Bridge between Classical & Quantum Machine Learning

Jack Y. Araz
Institute For Particle Physics Phenomenology
Durham University

Based on
JHEP 08 (2021) 112; arXiv: 2106.08334 [hep-ph] & arXiv: 2202.10471 [quant-ph]
with Michael Spannowsky

Machine Learning at GGI, Florence, IT.
September 6th, 2022

Durham University

Science & Technology Facilities Council

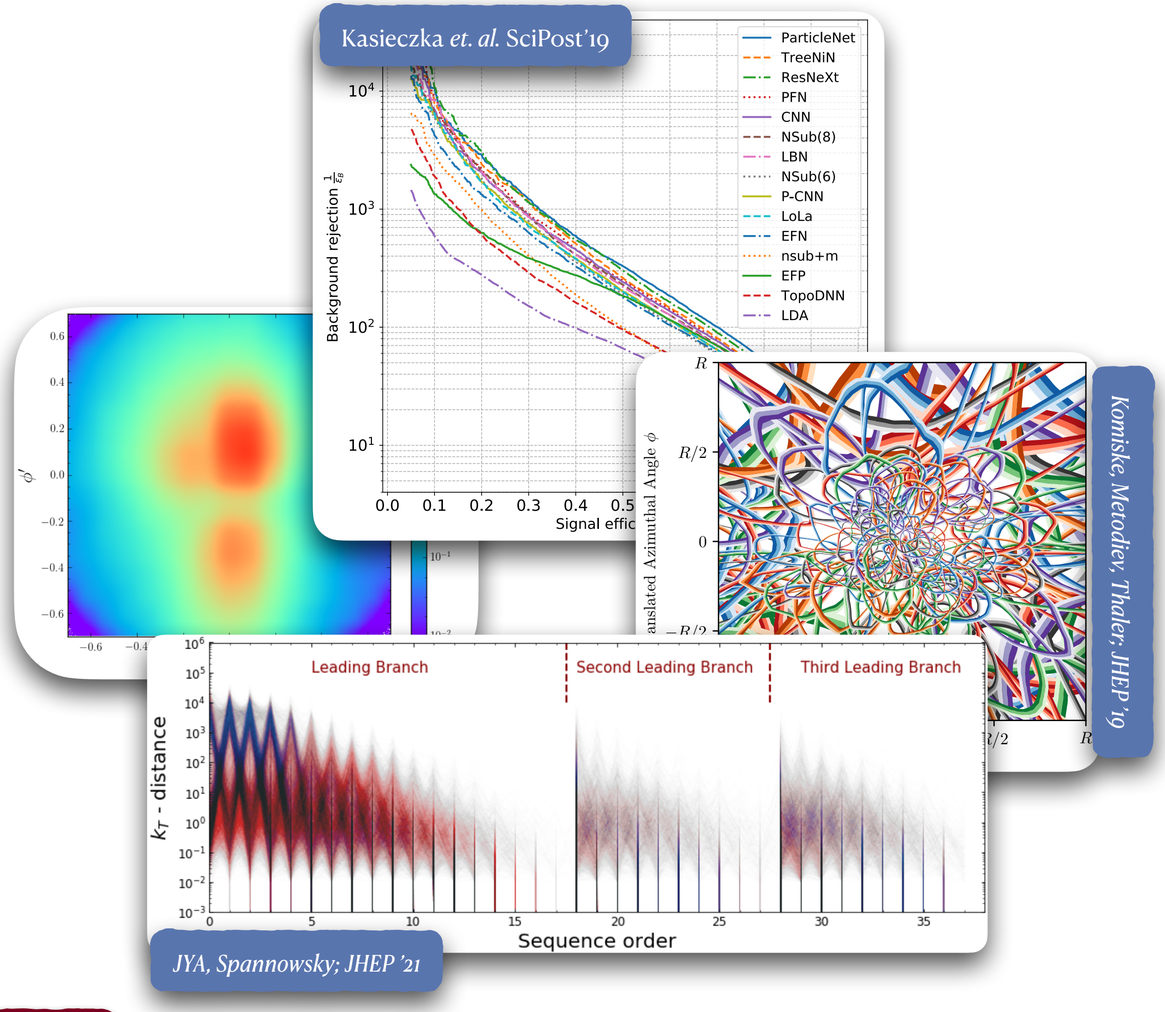
MCnet

IP3

Sales pitch of the talk!

- We more or less know how to get a well-performing Neural Network to classify jets, LHC events, even cats and dogs...
- What we don't know is what this network learns.
- Can we use Quantum Mechanics to have more insight into the learning process?
  - ✦ What has a model learned?
  - ✦ What is learning?
  - ✦ How do we develop "insightful" algorithms?
  - ✦ How to perform this on a Quantum device?

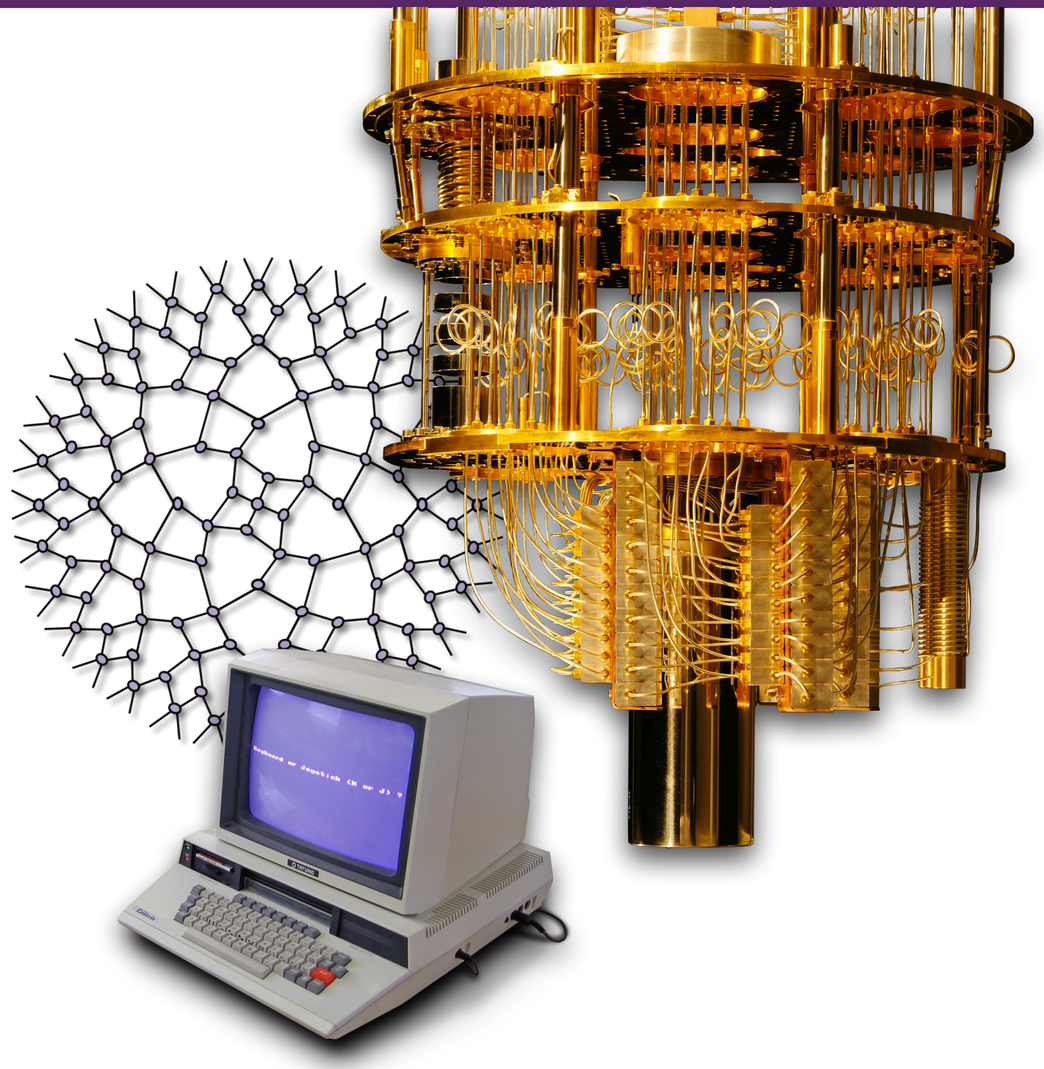
❖ All comes together with Tensor Networks!



See Jeff Byers' talk from yesterday

Outline

- Introduction
  - Representing a problem as a quantum many-body system
- Hello world of HEP-ML: Top Tagging
- Quantum Machine Learning on a Quantum device
- Beyond: Learning probability distributions with quantum-probabilistic hybrid learning (preliminary)
- Conclusion



Introduction

Tensor Networks: Origins

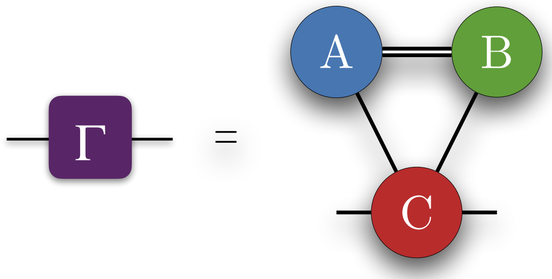
Tensor Diagram Notation

Scalar — Matrix, Λ^{ij}

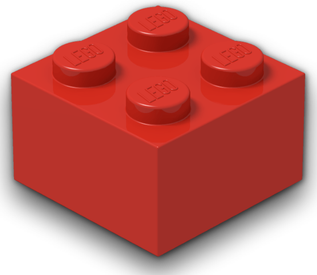
Vector, v^i — Rank 3 Tensor, Γ^{ijk}

$$|\Psi\rangle = \sum_{\phi_1,\ldots,\phi_n=0} \mathscr{W}_{\phi_1\ldots\phi_n} |\phi_1\rangle \otimes |\phi_2\rangle \otimes \ldots \otimes |\phi_n\rangle$$

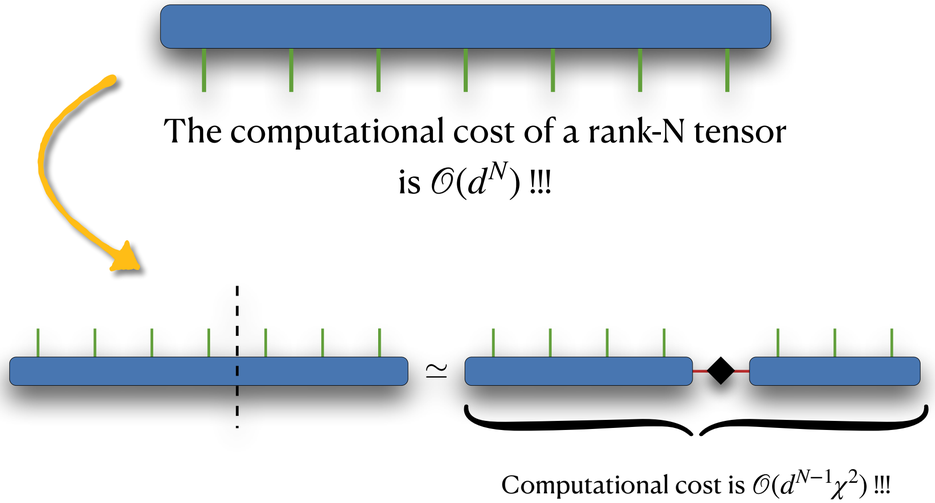
$$\forall |\phi_i\rangle \in \mathscr{H}^{\otimes 2^N} \quad \rightarrow \quad |\phi_i\rangle \in \{|\uparrow\rangle, |\downarrow\rangle\}$$



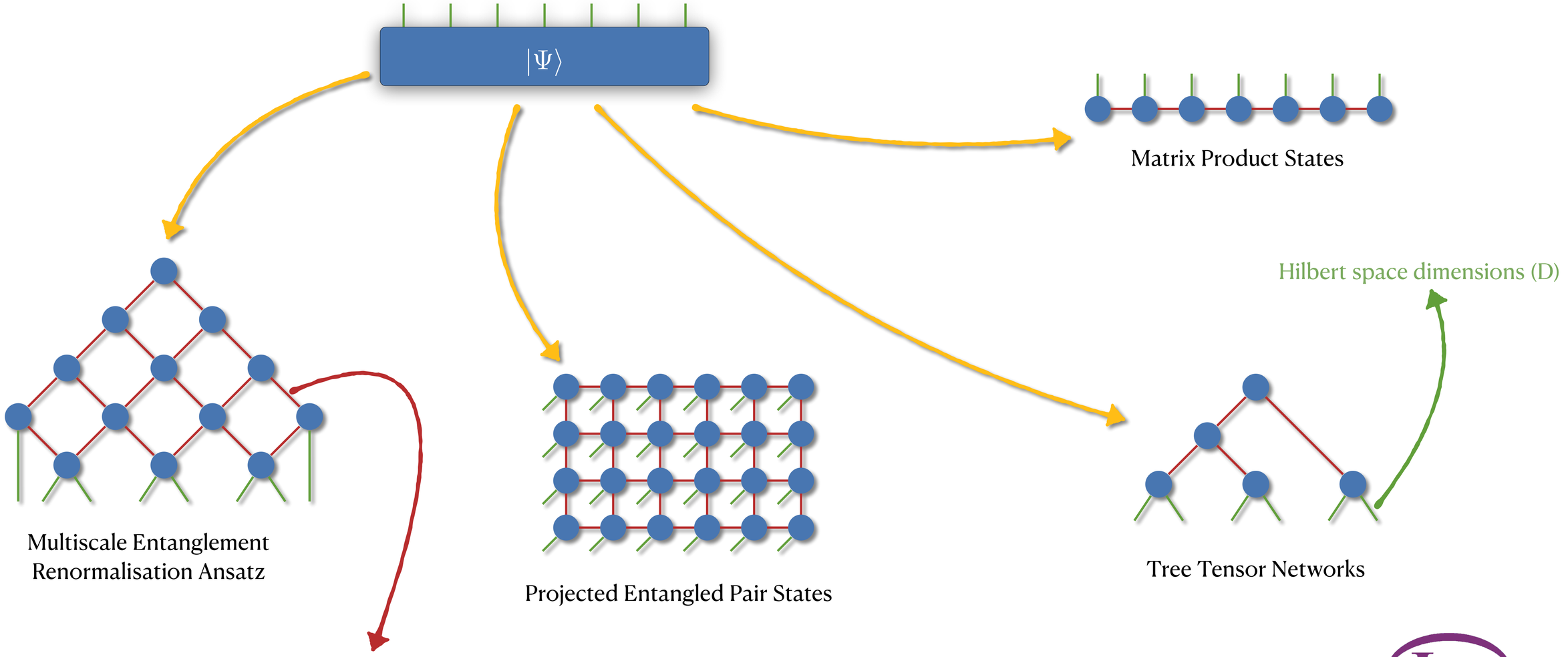
$$\Gamma^{ij} = A^{klm} B^{n}{}_{lm} C^{ij}{}_{kn}$$



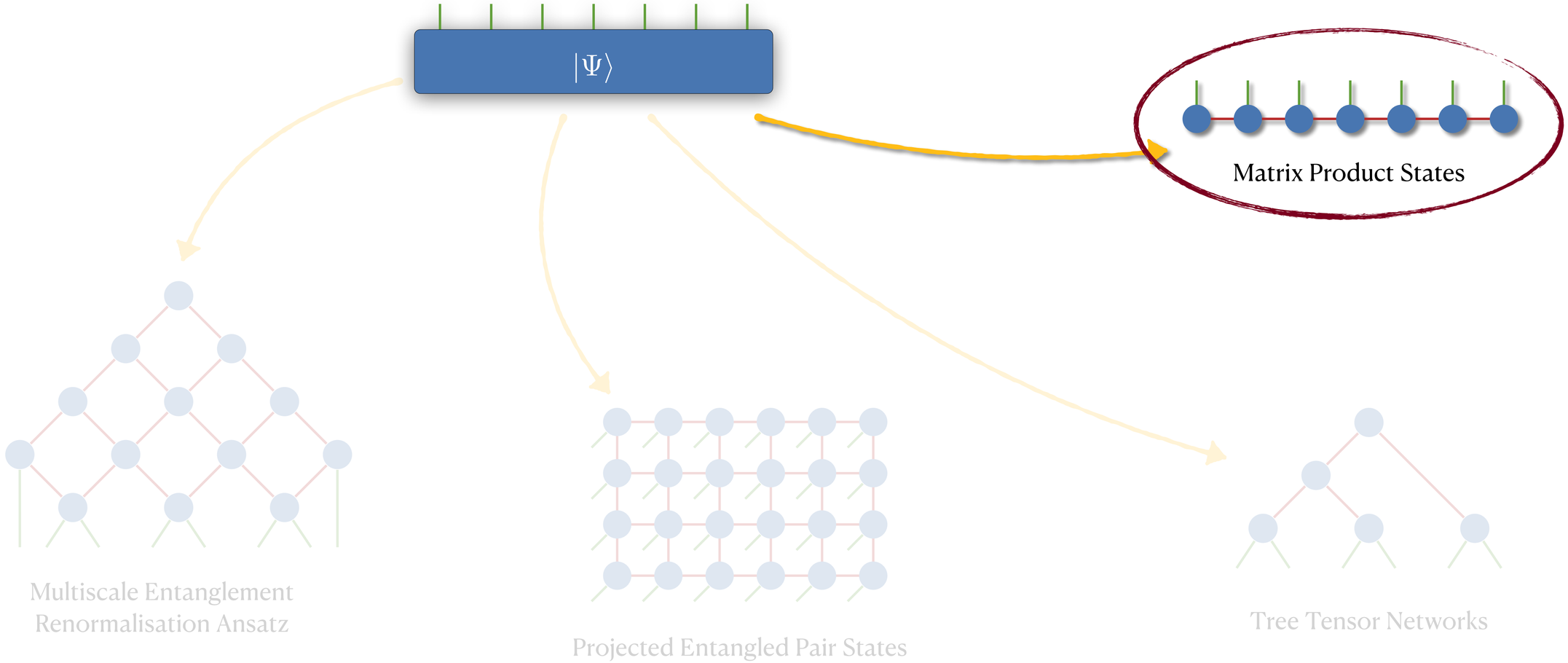
$$= L^{ijkl}{}_{\mu\nu\rho\theta}$$



Types of Tensor Networks (some of them)



Types of Tensor Networks (some of them)



Matrix Product States for Classification

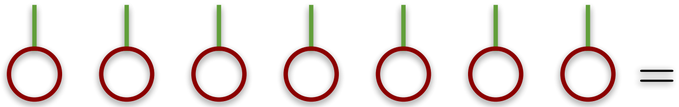
Sub-Outline

- How to embed the data?
- How to form a network?
- How to train the network?

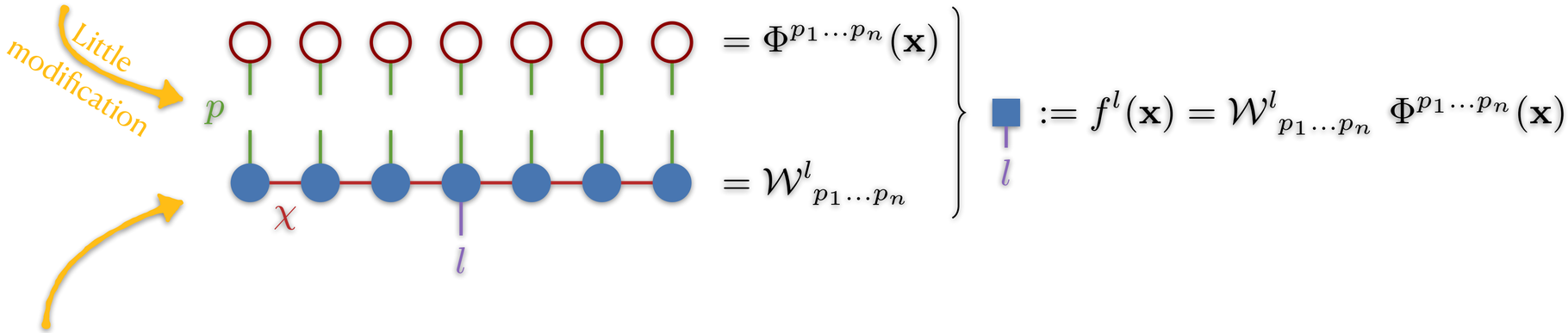
Data Embedding

$$\mathbf{X} = \{x_1, x_2, \cdots, x_n\} \in \mathbb{R} \quad , \quad \phi(x) := \forall x \in \mathbb{R} \rightarrow \mathbb{C}^m$$

$$\Phi^{p_1 \cdots p_n}(\mathbf{x}) = \bigotimes_{p_i=0}^{N} \phi^{p_i}(x_i) \qquad \phi^{p_i}(x_i) = \sum_{j=0}^{m-1} \alpha_j |j\rangle$$



$$|\Psi\rangle = \sum_{p_1,\ldots,p_n=0} \mathscr{W}_{p_1 \ldots p_n} |p_1\rangle \otimes |p_2\rangle \otimes \ldots \otimes |p_n\rangle$$

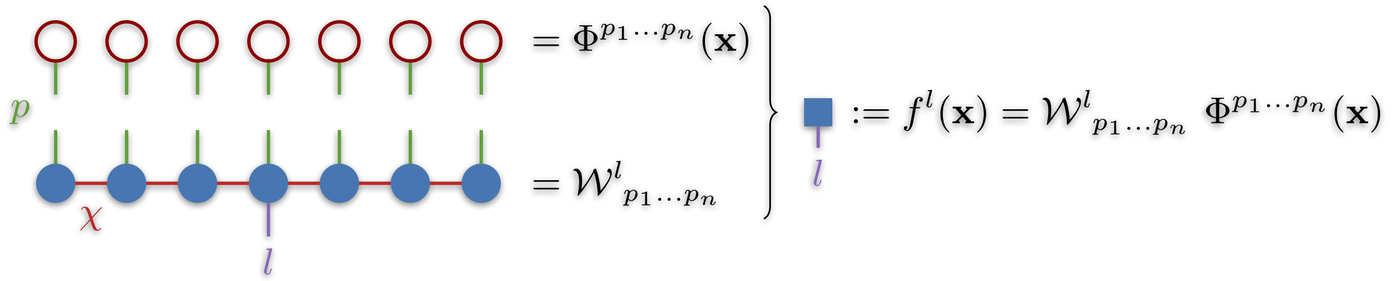


Trainable tensors: $\mathscr{W}(\theta_i)$

Matrix Product States for Classification

Sub-Outline

- How to embed the data?
- How to form a network?
- How to train the network?



$$p(x^{(i)};\theta) = \left| f^l(x^{(i)}) \right|^2$$

No activation function!!
Everything is linear!!

TNs intrinsically have
"topological nonlinearity"

Matrix Product States for Classification

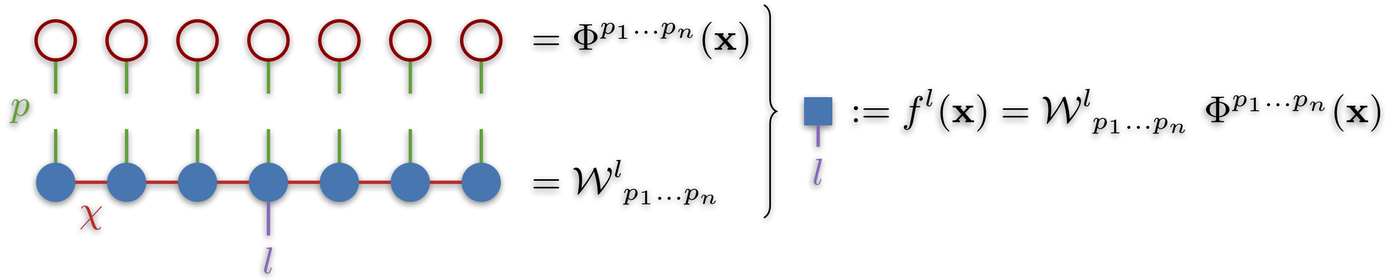
TNs vs Kernel Methods

- TNs can be considered as a group of non-linear kernel methods up to an approximation, depending on the size of the tensors. Novikov, Trofimov, Oseledets; arXiv: 1605.03795
- The cost for TNs scales at most linearly in the number of training examples, where quadratically for most kernel methods. Lu, May, et. al. arXiv: 1411.4000
- Representing data as TN gives them a structure for enhanced interpretability.
- TN-based optimisation algorithms are adaptive and automatically select the minimum number of parameters needed for the optimal representation. Stoudenmire, Schwab; arXiv: 1605.05775

Matrix Product States for Classification

Sub-Outline

- How to embed the data?
- How to form a network?
- How to train the network?



$$p(x^{(i)};\theta) = \left| f^l(x^{(i)}) \right|^2$$

No activation function!!
Everything is linear!!

TNs intrinsically have
"topological nonlinearity"

$$\mathscr{L} = \frac{1}{N} \sum_{x \in \mathbf{x}^N} q^{\text{truth}} \log\left(p(x^{(i)};\theta)\right)$$

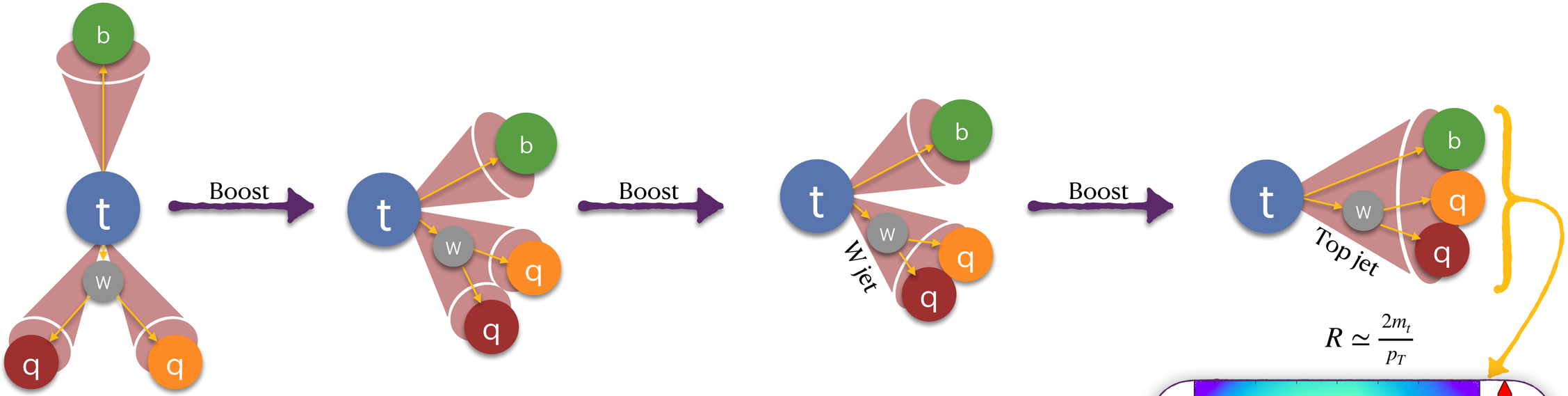
Or anything else you like to minimise...

$$\arg\min_{\theta_i \in \mathscr{W}} \mathscr{L}\left(q(x^{(i)}), p(x^{(i)};\theta)\right)$$

Traditionally NNs are trained with SGD, but MPS is trained with Density Matrix Renormalisation Group Algorithm

Hello World of HEP-ML: Top Tagging

Why Top Quarks?



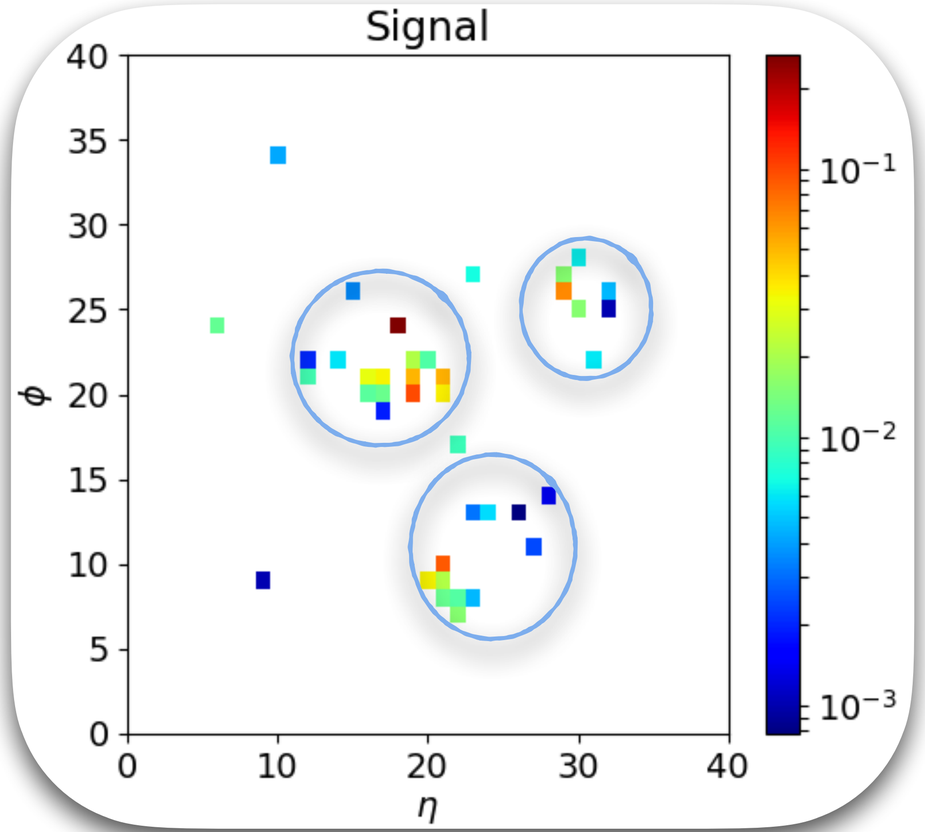
- With the increased boost factor, jets (top decay products) are getting more collimated.
- Hadronic top tagging tools: Mass grooming and filtering, Pruning, Trimming, Soft Drop Tagger, Mass Drop Tagger, HEPTopTagger, Machine Learning

JYA, Spannowsky; JHEP '21

Why TNs "might" perform well in classification tasks?

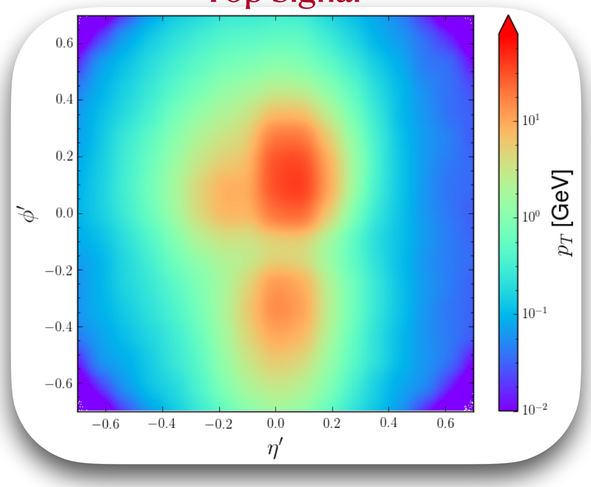
- The range of a node in a Tensor Network is bounded by its bond dimension.
- Tensor Networks can capture local "anomalies".
- We are dealing with sparse, locally correlated calorimeter pixels.

Kasieczka *et. al.* SciPost'19

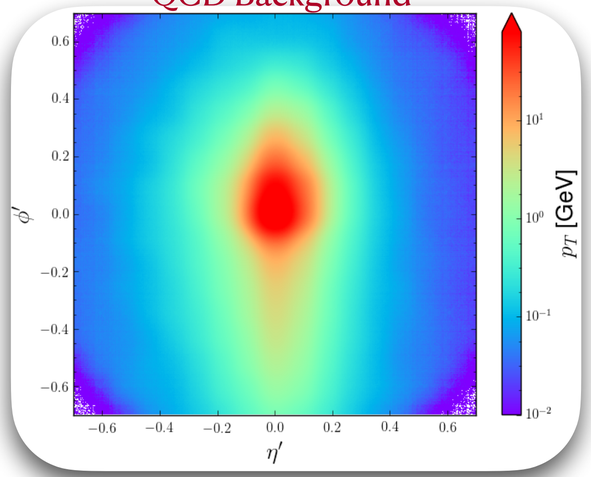


Top Tagging through MPS

Top Signal



QCD Background



Mean of 10K events

Data from: Kasieczka *et. al.* SciPost'19

- Leading FatJet Definition: anti-k_T algorithm with $R = 0.8, p_T \in [550,\ 650]$ GeV, $|\eta| < 2$
- Parton matching with $\Delta R(j, t_{truth}) < 0.8$
- Jets are centred with respect to p_T weighted centroid where jet vector is at $(\phi, \eta) = (0,0)$
- Principal axis has been rotated to $+\eta$ direction
- Energy deposits has been divided into 37×37 pixels which corresponds to η & $\phi \in [-1.5,\ 1.5]$.
- Image has been flipped to place the most energetic quadrant to the top right corner.

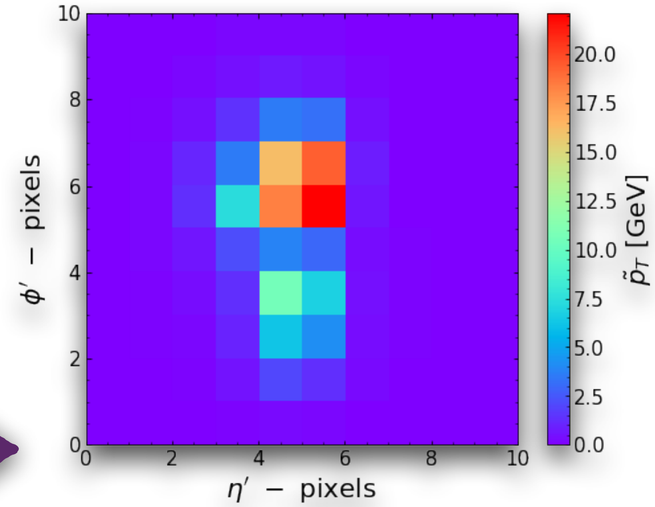
Similar preprocess, based on CNN:

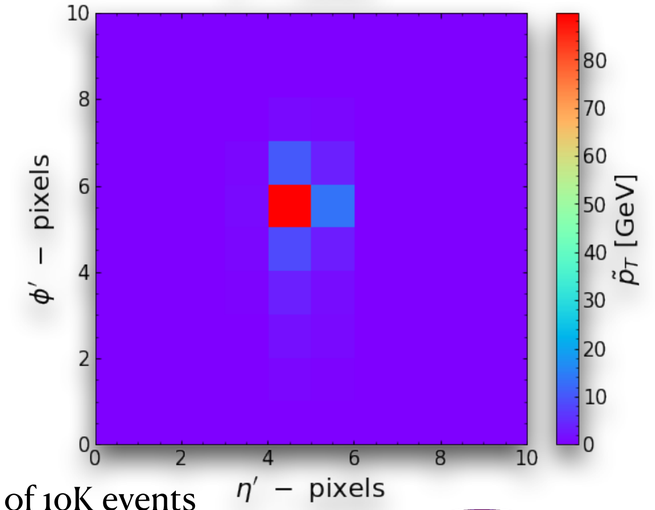
Kasieczka, Plehn, Russell, Schell; JHEP '17

Macaluso, Shih; JHEP '18

JYA, Spannowsky; JHEP '21

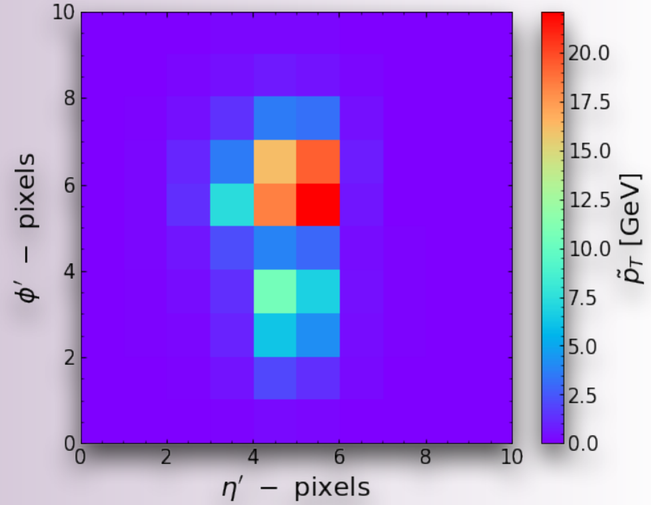
Crop & downsample

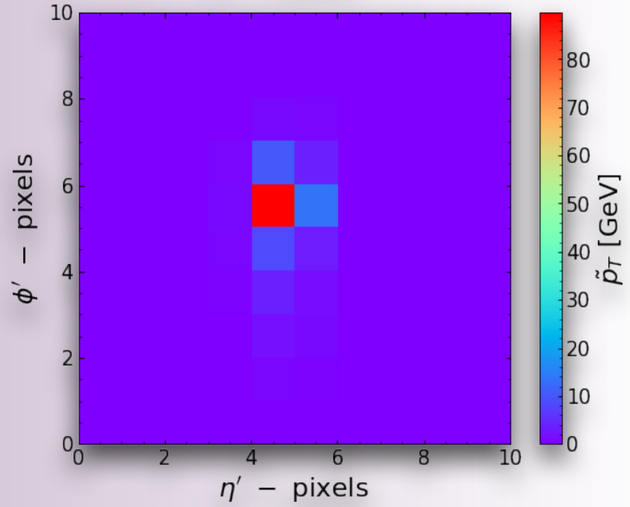




Mean of 10K events

Top Tagging through MPS





η - based ordering

$\tilde{p}_T^1$	$\tilde{p}_T^2$	$\tilde{p}_T^3$	$\tilde{p}_T^4$	$\tilde{p}_T^5$
$\tilde{p}_T^{10}$	$\tilde{p}_T^9$	$\tilde{p}_T^8$	$\tilde{p}_T^7$	$\tilde{p}_T^6$
$\tilde{p}_T^{11}$	$\tilde{p}_T^{12}$	$\tilde{p}_T^{13}$	$\tilde{p}_T^{14}$	$\tilde{p}_T^{15}$
$\tilde{p}_T^{20}$	$\tilde{p}_T^{19}$	$\tilde{p}_T^{18}$	$\tilde{p}_T^{17}$	$\tilde{p}_T^{16}$
$\tilde{p}_T^{21}$	$\tilde{p}_T^{22}$	$\tilde{p}_T^{23}$	$\tilde{p}_T^{24}$	$\tilde{p}_T^{25}$

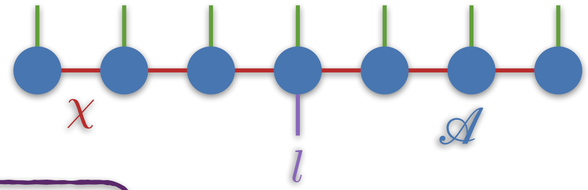
ϕ' – pixels

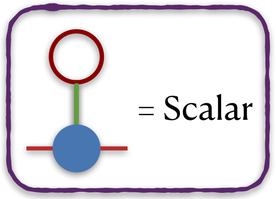
η' – pixels

Data Embedding

$$\Phi^{p_1\cdots p_n}(\mathbf{x}) = \bigotimes_i^N \phi^{p_i}(\tilde{p}_T^i)$$

$$\phi^{p_i}(\tilde{p}_T^i) := \forall \tilde{p}_T \in \mathbb{R} \to \mathbb{C}^{10} = \sqrt{\binom{D-1}{d_i-1}} \cos^{D-d_i}\left(\tilde{p}_T^i \frac{\pi}{2}\right) \sin^{d_i-1}\left(\tilde{p}_T^i \frac{\pi}{2}\right)$$



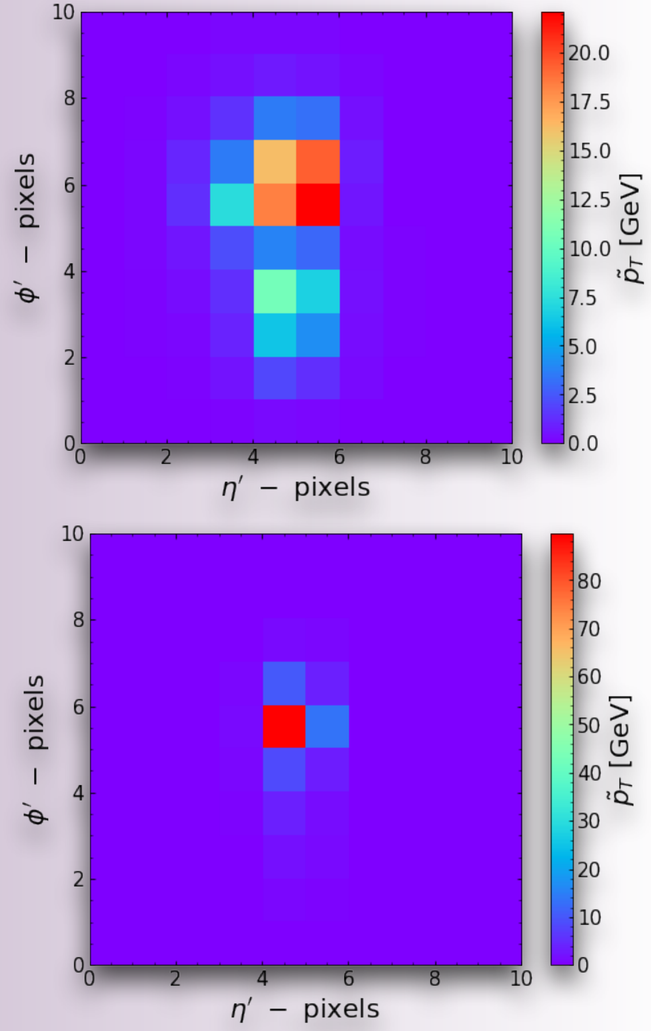


Network Initial Condition

- Each feature assumed to be uncorrelated

$$\mathscr{W}^l_{p_1\cdots p_n} = \mathscr{A}^{l,(1)}_{p_1} \otimes \mathscr{A}^{(2)}_{p_2} \otimes \cdots \otimes \mathscr{A}^{(n)}_{p_n}$$

Top Tagging through MPS

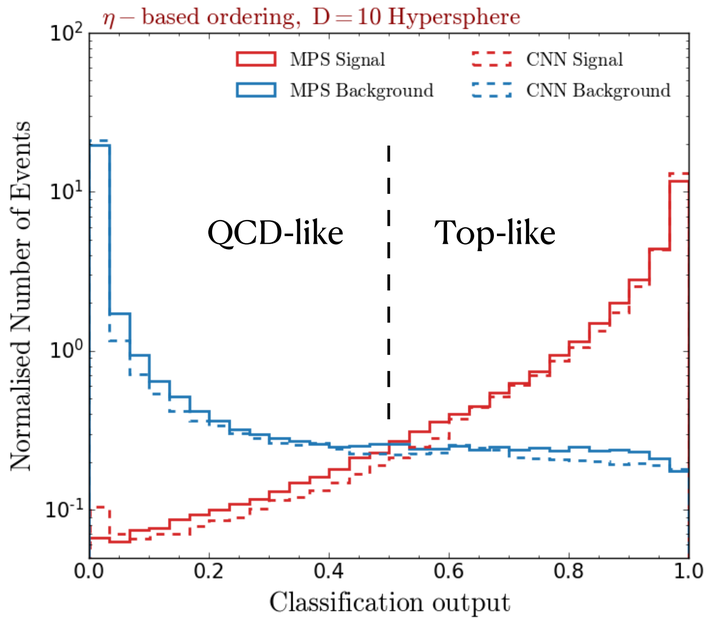


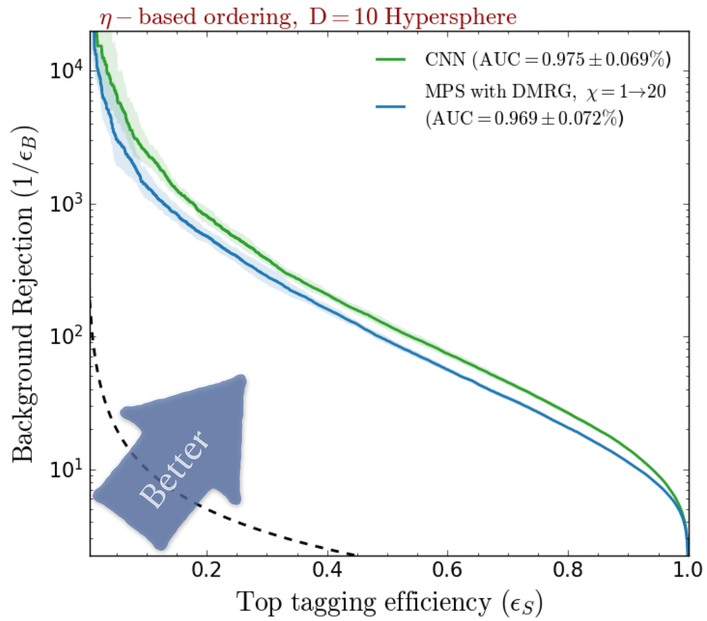
Assumptions & Requirements

- No prior entanglement/correlation between pixels
- Network is a Born Machine → square of the wave-function gives the probability of the classification.
- Maximum bond dimension that network can get is 20.

CNN architecture from:

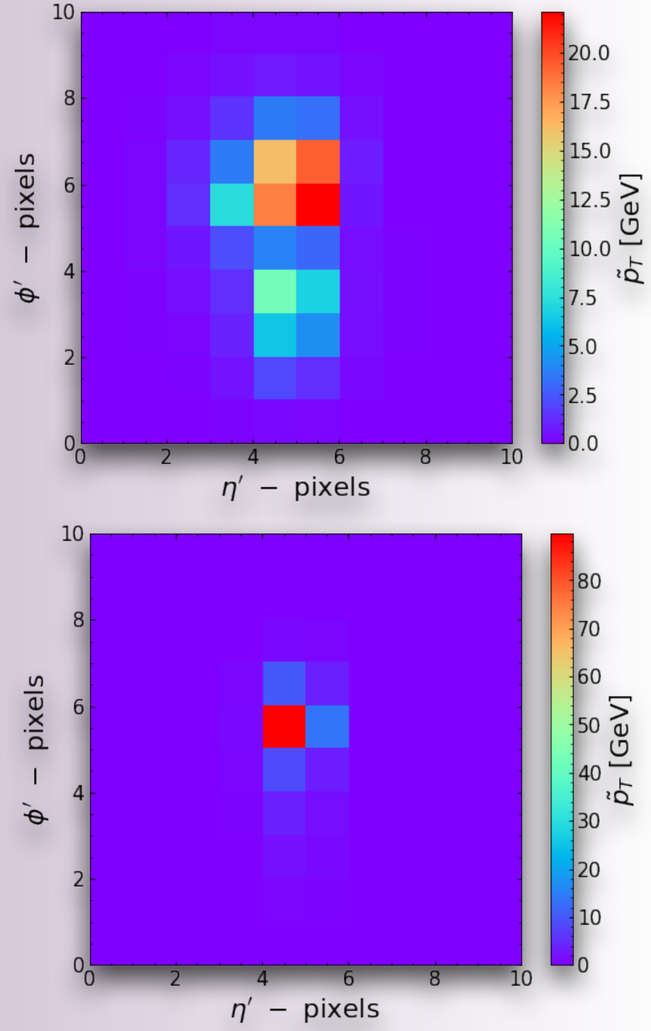
JYA, Spannowsky; JHEP '21





Q: If we have a trained network without data, what can we say about the data?

Top Tagging through MPS

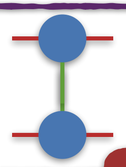


η - based ordering

$\tilde{p}_T^1$	$\tilde{p}_T^2$	$\tilde{p}_T^3$	$\tilde{p}_T^4$	$\tilde{p}_T^5$
$\tilde{p}_T^{10}$	$\tilde{p}_T^9$	$\tilde{p}_T^8$	$\tilde{p}_T^7$	$\tilde{p}_T^6$
$\tilde{p}_T^{11}$	$\tilde{p}_T^{12}$	$\tilde{p}_T^{13}$	$\tilde{p}_T^{14}$	$\tilde{p}_T^{15}$
$\tilde{p}_T^{20}$	$\tilde{p}_T^{19}$	$\tilde{p}_T^{18}$	$\tilde{p}_T^{17}$	$\tilde{p}_T^{16}$
$\tilde{p}_T^{21}$	$\tilde{p}_T^{22}$	$\tilde{p}_T^{23}$	$\tilde{p}_T^{24}$	$\tilde{p}_T^{25}$

ϕ' – pixels

η' – pixels

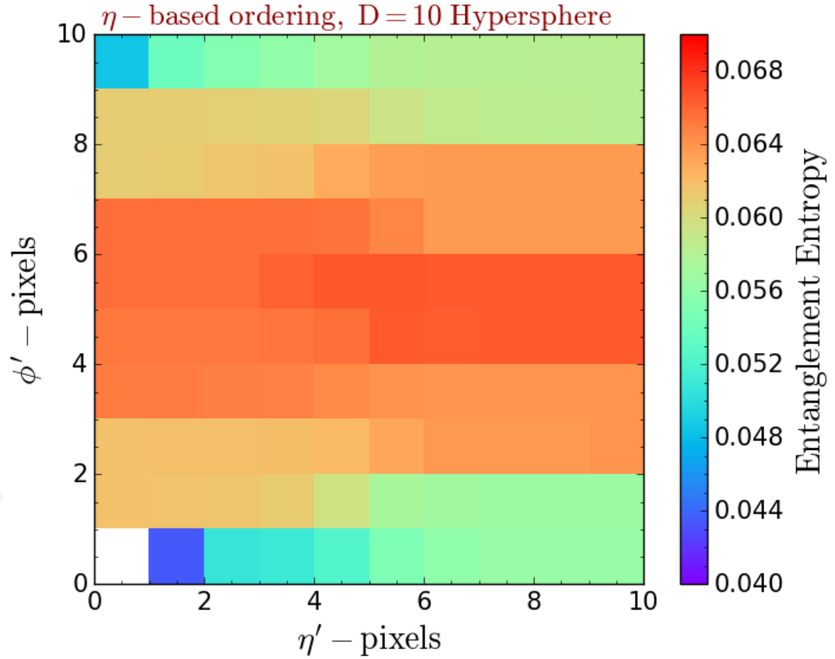
$\rho_i =$ 

Reduced density matrix

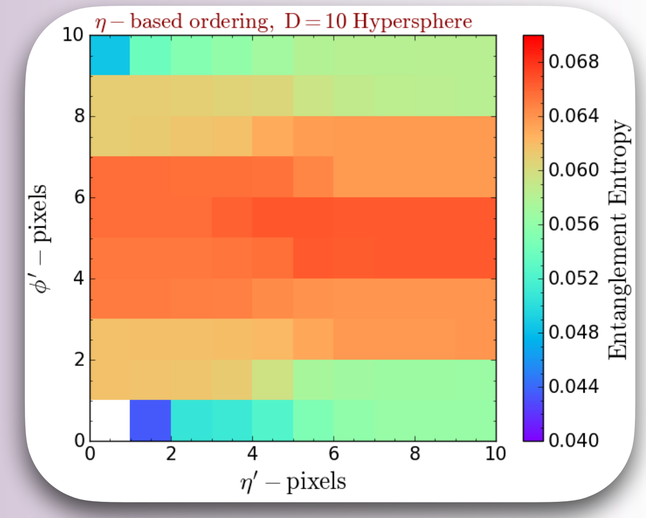
Entanglement Entropy

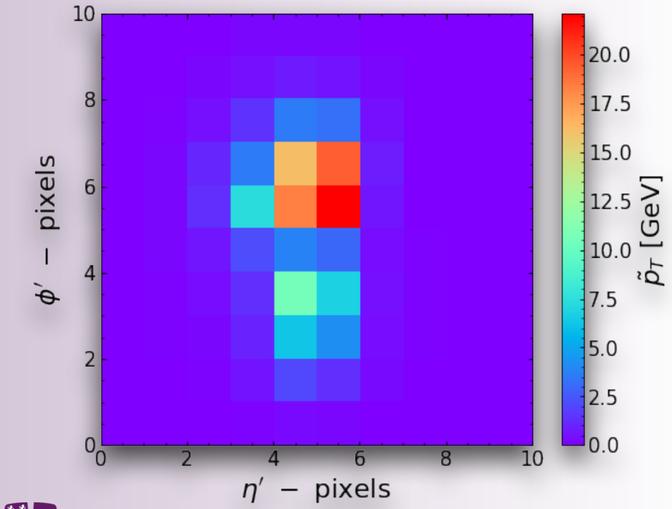
$$\mathcal{S}(\rho_i) = -\operatorname{Tr}[\rho_i \log \rho_i] \quad ; \quad \rho_i := |\phi_i\rangle\langle\phi_i|$$

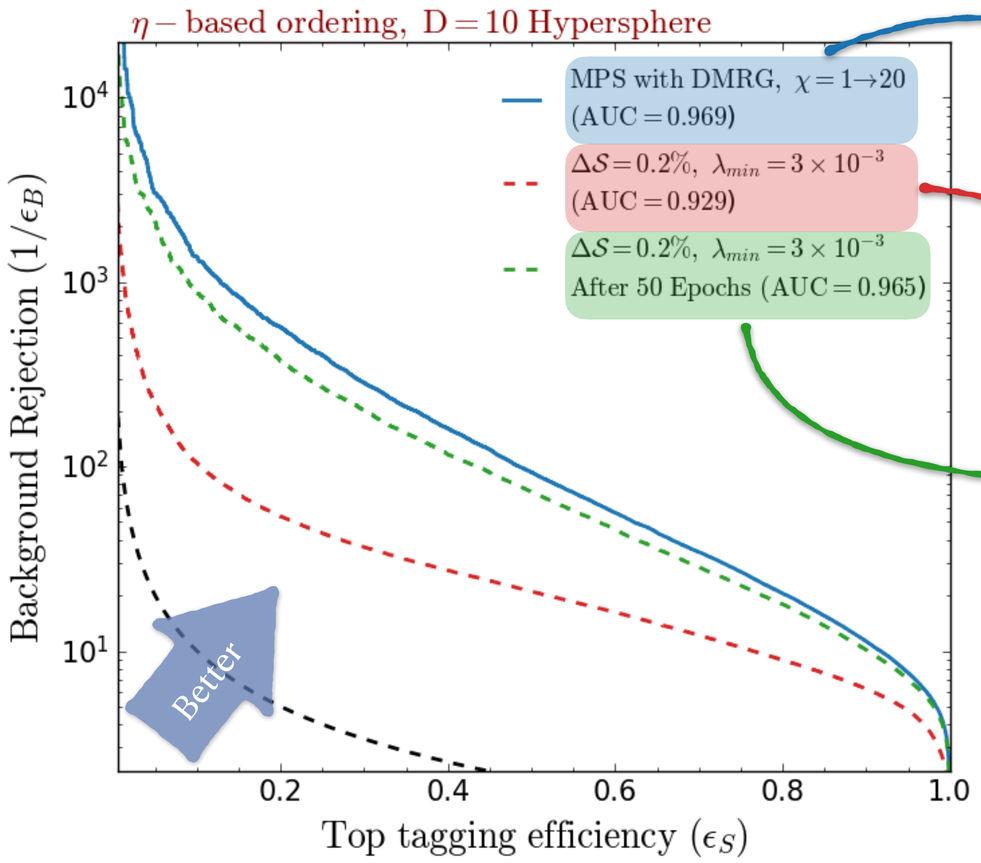
$\mathcal{S}(\rho)$ is a quantifiable measure to understand which pixels are not crucial for the process without the data!!!



Top Tagging through MPS







100 pixels, 390500 trainable parameters

54 pixels, 43410 trainable parameters

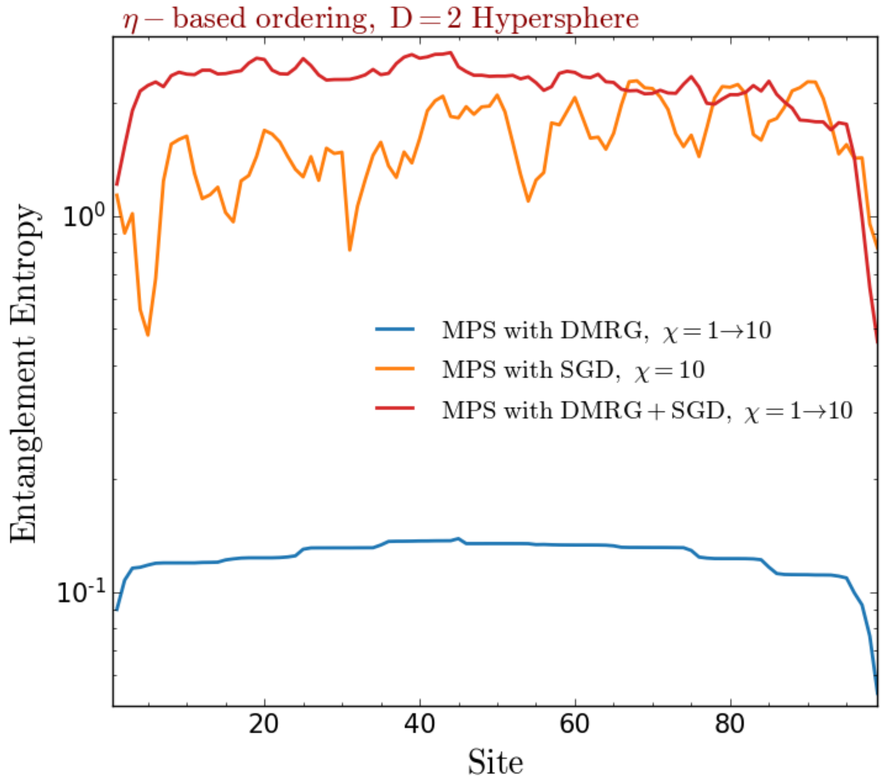
54 pixels, 34160 trainable parameters

91% Reduction

Almost half of the data did not contribute to the network's decision!

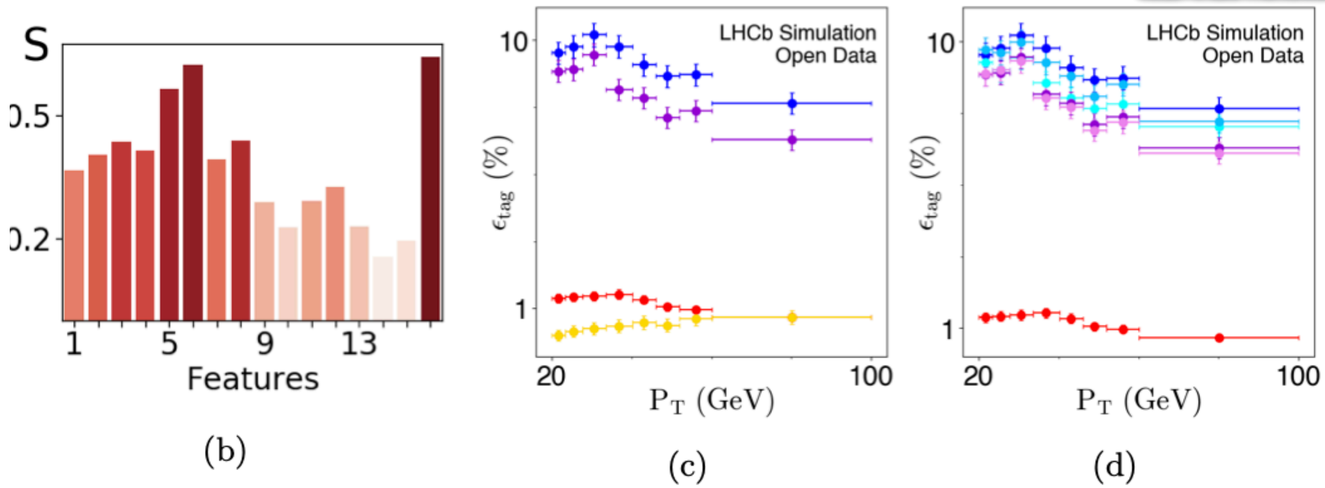
Why finding a quantitative measure is important?

- Understanding the network gives the ability to build better training algorithms.
- Scientific data is largely sparse; if we know where the information comes from, we can get rid of large amounts of data.
- Suppress the noise (and for pile-up mitigation to be confirmed)!



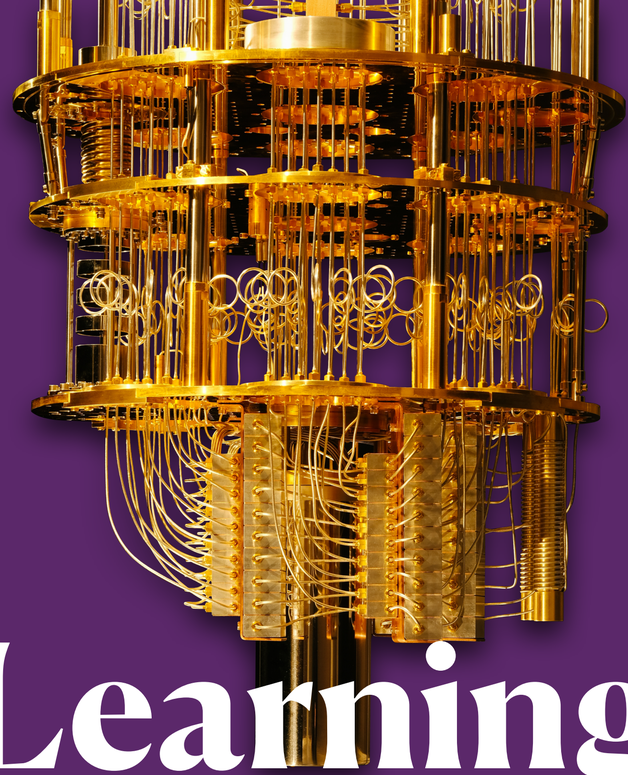
Why finding a quantitative measure is important?

Felser, Trenti, Sestini, Gianelle, Zuliani, Lucchessi, Montangero; npj 2021



(b) (c) (d)

FIG. 3: Exploiting the information provided by the learned TTN classifier: (a) Correlations between the 16 input features (blue for anti-correlated, white for uncorrelated, red for correlated). The numbers indicate q, p_T^{rel}, ΔR of the muon (1-3), kaon (4-6), pion (7-9), electron (10-12), proton (13-15) and the jet charge Q (16). (b) Entropy of each feature as the measure for the information provided for the classification. (c) Tagging power for learning on all features (blue), the best 8 proposed by QuIPS exploiting insights from (a)+(b) (magenta), the worst 8 (yellow) and the muon tagging (red). (d) Tagging power for decreasing bond dimension truncated after training: The complete model (blue shades for $\chi = 100$, $\chi = 50$, $\chi = 5$), for using the QuIPS best 8 features only (violet shades for $\chi = 16$, $\chi = 5$), and the muon tagging (red).

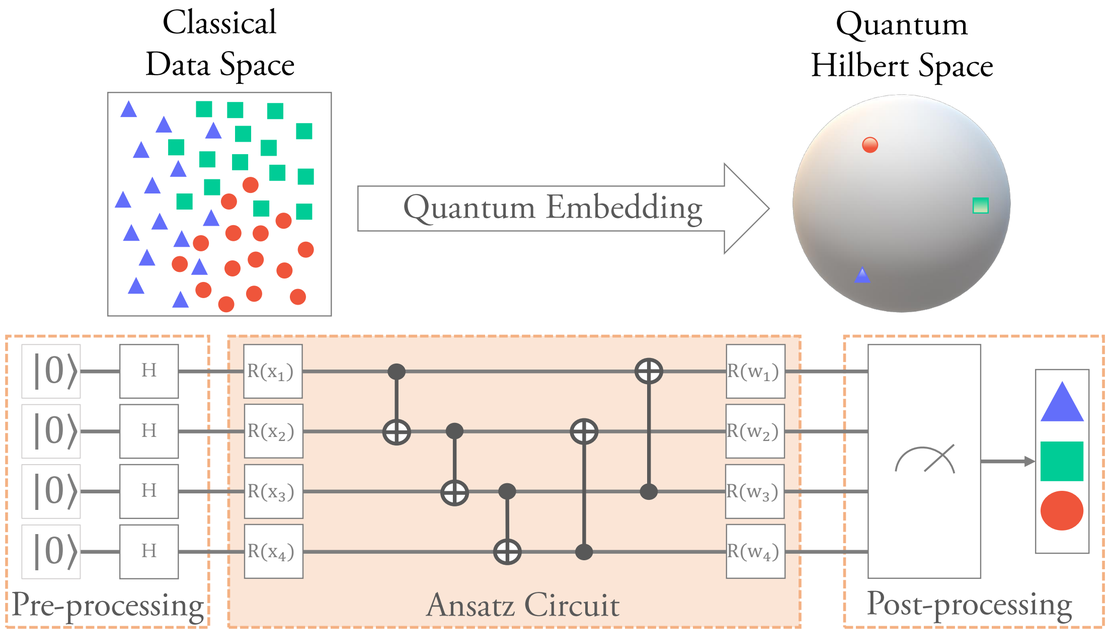


Quantum Machine Learning

Yes, up to now, it was “classical”...

Variational Quantum Circuits

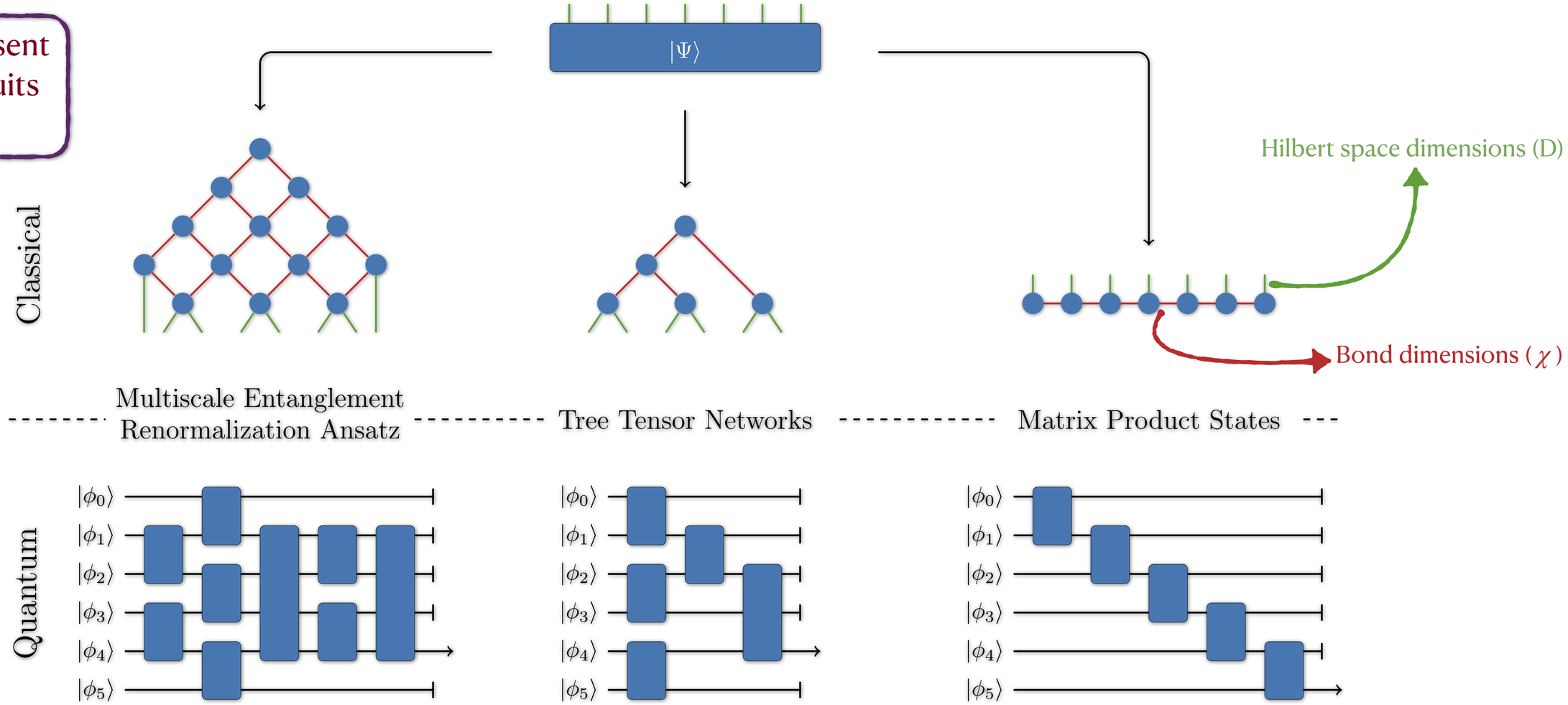
- Choose an ansatz variational quantum circuit.
- Embed the data by rotating each qubit.
- Update your ansatz with respect to the output!



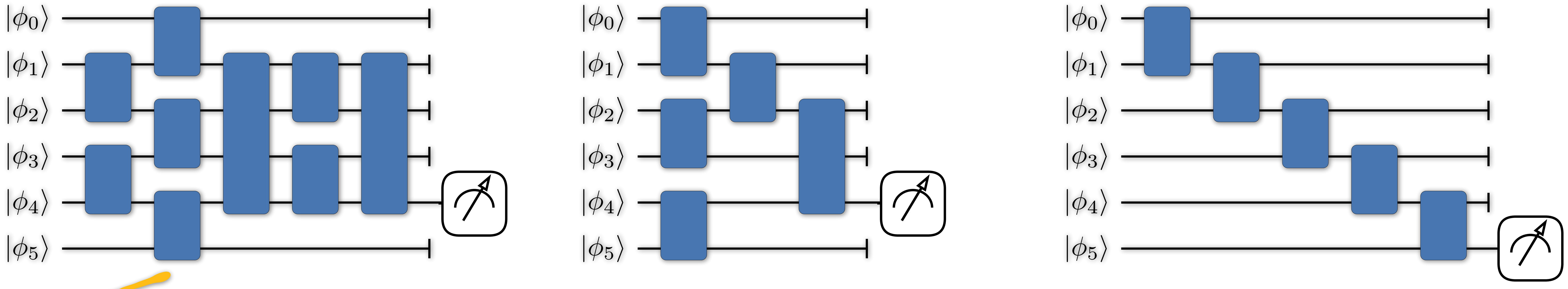
Nguyen, Chen; arXiv: 2105.11853

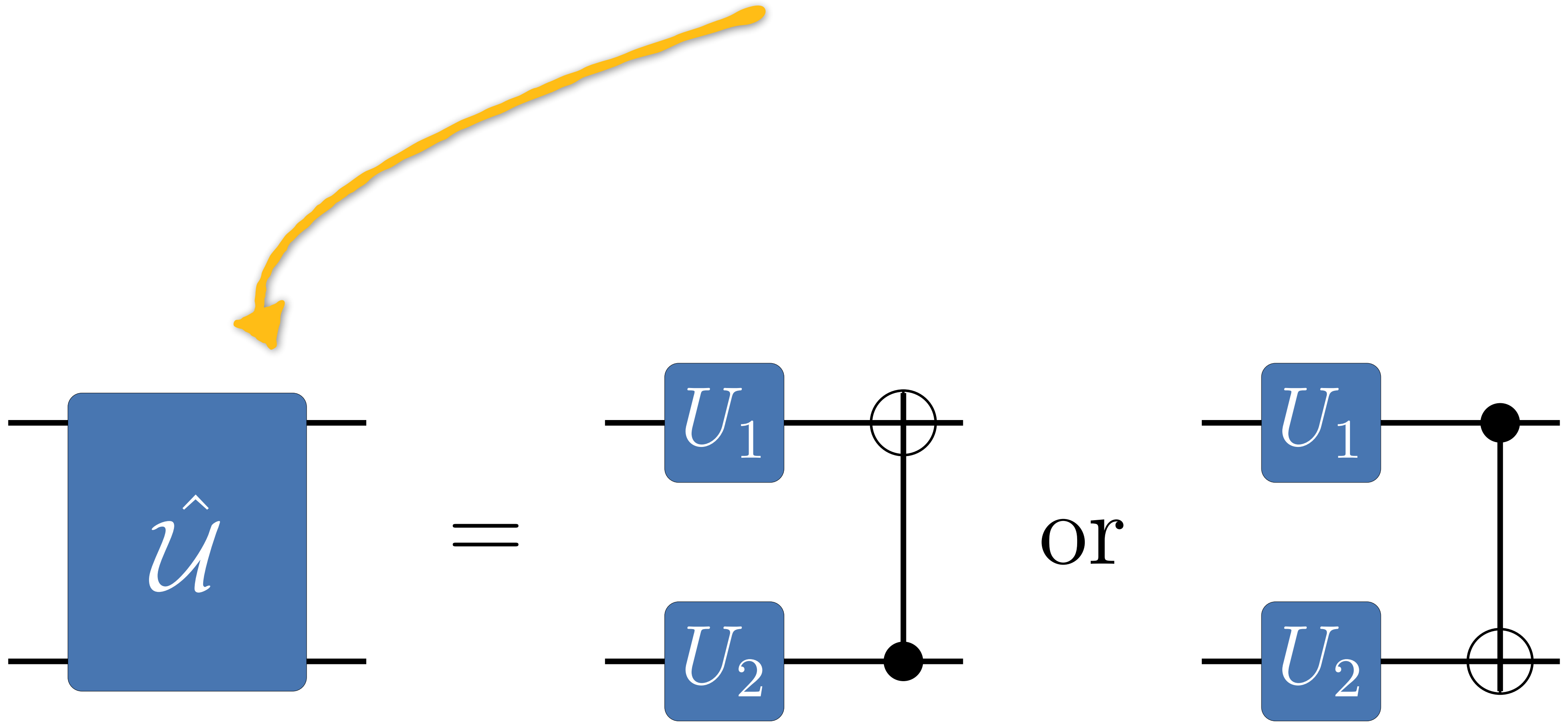
Quantum Tensor Networks

TNs can represent quantum circuits classically



Quantum Tensor Networks



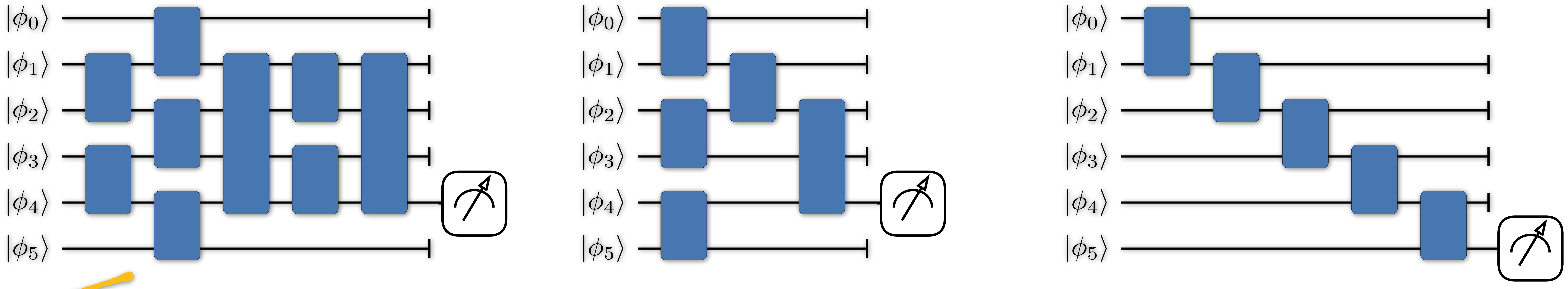


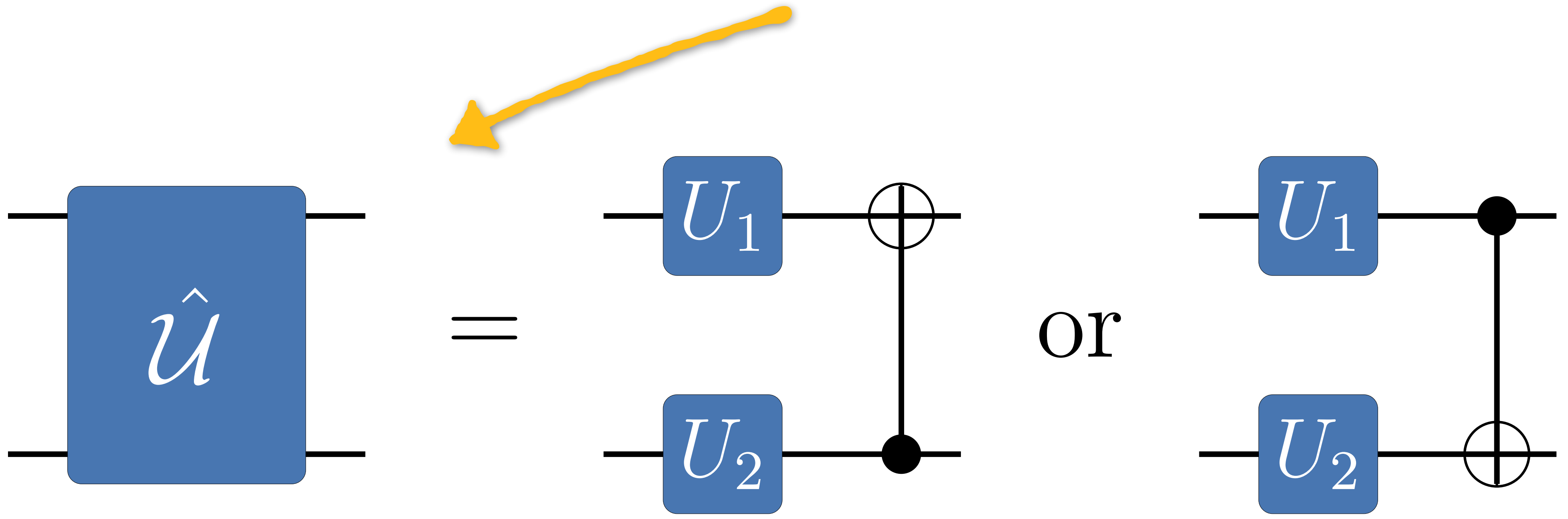
$$U(\theta,\varphi,\lambda) = \begin{pmatrix} \cos(\theta/2) & -e^{i\lambda}\sin(\theta/2) \\ e^{i\varphi}\sin(\theta/2) & e^{i(\lambda+\varphi)}\cos(\theta/2) \end{pmatrix}$$

$\varphi, \lambda = 0 \rightarrow$ Rotation around y-axis

$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Quantum Tensor Networks

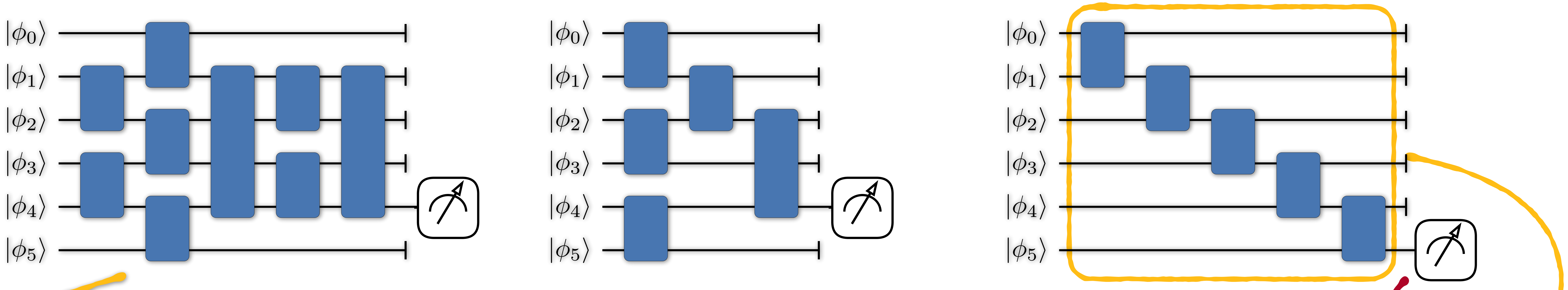


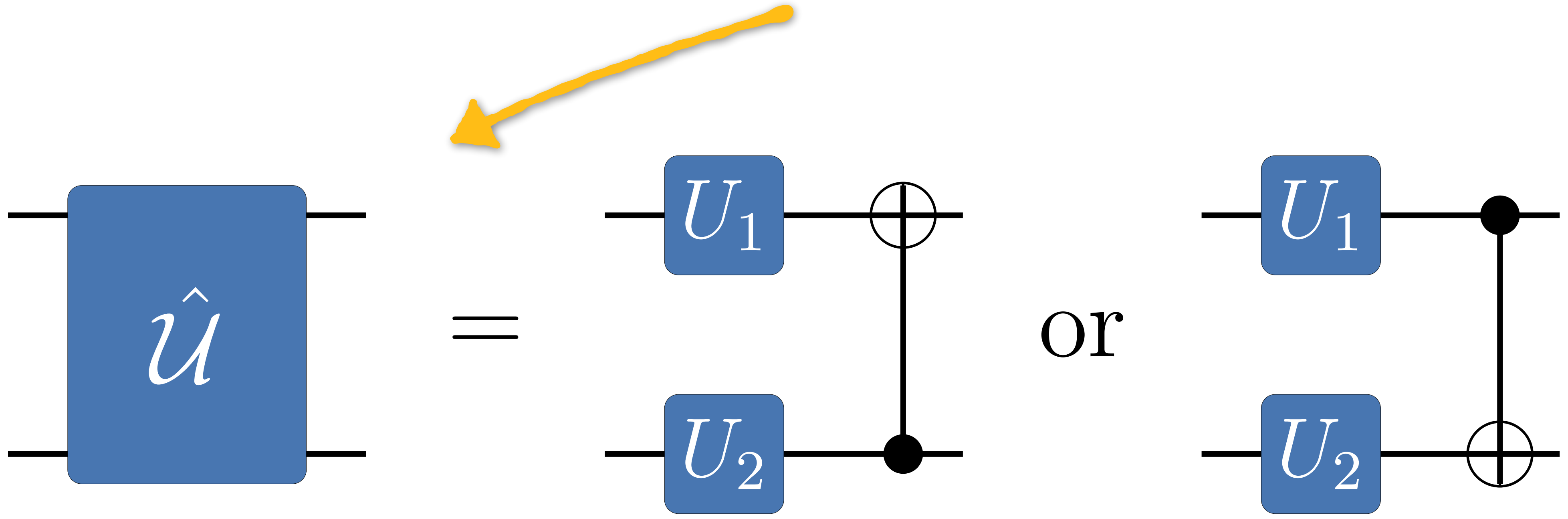


$$\mathrm{M}_\theta(\Phi^{\beta_1\cdots\beta_n}(\mathbf{x})) = \langle\Phi|\ \hat{\mathcal{U}}^\dagger_{\mathrm{QC}}(U_i(\theta_j))\ \hat{\mathrm{M}}\ \hat{\mathcal{U}}_{\mathrm{QC}}(U_i(\theta_j))\ |\Phi\rangle$$

$$p\left(\mathbf{x}^{(i)};\theta\right) = \left|\mathrm{M}_\theta\left(\Phi^{\beta_1\cdots\beta_n}\left(\mathbf{x}^{(i)}\right)\right)\right|^2$$

Quantum Tensor Networks





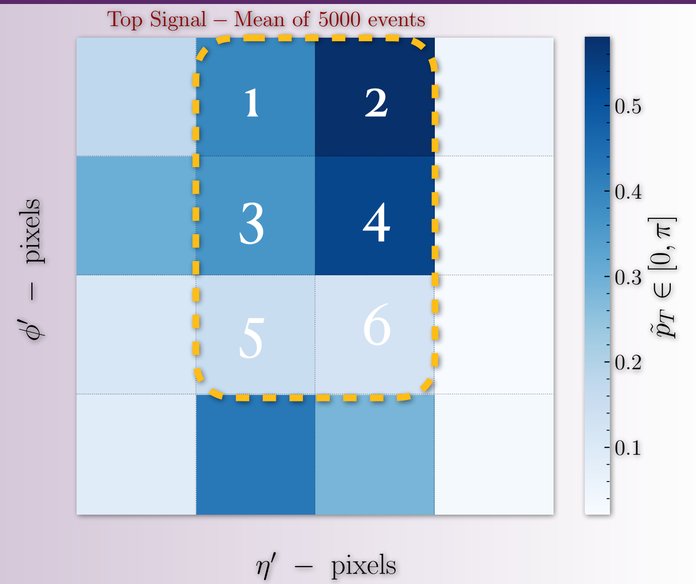
$$\mathrm{M}_\theta(\Phi^{\beta_1\cdots\beta_n}(\mathbf{x})) = \langle\Phi|\, \hat{\mathcal{U}}^\dagger_{\mathrm{QC}}(U_i(\theta_j))\, \hat{\mathrm{M}}\, \hat{\mathcal{U}}_{\mathrm{QC}}(U_i(\theta_j))\, |\Phi\rangle$$

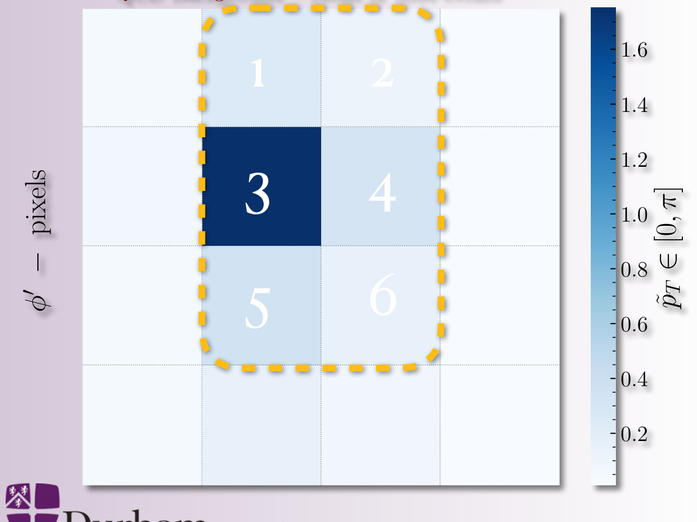
$\hat{\sigma}_z$

$$p\left(\mathbf{x}^{(i)};\theta\right) = \left|\mathrm{M}_\theta\left(\Phi^{\beta_1\cdots\beta_n}\left(\mathbf{x}^{(i)}\right)\right)\right|^2$$

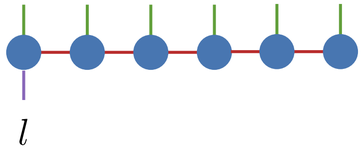
Same story: Calculate probability, minimise the objective function, update parameters...

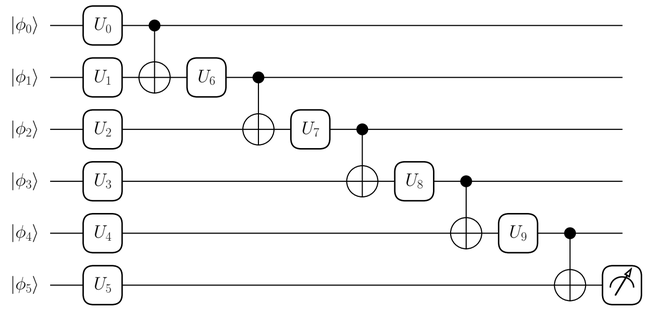
Experimenting with 6-Qubits



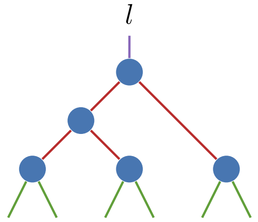


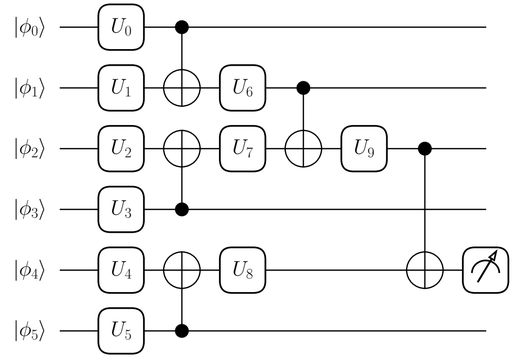
Matrix Product States



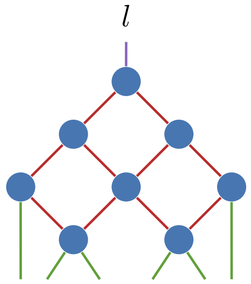


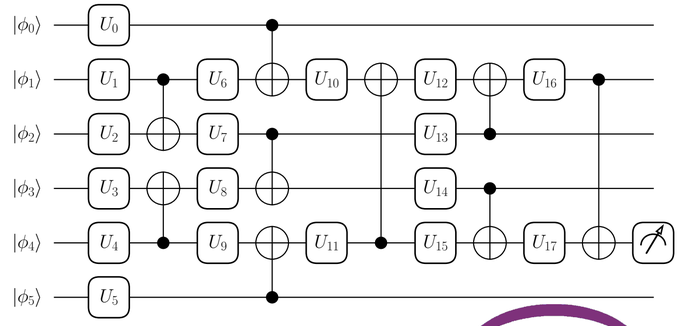
Tree Tensor Networks



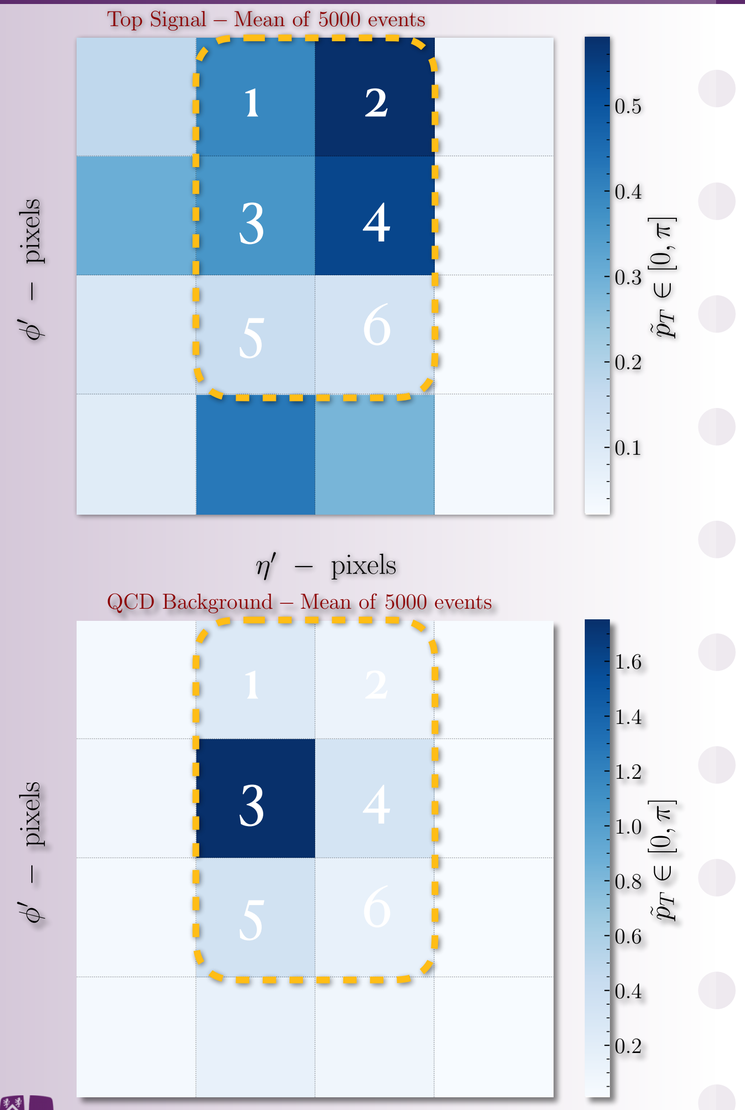


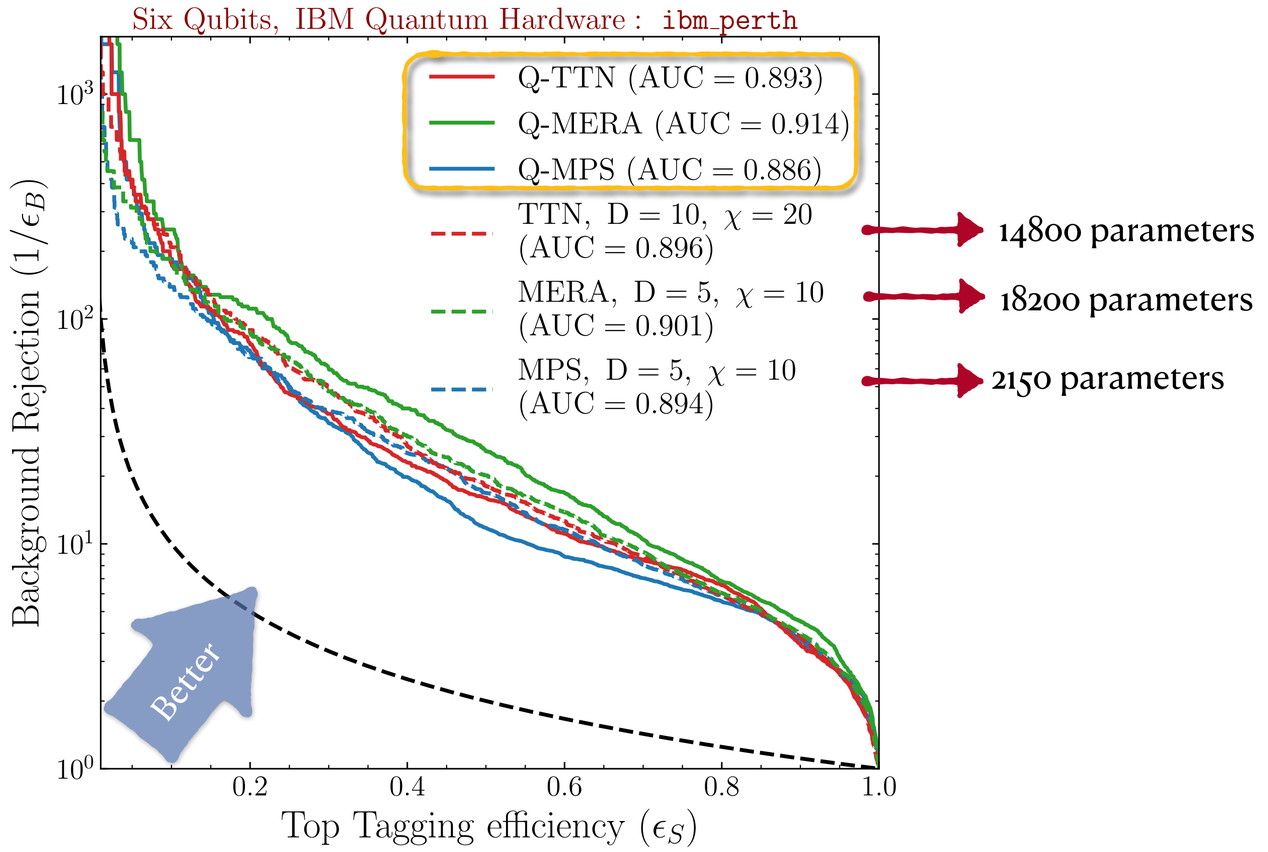
Multiscale Entanglement Renormalisation Ansatz



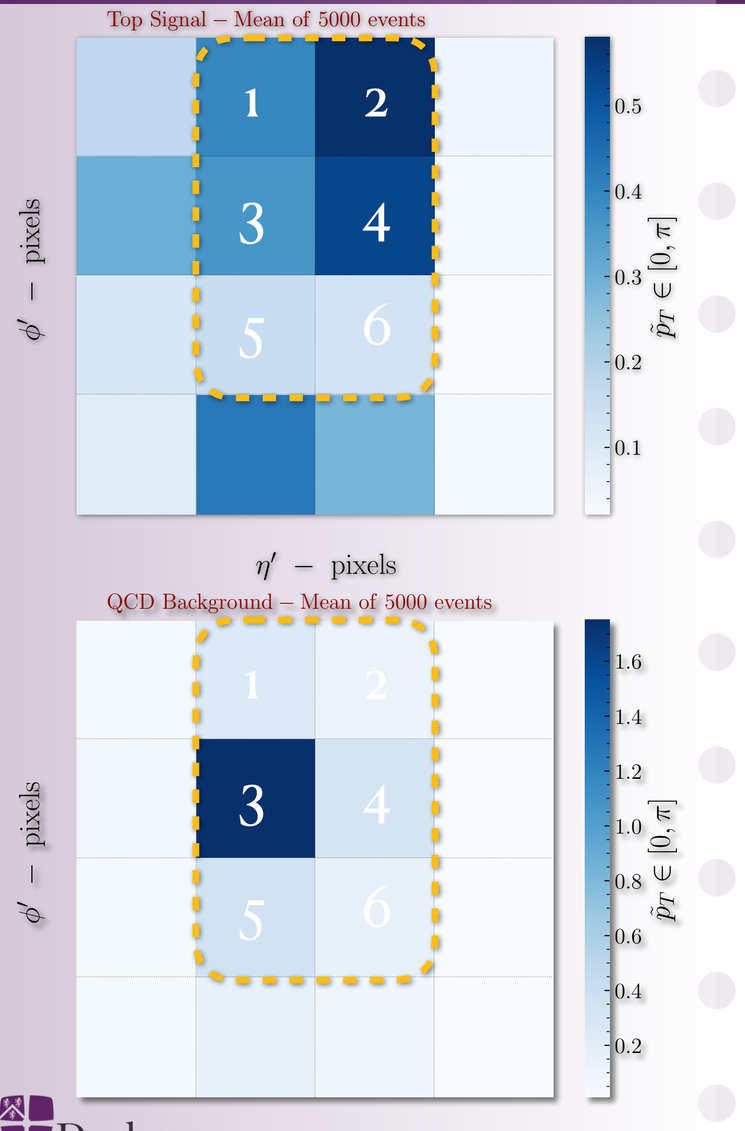


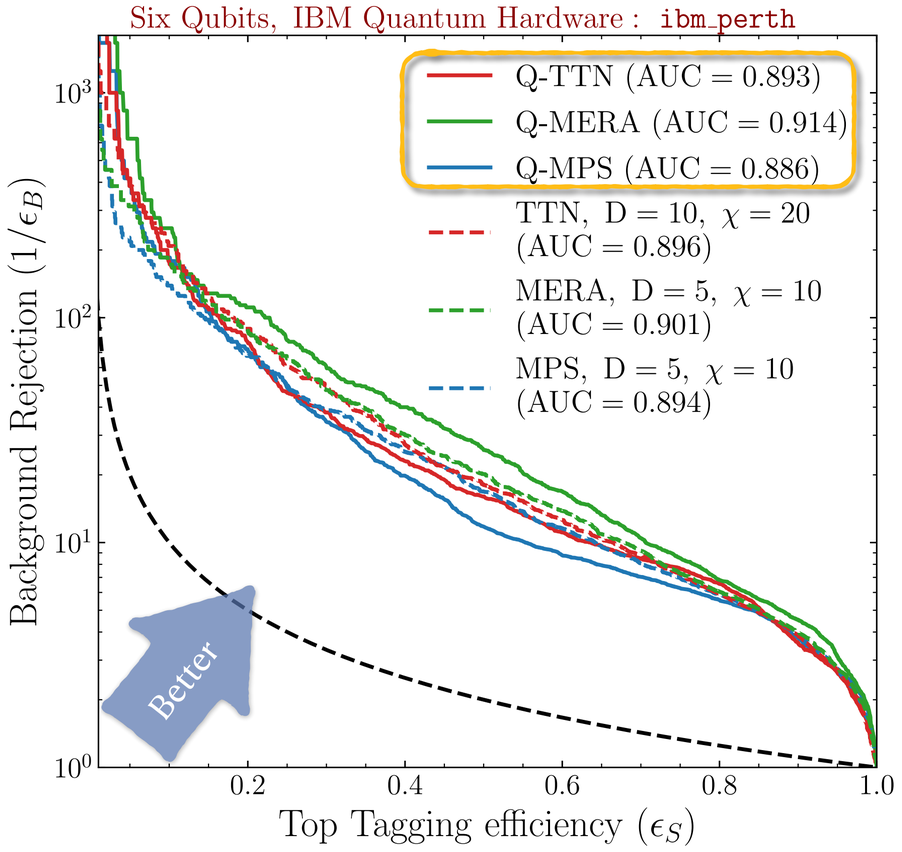
Experimenting with 6-Qubits





Experimenting with 6-Qubits





Ansatz	D	χ	# Parameters	AUC
TTN	2	5	235	0.755
	2	10	1320	0.803
	2	20	9040	0.849
	5	10	1950	0.873
	10	20	14800	0.896
MPS	2	5	230	0.811
	2	10	860	0.819
	2	20	3320	0.818
	5	10	2150	0.894
MERA	2	5	1225	0.850
	2	10	13400	0.840
	2	20	181600	0.848
	5	10	18200	0.901
Q-TTN	-	-	9	0.893
Q-MPS	-	-	9	0.886
Q-MERA	-	-	17	0.914

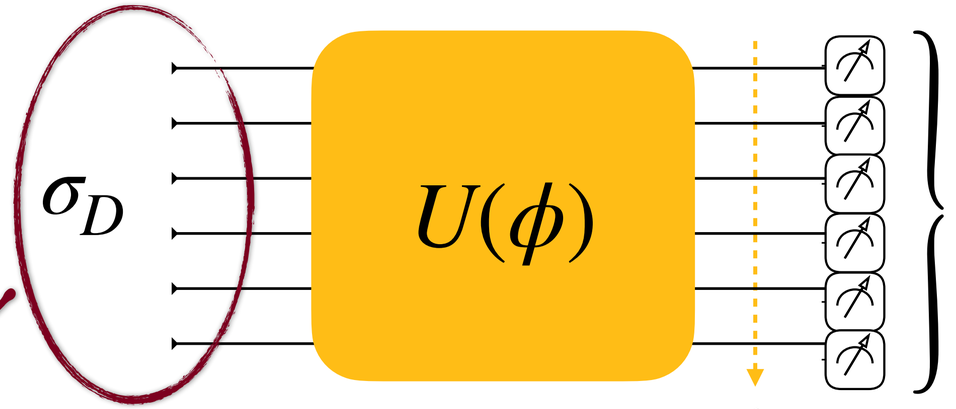
Loss landscape for classical TNs becomes exponentially flat!

Learning probability distributions with quantum-probabilistic hybrid learning

Quantum Modular Hamiltonian Learning

A data point can be represented as a mixed state

$$\sigma_D = \sum_i p_i |\psi_i\rangle \ , \ |\psi_i\rangle := \text{pure states}$$



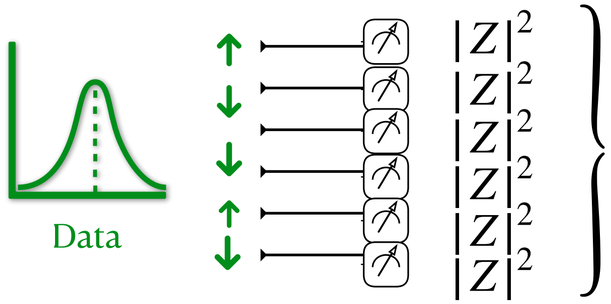
$$K_\theta = \sum E_\theta(x)\,|x\rangle\langle x|$$

An energy-based model
i.e. Boltzmann Machine

See Eliska's talk for details on RBM

Also Andrea's talk

$$\rho_{\theta,\phi} = \frac{1}{Z_\theta} e^{-U(\phi)K_\theta U^\dagger(\phi)}$$



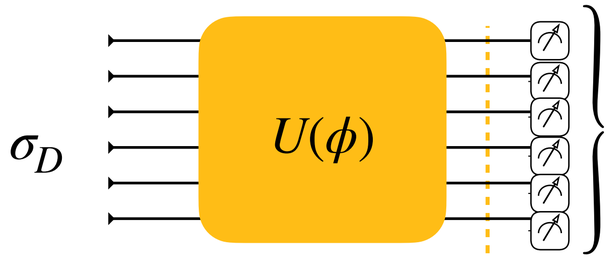
$\sim$ sigmoid(Data)

We want to approximate σ_D with $\rho_{\theta,\phi}$

$$\mathcal{L}(\theta,\phi) = \underbrace{\text{Tr}[\sigma_D U(\phi) K_\theta U^\dagger(\phi)] + \log Z_\theta}$$

$$\lim_{\rho_{\theta,\phi}\to\sigma_D} \mathcal{L}(\theta,\phi) = \mathcal{S}(\sigma_D)$$

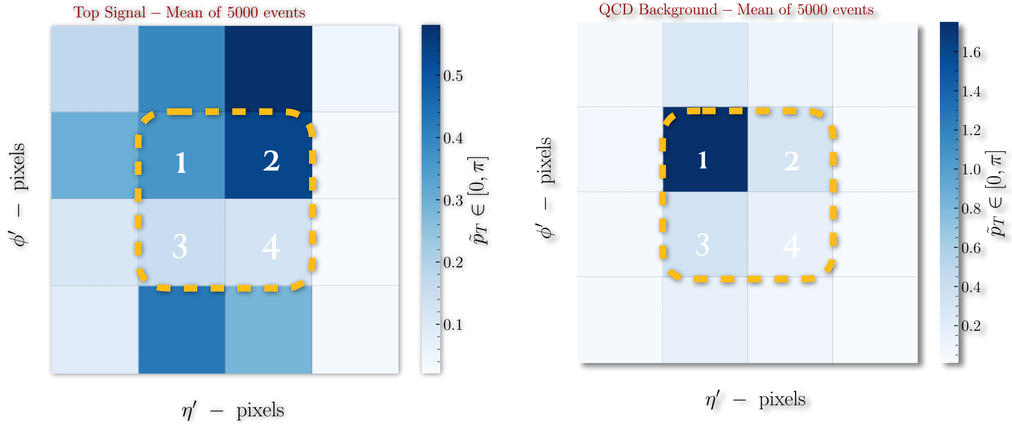
Quantum Modular Hamiltonian Learning



An energy-based model
i.e. Boltzmann Machine

$$K_\theta = \sum E_\theta(x)\,|x\rangle\langle x|$$

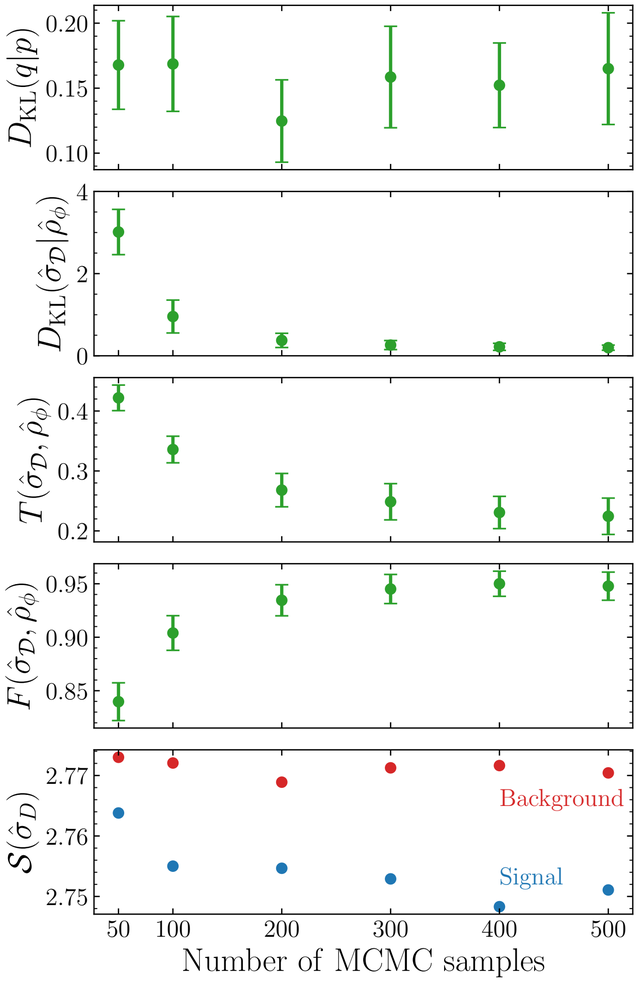
$$\rho_{\theta,\phi} = \frac{1}{Z_\theta} e^{-U(\phi)K_\theta U^\dagger(\phi)}$$



Trace Distance

Fidelity

Von Neumann Entropy



Preliminary Results

Conclusion

Conclusion

Classical

- Tensor Networks opens up the entire world of techniques developed for Quantum Mechanics to ML applications.
- A linear network allows a more straightforward interpretation.
- The perfect tool to do linear algebra in higher-dimensional spaces.

Main Drawbacks

- Cost to train can be high
- Choice of architecture is still a research area.

Advantages

- ☑ Interpretability
- ☑ Understanding
- ☑ Theory

Conclusion

Classical

- Tensor Networks opens up the entire world of techniques developed for Quantum Mechanics to ML applications.
- A linear network allows a more straightforward interpretation.
- The perfect tool to do linear algebra in higher-dimensional spaces.
- The optimisation landscape becomes exponentially flat with increasing bond dimensions and Hilbert space mapping.

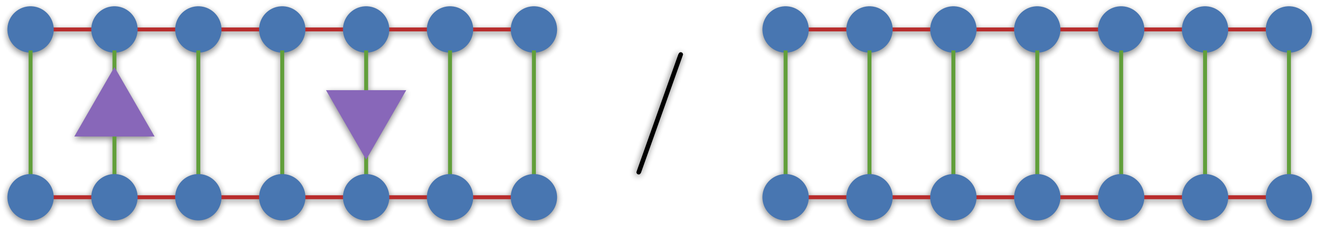
Quantum

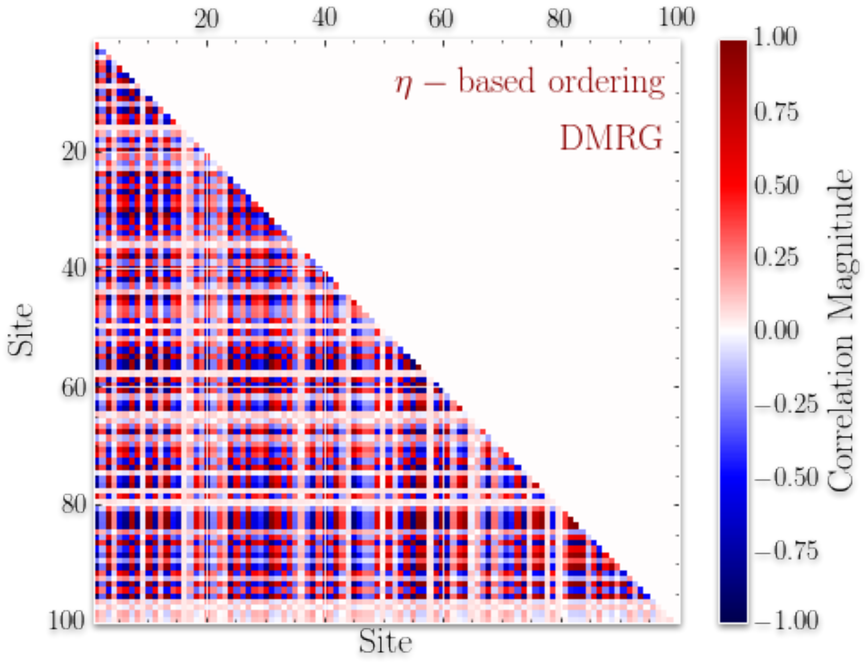
- Natural quantum systems have more representation capacity.
- Quantum Natural Gradient Descent allows faster optimisation compared to classical networks.
- BUT near term quantum devices are still very much limited to a few qubits.

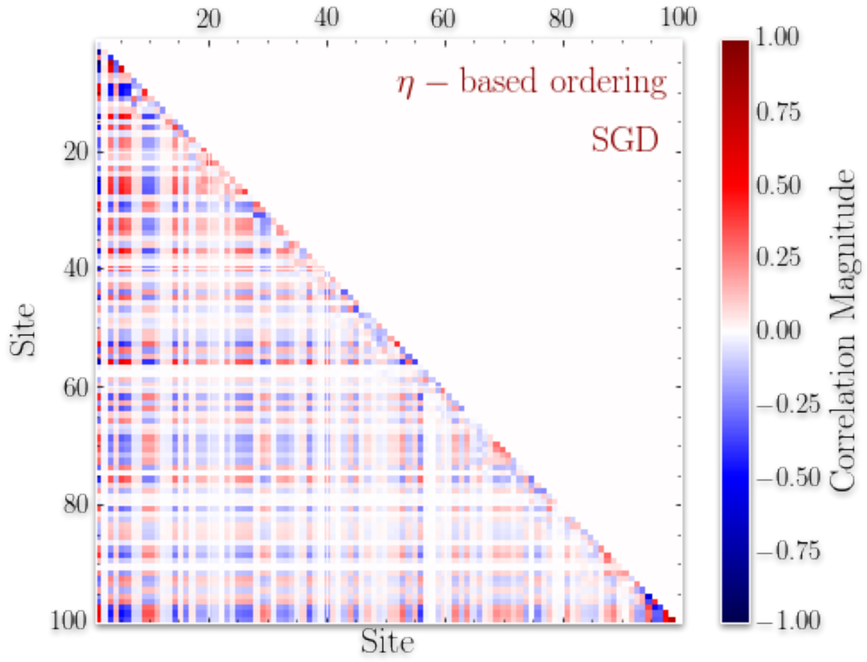
BACKUP

Correlations by SU(2) generators

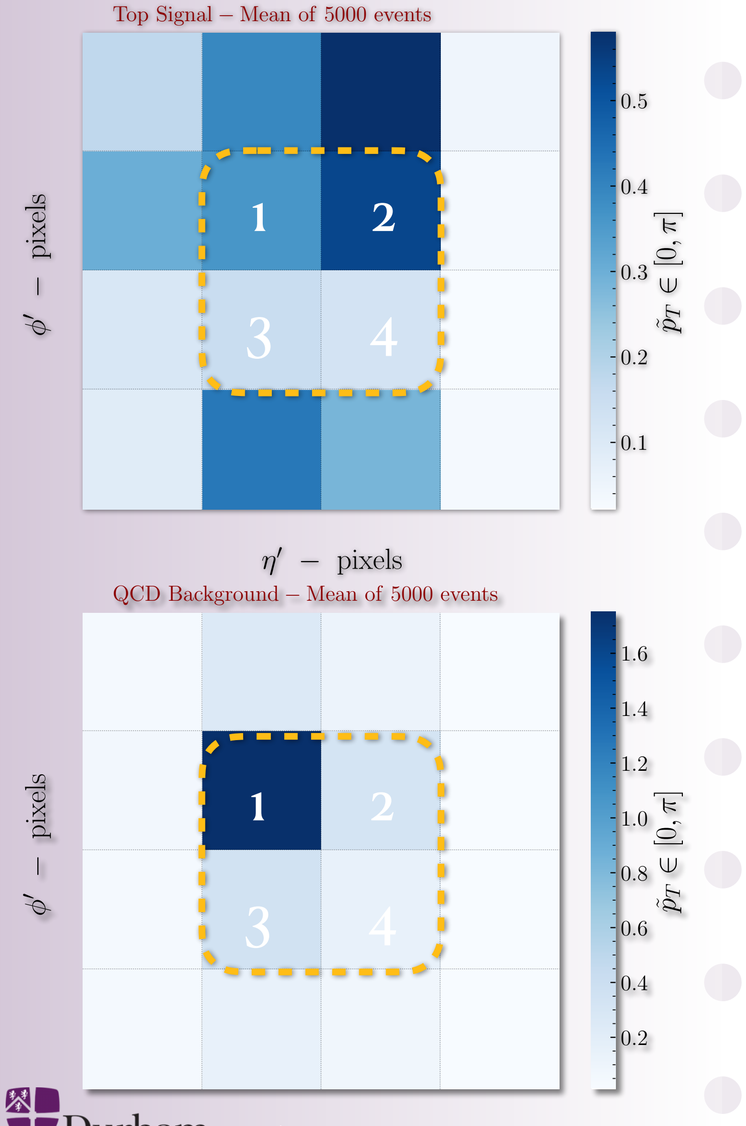
$$C_{ij}^{l} = \frac{\langle W^{l} | \mathcal{O}_i \mathcal{O}_j^{\dagger} | W^{l} \rangle}{\langle W^{l} | W^{l} \rangle} =$$



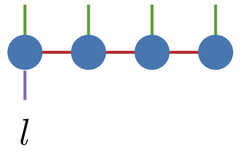


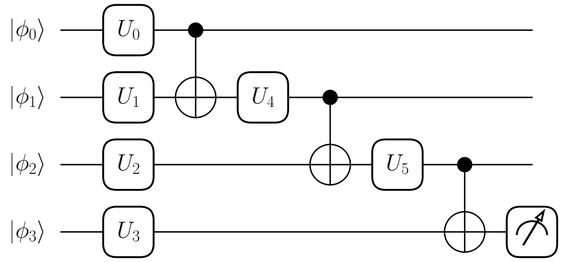


Experimenting with 4-Qubits

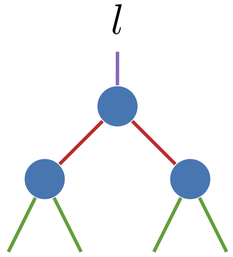


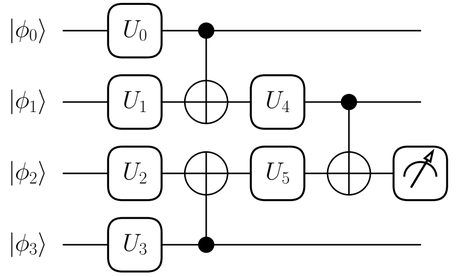
Matrix Product States



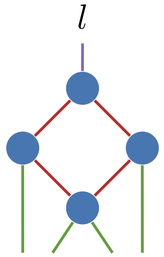


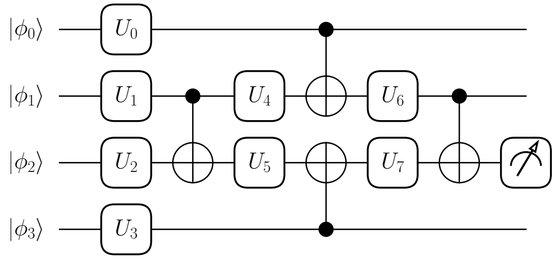
Tree Tensor Networks



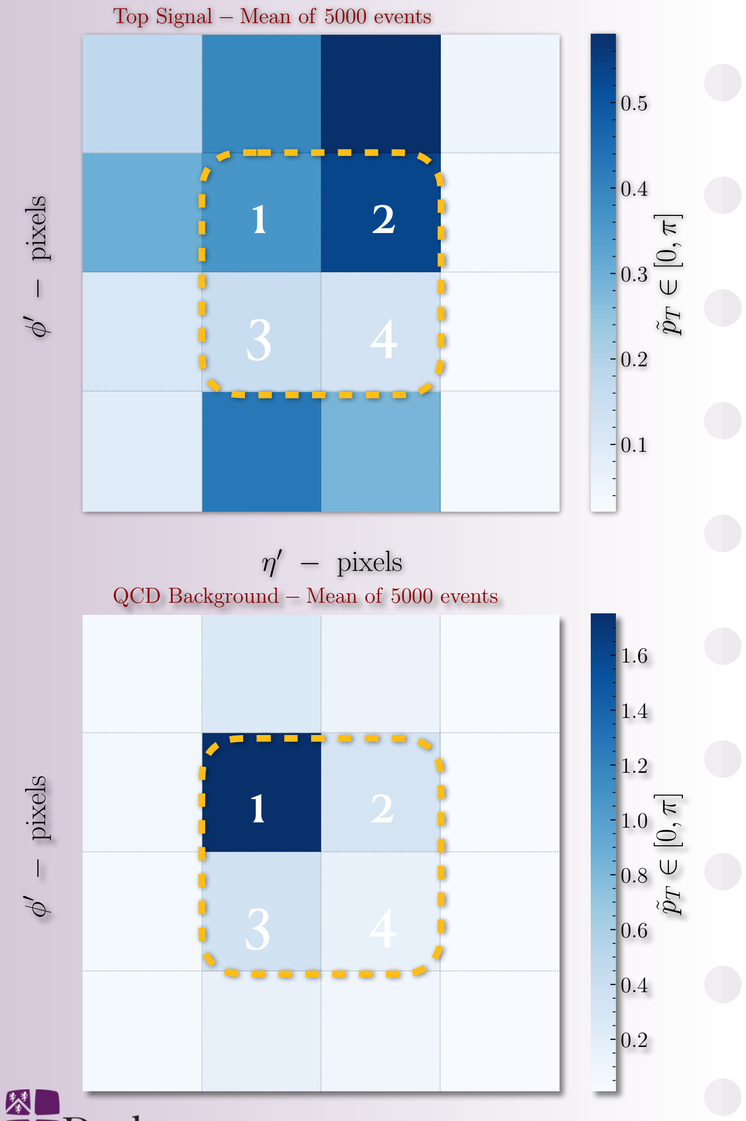


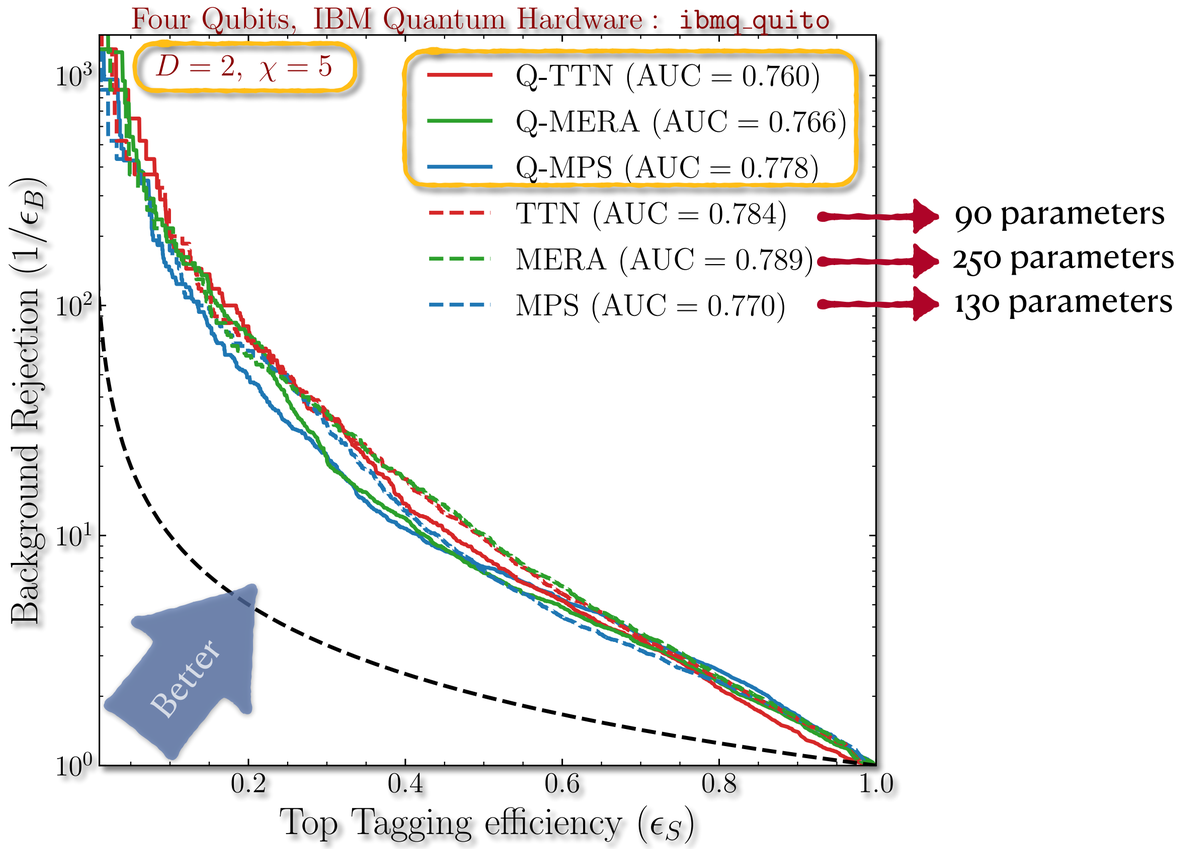
Multiscale Entanglement Renormalisation Ansatz



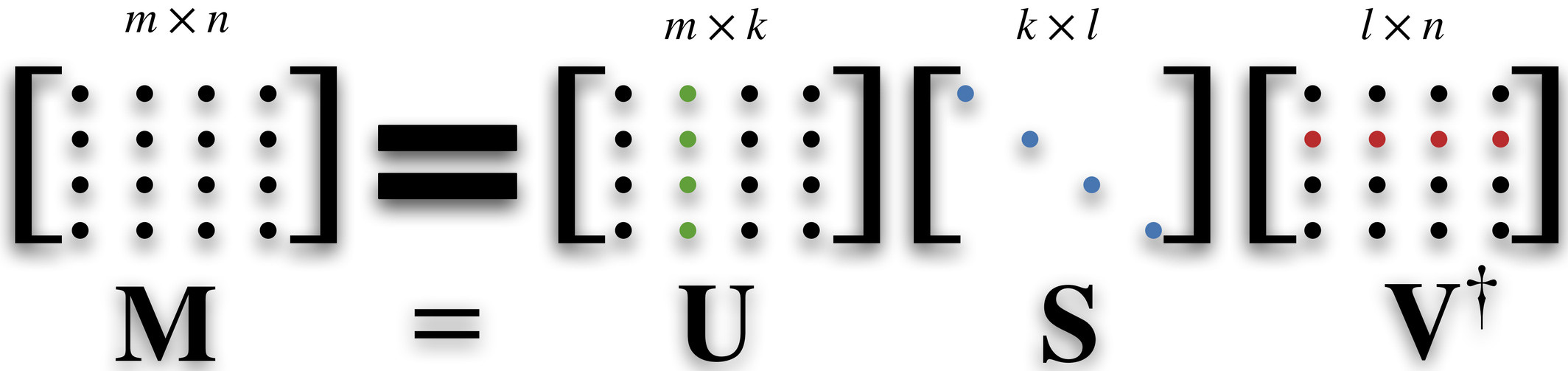


Experimenting with 4-Qubits





Singular Value Decomposition



Orthogonal singular column vectors

Positive definite singular values in descending order

Orthogonal singular row vectors

By changing the number of singular values one can change the accuracy of the decomposition!!!

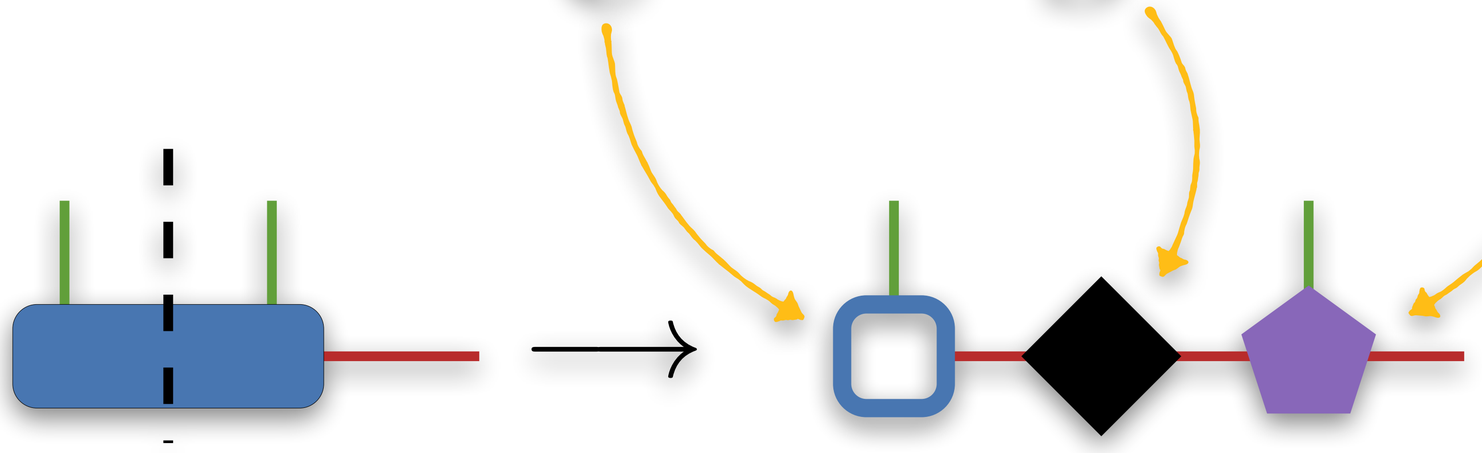
$$\mathbf{S} = \mathrm{diag}(\lambda_1, \lambda_2, \cdots, \lambda_n)$$

λ_i also known as Schmidt values

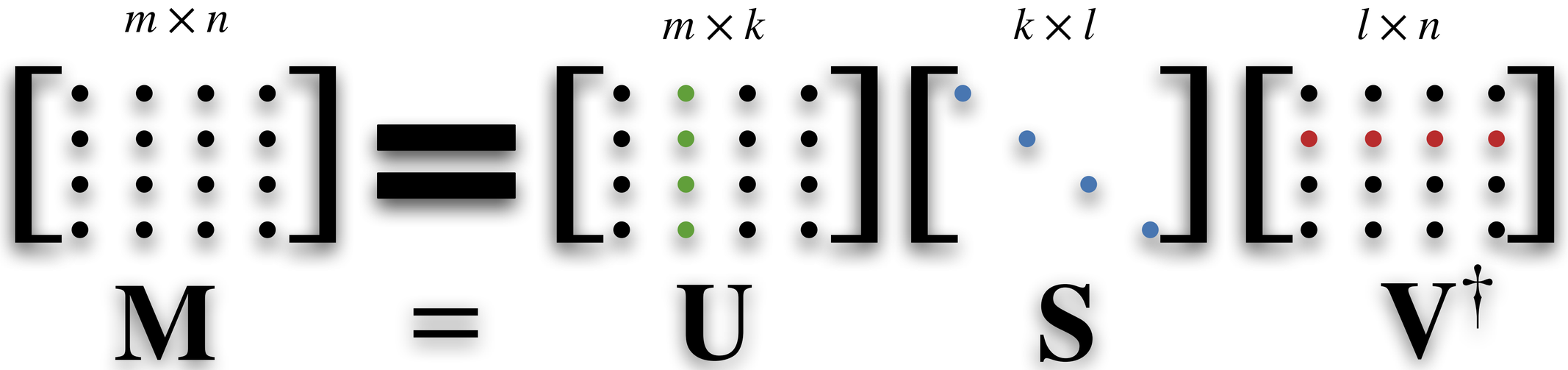
Singular Value Decomposition

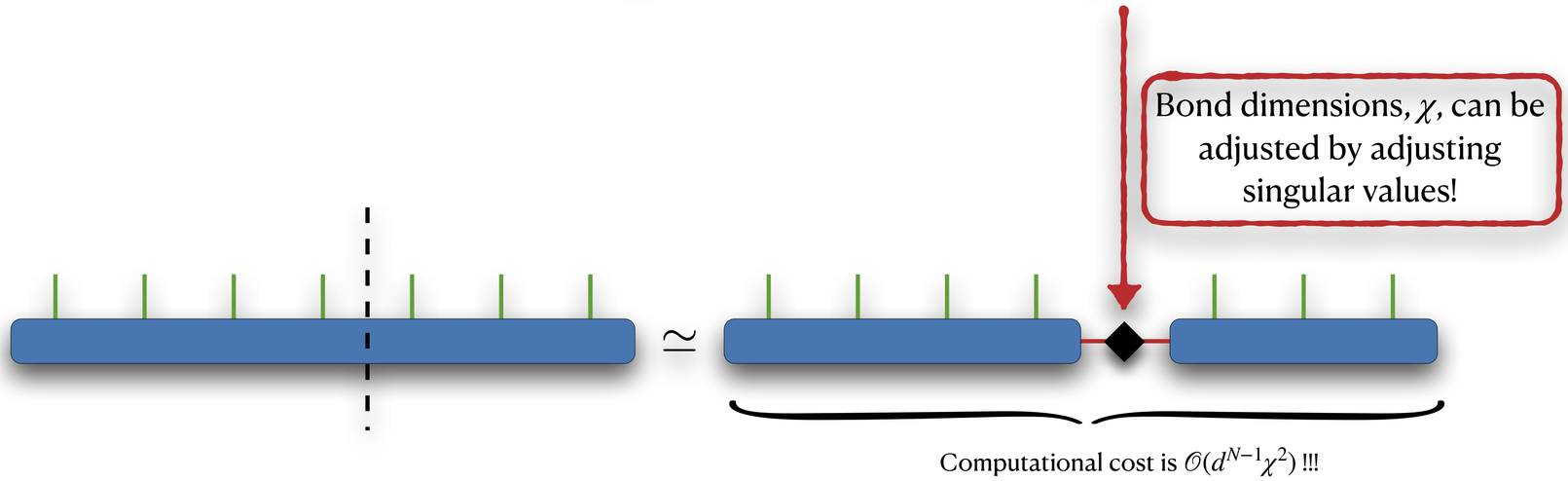
$m \times n$ $m \times k$ $k \times l$ $l \times n$

$$\mathbf{M} = \mathbf{U}\,\mathbf{S}\,\mathbf{V}^{\dagger}$$



Singular Value Decomposition





Matrix Product States for Classification

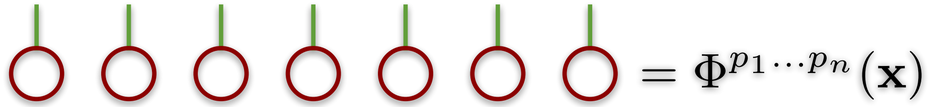
Sub-Outline

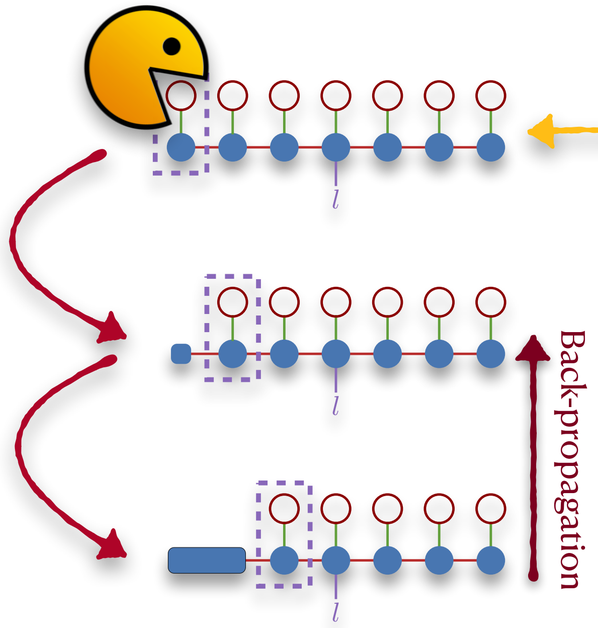
- How to embed the data?
- How to form a network?
- How to train the network?

Data Embedding

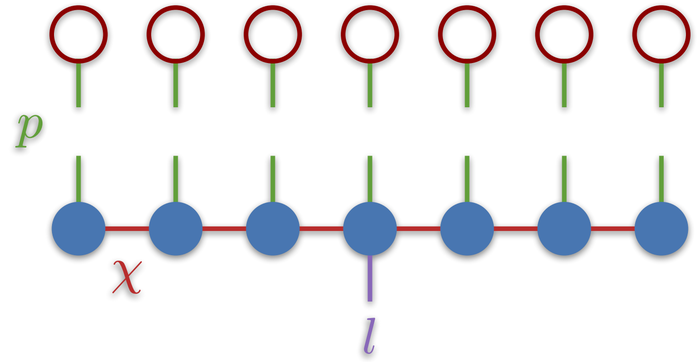
$$\Phi^{p_1 \cdots p_n}(\mathbf{x}) = \phi^{p_1}(x_1) \otimes \phi^{p_2}(x_2) \otimes \cdots \otimes \phi^{p_n}(x_n)$$

$$\phi^{p_i}(x_i) = \begin{bmatrix} \cos(x_i\ \pi/2) \\ \sin(x_i\ \pi/2) \end{bmatrix} \quad \text{or} \quad \phi^{p_i}(x_i) = \begin{bmatrix} 1 \\ x_i \\ x_i^2 \end{bmatrix} \quad \text{or} \quad \cdots$$

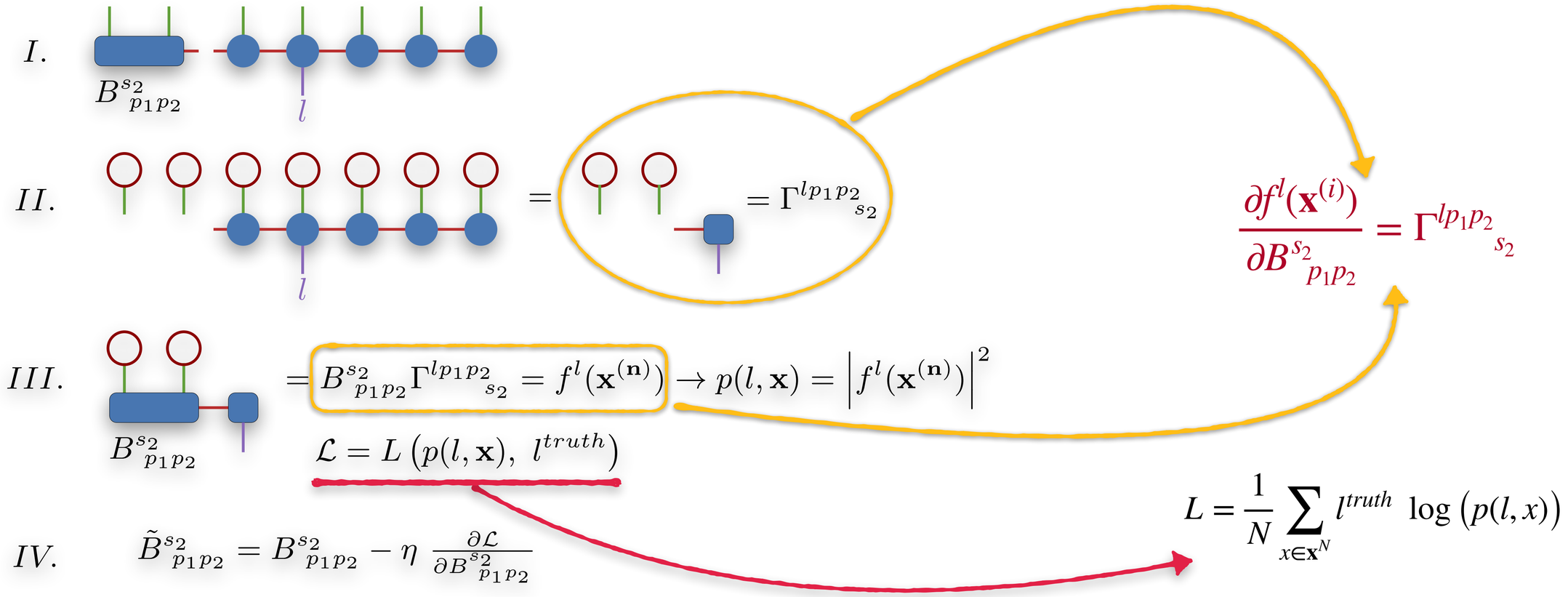




An efficient contraction algorithm is essential for training!



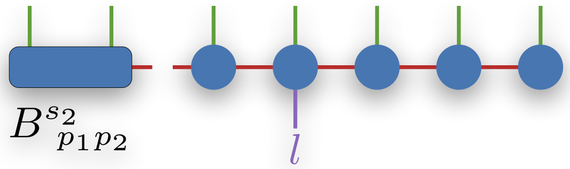
Density Matrix Renormalization Group Algorithm



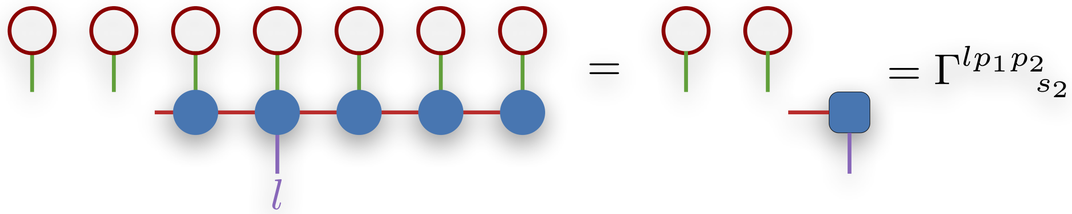
Initially proposed in: Stoudenmire, Schwab; '16

Density Matrix Renormalization Group Algorithm

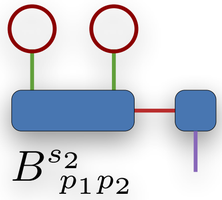
I.



II.



III.



$$= B^{s_2}_{p_1 p_2} \Gamma^{l p_1 p_2}{}_{s_2} = f^l(\mathbf{x}^{(\mathbf{n})}) \rightarrow p(l, \mathbf{x}) = \left| f^l(\mathbf{x}^{(\mathbf{n})}) \right|^2$$

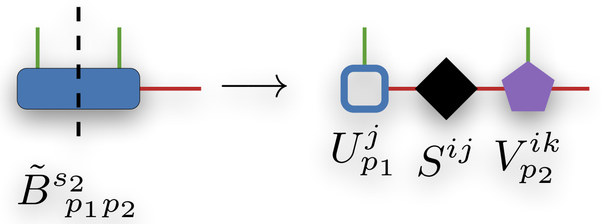
$$\mathcal{L} = L\left(p(l, \mathbf{x}),\ l^{truth}\right)$$

IV.

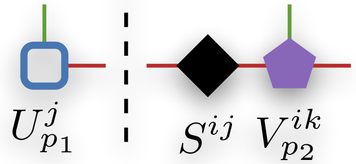
$$\tilde{B}^{s_2}_{p_1 p_2} = B^{s_2}_{p_1 p_2} - \eta \frac{\partial \mathcal{L}}{\partial B^{s_2}_{p_1 p_2}}$$

Adjust the bond dimension via SVD!

V.

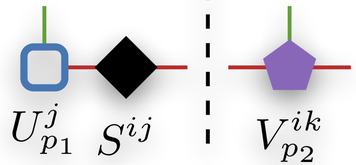


VI. Sweep direction to the right:



Node I Node II

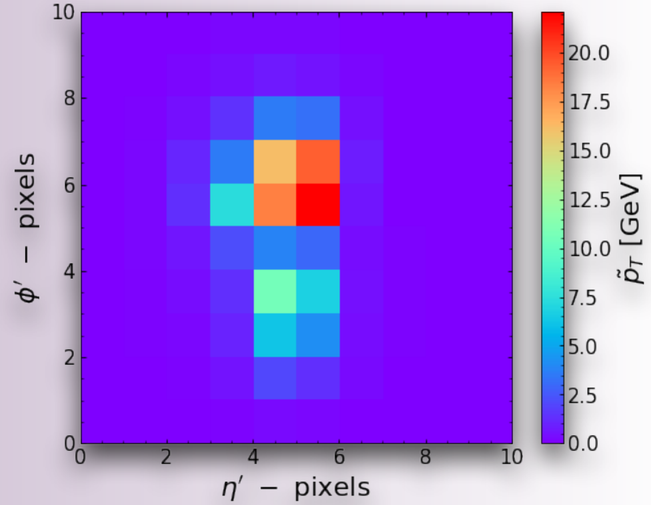
Sweep direction to the left:

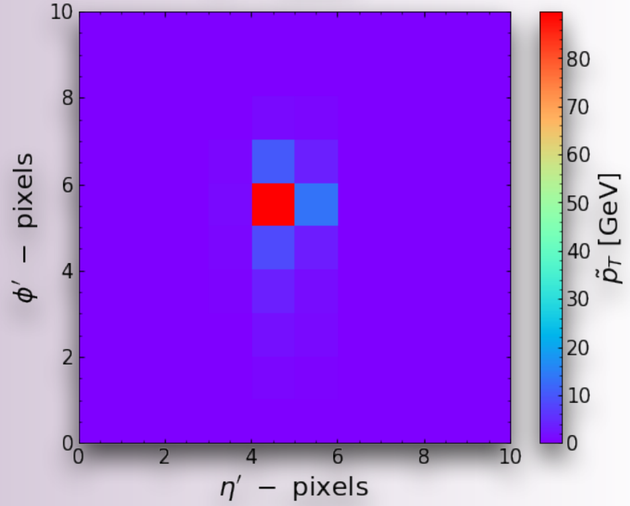


Node I Node II

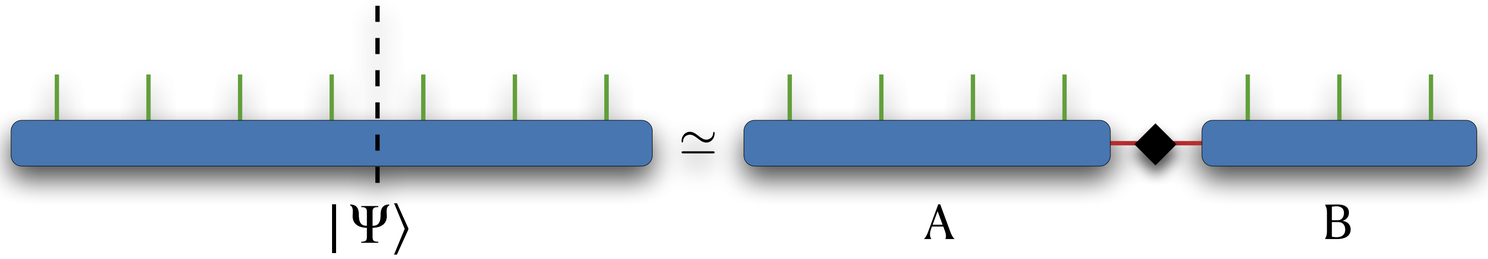
Initially proposed in: Stoudenmire, Schwab; '16

Top Tagging through MPS





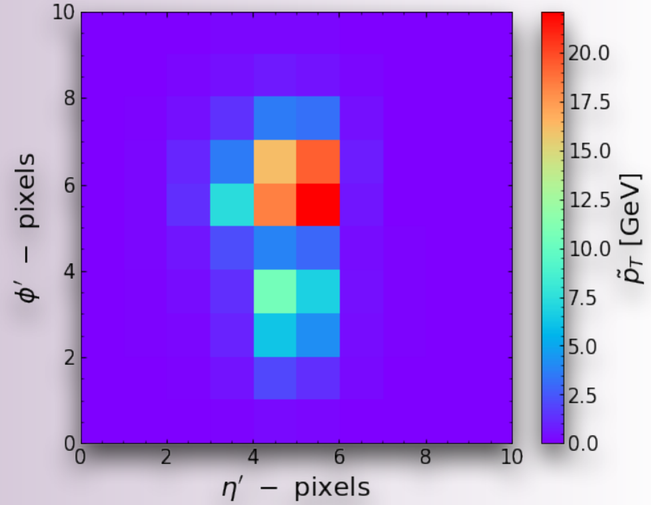
Entanglement Entropy & Schmidt Decompostion

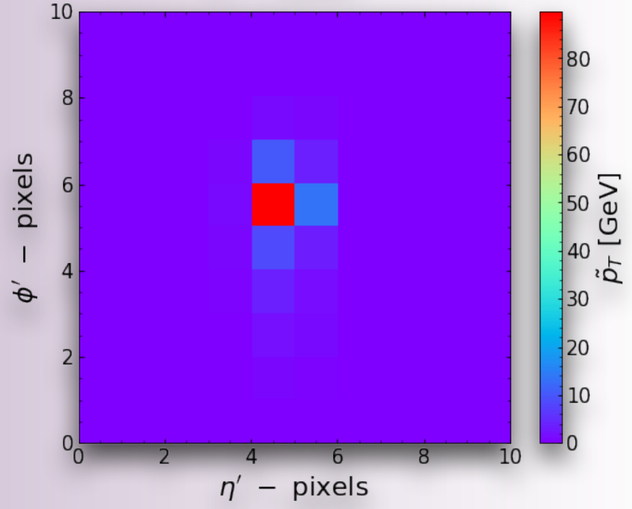


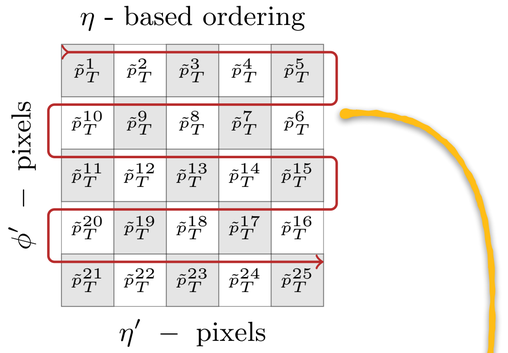
$$|\Psi\rangle_{AB} = \sum_{i}^{\chi} \lambda_\alpha |\alpha\rangle_A |\alpha\rangle_B \quad \rightarrow \quad \lambda_\alpha := \text{Schmidt values}$$

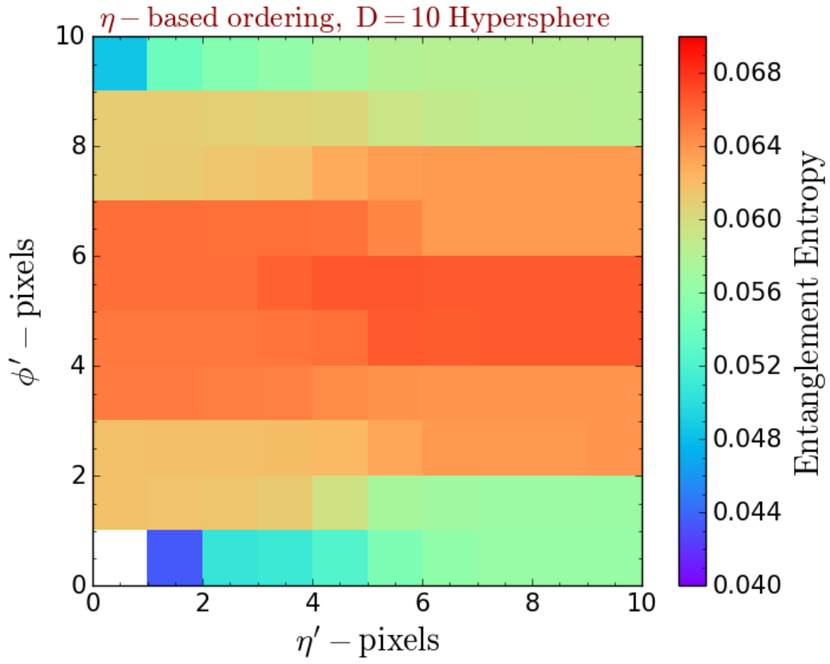
$$\mathcal{S} = -\sum_{i}^{\chi} \lambda_i^2 \log \lambda_i^2 \quad \rightarrow \quad \text{von Neumann entropy}$$

Top Tagging through MPS

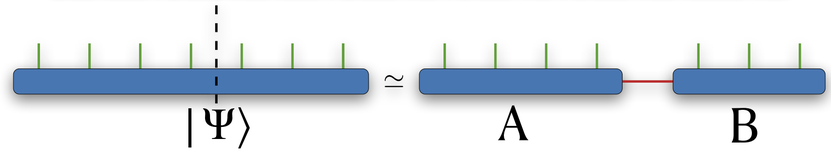






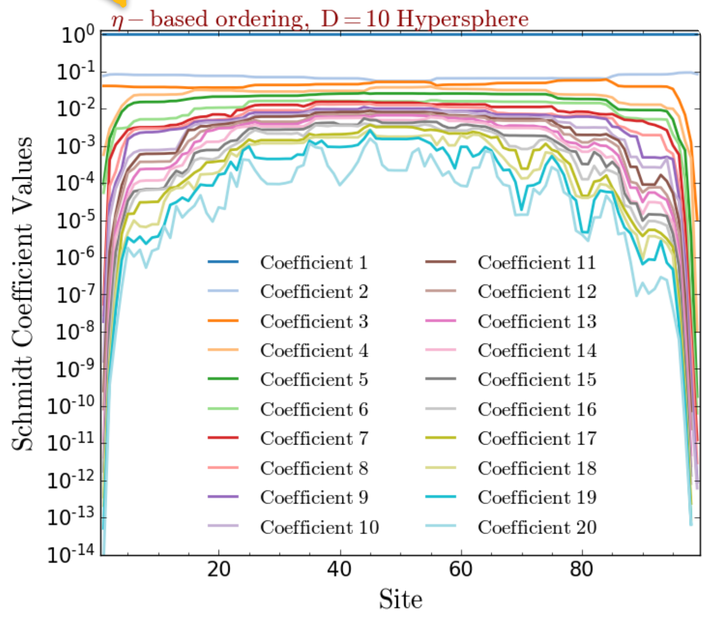


Entanglement Entropy & Schmidt Decompostion



$$|\Psi\rangle_{AB} = \sum_i^{\chi} \lambda_\alpha |\alpha\rangle_A |\alpha\rangle_B$$

$$\mathcal{S} = -\sum_i^{\chi} \lambda_i^2 \log \lambda_i^2$$



Fisher Information & Effective Dimensions