOs: SVD, and Controlling Site of Orthogonality

- With SVD we can **control** wher our singular values lie within ou MPO
- U and V tensors can be though of as our basis tensors
  - **U** are left singular value basis tensors
  - **V** are right singular value basis tensors
- **T** is site of orthogonality where our singular values lie
  - **Optimal** area to do contractions around





Creating a QFT MPO: The Zip-Up Algorithm



Code Implementation Results

Code Demo 2:

Implementation
of Zip-Up
Algorithm

QFT-MPO vs FFT Runtime Results (From Paper)

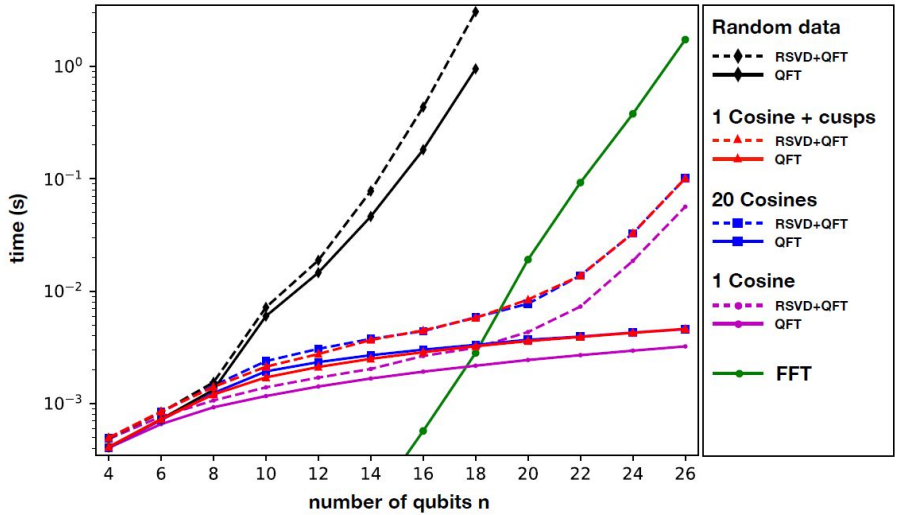
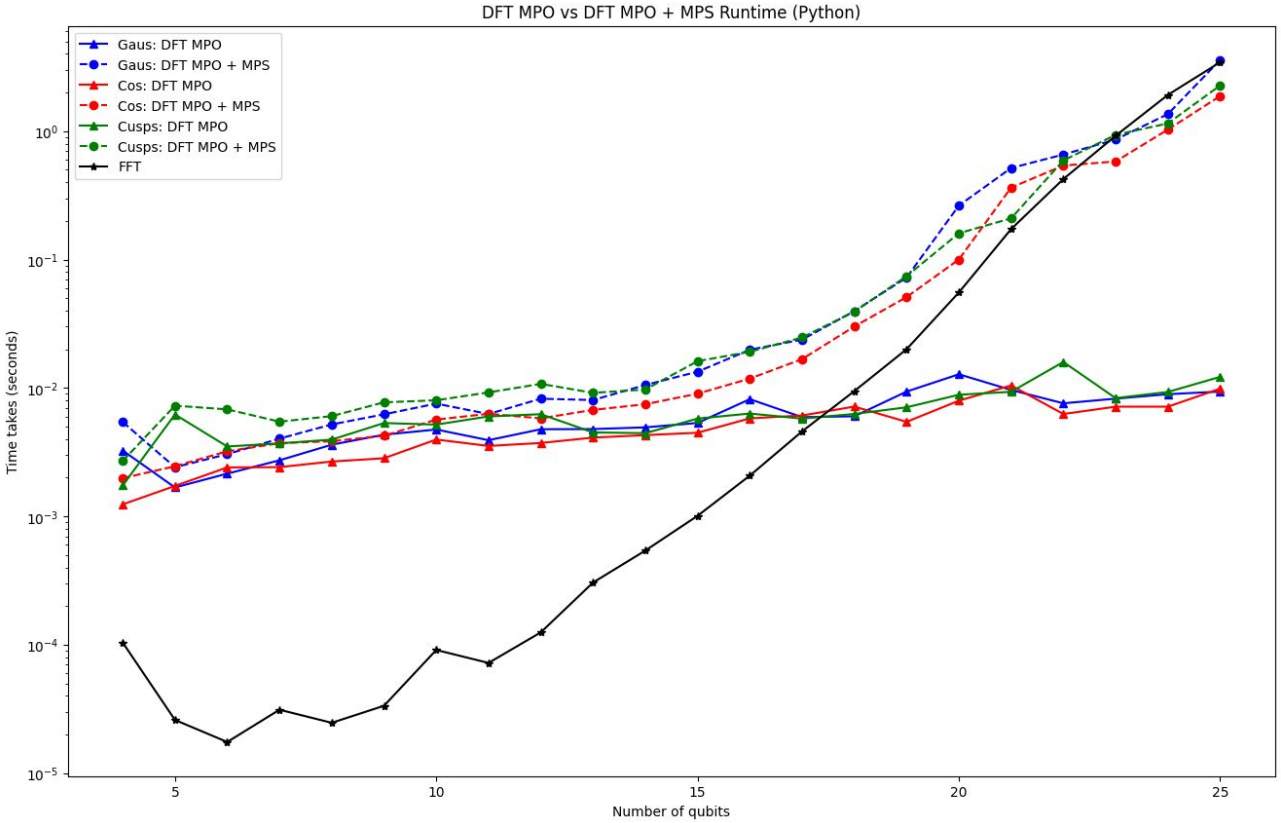


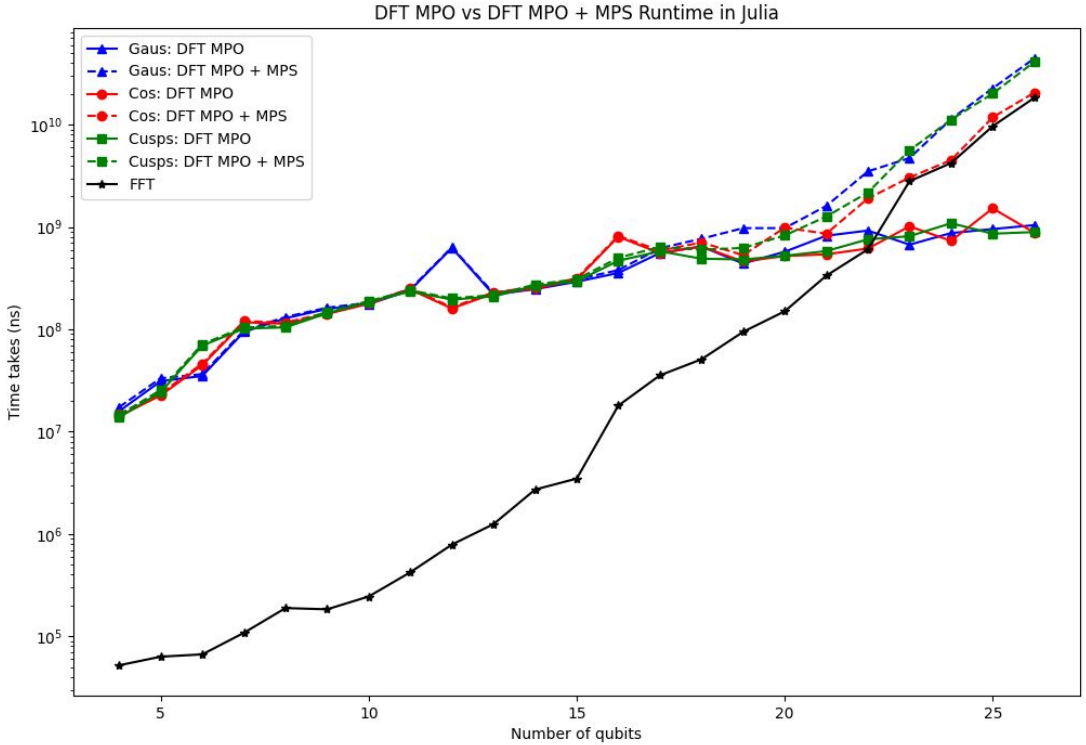
FIG. 4. Time in seconds to compute the discrete Fourier transform using either the fast Fourier transform (FFT) algorithm for data of size 2^n or the quantum Fourier transform (QFT) MPO acting on various functions represented as MPS with n sites (qubits). Dashed curves include the time to convert the data to the MPS format using a randomized SVD (RSVD) algorithm plus the application of the QFT, while solid curves are the time of the QFT step only. Timings were performed on a 2021 10-core Macbook Pro with an M1 Max processor, using the FFTW and ITensor software [36, 37].

Runtime Results for our model (Using Quimb in Python)



max_bond_mps=10,
max_bond_mpo=8,
cutoff_mps=1e-12,
cutoff_mpo=1e-12,

Runtime Results for our model (Using ITensor in Julia)



Checking our answers

$$e^{-(x-x_0)^2/\sigma^2}$$

```
n = 20
s = siteinds("Qubit", n)

x_not = 0.5
sigma = 0.1
f(x) = exp(-(x - x_not)^2 / sigma^2)
psi = function_to_mps(f, s, 0.0, 1.0; cutoff=1e-15)
```

✓ 0.5s

```
MPS
[1] ((dim=2|id=222|"Qubit,Site,n=1"), (dim=2|id=904|"Link,n=1"))
[2] ((dim=2|id=904|"Link,n=1"), (dim=2|id=416|"Qubit,Site,n=2"), (dim=4|id=35|"Link,n=2"))
[3] ((dim=4|id=35|"Link,n=2"), (dim=2|id=268|"Qubit,Site,n=3"), (dim=6|id=864|"Link,n=3"))
[4] ((dim=6|id=864|"Link,n=3"), (dim=2|id=78|"Qubit,Site,n=4"), (dim=8|id=818|"Link,n=4"))
[5] ((dim=8|id=818|"Link,n=4"), (dim=2|id=470|"Qubit,Site,n=5"), (dim=6|id=554|"Link,n=5"))
[6] ((dim=6|id=554|"Link,n=5"), (dim=2|id=355|"Qubit,Site,n=6"), (dim=5|id=929|"Link,n=6"))
[7] ((dim=5|id=929|"Link,n=6"), (dim=2|id=615|"Qubit,Site,n=7"), (dim=5|id=308|"Link,n=7"))
[8] ((dim=5|id=308|"Link,n=7"), (dim=2|id=718|"Qubit,Site,n=8"), (dim=4|id=306|"Link,n=8"))
[9] ((dim=4|id=306|"Link,n=8"), (dim=2|id=384|"Qubit,Site,n=9"), (dim=4|id=925|"Link,n=9"))
[10] ((dim=4|id=925|"Link,n=9"), (dim=2|id=391|"Qubit,Site,n=10"), (dim=3|id=679|"Link,n=10"))
[11] ((dim=3|id=679|"Link,n=10"), (dim=2|id=690|"Qubit,Site,n=11"), (dim=3|id=407|"Link,n=11"))
[12] ((dim=3|id=407|"Link,n=11"), (dim=2|id=750|"Qubit,Site,n=12"), (dim=3|id=53|"Link,n=12"))
[13] ((dim=3|id=53|"Link,n=12"), (dim=2|id=66|"Qubit,Site,n=13"), (dim=3|id=506|"Link,n=13"))
[14] ((dim=3|id=506|"Link,n=13"), (dim=2|id=286|"Qubit,Site,n=14"), (dim=2|id=928|"Link,n=14"))
[15] ((dim=2|id=928|"Link,n=14"), (dim=2|id=752|"Qubit,Site,n=15"), (dim=2|id=150|"Link,n=15"))
[16] ((dim=2|id=150|"Link,n=15"), (dim=2|id=176|"Qubit,Site,n=16"), (dim=2|id=222|"Link,n=16"))
[17] ((dim=2|id=222|"Link,n=16"), (dim=2|id=0|"Qubit,Site,n=17"), (dim=2|id=69|"Link,n=17"))
[18] ((dim=2|id=69|"Link,n=17"), (dim=2|id=898|"Qubit,Site,n=18"), (dim=2|id=591|"Link,n=18"))
[19] ((dim=2|id=591|"Link,n=18"), (dim=2|id=871|"Qubit,Site,n=19"), (dim=2|id=818|"Link,n=19"))
[20] ((dim=2|id=818|"Link,n=19"), (dim=2|id=316|"Qubit,Site,n=20"))
```

Checking our answers


```
f_psi = apply_dft_mpo(psi; cutoff=1e-15)
✓ 0.4s

MPS
[1] ((dim=2|id=222|"Qubit,Site,n=1"), (dim=2|id=151|"Link,n=19"))
[2] ((dim=2|id=416|"Qubit,Site,n=2"), (dim=2|id=151|"Link,n=19"), (dim=4|id=673|"Link,n=18"))
[3] ((dim=2|id=268|"Qubit,Site,n=3"), (dim=4|id=673|"Link,n=18"), (dim=4|id=735|"Link,n=17"))
[4] ((dim=2|id=78|"Qubit,Site,n=4"), (dim=4|id=735|"Link,n=17"), (dim=4|id=253|"Link,n=16"))
[5] ((dim=2|id=470|"Qubit,Site,n=5"), (dim=4|id=253|"Link,n=16"), (dim=4|id=968|"Link,n=15"))
[6] ((dim=2|id=355|"Qubit,Site,n=6"), (dim=4|id=968|"Link,n=15"), (dim=4|id=270|"Link,n=14"))
[7] ((dim=2|id=615|"Qubit,Site,n=7"), (dim=4|id=270|"Link,n=14"), (dim=4|id=74|"Link,n=13"))
[8] ((dim=2|id=718|"Qubit,Site,n=8"), (dim=4|id=74|"Link,n=13"), (dim=4|id=401|"Link,n=12"))
[9] ((dim=2|id=384|"Qubit,Site,n=9"), (dim=4|id=401|"Link,n=12"), (dim=4|id=235|"Link,n=11"))
[10] ((dim=2|id=391|"Qubit,Site,n=10"), (dim=4|id=235|"Link,n=11"), (dim=4|id=538|"Link,n=10"))
[11] ((dim=2|id=690|"Qubit,Site,n=11"), (dim=4|id=538|"Link,n=10"), (dim=4|id=213|"Link,n=9"))
[12] ((dim=2|id=750|"Qubit,Site,n=12"), (dim=4|id=213|"Link,n=9"), (dim=4|id=214|"Link,n=8"))
[13] ((dim=2|id=66|"Qubit,Site,n=13"), (dim=4|id=214|"Link,n=8"), (dim=4|id=888|"Link,n=7"))
[14] ((dim=2|id=286|"Qubit,Site,n=14"), (dim=4|id=888|"Link,n=7"), (dim=4|id=603|"Link,n=6"))
[15] ((dim=2|id=752|"Qubit,Site,n=15"), (dim=4|id=603|"Link,n=6"), (dim=4|id=888|"Link,n=5"))
[16] ((dim=2|id=176|"Qubit,Site,n=16"), (dim=4|id=888|"Link,n=5"), (dim=4|id=694|"Link,n=4"))
[17] ((dim=2|id=0|"Qubit,Site,n=17"), (dim=4|id=694|"Link,n=4"), (dim=4|id=788|"Link,n=3"))
[18] ((dim=2|id=898|"Qubit,Site,n=18"), (dim=4|id=788|"Link,n=3"), (dim=4|id=640|"Link,n=2"))
[19] ((dim=2|id=871|"Qubit,Site,n=19"), (dim=4|id=640|"Link,n=2"), (dim=2|id=308|"Link,n=1"))
[20] ((dim=2|id=308|"Link,n=1"), (dim=2|id=316|"Qubit,Site,n=20"))
```

Checking our answers

```
@show norm(fft(mps_to_discrete_function(psi)) / 2^(n/2) - mps_to_discrete_function(f_psi))
```

✓ 0.0s

```
norm(fft(mps_to_discrete_function(psi)) / 2 ^ (n / 2) - mps_to_discrete_function(f_psi)) = 9.459558950856061e-6
```

Real-life Use Case: Image Blurring and Deblurring

- X is the Image Matrix, B is the Blur Matrix, and Y is the Blurred Image
- Y = B * X requires matrix multiplication and is computationally intensive: O(n^3)
- So is the inverse calculation for the Deblurred image (X = B^-1 * Y)
- With the FFT, the convolution of two n-vectors is O(n log(n))
- Using the FFT and component-wise multiplication/division, matrix multiplication and inverse calculation can be avoided.

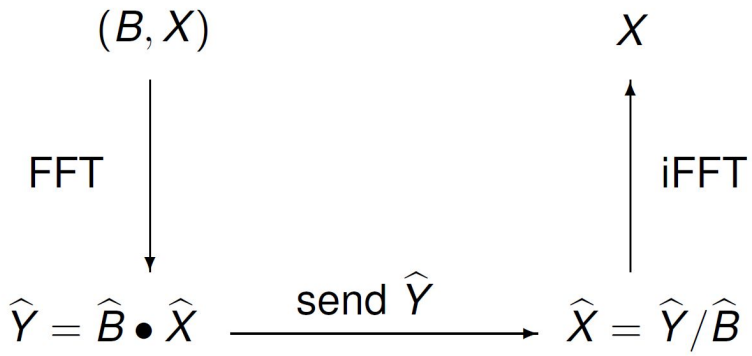


Image Credit: Jan Verschelde, University of Illinois Chicago.

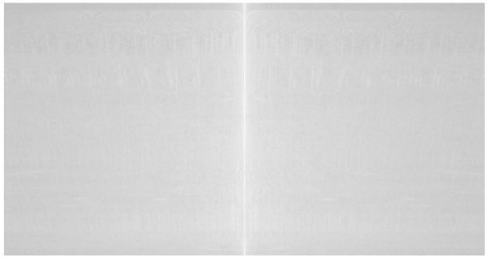
Code Implementation Results

Code Demo 3:

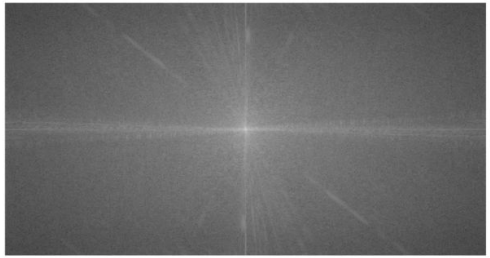
Blur and Deblur Image using 2D Fourier Transforms in FFT and QFT-MPO in Julia

Image Blurring and Deblurring Results

*Original Image (512 H * 1024 W pixels in grayscale)*



Row-wise Discrete Fourier Transform on Royce Hall Image



Final 2D Fourier Transform on Royce Hall Image

- Both Blurring and Deblurring were conducted using classical FFT and QFT-MPO methods
- However, since the QFT-MPO is not optimized for 2D fourier transforms, runtimes for the QFT-MPO were much slower (1 min vs 0.75 seconds)

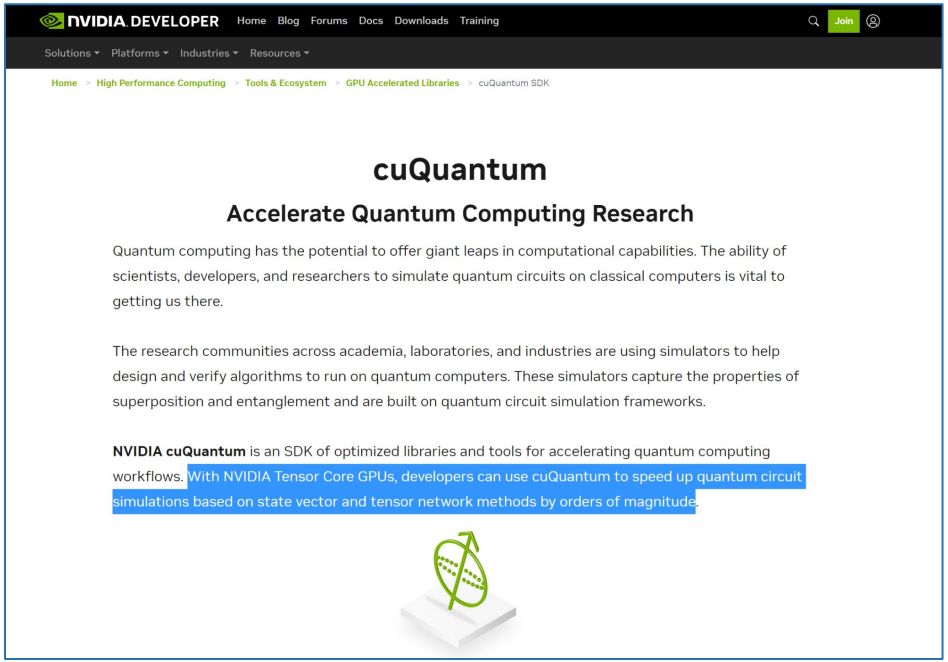
Summary and Future Work

- Prepared working models for the QFT-MPO Algorithm
  - Bitwise encoding of functions as Matrix Product States (MPS)
  - Matrix Product Operator version of the Quantum Fourier Transform (QFT)
- Compared runtime results for calculating the DFT using both classical FFT and quantum-inspired QFT-MPO approach
- Outlined a working model for extending the QFT-MPO to 2D Discrete Fourier Transforms using FFT and QFT-MPO approaches
- Areas for future work using Tensor Network methods:
  - Simulation: Finding Ground State Energies for a given Hamilton using DMRG methods
  - Parallelization: Explore parallel hardware for optimizing tensor contractions
  - Machine learning: Represent weights in a typical neural network as an MPO

Appendix

Motivation - Why study Tensor Network Methods?

- For e.g., Nvidia launched addition to core CUDA library in 2022 called CuQuantum



Motivation - Why study Tensor Network Methods?

- Tensor Network Methods are a core offering within Nvidia's library for implementing quantum circuit simulations

Tensor Network Method

Tensor network methods are rapidly gaining popularity as a way to simulate hundreds or thousands of qubits for near-term quantum algorithms. Tensor networks scale with the number of quantum gates rather than the number of qubits. This makes it possible to simulate very large qubit counts with smaller gate counts on large supercomputers.

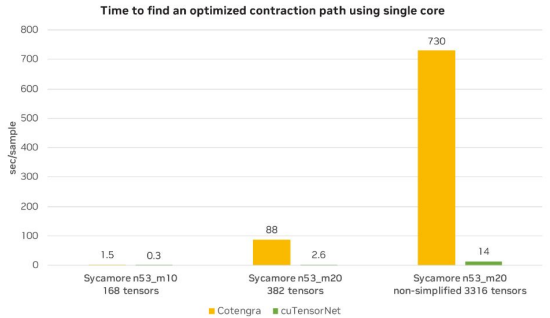
Tensor contractions dramatically reduce the memory requirement for running a circuit on a tensor network simulator. The research community is investing heavily in improving pathfinding methods for quickly finding near-optimal tensor contractions before running a simulation.

cuTensorNet provides state-of-the-art performance for both the pathfinding and contraction stages of tensor network simulation. See the NVIDIA Technical Blog for more details.

Using cuQuantum, NVIDIA researchers were able to simulate a variational quantum algorithm for solving the MaxCut optimization problem using 1,688 qubits to encode 3,375 vertices on an NVIDIA DGX SuperPOD™ system, a 16X improvement over the previous largest simulation — and multiple orders of magnitude larger than the largest problem run on quantum hardware to date.

Pathfinding and Contraction Performance

State-of-the-Art Performance for Pathfinding



Performance for cuTensorNet pathfinding compared to Cotengra in terms of seconds per sample. Both runs are leveraging a single core EPYC 7742 CPU.

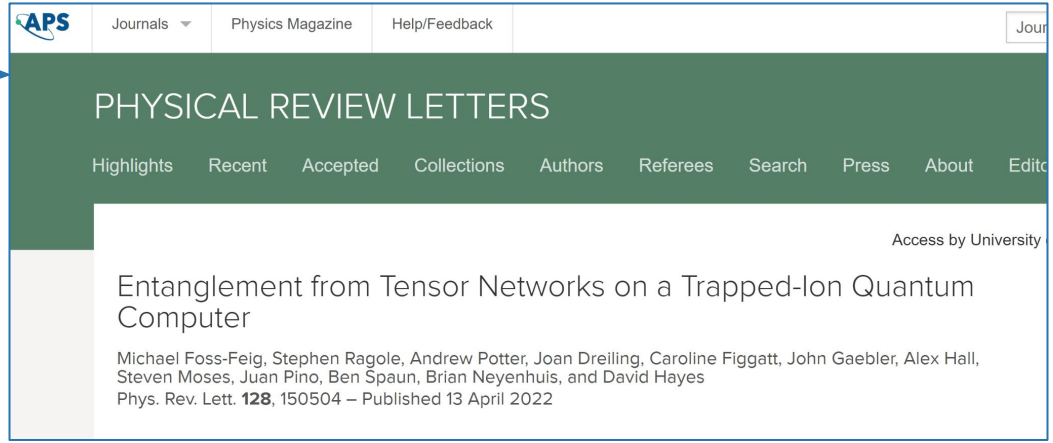
Sycamore refers to 53 qubit random quantum circuits of depth 10, 12, 14, and 20 from Arute et. al. Quantum Supremacy using a Programmable Superconducting Processor.
www.nature.com/articles/s41586-019-1666-5

Cotengra: Gray & Kourtis, Hyper-optimized Tensor Network Contraction, 2021.
quantum-journal.org/papers/q-2021-03-15-410

Motivation - Why study Tensor Network Methods?

- Many Quantum Computing startups already
  - Use them internally for research and improving workloads on real QCs

QUANTINUUM



Why the authors claim the QFT has low entanglement

An Operator Schmidt Decomposition of a unitary matrix acting on a bipartite systems $\mathcal{A} \otimes \mathcal{B}$ is a decomposition of the form:

$$U = \sqrt{N} \sum_{k=0}^{\chi-1} \sigma_k A_k \otimes B_k$$

where:

- χ is the Schmidt rank, $\{\sigma_k\}$ are the Schmidt coefficients
- $\mathrm{Tr}(A_k^\dagger A_{k'}) = \delta_{k,k'}, \mathrm{Tr}(B_k^\dagger B_{k'}) = \delta_{k,k'}$
- $N = dim(\mathcal{A} \otimes \mathcal{B}) = dim(\mathcal{A}) \times dim(\mathcal{B})$ (only apply to unitary)

Image credits: Chris Chen, Caltech

Why the authors claim the QFT has low entanglement

Consider the QFT operator Q_n. By partitioning the Hilbert space into $\mathcal{A}$ with qubits 1 to j, and $\mathcal{B}$ with qubits $j+1$ to n, Q_n has the operator Schmidt decomposition:

$$Q_n = \sqrt{2^n} \sum_{k=0}^{\min(2^j, 2^{n-j})-1} \sigma_{n,j}^k A_{n,j}^k \otimes B_{n,j}^k$$

It then follows that for $k \geq 2$, the Schmidt coefficients $\{\sigma_{n,j}^k\}$ satisfy:

$$\sigma_{n,j}^k \leq \frac{1}{\sqrt{k}} \exp\left(-\frac{2k+1}{2} \log\left(\frac{4k+4}{e\pi}\right)\right)$$

Coefficient bounds are independent of n or j!

Proof Summary

- The Schmidt coefficients of the QFT are singular values of the DFT's submatrix.
- The DFT's submatrix is related to the spectral concentration problem.
- Spectral concentration has decaying solutions (singular values), corresponding to the QFT's decaying Schmidt coefficients.
- The upper-bounds are independent of the number of qubits and the partition of the system.

Schmidt Decomposition

Schmidt Decomposition Theorem 1.5.1. Any vector $\Psi \in \mathcal{H}_1 \otimes \mathcal{H}_2$ can be expressed in the form

$$\Psi = \sum_j c_j |\xi_j\rangle |\eta_j\rangle,$$

for non-negative real c_j, and orthonormal sets $\xi_j \in \mathcal{H}_1$ and $\eta_j \in \mathcal{H}_2$ $(j = 1, 2, \ldots)$. There are density operators ρ_1 on $\mathcal{H}_1$ and ρ_2 on $\mathcal{H}_2$ such that

$$\langle \Psi | (A \otimes 1) \Psi \rangle = \mathrm{tr}[A\rho_1], \qquad \langle \Psi | (1 \otimes B) \Psi \rangle = \mathrm{tr}[B\rho_2],$$

for all observables A and B on $\mathcal{H}_1$ and $\mathcal{H}_2$, respectively, and the $\{\xi_j\}$ may be chosen to be the eigenvectors of ρ_1 corresponding to non-zero eigenvalues $\{p_j\}$, the vectors $\{\eta_j\}$, the corresponding eigenvectors for ρ_2, and the positive scalars $c_j = \sqrt{p_j}$.

Results - Benchmarking and Error

```
[10] run_benchmark(16, lambda x: np.cos(x), verbose=True, max_bond_mps=10, max_bond_mpo=8, cutoff_mps=1e-12, cutoff_mpo=1e-12)
✓ 0.8s
```

```
====== Results for benchmark (N=16, max_bond=10, cutoff=1e-12) ======
MPS Construct Time:  0.01427965199764003
QFT Construct Time:  0.7439916980001726
QFT Apply Time:  0.004576500999974087
MPS + Apply Time:  0.018856152997614117

QFT vs FFT Error:  8.932863093613526e-05
```

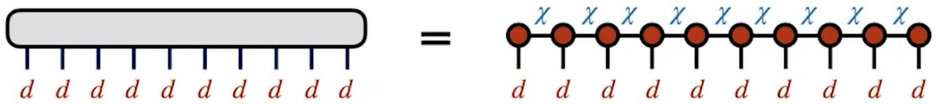
```
[13] run_benchmark(16, lambda x: np.cos(x), verbose=True, max_bond_mps=6, max_bond_mpo=6, cutoff_mps=1e-12, cutoff_mpo=1e-12)
✓ 0.5s
```

```
====== Results for benchmark (N=16, max_bond=6, cutoff=1e-12) ======
MPS Construct Time:  0.009470198998315027
QFT Construct Time:  0.49391406100039603
QFT Apply Time:  0.004518500998528907
MPS + Apply Time:  0.013988699996843934

QFT vs FFT Error:  0.03690362135397189
```

Bond Dimension of an MPS

- Any tensor can be represented in an MPS given a large enough bond dimension
- If bond dimension is small (or grows logarithmically), significant compression of original circuit is possible



If modest χ yields good approximation,
obtain massive compression:

$$d^N \longrightarrow N d \chi^2$$

Image Credits: iTensor