計算科学・量子計算における情報圧縮

Data Compression in Computational Science and Quantum Computing

2022.12.8

#8: テンソルネットワークによる情報圧縮

Data compression in tensor networks

理学系研究科　大久保　毅

Graduate school of science, **Tsuyoshi Okubo**

I put (URLs of) recordings of previous lectures on ITC-LMS.
You can also download lecture slide from ITC-LMS.

Today's topic

1. Computational science, quantum computing, and data compression
2. Review of linear algebra
3. Singular value decomposition
4. Application of SVD and generalization to tensors
5. Entanglement of information and matrix product states
6. Application of MPS to eigenvalue problems
7. Tensor network representation
8. Data compression in tensor network
9. Tensor network renormalization
10. Quantum mechanics and quantum computation
11. Simulation of quantum computers
12. Quantum-classical hybrid algorithms and tensor network
13. Quantum error correction and tensor network

Outline

- Simple data compression by tensor network
- Application of TN to data science
  - Tensor ring decomposition
  - Classification problem with TN
  - Generative models with TN
  - Compressing (deep) neural network
- Application to differential equations
- Comment: tensor network and quantum computing
- Report 1

Simple data compression by tensor network

Tensor network decomposition of a vector

Target:

Exponentially large Hilbert space

$\vec{v} \in \mathbb{C}^M$

with $M \sim a^N$

+

Total Hilbert space is decomposed as a product of "local" Hilbert space.

$$\mathbb{C}^M = \mathbb{C}^a \otimes \mathbb{C}^a \otimes \cdots \mathbb{C}^a$$

*Local Hilbert space dimensions can be different.

Examples:

Picture image:

256 × *256* pixel image → 2^{16} dimensional vector

→ *16*-leg tensor (with *a* = *2*)

Probability distribution:

e.g. Ising model $P(\{S_i\}) = \dfrac{e^{\beta J \sum_{\langle i,j \rangle} S_i S_j}}{Z}$

→ 2^N vector → *N*-leg tensor (with *a* = *2*)

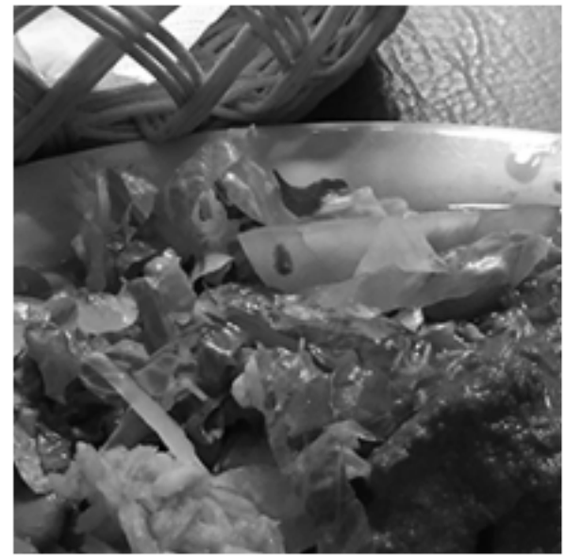
Wave function:

$$|\Psi\rangle = \sum_{\{m_i=0,1\}} T_{m_1,m_2,\cdots,m_N} |m_1, m_2, \cdots, m_N\rangle \quad \rightarrow \quad T_{m_1,m_2,\cdots,m_N} \text{ :}N\text{-leg tensor}$$

Example: picture

256 × *256* pixel image

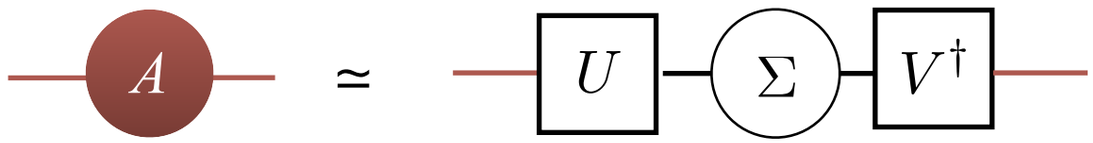
$256=2^8$



$256=2^8$

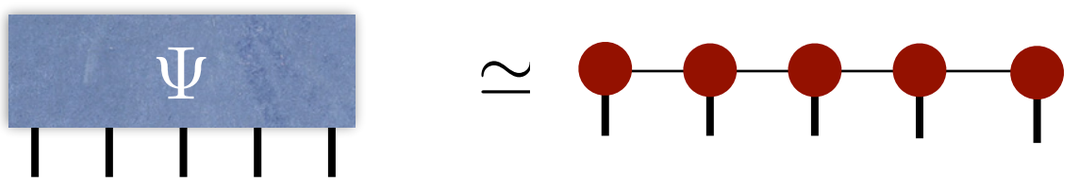
2 types of data structure

1. *256* × *256* matrix (as we see)
   - We can apply simple SVD.



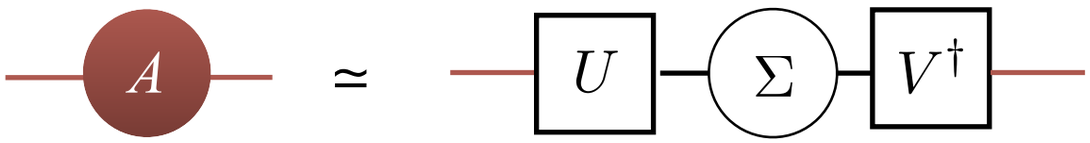
2. 2 × 2 × ...× 2 16-leg tensor
   - We can apply MPS approximation
   - Degrees of freedom to order data (indices)

MPS



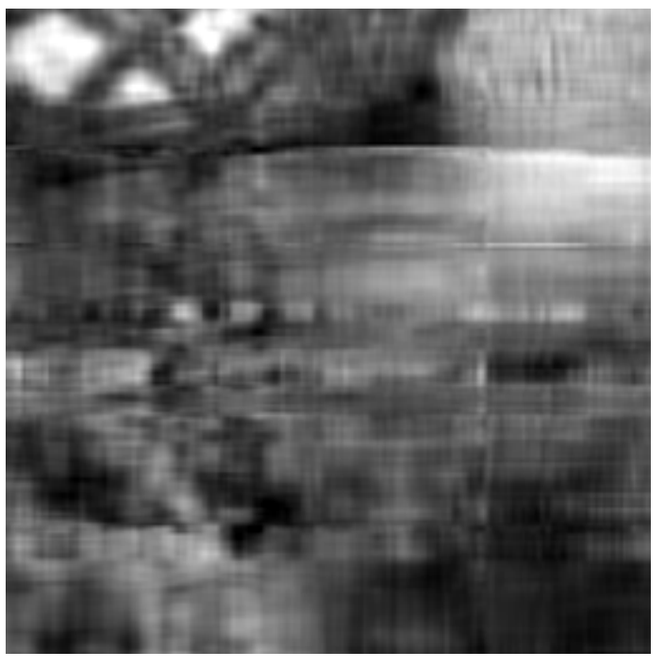
Compression by SVD

1. 256×256 matrix



Distance: $\Delta \equiv \|A - \tilde{A}\| / \|A\|$

Amount of data: $D \equiv 256 \times \chi + \chi \times 256 + \chi = 513\chi$



$\chi = 10$

$\Delta = 0.2025$

$D = 5130$

$\chi = 20$

$\Delta = 0.1529$

$D = 10260$

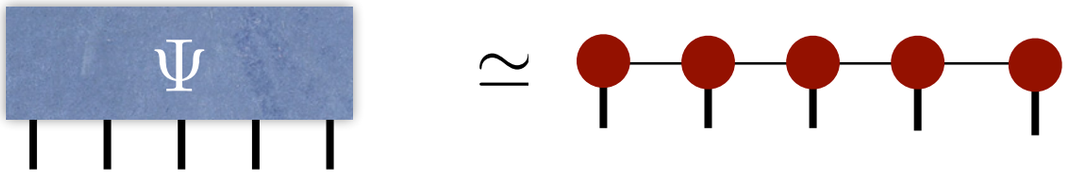
$\chi = 50$

$\Delta = 0.0885$

$D = 25650$

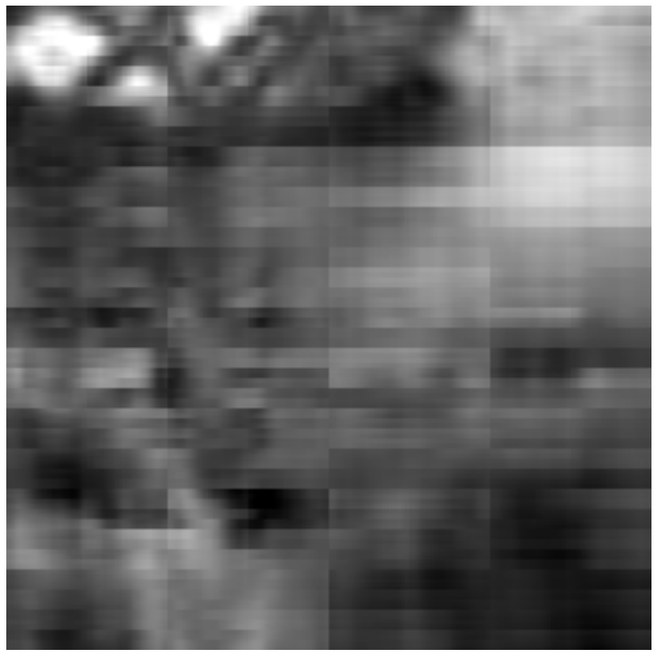
Compression by MPS: case1

2. $2 \times 2 \times \ldots \times 2$, 16-leg tensor



Distance: $\Delta \equiv \|A - \tilde{A}\| / \|A\|$

Amount of data: $D \simeq N\chi_{max}^2$



$\chi_{max} = 10$

$\Delta = 0.2405$

$D = 2088$

$\chi_{max} = 20$

$\Delta = 0.1873$

$D = 6760$

$256 = 2^8$

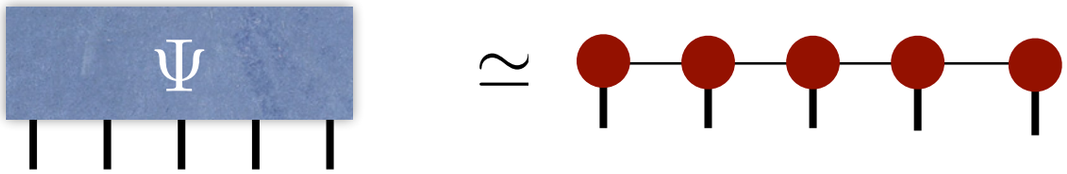
$256 = 2^8$

0	1	2	…	254	255
256	257	258	…		
				…	2^{16}-1

From the binary representation of numbers, we make 16-leg tensor.

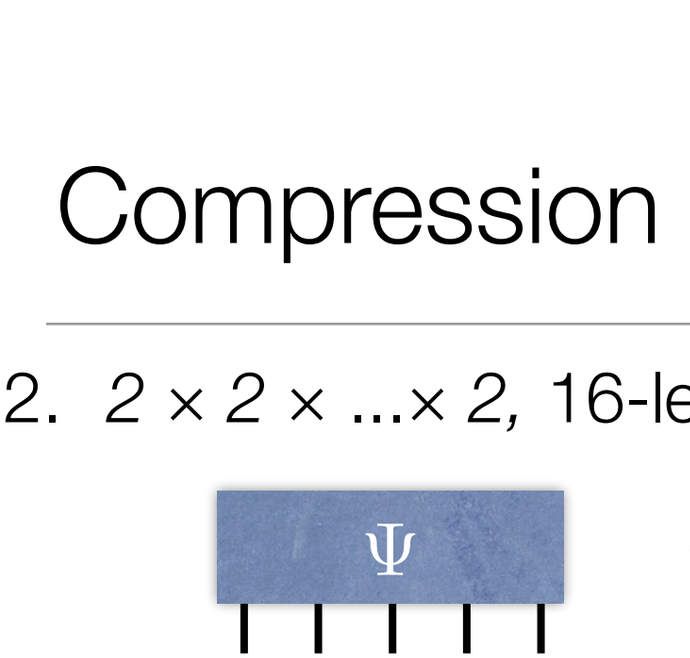
Compression by MPS: randomized

2. $2 \times 2 \times \ldots \times 2$, 16-leg tensor



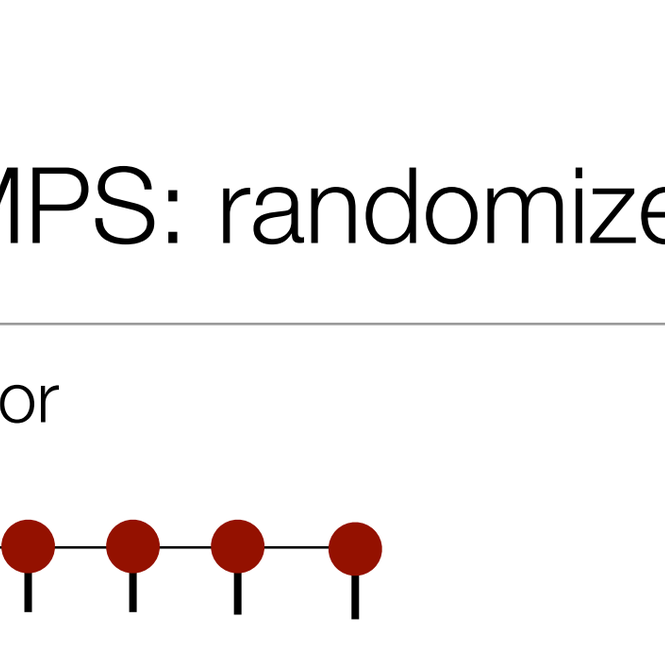
$\chi_{max} = 50$

$D = 29128$



case 1

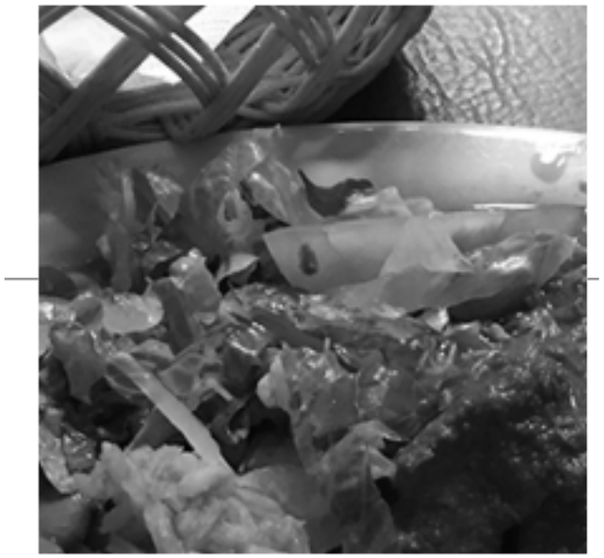
$\Delta = 0.1122$



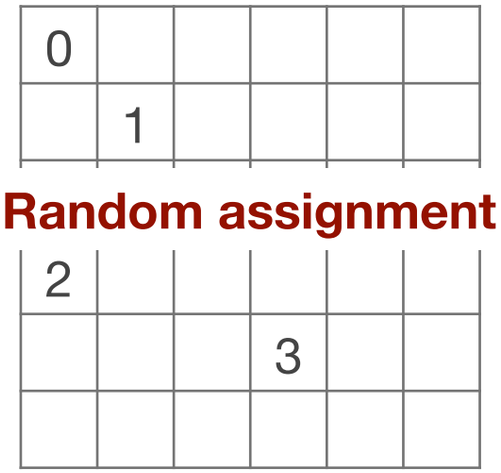
Random

$\Delta = 0.4227$

$256=2^8$



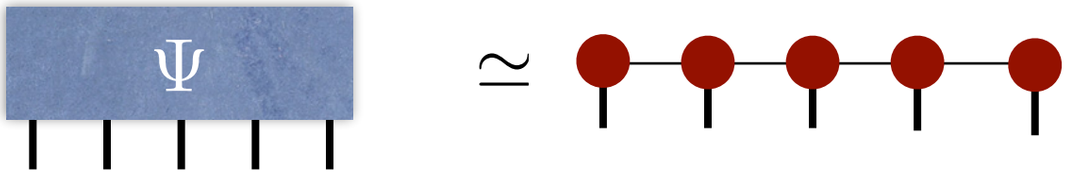
$256=2^8$



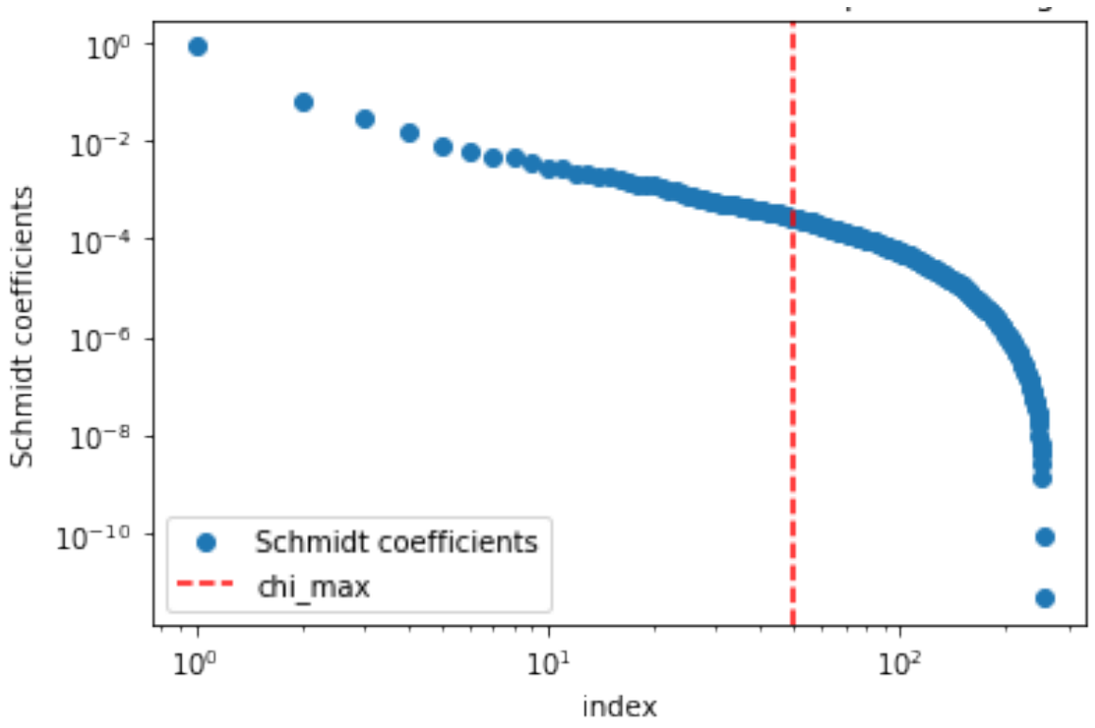
From the binary representation of numbers, we make 16-leg tensor.

Singular value spectrum

2. $2 \times 2 \times \ldots \times 2$, 16-leg tensor

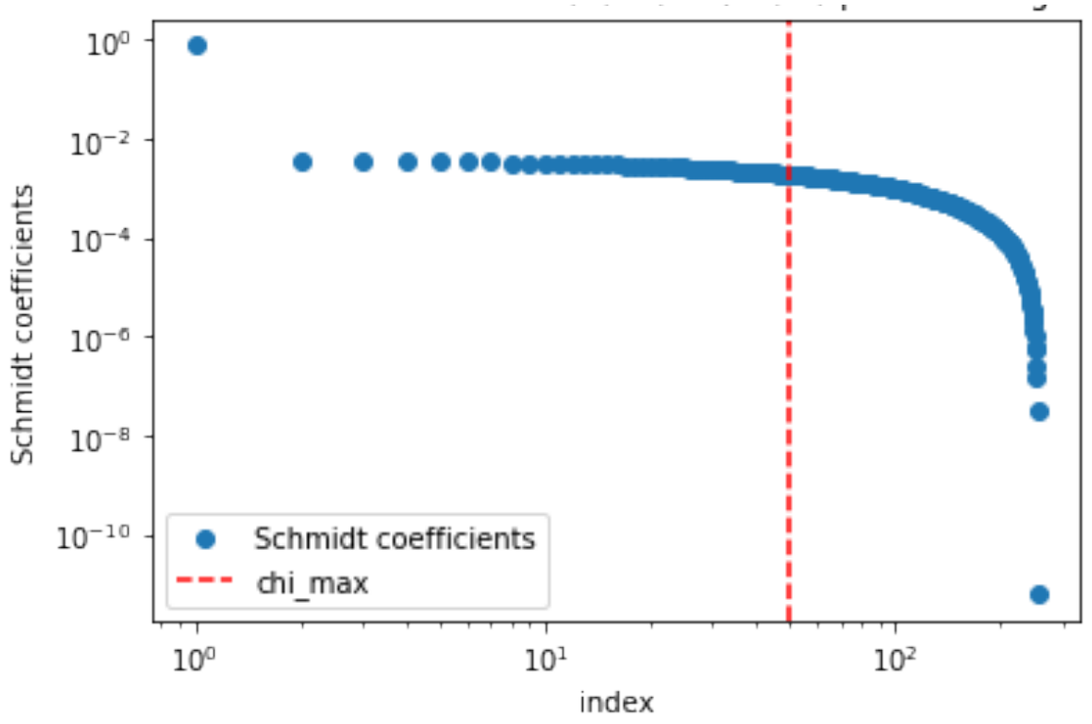


Singular values (Schmidt coefficient) at bipartition



case 1

$\Delta = 0.1122$



Random

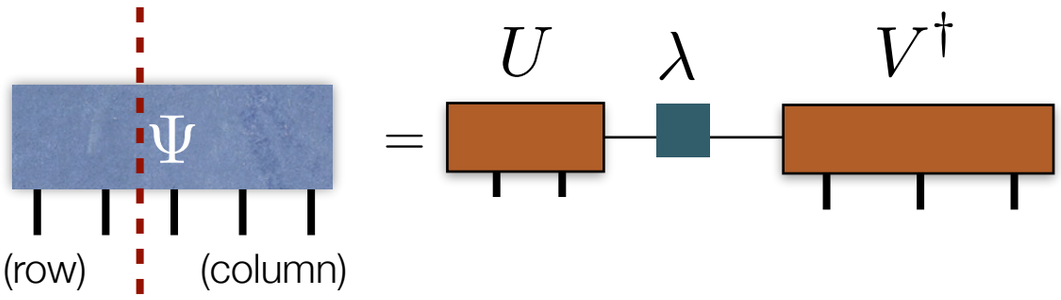
$\Delta = 0.4227$

Singular value spectrums are very different!

Entanglement entropy

Distribution of the singular values for dividing tensor two parts.

It characterizes the entanglement of information between two parts.



Entanglement entropy = Entropy for normalized singular values

$$\tilde{\lambda}_i \equiv \lambda_i / \sqrt{\sum_i \lambda_i^2}$$

$$S = -\sum_i \tilde{\lambda}_i^2 \log \tilde{\lambda}_i^2$$

$$0 \leq S \leq \log \chi$$

χ: # of singular values

Entanglement entropy of the picture

	case 1	random
Error at $\chi_{max} = 50$	$\Delta = 0.1122$	$\Delta = 0.4231$
Entanglement entropy	$S = 0.9494$	$S = 1.801$

$\log 256 \simeq 5.545$

Application to data science

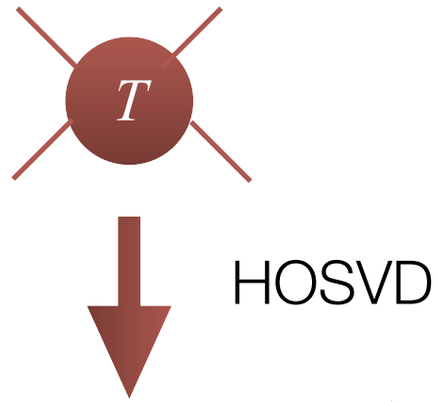
Tensor ring decomposition

Decomposition of tensor data

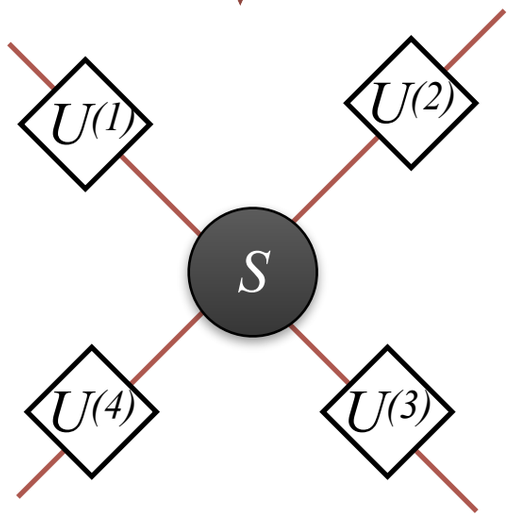
Eg. COIL-100 dataset = 128 x 128 x 3 x 100 x 72 tensor

Pixel color object direction

Objects:



Several directions:

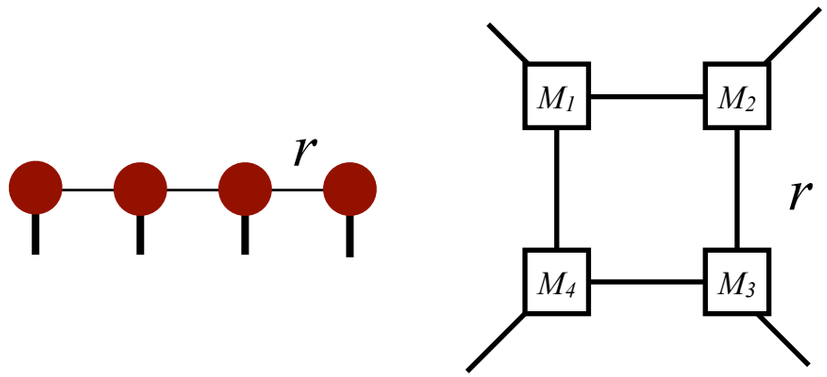


Previously, we considered law-rank approximation by HOSVD in #4.

→ Here we consider a tensor network decomposition.

Application of MPS to tensor data

Tensor train (TT) and tensor ring (TR) decompositions for real data.



*CP

$$\sum_{\alpha=1}^{r} V_\alpha^{(1)} V_\alpha^{(2)} V_\alpha^{(4)} V_\alpha^{(3)}$$

COIL-100 dataset = 32 x 32 x 3 x 7200 tensor

(Downsampled) Pixel color # of pictures

Compression by tensor ring decomposition.

(Q. Zhao, et al arXiv:1606.05535)

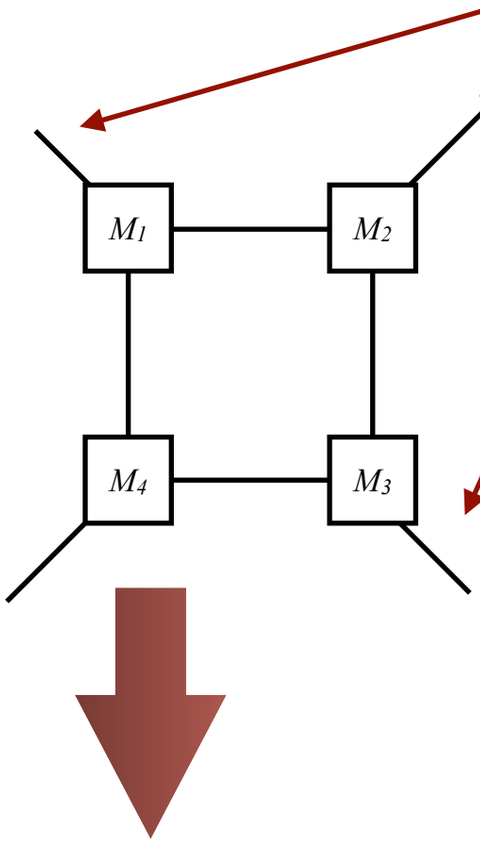
error Rank

	ϵ	r_{max}	$\bar{r}$	Acc. (%) ($\rho = 50\%$)	Acc. (%) ($\rho = 10\%$)
CP-ALS	0.20	70	70	97.46	80.03
	0.30	17	17	97.56	83.38
	0.39	5	5	90.40	77.70
	0.47	2	2	45.05	39.10
TT-SVD	0.19	67	47.3	99.05	89.11
	0.28	23	16.3	98.99	88.45
	0.37	8	6.3	96.29	86.02
	0.46	3	2.7	47.78	44.00
TR-SVD	0.19	23	12.0	99.14	89.29
	0.28	10	6.0	99.19	89.89
	0.36	5	3.5	98.51	88.10
	0.43	3	2.3	83.43	73.20

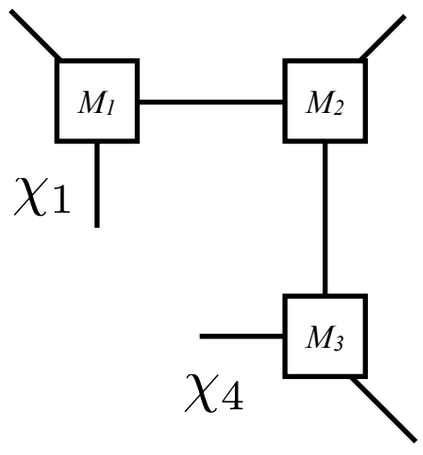
Extract features

COIL-100 dataset = 32 x 32 x 3 x 7200 tensor

Pixel color # of pictures



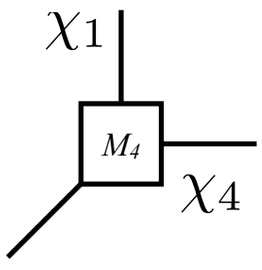
Map: M



M maps $32 \times 32 \times 3$ dimensional image data to $\chi_1 \times \chi_4$ dimensional feature space.

Applying inverse map M^{-1} to an image data, we obtain the corresponding feature vector.

Feature matrix: F



F relates $\chi_1 \times \chi_4$ dimensional features to picture indices

From M and F, we may solve

- Classification problem for unknown image
- Clustering problems for given tensor data

Classification problem with TN

Machine learning for classification problem

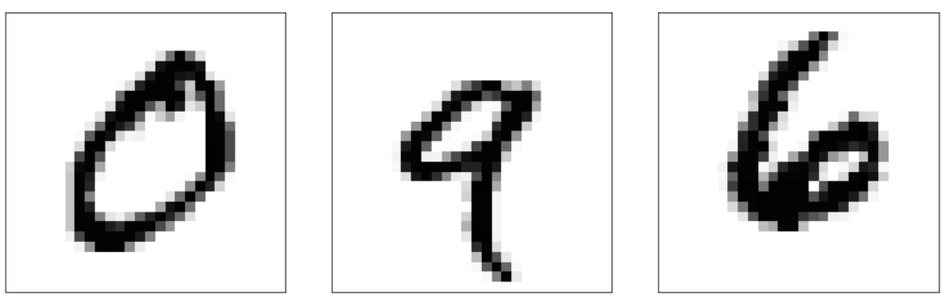
Problem: we want to classify an input vector by several labels

E.g. Classification of handwriting images

Input: grayscale image (28 × 28 pixel)

output: 10 dimensional vector corresponding

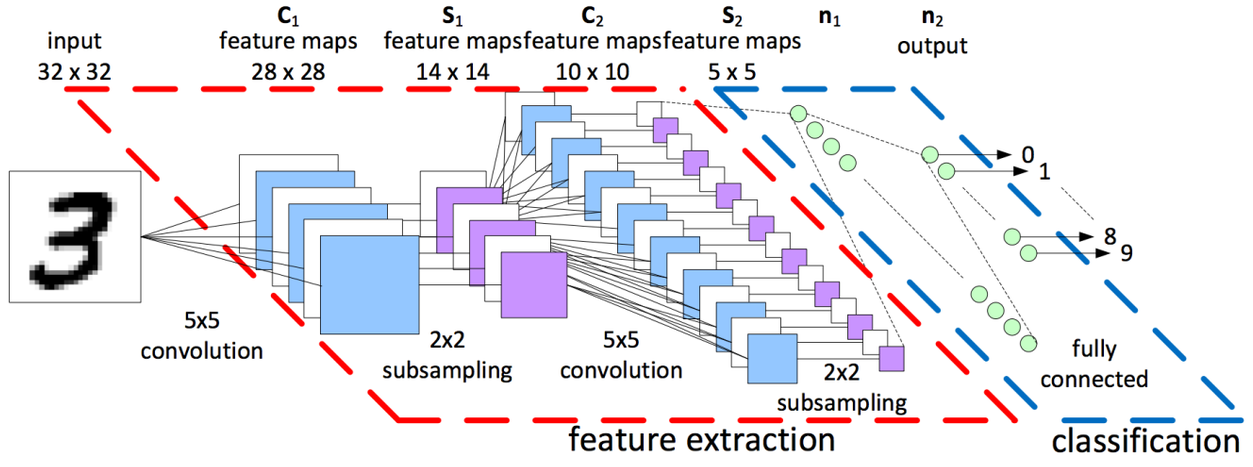
$$0, 1, 2, \ldots, 9$$



(From MNIST dataset)

How to design the classification machine?

- (Deep) neural network
- Tensor network



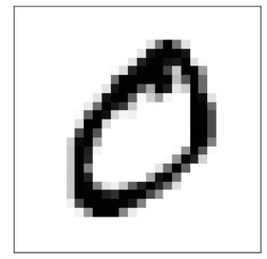
Taken from https://www.kaggle.com/code/cdeotte/how-to-choose-cnn-architecture-mnist

A model for supervised learning

E. Miles Stoudenmire and D. J. Schwab, NIPS 2016. (arXiv:1605.05775)

1. The input vector $\boldsymbol{x}$ is mapped onto higher dimensional space

$$\vec{\psi}(\boldsymbol{x})$$ (non-linear feature map)

$\boldsymbol{x} =$ 

2. It is transformed into labels through a linear map.

$$f^l(\boldsymbol{x}) = W^l \vec{\psi}(\boldsymbol{x})$$

W : a linear map onto labels space

e.g. MNIST → $l = 0, 1, \ldots 9$

3. We optimize W based on the correct labels of the training data.

Simple cost function: $C = \frac{1}{2} \sum_n \sum_l \left(f^l(\boldsymbol{x}_n) - y_n^l \right)^2$

y_n^l : correct label

MPS representation of the classification problem

E. Miles Stoudenmire and D. J. Schwab, NIPS 2016. (arXiv:1605.05775)

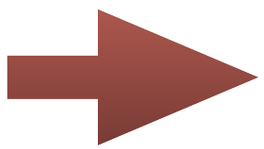
If we select a "product state" as a feature map

$$\psi_{i_1,i_2,\ldots,i_N}(\boldsymbol{x}) = \phi_{i_1}(x_1) \otimes \phi_{i_2}(x_2) \otimes \cdots \otimes \phi_{i_N}(x_N)$$

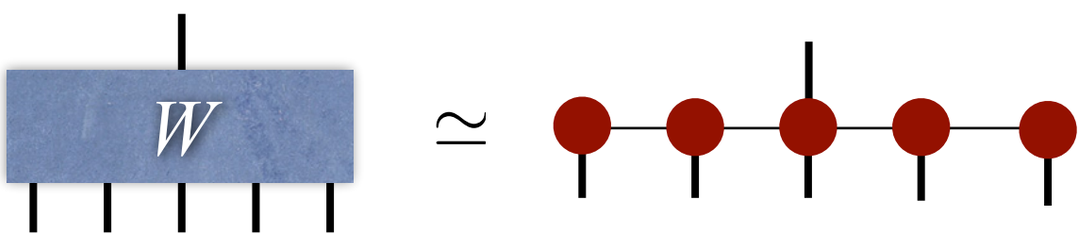
$\vec{\psi} =$

$\vec{\phi}(x_1) \quad \cdots \quad \vec{\phi}(x_N)$

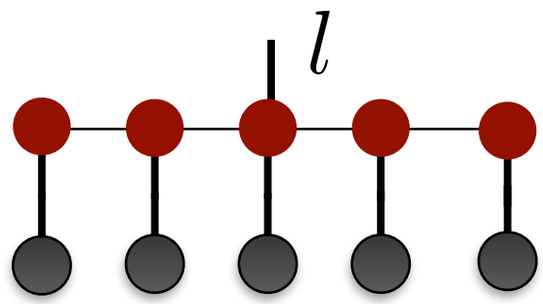
The dimension of vector space is a^N



Then we can apply MPS approximation for W



$$f^l(\boldsymbol{x}) = W^l \vec{\psi}(\boldsymbol{x}) =$$



MPS representation of the classification problem

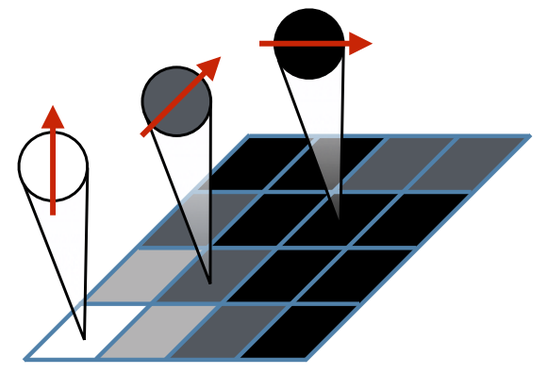
E. Miles Stoudenmire and D. J. Schwab, NIPS 2016. (arXiv:1605.05775)

Feature map

$$\psi_{i_1,i_2,\ldots,i_N}(\boldsymbol{x}) = \phi_{i_1}(x_1) \otimes \phi_{i_2}(x_2) \otimes \cdots \otimes \phi_{i_N}(x_N)$$

Their feature map:

$$\phi^{s_j}(x_j) = \left[\cos\left(\frac{\pi}{2}x_j\right),\ \sin\left(\frac{\pi}{2}x_j\right)\right]$$



For the case of grayscale image

Application to MNIST database of handwritten digits

(handwritten numbers from 0 to 9)

χ	Test set error	
10	~5%	500/10000
20	~2%	200/10000
120	~0.97%	97/10000

<1%

It is comparable with the state of the art!

Generative model with TN

Unsupervised Generative Modeling

Z.-Y. Han et al, Phys. Rev. X **8**, 031012 (2018).

N pixel grey scale image

Binary data: $\vec{v} \in \mathbb{V} = \{0,1\}^{\otimes N}$

Probability distribution of images: $P(\vec{v})$

$\vec{v} \sim (0,1,0,1,\ldots,0,0)$ = Sequence of 0 and 1

Born machine:

Z.-Y. Han et al, Phys. Rev. X **8**, 031012 (2018).

It has a similar structure to qubits.

$\vec{v} \to |0101\ldots00\rangle = |\vec{v}\rangle$

The probability distribution is related to a quantum state.

$$P(\vec{v}) = \frac{|\Psi(\vec{v})|^2}{Z} \qquad (Z = \sum_{\vec{v}} |\Psi(\vec{v}_i)|^2)$$

$$|\Psi\rangle = \sum_i \Psi(\vec{v}_i)|\vec{v}_i\rangle$$

*When we measure $|\Psi\rangle$, we observe $|\vec{v}_i\rangle$ with probability $\frac{|\Psi(\vec{v}_i)|^2}{Z}$.

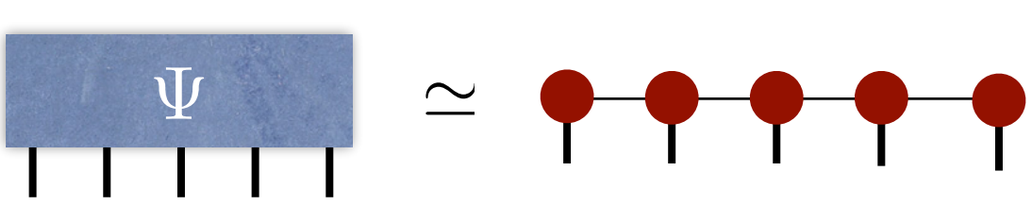
Unsupervised Generative Modeling

Z.-Y. Han et al, Phys. Rev. X **8**, 031012 (2018).

Born machine:

$$P(\vec{v}) = \frac{|\Psi(\vec{v})|^2}{Z} \qquad (Z = \sum_{\vec{v}} |\Psi(\vec{v}_i)|^2)$$

We can use tensor network representation for a quantum state!

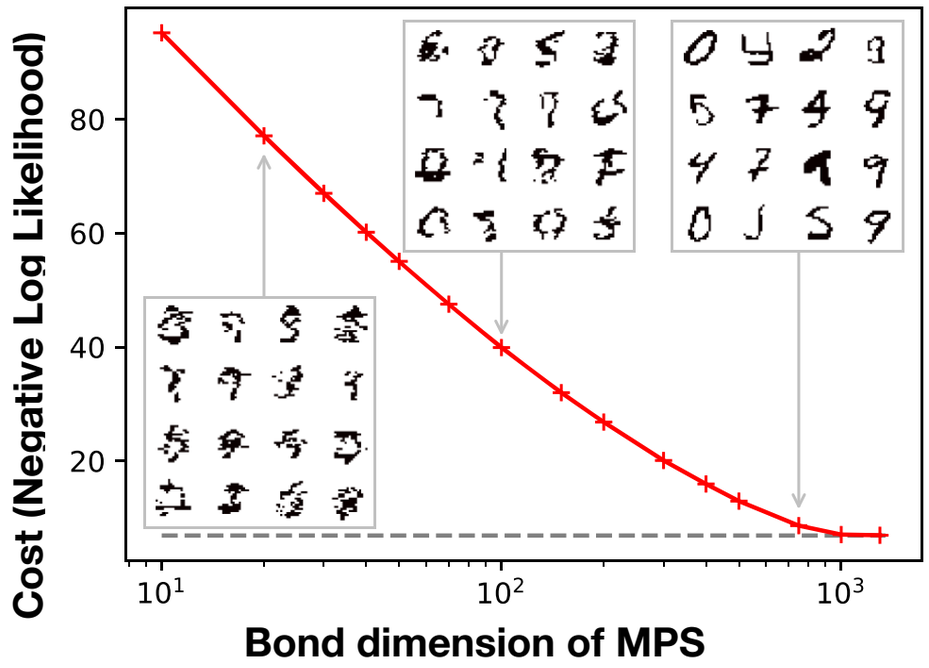


*Notice that *a priori*, we don't know whether it is efficient.

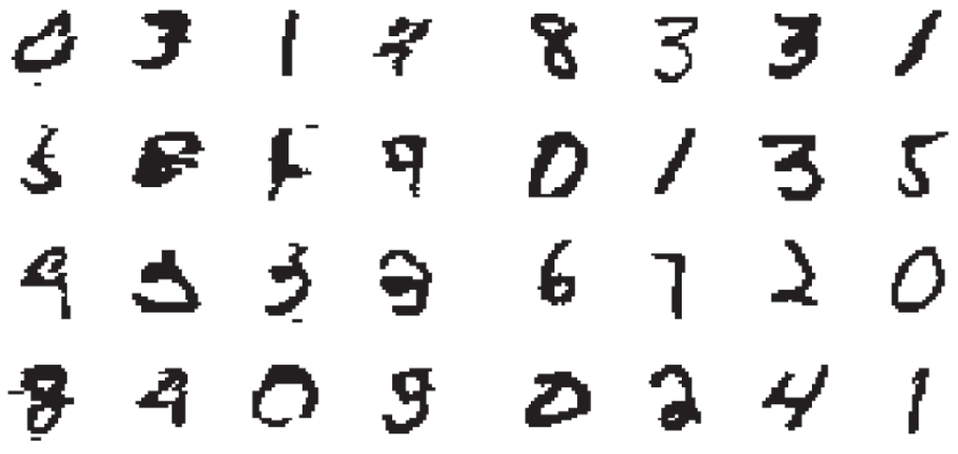
For example, we find optimal MPS to minimize

$$F = -\frac{1}{\mathbb{T}} \sum_{\vec{v} \in \mathbb{T}} \ln P(\vec{v})$$

$\mathbb{T}$: Set of training data



Images generation



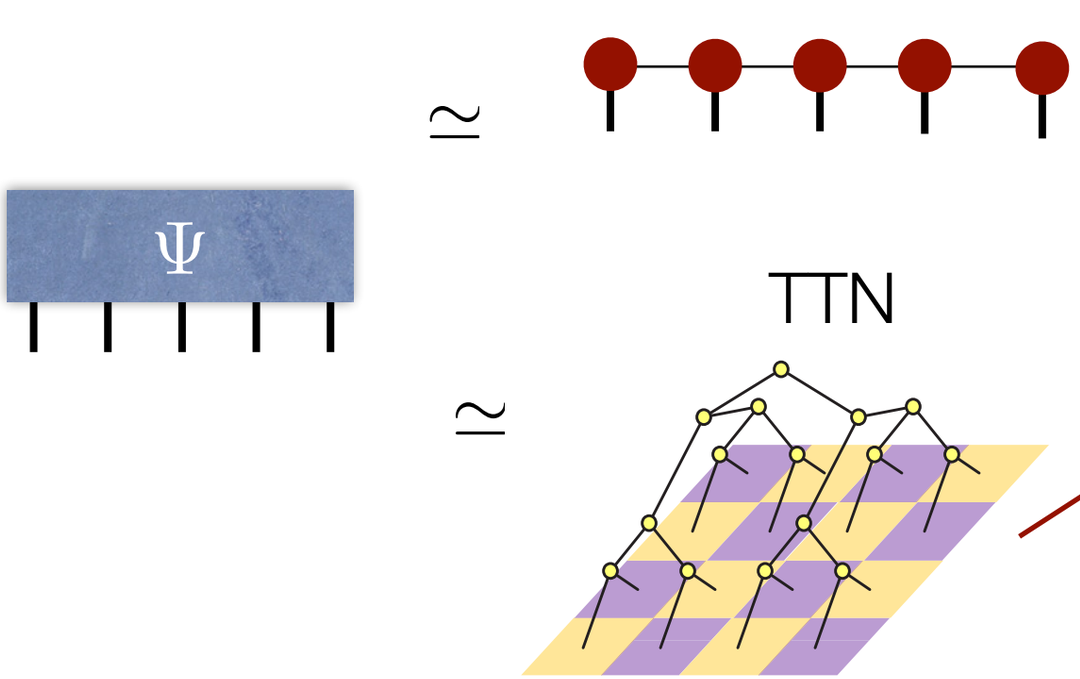
(a) Generated (b) Original

Unsupervised Generative Modeling

Born machine:

$$P(\vec{v}) = \frac{|\Psi(\vec{v})|^2}{Z}$$

As in the case of quantum many-body states,
the appropriate tensor network structure
depends on the nature of the data set.



(From S. Cheng et al, Phys. Rev. B **99**, 155131 (2019).)

TABLE I. Test NLL of different models for the binary MNIST data set.

Model	Test NLL
Tree factor graph	175.8
MPS	101.5
TTN, 1D	96.9
TTN, 2D	94.3
RBM	86.3[a] [43]
VAE	84.8[a] [45]
PixelCNN	81.3 [10]

[a] Approximated NLL.

*There is another issue that originated from neglecting the phases in the Born machine.

Many different quantum states give the same probability distribution!

Some of them are more entangled, and others are less entangled…

Compressing (deep) neural networks

Compressing deep neural network

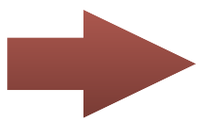
Z.-F. Gao et al, Phys. Rev. Research **2**, 023300 (2020).

MPO approximation of the weight matrix

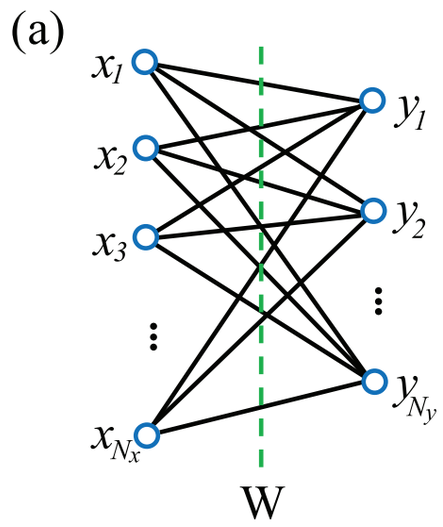
x_i: input neuron (pixel)

y_i: output neuron

W_{ij}: weight matrix connecting x and y



MPO approximation of W



Example: application to classification problems

TABLE I. Test accuracy a and compression ratios ρ obtained in the original and MPO representations of LeNet-5 on MNIST and VGG on CIFAR-10.

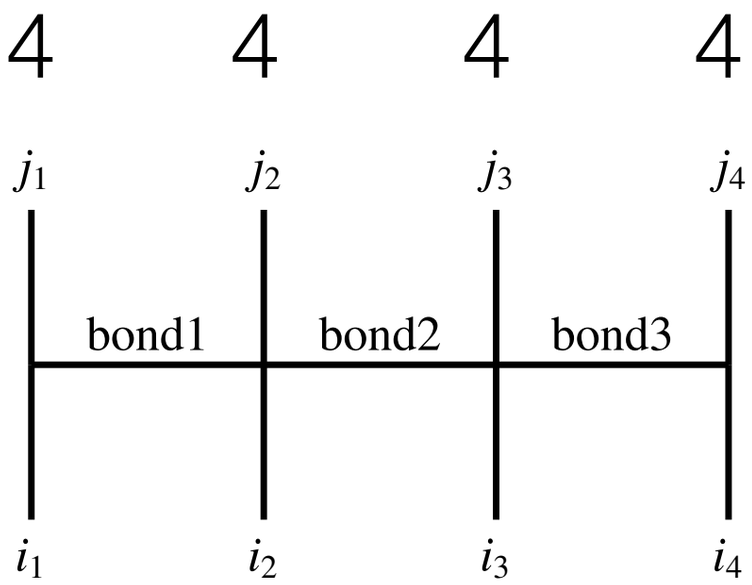
Data set	Network	Original Rep a (%)	MPO-Net a (%)	MPO-Net ρ
MNIST	LeNet-5	99.17 ± 0.04	99.17 ± 0.08	0.05
CIFAR-10	VGG-16	93.13 ± 0.39	93.76 ± 0.16	$\sim$0.0005
	VGG-19	93.36 ± 0.26	93.80 ± 0.09	$\sim$0.0005

a:**accuracy (%)**

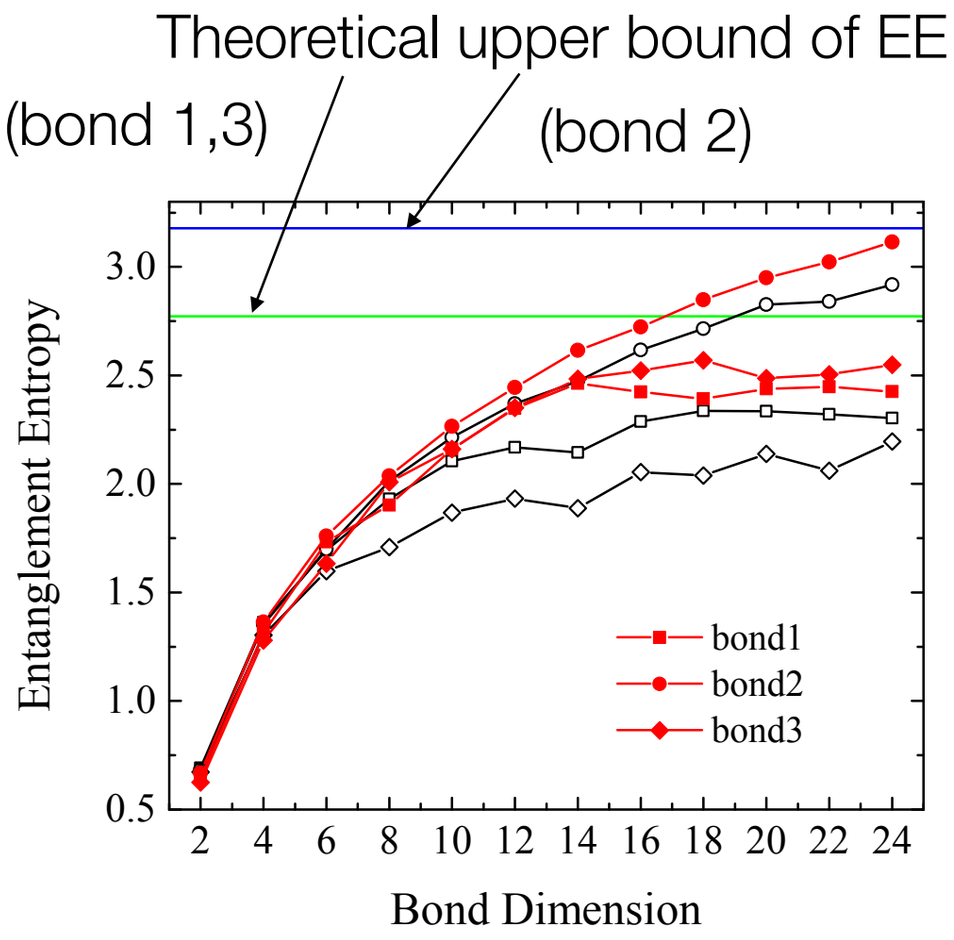
ρ:**compression ratio**

Entanglement entropy of (trained) MPO

Z.-F. Gao et al, Phys. Rev. Research **2**, 023300 (2020).



input: 28 x 28 pixel image

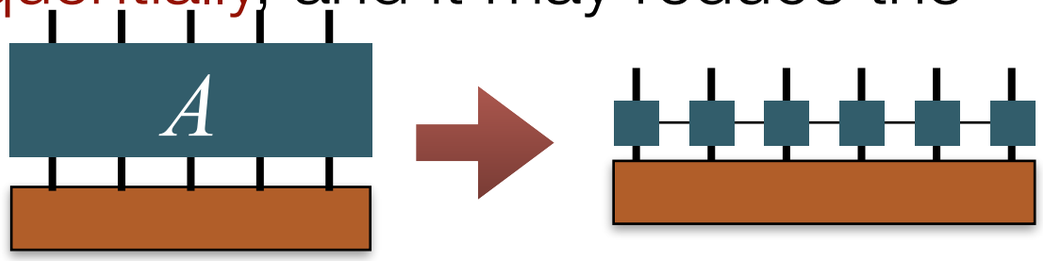


- Fashion-MNIST gives larger EE
  - It might be related to the difficulty of the dataset
- For bond 1 and 3, EE saturated smaller values than the theoretical upper bound
  - It indicate we can use MPO for approximation

Black: MNIST
Red: Fashion-MNIST
(more complicated)

Benefit of data compression in deep learning

- We can reduce the memory cost to store the matrix data.
  - It is the same with data compression for tensors and matrices.
- We can also reduce the calculation cost for multiplying the matrix
  - We can multiply small tensors sequentially, and it may reduce the computation cost largely.



  - It reduces the time for both training and evaluation.
- The increase of efficiency at optimization due to a smaller # of parameters.
  - Almost unchanged performance after data compression indicates that the original dense matrix contains lots of “redundant” information.
  - It may affect overfitting and generalization performance.

Application to differential equation

Application to differential equation

Nikita Gourianov et. al., Nat. Comput. Sci. **2**, 20 (2022).

Incompressible Naiver-Stokes equation

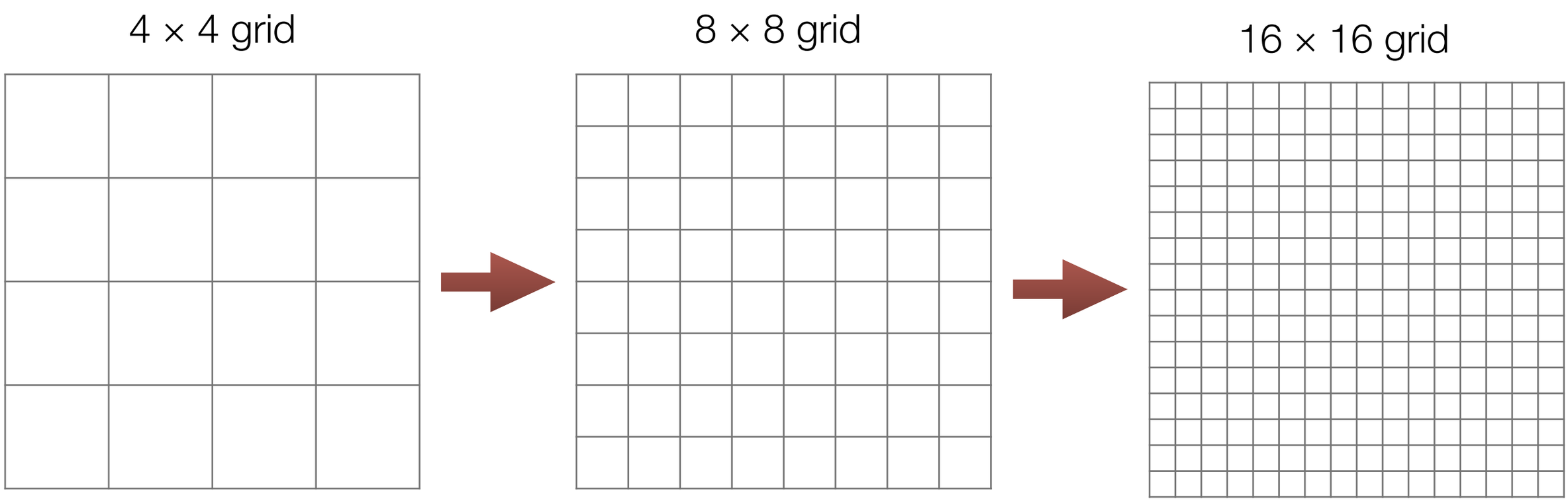
$$\nabla \cdot V = 0$$

$$\frac{\partial V}{\partial t} + (V \cdot \nabla) V = -\nabla p + \nu \nabla^2 V,$$

Direct numerical simulation (DNS) : Solve the equation in a discretized grid.

In the case of turbulence, we need finer mesh due to couplings of multi-scale physics.

$$V(t, r_q) = \sum_{i=1}^{K} u_i(t, r_q) \hat{\mathbf{e}}_i$$



Can we apply data compression by tensor-network?

Application to differential equation

Nikita Gourianov et. al., Nat. Comput. Sci. **2**, 20 (2022).

Can we apply data compression by tensor-network?

We can apply TN decomposition for the grid data
as we did for pictures.
However, its efficiency largely depends on physics.

Let's entanglement in this system!

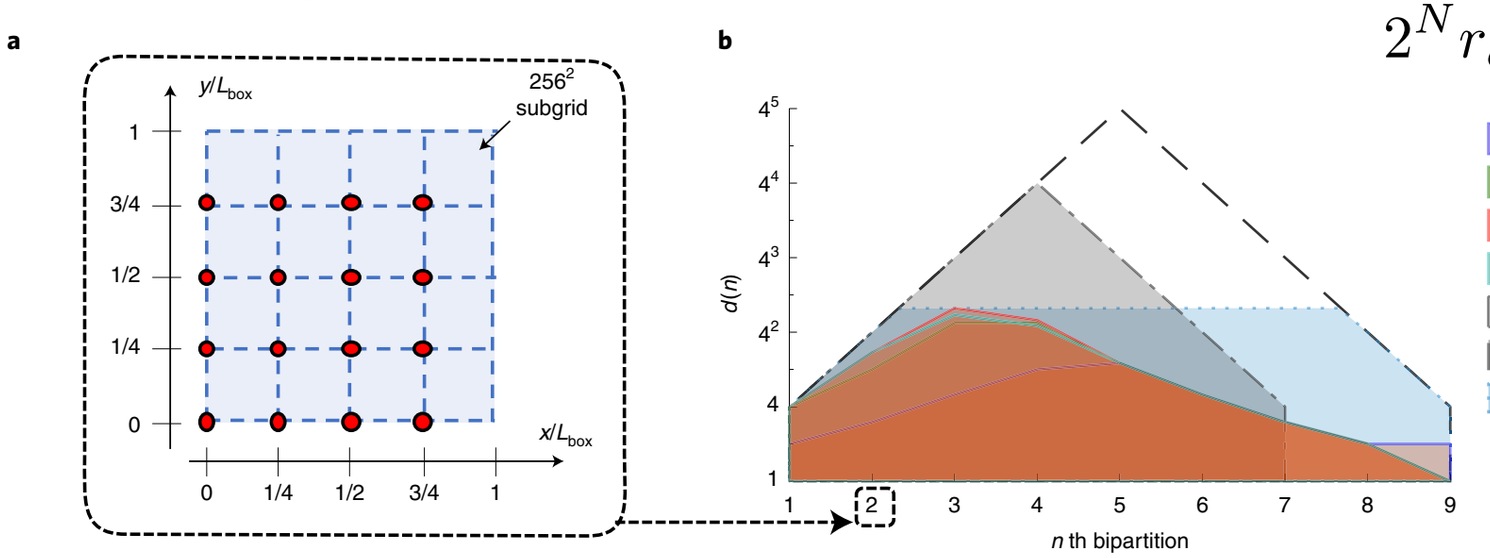
Schmidt decomposition between larger and smaller length scales.

$$u_i(t, r_q) = \sum_{\alpha=1}^{d(n)} \lambda_\alpha(t) R_\alpha(t, X_k) f_\alpha(t, x_l), \quad r_q = X_k + x_l.$$

$$V(t, r_q) = \sum_{i=1}^{K} u_i(t, r_q)\hat{\mathbf{e}}_i$$

*Please imagine a binary representation of grid points

$$2^N r_q = i_0 + 2^1 i_1 + 2^2 i_2 + \cdots + 2^{N-1} i_{N-1}$$



Required Schmidt coefficients for 99% accuracy in the simulation.
→1d area law!

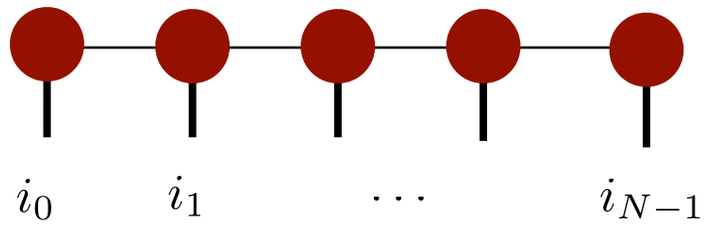
Application to differential equation

Nikita Gourianov et. al., Nat. Comput. Sci. **2**, 20 (2022).

The 1d area law indicates that we may use MPS to simulate this system.

Suppose we treat 2^N grid points in 1d.

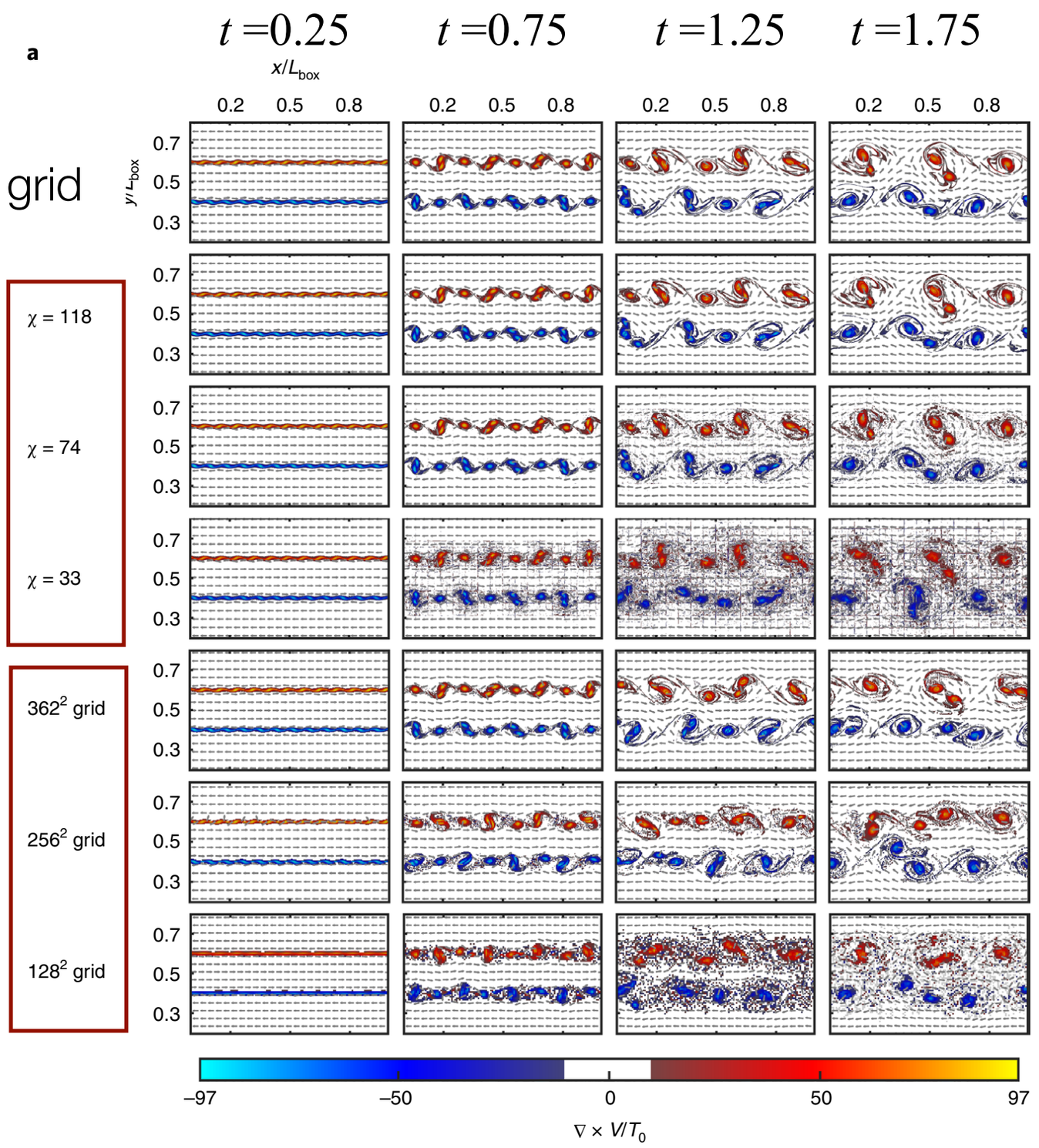
$$u_i(r_q) \simeq$$



$$2^N r_q = i_0 + 2^1 i_1 + 2^2 i_2 + \cdots + 2^{N-1} i_{N-1}$$

(In the paper, they simulate 2d and 3d systems with MPS. In these cases, local Hilbert space dimensions are 2^2 and 2^3.)

The time evolution based on MPS correctly captures true behaviors.



Comment: tensor network and quantum computing

Quantum circuit

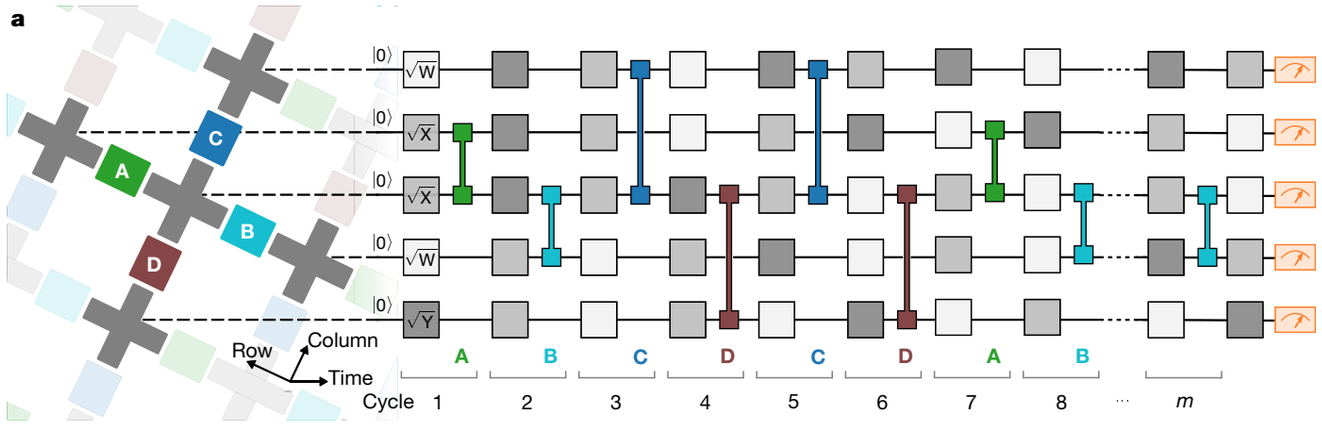
Quantum circuit:

Circuit based on gate operations on quantum bits

Unitary matrices (tensors)

2-dim complex vectors

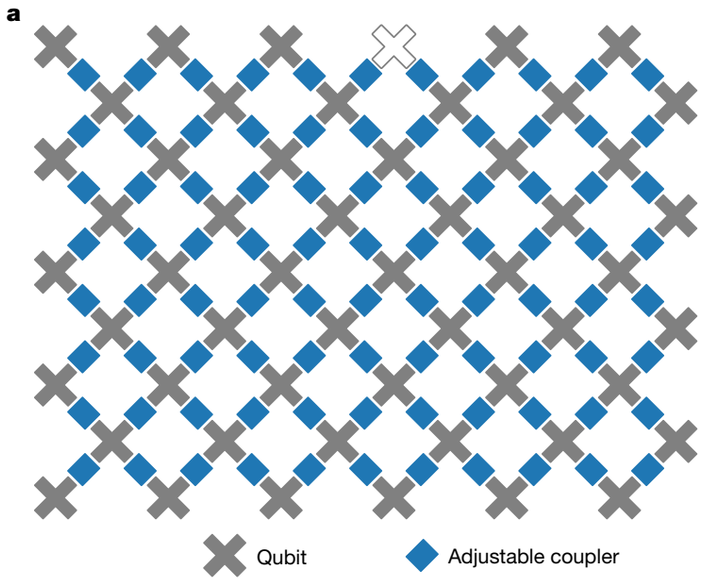
Google's "quantum supremacy" circuit

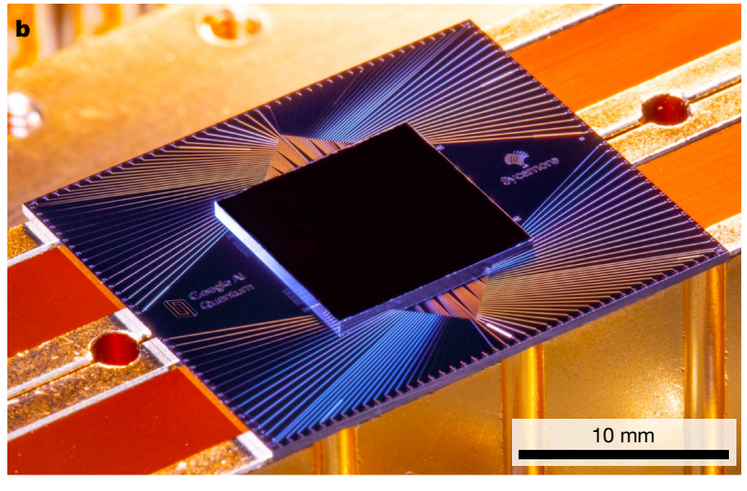


Quantum circuit = tensor network

"quantum supremacy" circuit

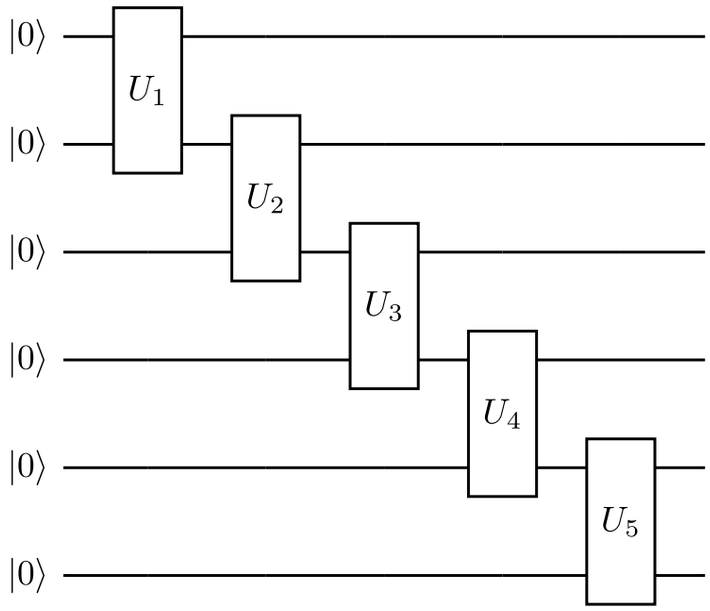
F. Arute, *et al.*, Nature 574, 505 (2019)



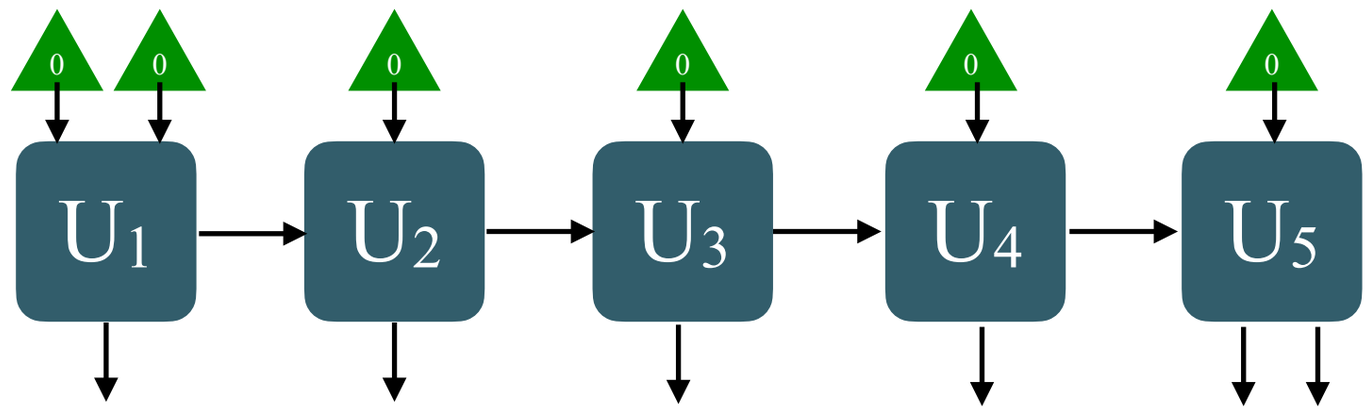


Transform TNs to quantum circuit

We can relate a quantum circuit to a tensor network.



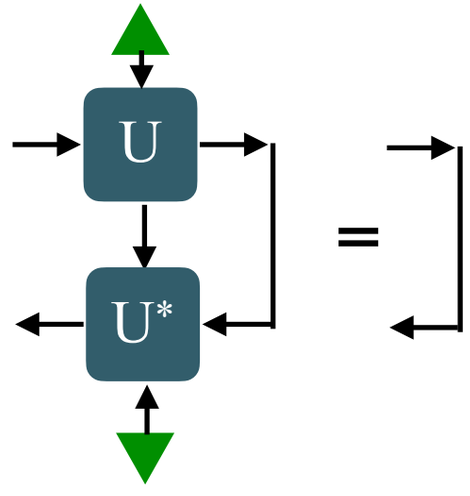
=



This is a MPS in the right canonical form.

*U_i is a unitary matrix.

We may use quantum computers for calculations expressed by TNs, such as today's examples.



Next (Dec. 15)

Notice

- No classes on Nov. 3, Nov. 17, and Nov. 22
- Classes will also be held on Jan. 5 and Jan. 19

---

1. **Computational science, quantum computing, and data compression**
2. **Review of linear algebra**
3. **Singular value decomposition**
4. **Application of SVD and generalization to tensors**
5. **Entanglement of information and matrix product states**
6. **Application of MPS to eigenvalue problems**
7. **Tensor network representation**
8. **Data compression in tensor network**
9. **Tensor network renormalization**
10. **Quantum mechanics and quantum computation**
11. **Simulation of quantum computers**
12. **Quantum-classical hybrid algorithms and tensor network**
13. **Quantum error correction and tensor network**