Tensor Networks for Generative Modeling

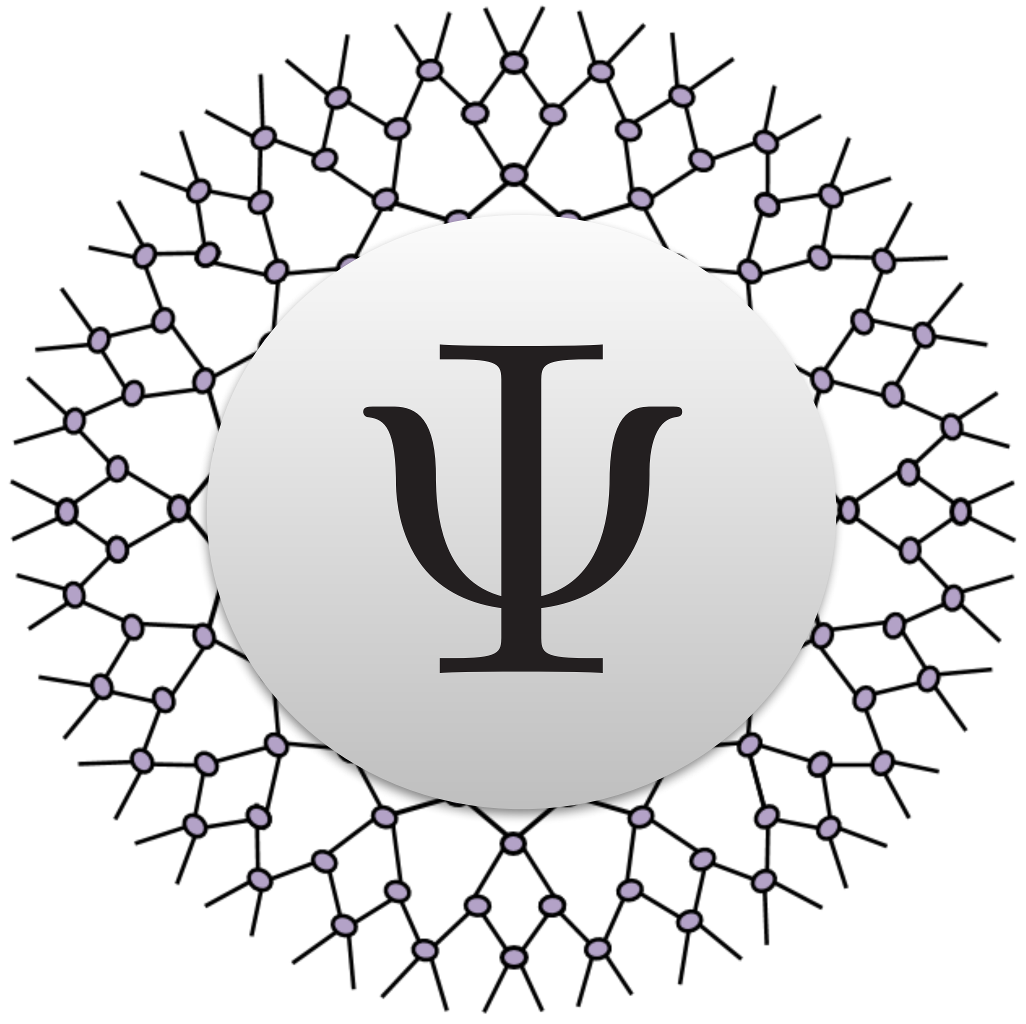
From Boltzmann machines to Born machines, and back

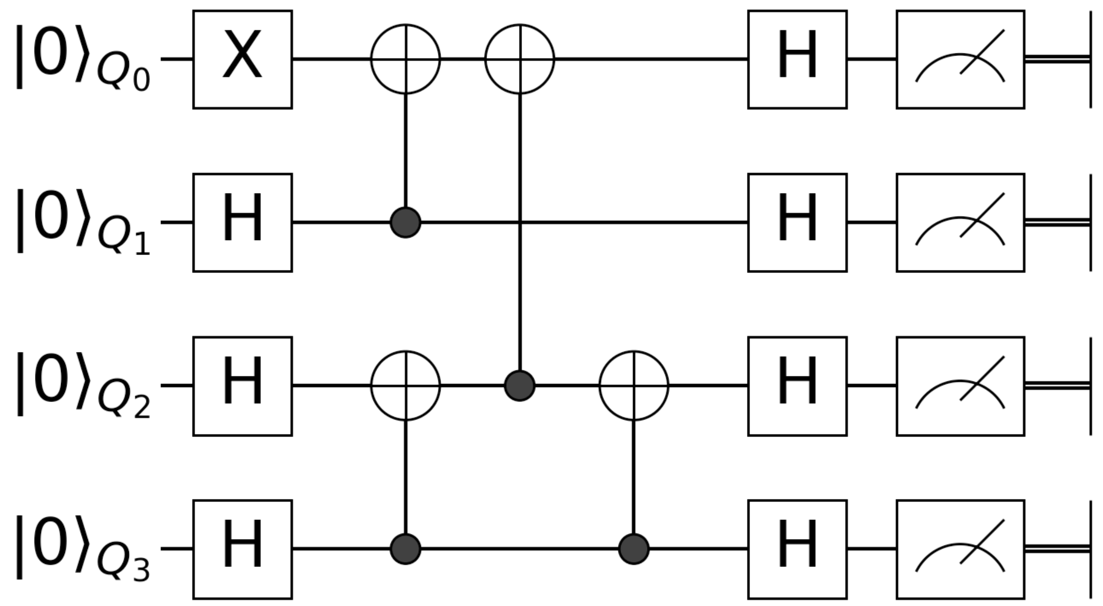
Lei Wang (王磊)
https://wangleiphy.github.io

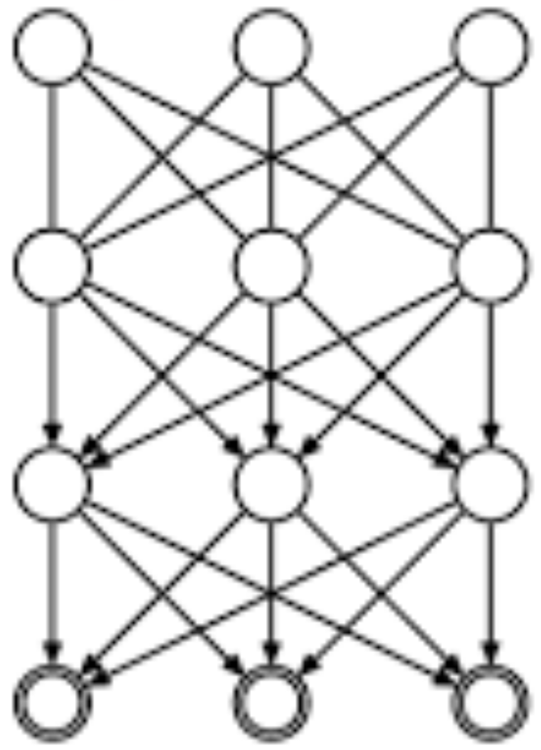
Institute of Physics, Beijing
Chinese Academy of Sciences

Han, Wang, Fan, LW, Zhang, PRX18'
Chen, Cheng, Xie, LW, Xiang, PRB 18'
Cheng, Chen, LW, Entropy 18'+unpublished

TNSAA: Live long and prosper







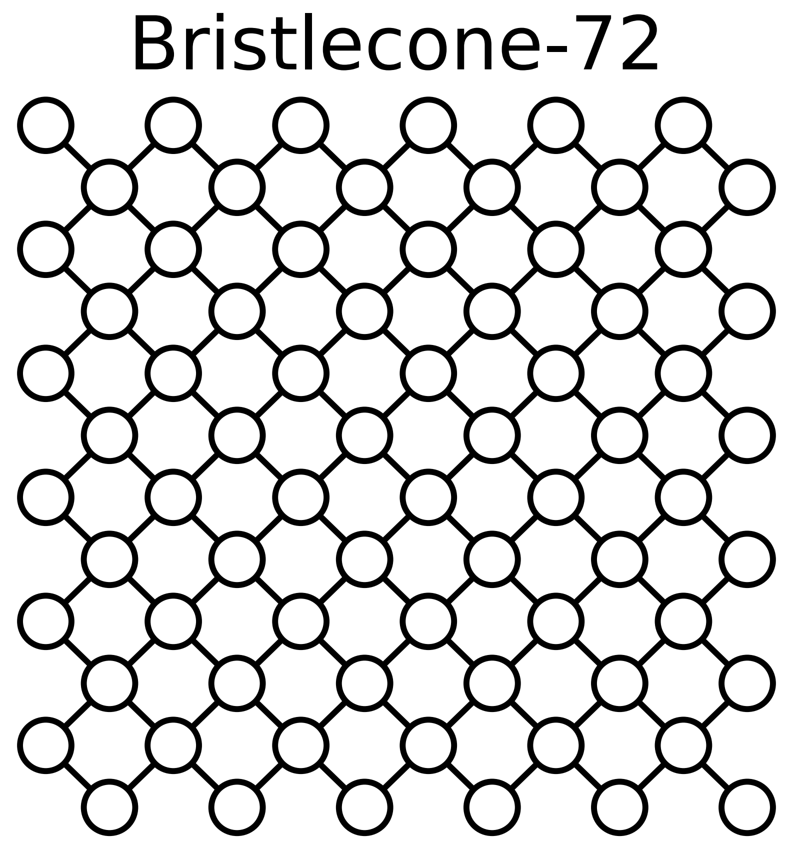
Quantum circuits architecture and parametrization

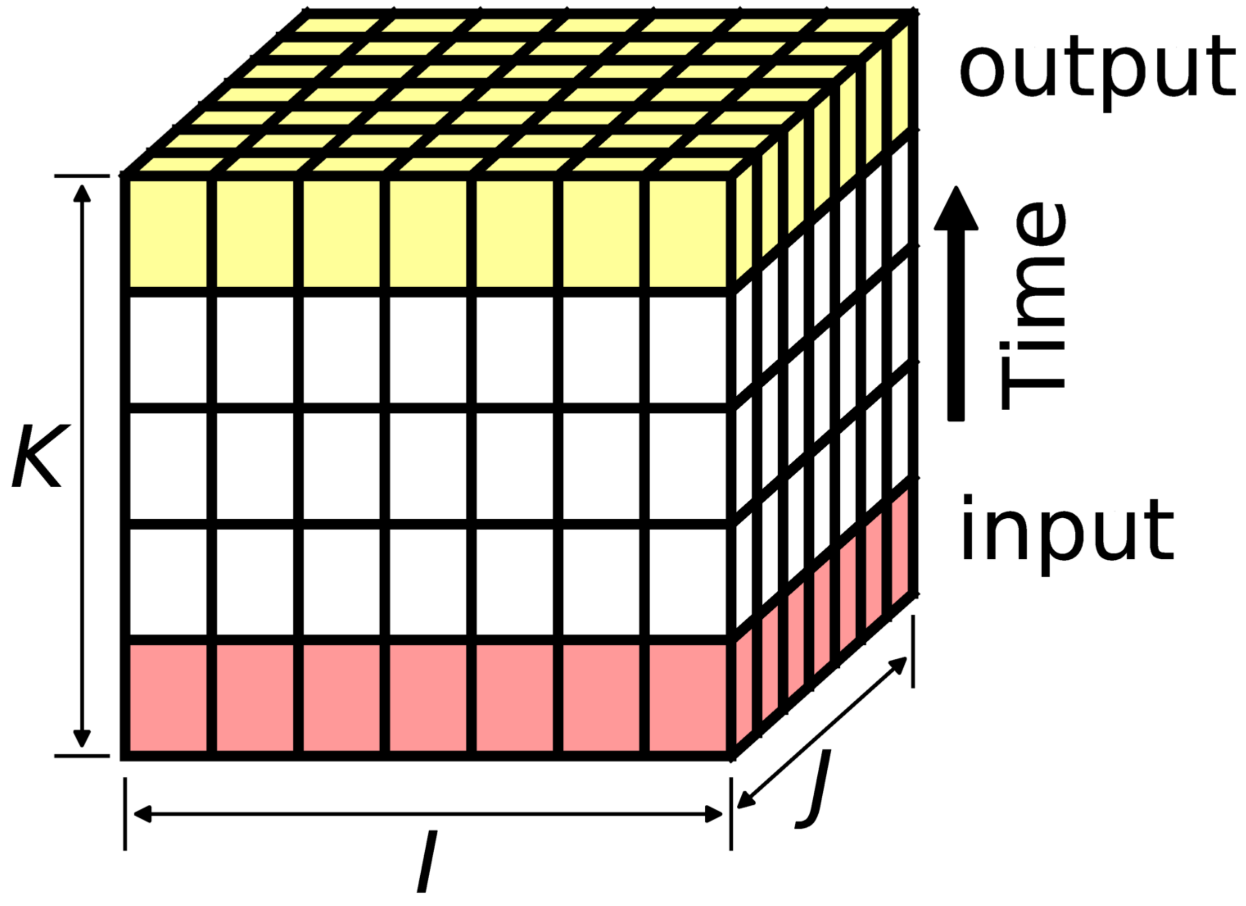
Neural networks and Probabilistic graphical models

TNSAA in the era of quantum computing

Simulation and validation of quantum circuits

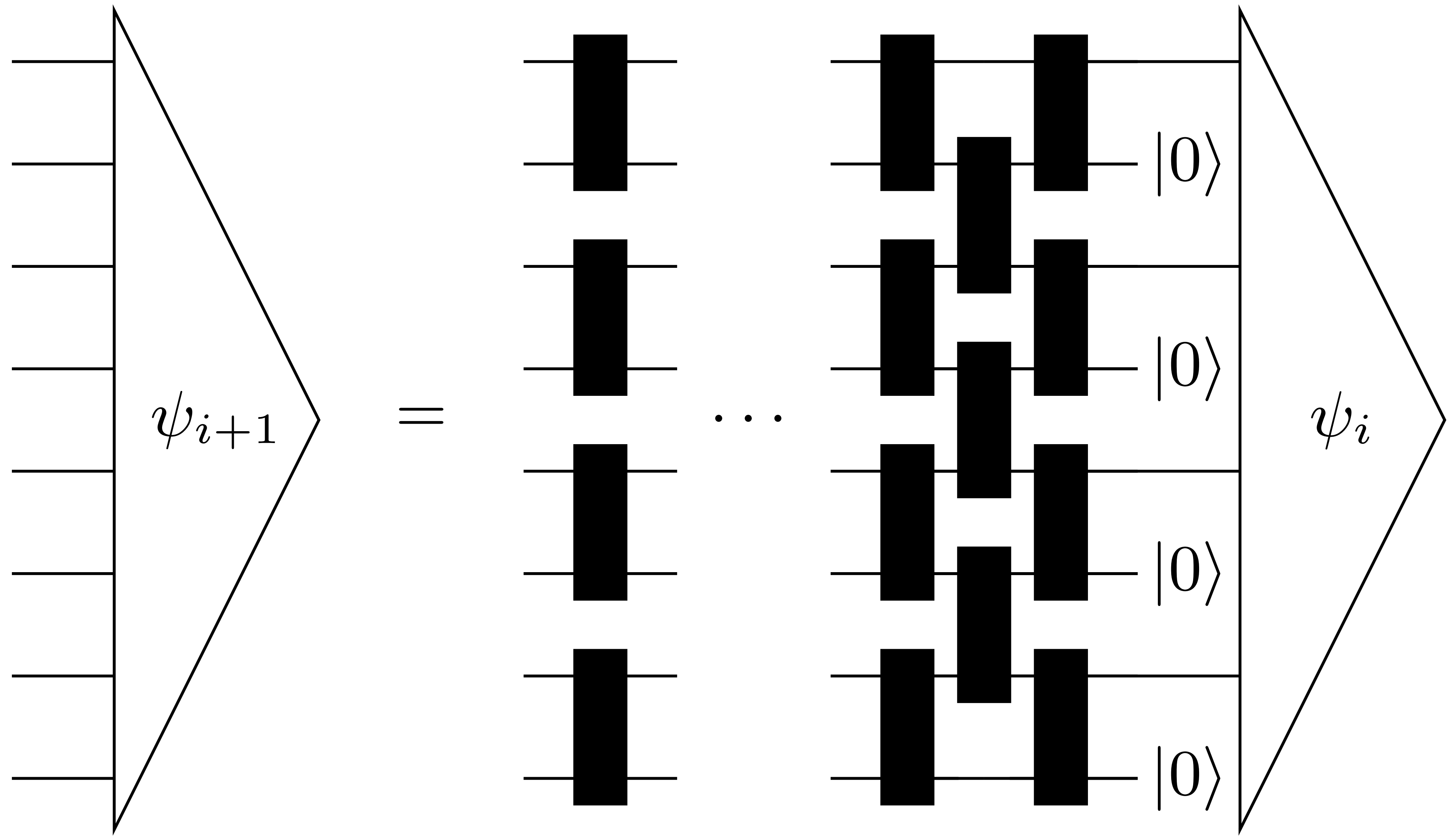
Villalonga et al, 1811.09599





Noise-resilient quantum circuits with TNS architecture

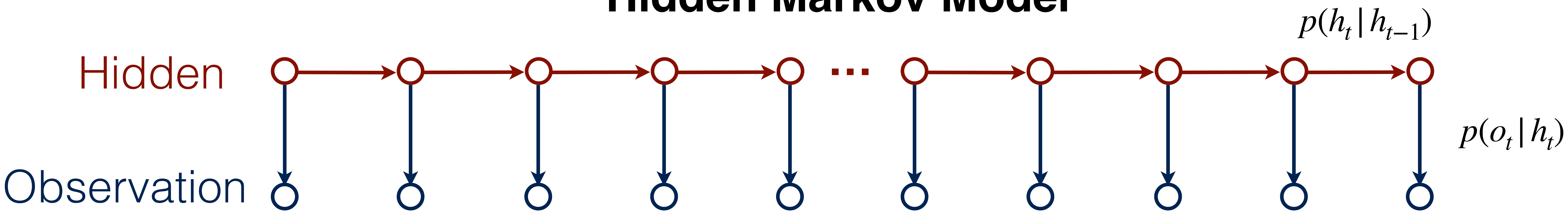
Kim and Swingle, 1711.07500



+ quantum tomography, quantum annealing, quantum error correction, holographic duality, and more

TNSAA in the era of deep learning

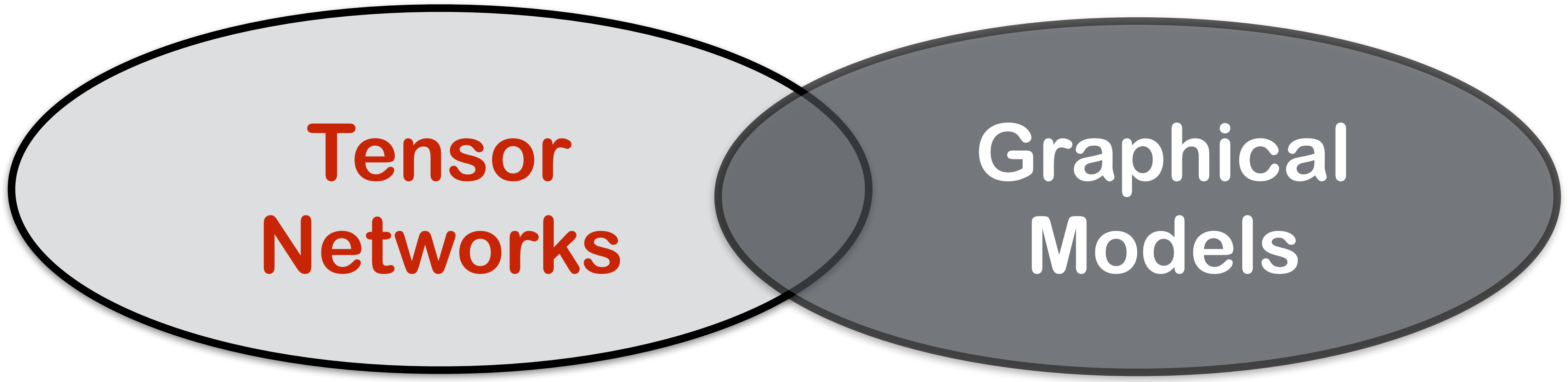
Hidden Markov Model



- Widely used in speech recognition, bioinformatics, cryptography…

- Learning, inference, and sampling algorithms (Forward-backward message passing, Viterbi, Baum-Welch) => **Think of positive MPS**

Temme, Verstraete,1003.2545
Critch, Morton, 1210.2812



Zhao, Jaeger, Neural Comput. 2010 Bailly, J. Mach. Learn. Res. 2011 Robeva, Seigal, 1710.01437 Chen, Cheng, Xie, LW, Xiang, 1701.04831

Unleash the power of tensor network states and algorithms for generative modeling

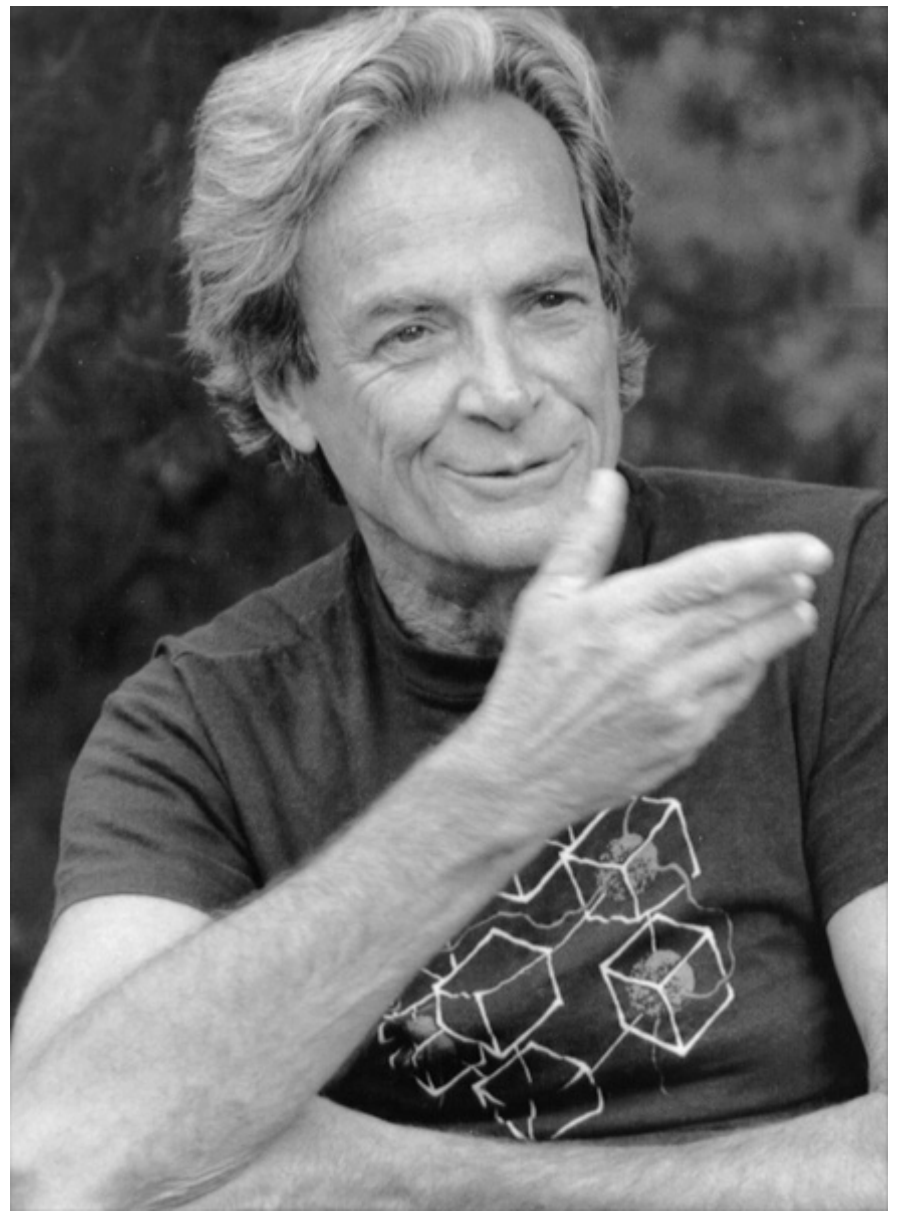
Discriminative and generative learning

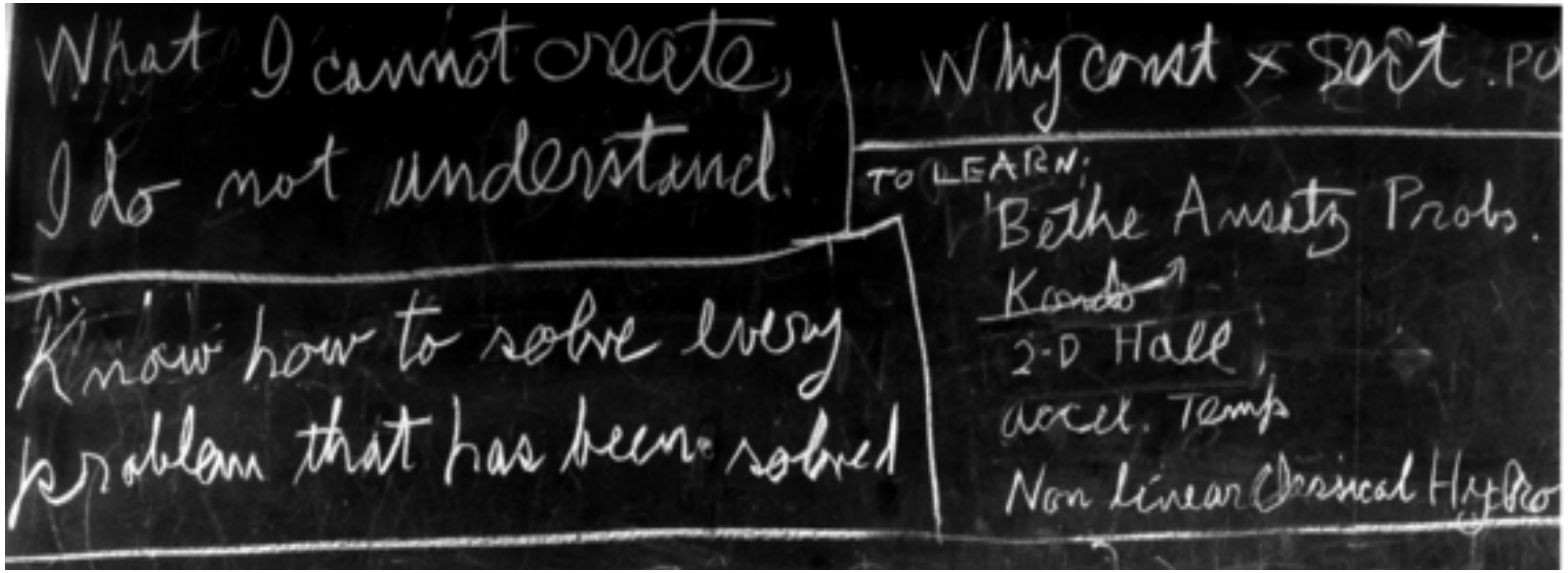
$$y = f(\boldsymbol{x})$$

or $p(y|\boldsymbol{x})$

$$p(\boldsymbol{x}, y)$$

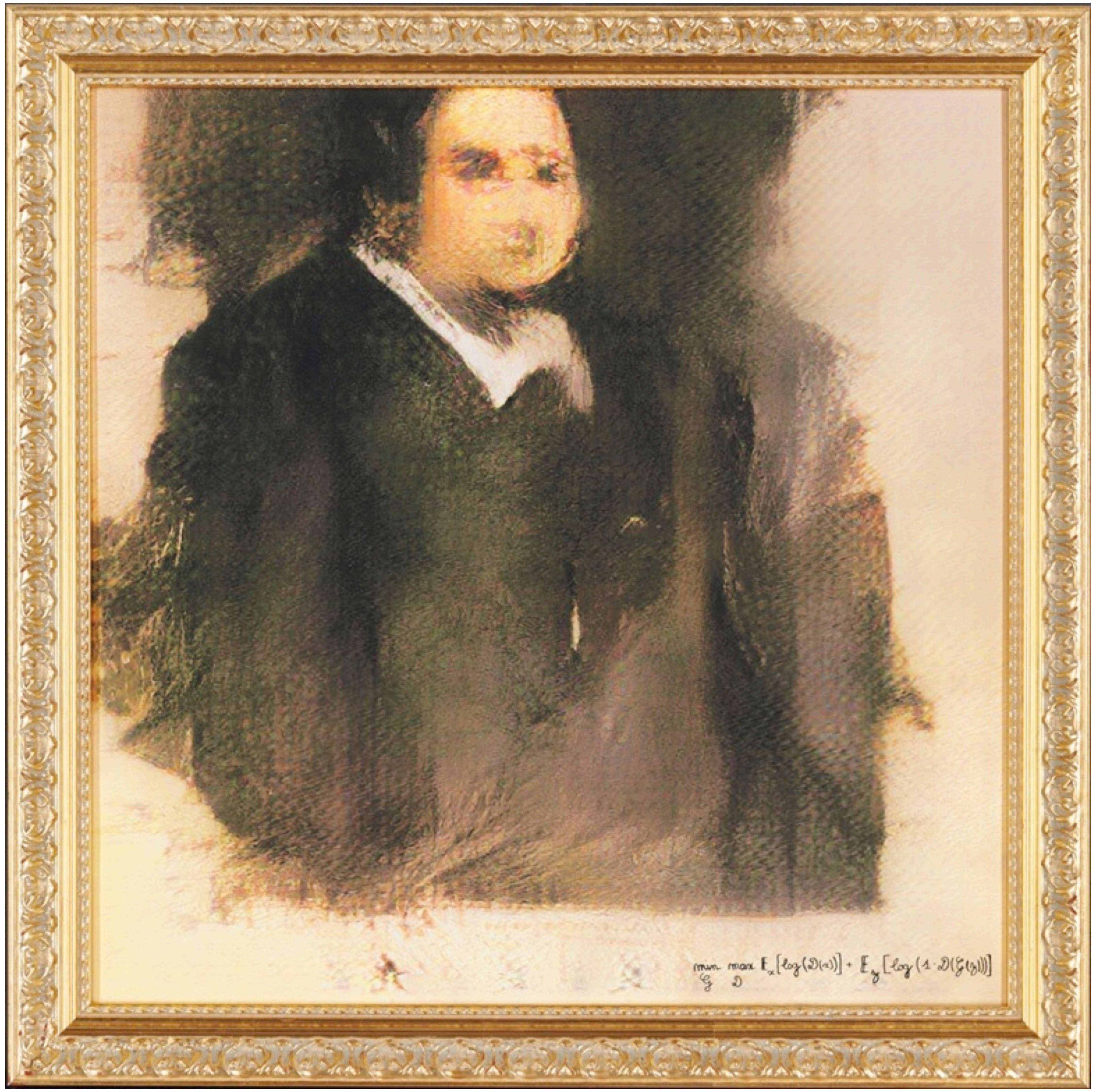
Discriminative and generative learning





"What I can not create, I do not understand"

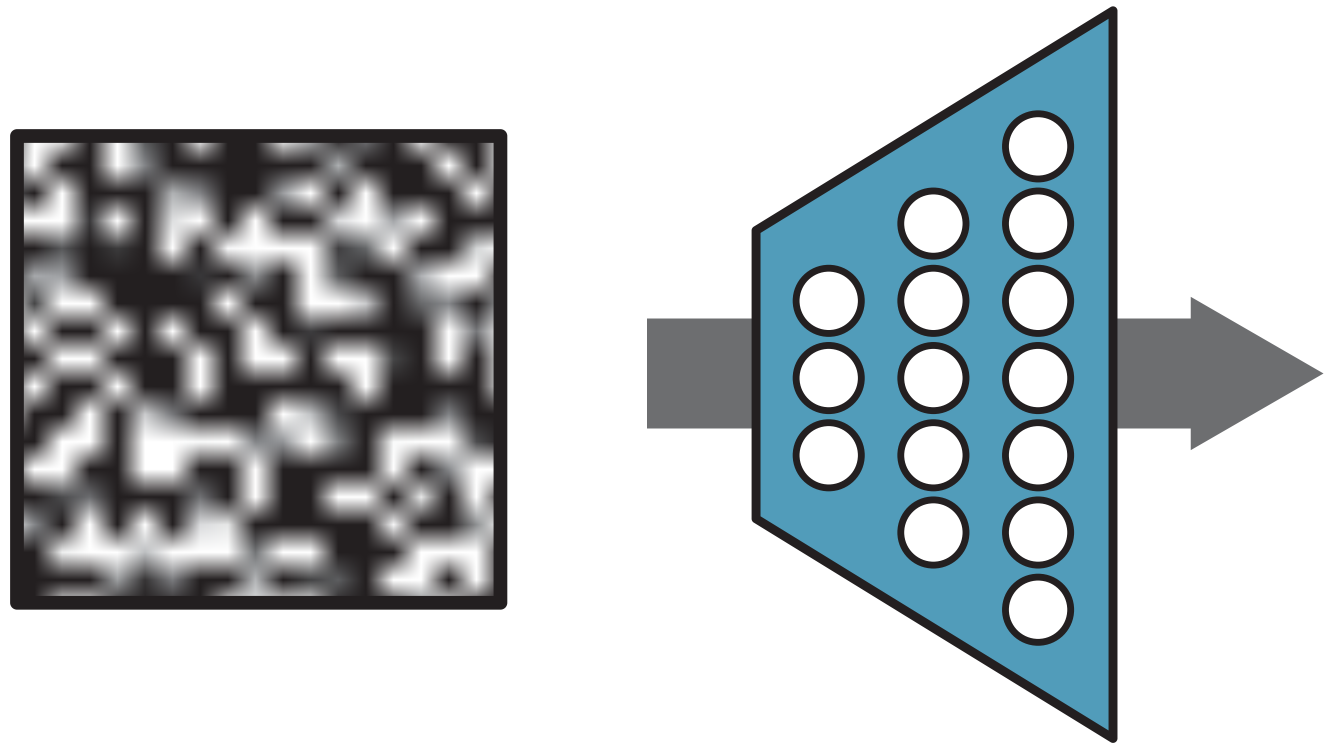
Generated Arts



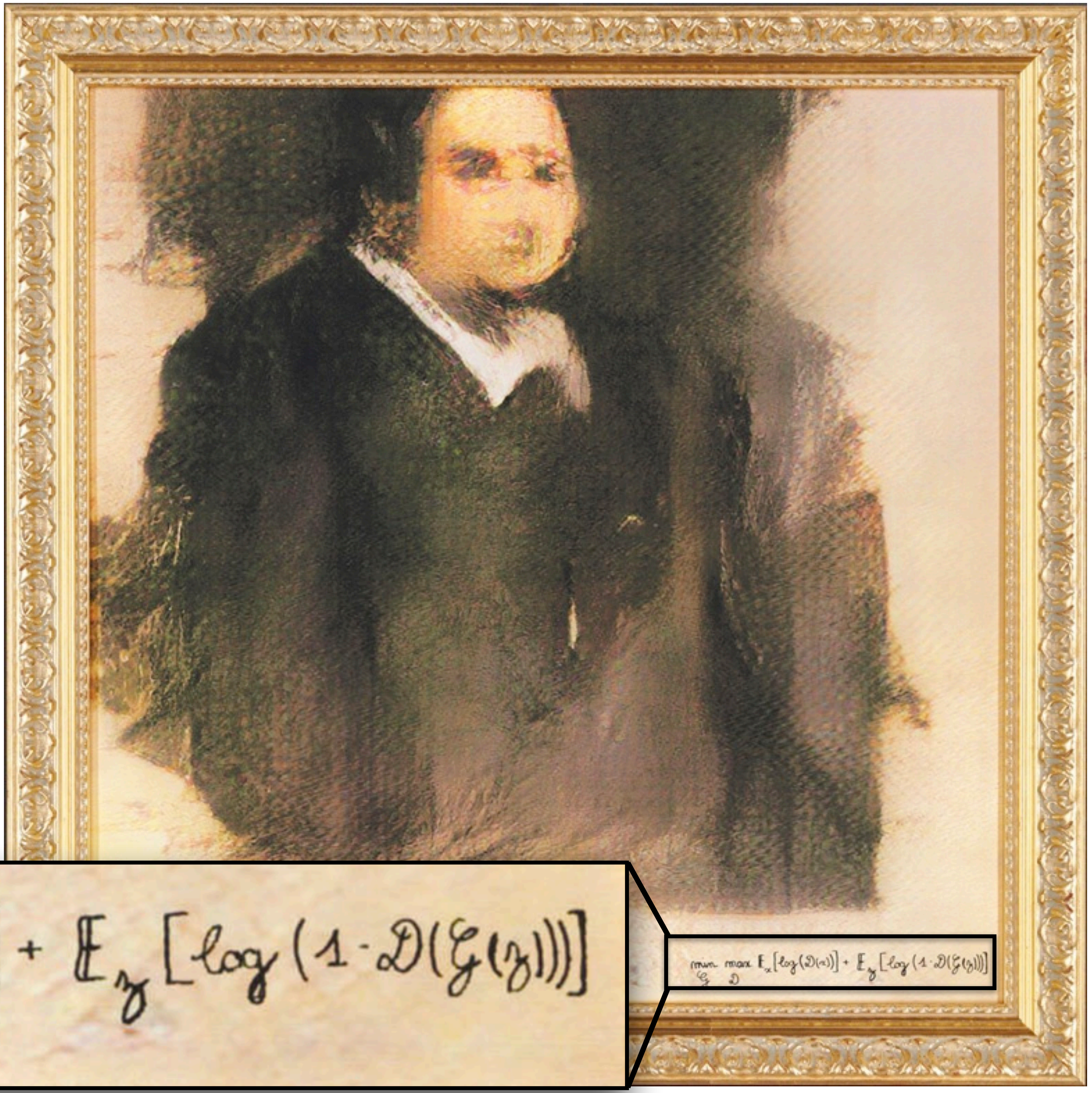
$432,500
25 October 2018
Christie's New York

https://www.christies.com/Features/A-collaboration-between-two-artists-one-human-one-a-machine-9332-1.aspx

Generated Arts



Gaussian Noise Generative Network

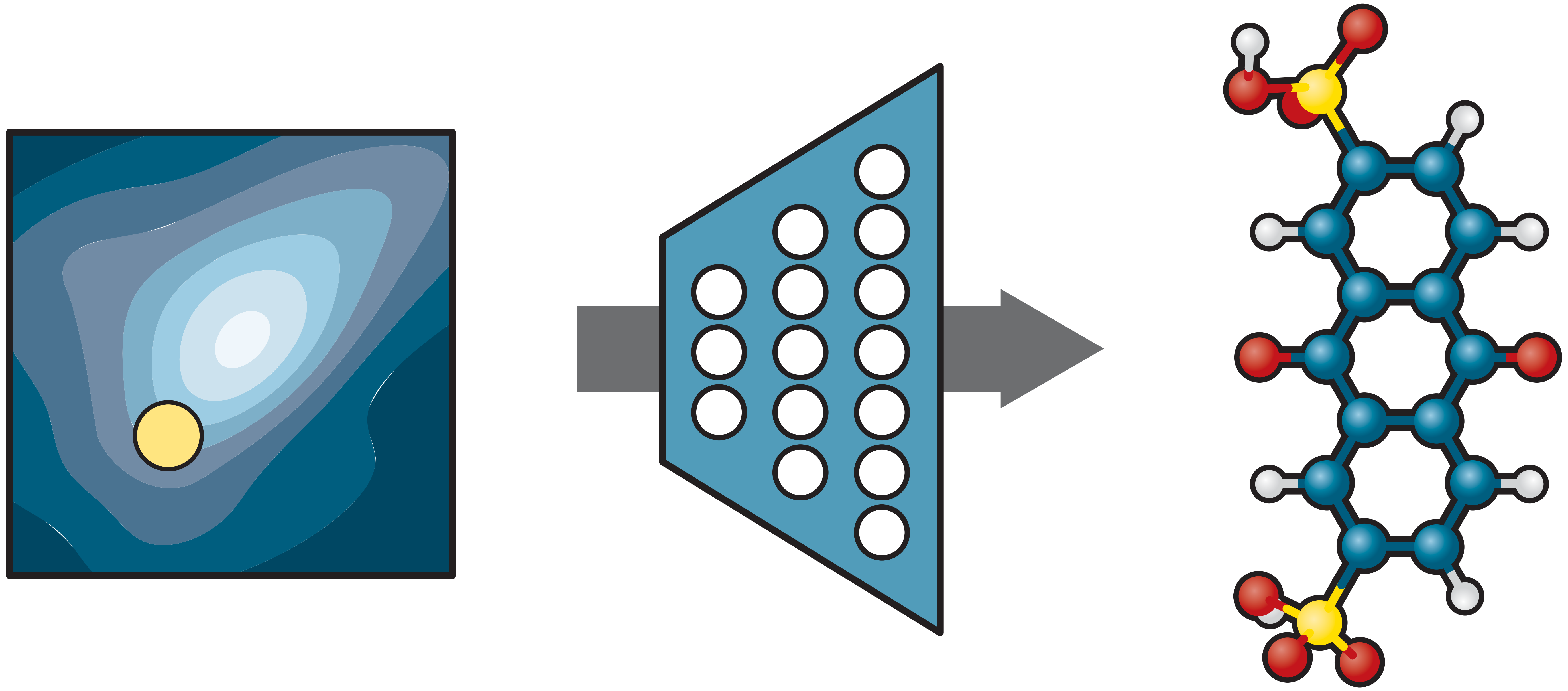


\$432,500
25 October 2018
Christie's New York

https://www.christies.com/Features/A-collaboration-between-two-artists-one-human-one-a-machine-9332-1.aspx

Generate Molecules

Latent attributes



Physical configurations

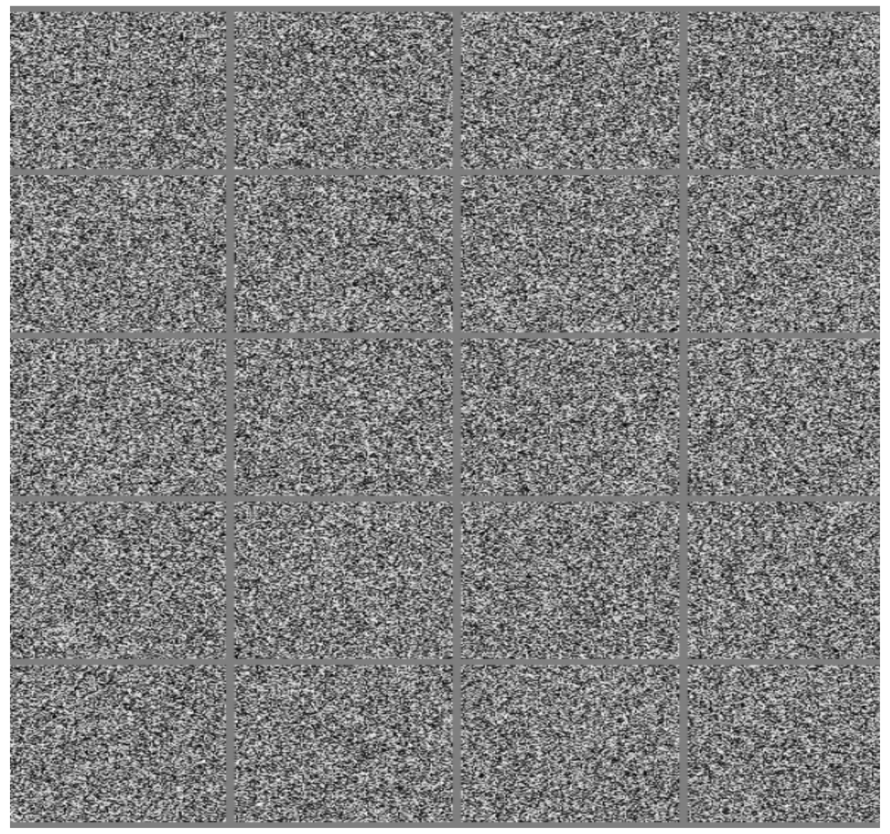
Simple Distributions $\underset{\text{Inference}}{\overset{\text{Generate}}{\rightleftarrows}}$ Complex Distribution

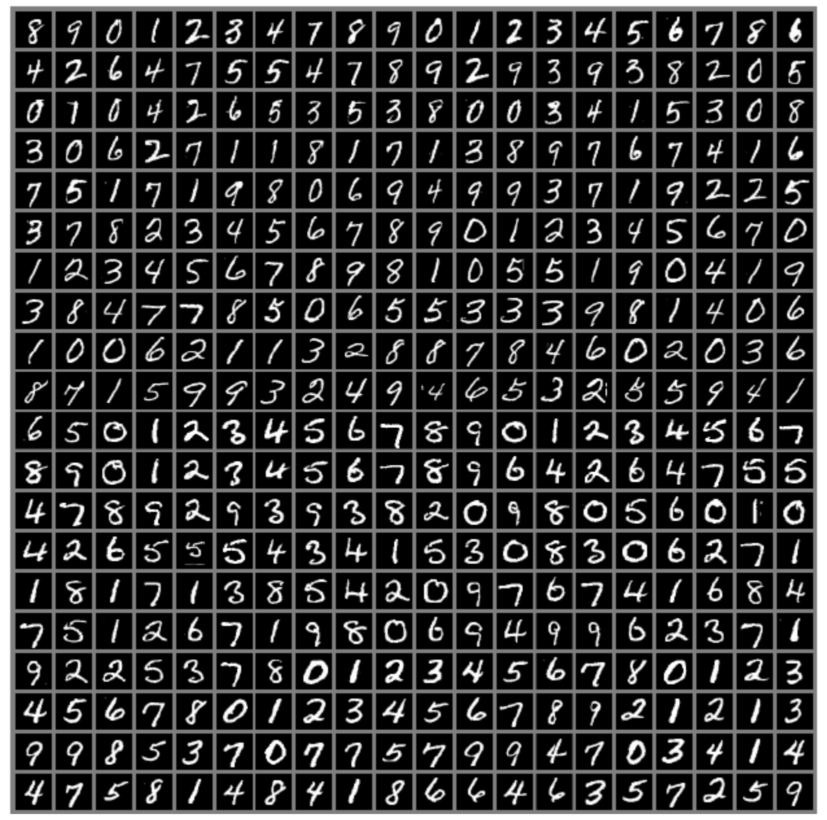
Sanchez-Lengeling & Aspuru-Guzik, Science 2018

Probabilistic Generative Modeling

$$p(\boldsymbol{x})$$

How to express, learn, and sample from a high-dimensional probability distribution ?



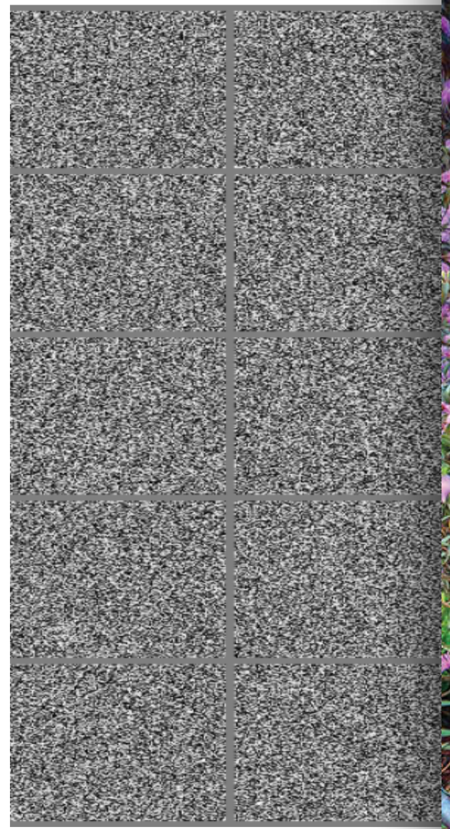


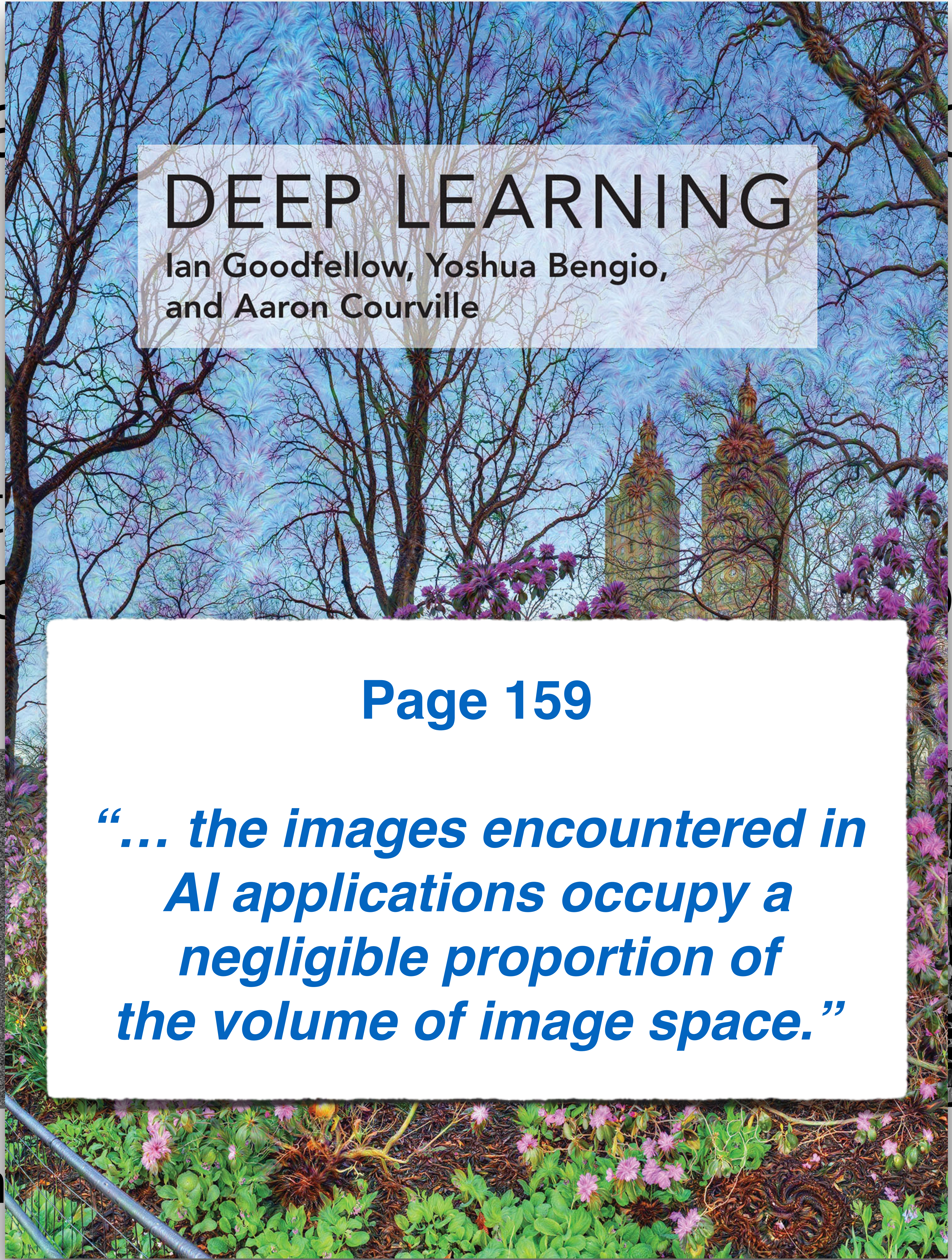
“random” images

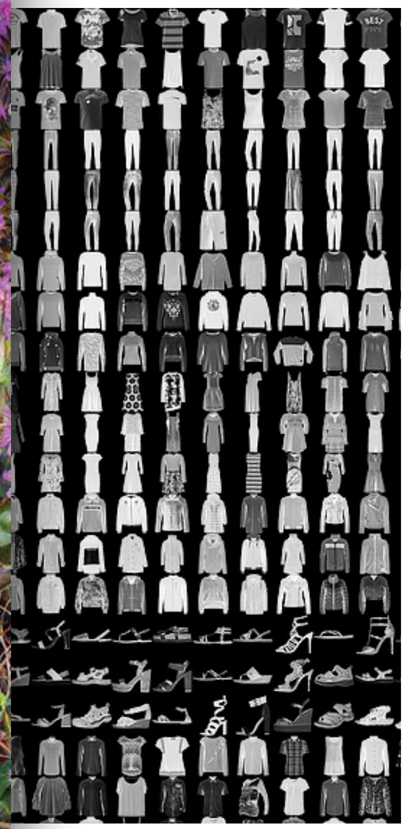
“natural” images

Proba… odeling

How t… from a
high-d… ution ?





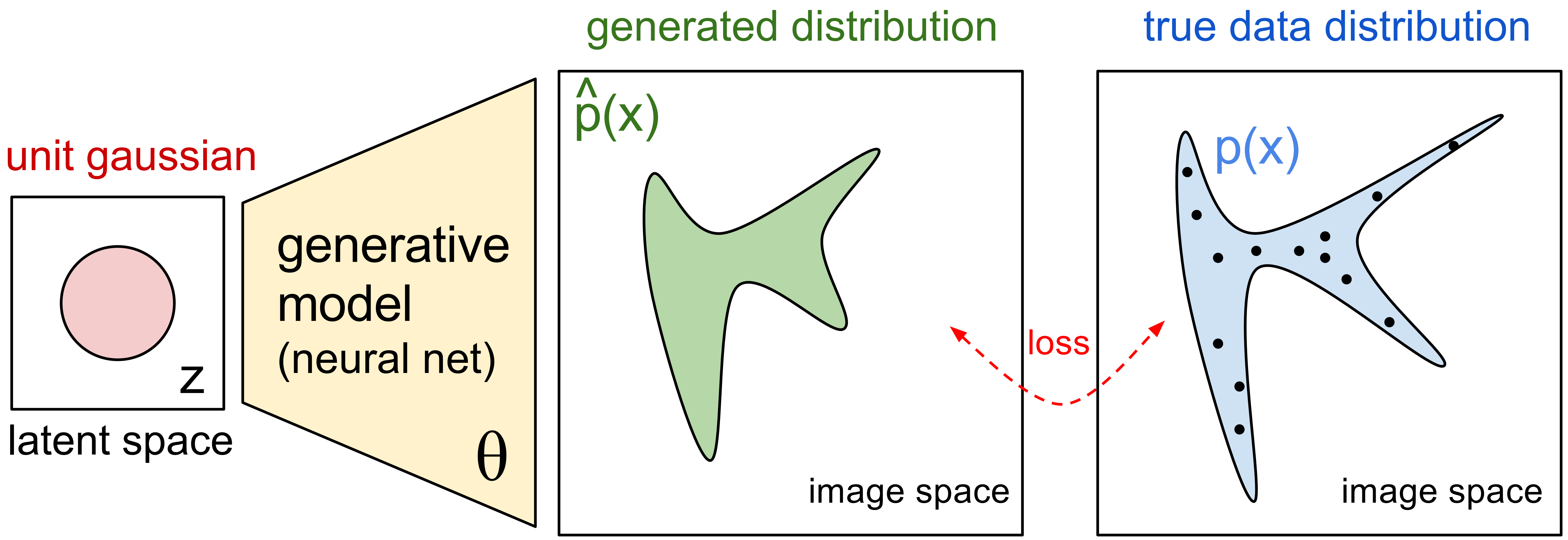


“random

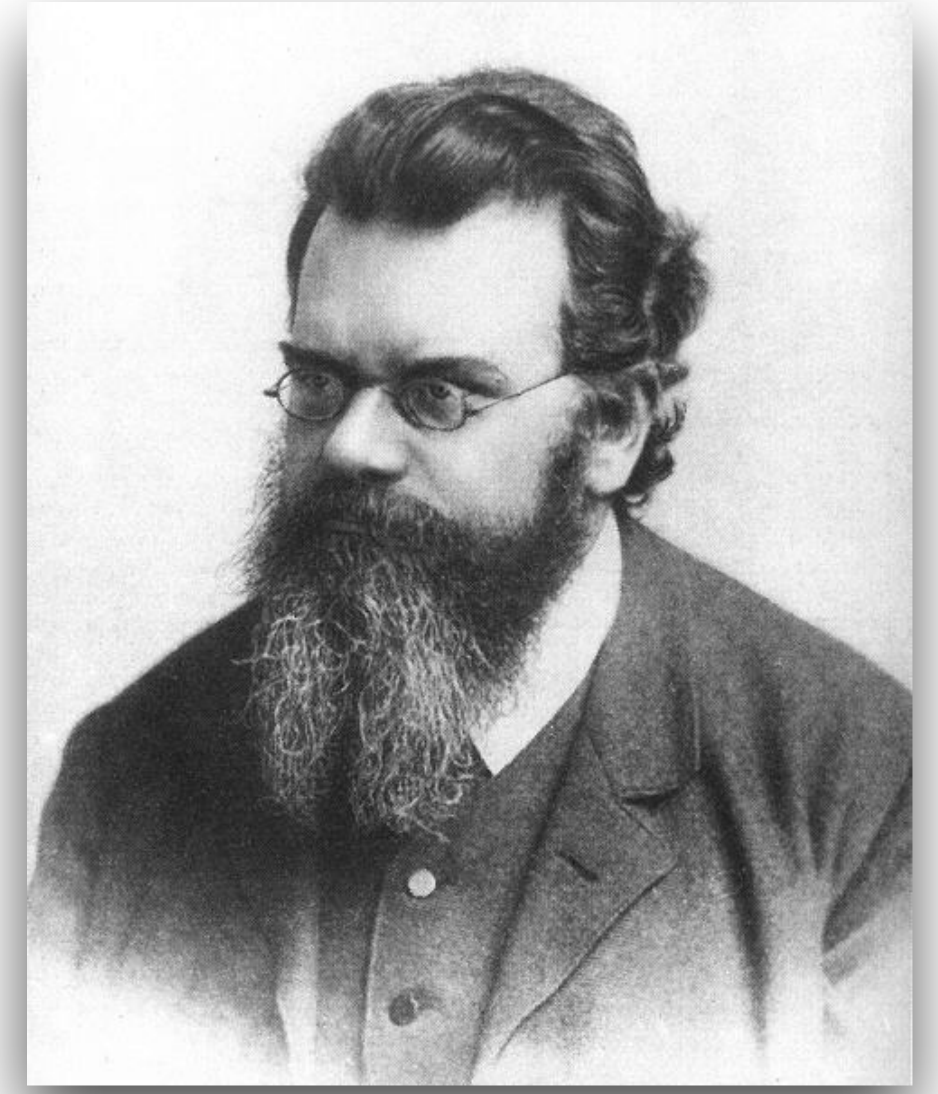
Probabilistic Generative Modeling

$$p(\boldsymbol{x})$$

How to express, learn, and sample from a high-dimensional probability distribution ?



https://blog.openai.com/generative-models/



Boltzmann Machines

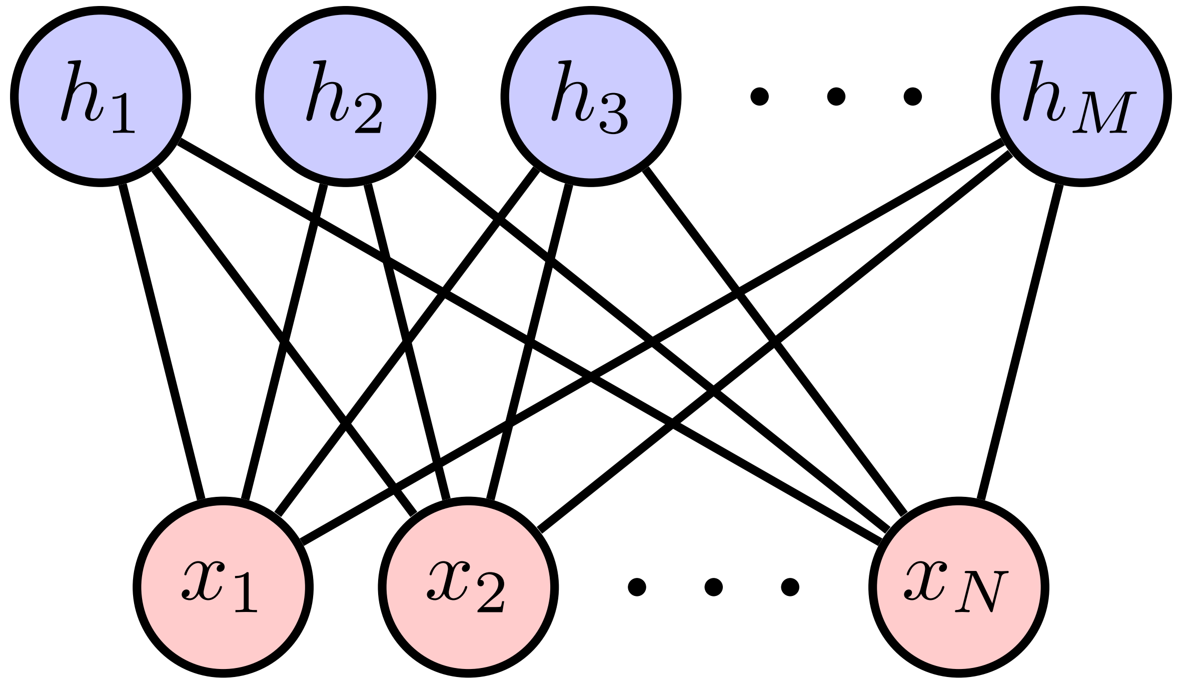
$$p(\boldsymbol{x}) = \frac{e^{-E(\boldsymbol{x})}}{Z}$$

statistical physics

Generative Modeling using Boltzmann Machines

Negative log-likelihood loss $$\mathscr{L} = -\frac{1}{|\mathscr{D}|}\sum_{\boldsymbol{x}\in\mathscr{D}} \ln p(\boldsymbol{x})$$

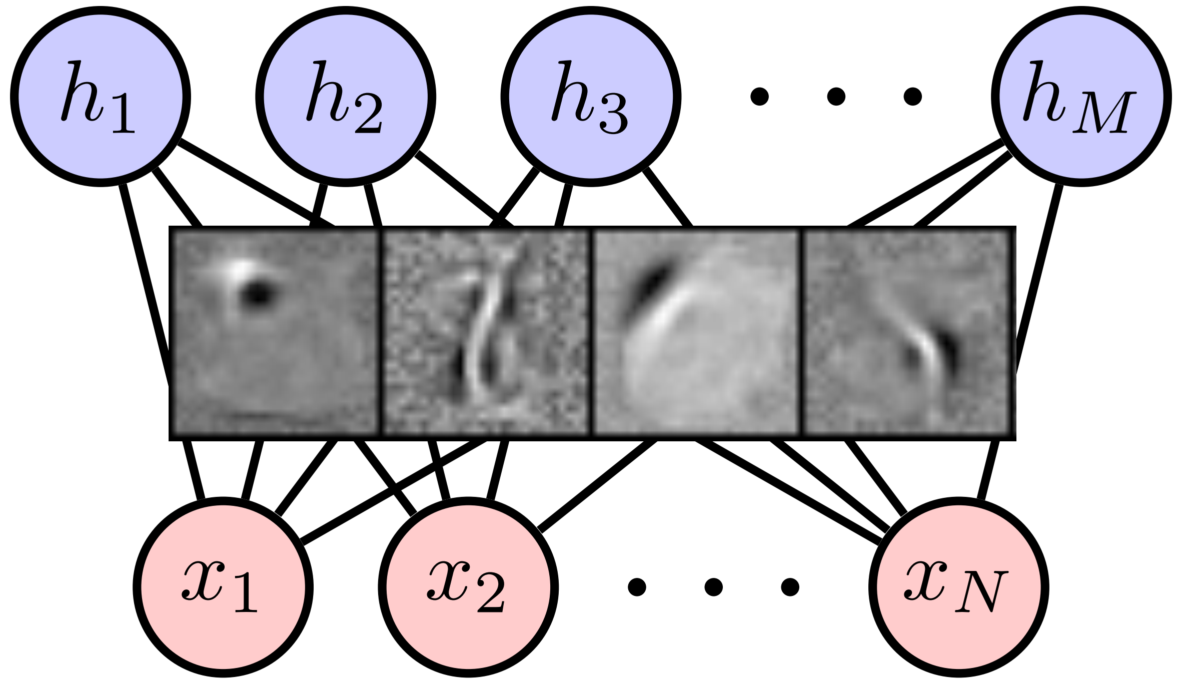
Learn



Generative Modeling using Boltzmann Machines

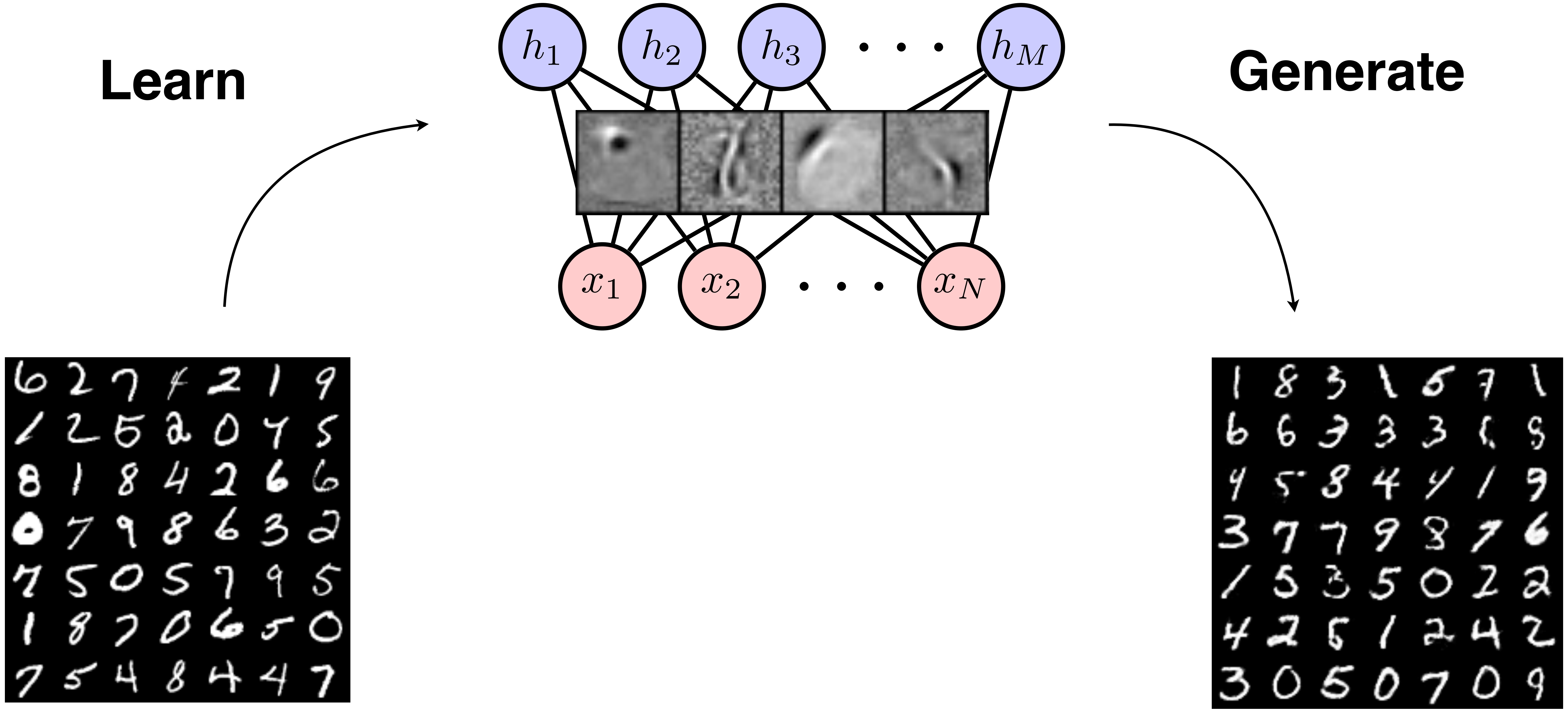
Negative log-likelihood loss $\mathscr{L} = -\frac{1}{|\mathscr{D}|}\sum_{\boldsymbol{x}\in\mathscr{D}} \ln p(\boldsymbol{x})$

Learn



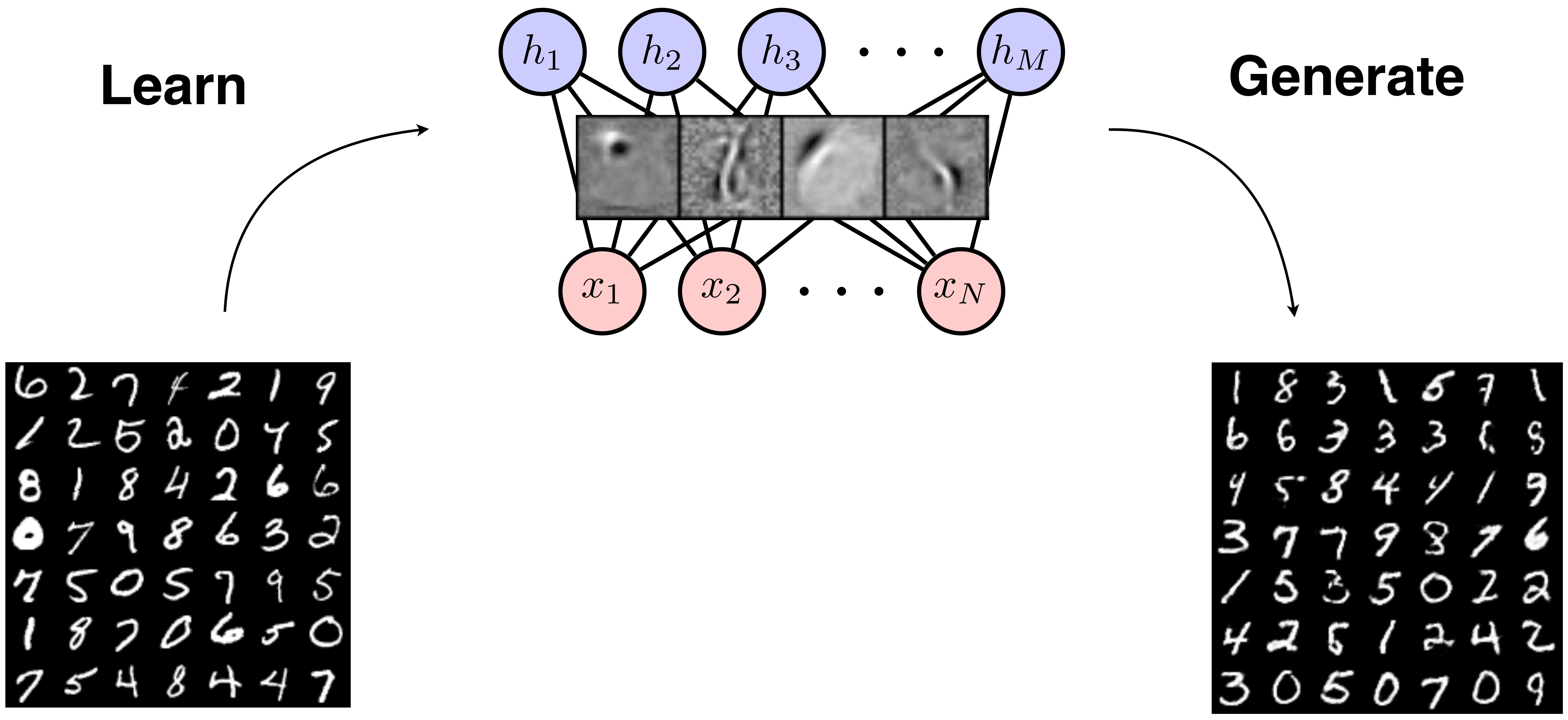
Generative Modeling using Boltzmann Machines

Negative log-likelihood loss $$\mathscr{L} = -\frac{1}{|\mathscr{D}|}\sum_{\boldsymbol{x}\in\mathscr{D}} \ln p(\boldsymbol{x})$$



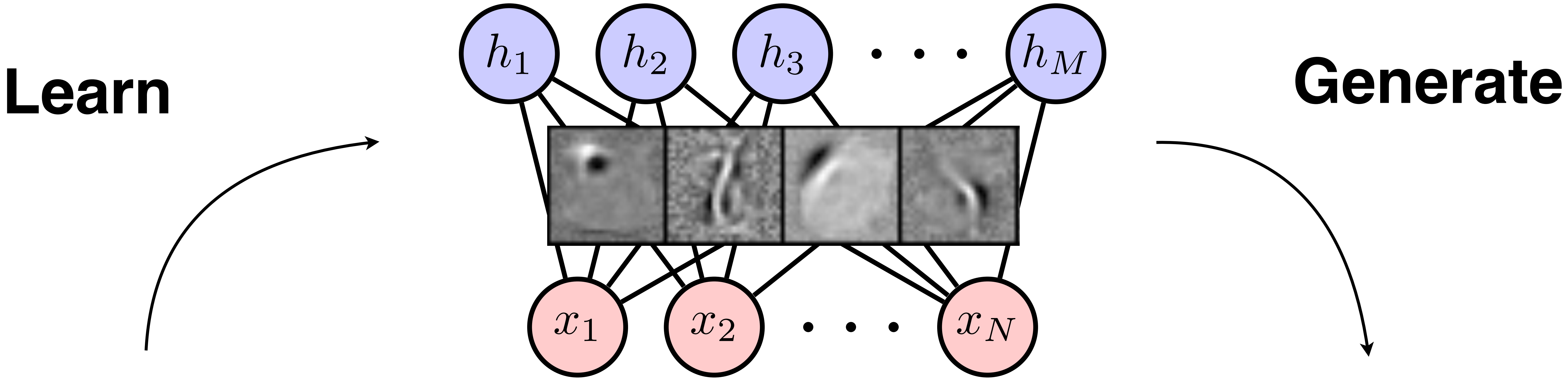
Generative Modeling using Boltzmann Machines

Negative log-likelihood loss $\mathscr{L} = -\frac{1}{|\mathscr{D}|}\sum_{\boldsymbol{x}\in\mathscr{D}} \ln p(\boldsymbol{x}) = \langle E(\boldsymbol{x})\rangle_{\boldsymbol{x}\sim\mathscr{D}} + \ln Z$



Generative Modeling using Boltzmann Machines

Negative log-likelihood loss $\mathscr{L} = -\frac{1}{|\mathscr{D}|} \sum_{\boldsymbol{x} \in \mathscr{D}} \ln p(\boldsymbol{x}) = \langle E(\boldsymbol{x}) \rangle_{\boldsymbol{x} \sim \mathscr{D}} + \ln Z$

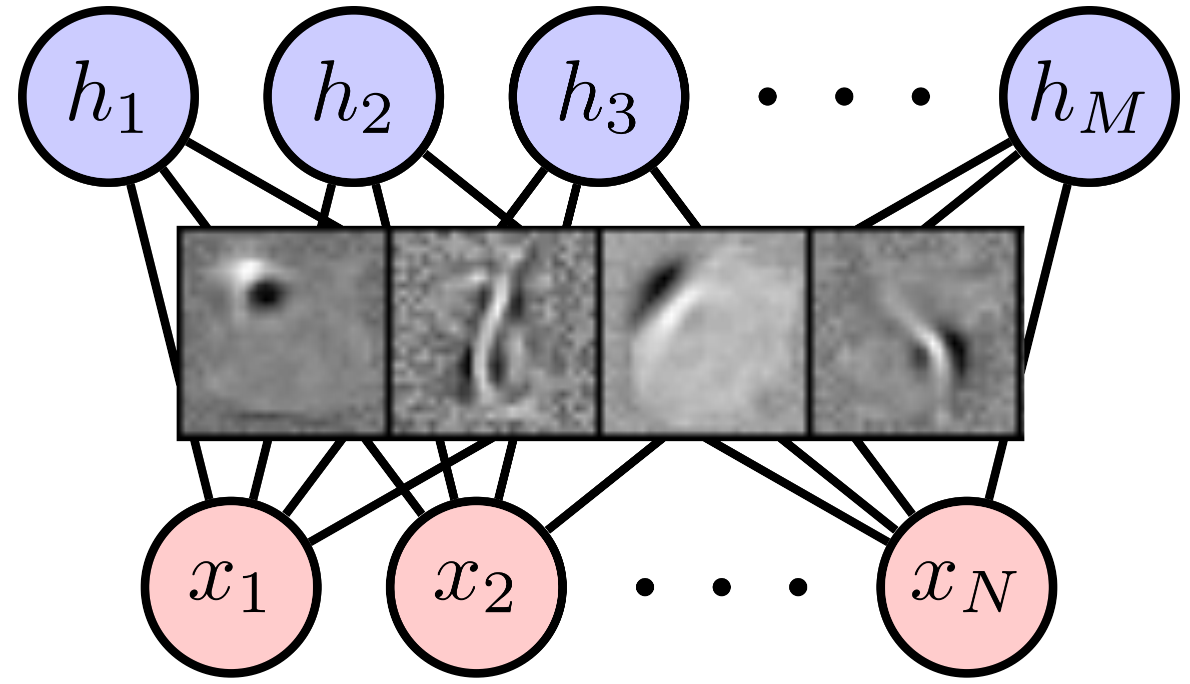


$$\nabla \mathscr{L} = \langle \nabla E \rangle_{\boldsymbol{x} \sim \mathscr{D}} - \langle \nabla E \rangle_{\boldsymbol{x} \sim p(\boldsymbol{x})}$$

Generative Modeling using Boltzmann Machines

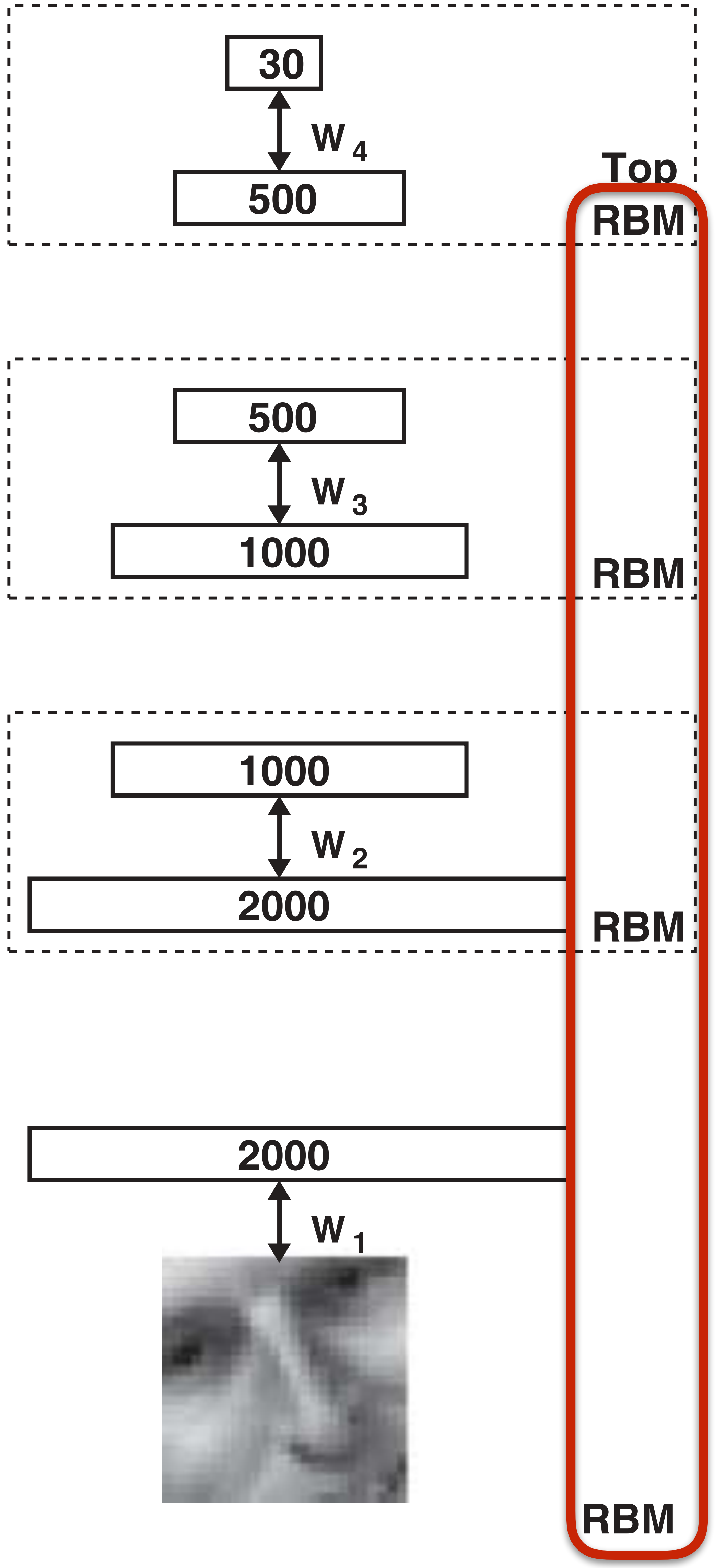
Negative log-likelihood loss $\mathscr{L} = -\frac{1}{|\mathscr{D}|}\sum_{\boldsymbol{x}\in\mathscr{D}} \ln p(\boldsymbol{x}) = \langle E(\boldsymbol{x})\rangle_{\boldsymbol{x}\sim\mathscr{D}} + \ln Z$

Learn

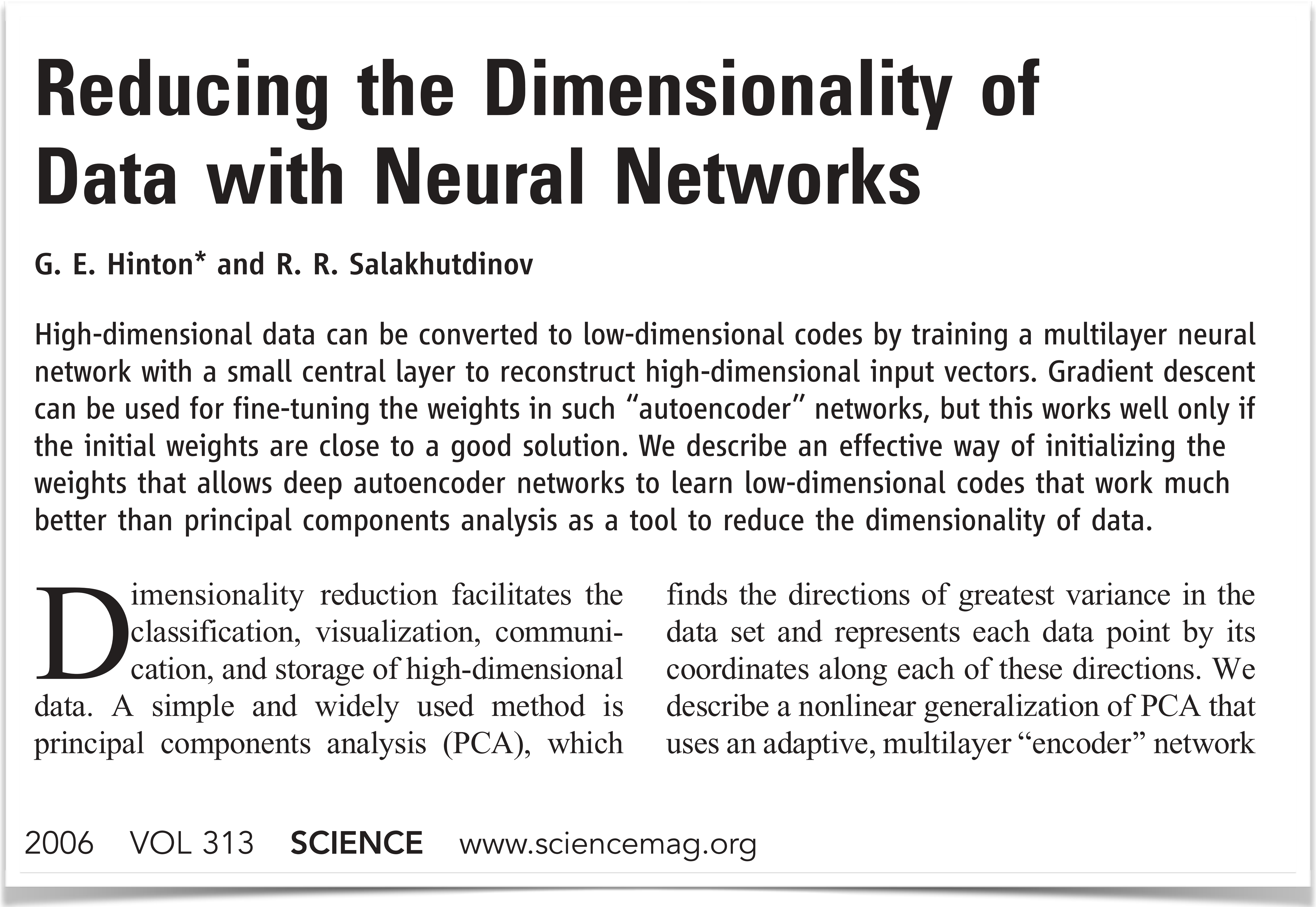


Generate

$$\nabla\mathscr{L} = \langle \nabla E\rangle_{\boldsymbol{x}\sim\mathscr{D}} - \langle \nabla E\rangle_{\boldsymbol{x}\sim p(\boldsymbol{x})}$$



Pretraining



Reducing the Dimensionality of Data with Neural Networks

G. E. Hinton* and R. R. Salakhutdinov

High-dimensional data can be converted to low-dimensional codes by training a multilayer neural network with a small central layer to reconstruct high-dimensional input vectors. Gradient descent can be used for fine-tuning the weights in such "autoencoder" networks, but this works well only if the initial weights are close to a good solution. We describe an effective way of initializing the weights that allows deep autoencoder networks to learn low-dimensional codes that work much better than principal components analysis as a tool to reduce the dimensionality of data.

Dimensionality reduction facilitates the classification, visualization, communication, and storage of high-dimensional data. A simple and widely used method is principal components analysis (PCA), which finds the directions of greatest variance in the data set and represents each data point by its coordinates along each of these directions. We describe a nonlinear generalization of PCA that uses an adaptive, multilayer "encoder" network

2006 VOL 313 **SCIENCE** www.sciencemag.org

Renaissance of deep learning

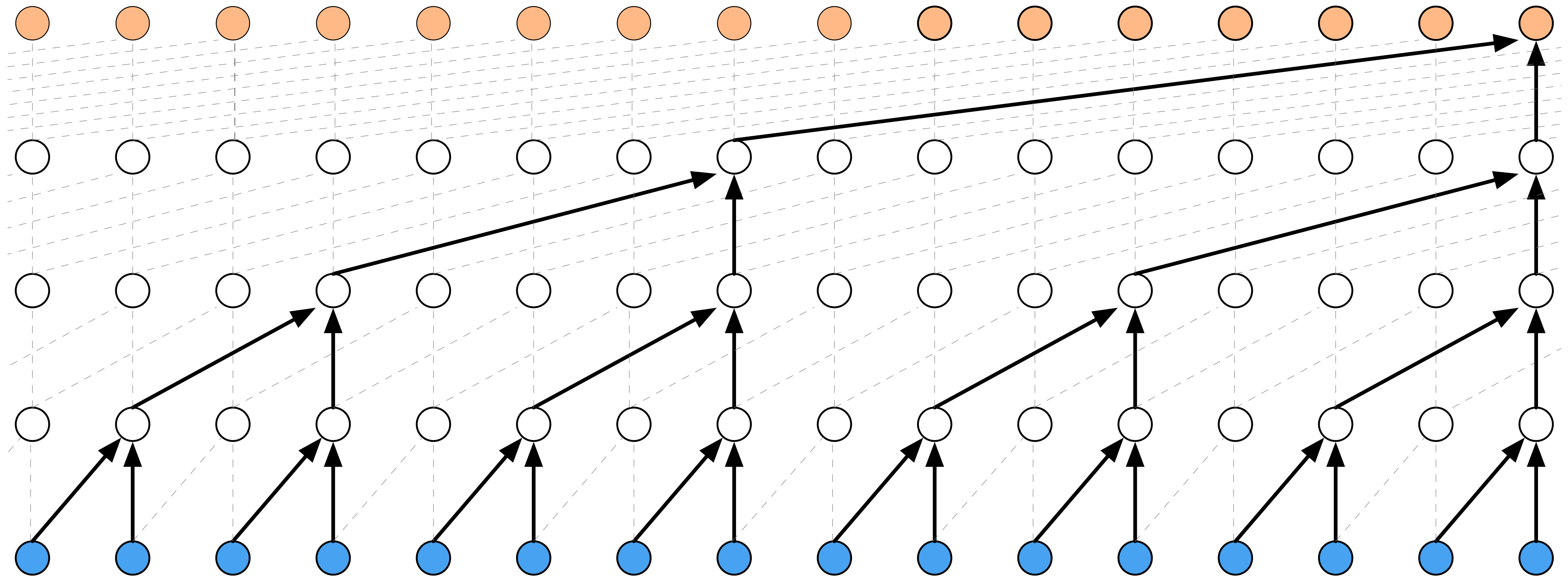
Feedback to physics

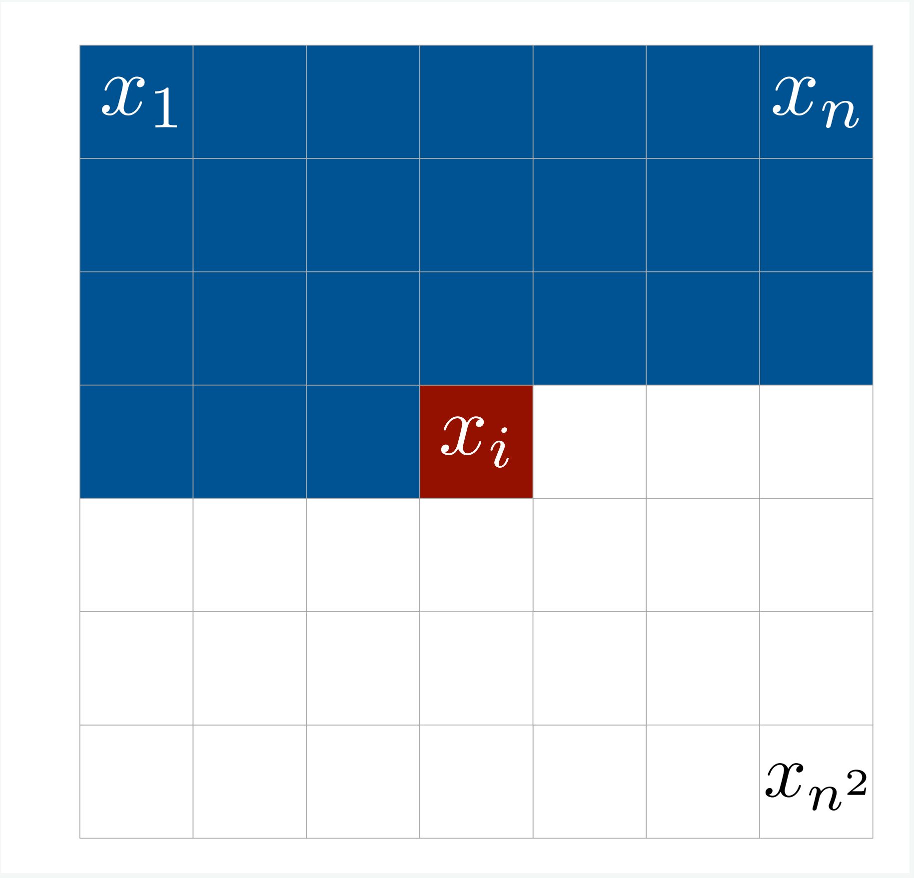
Wavefunctions ansatz
Quantum state tomography
Quantum error correction

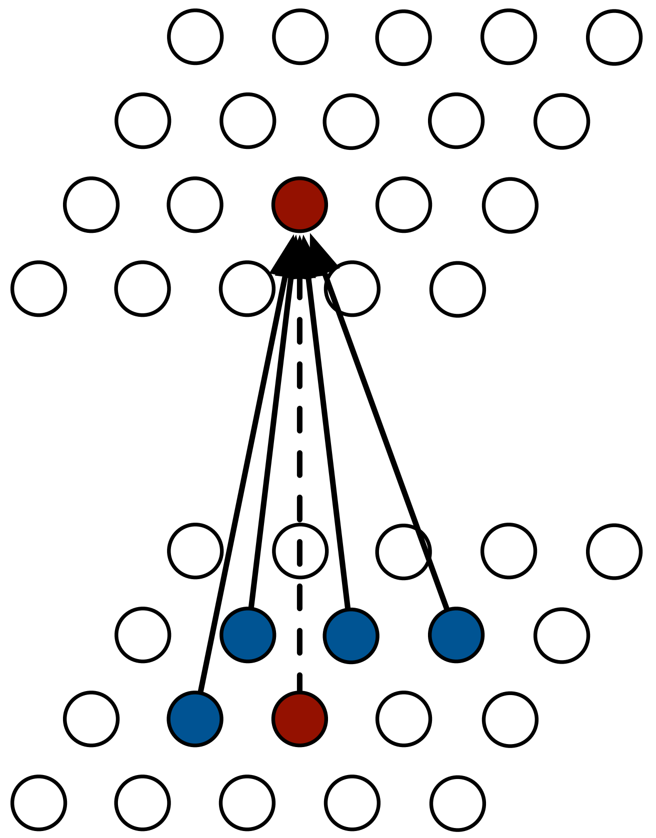
Monte Carlo updates
Renormalization group
…… **see next talk !**

State-of-the-Art: Autoregressive Models

$$p(\boldsymbol{x}) = \prod_i p(x_i \,|\, \boldsymbol{x}_{<i})$$







Speech data

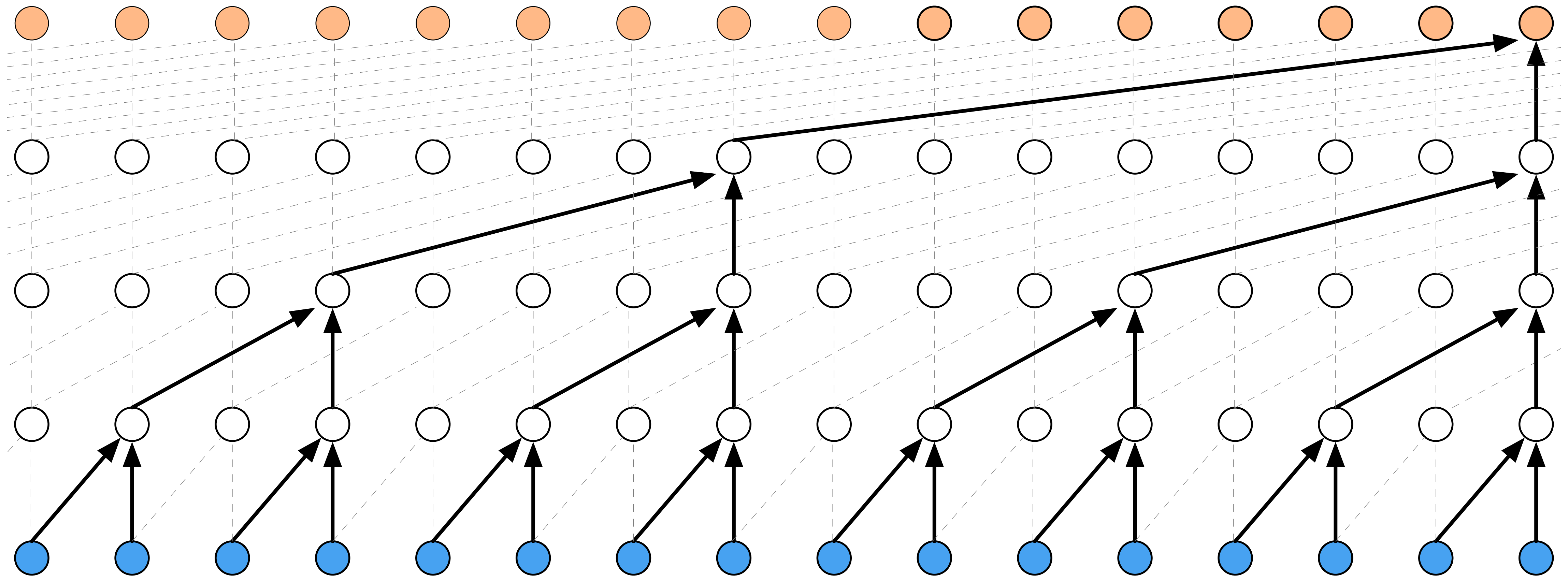
WaveNet 1609.03499,1711.10433

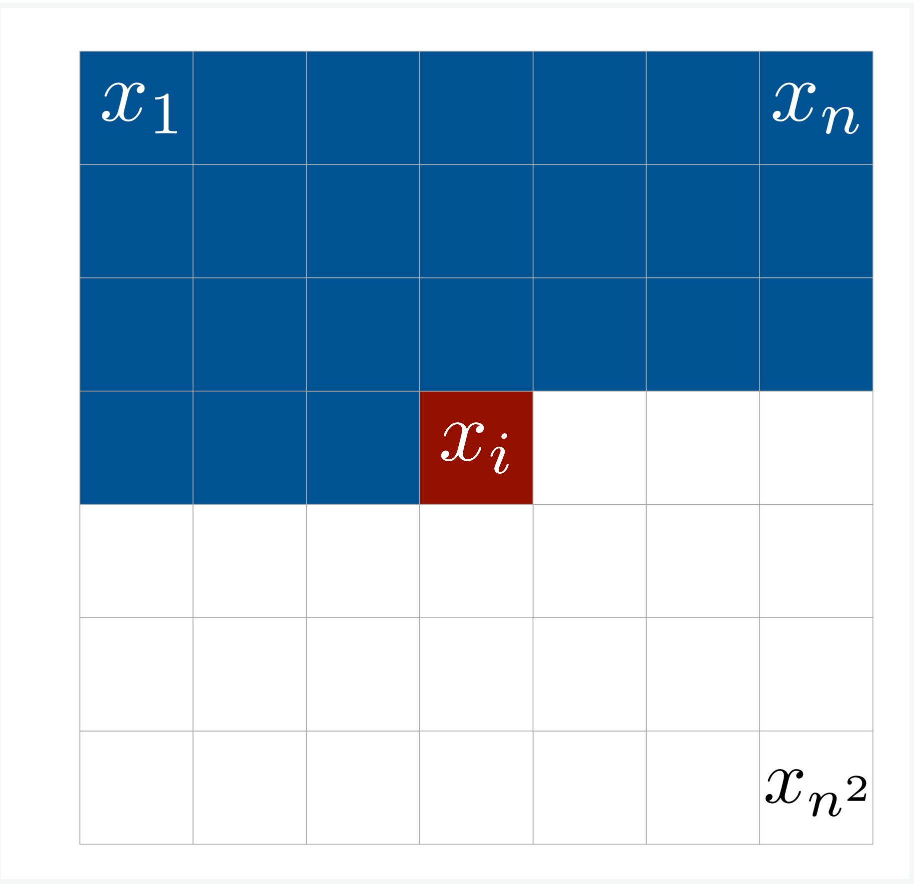
Image data

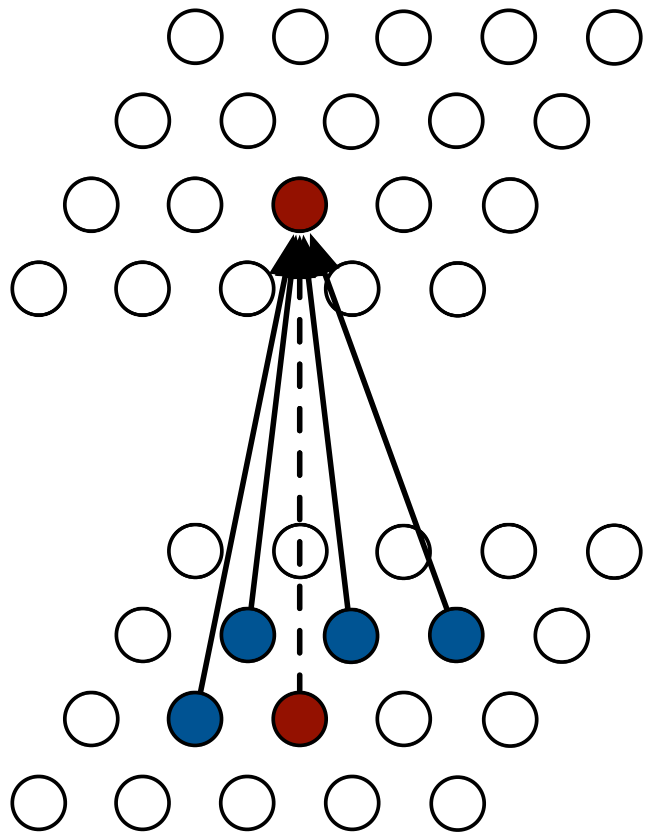
PixelCNN 1601.06759, 1606.05328

State-of-the-Art: Autoregressive Models

$$p(\boldsymbol{x}) = \prod_i p(x_i \,|\, \boldsymbol{x}_{<i})$$
$$= p(x_1)p(x_2 \,|\, x_1)p(x_3 \,|\, x_1, x_2)\cdots$$







Speech data

WaveNet 1609.03499,1711.10433

Image data

PixelCNN 1601.06759, 1606.05328

WaveNet in the Real World

2018 Google I/O

https://deepmind.com/blog/wavenet-generative-model-raw-audio/
https://deepmind.com/blog/high-fidelity-speech-synthesis-wavenet/
https://deepmind.com/blog/wavenet-launches-google-assistant/

WaveNet in the Real World

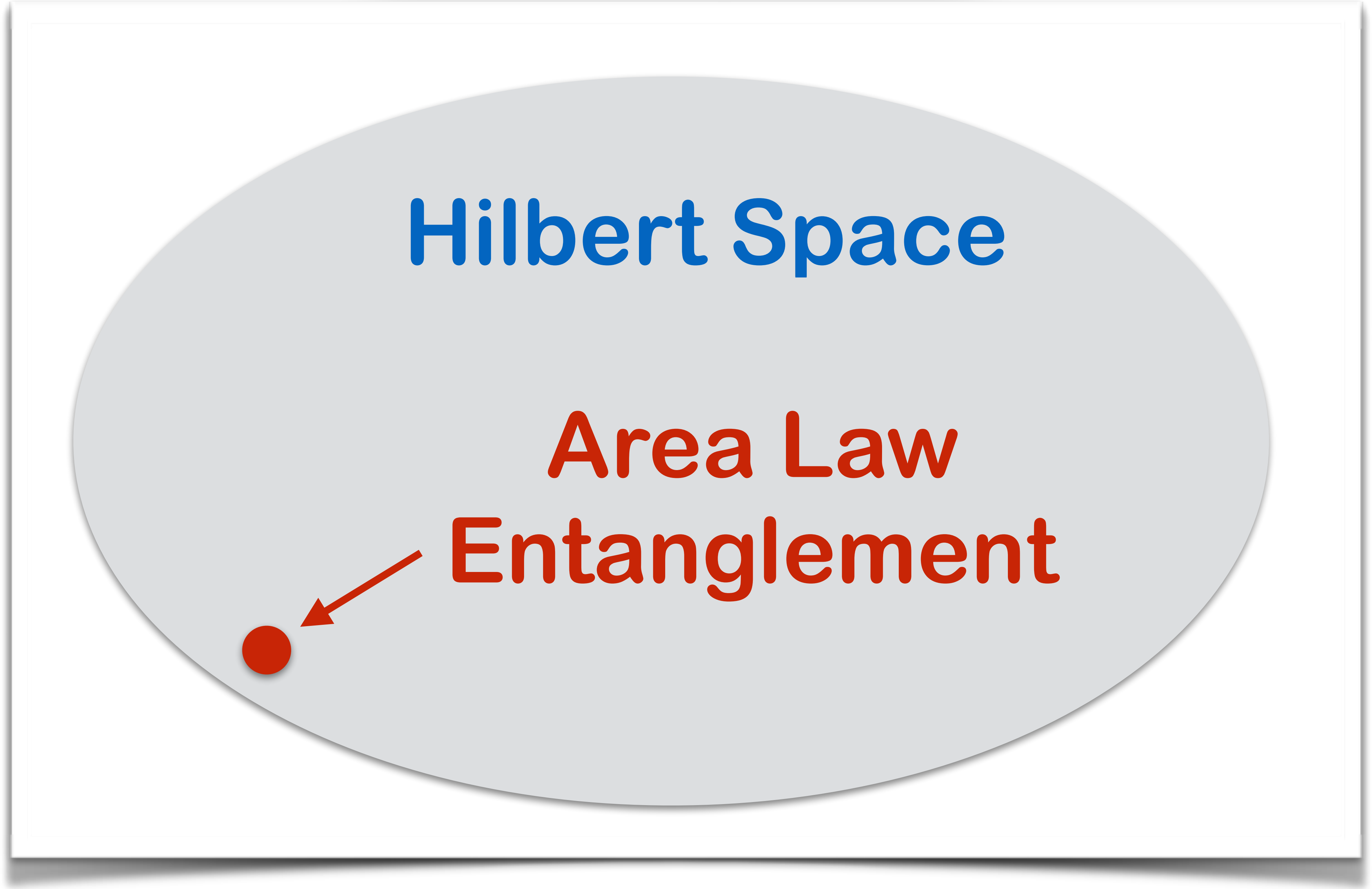
2018 Google I/O

https://deepmind.com/blog/wavenet-generative-model-raw-audio/
https://deepmind.com/blog/high-fidelity-speech-synthesis-wavenet/
https://deepmind.com/blog/wavenet-launches-google-assistant/

Born Machines

$$p(\boldsymbol{x}) = \frac{|\Psi(\boldsymbol{x})|^2}{Z}$$

quantum physics

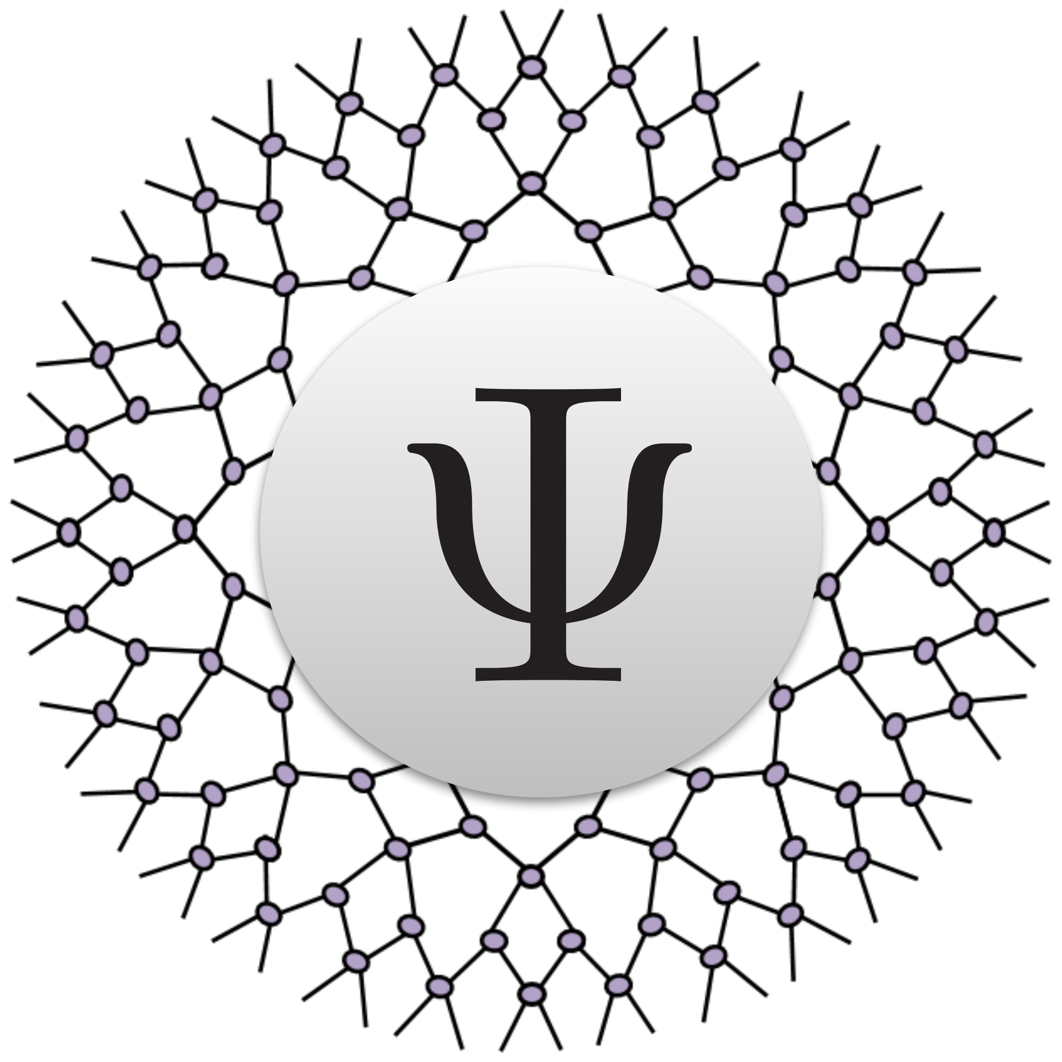


Born Machines

$$p(\boldsymbol{x}) = \frac{|\Psi(\boldsymbol{x})|^2}{Z}$$

quantum physics

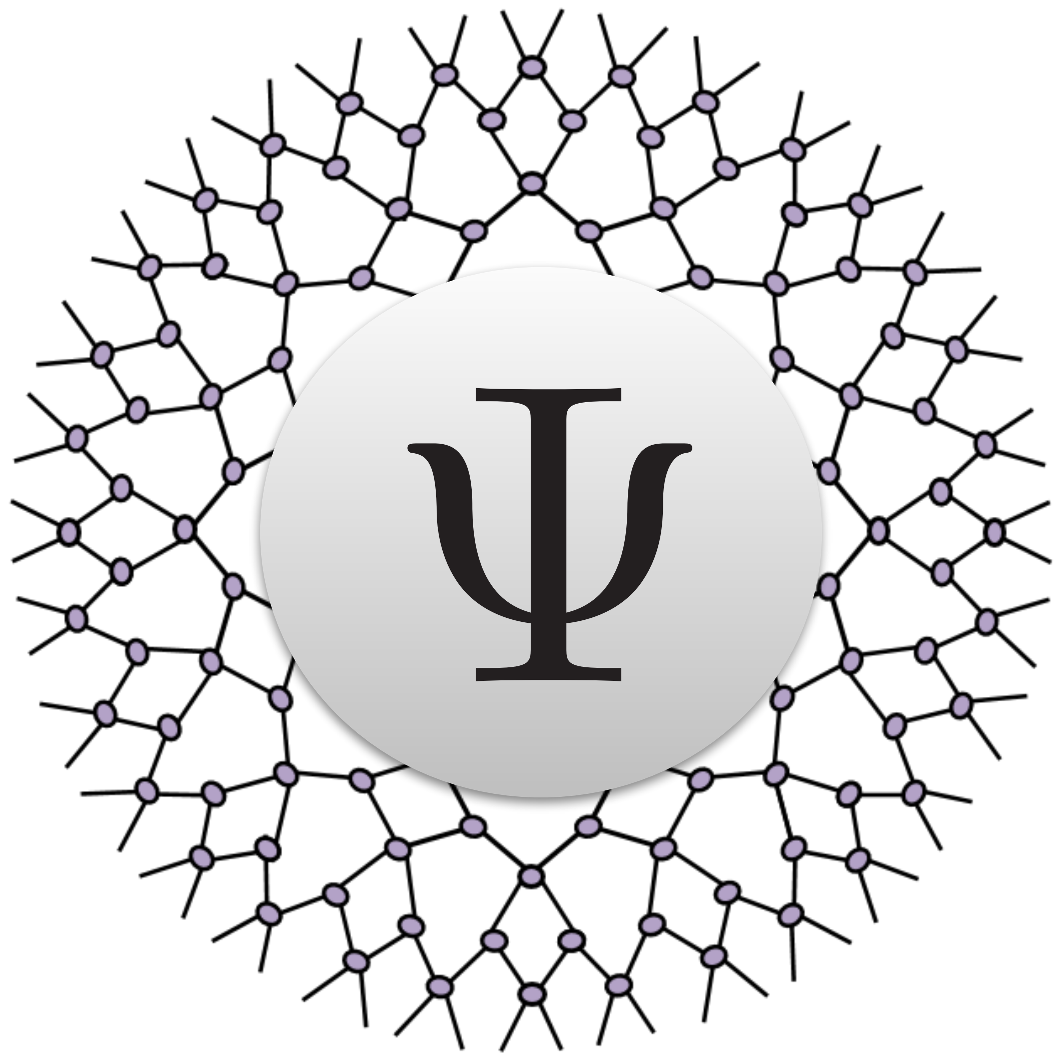
Quantum inspired generative modeling

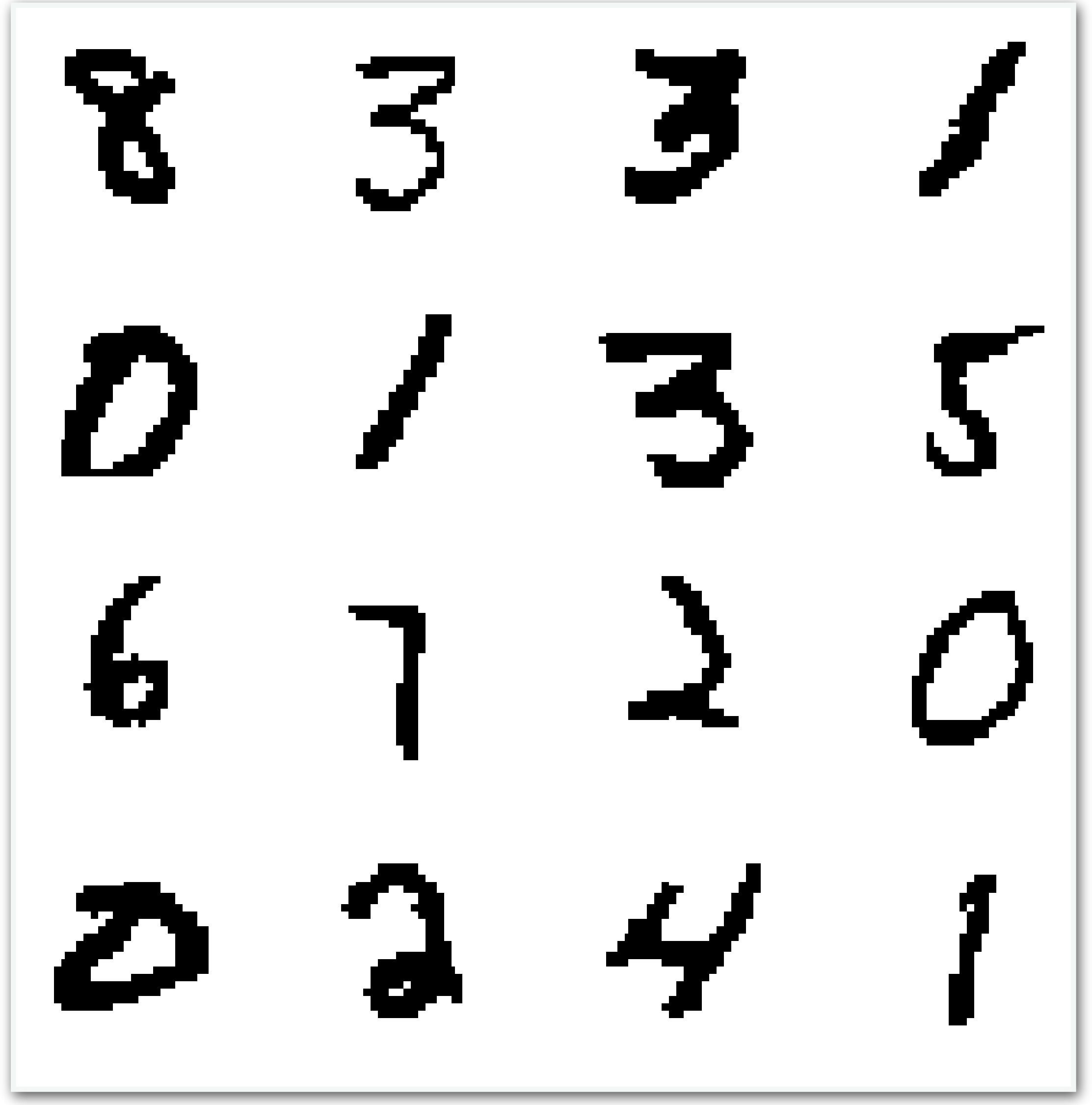


Han, Wang, Fan, LW, Zhang, 1709.01662, PRX 18’

Quantum inspired generative modeling

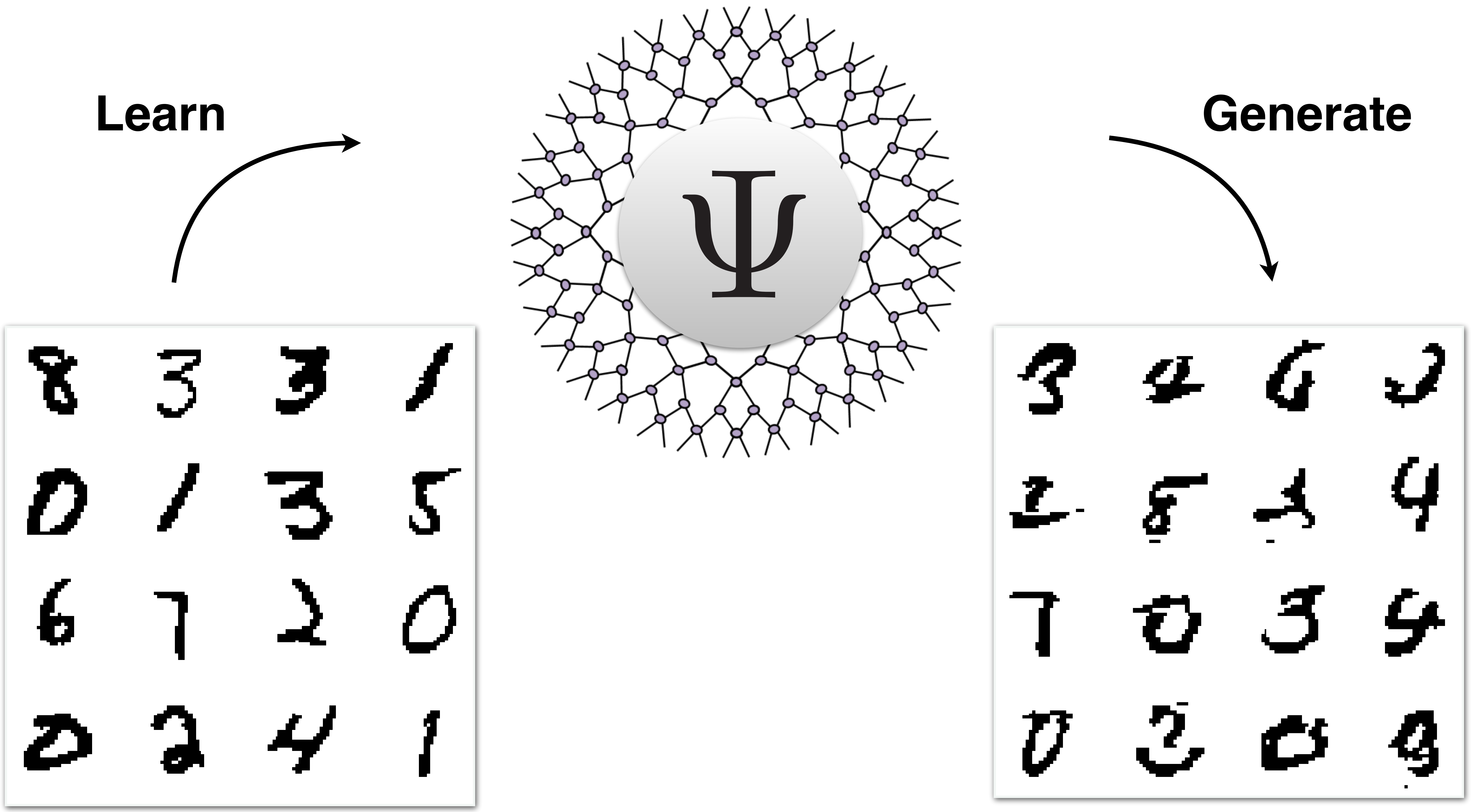
Learn





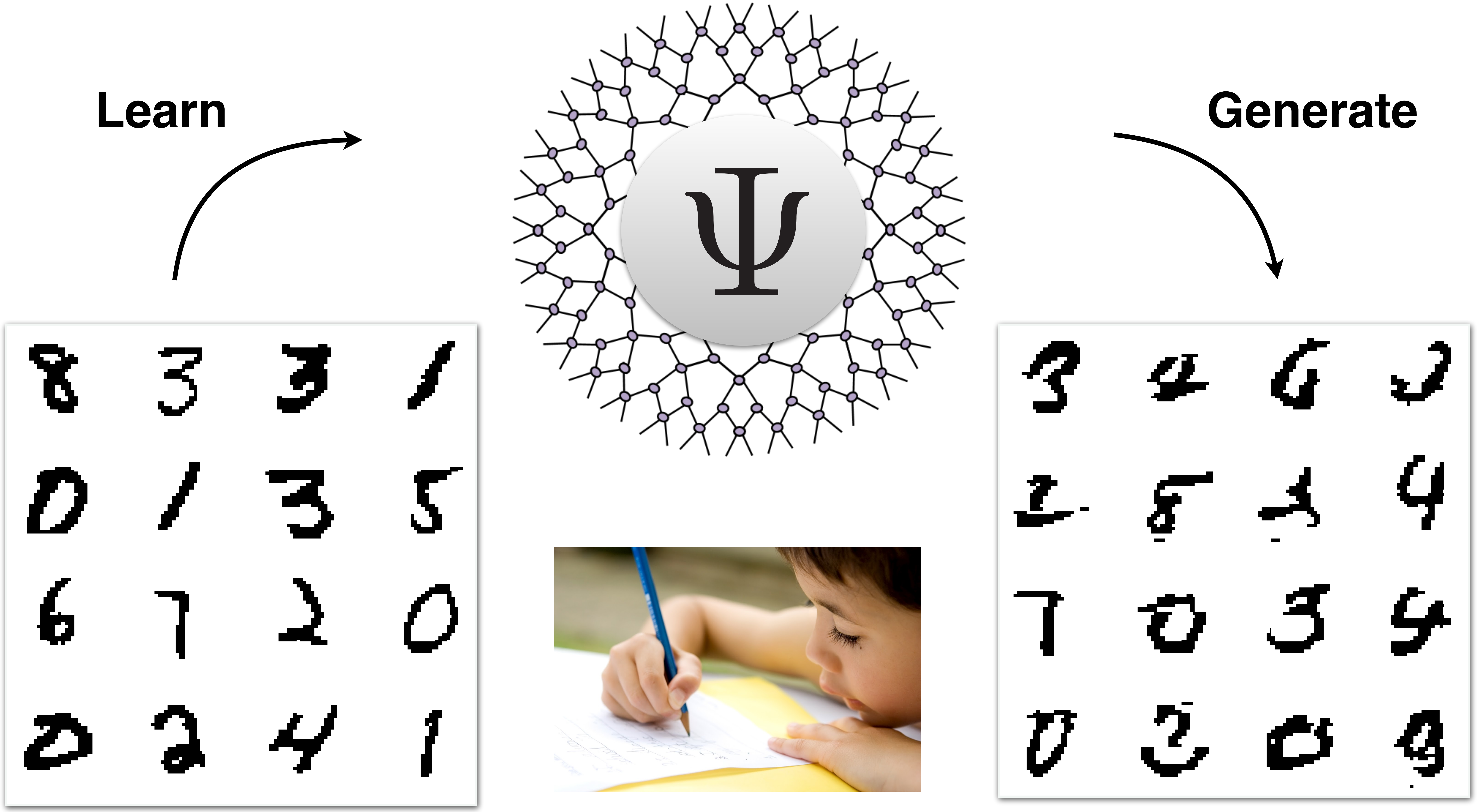
Han, Wang, Fan, LW, Zhang, 1709.01662, PRX 18’

Quantum inspired generative modeling



Han, Wang, Fan, LW, Zhang, 1709.01662, PRX 18’

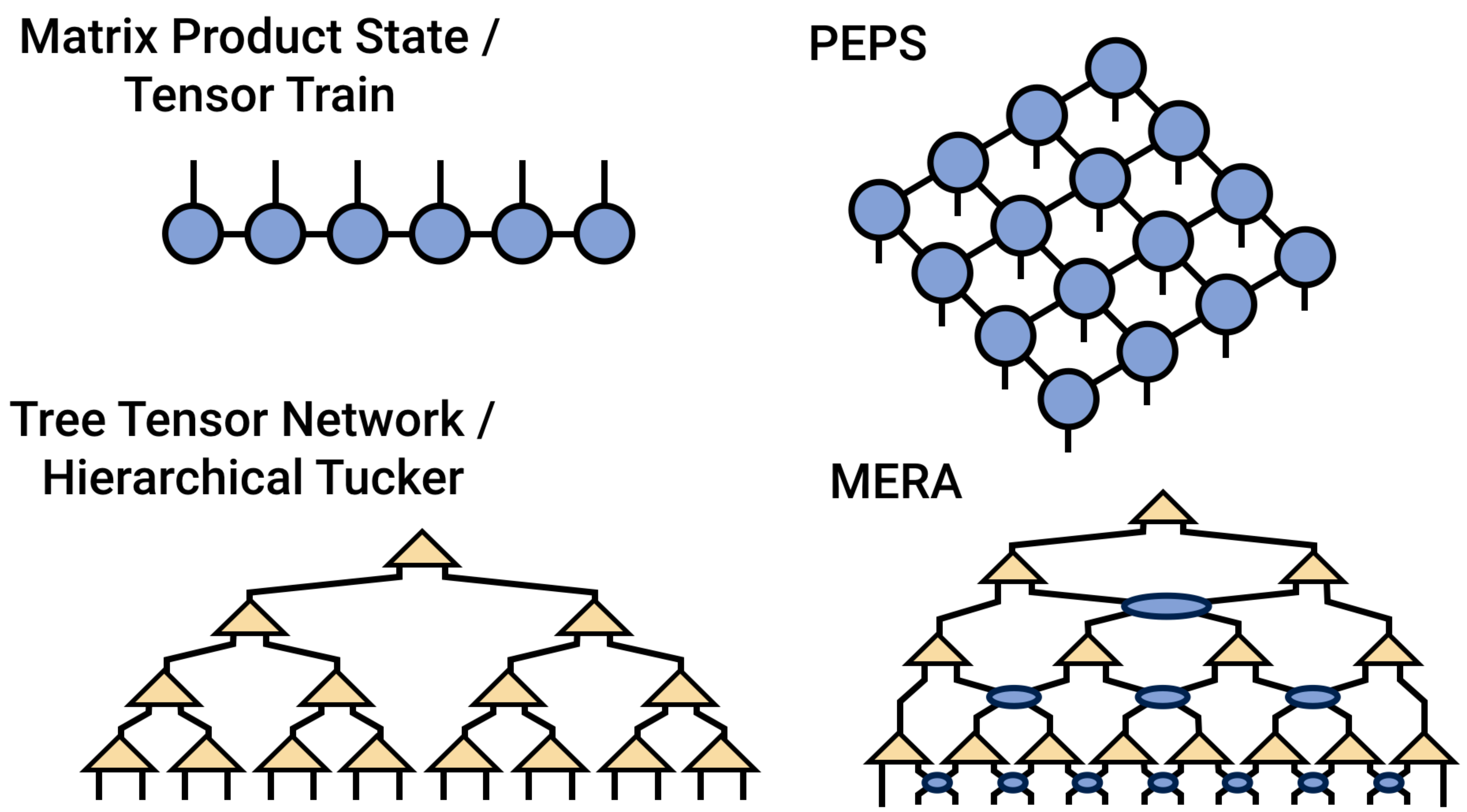
Quantum inspired generative modeling



“Teach a quantum state to write digits”

Han, Wang, Fan, LW, Zhang, 1709.01662, PRX 18’

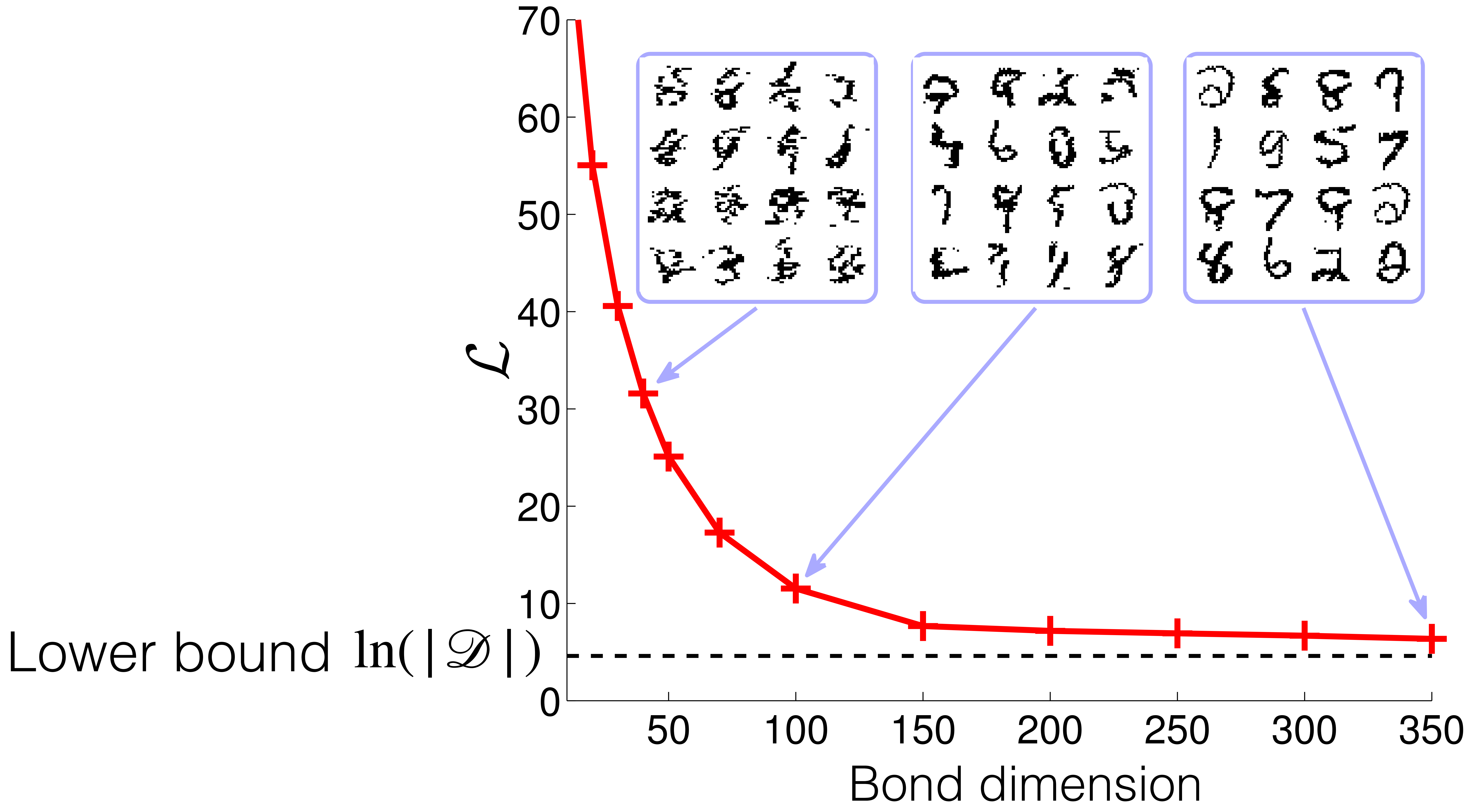
Generative modeling using Tensor Network States



Tensor Network Machine Learning

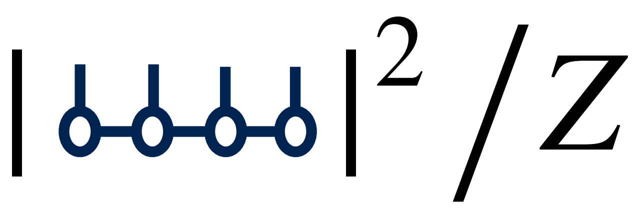
Cichocki et al 1604.05271,1609.00893,1708.09165…
Stoudenmire, Schwab NIPS 2016
Stoudenmire Q. Sci. Tech. 2018
Liu et al 1710.04833
Liu et al 1803.09111
Novikov et al 1509.06569
Hallam et al 1711.03357
Glasser et al 1806.05964
Kossaifi et al 1707.08308
Gallego, Orus 1708.01525
Pestun et al 1711.01416…

Name of the game



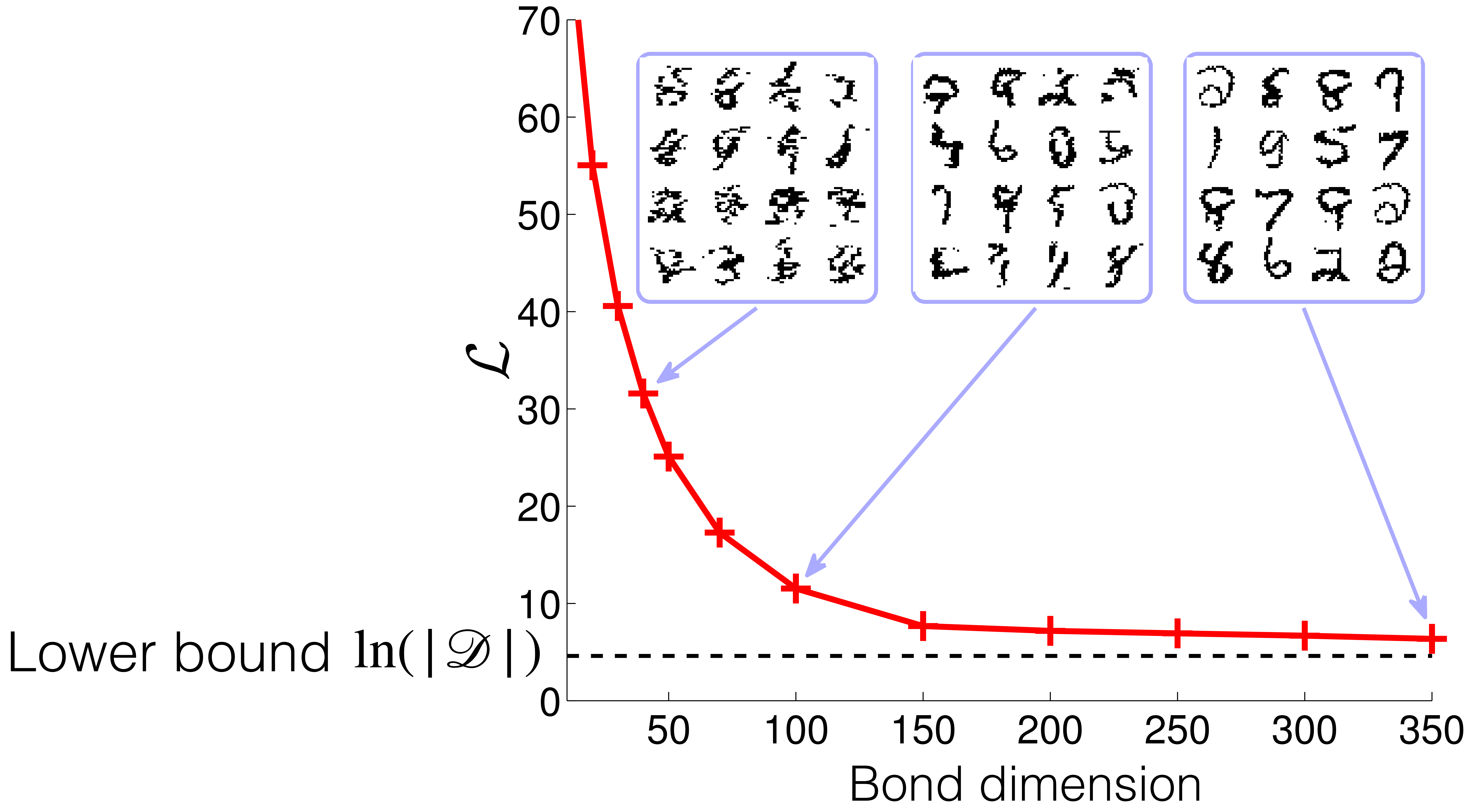
$$\mathscr{L} = -\frac{1}{|\mathscr{D}|}\sum_{x\in\mathscr{D}} \ln p(\boldsymbol{x})$$

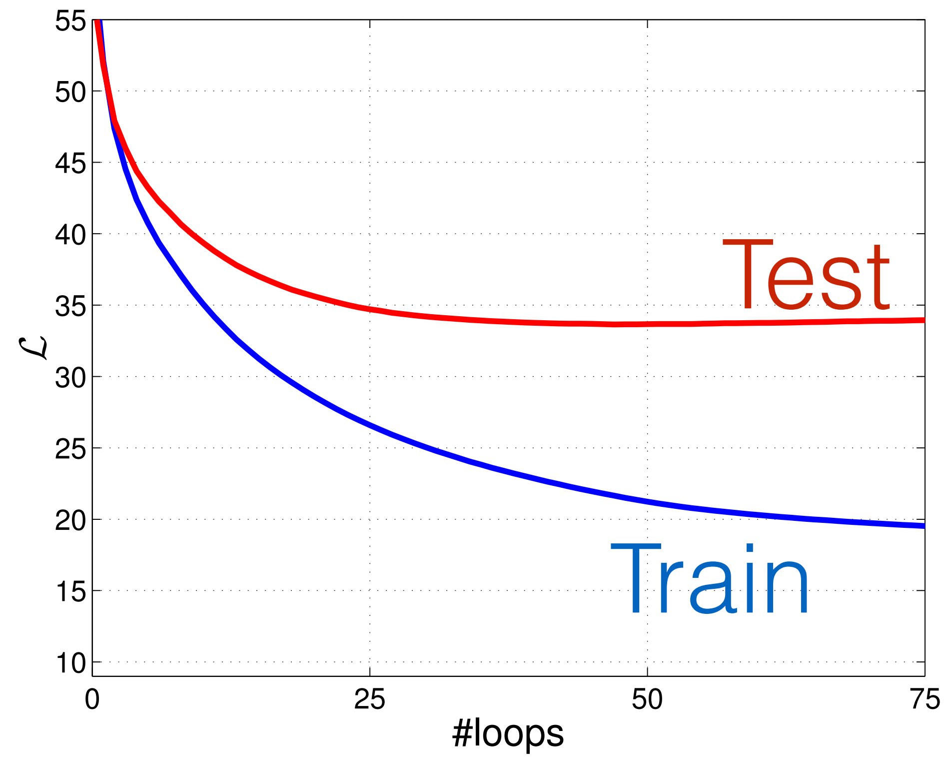
$$\downarrow$$



The ultimate goal is "learn to forget"

Name of the game





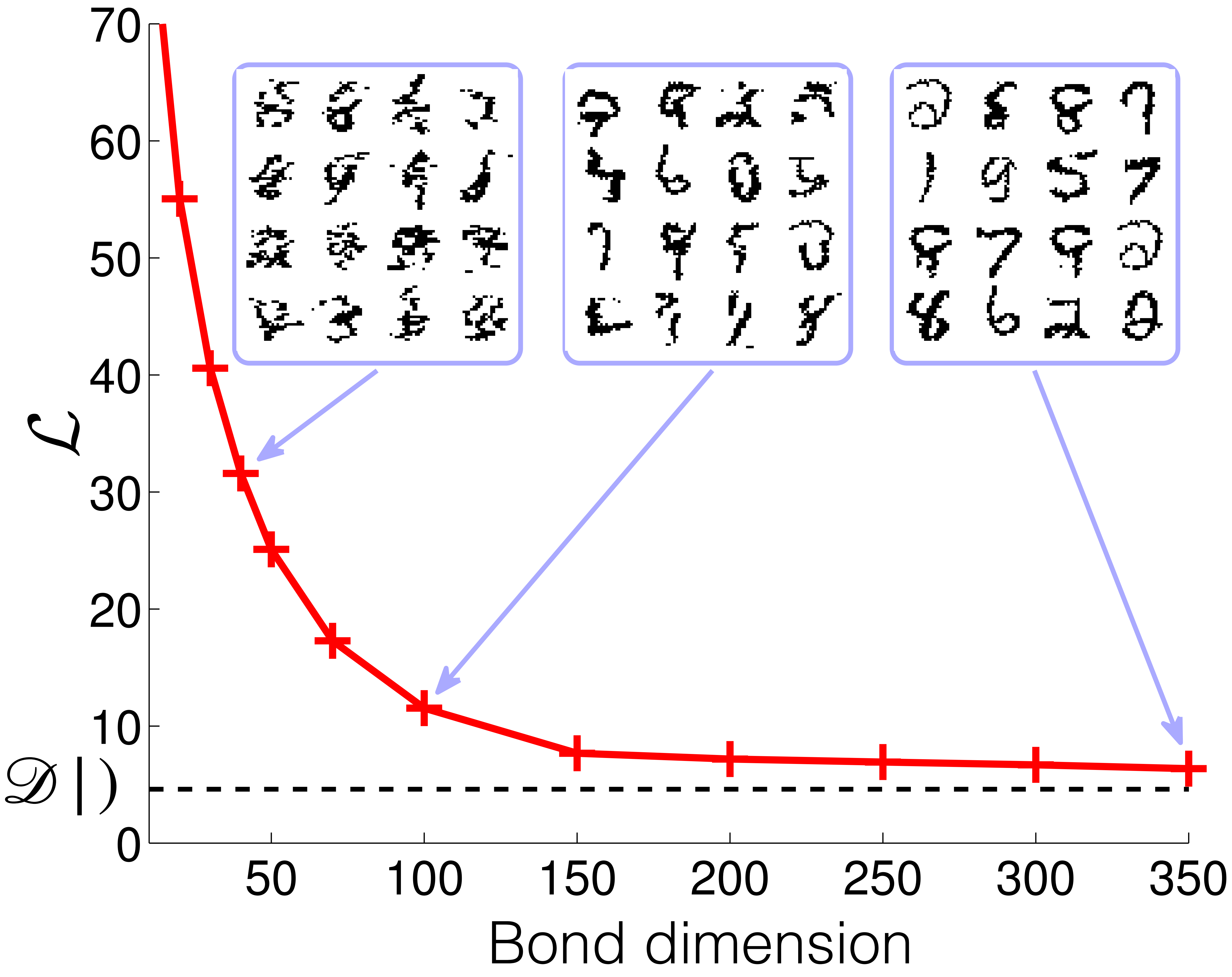
$$\mathcal{L} = -\frac{1}{|\mathcal{D}|}\sum_{x\in\mathcal{D}} \ln p(\boldsymbol{x})$$

$$\downarrow$$

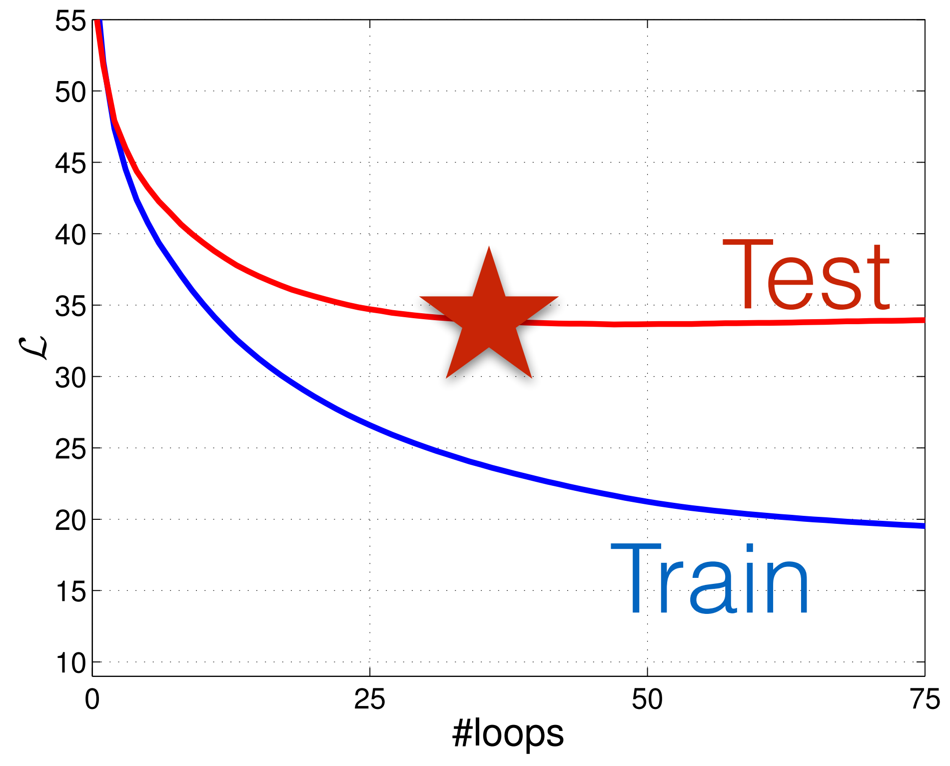
$$|\cdots|^2/Z$$

The ultimate goal is "learn to forget"

Name of the game



Lower bound $\ln(|\mathcal{D}|)$



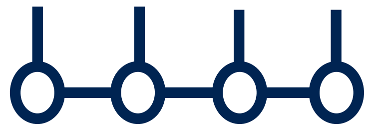
$$\mathcal{L} = -\frac{1}{|\mathcal{D}|}\sum_{x\in\mathcal{D}} \ln p(\boldsymbol{x})$$

$$\downarrow$$

$$|\cdots|^2 / Z$$

The ultimate goal is "learn to forget"

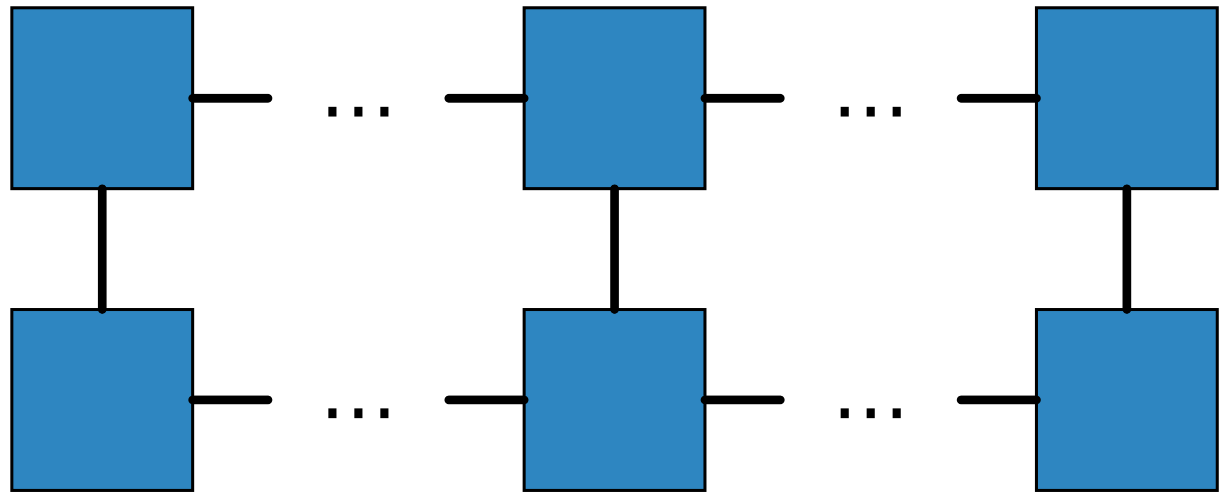
Nice features of an MPS Born Machine

$p(\boldsymbol{x}) = \Big|$  $\Big|^2 / Z$

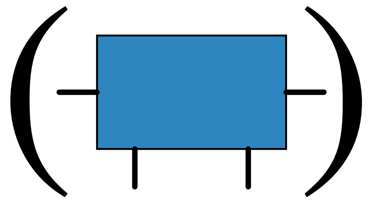
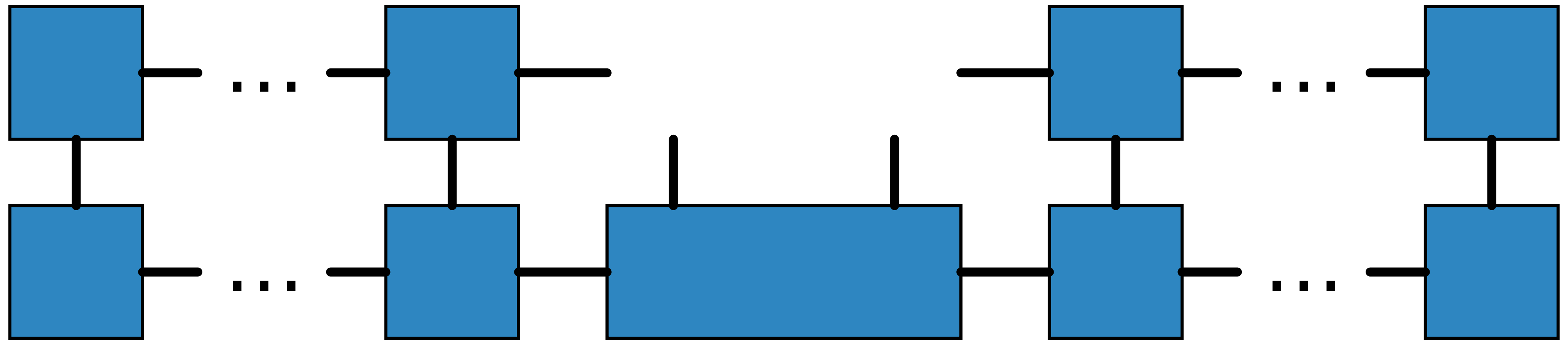
Learning **Inference** **Sampling** **Expressibility**

Glasser, Clark, Deng, Gao, Chen, Huang… 17’

Tractable Likelihood

$Z =$ 

tractable via
efficient tensor contraction

$\partial Z / \partial ($  $) = 2 \times$ 

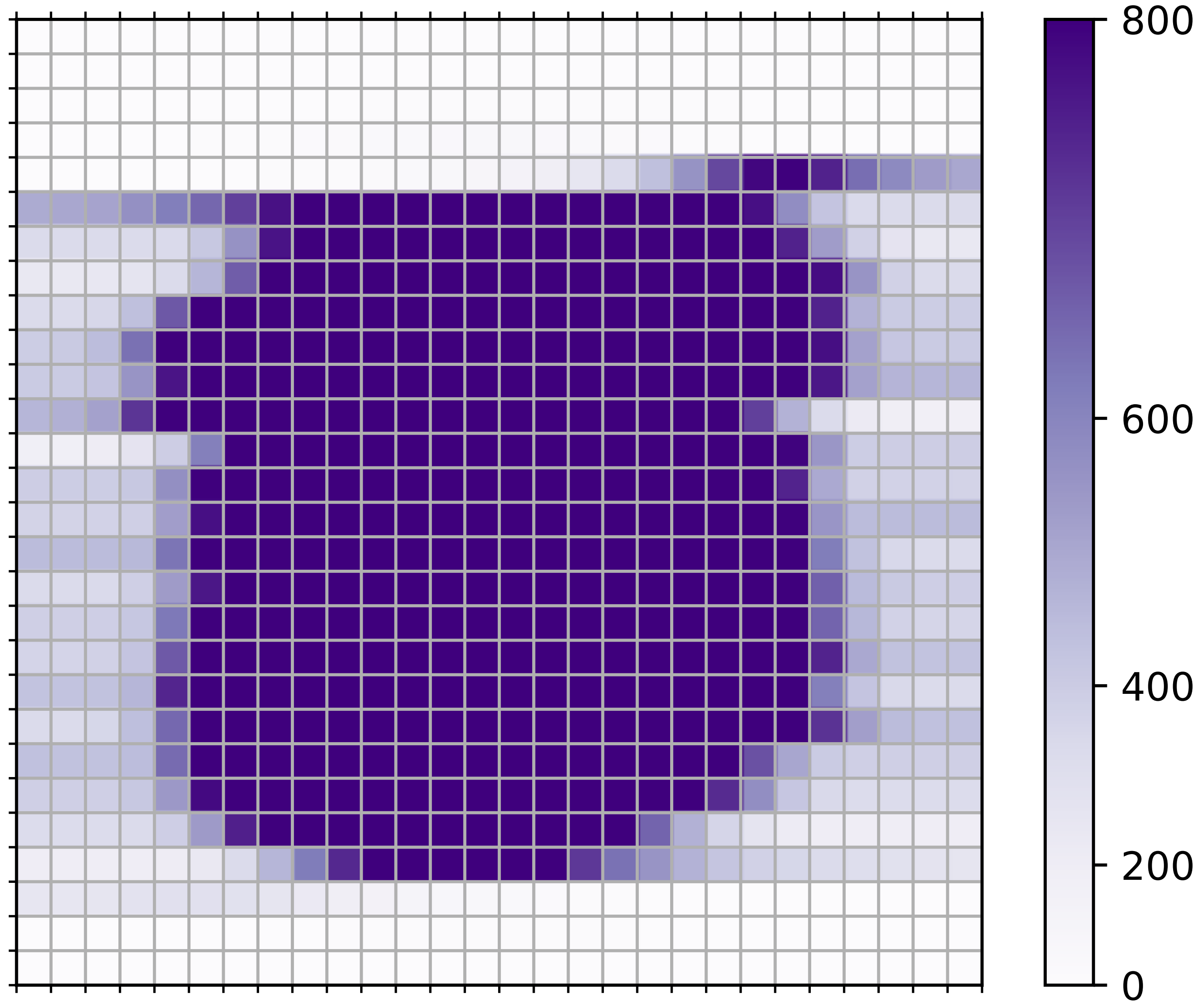
*applies to TTN
and MERA as well

Efficient & unbiased learning compared to models with intractable partition functions

Adaptive Learning

Training images

Bond dimensions



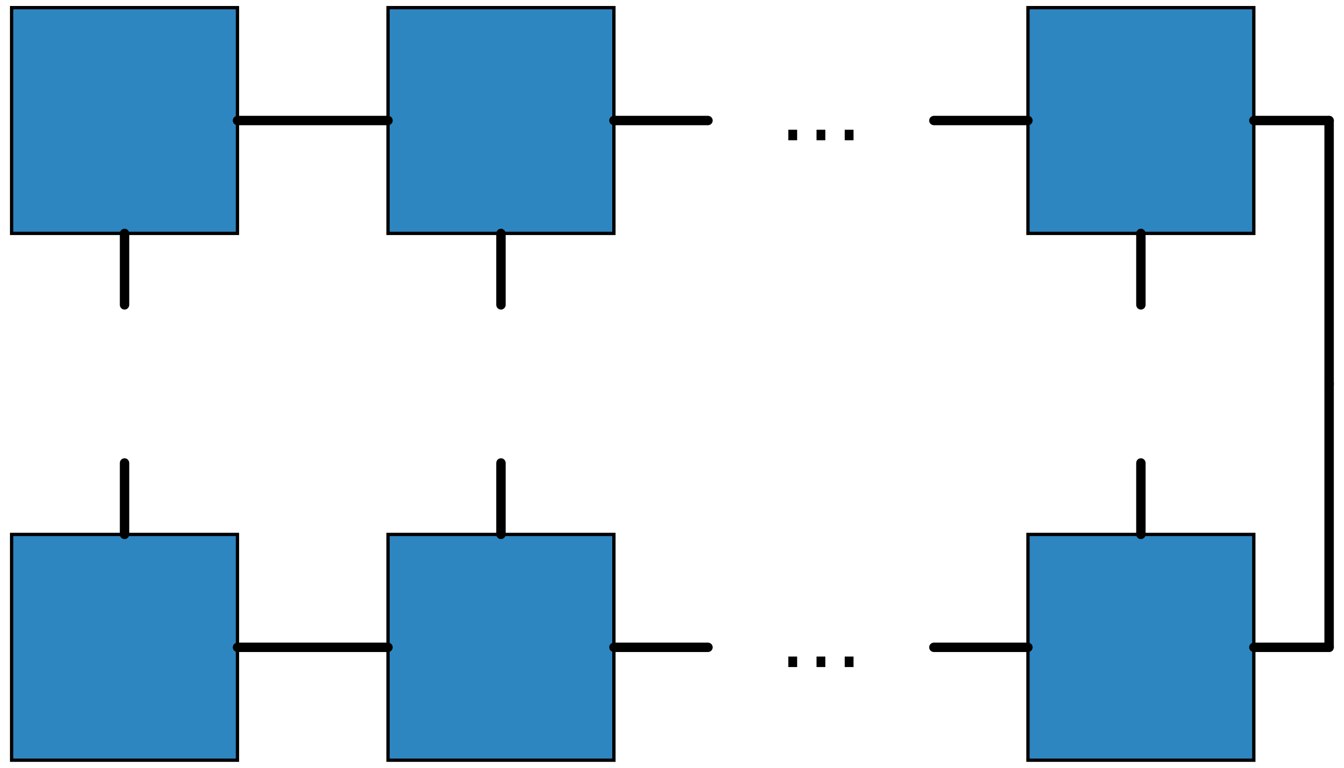
*applies to 2-site optimization

Adaptively grows the bond dimensions, thus dynamically tuning the expressibility

Direct Generation

$$p(\boldsymbol{x}) = \prod_i p(x_i | \boldsymbol{x}_{<i}) = \prod_i \frac{p(\boldsymbol{x}_{<i+1})}{p(\boldsymbol{x}_{<i})}$$

Ferris & Vidal 2012

$p(\boldsymbol{x}_{<i}) =$ 

*applies to TTN and MERA as well

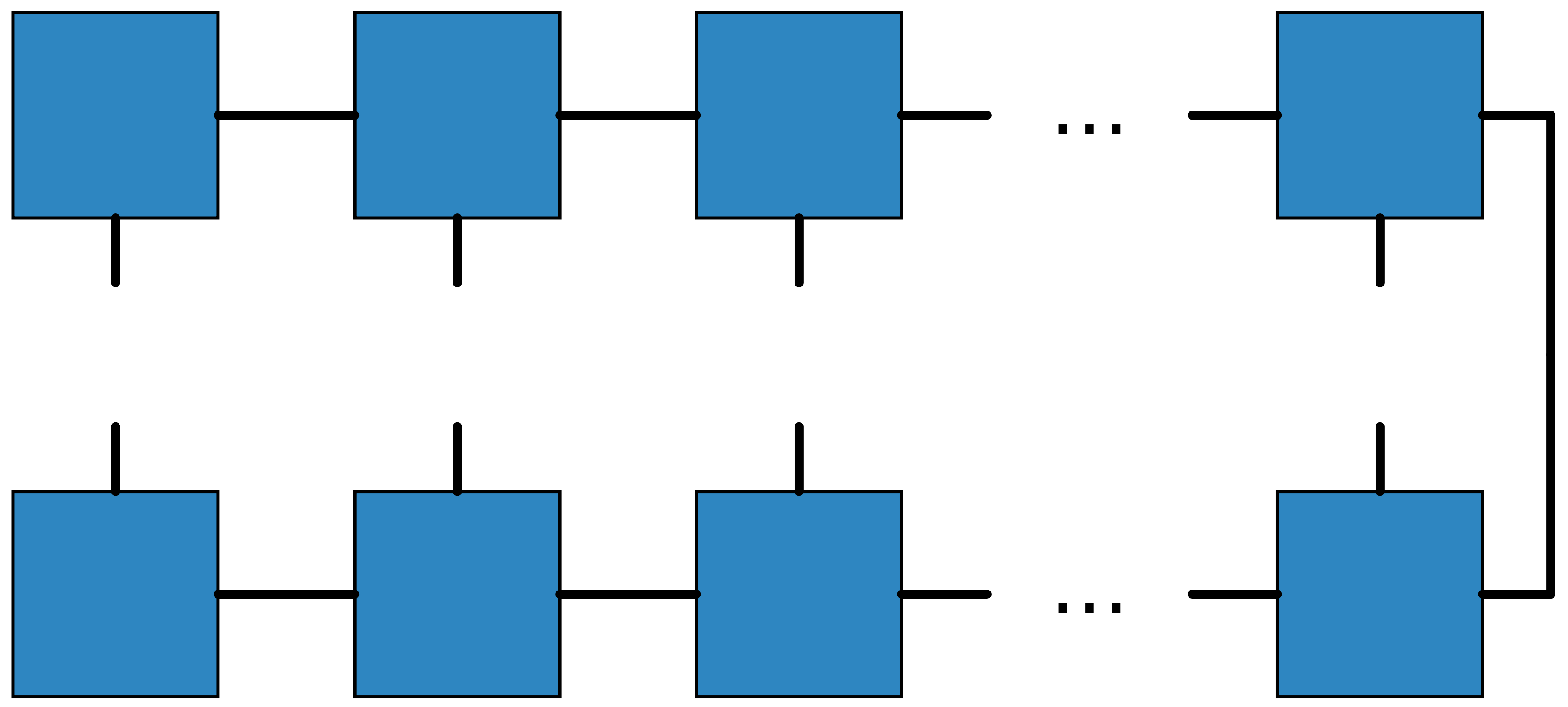
No thermalization issue compared to slow mixing Gibbs sampling of Boltzmann Machines

Direct Generation

$$p(\boldsymbol{x}) = \prod_i p(x_i | \boldsymbol{x}_{<i}) = \prod_i \frac{p(\boldsymbol{x}_{<i+1})}{p(\boldsymbol{x}_{<i})}$$

Ferris & Vidal 2012

$$p(\boldsymbol{x}_{<i+1}) =$$



*applies to TTN and MERA as well

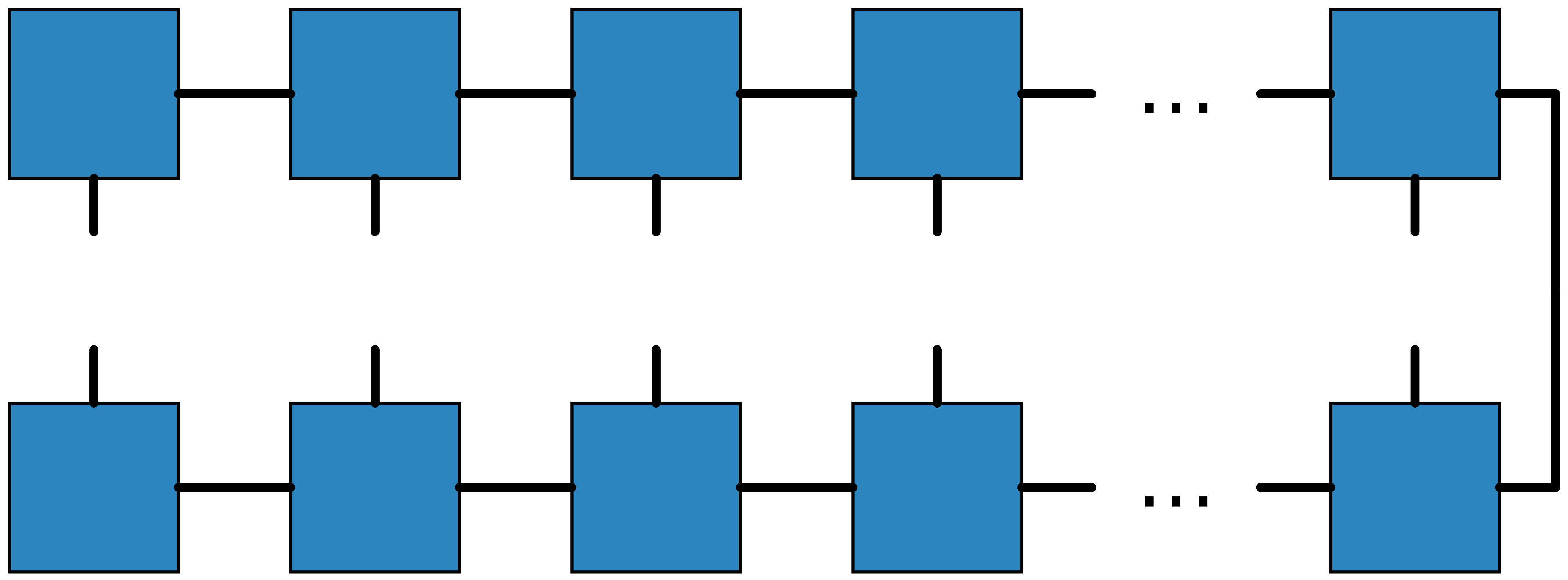
No thermalization issue compared to slow mixing Gibbs sampling of Boltzmann Machines

Direct Generation

$$p(\boldsymbol{x}) = \prod_i p(x_i | \boldsymbol{x}_{<i}) = \prod_i \frac{p(\boldsymbol{x}_{<i+1})}{p(\boldsymbol{x}_{<i})}$$

Ferris & Vidal 2012

$p(\boldsymbol{x}_{<i+2}) =$



*applies to TTN and MERA as well

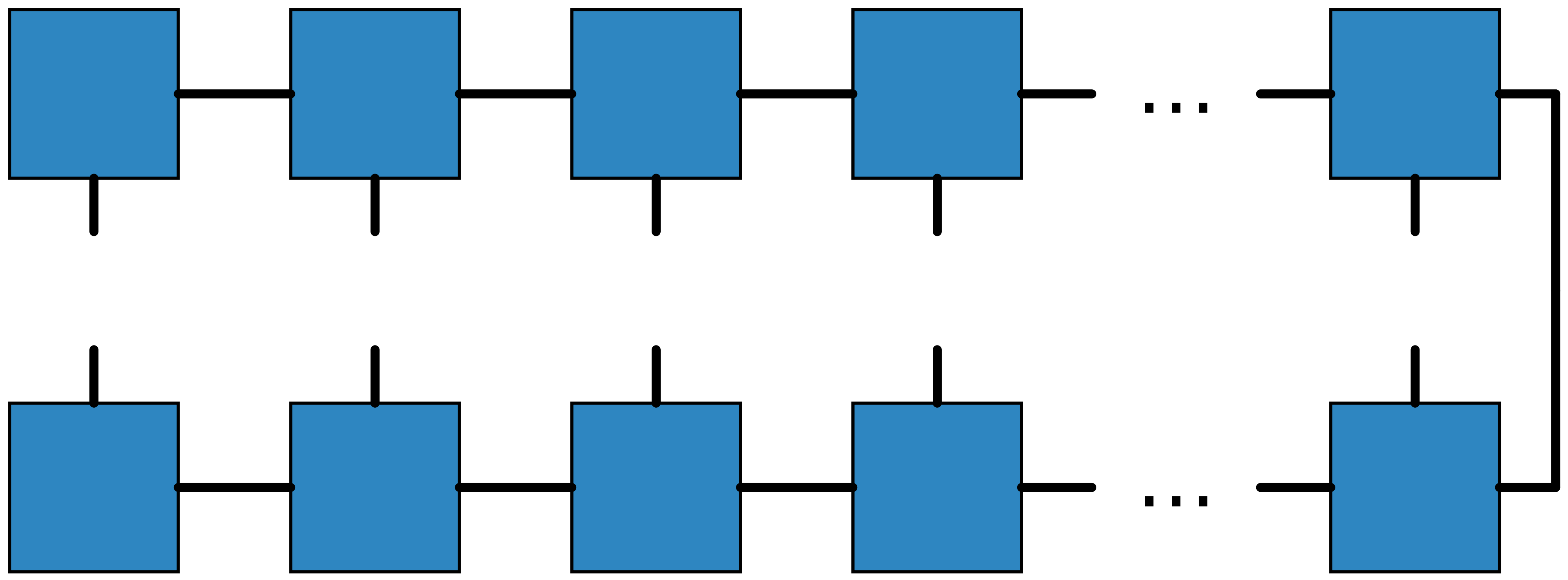
No thermalization issue compared to slow mixing Gibbs sampling of Boltzmann Machines

Direct Generation

$$p(\boldsymbol{x}) = \prod_i p(x_i | \boldsymbol{x}_{<i}) = \prod_i \frac{p(\boldsymbol{x}_{<i+1})}{p(\boldsymbol{x}_{<i})}$$

Ferris & Vidal 2012

$$p(\boldsymbol{x}_{<i+2}) =$$



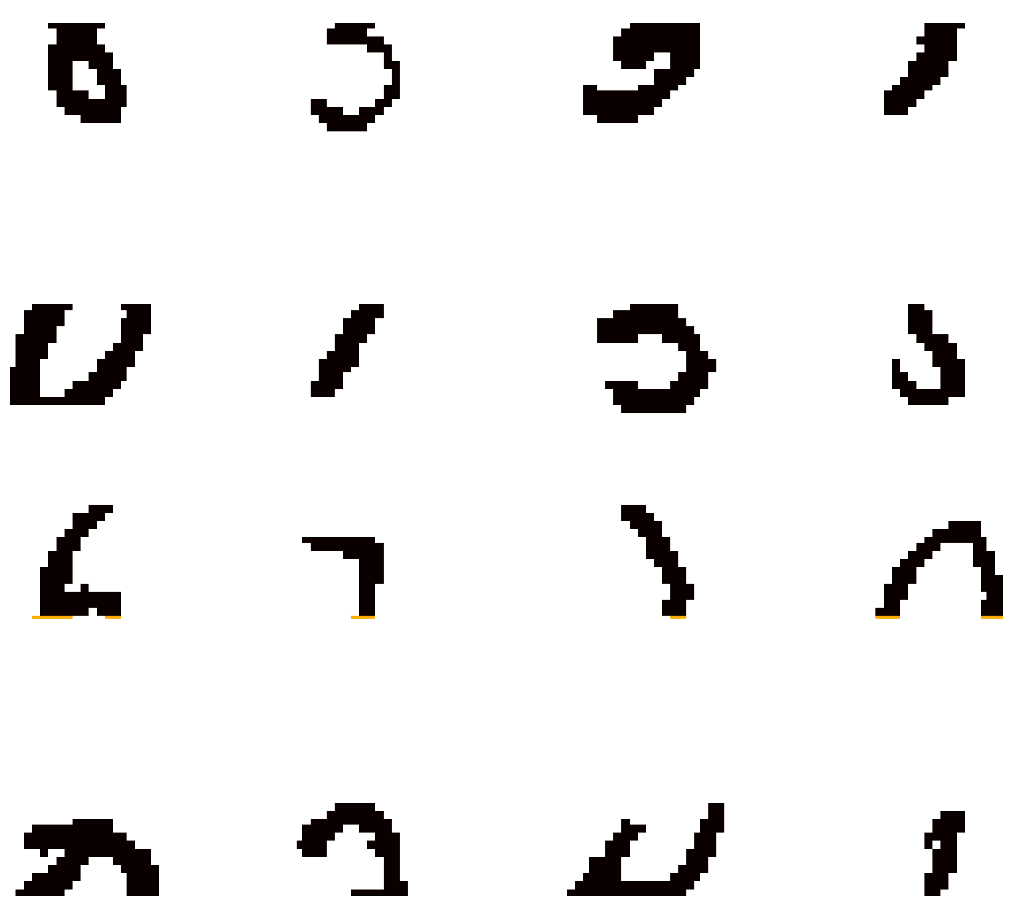
"Zipper Sampling"

*applies to TTN and MERA as well

No thermalization issue compared to slow mixing Gibbs sampling of Boltzmann Machines

Image Restoration

Han, Wang, Fan, LW, Zhang, 1709.01662, PRX 18’



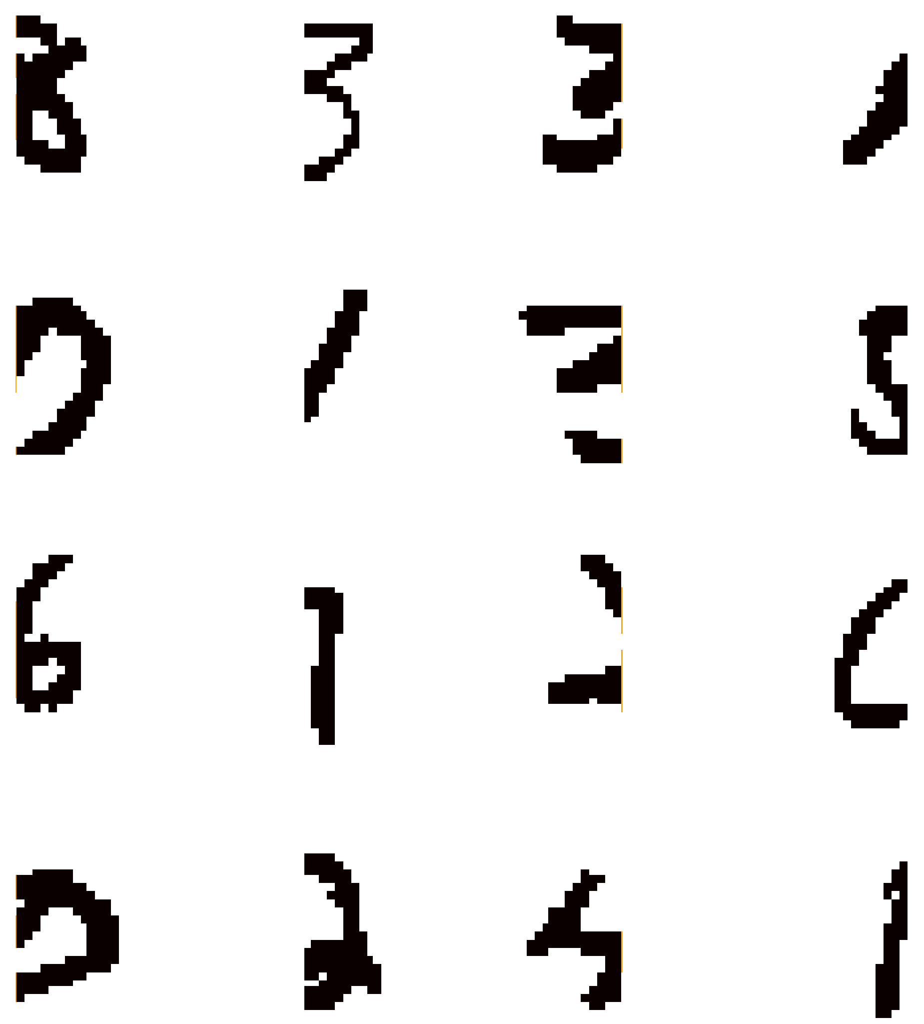
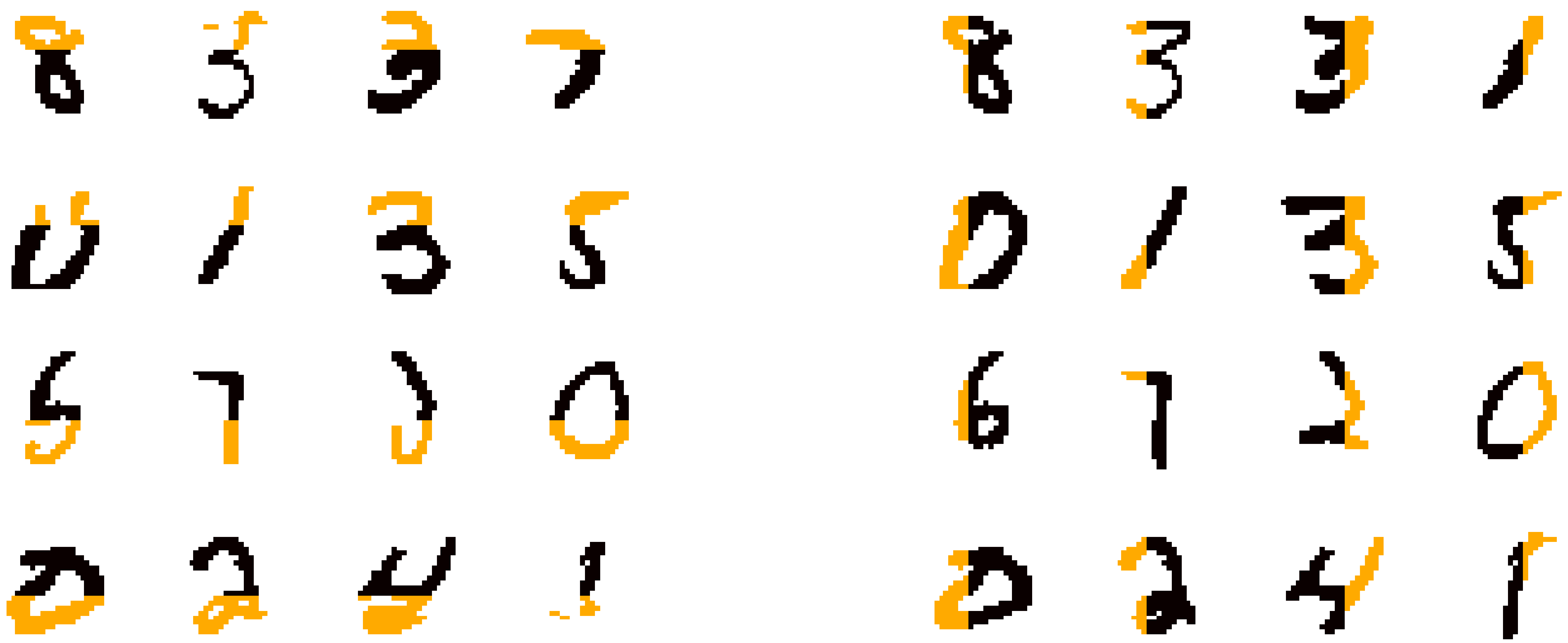


Image Restoration

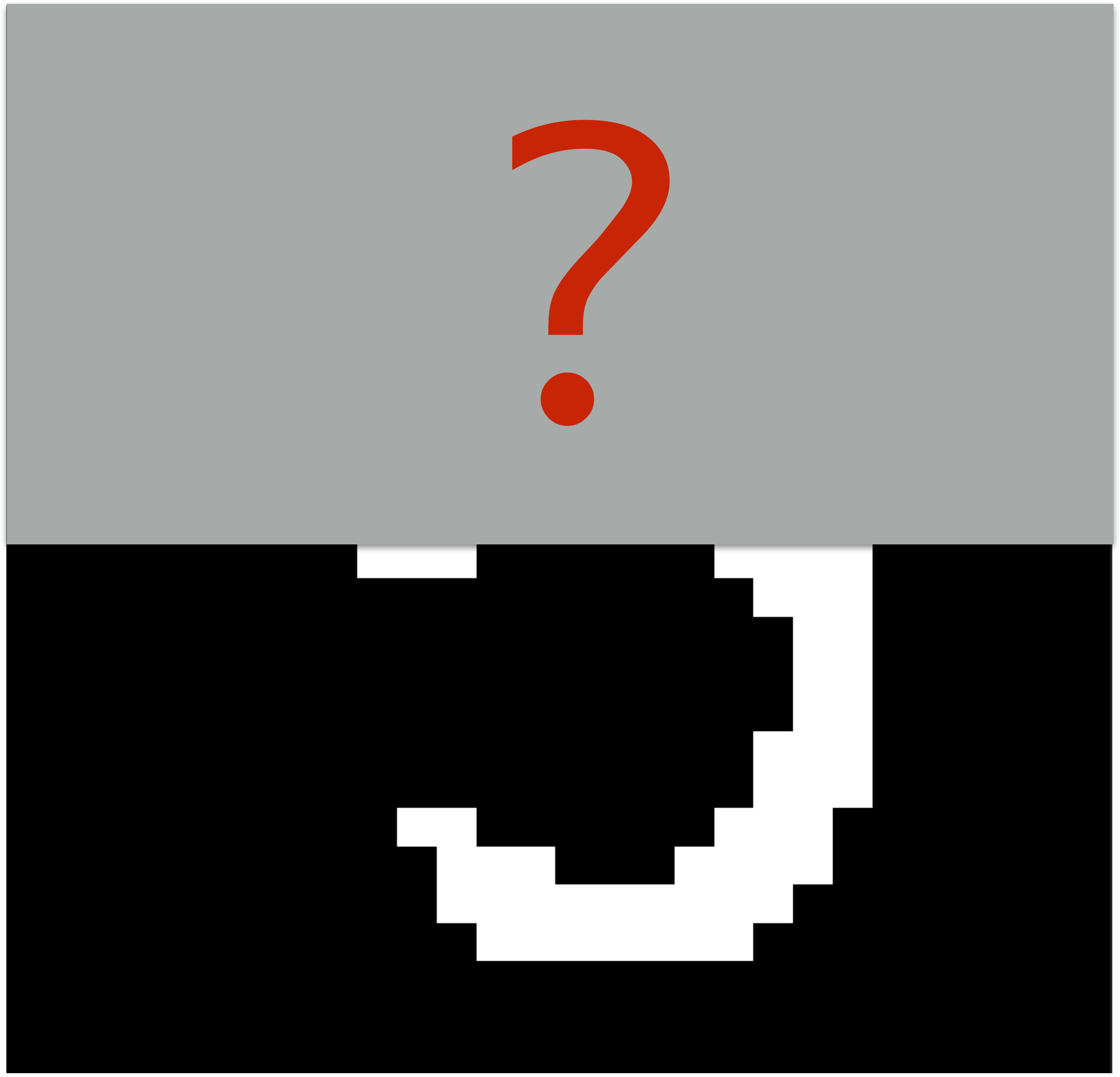
Han, Wang, Fan, LW, Zhang, 1709.01662, PRX 18'



Arbitrary order, in contrast to autoregressive models

PixelCNN

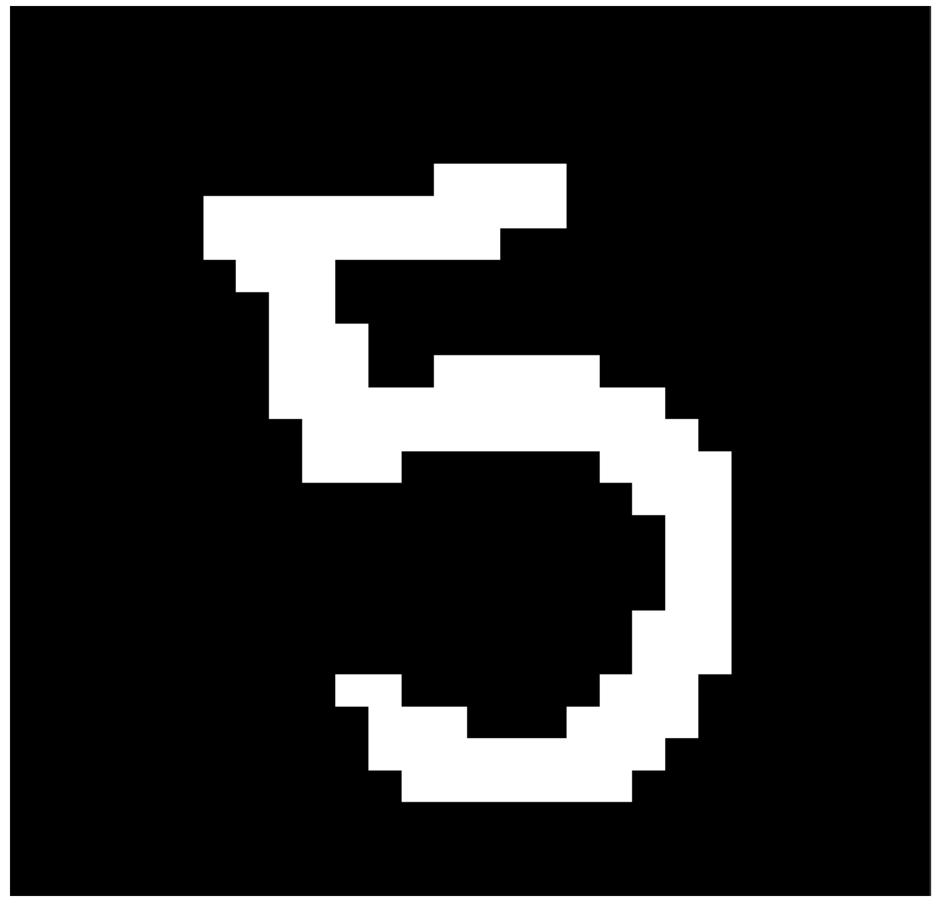
Quantum Perspective on Deep Learning

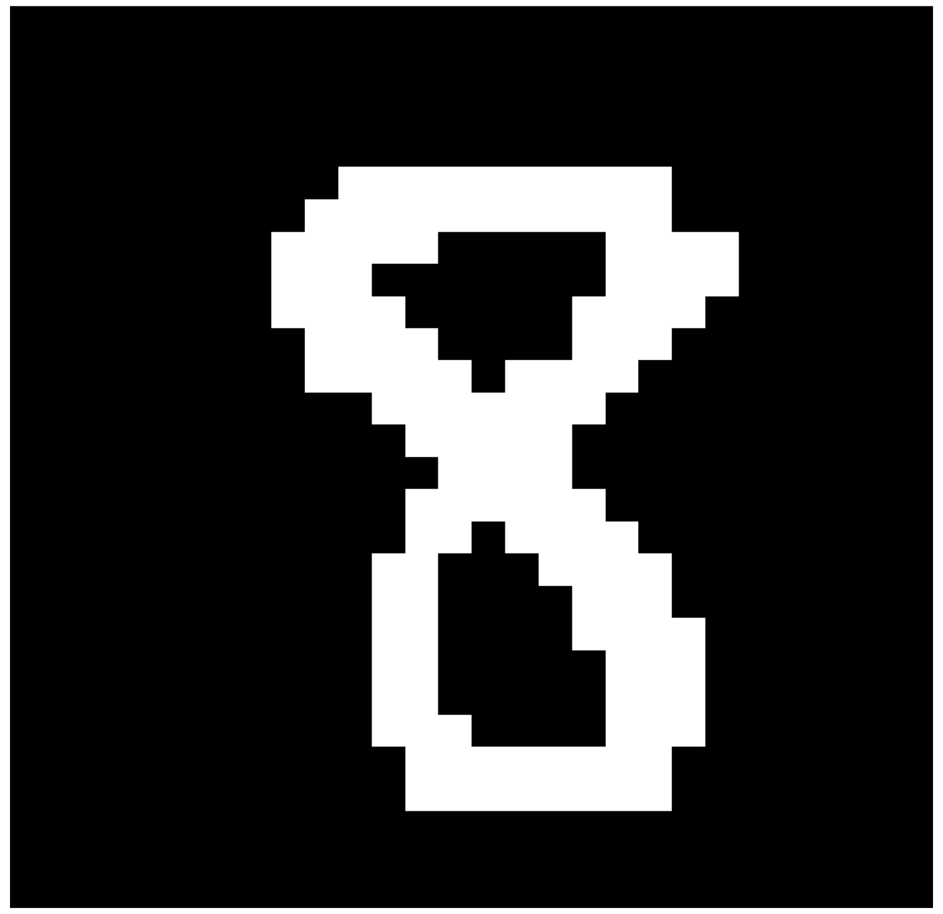


Quantum Perspective on Deep Learning

Q: How to quantify our inductive biases ?

A: Information pattern of probability distributions





Quantum Perspective on Deep Learning

Q: How to quantify our inductive biases ?

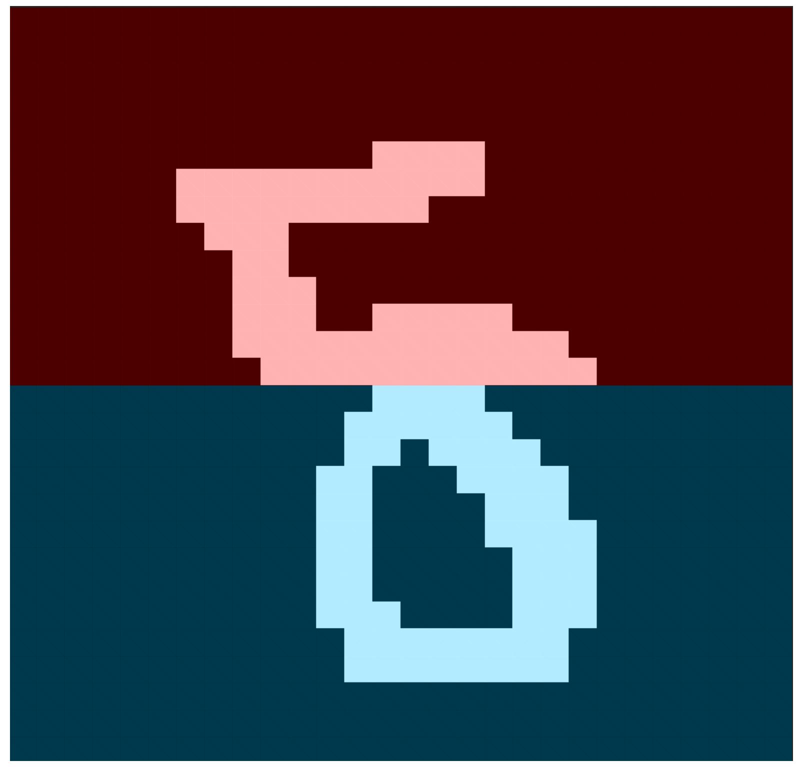
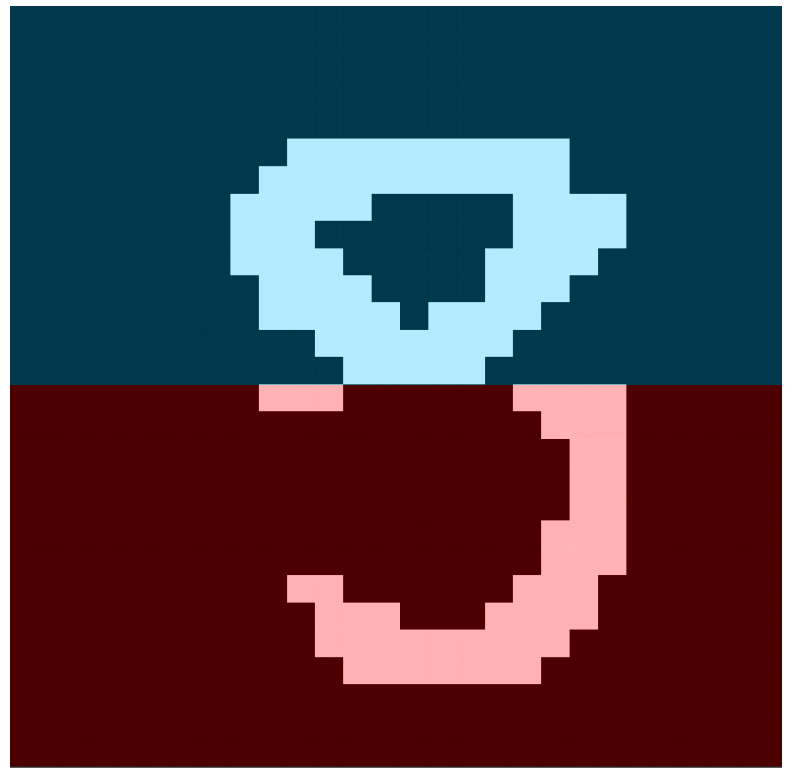
A: Information pattern of probability distributions

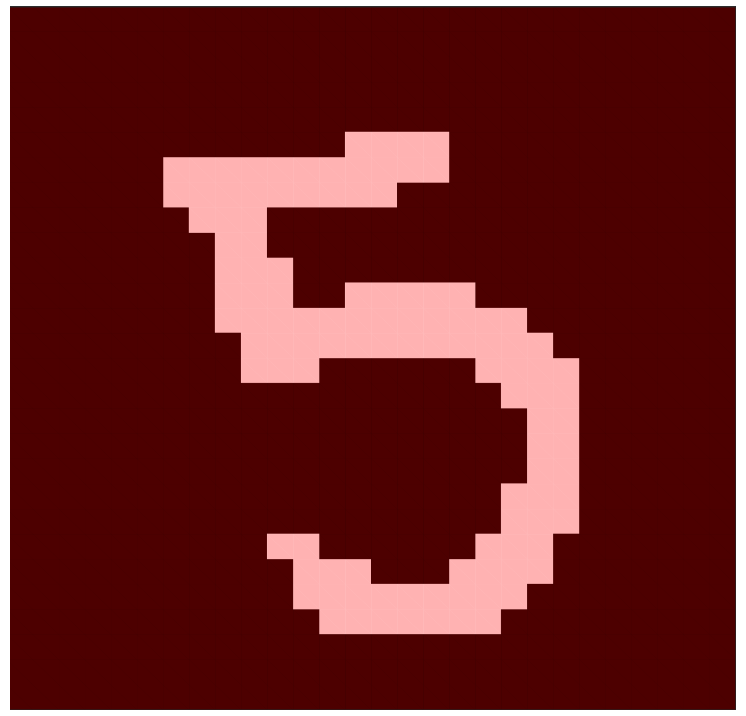
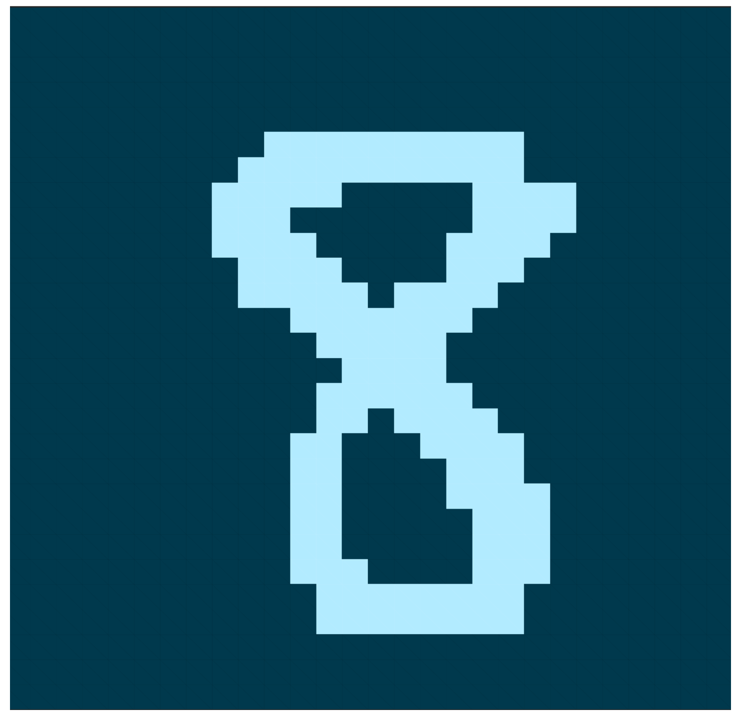
Quantum Perspective on Deep Learning

Q: How to quantify our inductive biases ?

A: Information pattern of probability distributions

Quantum Perspective on Deep Learning

$p\Big($  $\Big) \times p\Big($  $\Big)$

$p\Big($  $\Big) \times p\Big($  $\Big)$

Quantum Perspective on Deep Learning

Classical mutual information

$$I = -\left\langle \ln \left\langle \frac{p(\boldsymbol{x},\boldsymbol{y}')p(\boldsymbol{x}',\boldsymbol{y})}{p(\boldsymbol{x}',\boldsymbol{y}')p(\boldsymbol{x},\boldsymbol{y})} \right\rangle_{\boldsymbol{x}',\boldsymbol{y}'} \right\rangle_{\boldsymbol{x},\boldsymbol{y}}$$

Quantum Renyi entanglement entropy

$$S = -\ln \left\langle \left\langle \frac{\Psi(\boldsymbol{x},\boldsymbol{y}')\Psi(\boldsymbol{x}',\boldsymbol{y})}{\Psi(\boldsymbol{x}',\boldsymbol{y}')\Psi(\boldsymbol{x},\boldsymbol{y})} \right\rangle_{\boldsymbol{x}',\boldsymbol{y}'} \right\rangle_{\boldsymbol{x},\boldsymbol{y}}$$

Striking similarity implies common inductive bias

+Quantitative & interpretable approaches

+Principled structure design & learning

Cheng, Chen, LW,
1712.04144, Entropy 18’

DEEP LEARNING AND QUANTUM ENTANGLEMENT: FUNDAMENTAL CONNECTIONS WITH IMPLICATIONS TO NETWORK DESIGN

Yoav Levine, David Yakira, Nadav Cohen & Amnon Shashua
The Hebrew University of Jerusalem
{yoavlevine,davidyakira,cohennadav,shashua}@cs.huji.ac.il

ABSTRACT

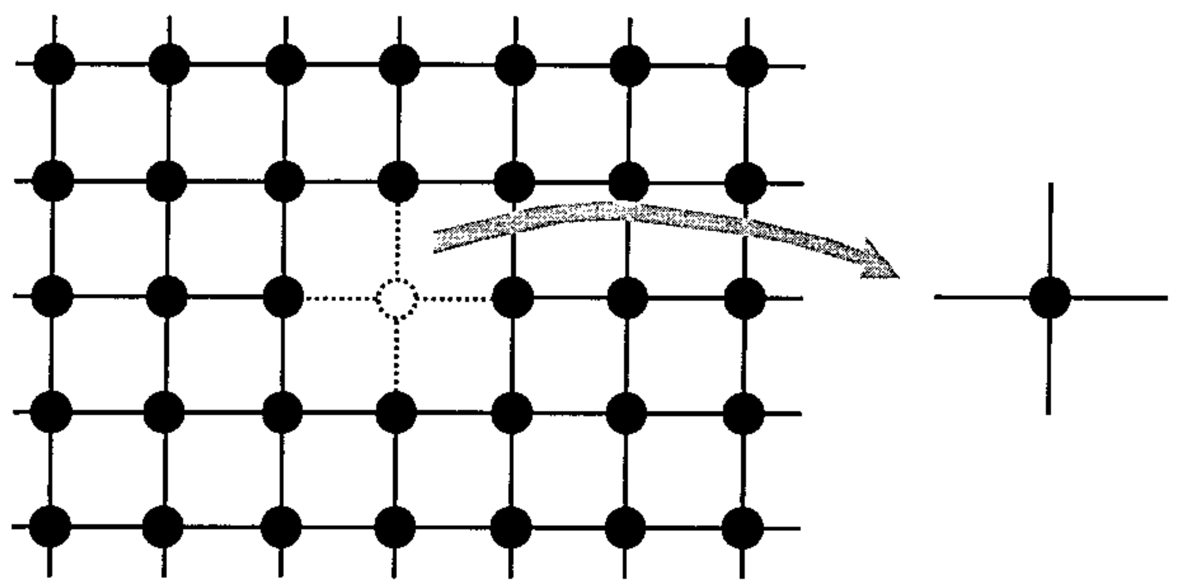
Formal understanding of the inductive bias behind deep convolutional networks, i.e. the relation between the network's architectural features and the functions it is able to model, is limited. In this work, we establish a fundamental connection between the fields of quantum physics and deep learning, and use it for obtaining novel theoretical observations regarding the inductive bias of convolutional networks. Specifically, we show a structural equivalence between the function realized by a convolutional arithmetic circuit (ConvAC) and a quantum many-body wave function, which facilitates the use of quantum entanglement measures as quantifiers of a deep network's expressive ability to model correlations. Furthermore, the construction of a deep ConvAC in terms of a quantum Tensor Network is enabled. This allows us to perform a graph-theoretic analysis of a convolutional network, tying its expressiveness to a min-cut in its underlying graph. We demonstrate a practical outcome in the form of a direct control over the inductive bias via the number of channels (width) of each layer. We empirically validate our findings on standard convolutional networks which involve ReLU activations and max pooling. The description of a deep convolutional network in well-defined graph-theoretic tools and the structural connection to quantum entanglement, are two interdisciplinary bridges that are brought forth by this work.

DEEP LEARNING AND QUANTUM ENTANGLEMENT: FUNDAMENTAL CONNECTIONS WITH IMPLICATIONS TO NETWORK DESIGN

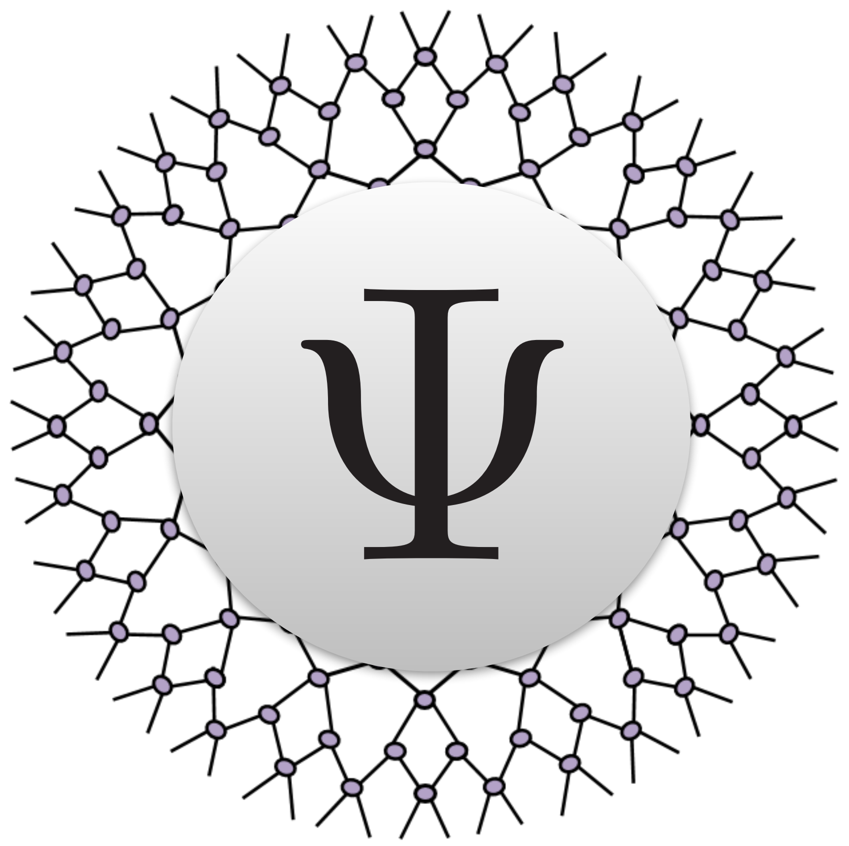
Yoav Levine, David Yakira, Nadav Cohen & Amnon Shashua
The Hebrew University of Jerusalem

Physicists' gifts to Machine Learning

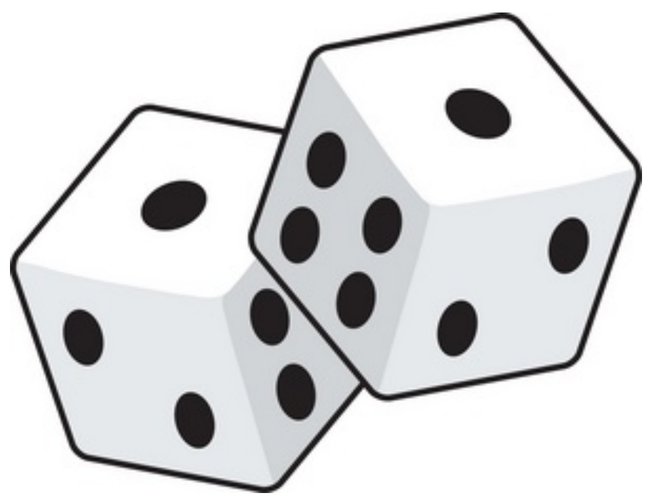
Mean Field Theory



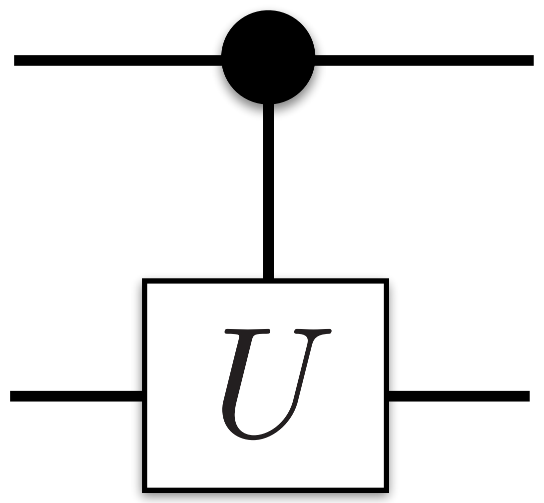
Tensor Networks



Monte Carlo Methods

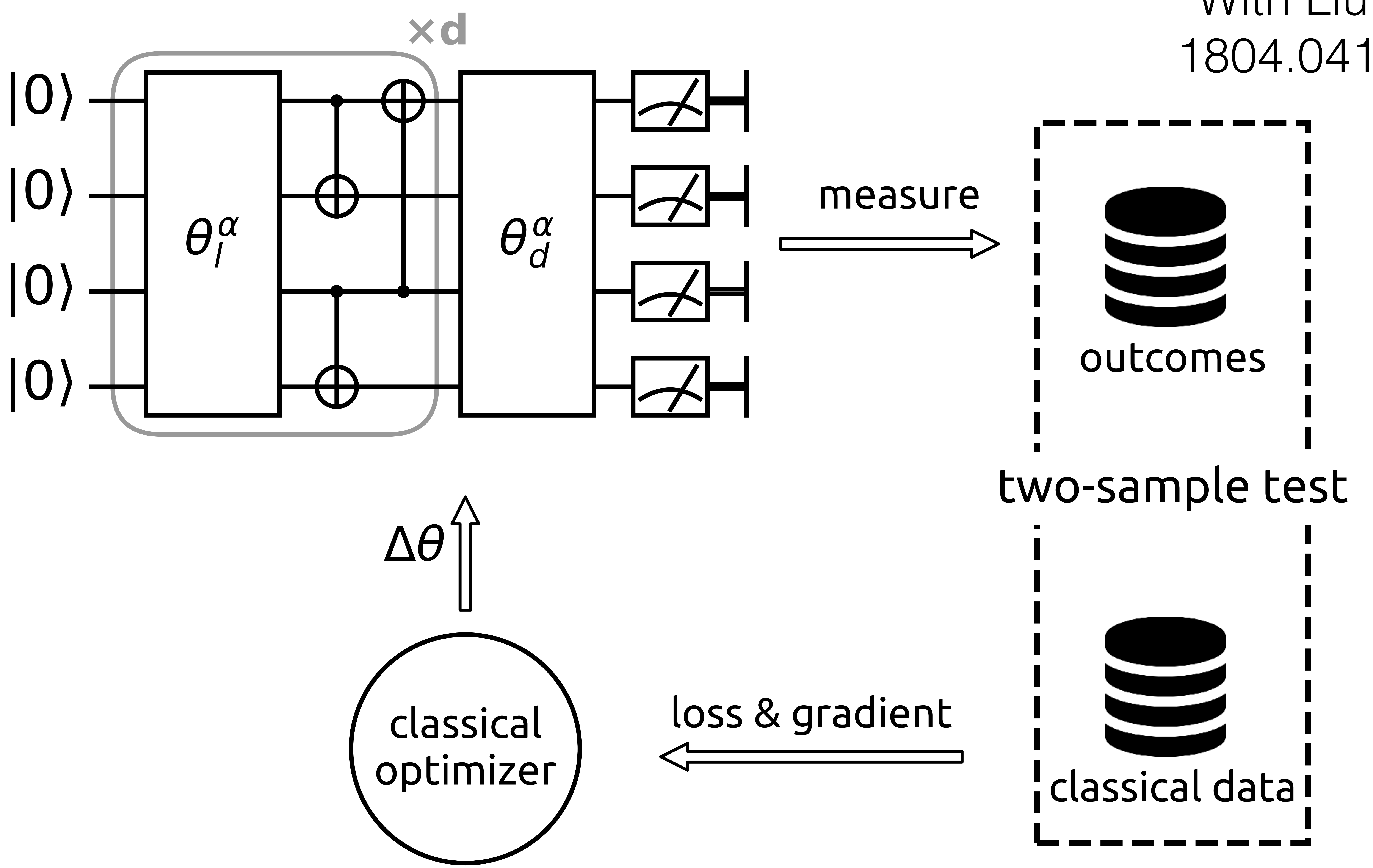


Quantum Computing



Quantum Circuit Born Machine

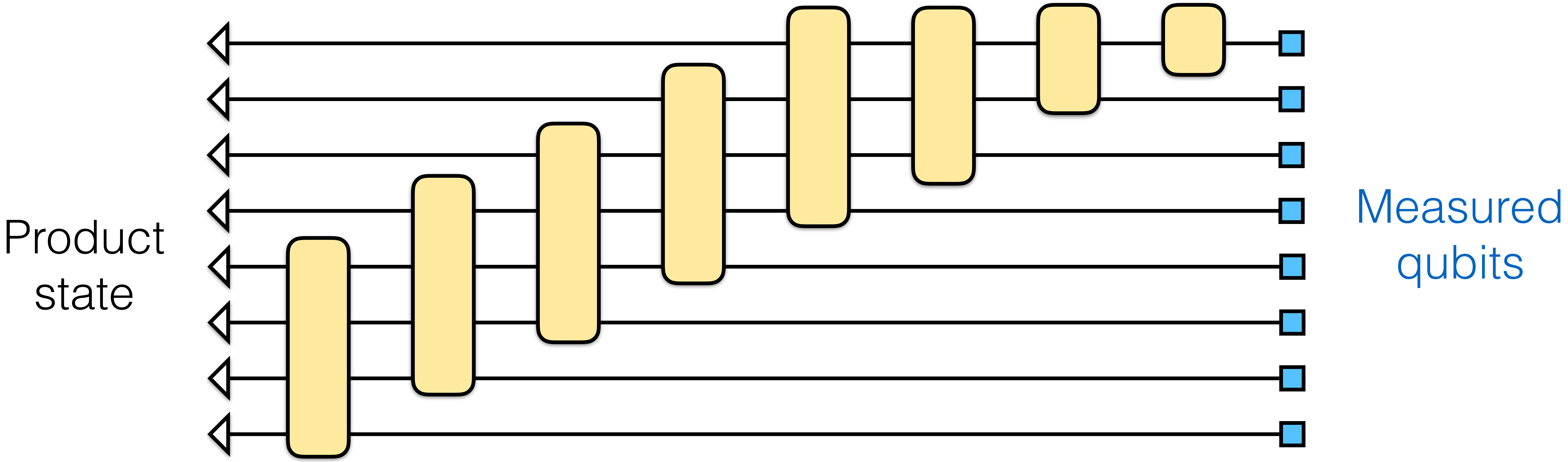
With Liu, Zeng, Wu, Hu
1804.04168, 1808.03425



Train the quantum circuit as a probabilistic generative model

Quantum sampling complexity underlines the "quantum supremacy"

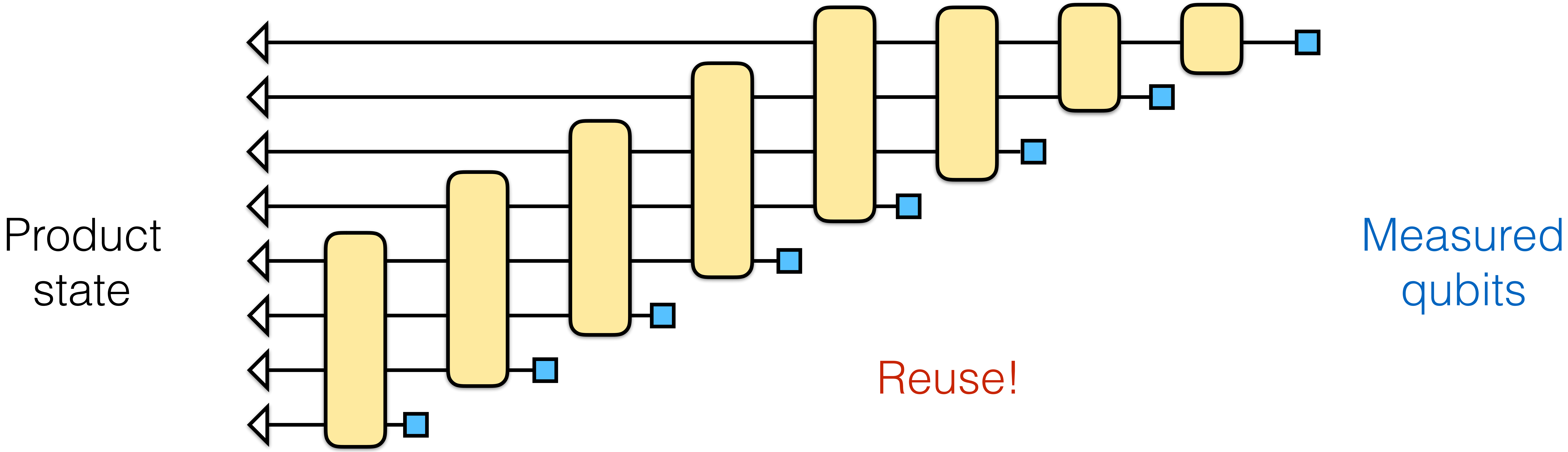
Qubit efficient scheme



Huggins, Patel, Whaley, Stoudenmire, 1803.11537
see also Cramer et al, Nat. Comm. 2010

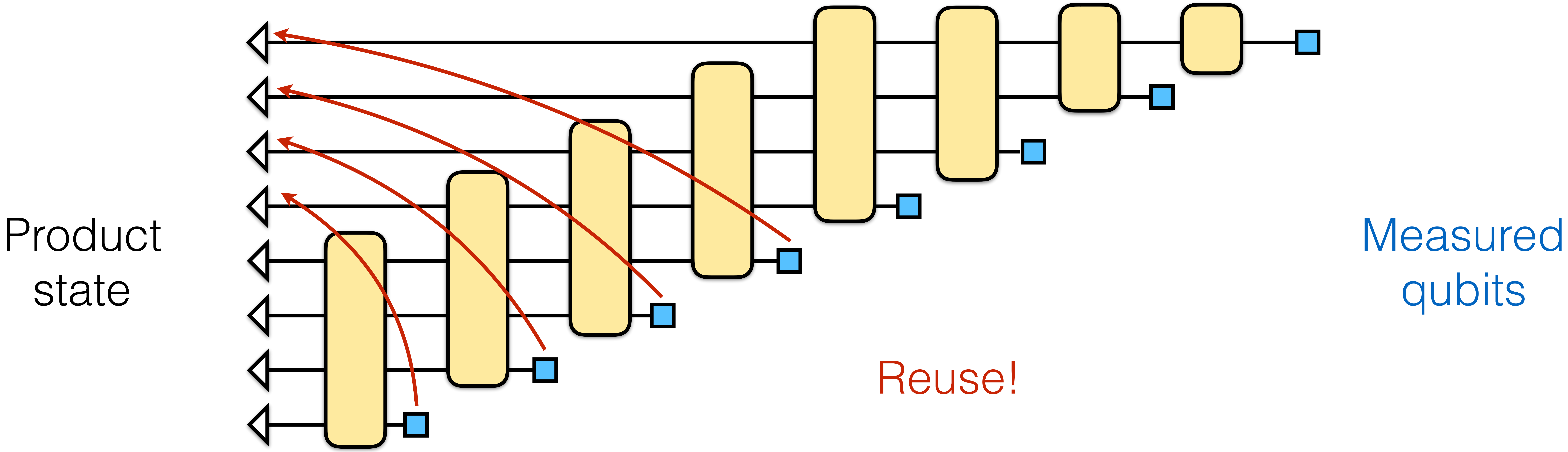
Tensor network inspired quantum circuit architecture

Qubit efficient scheme



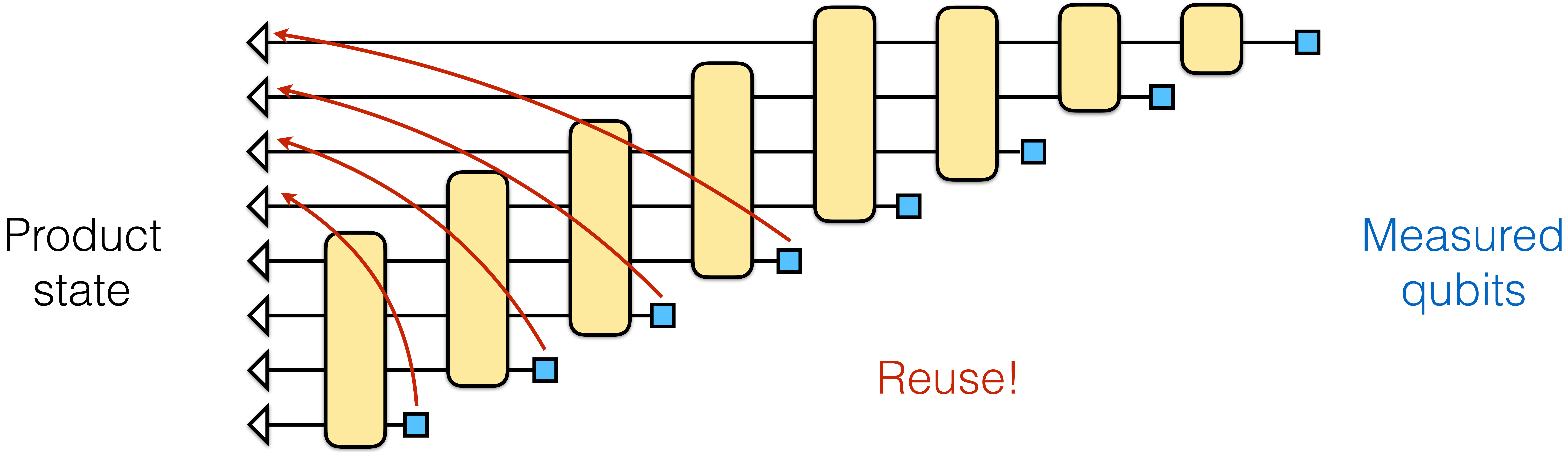
Huggins, Patel, Whaley, Stoudenmire, 1803.11537
see also Cramer et al, Nat. Comm. 2010

Qubit efficient scheme

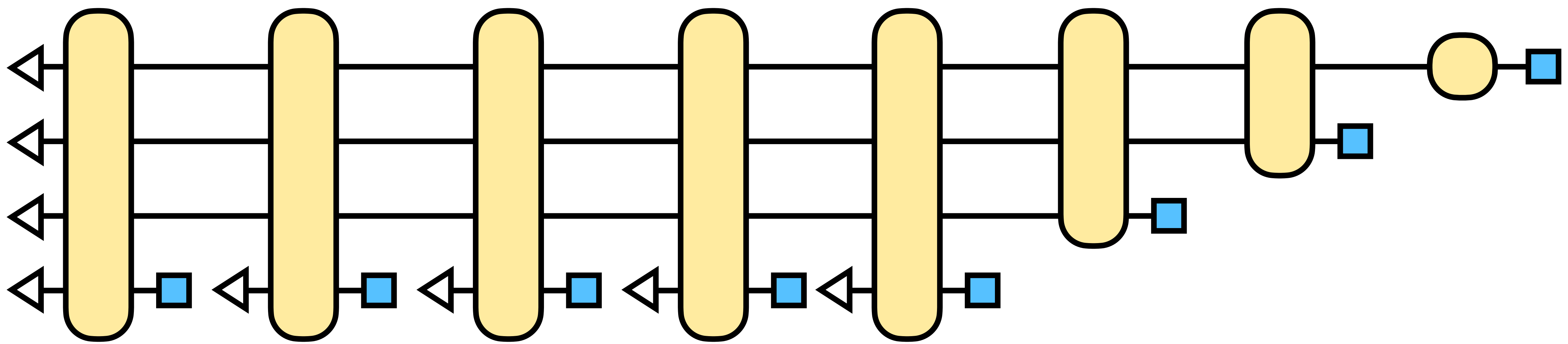


Huggins, Patel, Whaley, Stoudenmire, 1803.11537
see also Cramer et al, Nat. Comm. 2010

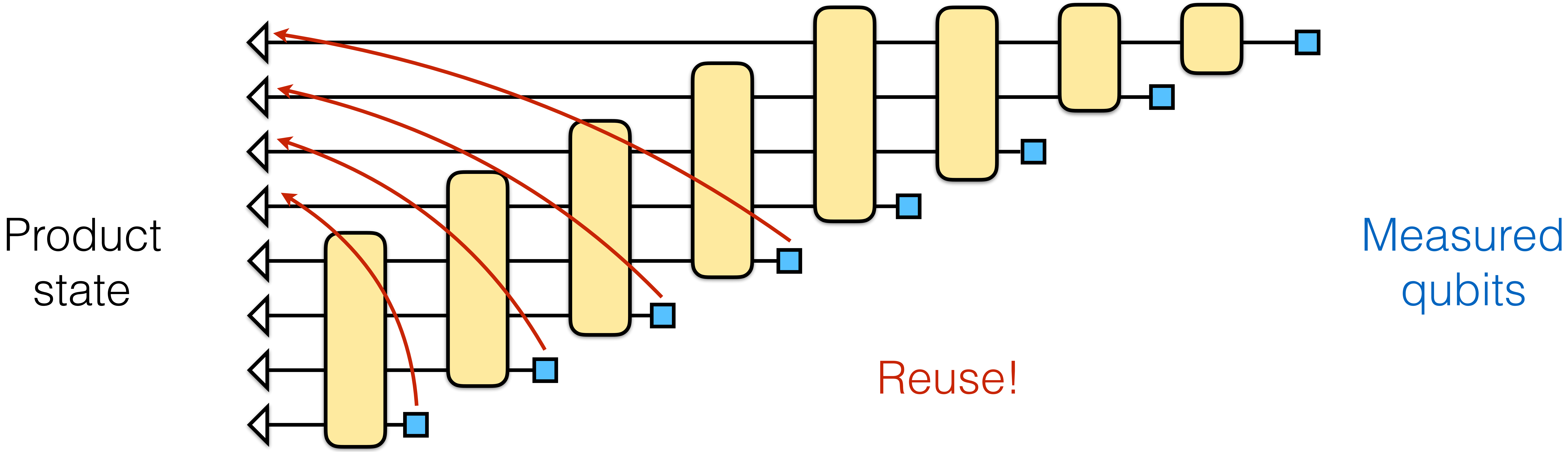
Qubit efficient scheme



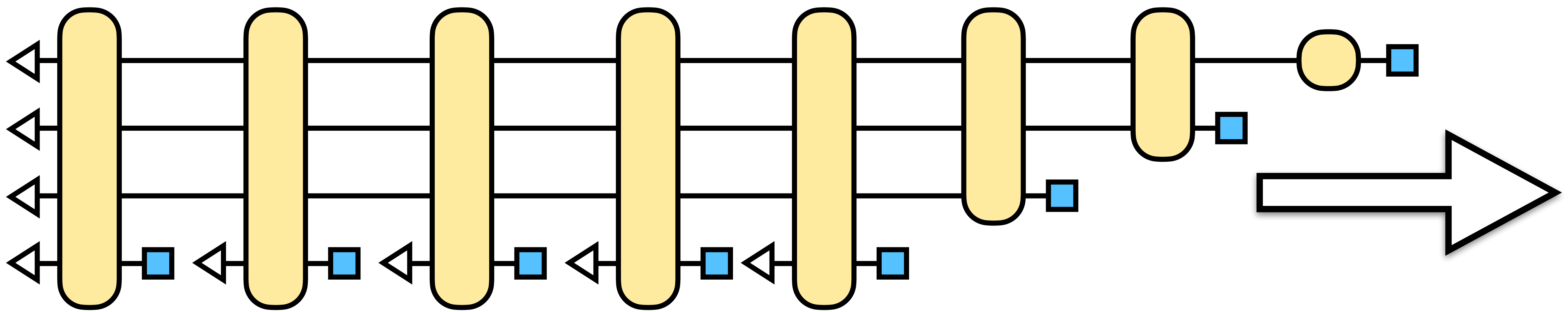
Huggins, Patel, Whaley, Stoudenmire, 1803.11537
see also Cramer et al, Nat. Comm. 2010

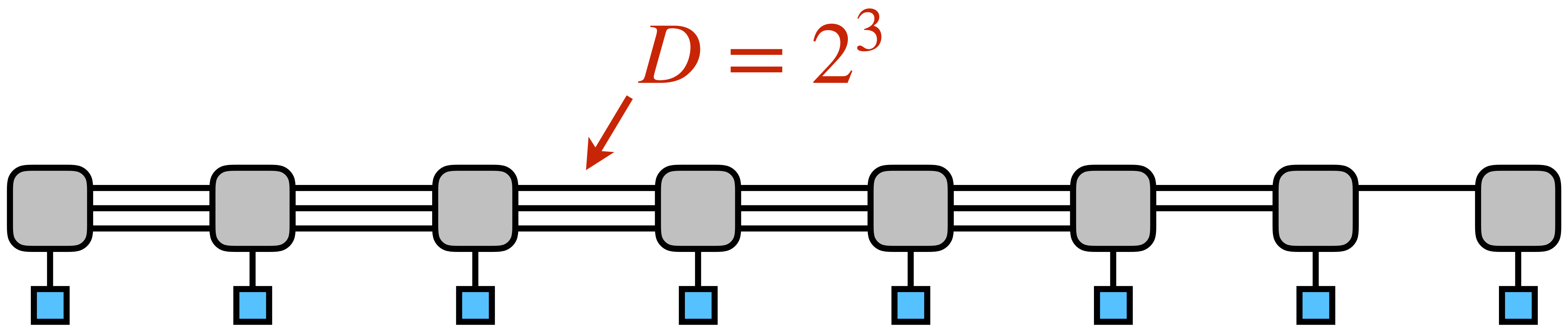


Qubit efficient scheme



Huggins, Patel, Whaley, Stoudenmire, 1803.11537
see also Cramer et al, Nat. Comm. 2010





MPS with exponentially large bond dimensions

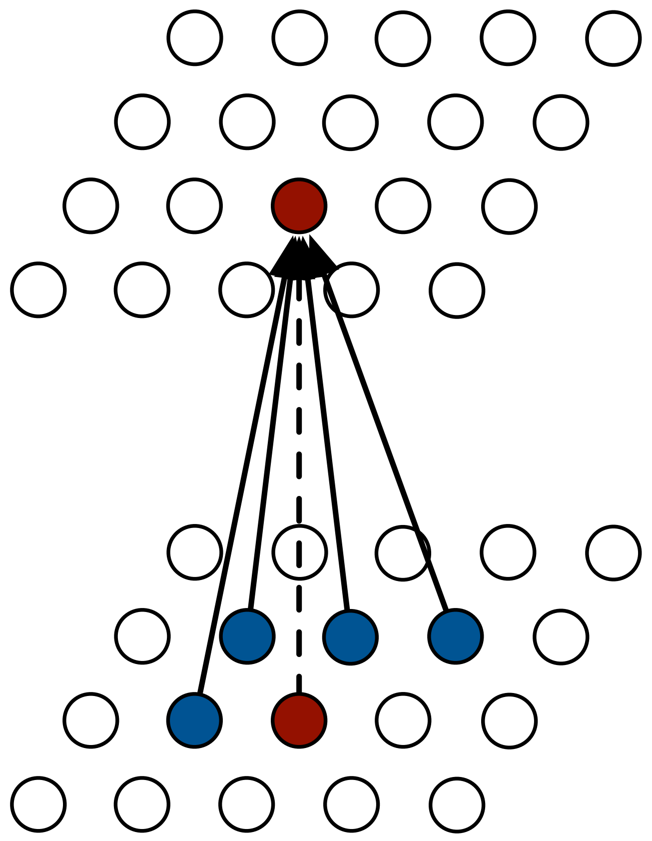
Well, back to Boltzmann Machines

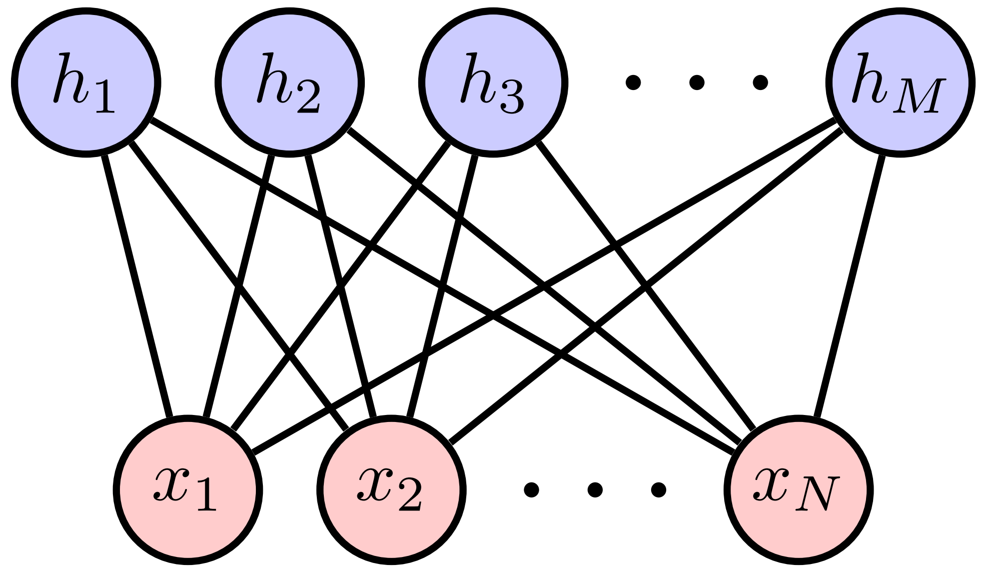
Quantitative Performance

Binarized MNIST test NLL, lower is better*

81.3 ~84.6 ~86.3

PixelCNN DBM RBM



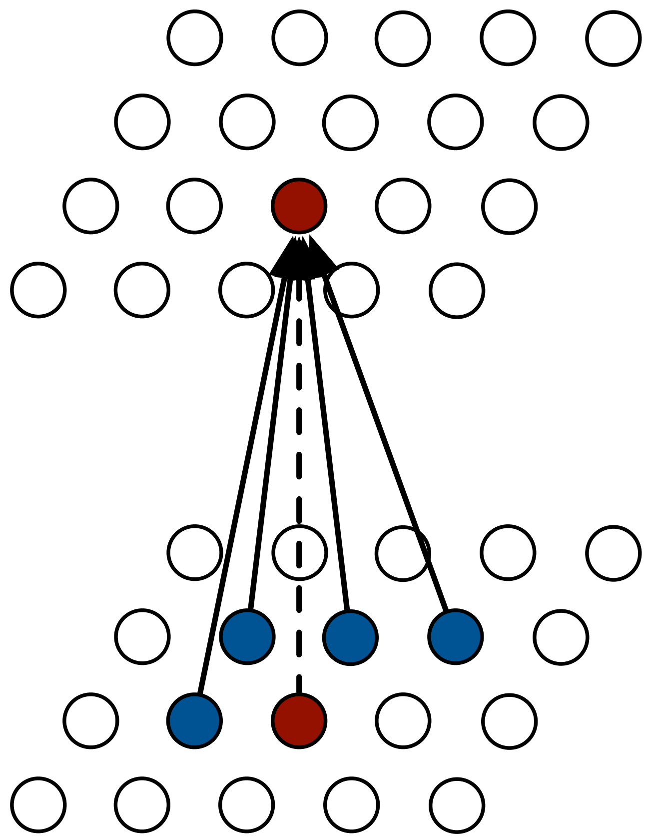


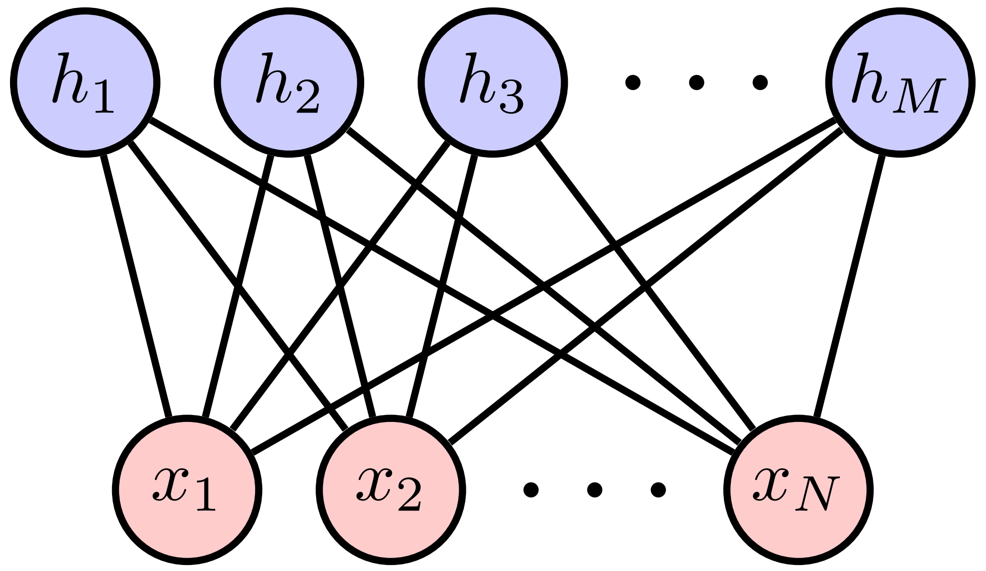
*Lower test NLL may not imply better sample quality, Theis et al, 1511.01844

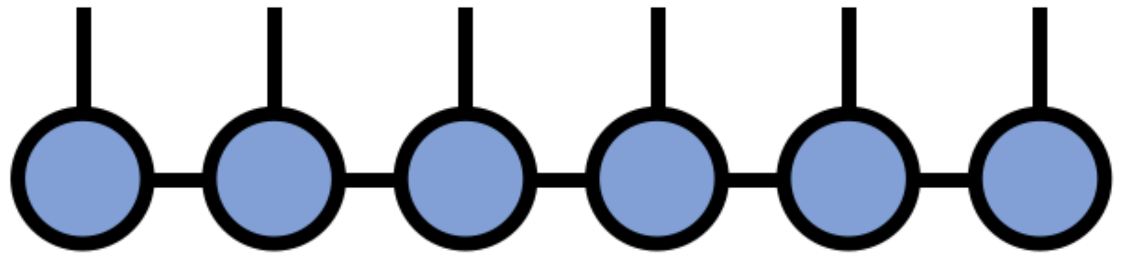
Quantitative Performance

Binarized MNIST test NLL, lower is better*

81.3	~84.6	~86.3	101.5
PixelCNN	DBM	RBM	MPS (D=100)







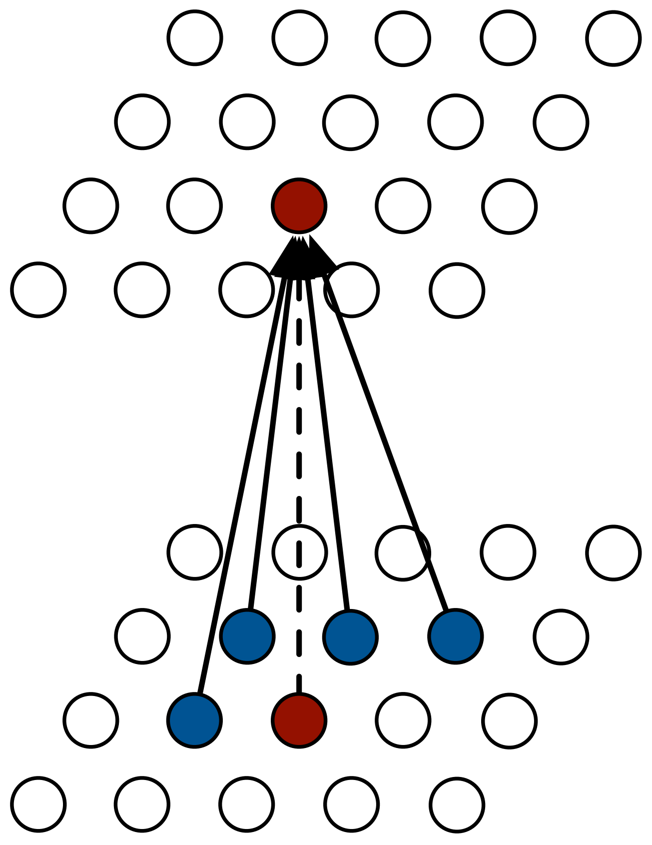
Han et al, PRX 18'

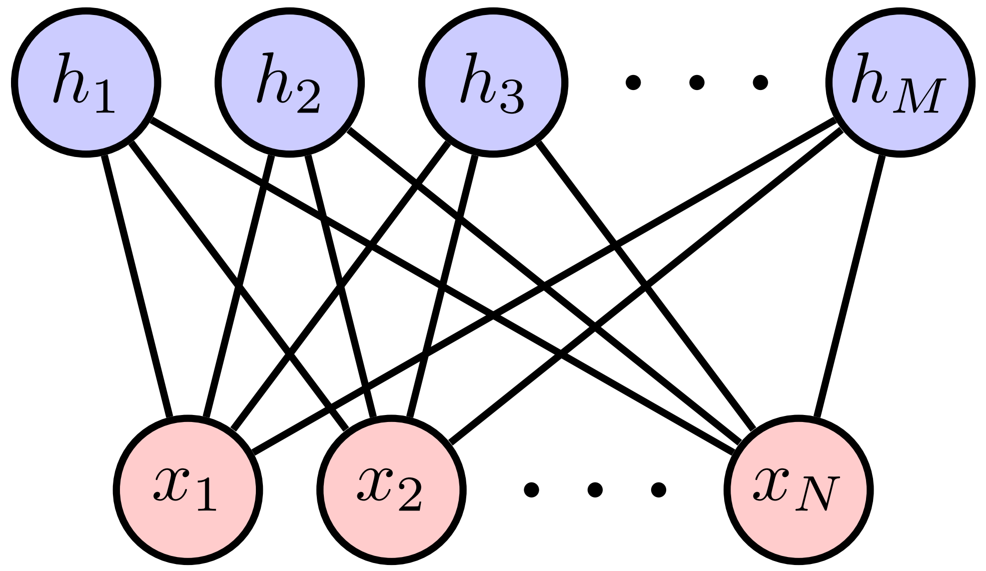
*Lower test NLL may not imply better sample quality, Theis et al, 1511.01844

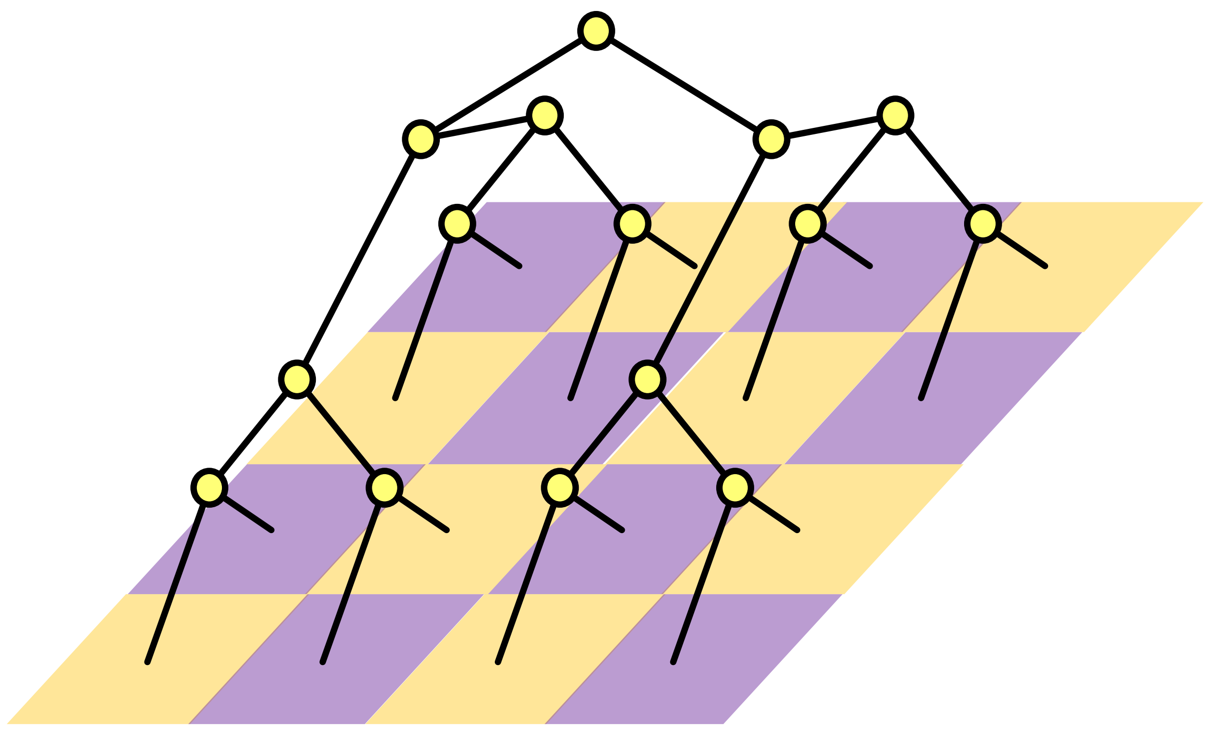
Quantitative Performance

Binarized MNIST test NLL, lower is better*

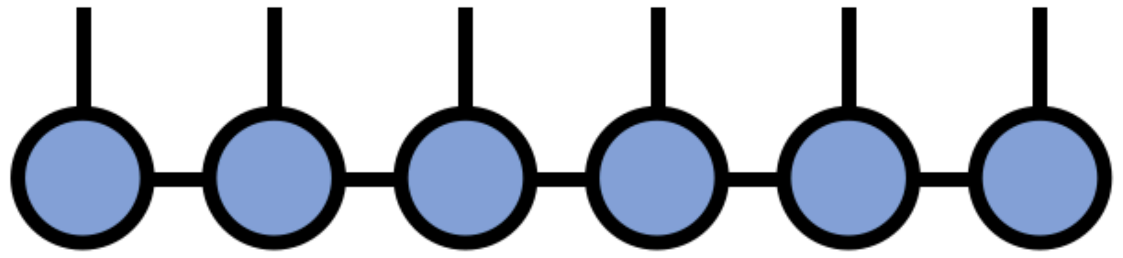
81.3	~84.6	~86.3	94.3	101.5
PixelCNN	DBM	RBM	TTN (D=50)	MPS (D=100)







With Song Cheng &
Pan Zhang, unpublished



Han et al, PRX 18’

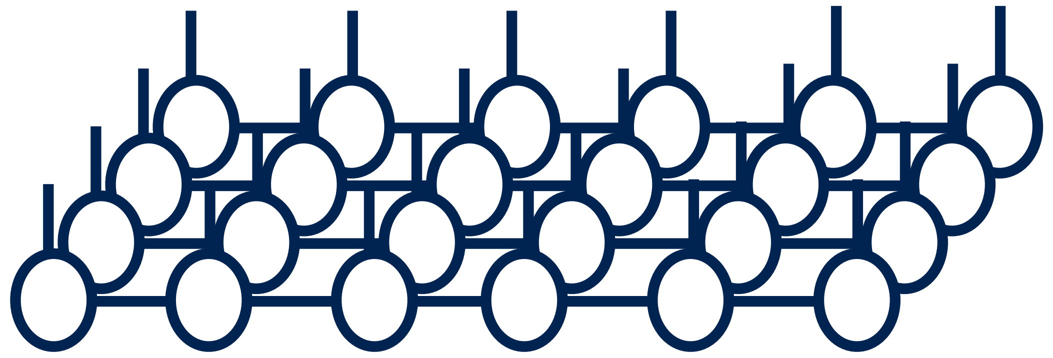
*Lower test NLL may not imply better sample quality, Theis et al, 1511.01844

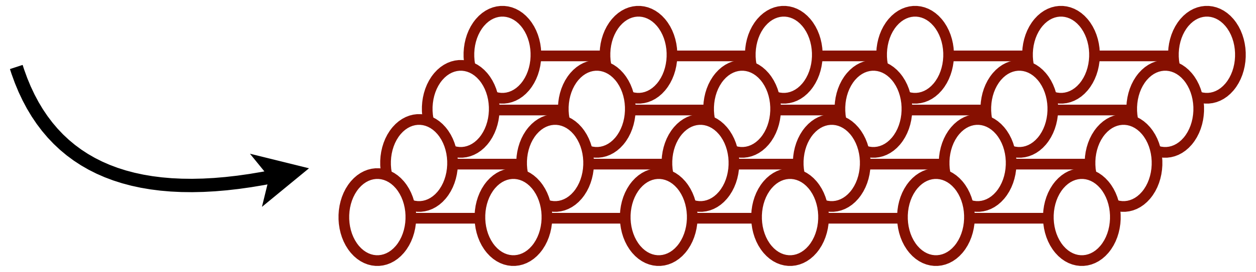
“Positive” PEPS

aka hidden Markov random field model

$$p(\boldsymbol{x}) = \sum_{\boldsymbol{h}} \prod_i f^i(x_i, \boldsymbol{h})/Z$$

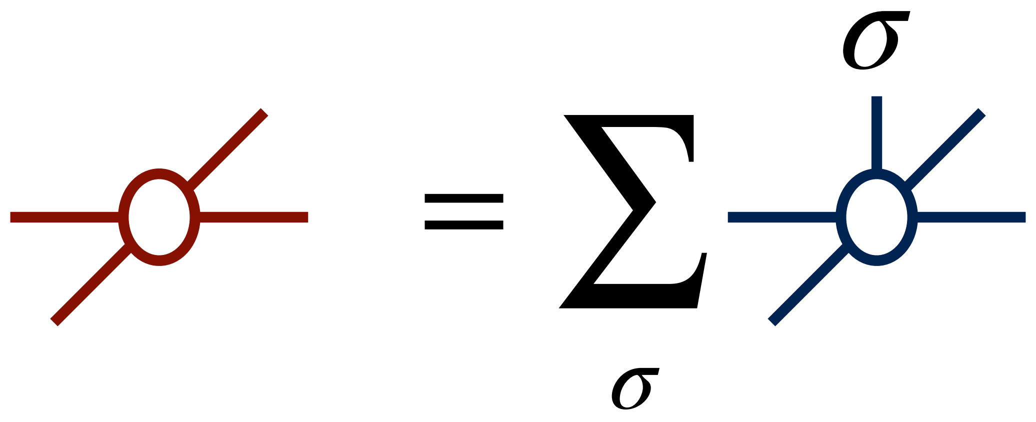
Probability model

$p(\boldsymbol{x}) =$  $/Z$



Why positive?

- Smaller truncation error
- No need for double layer tensor

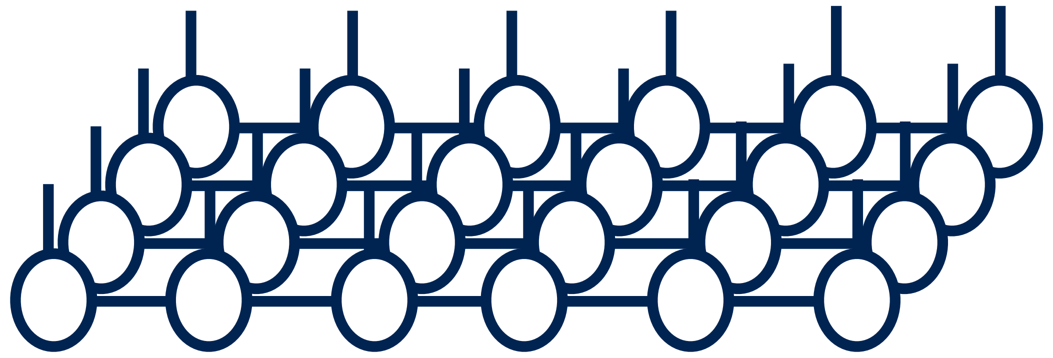


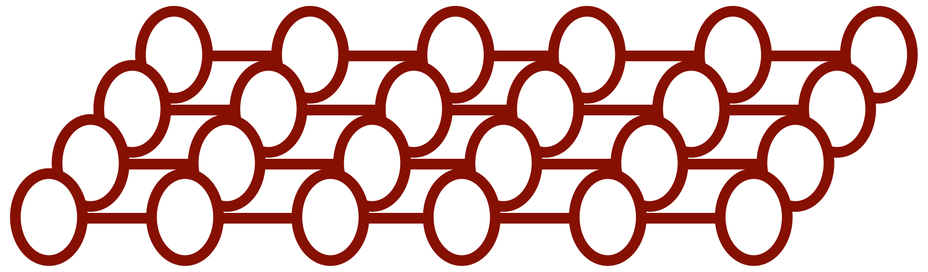
“Positive” PEPS

aka hidden Markov random field model

$$p(\boldsymbol{x}) = \sum_{\boldsymbol{h}} \prod_i f^i(x_i, \boldsymbol{h})/Z$$

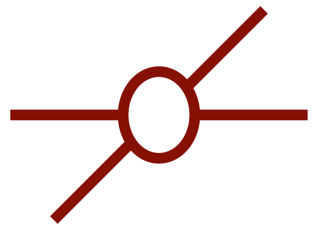
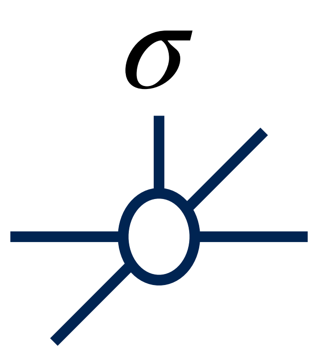
Probability model

$p(\boldsymbol{x}) =$  $/Z$

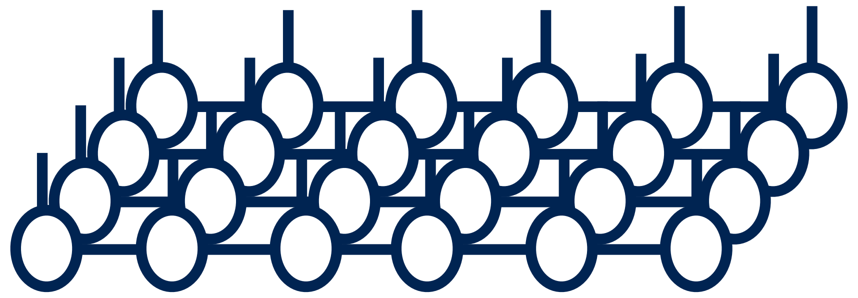
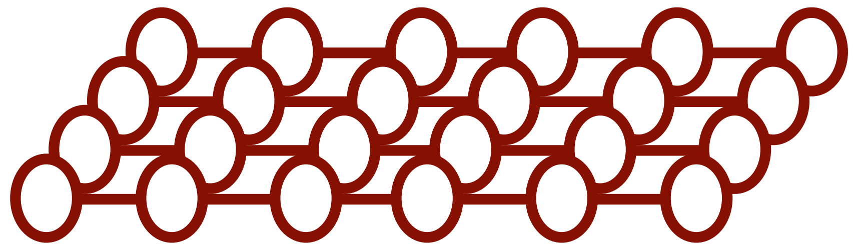


Why positive?

- Smaller truncation error
- No need for double layer tensor

 $= \sum_{\sigma}$ 

Loss function

$\mathscr{L} = -\langle \ln ($  $)\rangle_{\boldsymbol{x}\sim\mathscr{D}} + \ln ($  $)$

Contract for each given sample

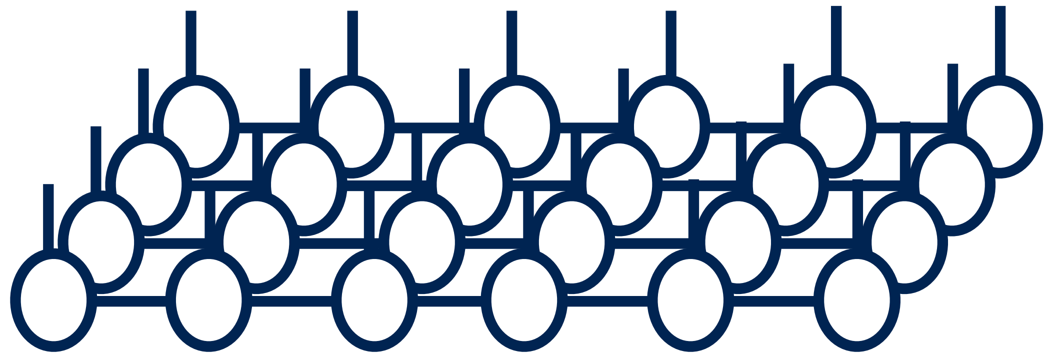
Contract once

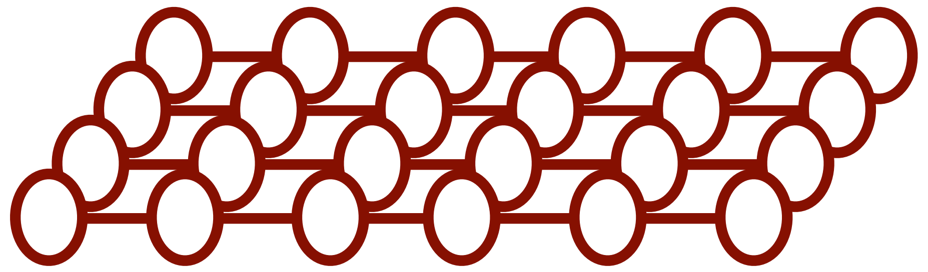
"Positive" PEPS

aka hidden Markov random field model

$$p(\boldsymbol{x}) = \sum_{\boldsymbol{h}} \prod_{i} f^i(x_i, \boldsymbol{h})/Z$$

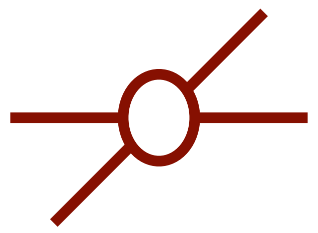
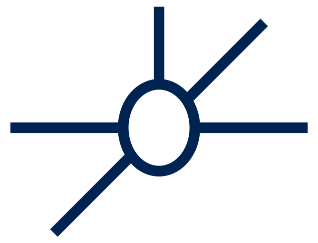
Probability model

$p(\boldsymbol{x}) =$  $/Z$



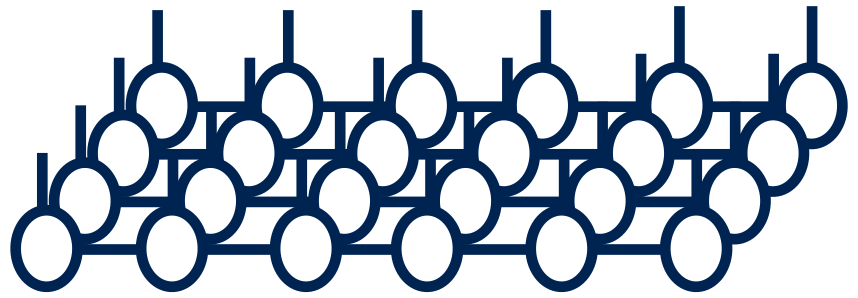
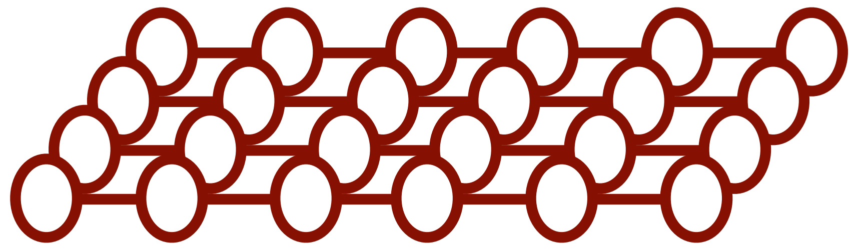
Why positive?

- Smaller truncation error
- No need for double layer tensor

 $= \sum_{\sigma}$ 

$\boldsymbol{\sigma}$

Loss function

$\mathscr{L} = - \langle \ln ($  $) \rangle_{\boldsymbol{x} \sim \mathscr{D}} + \ln ($  $)$

Contract for each given sample

Contract once

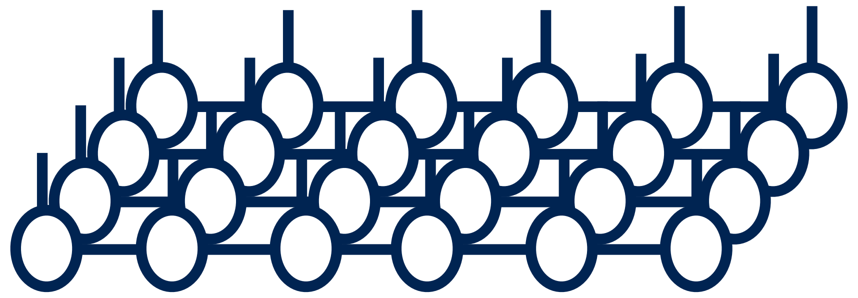
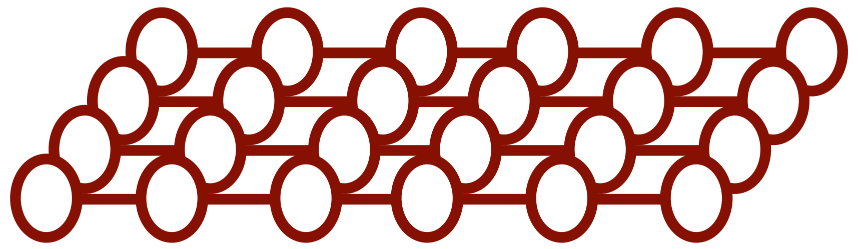
Contract 28x28 PEPS 60000+1 times per epoch!

Quantitative Performance

Binarized MNIST test NLL, lower is better

81.3	~84.6	~86.3	94.3	101.5
PixelCNN	DBM	RBM	TTN (D=50)	MPS (D=100)

Loss function

$\mathscr{L} = -\langle \ln ($  $) \rangle_{\boldsymbol{x} \sim \mathscr{D}} + \ln ($  $)$

Contract for each given sample

Contract once

Contract 28x28 PEPS 60000+1 times per epoch!

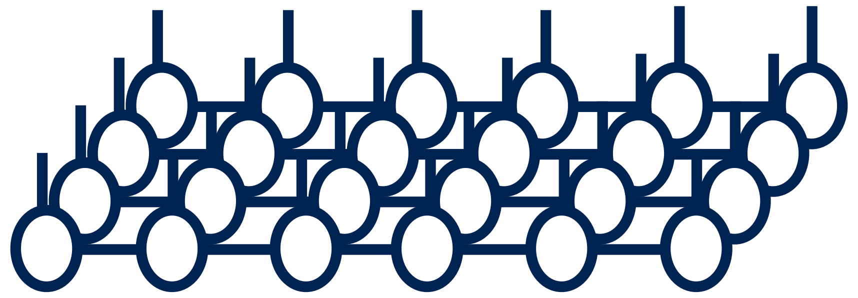
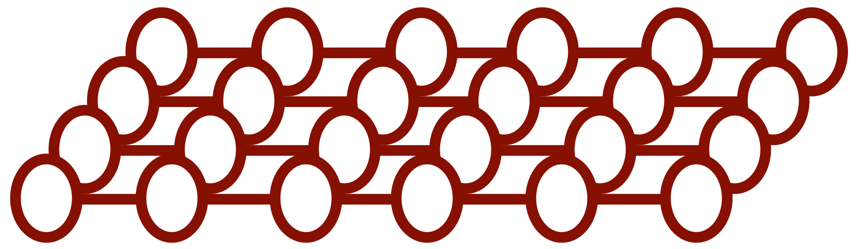
Quantitative Performance

Binarized MNIST test NLL, lower is better

81.3	~84.6	~86.3	**92**	94.3	101.5
PixelCNN	DBM	RBM	**PEPS+ (D=5)**	TTN (D=50)	MPS (D=100)

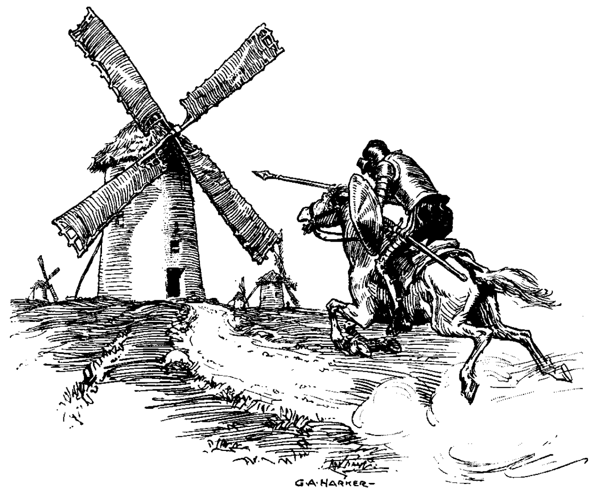
With Song Cheng &
Pan Zhang, unpublished

Loss function

$\mathscr{L} = -\big\langle \ln \big($  $\big)\big\rangle_{\boldsymbol{x}\sim\mathscr{D}} + \ln \big($  $\big)$

Contract for each given sample

Contract once

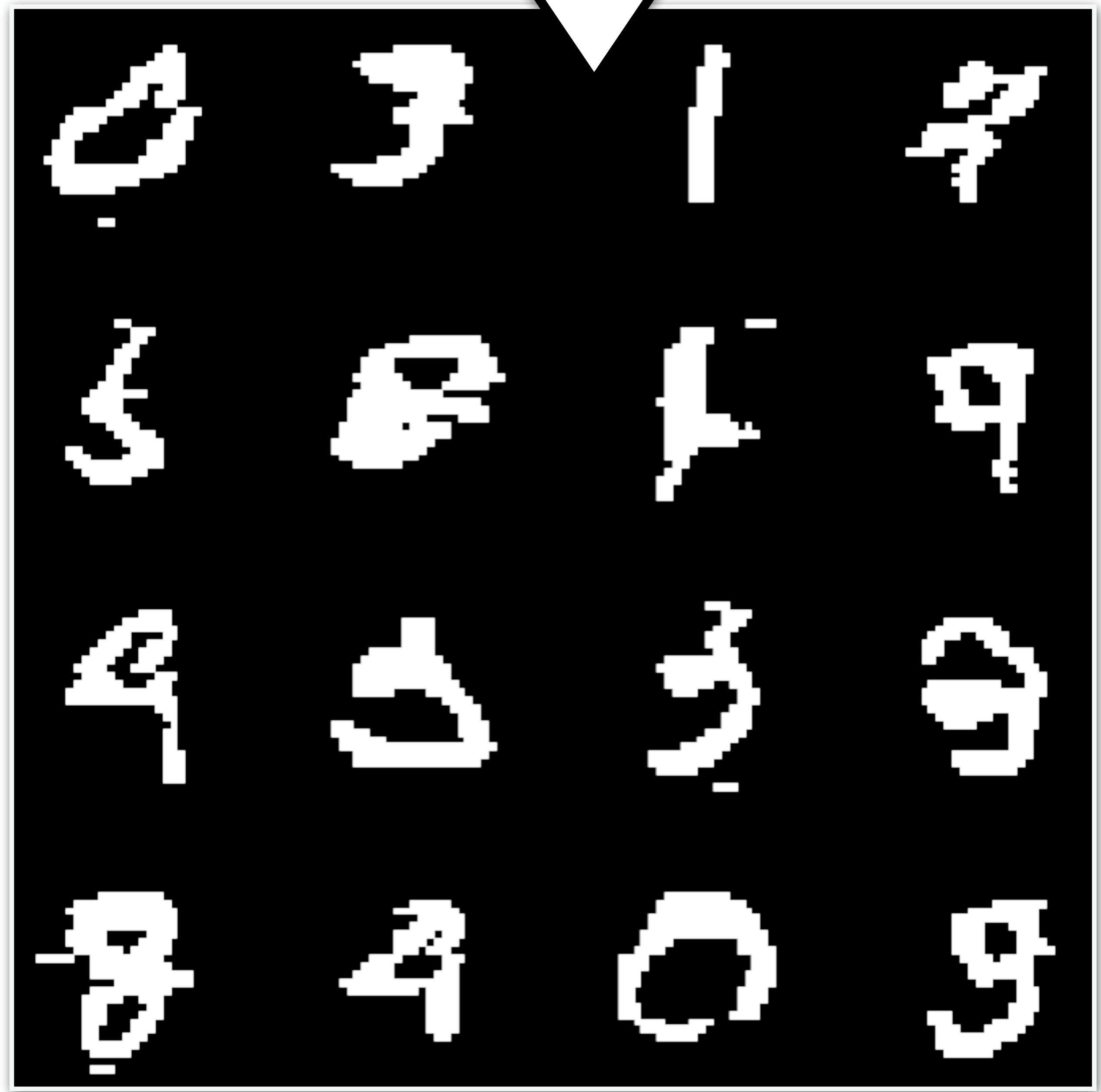


Contract 28x28 PEPS 60000+1 times per epoch!

Quantitative Performance

Binarized MNIST test NLL, lower is better

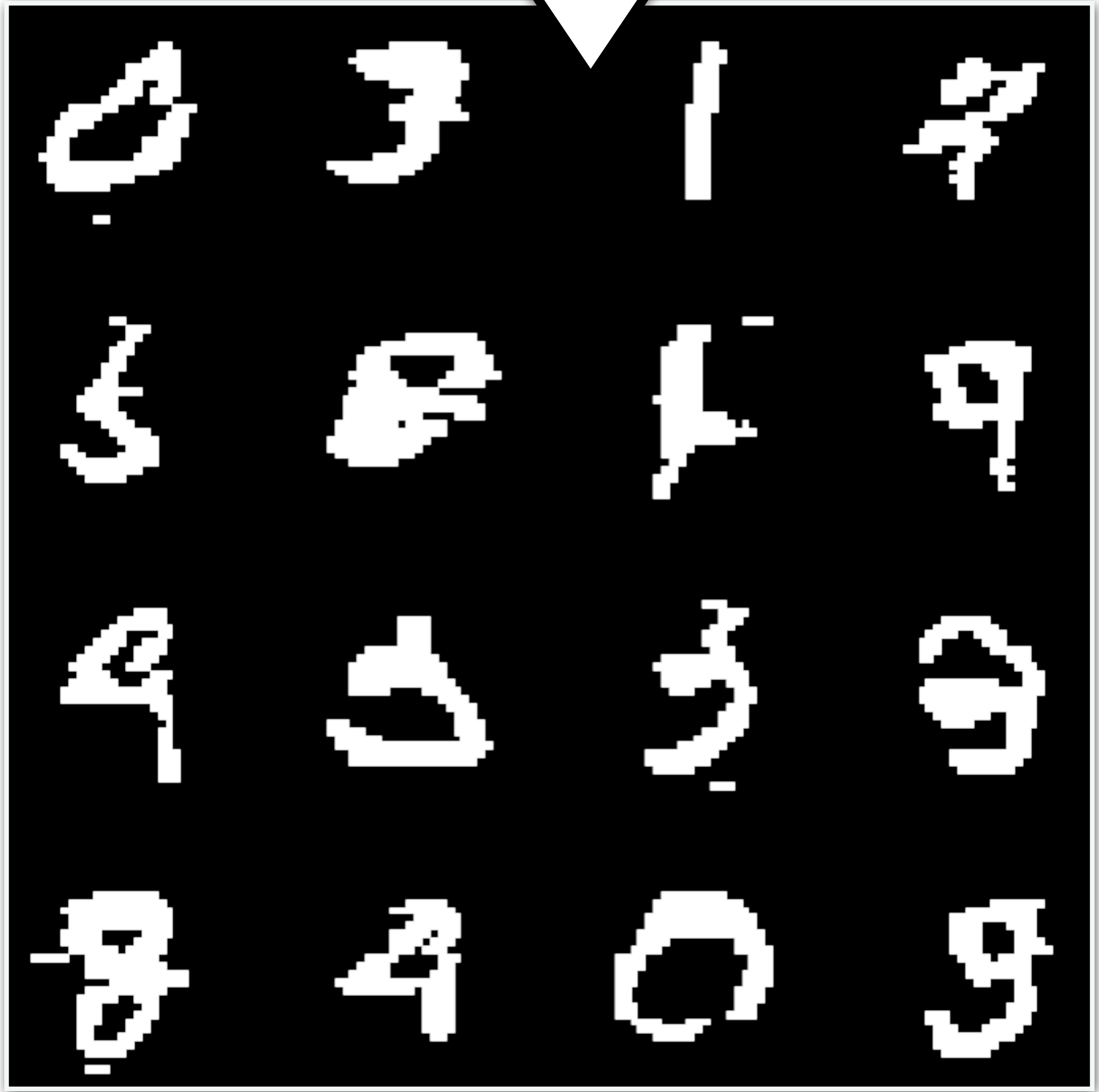
81.3	~84.6	~86.3	**92**	94.3	101.5
PixelCNN	DBM	RBM	**PEPS+ (D=5)**	TTN (D=50)	MPS (D=100)



Quantitative Performance

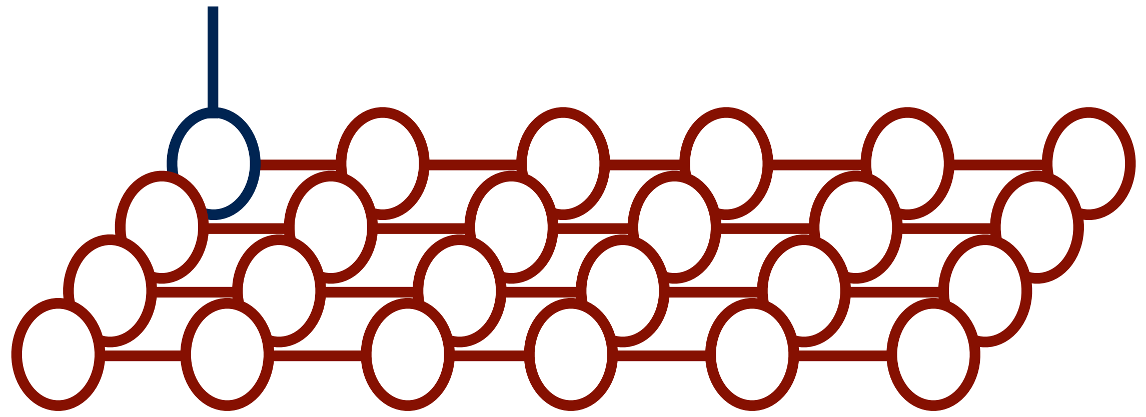
Binarized MNIST test NLL, lower is better

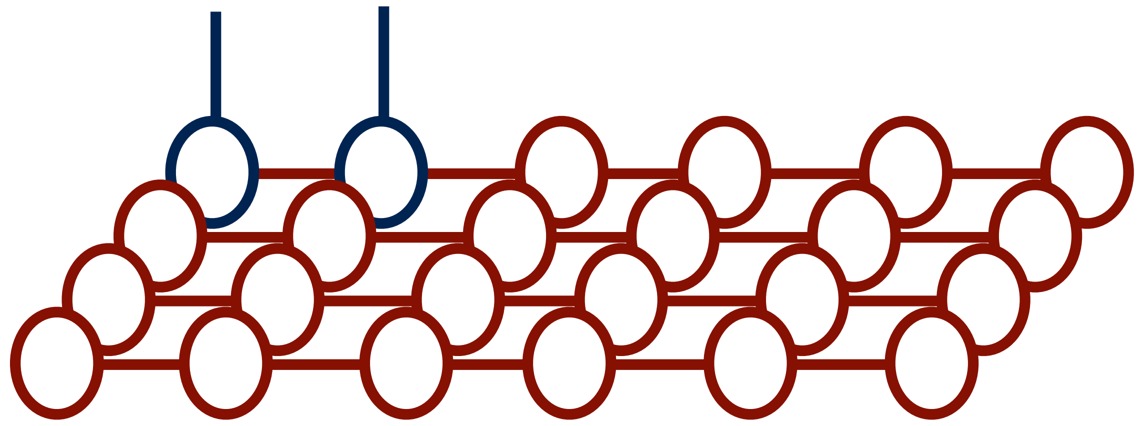
81.3	~84.6	~86.3	**92**	94.3	101.5
PixelCNN	DBM	RBM	**PEPS+ (D=5)**	TTN (D=50)	MPS (D=100)

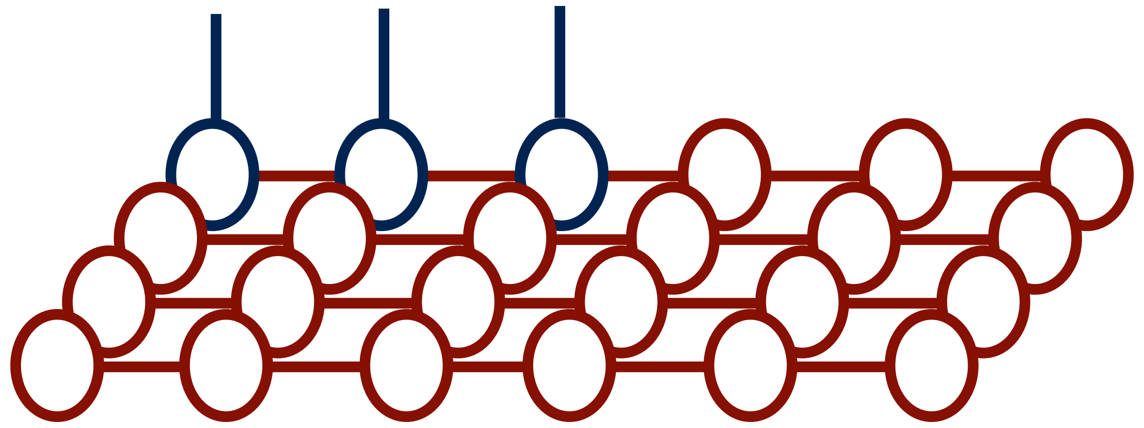


Direct Sampling PEPS+

$$p(\boldsymbol{x}) = \prod_i p(x_i | \boldsymbol{x}_{<i}) = \prod_i \frac{p(\boldsymbol{x}_{<i+1})}{p(\boldsymbol{x}_{<i})}$$

$p(x_1) \sim$ 

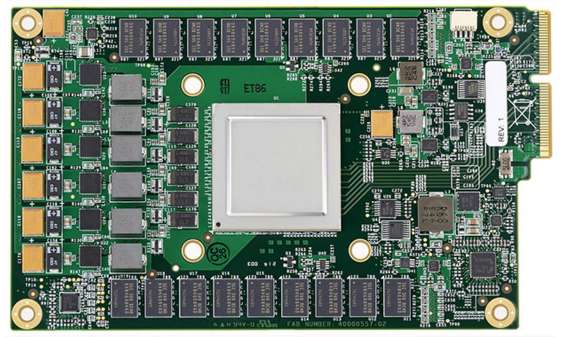
$p(x_1, x_2) \sim$ 

$p(x_1, x_2, x_3) \sim$ 

See also Rams et al, 1811.06518

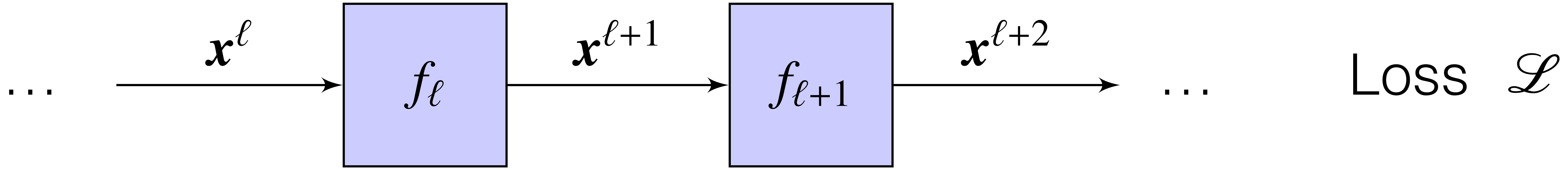
Bonus: What does deep learning offer to tensor network states & algorithms?

Differentiable tensor libraries with GPU/TPU/FPGA/… support

The engine of deep learning: Back-Propagation algorithm

computation graph



$$\frac{\partial \mathscr{L}}{\partial \boldsymbol{x}^{\ell}} = \frac{\partial \mathscr{L}}{\partial \boldsymbol{x}^{\ell+1}} \left(\frac{\partial \boldsymbol{x}^{\ell+1}}{\partial \boldsymbol{x}^{\ell}} \right)$$ Jacobian-Vector Product

Computes gradients efficiently & accurately
Via reverse mode automatic differentiation

Differentiable Programming

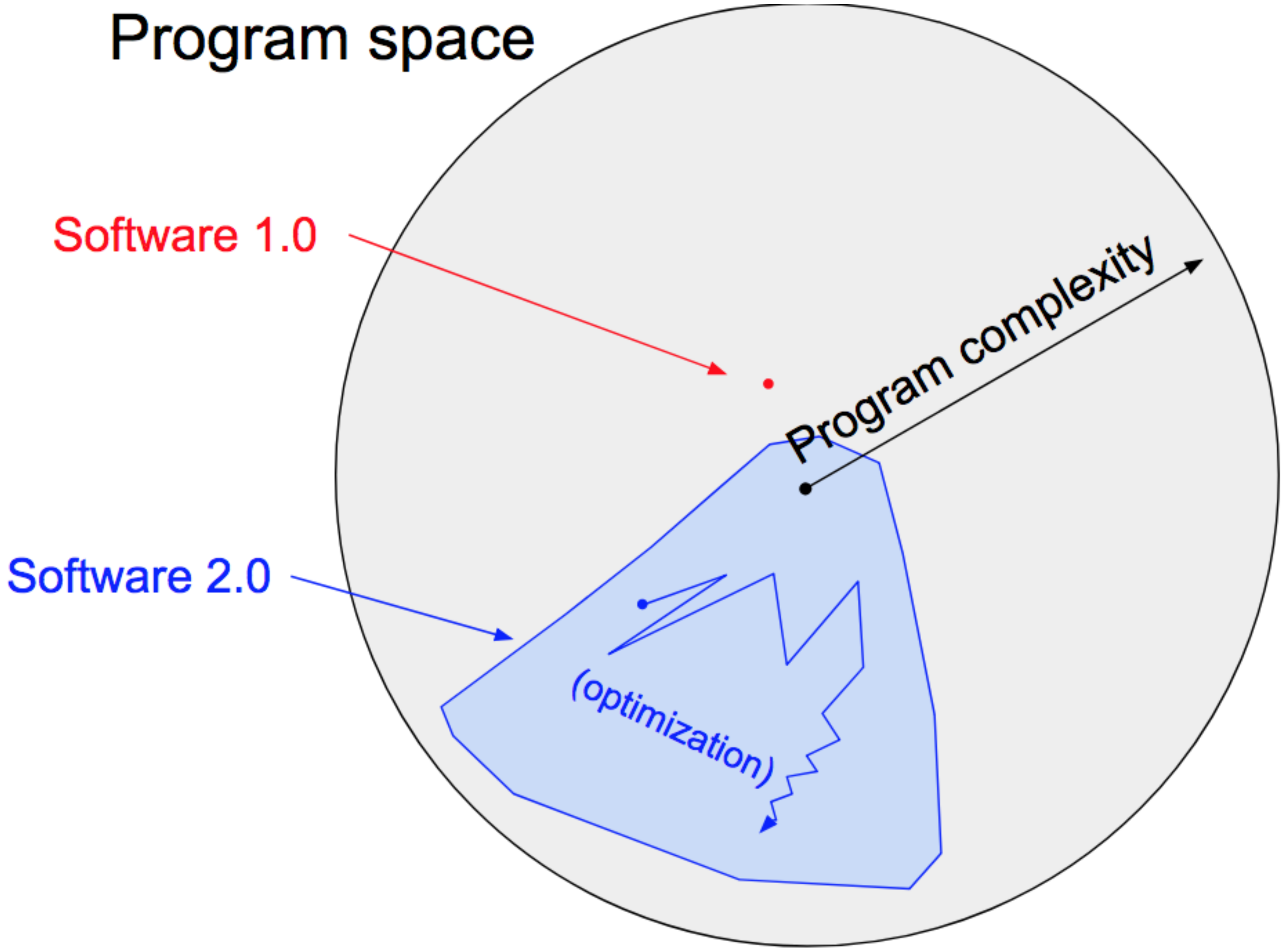
Benefits

- Computationally homogeneous
- Simple to bake into silicon
- Constant running time
- Constant memory usage
- Highly portable & agile
- Modules can meld into an optimal whole
- Better than humans

Andrej Karpathy

Director of AI at Tesla. Previously Research Scientist at OpenAI and PhD student at Stanford. I like to train deep neural nets on large datasets.

https://medium.com/@karpathy/software-2-0-a64152b37c35



Writing software 2.0 by gradient search in the program space

Differentiable Scientific Programming

- Most linear algebra operations (Eigen, SVD!) are differentiable
- Loop/Condition/Sort/Permutations are also differentiable
- ODE integrators are differentiable with O(1) memory
- Differentiable ray tracer and Differentiable fluid simulations
- Differentiable Monte Carlo/Tensor Network/Functional RG/ Dynamical Mean Field Theory/Density Functional Theory…

Differentiable programming is more than training neural networks

Differentiable Eigensolver

$$H\Psi = \Psi\Lambda$$

Forward mode: What happen if $H = H + dH$? Perturbation theory

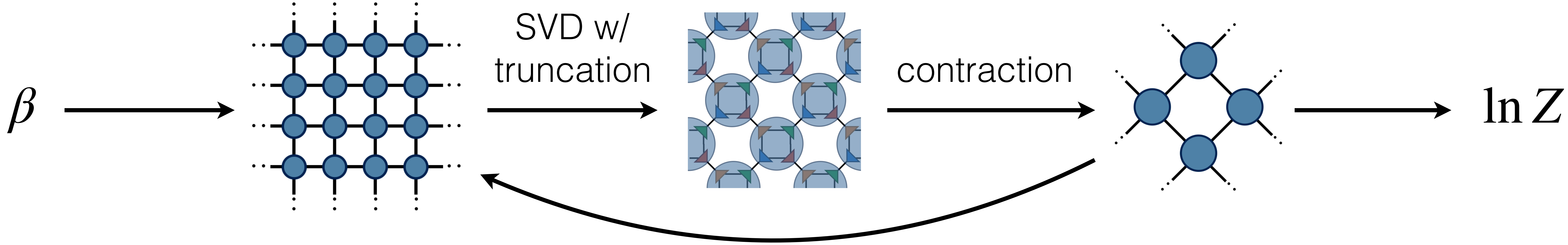
Reverse mode: How should I change H given $\partial\mathscr{L}/\partial\Psi$ and $\partial\mathscr{L}/\partial\Lambda$? **Inverse perturbation theory**

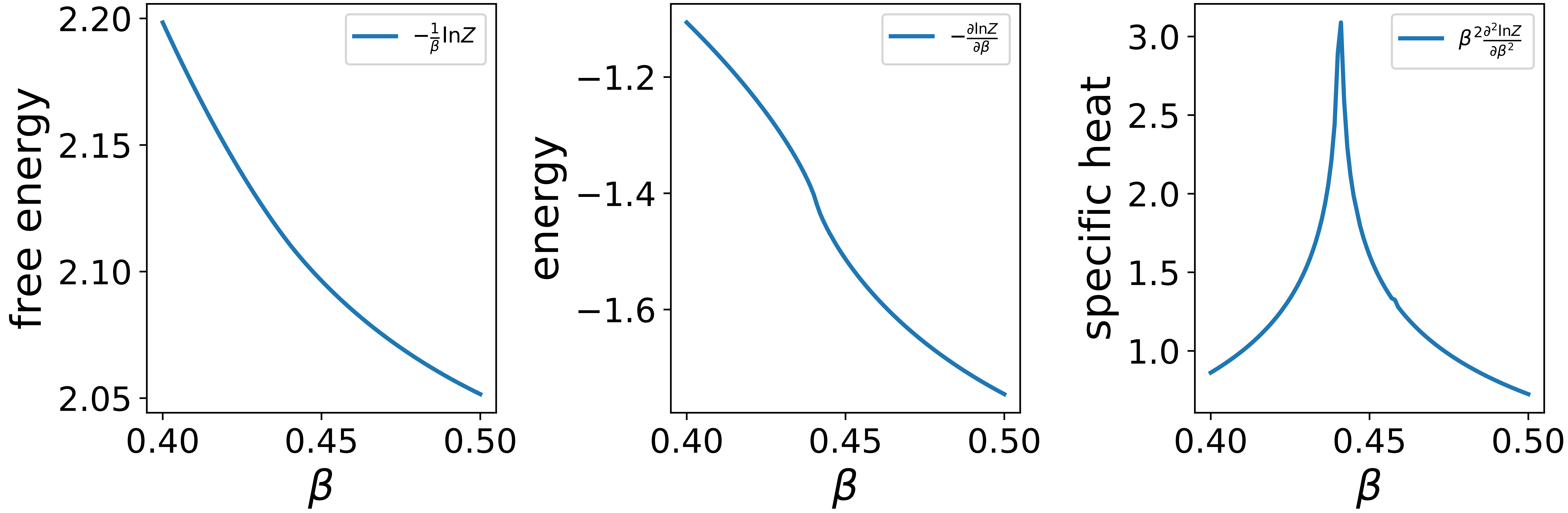
Hamiltonian engineering via differentiable programming

https://github.com/wangleiphy/DL4CSRC/tree/master/2-ising

Differentiable Levin-Nave TRG

computation graph



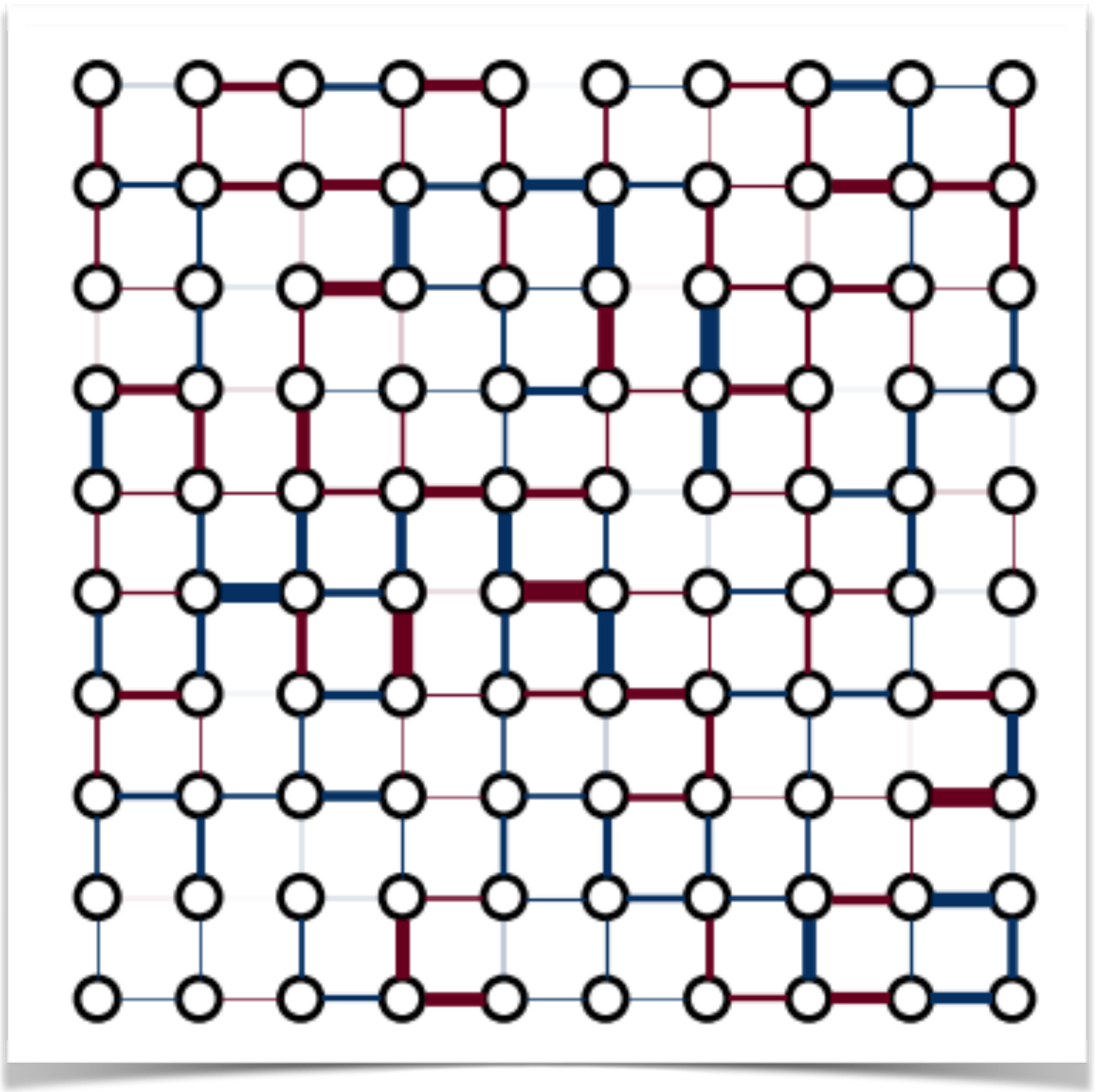


https://github.com/wangleiphy/TRG

- Exact gradient
- Not finite-difference
- No additional codes
- Efforts comparable to the forward pass

Automatic differentiation for high order gradients

Differentiable spin glass solver



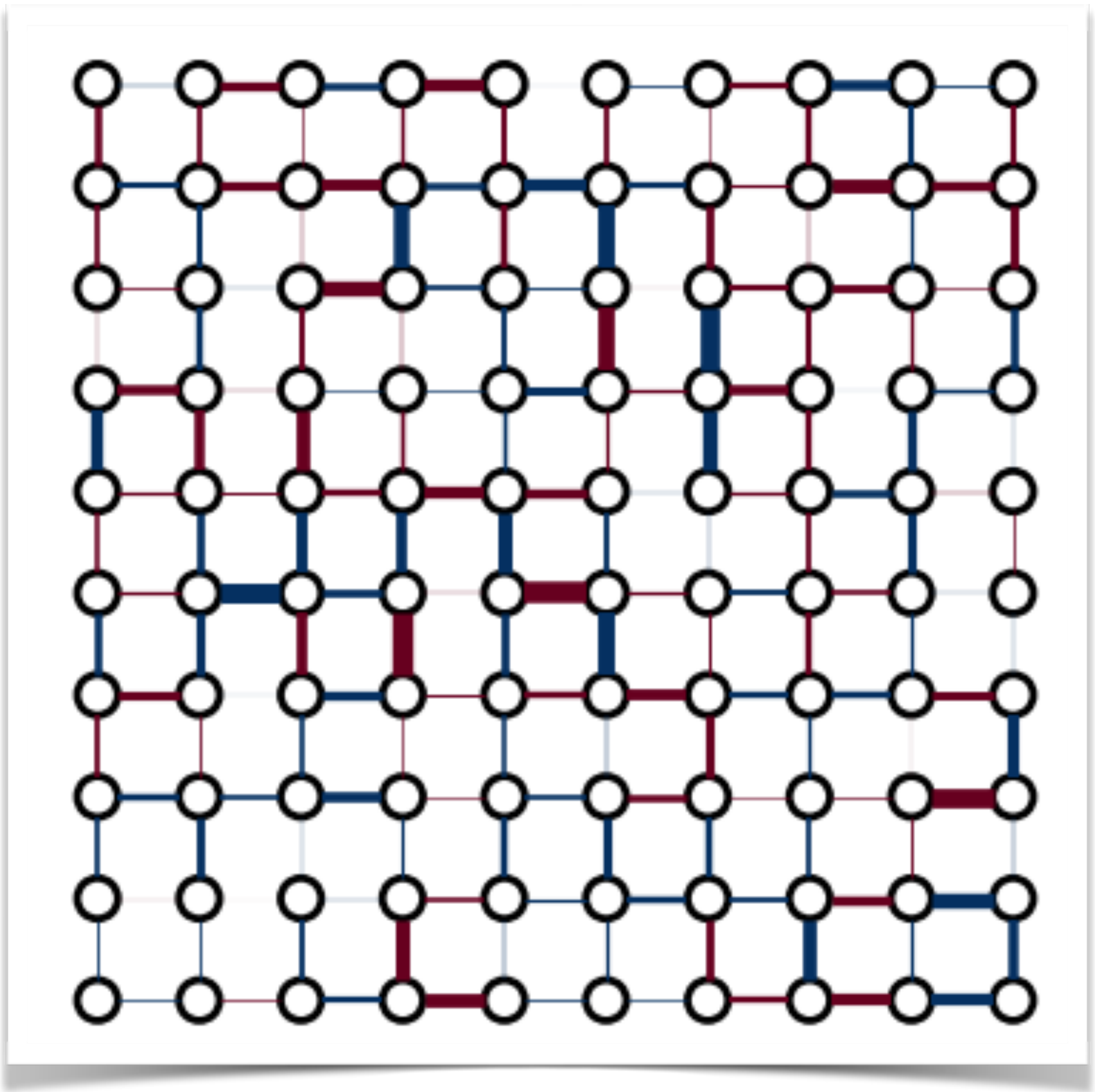
K_{ij} random couplings

$$Z = \sum_{\{\sigma\}} \exp\left(\frac{1}{2} K_{ij}\sigma_i\sigma_j\right)$$

$$\xrightarrow{\text{Contraction}} \ln Z$$

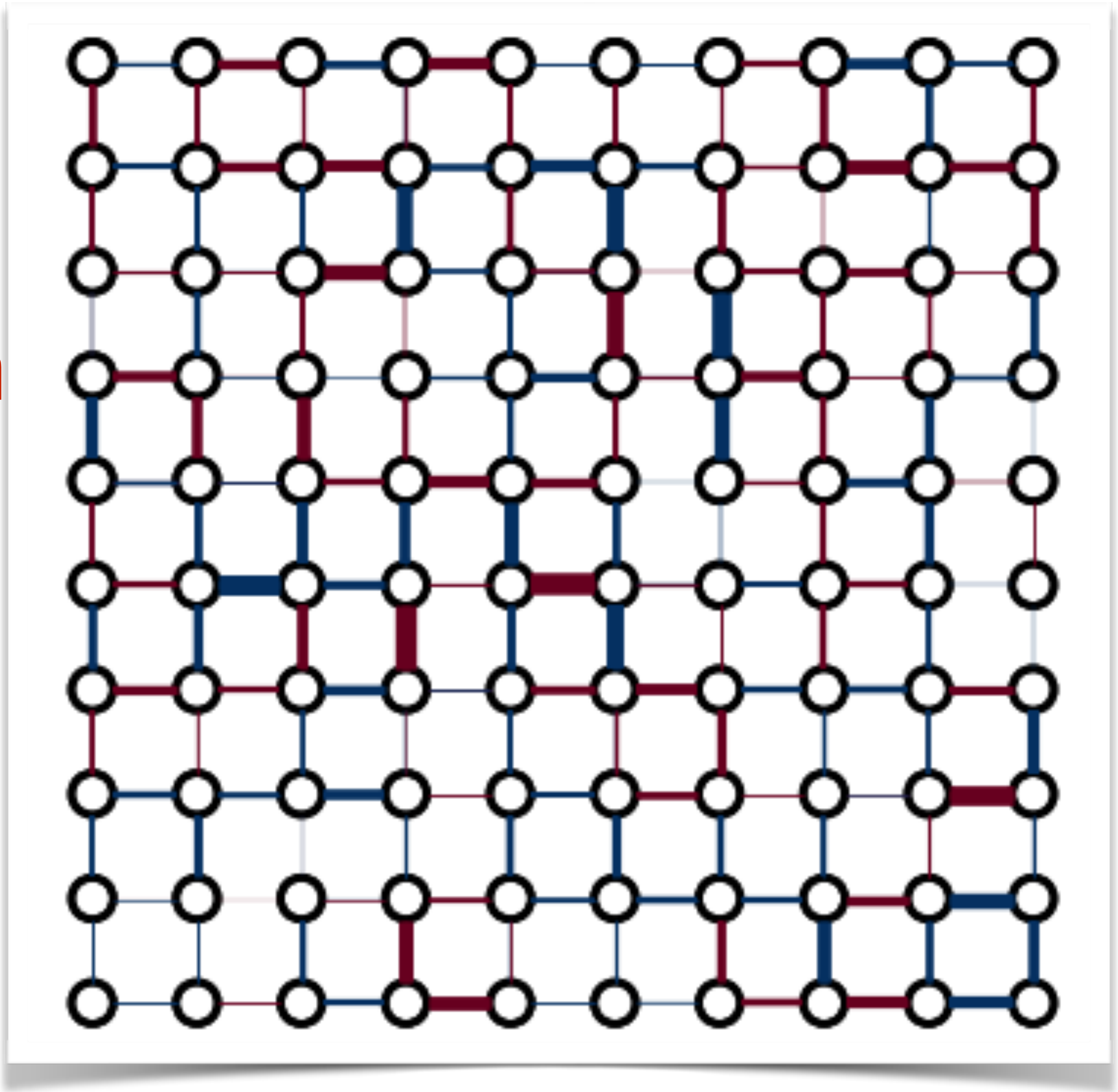
TNS studies of Ising spin glasses: Wang, Qin, Zhou, PRB 2014 Rams et al, 1811.06518

Differentiable spin glass solver



$$Z = \sum_{\{\sigma\}} \exp\left(\frac{1}{2} K_{ij}\sigma_i\sigma_j\right)$$

Contraction $\longrightarrow \ln Z$ Differentiation $\longrightarrow$

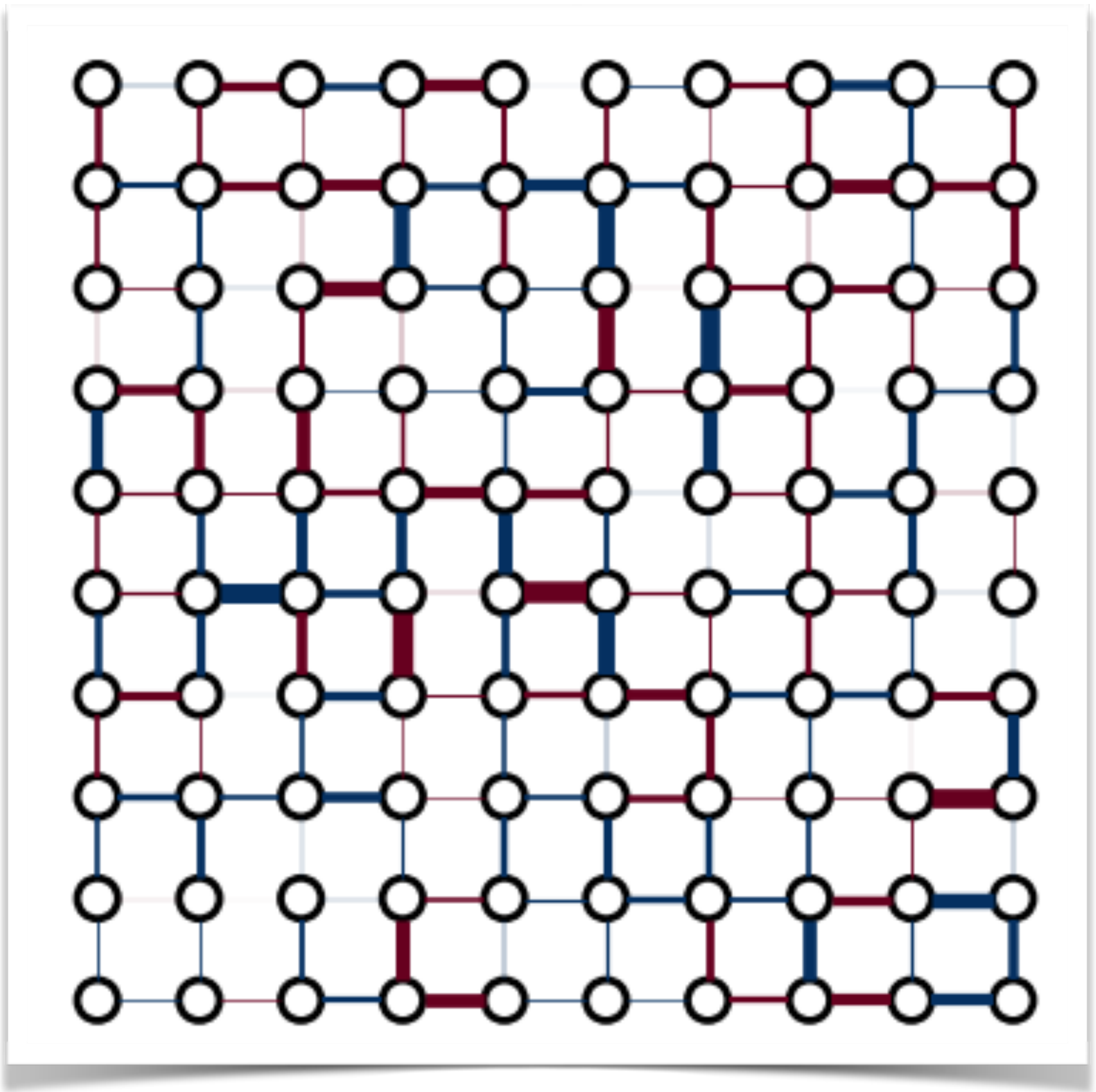


K_{ij} random couplings

$\frac{d \ln Z}{dK_{ij}} = \langle \sigma_i\sigma_j \rangle$ correlations

TNS studies of Ising spin glasses: Wang, Qin, Zhou, PRB 2014 Rams et al, 1811.06518

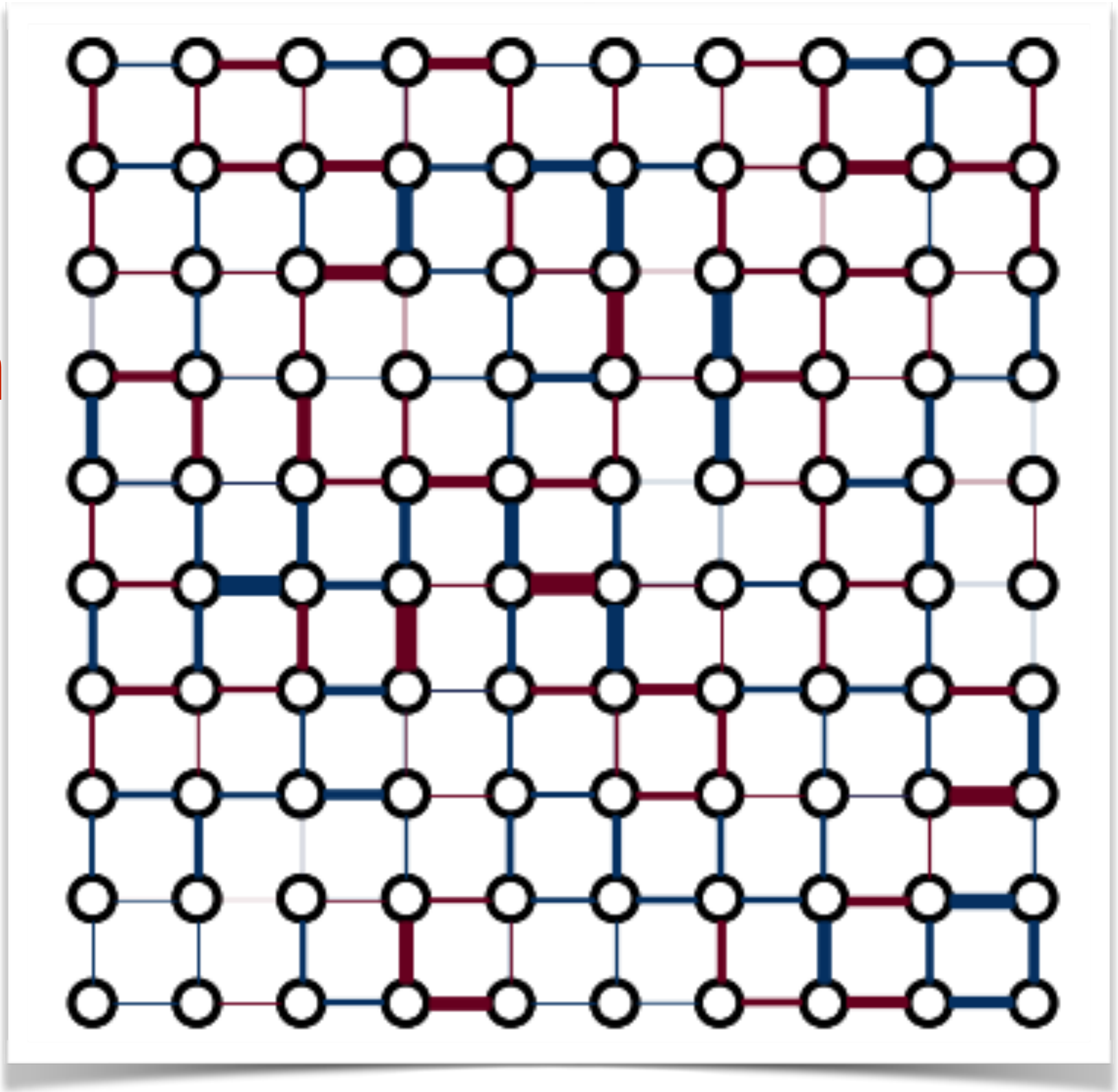
Differentiable spin glass solver



$$Z = \sum_{\{\sigma\}} \exp\left(\frac{1}{2} K_{ij}\sigma_i\sigma_j\right)$$

Contraction $\longrightarrow \ln Z$ Differentiation $\longrightarrow$

Gradient descend $\longleftarrow$



K_{ij} random couplings

$\frac{d \ln Z}{dK_{ij}} = \langle \sigma_i \sigma_j \rangle$ correlations

Tensor network solver for inverse Ising problems

TNS studies of Ising spin glasses: Wang, Qin, Zhou, PRB 2014 Rams et al, 1811.06518

Gradient based variational optimization

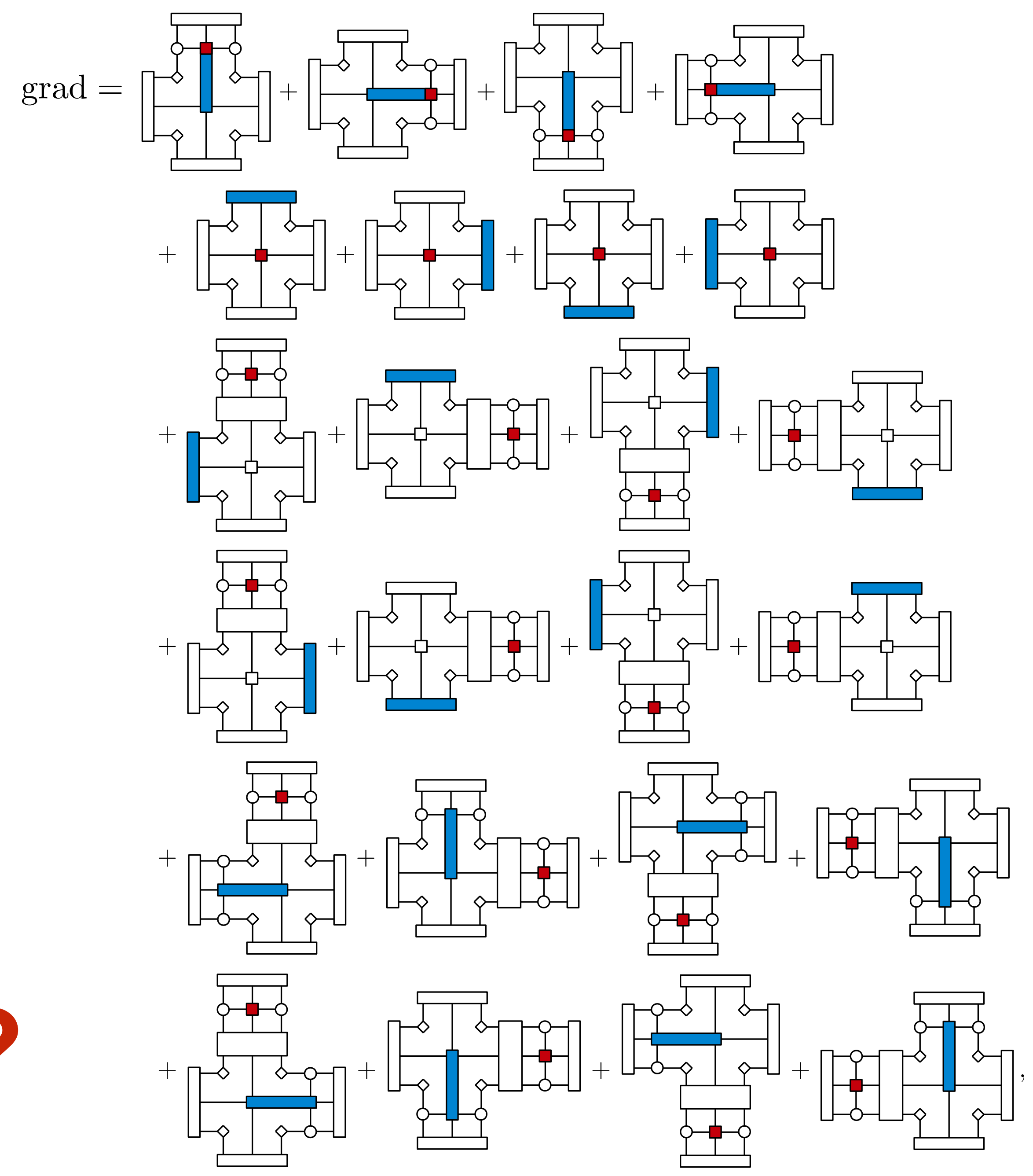
Generative Modeling

$$\mathscr{L} = \langle E(\boldsymbol{x}) \rangle_{\boldsymbol{x} \sim \mathscr{D}} + \ln Z$$

Gradient optimization works fine

Variational ground state, why not ?

$$\mathscr{L} = \ln \langle \Psi | \hat{H} | \Psi \rangle - \ln \langle \Psi | \Psi \rangle$$



Vanderstraeten et al, PRB 16'

Human only cares about tensor contraction
Differentiable programing takes care of the gradients

Thank You!

Song Cheng Jin-Guo Liu Jing Chen Zhiyuan Xie Pan Zhang Tao Xiang