Bridge between Classical & Quantum Machine Learning

Jack Y. Araz
INSTITUTE FOR PARTICLE PHYSICS PHENOMENOLOGY
DURHAM UNIVERSITY

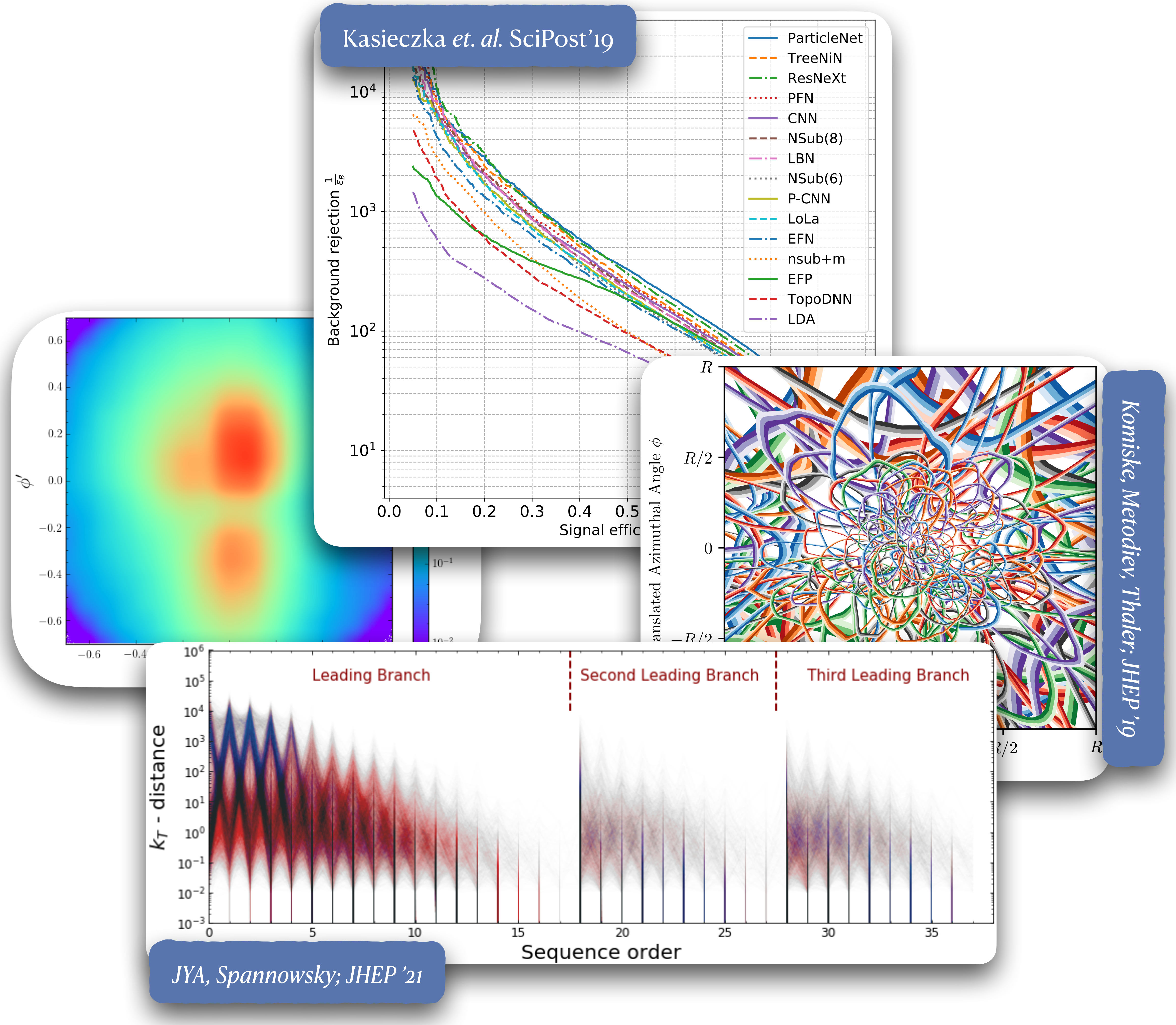
Based on
JHEP 08 (2021) 112; arXiv: 2106.08334 [hep-ph] & arXiv: 2202.10471 [quant-ph]
with Michael Spannowsky

Mainz Institute for Theoretical Physics
June 29th, 2022

Sales pitch of the talk!

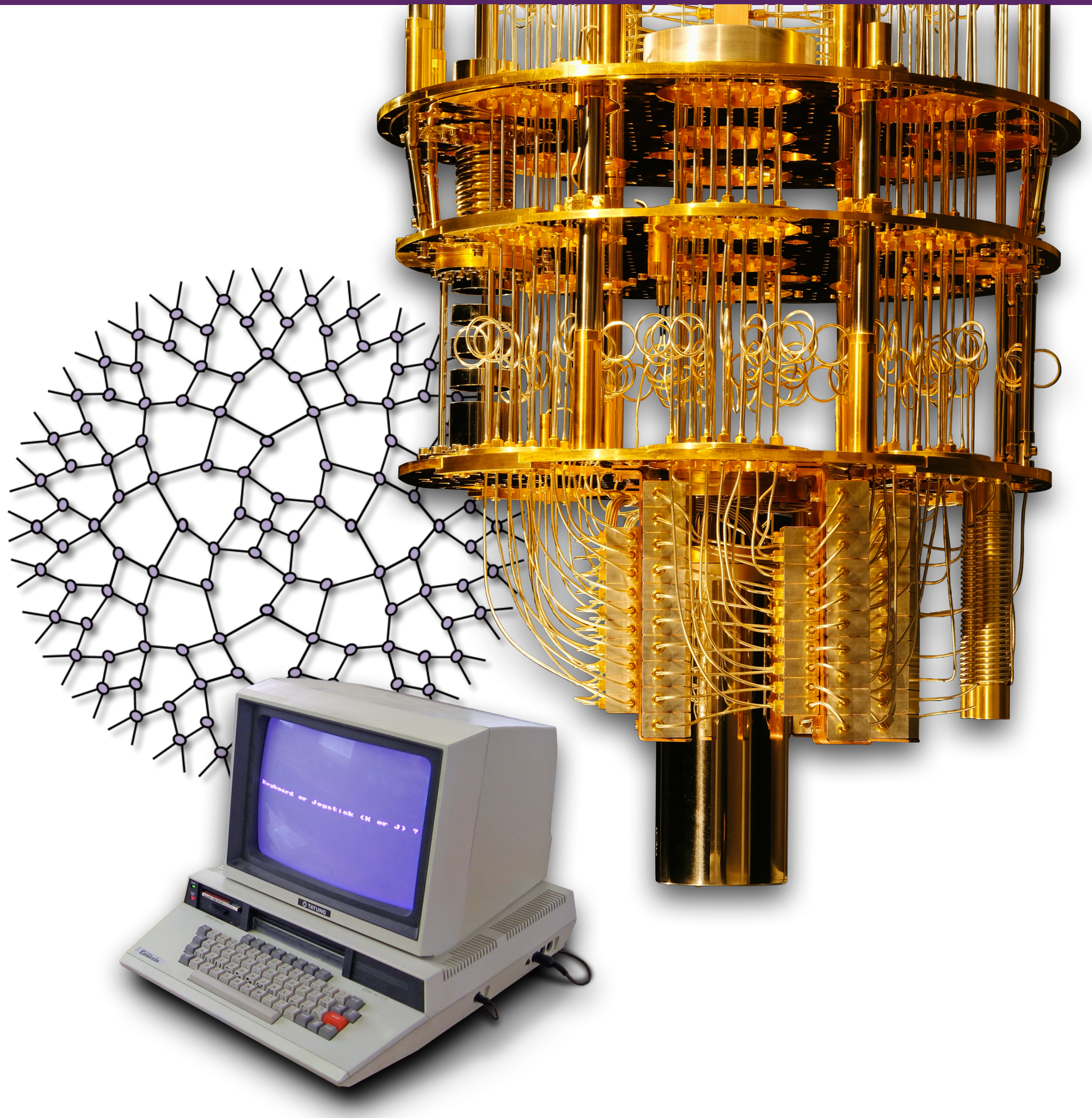
- We more or less know how to get a well-performing Neural Network
- What we don't know is what this network learns.
- Can we use Quantum Mechanics to have more insight into the learning process?
  - ✦ What has a model learned?
  - ✦ What is learning?
  - ✦ How do we develop "insightful" algorithms?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

❖ All comes together with Tensor Networks!



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
- Conclusion

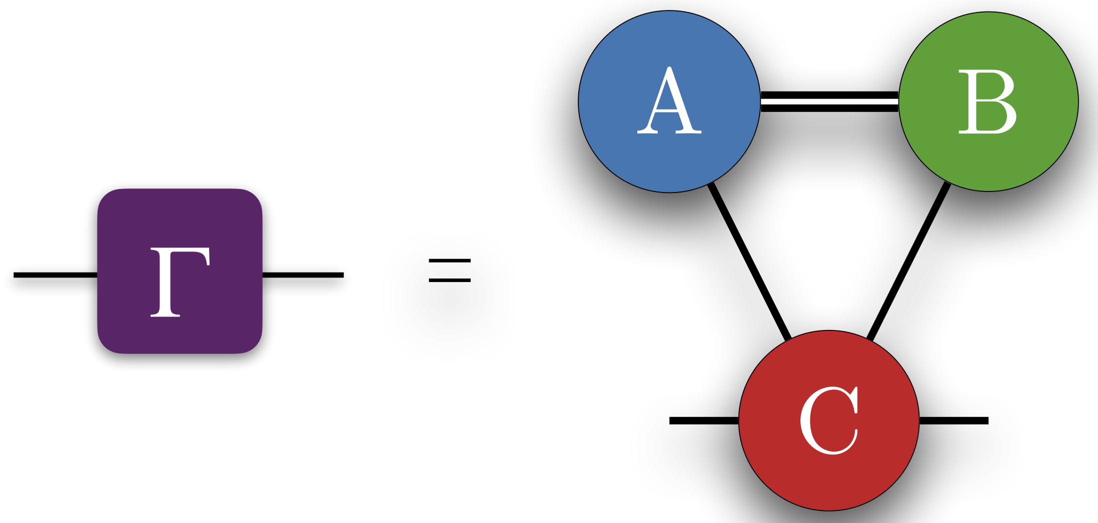


Introduction

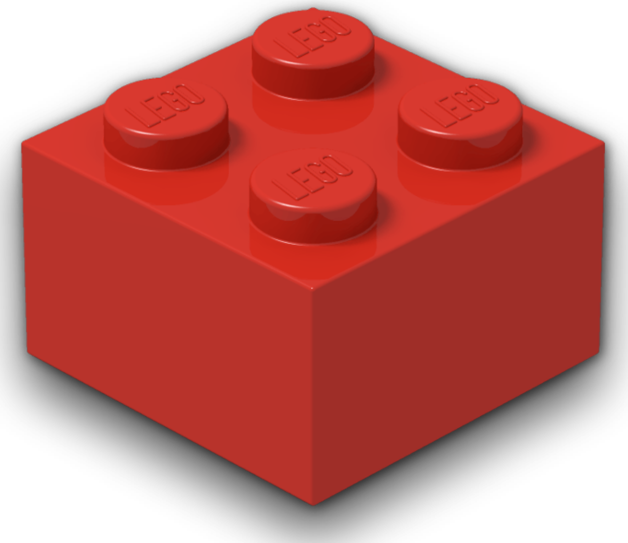
Tensor Networks: Origins

Tensor Diagram Notation

- Scalar
- Vector, v^i
- Matrix, Λ^{ij}
- Rank 3 Tensor, Γ^{ijk}



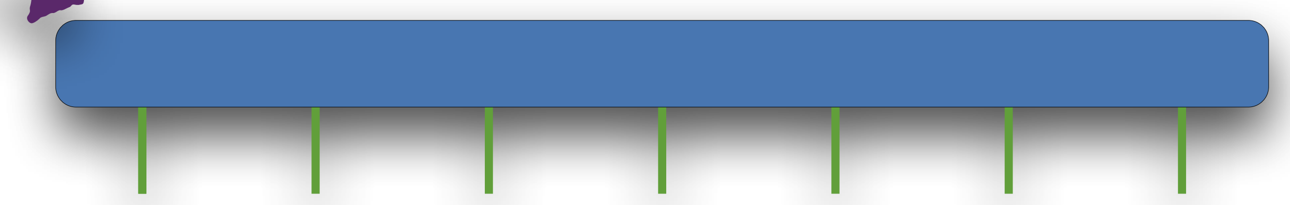
$$\Gamma^{ij} = A^{klm} B^{n}{}_{lm} C^{ij}{}_{kn}$$



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

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

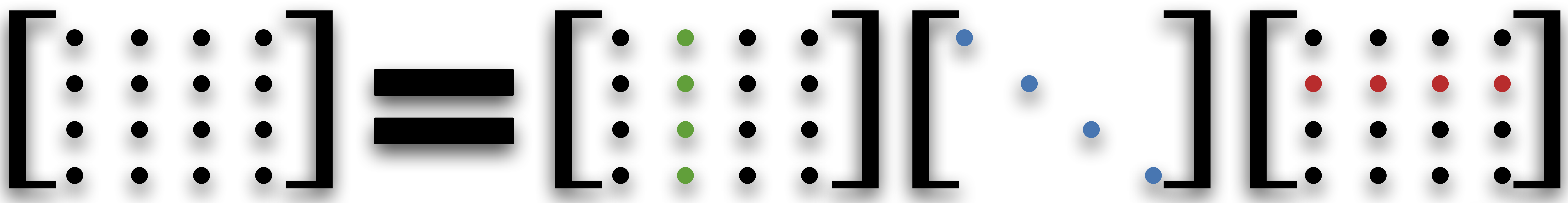
$$\forall |\phi_i\rangle \in \mathscr{H}^{\otimes 2^N} \quad \to \quad |\phi_i\rangle \in \{|\uparrow\rangle, |\downarrow\rangle\}$$



The computational cost of a rank-N tensor is $\mathcal{O}(d^N)$!!!

Throwback linear algebra:

Singular Value Decomposition



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

Recall: Principal Component Analysis

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!!!

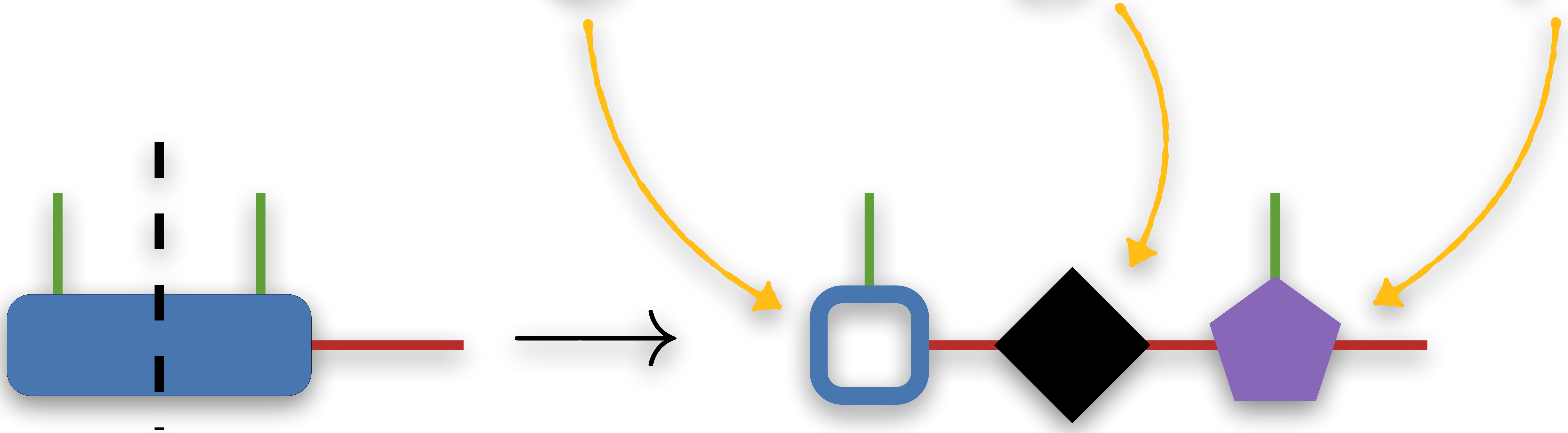
$$\mathbf{S} = \mathrm{diag}(\lambda_1, \lambda_2, \cdots, \lambda_n)$$

λ_i also known as Schmidt values

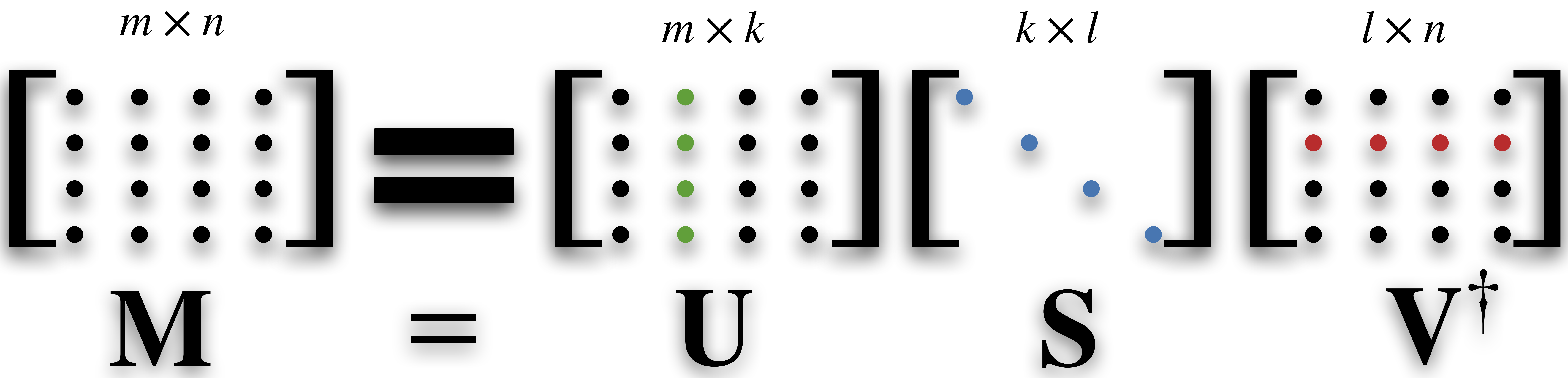
Singular Value Decomposition

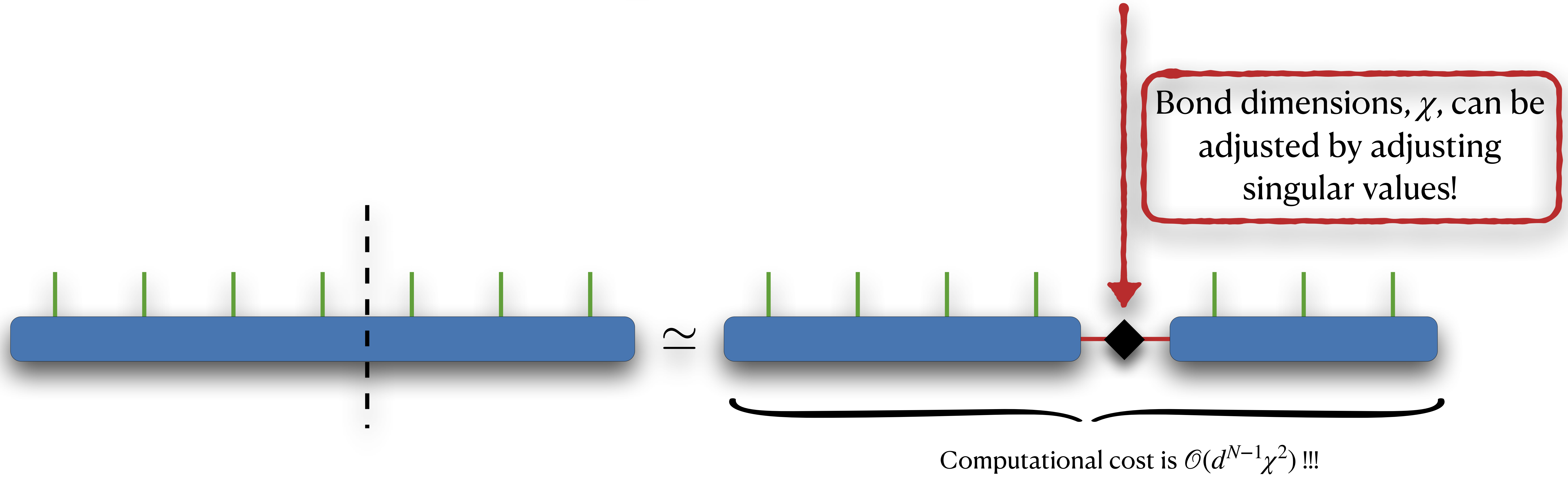
$$
\underset{m \times n}{\begin{bmatrix} \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \end{bmatrix}} = \underset{m \times k}{\begin{bmatrix} \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \end{bmatrix}} \underset{k \times l}{\begin{bmatrix} \bullet & & & \\ & \bullet & & \\ & & \bullet & \\ & & & \bullet \end{bmatrix}} \underset{l \times n}{\begin{bmatrix} \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \end{bmatrix}}
$$

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

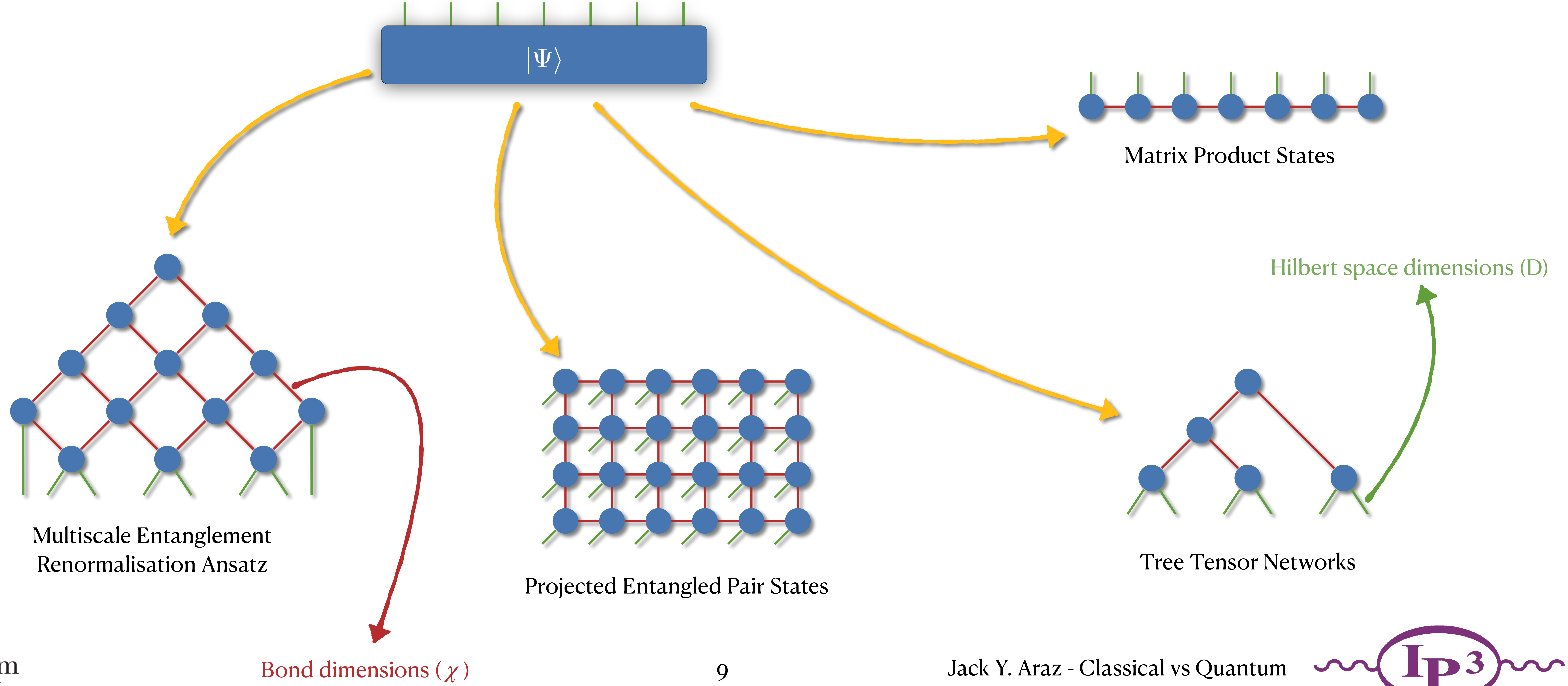


Singular Value Decomposition

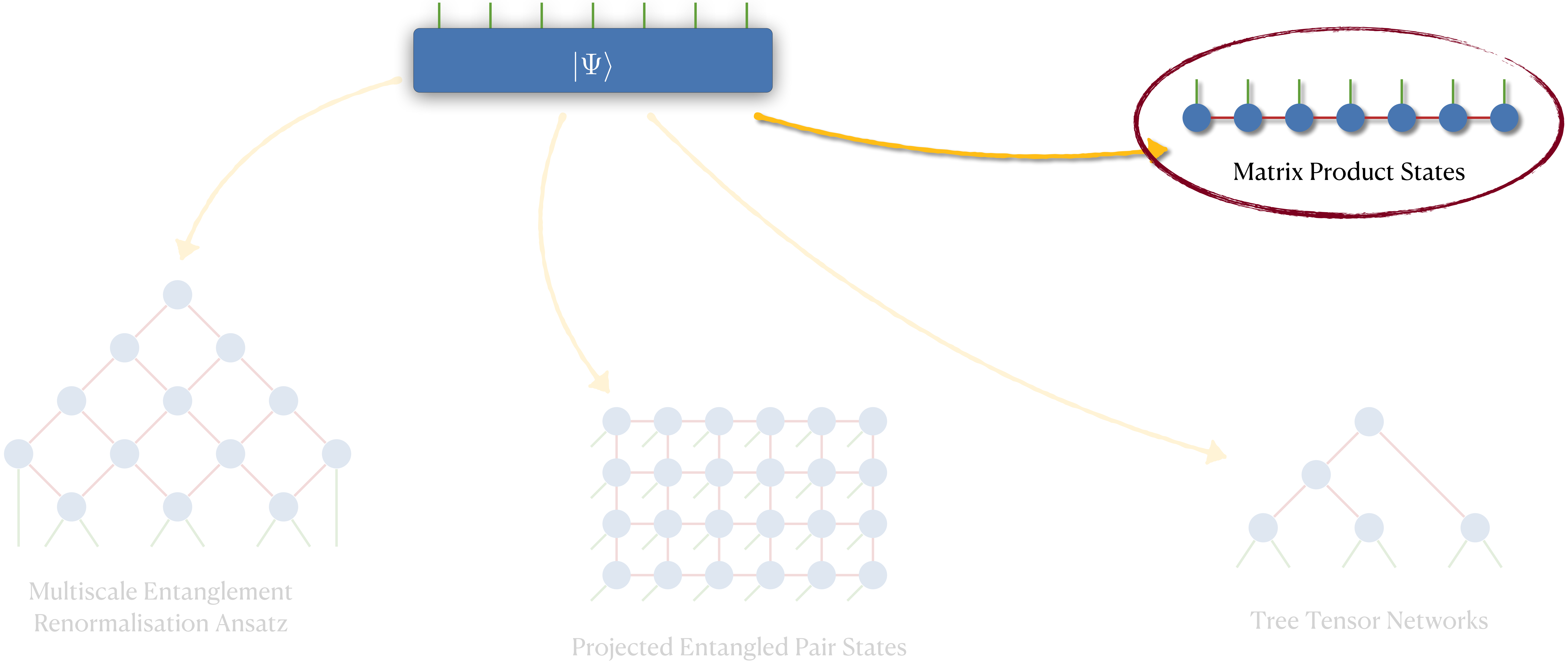




Types of Tensor Networks (some of them)



Types of Tensor Networks (some of them)



Matrix Product States for Classification

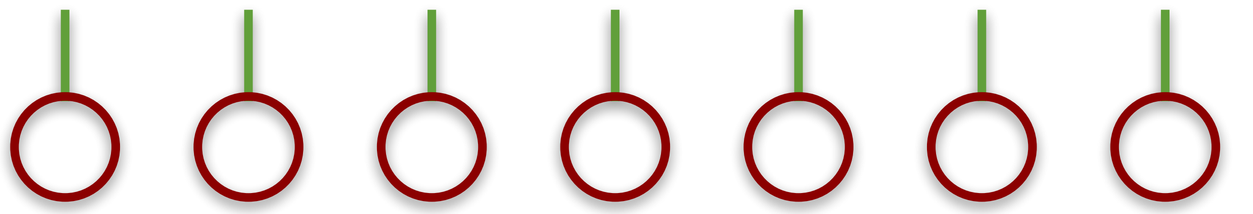
Sub-Outline

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

Data Embedding

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

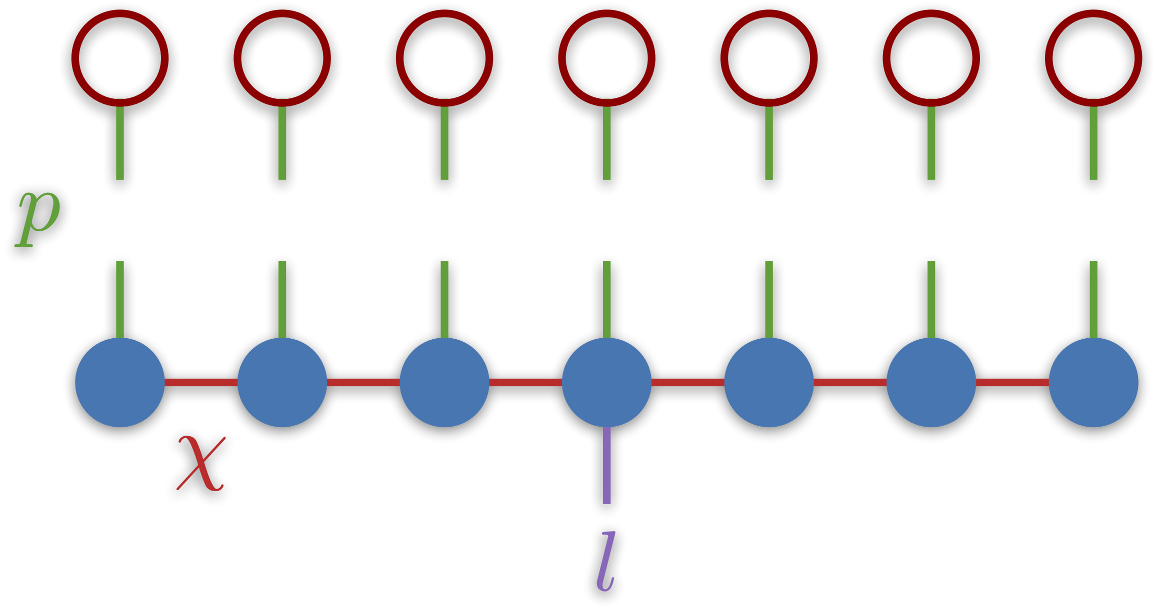
$$\Phi^{p_1\cdots p_n}(\mathbf{x}) = \bigotimes_{p_i=0}^{N} \phi^{p_i}(x_i) \qquad \phi^{p_i}(x_i) = \sum_{j=0}^{m-1} \alpha_j |j\rangle$$



$= \Phi^{p_1\cdots p_n}(\mathbf{x})$

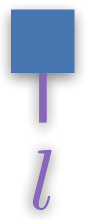
$$|\Psi\rangle = \sum_{p_1,\ldots,p_n=0} \mathscr{W}_{p_1\ldots p_n} |p_1\rangle \otimes |p_2\rangle \otimes \ldots \otimes |p_n\rangle$$

Little modification



$= \Phi^{p_1\cdots p_n}(\mathbf{x})$

$= \mathcal{W}^l{}_{p_1\cdots p_n}$



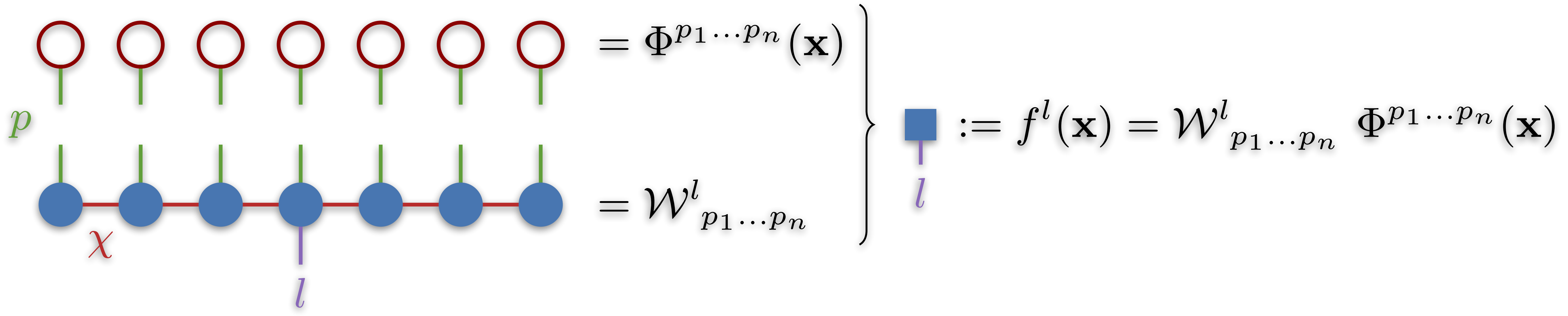
$:= f^l(\mathbf{x}) = \mathcal{W}^l{}_{p_1\cdots p_n}\ \Phi^{p_1\cdots p_n}(\mathbf{x})$

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!!

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

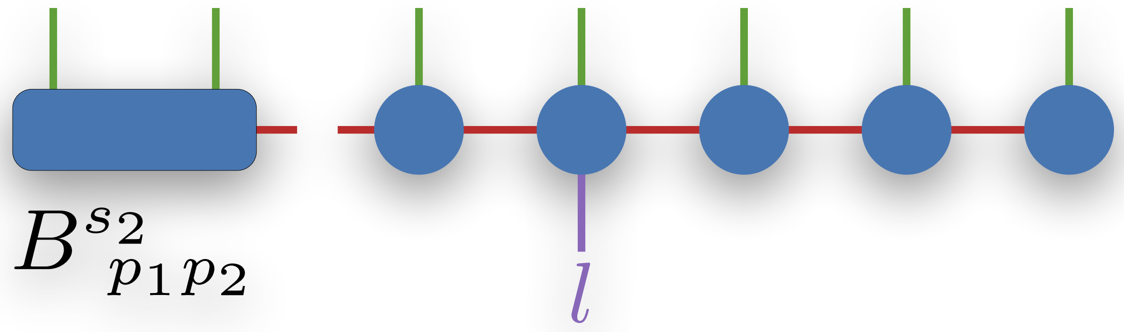
$$\mathscr{L} = \frac{1}{N}\sum_{x\in\mathbf{x}^N} q^{\text{truth}}\ \log\left(p(x^{(i)};\theta)\right)$$

Or anything else you like to minimize...

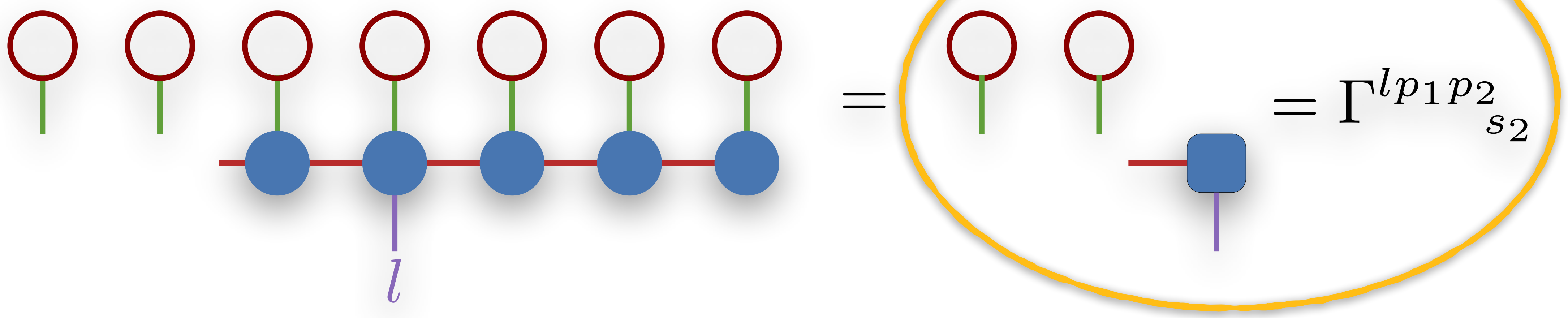
Traditionally NNs are trained with SGD, but MPS is trained with Density Matrix Renormalisation Group Algorithm

Density Matrix Renormalisation Group Algorithm

$I.$

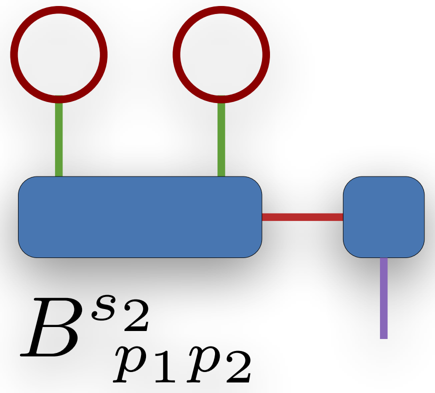


$II.$



$$\frac{\partial f^l(\mathbf{x}^{(i)})}{\partial B^{s_2}{}_{p_1 p_2}} = \Gamma^{l p_1 p_2}{}_{s_2}$$

$III.$



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

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

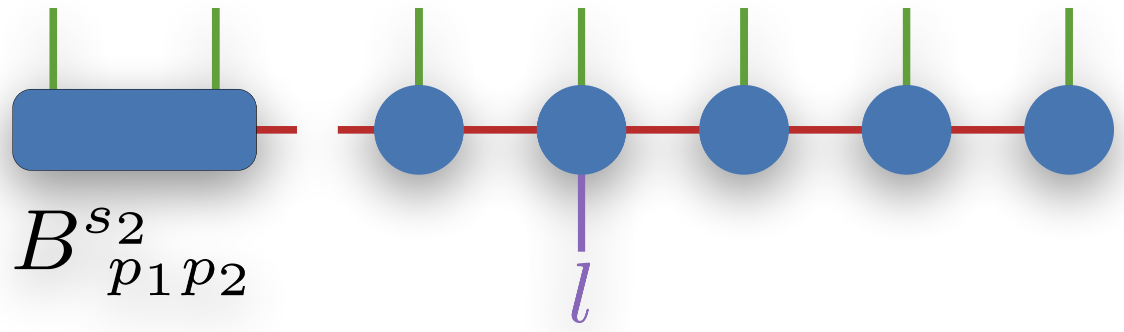
$$L = \frac{1}{N} \sum_{x \in \mathbf{x}^N} l^{truth} \ \log\left(p(l, x)\right)$$

$IV.$

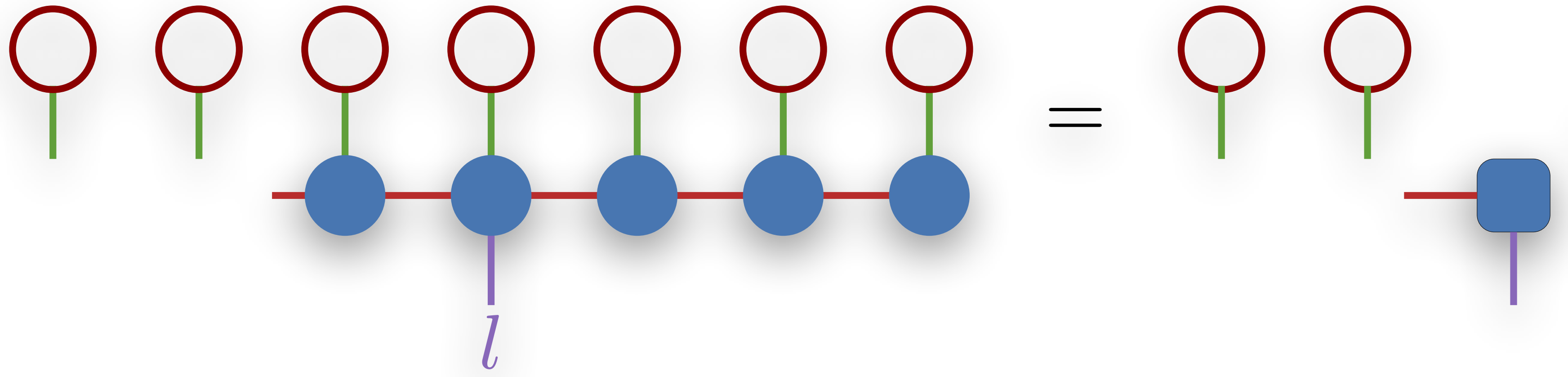
$$\tilde{B}^{s_2}{}_{p_1 p_2} = B^{s_2}{}_{p_1 p_2} - \eta \ \frac{\partial \mathcal{L}}{\partial B^{s_2}{}_{p_1 p_2}}$$

Density Matrix Renormalisation Group Algorithm

$I.$

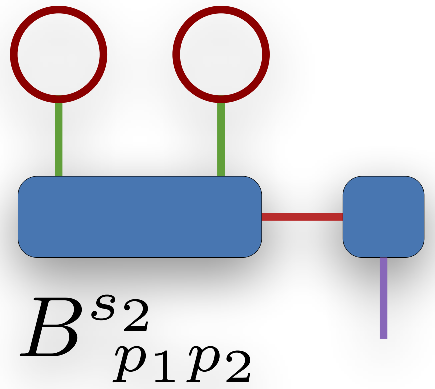


$II.$



$= \Gamma^{lp_1p_2}{}_{s_2}$

$III.$



$$= B^{s_2}{}_{p_1p_2}\Gamma^{lp_1p_2}{}_{s_2} = f^l(\mathbf{x}^{(\mathbf{n})}) \to p(l,\mathbf{x}) = \left|f^l(\mathbf{x}^{(\mathbf{n})})\right|^2$$

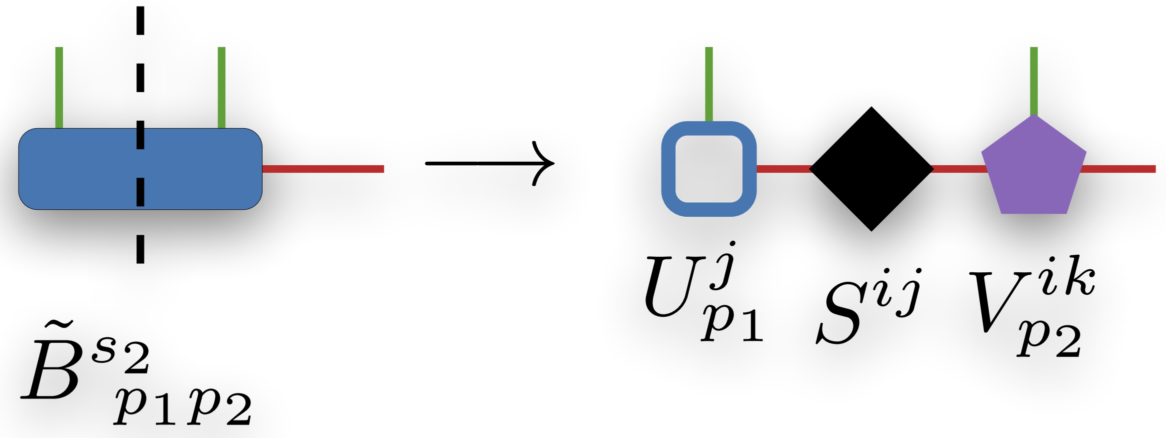
$$\mathcal{L} = L\left(p(l,\mathbf{x}),\ l^{truth}\right)$$

$IV.$

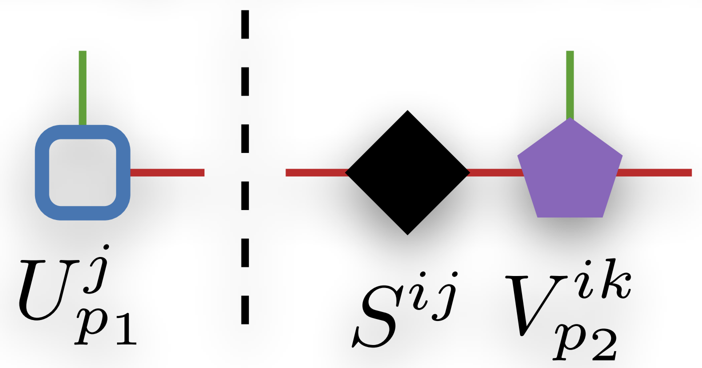
$$\tilde{B}^{s_2}{}_{p_1p_2} = B^{s_2}{}_{p_1p_2} - \eta\ \frac{\partial\mathcal{L}}{\partial B^{s_2}{}_{p_1p_2}}$$

Adjust the bond dimension via SVD!

$V.$

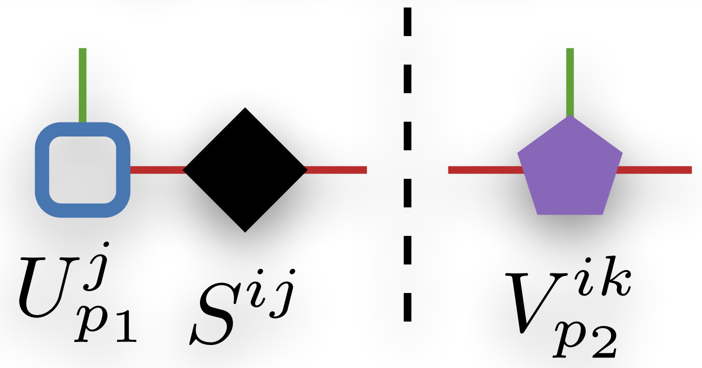


$VI.$ Sweep direction to the right:



Node I Node II

Sweep direction to the left:

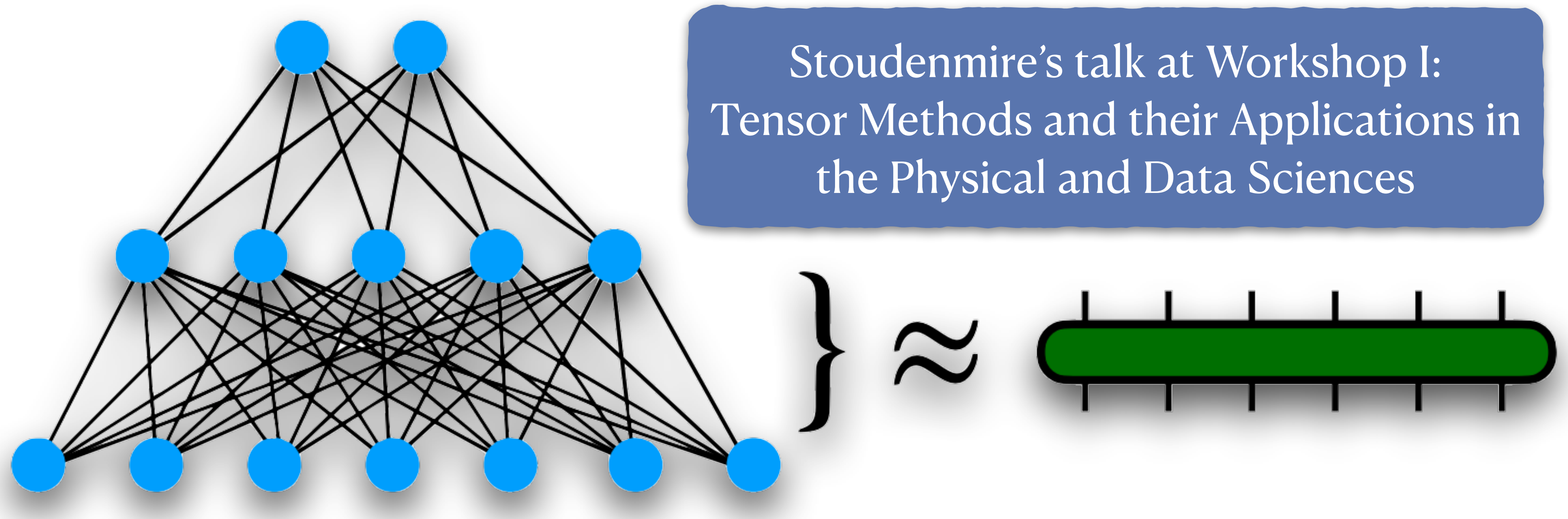


Node I Node II

Why TNs "might" perform well in classification tasks?

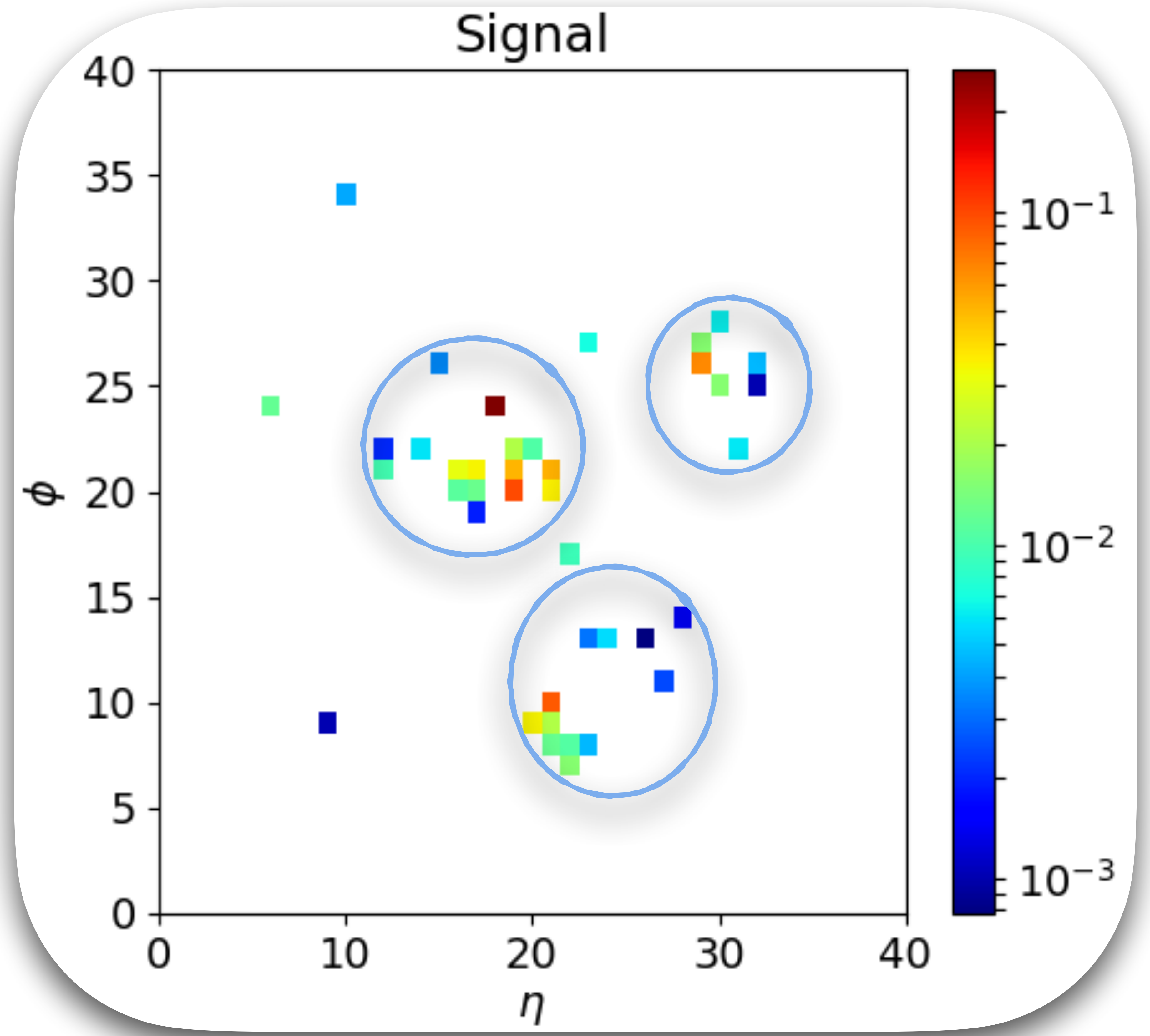
Garipov, Podoprikhin, Novikov, Vetrov
arXiv:1611.03214

Stoudenmire's talk at Workshop I: Tensor Methods and their Applications in the Physical and Data Sciences



Not in this talk!

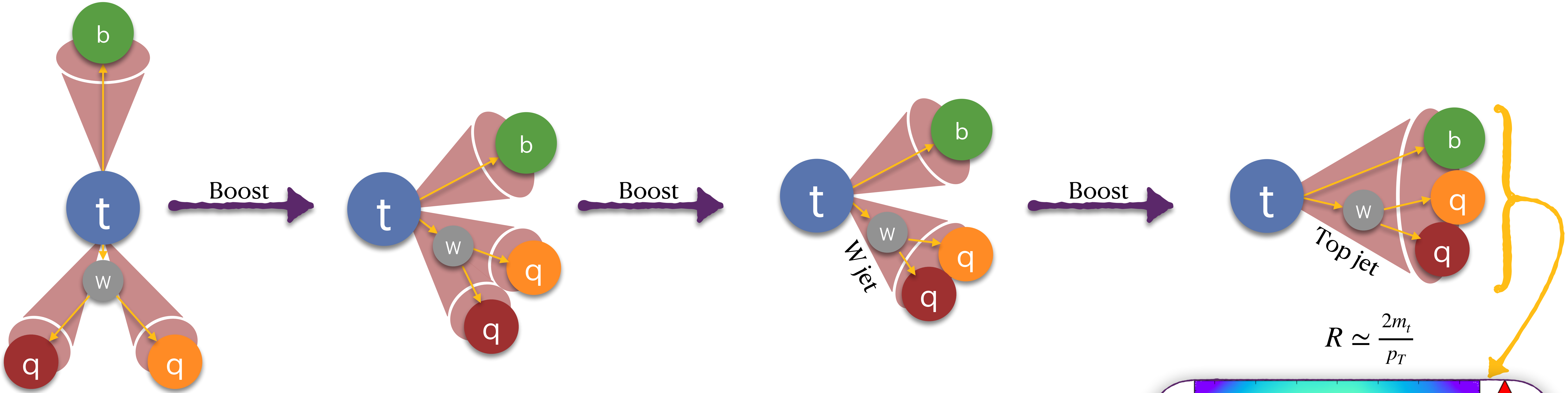
Kasieczka *et. al.* SciPost'19



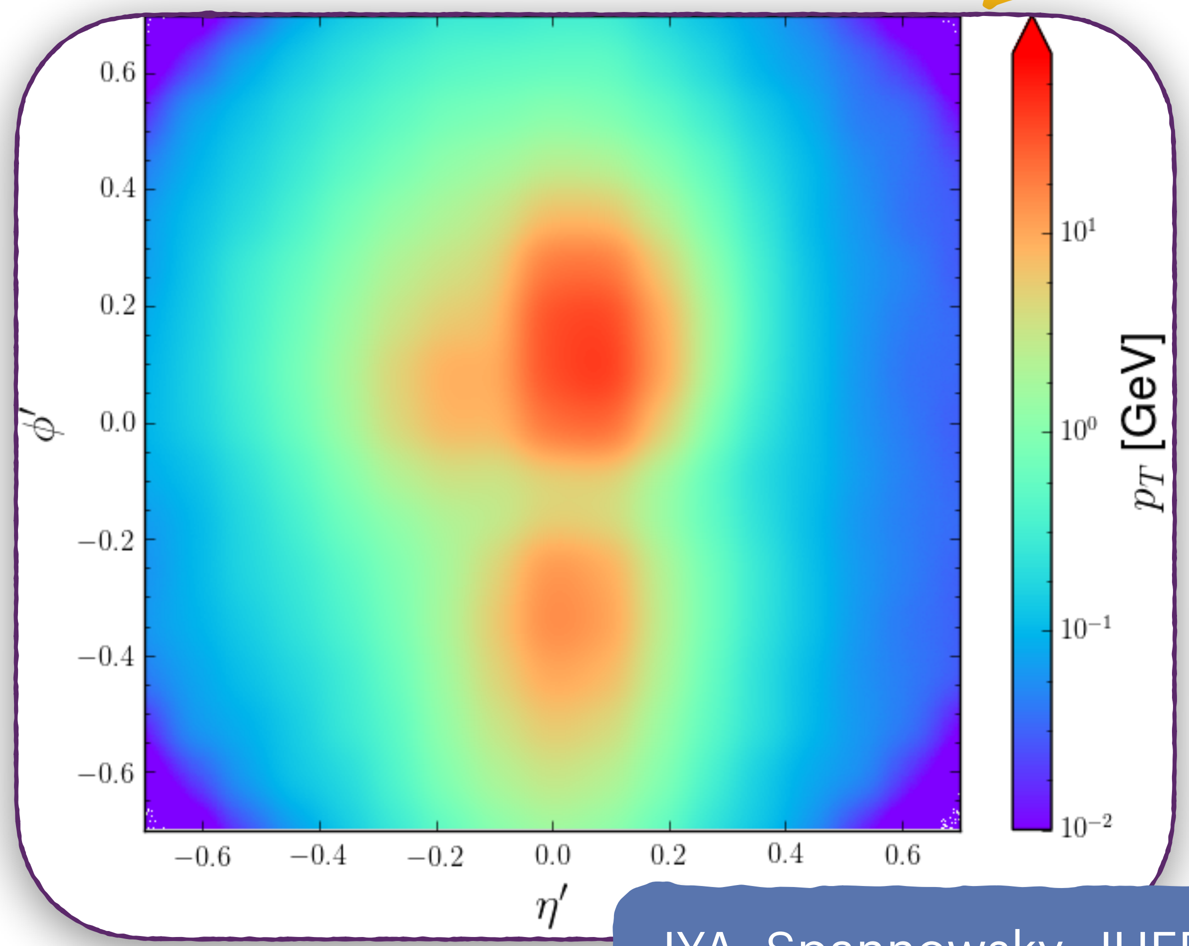
- The range of a node in a Tensor Network bounded by its bond dimension.
- Tensor Networks can capture local "anomalies".
- Jets can produce localized clusters!!

Hello World of HEP-ML: Top Tagging

Why Top Quarks?



$$R \simeq \frac{2m_t}{p_T}$$

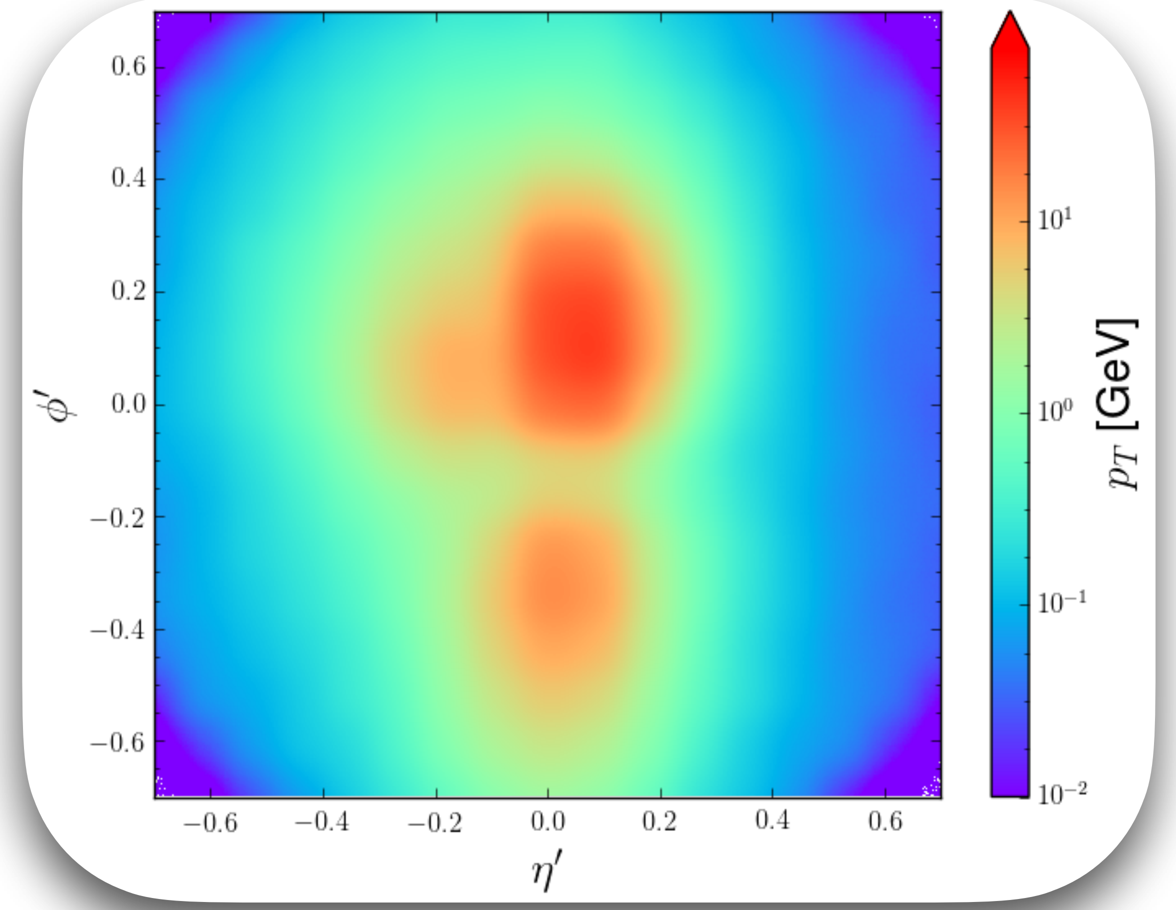


JYA, Spannowsky; JHEP '21

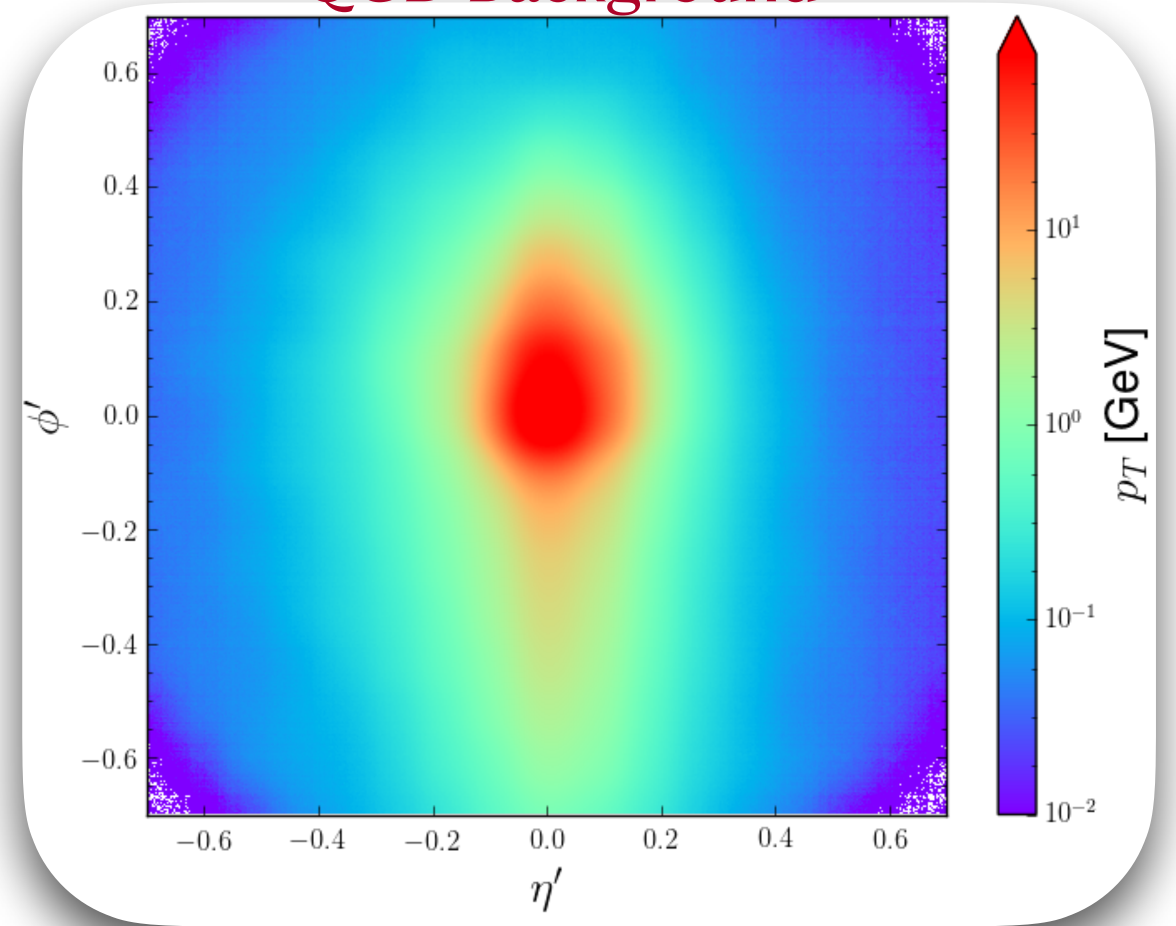
- ❖ With the increased boost factor, jets (top decay products) are getting more collimated.
- ❖ Hadronic top tagging tools: Mass grooming and filtering, Pruning, Trimming, Soft Drop Tagger, Mass Drop Tagger, HEPTopTagger, Machine Learning

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.

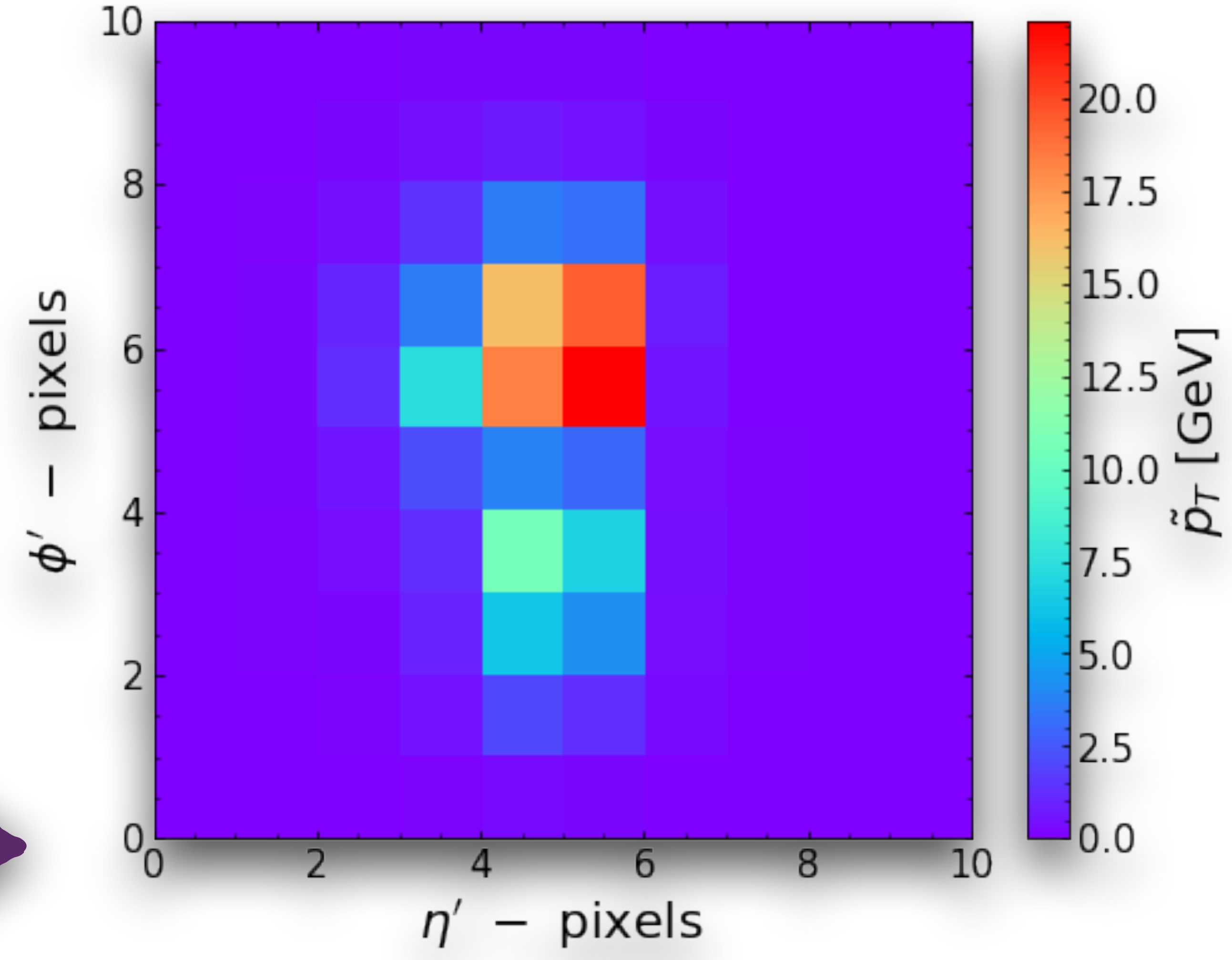
Similar preprocess, based on CNN:

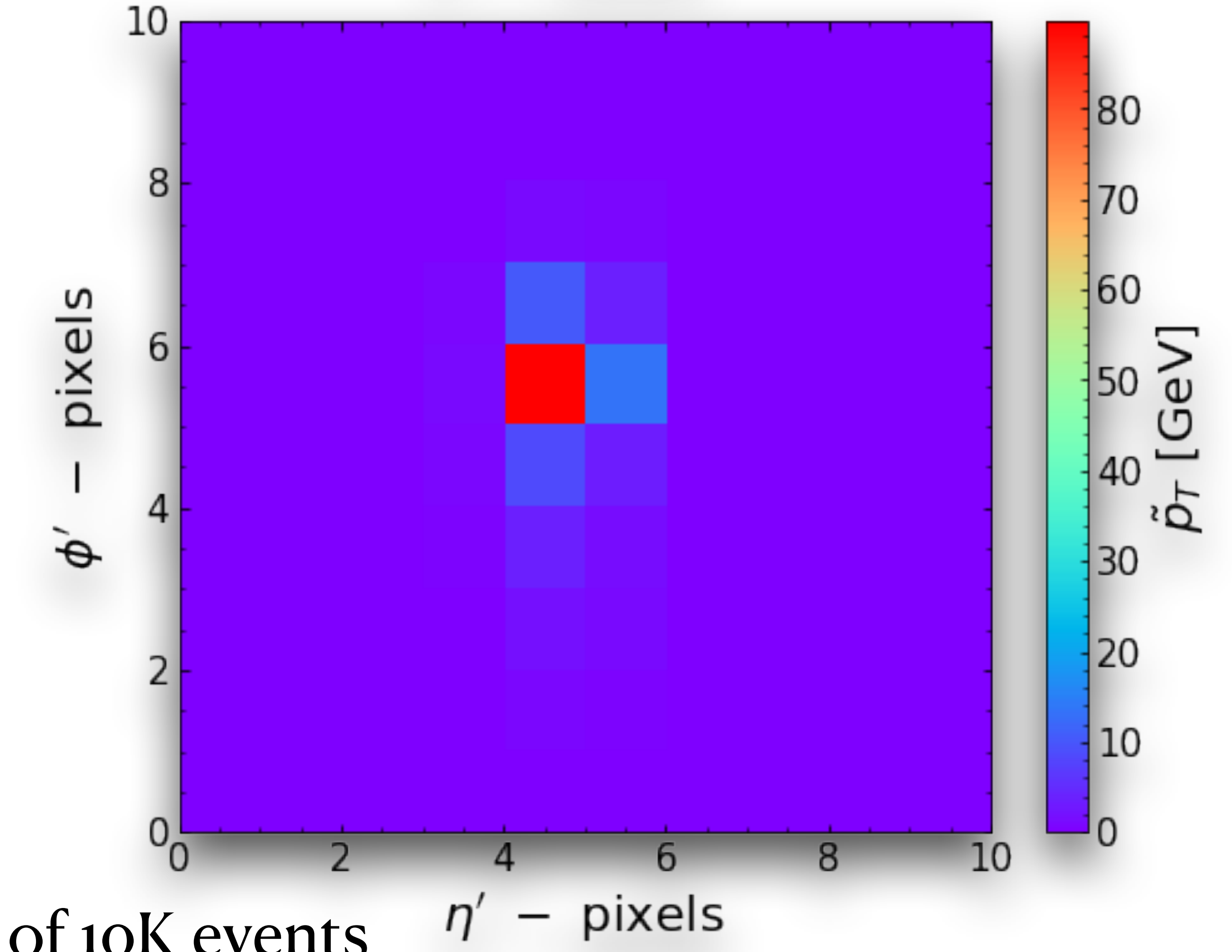
Kasieczka, Plehn, Russell, Schell; JHEP '17

Macaluso, Shih; JHEP '18

JYA, Spannowsky; JHEP '21

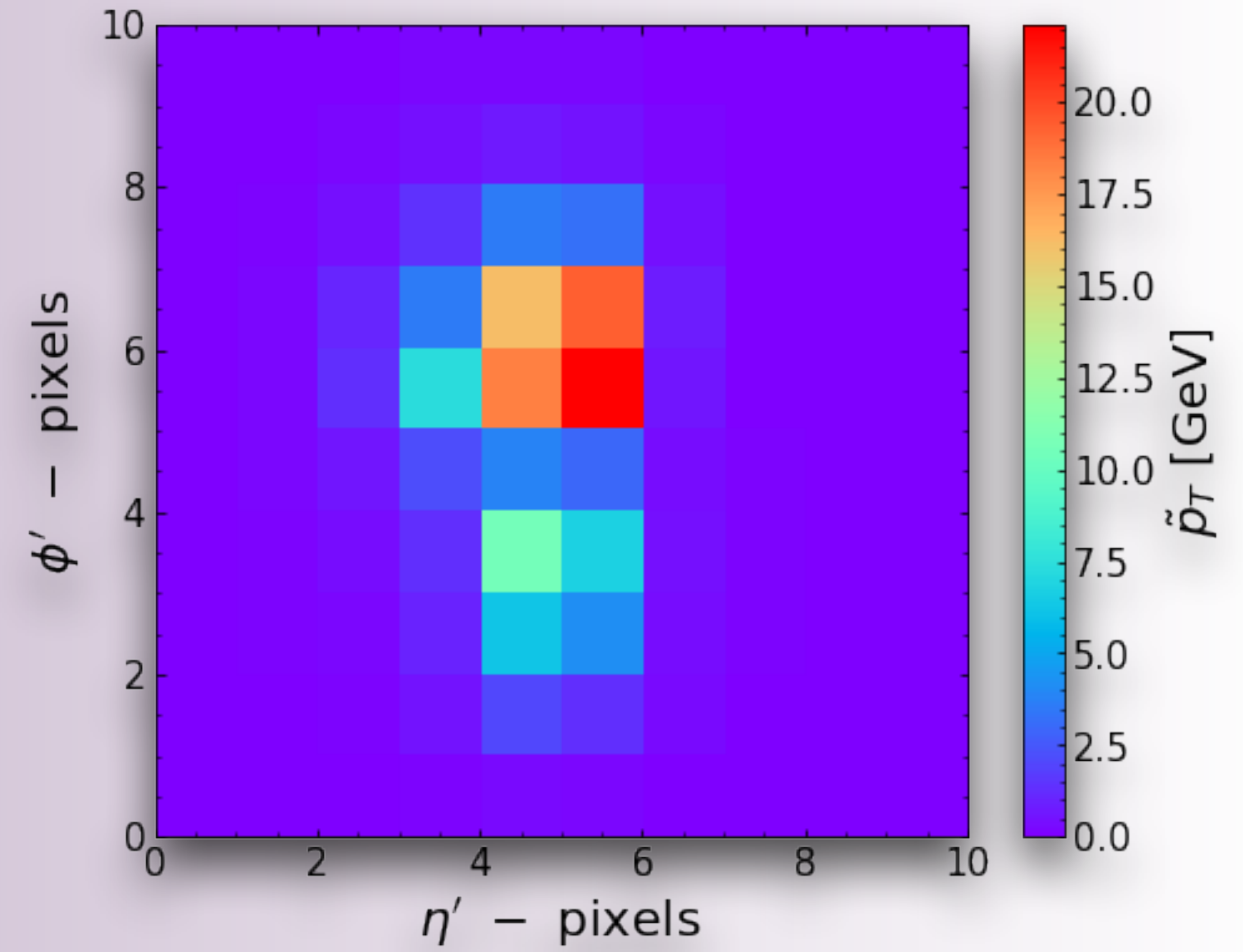
Crop & downsample

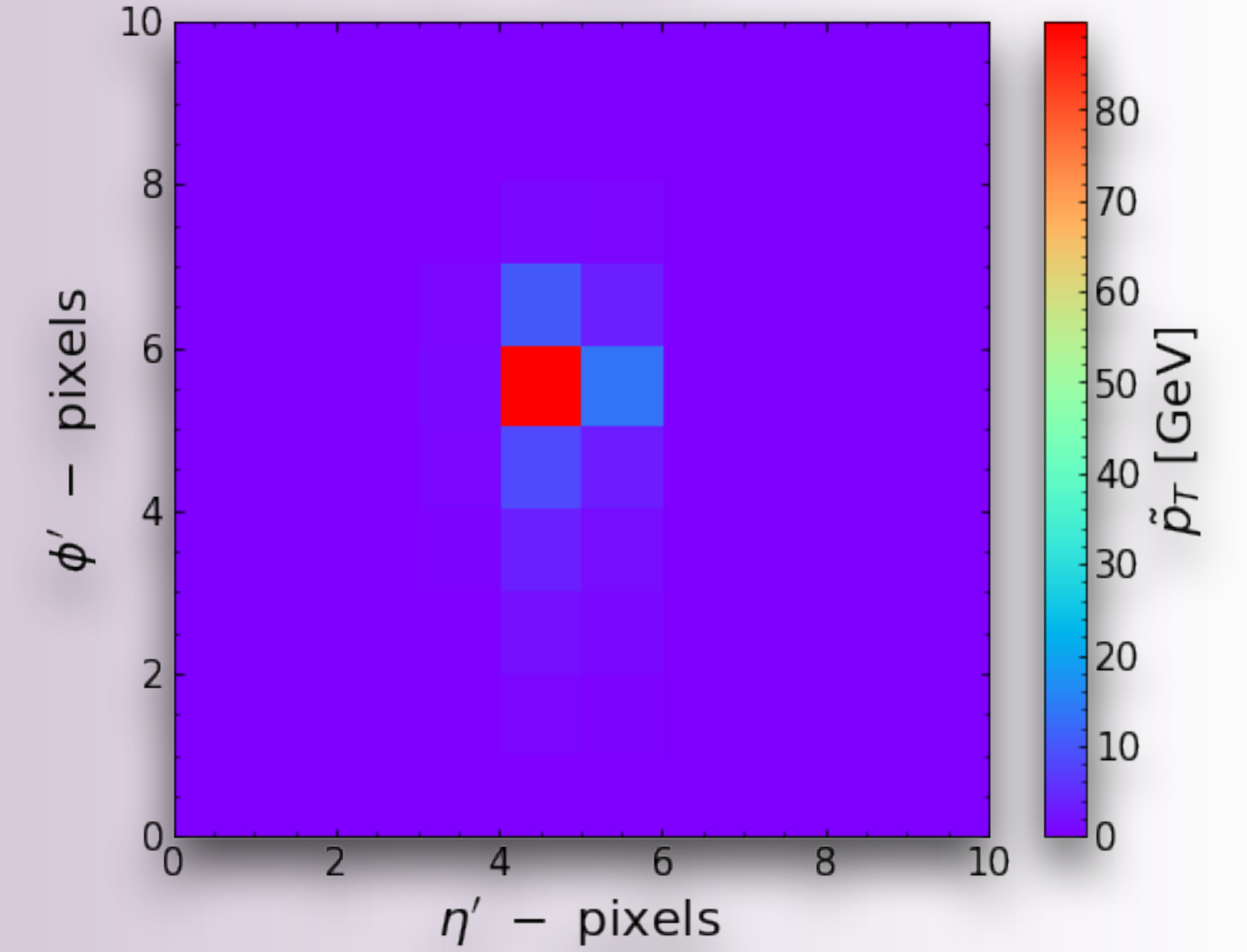




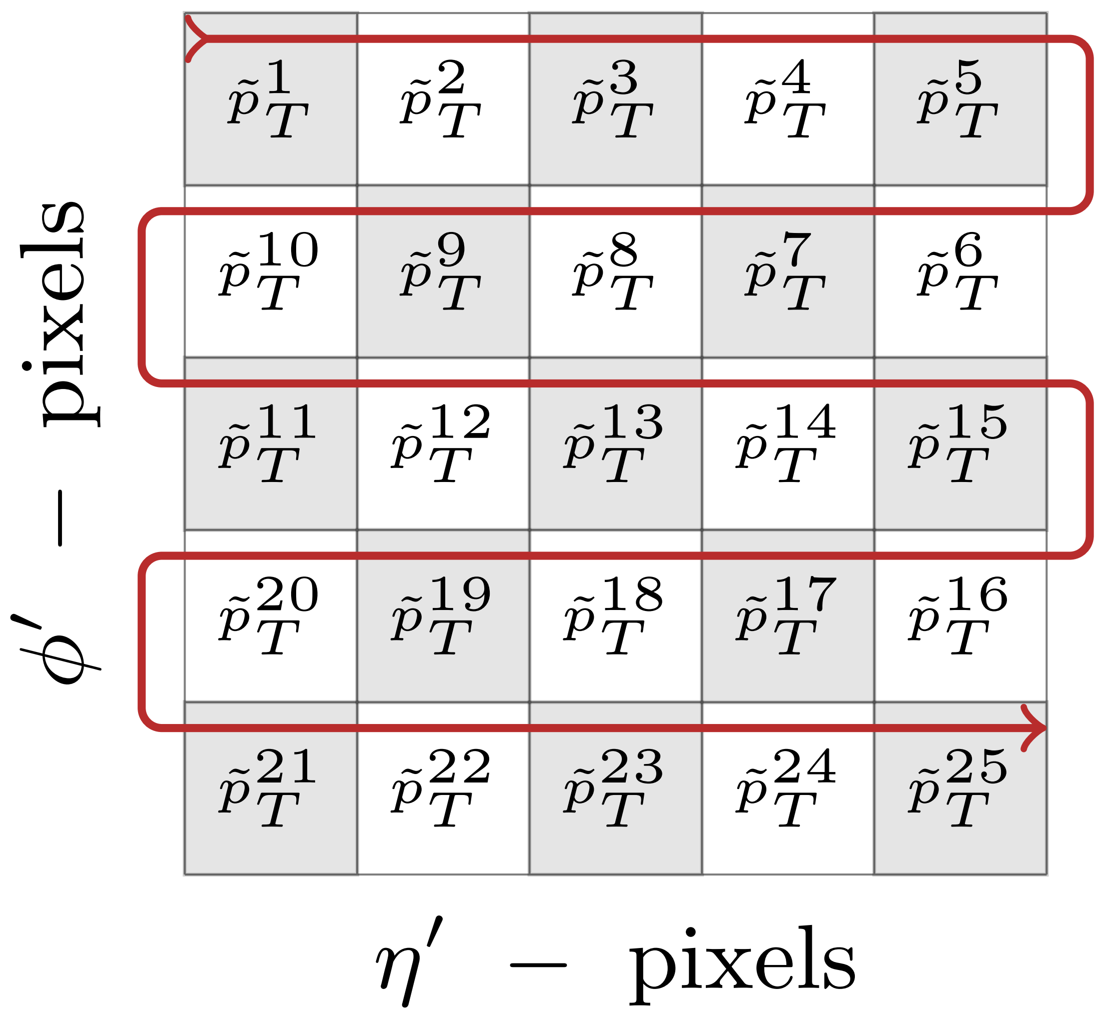
Mean of 10K events

Top Tagging through MPS





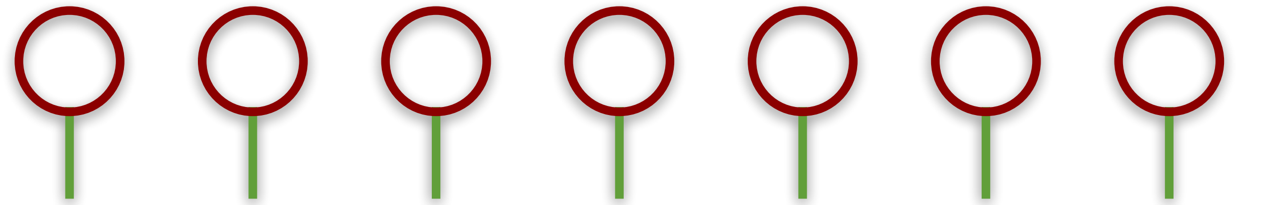
η - based ordering



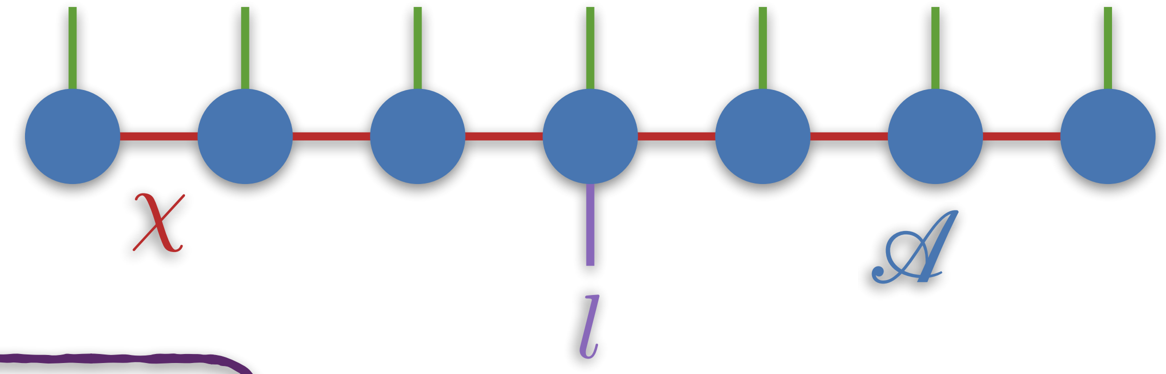
Data Embedding

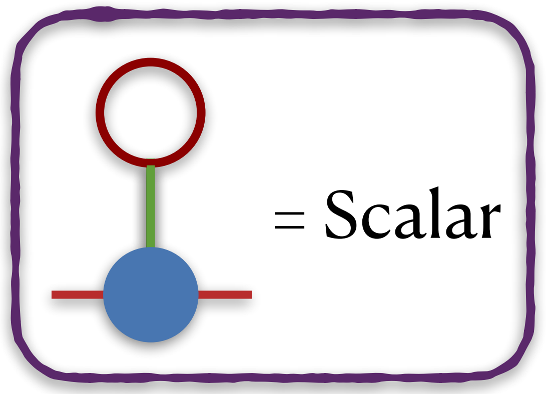
$$\Phi^{p_1\cdots p_n}(\mathbf{x}) = \bigotimes_i^N \phi^{p_i}(\tilde{p}_T^i)$$

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



$$= \Phi^{p_1\cdots p_n}(\mathbf{x})$$



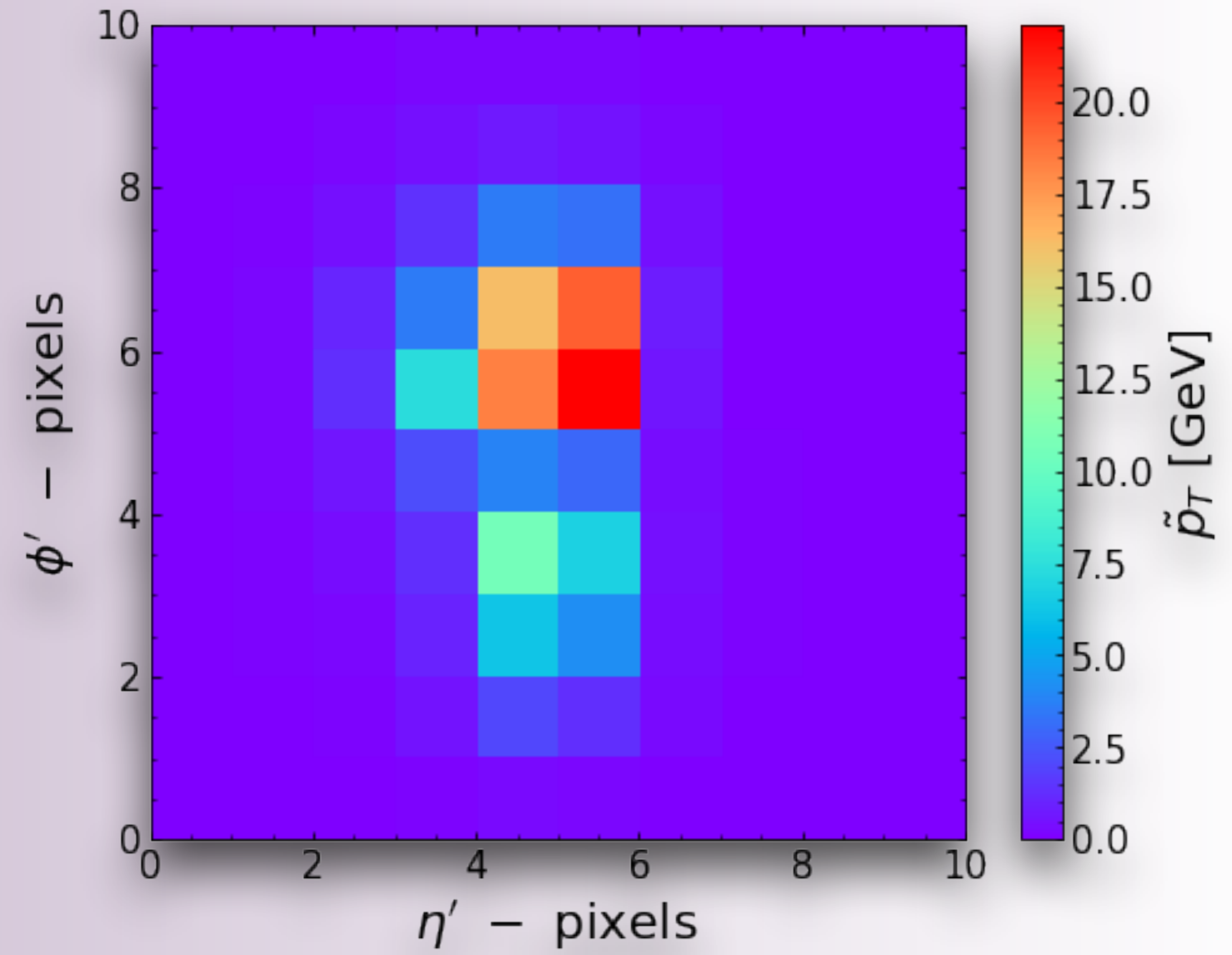


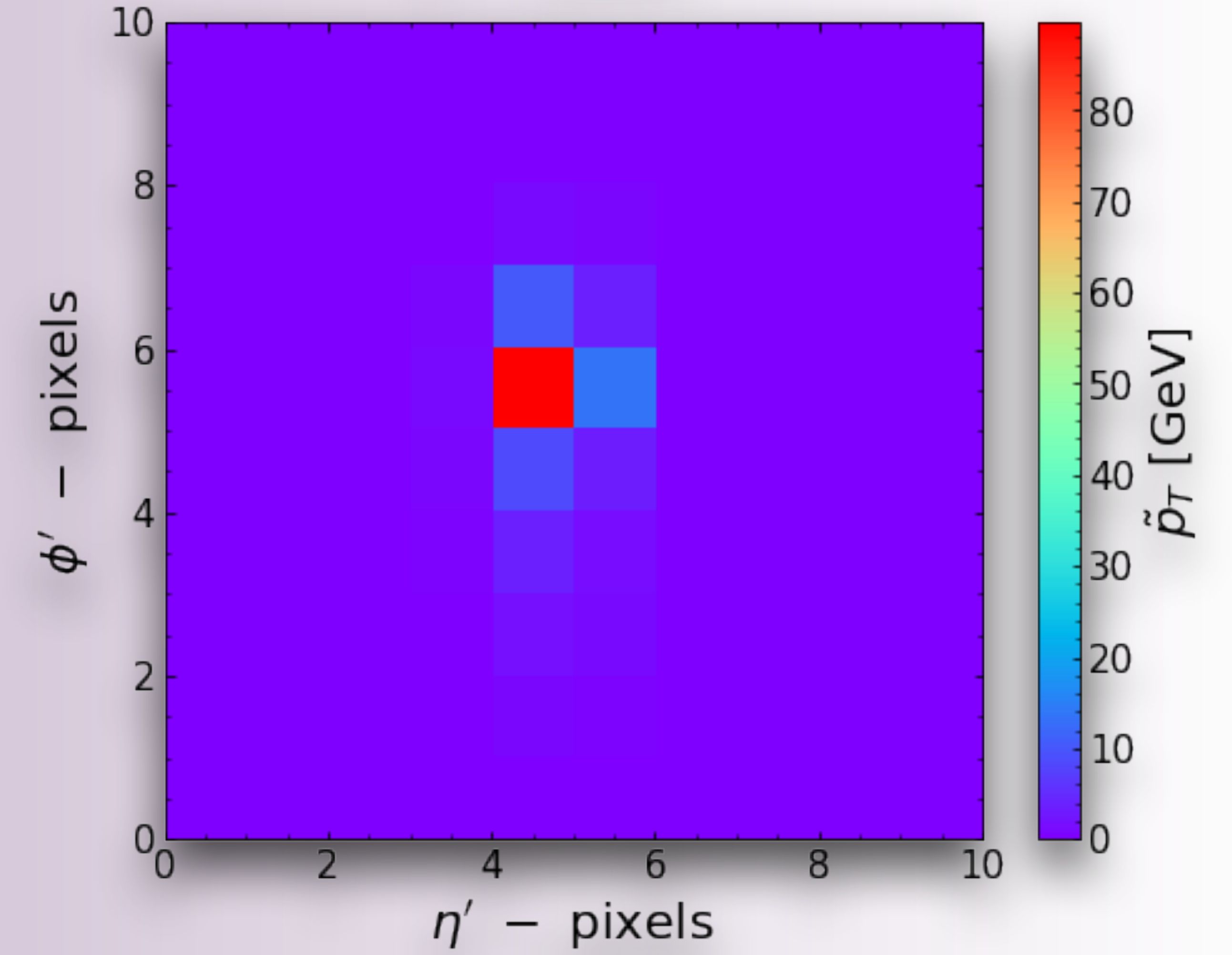
Network Initial Condition

- 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



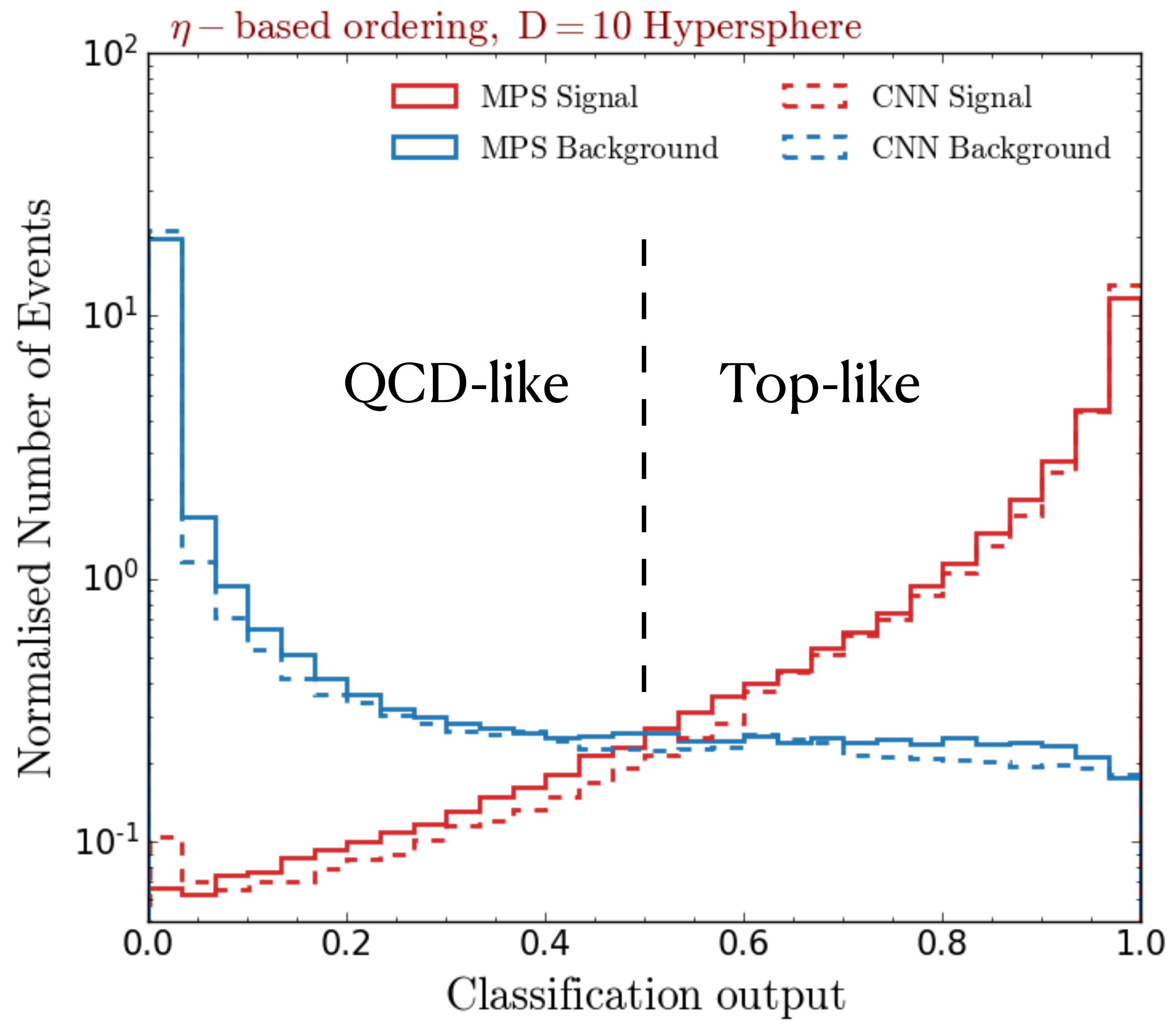


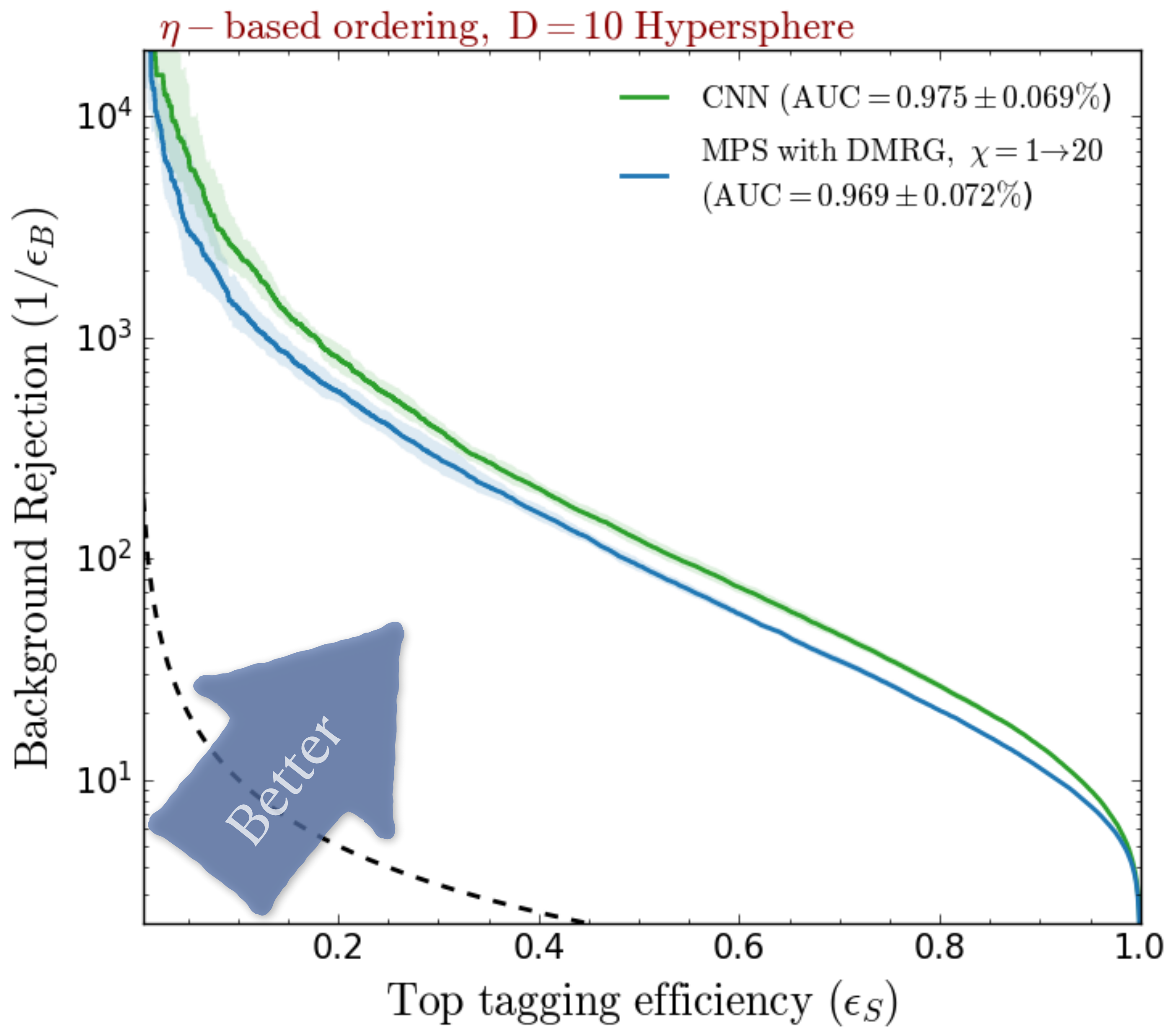
Assumptions & Requirements

- 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

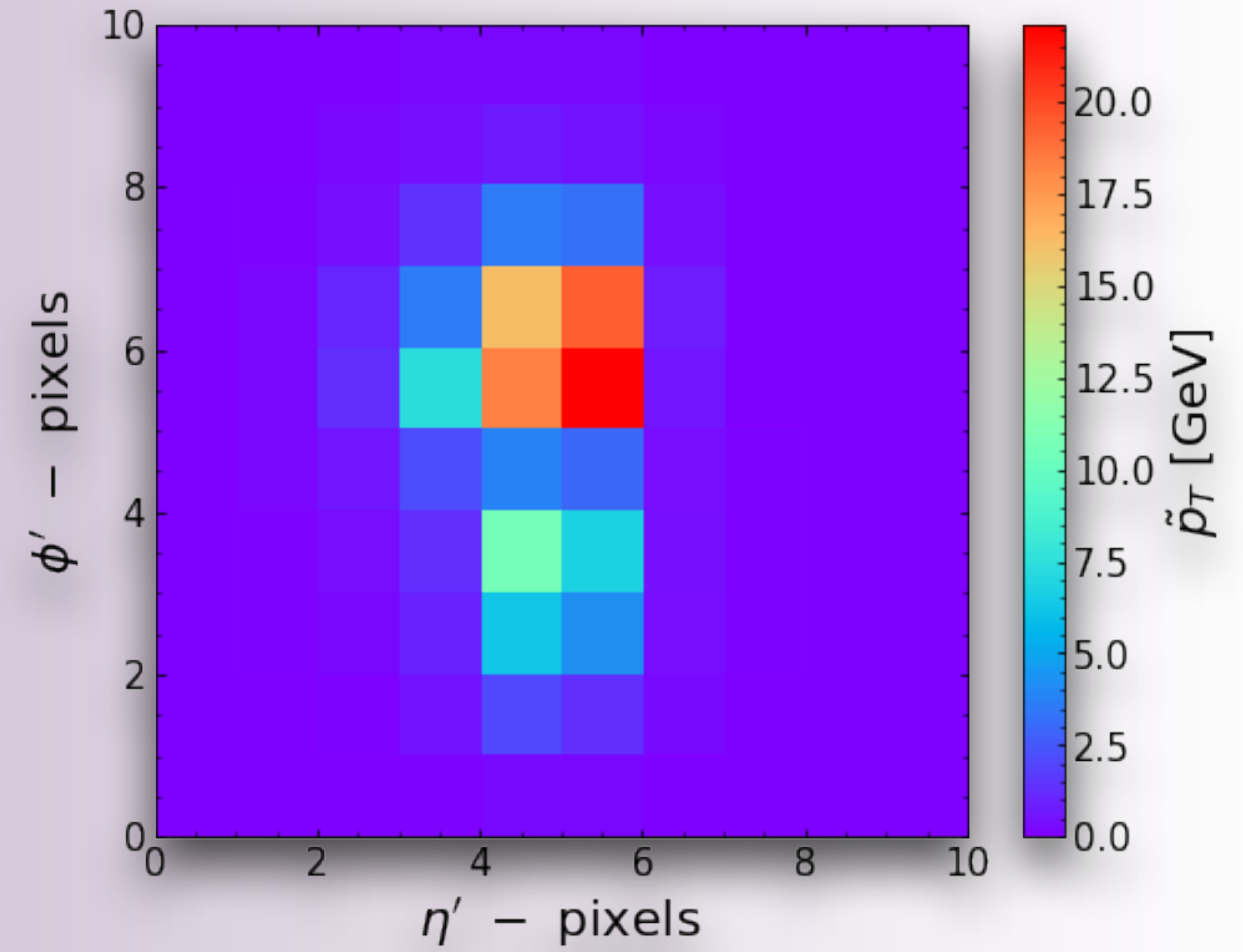


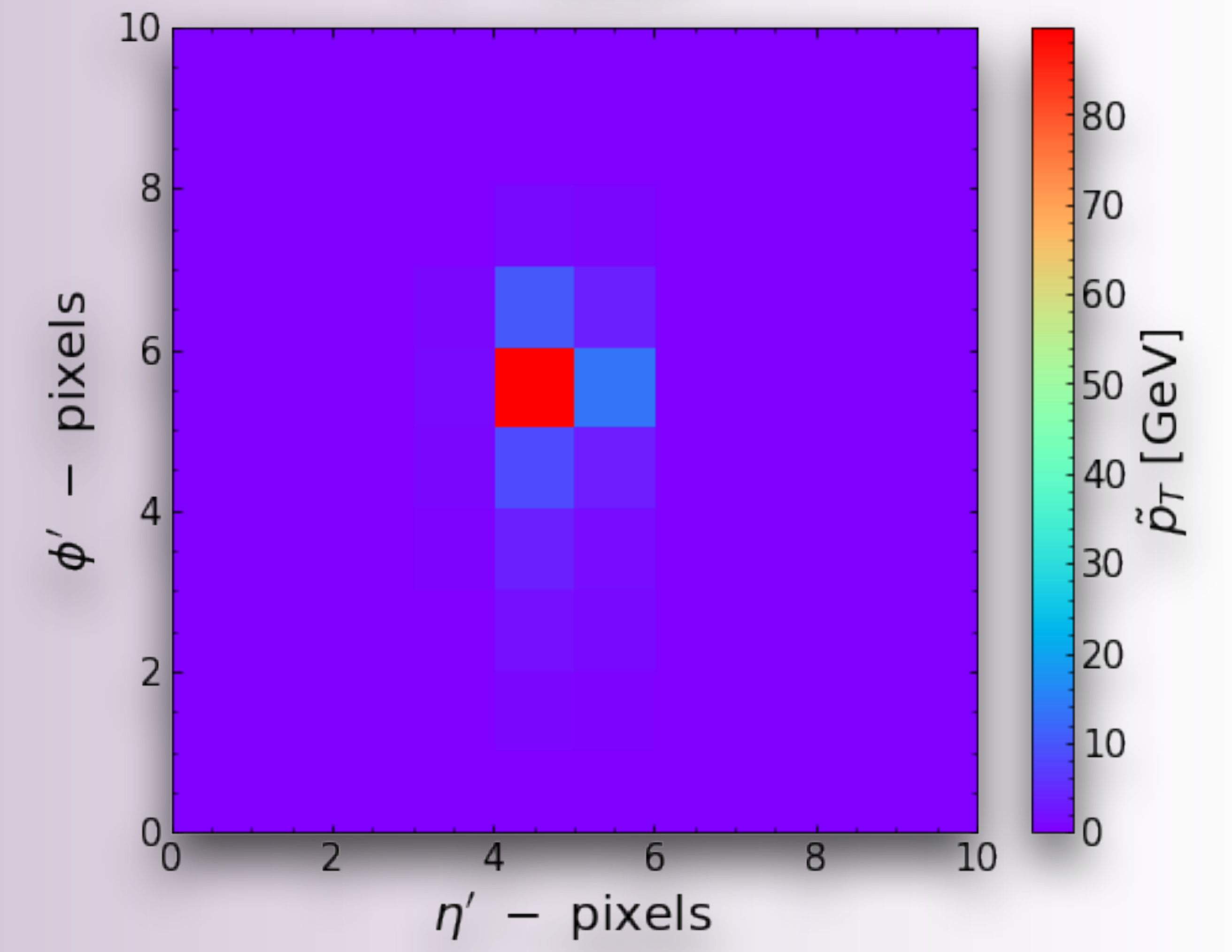


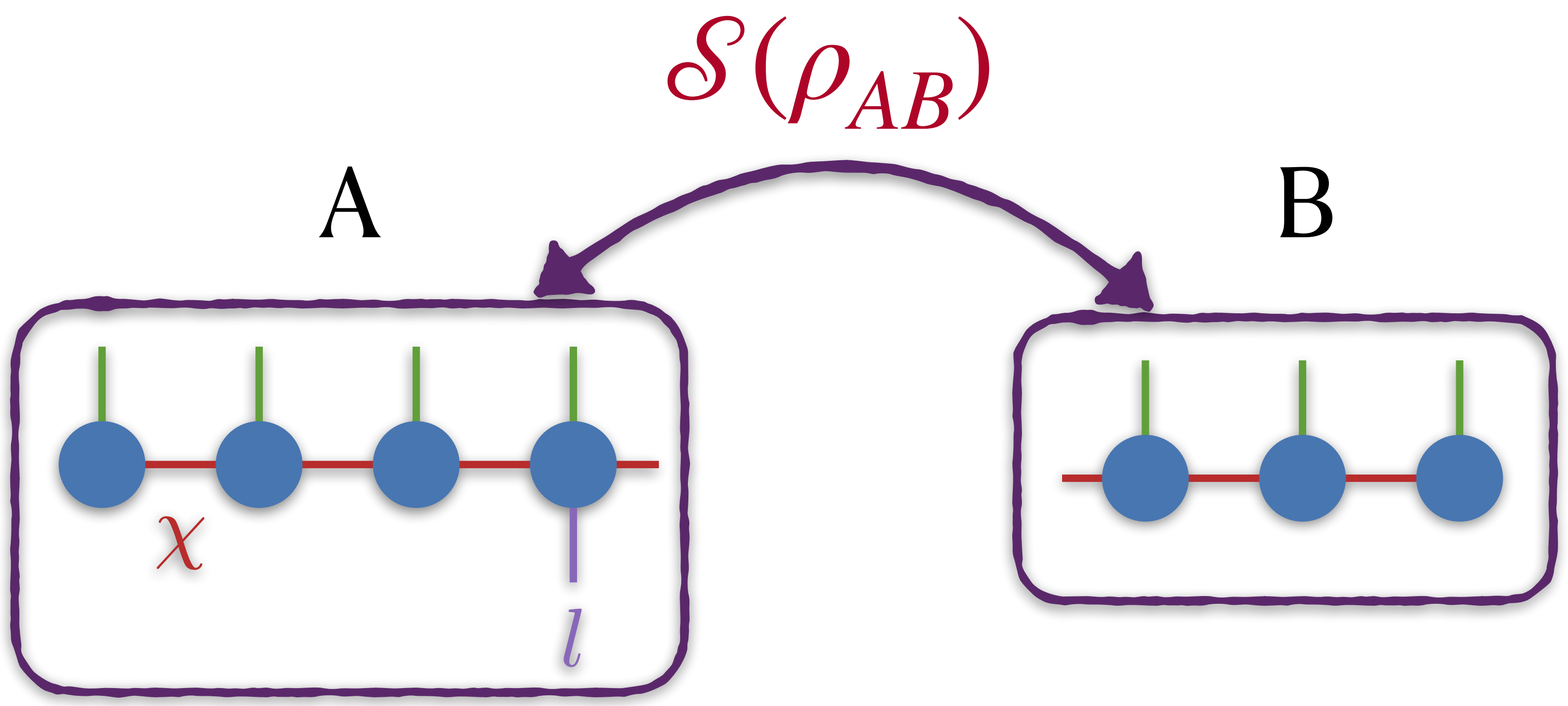
Question:

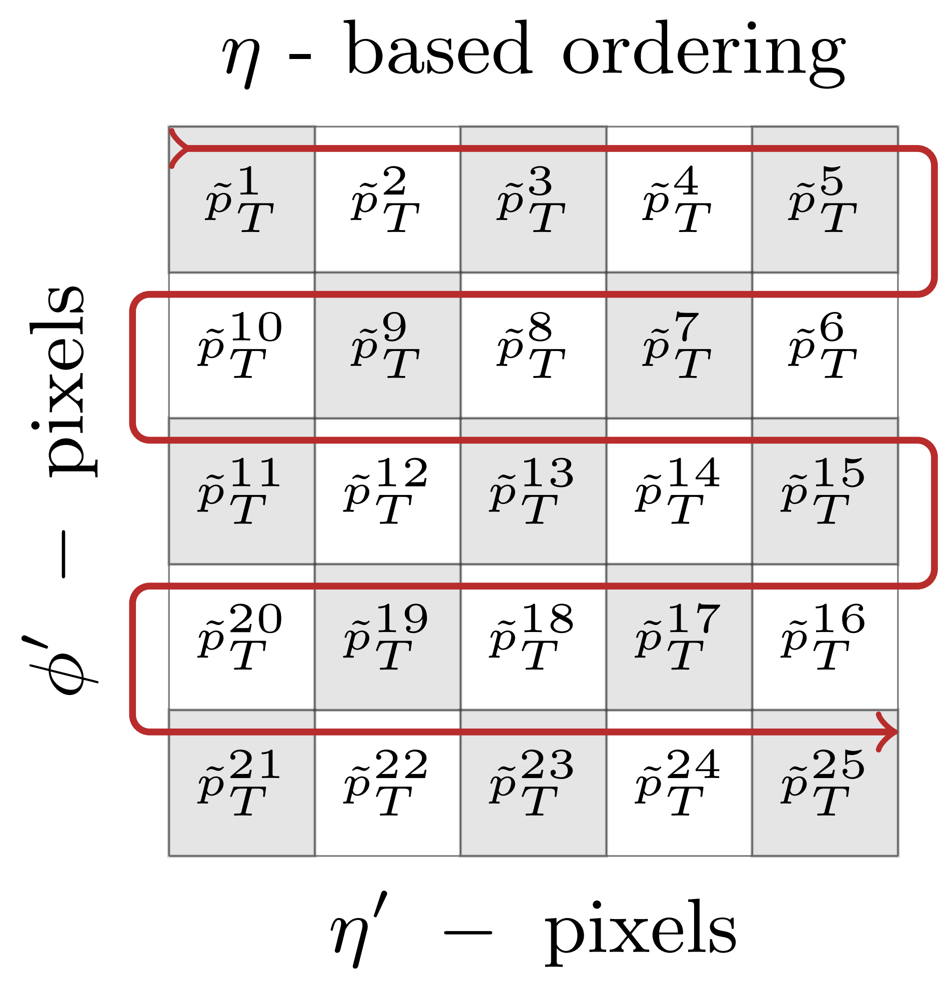
If I give you a trained network and no data, what can you tell me about the data?

Top Tagging through MPS



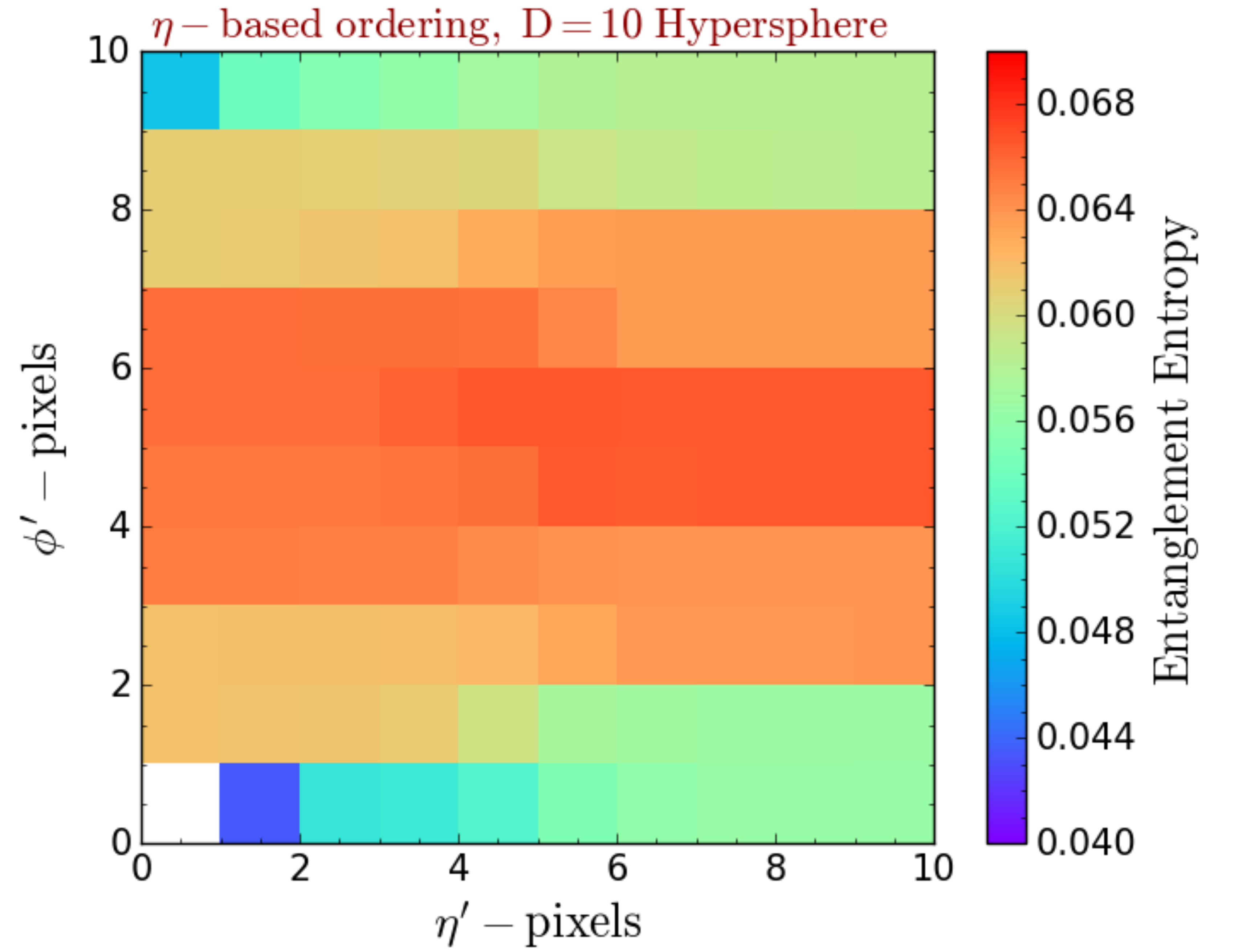




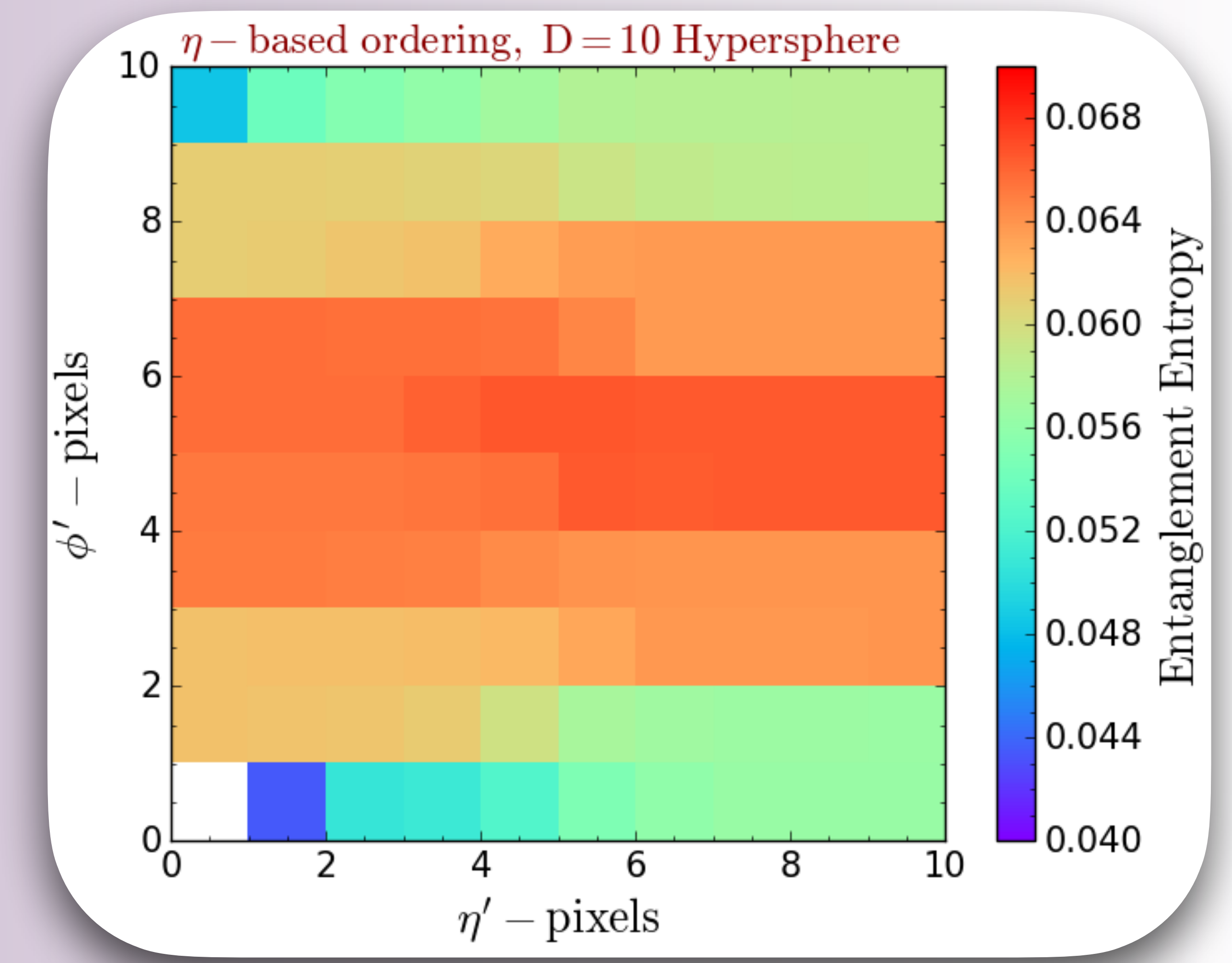


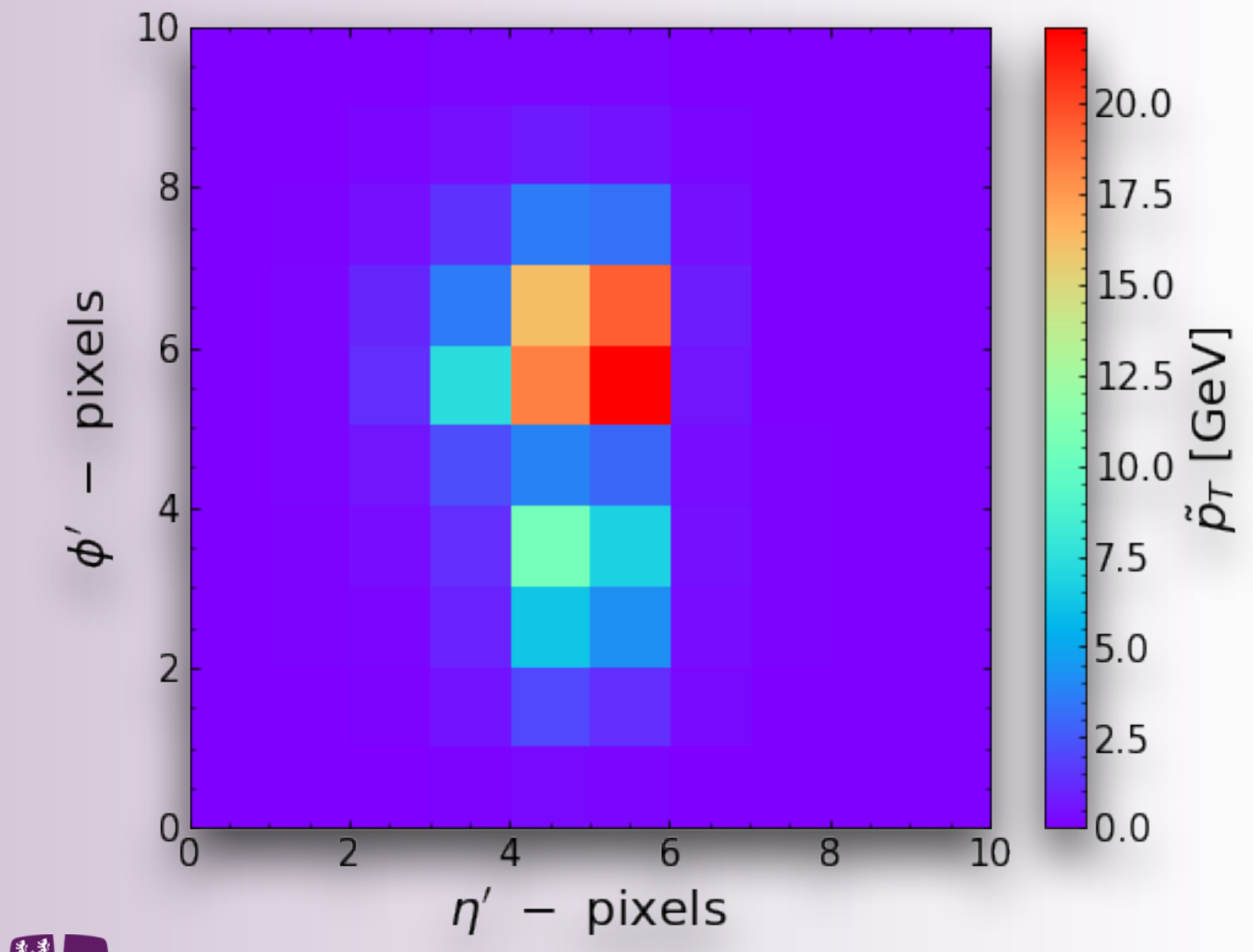
Entanglement Entropy

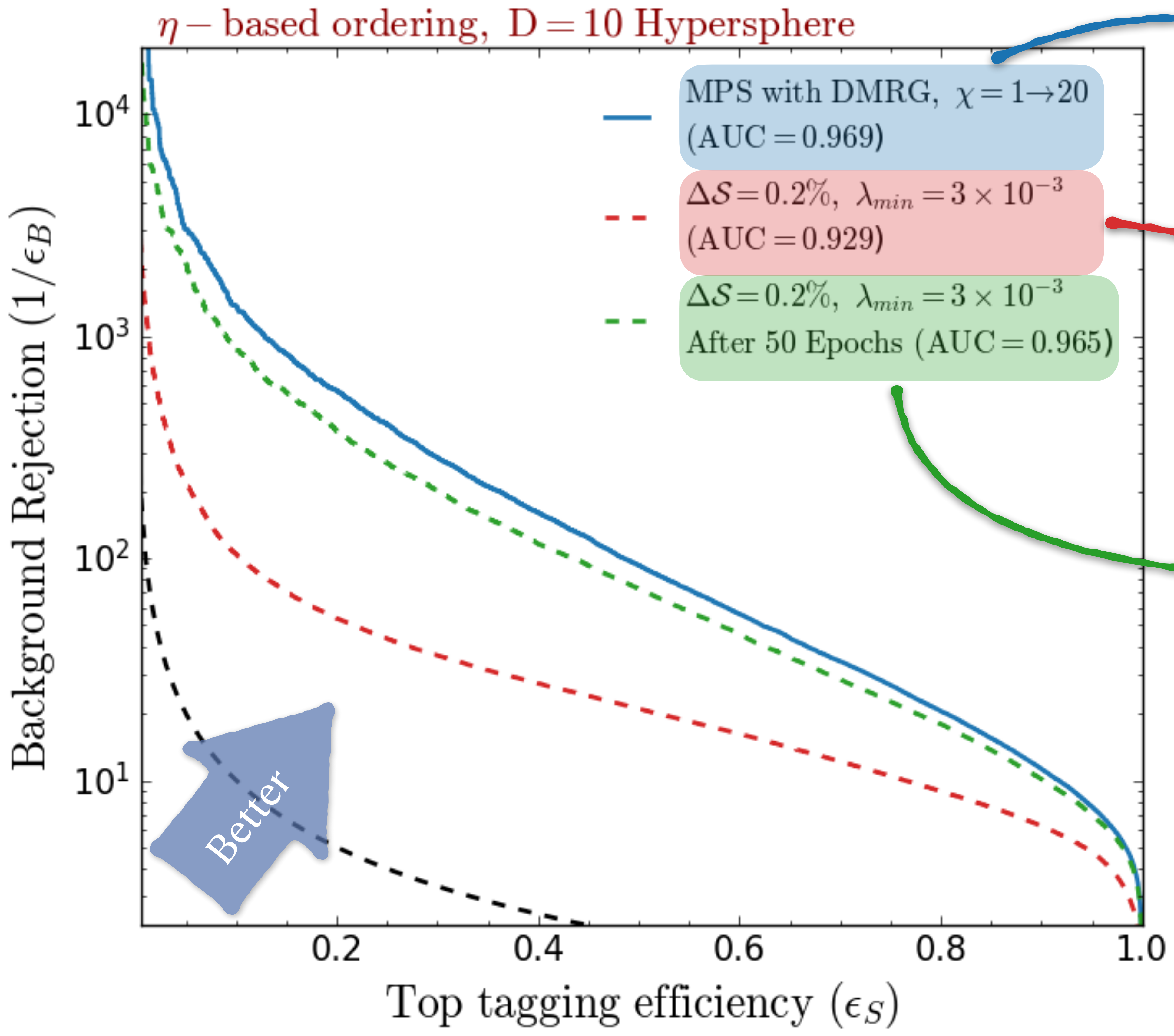
$$\mathcal{S}(\rho) = -\operatorname{Tr}[\rho \log_2 \rho] \quad ; \quad \rho := |\Psi\rangle\langle\Psi|$$



Top Tagging through MPS







100 pixels, 390500 trainable parameters

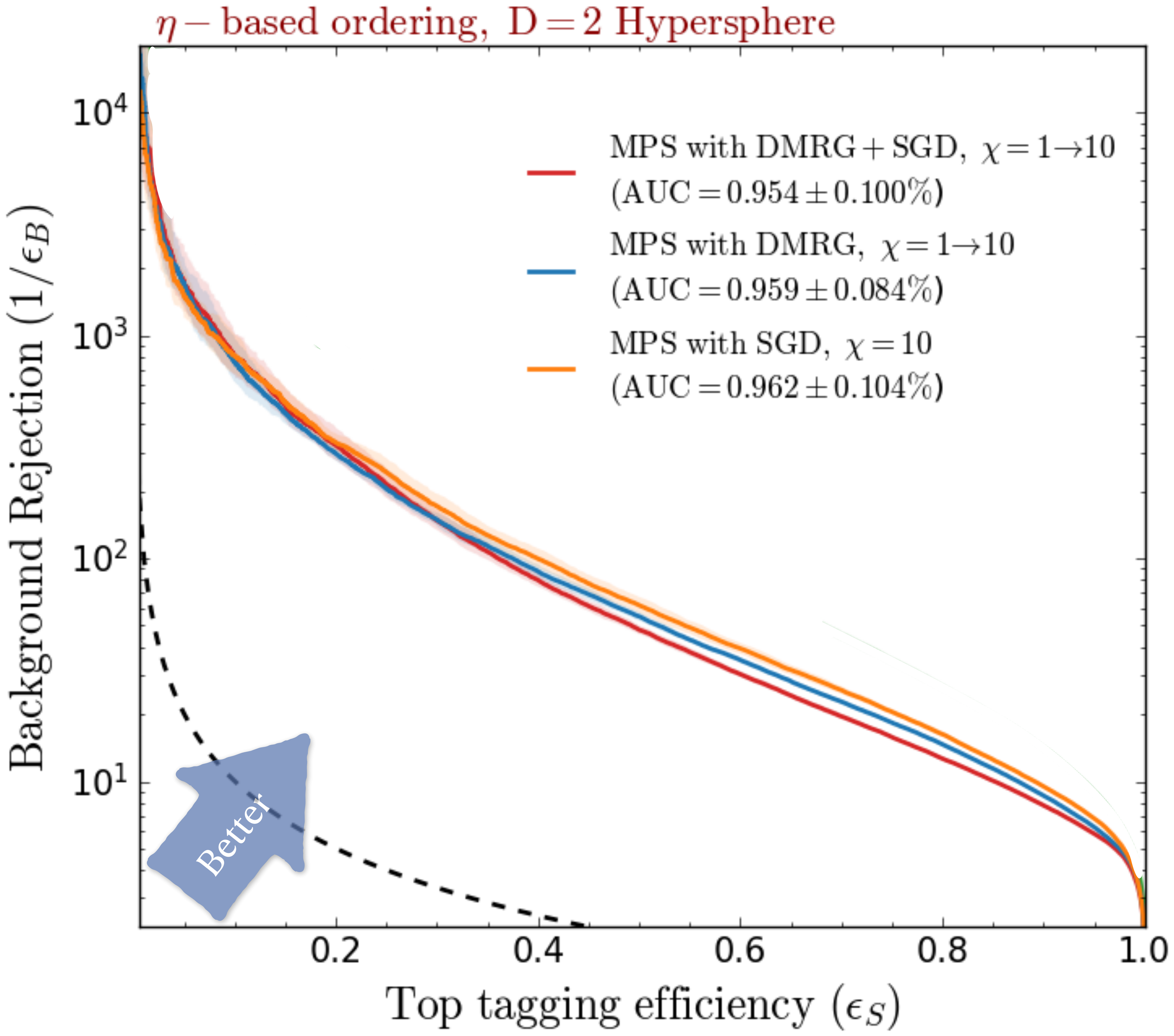
54 pixels, 43410 trainable parameters

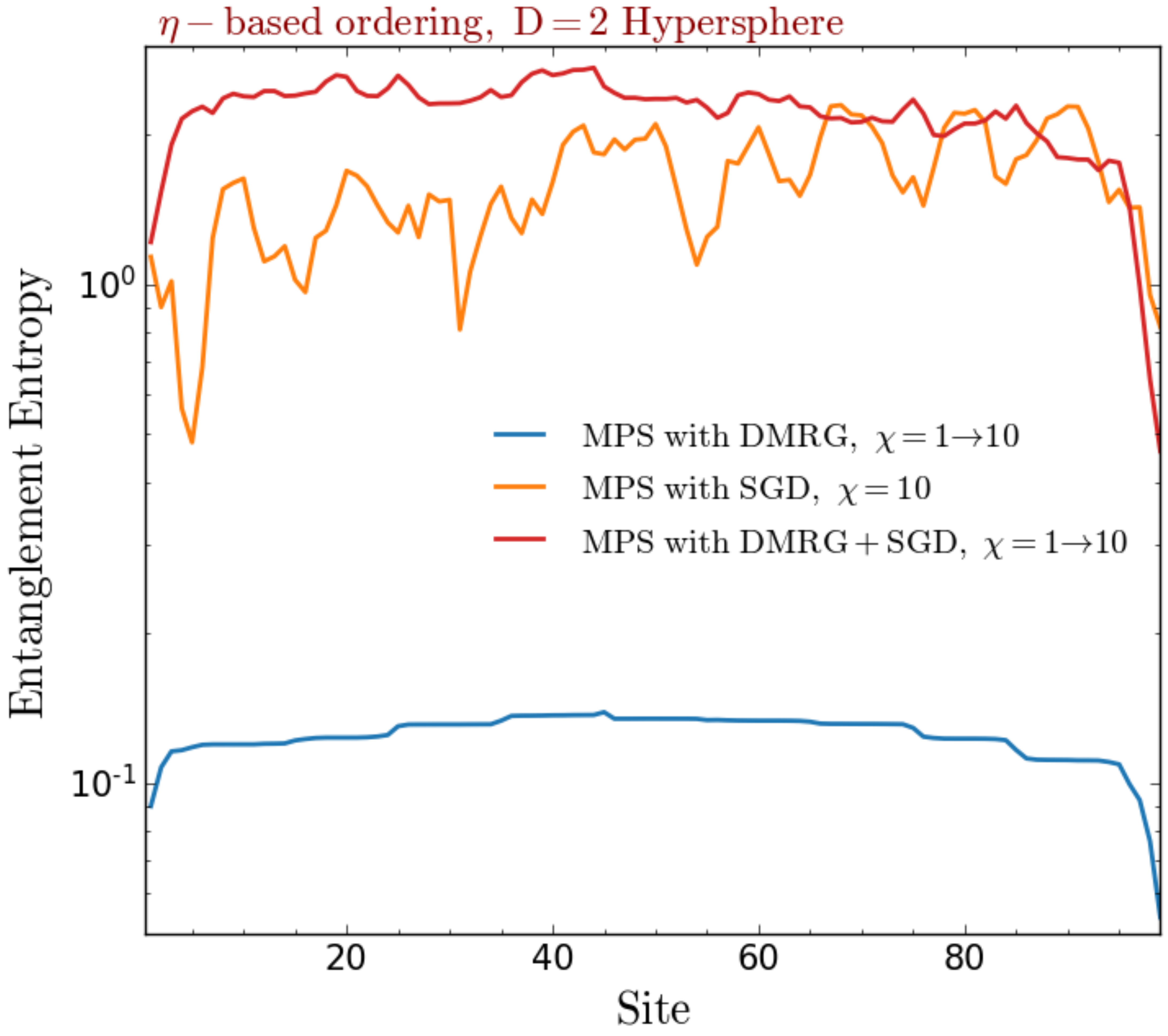
54 pixels, 34160 trainable parameters

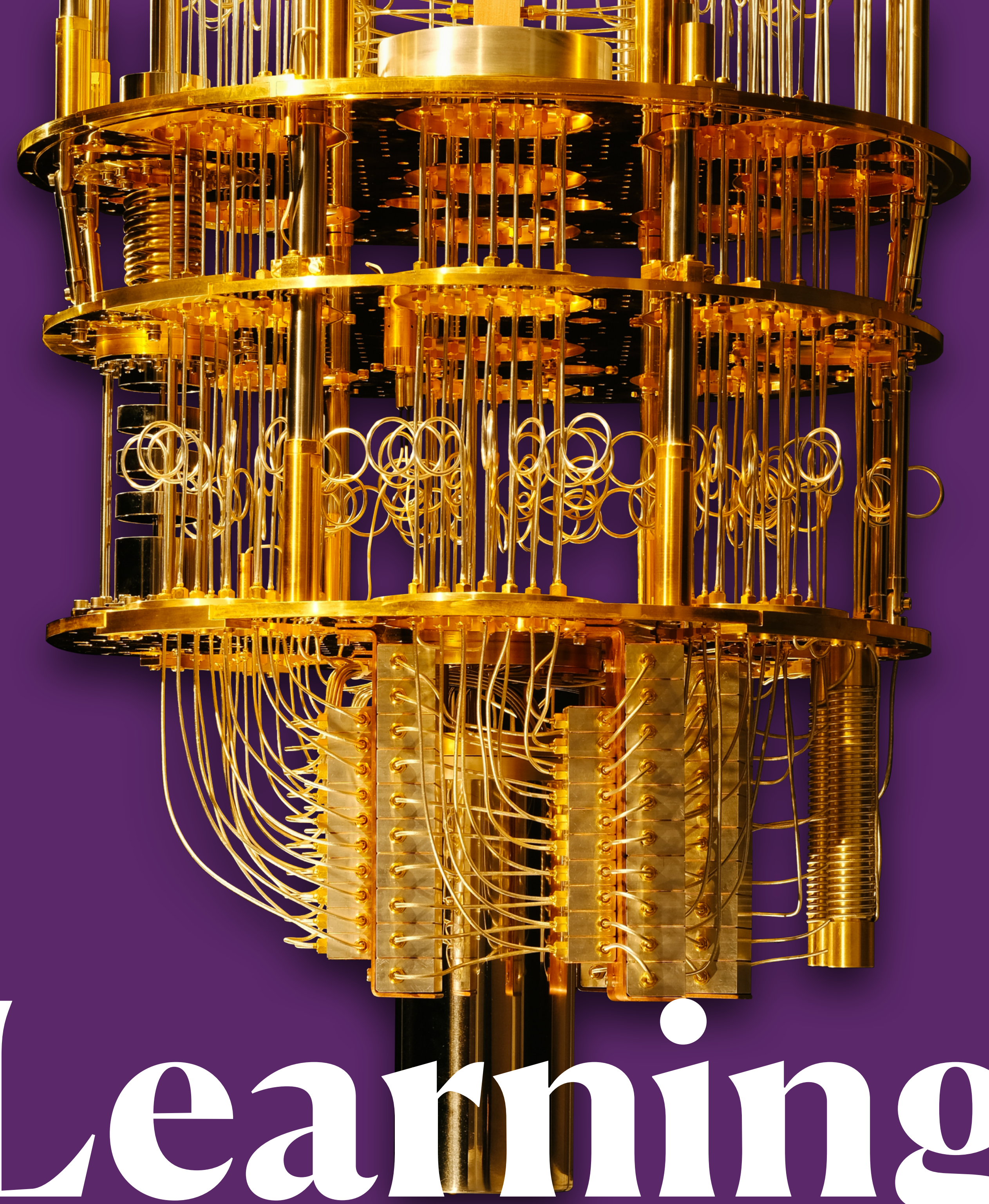
91% Reduction

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

How about good old SGD?





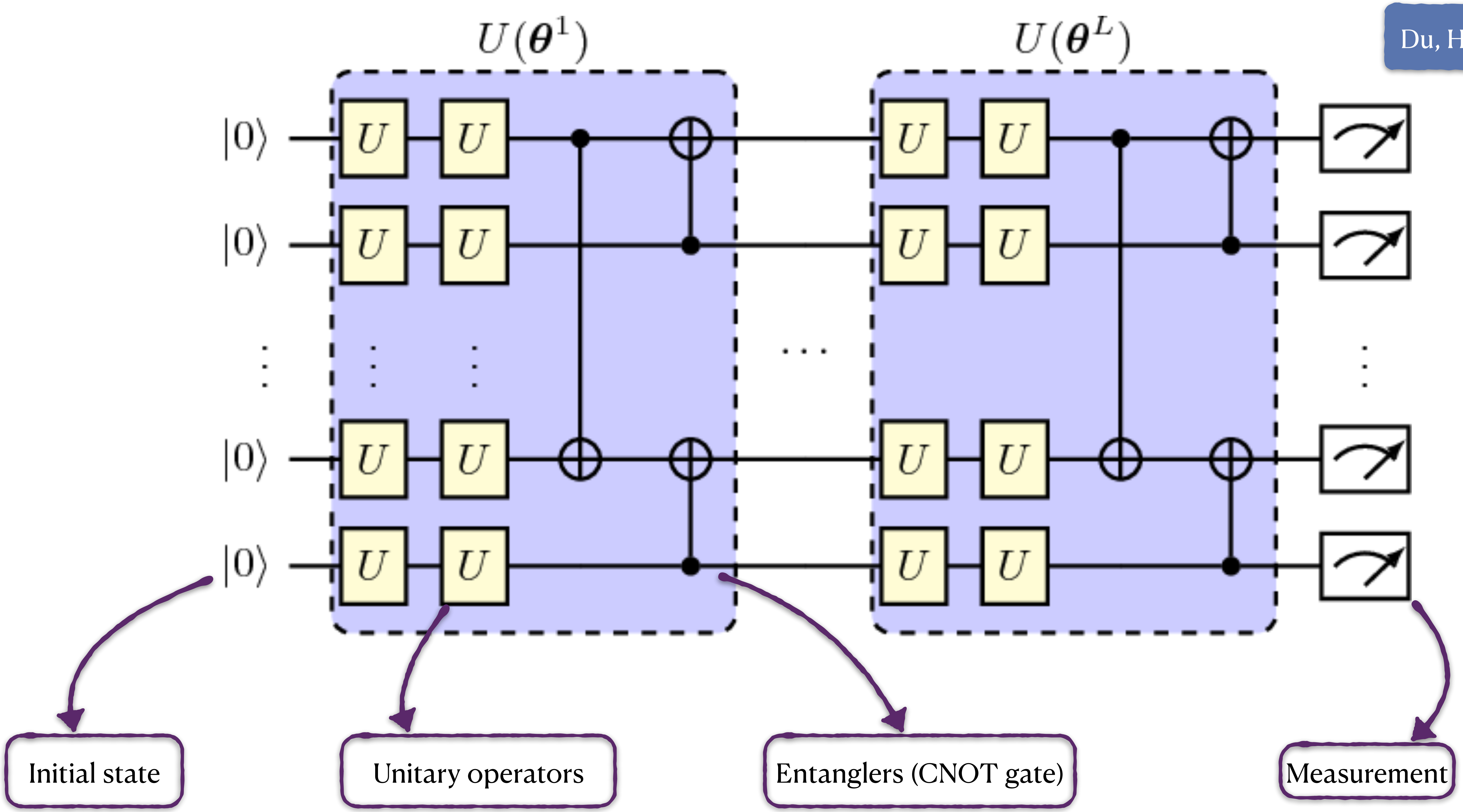


Quantum Machine Learning

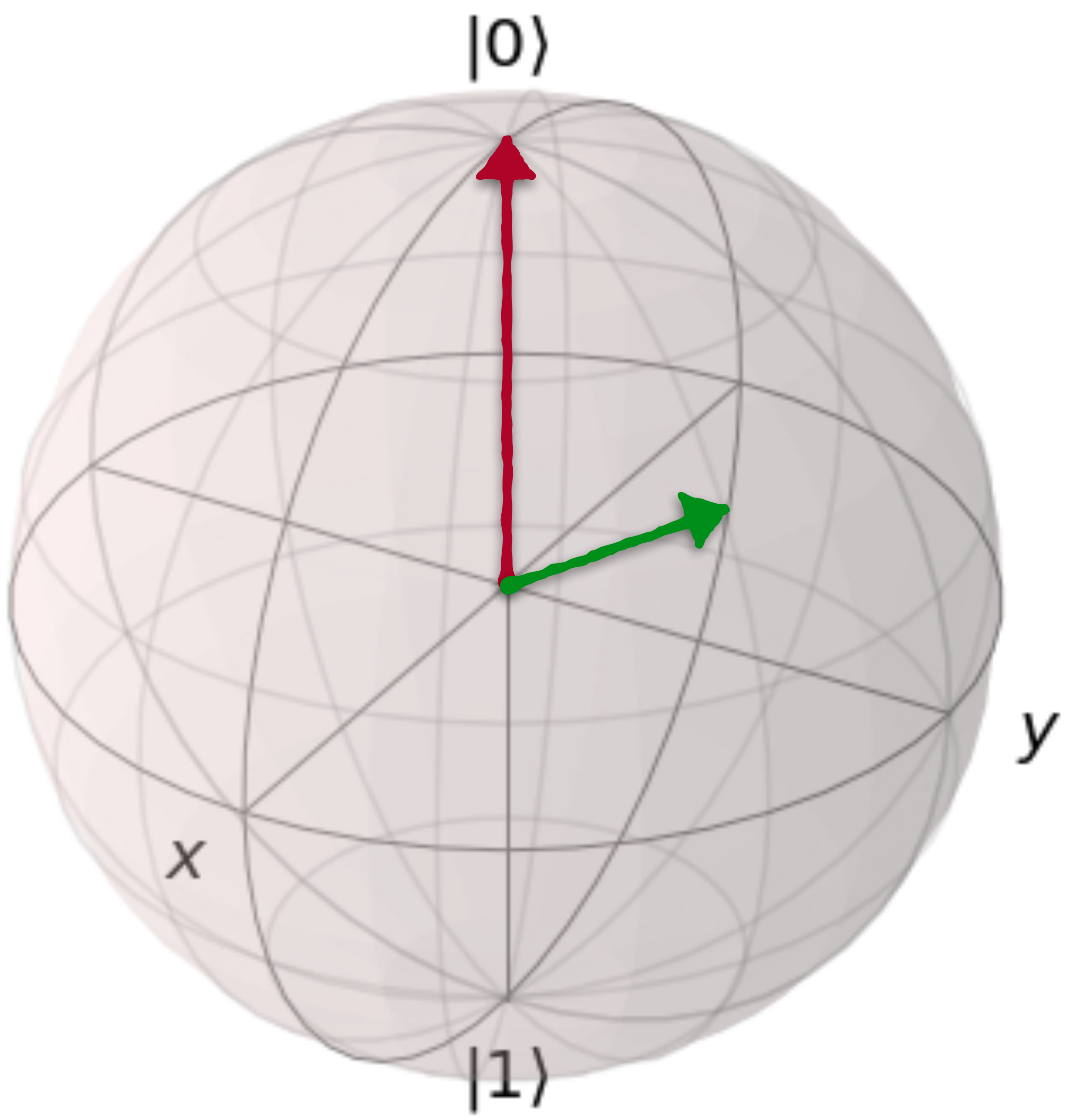
Yes, up to now, it was "classical"...

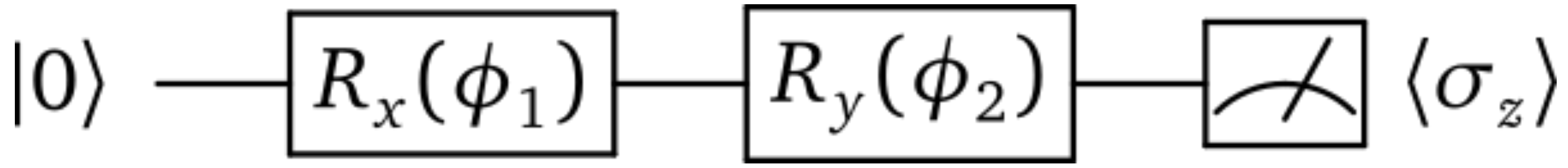
Introduction Vol. III: Quantum Circuits

Du, Hsieh, Liu, Tao; Phys. Rev. `20



Introduction Vol. III: Quantum Circuits





$$|\Phi\rangle = R_y(\phi_1)\ \left\{ R_x(\phi_2)\,|\,0\rangle \right\}$$

$$R_x\,|\,0\rangle = e^{-i\phi\sigma_x}\,|\,0\rangle = \begin{bmatrix} \cos\left(\frac{\phi}{2}\right) & -i\sin\left(\frac{\phi}{2}\right) \\ -i\sin\left(\frac{\phi}{2}\right) & \cos\left(\frac{\phi}{2}\right) \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \cos\left(\frac{\phi}{2}\right)|\,0\rangle - i\sin\left(\frac{\phi}{2}\right)|\,1\rangle$$

$$\langle \Phi\,|\,\sigma_z\,|\,\Phi\rangle = \langle 0\,|\,R_x^\dagger(\phi_2)R_y^\dagger(\phi_1)\ \sigma_z\ R_y(\phi_1)R_x(\phi_2)\,|\,0\rangle$$

$$R_y(\phi) = e^{-i\phi\sigma_y} = \begin{bmatrix} \cos\left(\frac{\phi}{2}\right) & -\sin\left(\frac{\phi}{2}\right) \\ \sin\left(\frac{\phi}{2}\right) & \cos\left(\frac{\phi}{2}\right) \end{bmatrix}$$

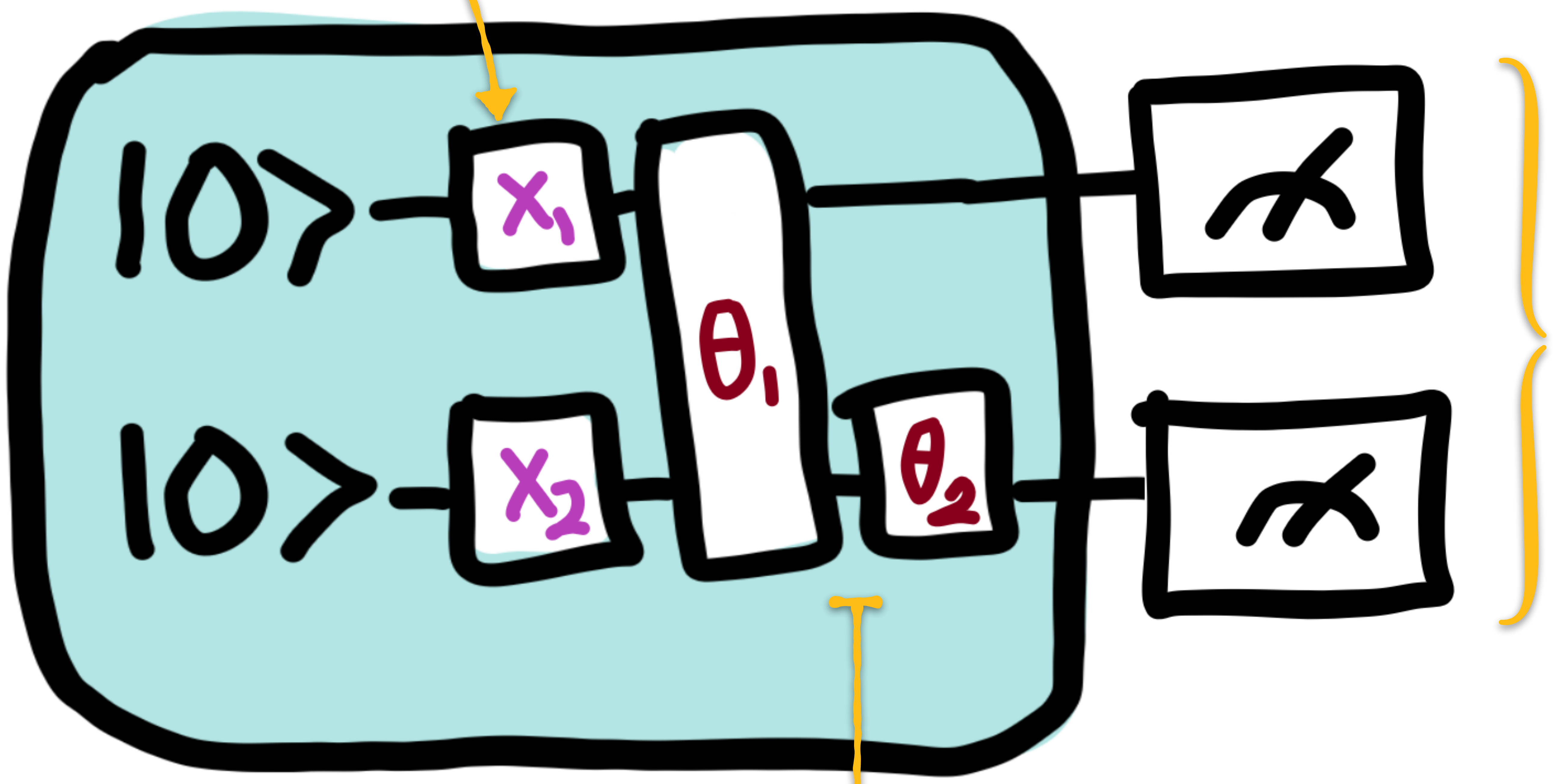
Introduction Vol. III: Quantum Variational Circuits

PennyLane: Variational Circuits

Input vector

$$R_y(x_i)\,|0\rangle = \cos\left(\frac{x_i}{2}\right)|0\rangle + \sin\left(\frac{x_i}{2}\right)|1\rangle$$

x_i can be any observable i.e.
p_T, η, m, ϕ etc.

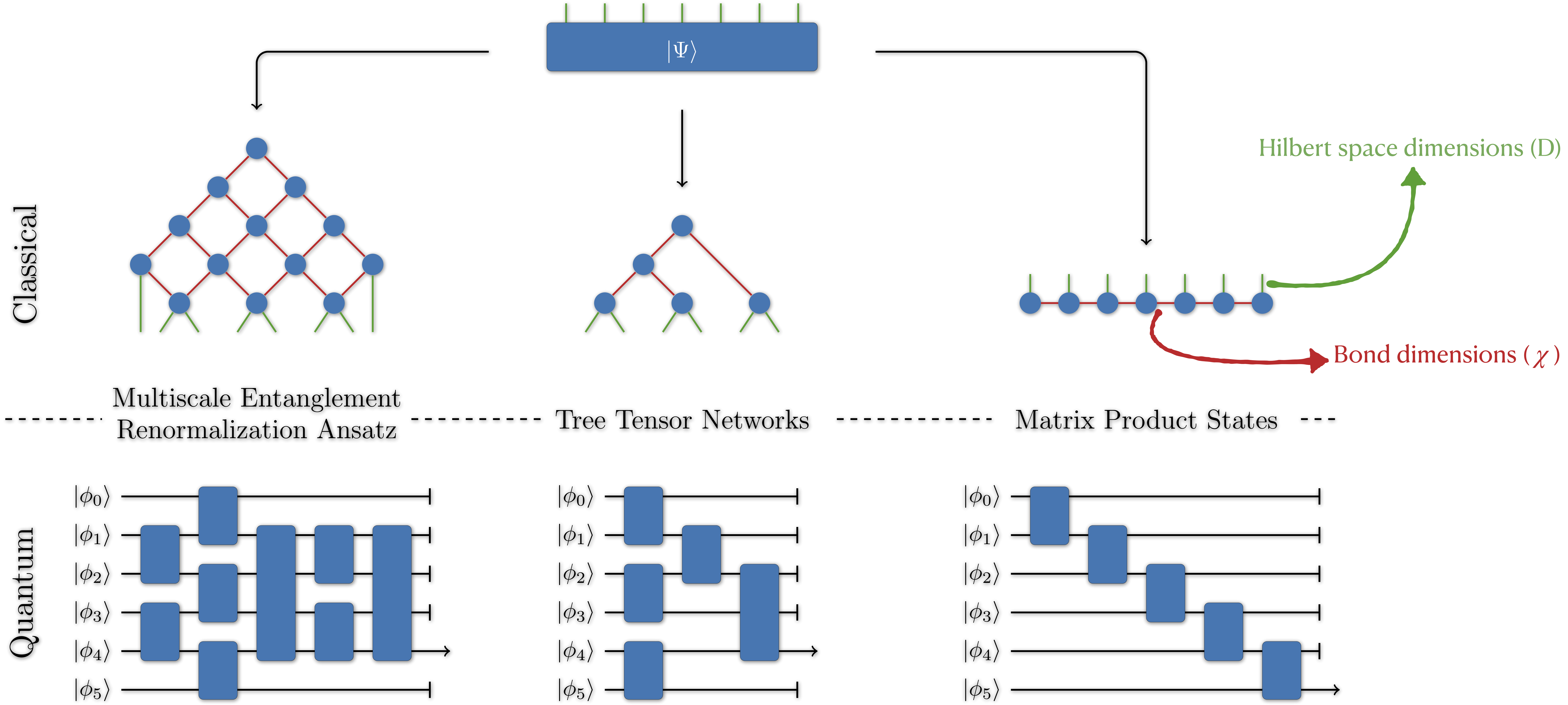


$$\hat{y} \rightarrow f(\mathbf{x}; \theta_i)$$

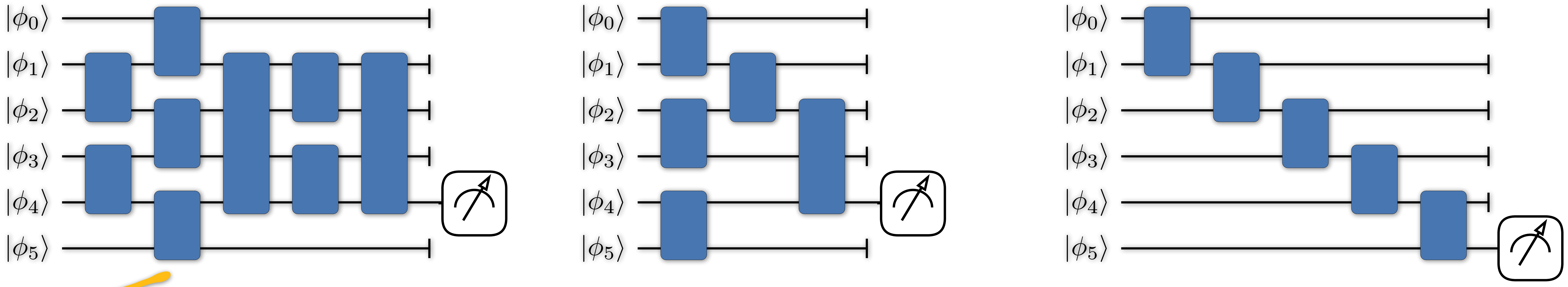
Cost function

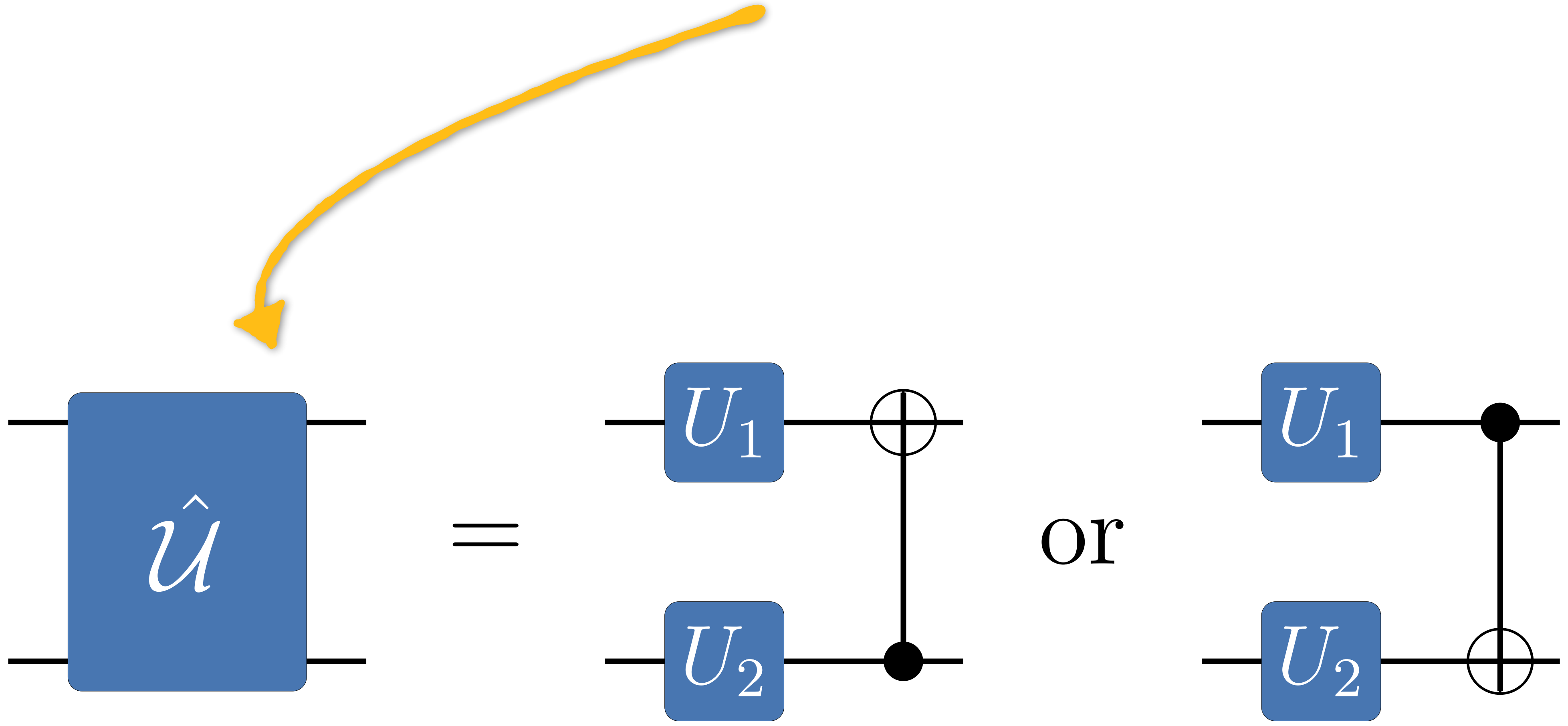
Update trainable parameters in the circuit

Quantum Tensor Networks



Quantum Tensor Networks



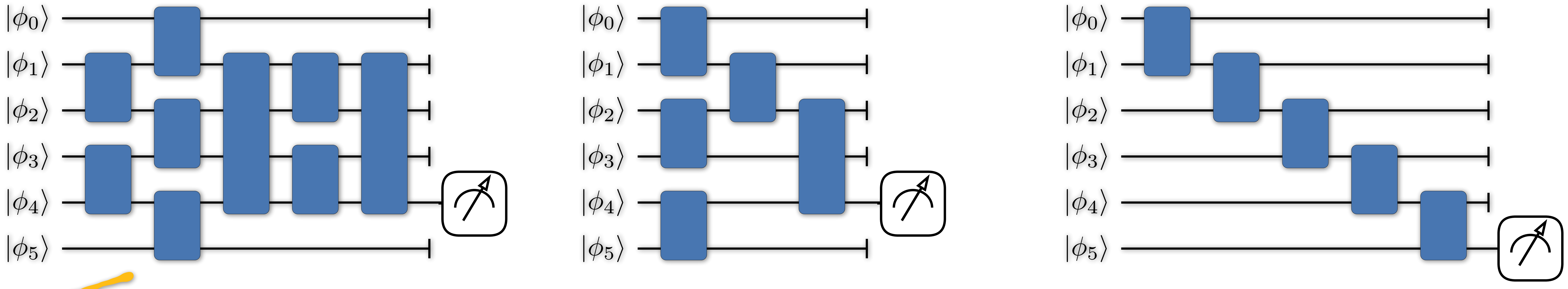


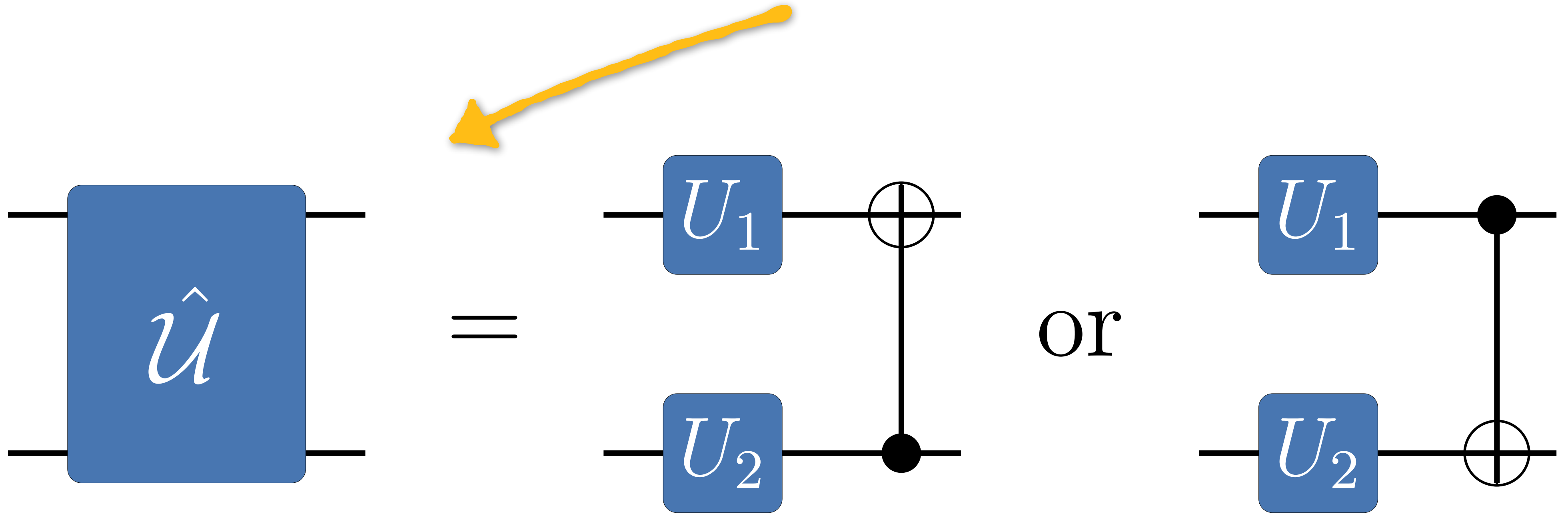
$$U(\theta,\varphi,\lambda) = \begin{pmatrix} \cos(\theta/2) & -e^{i\lambda}\sin(\theta/2) \\ e^{i\varphi}\sin(\theta/2) & e^{i(\lambda+\varphi)}\cos(\theta/2) \end{pmatrix}$$

$\varphi, \lambda = 0 \rightarrow$ 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

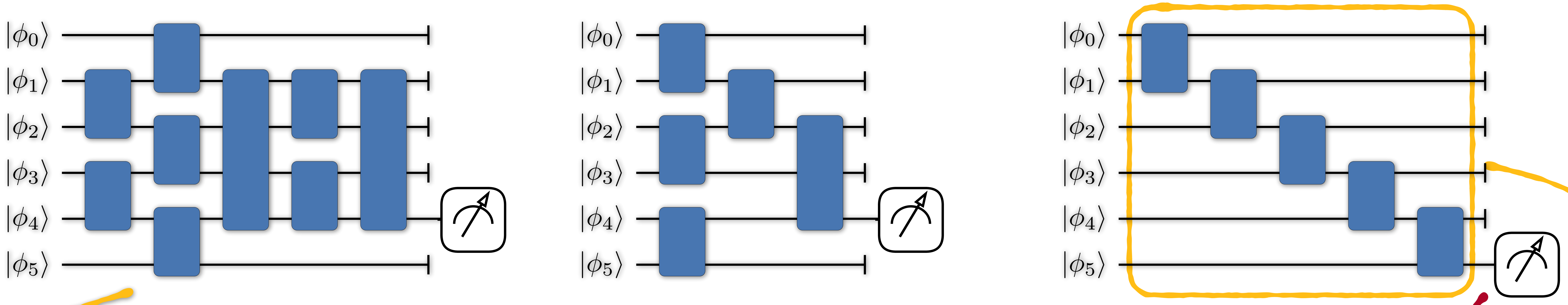


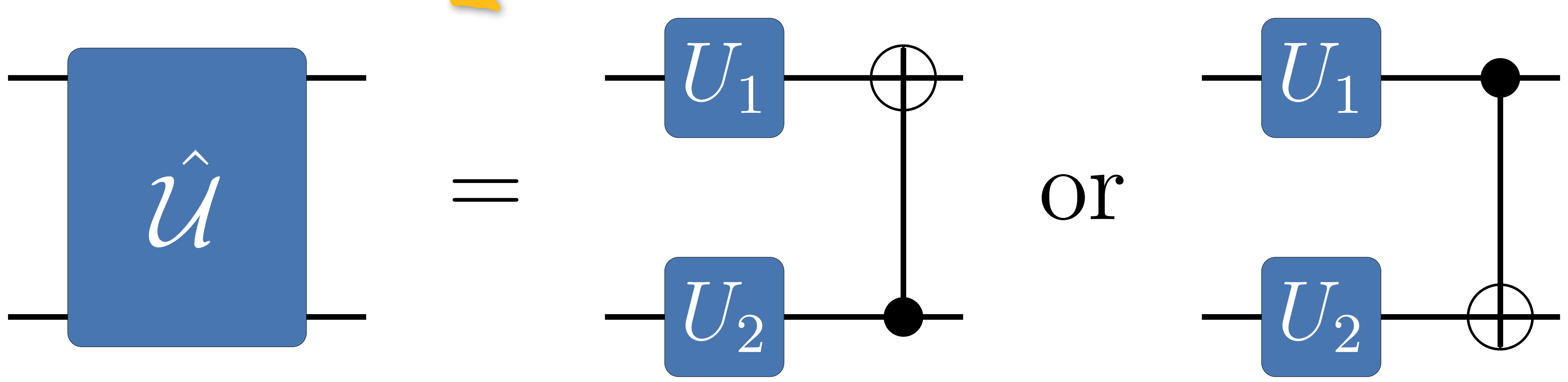


$$\mathrm{M}_\theta(\Phi^{\beta_1\cdots\beta_n}(\mathbf{x})) = \langle\Phi|\ \hat{\mathcal{U}}^\dagger_{\rm QC}(U_i(\theta_j))\ \hat{\mathrm{M}}\ \hat{\mathcal{U}}_{\rm QC}(U_i(\theta_j))\ |\Phi\rangle$$

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

Quantum Tensor Networks





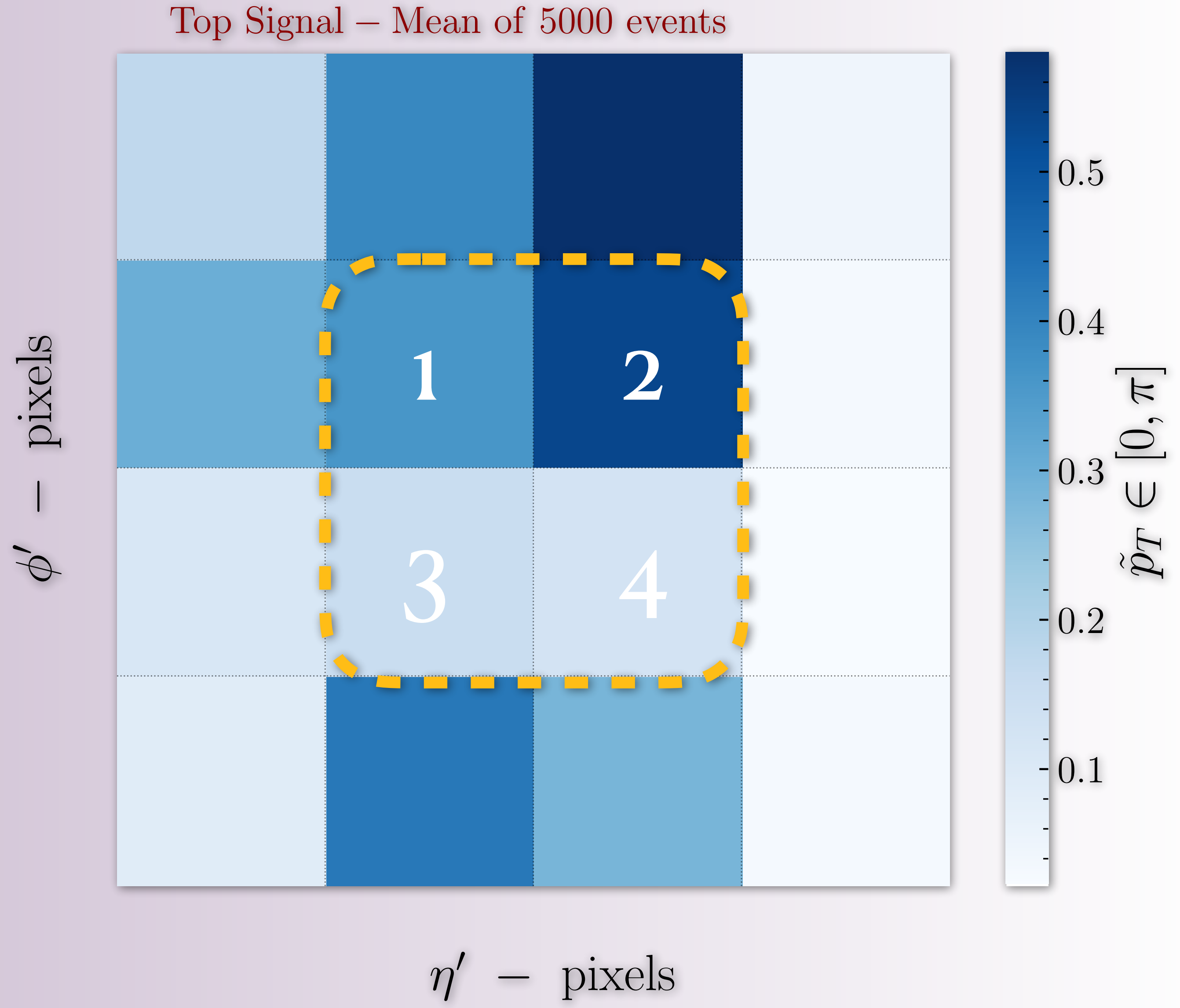
$$\mathrm{M}_\theta(\Phi^{\beta_1\cdots\beta_n}(\mathbf{x})) = \langle\Phi|\ \hat{\mathcal{U}}^\dagger_{\mathrm{QC}}(U_i(\theta_j))\ \hat{\mathrm{M}}\ \boxed{\hat{\mathcal{U}}_{\mathrm{QC}}(U_i(\theta_j))}\ |\Phi\rangle$$

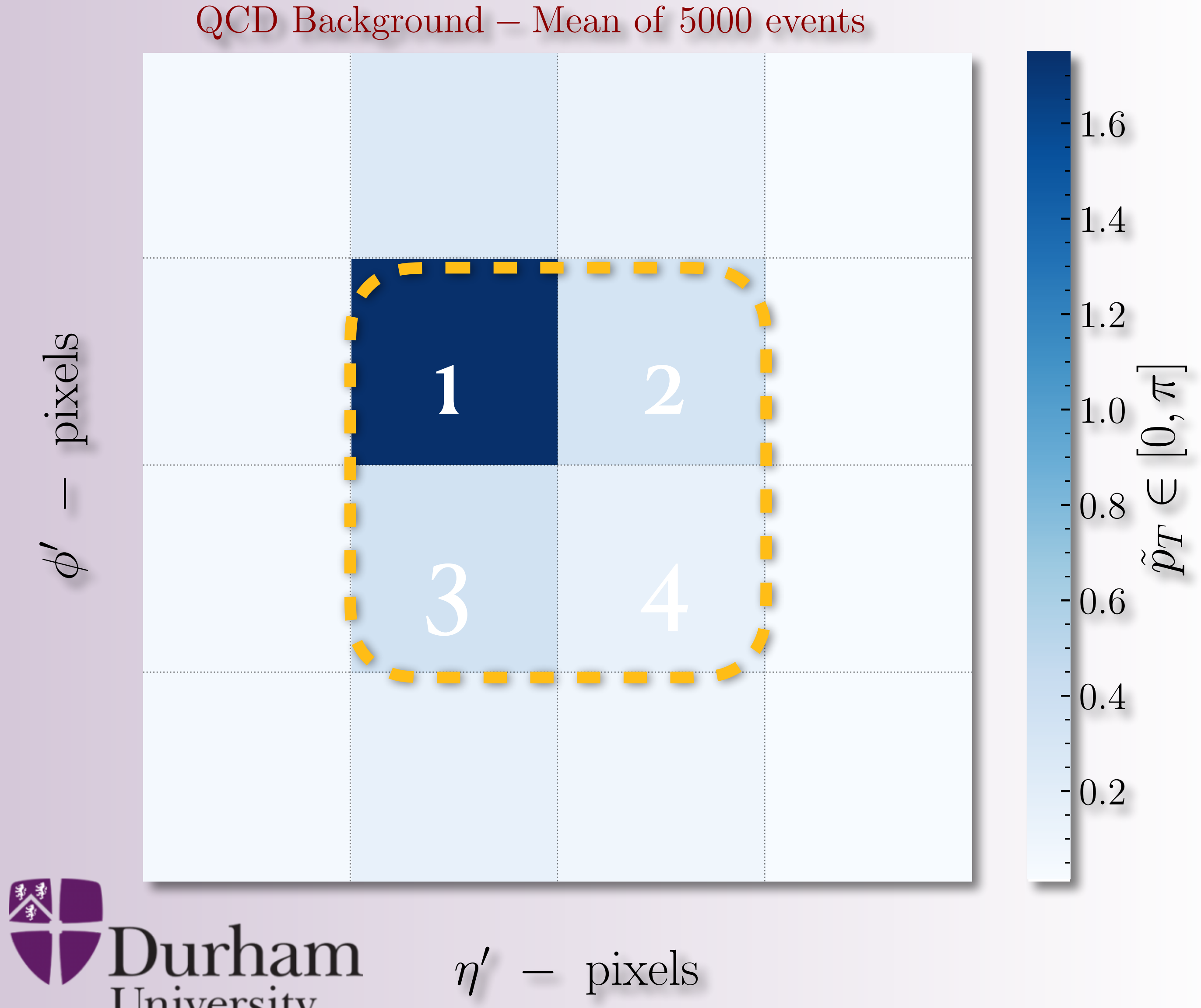
$\hat{\sigma}_z$

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

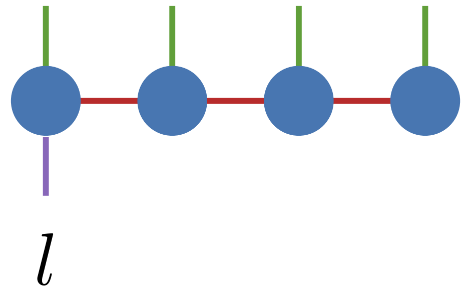
Same story: Calculate probability, minimize the objective function, update parameters...

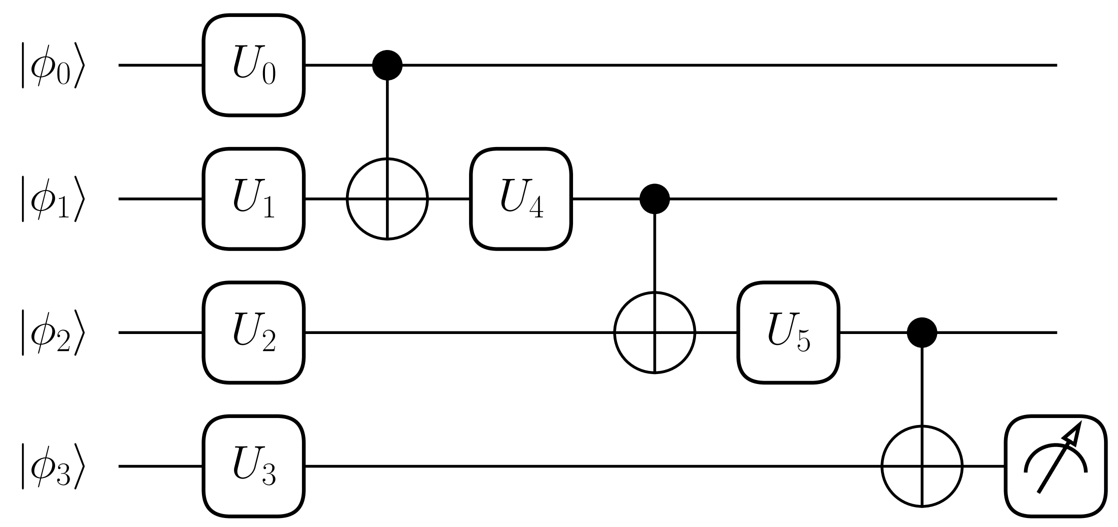
Experimenting with 4-Qubits



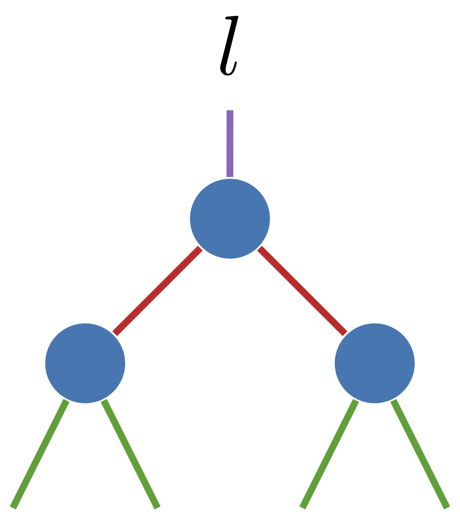


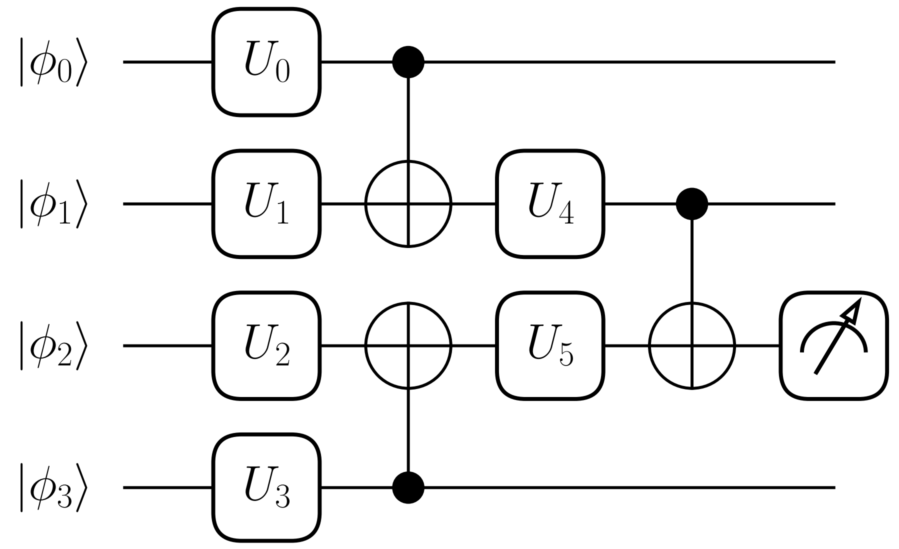
Matrix Product States



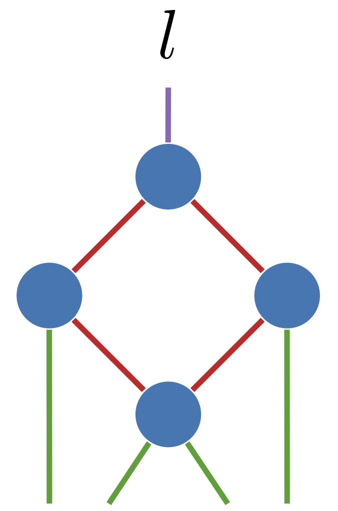


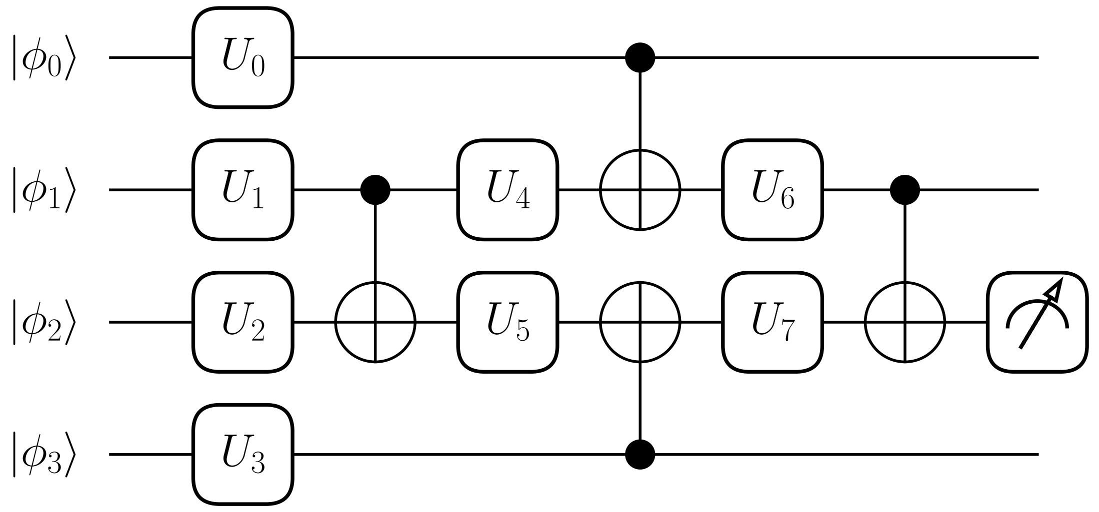
Tree Tensor Networks



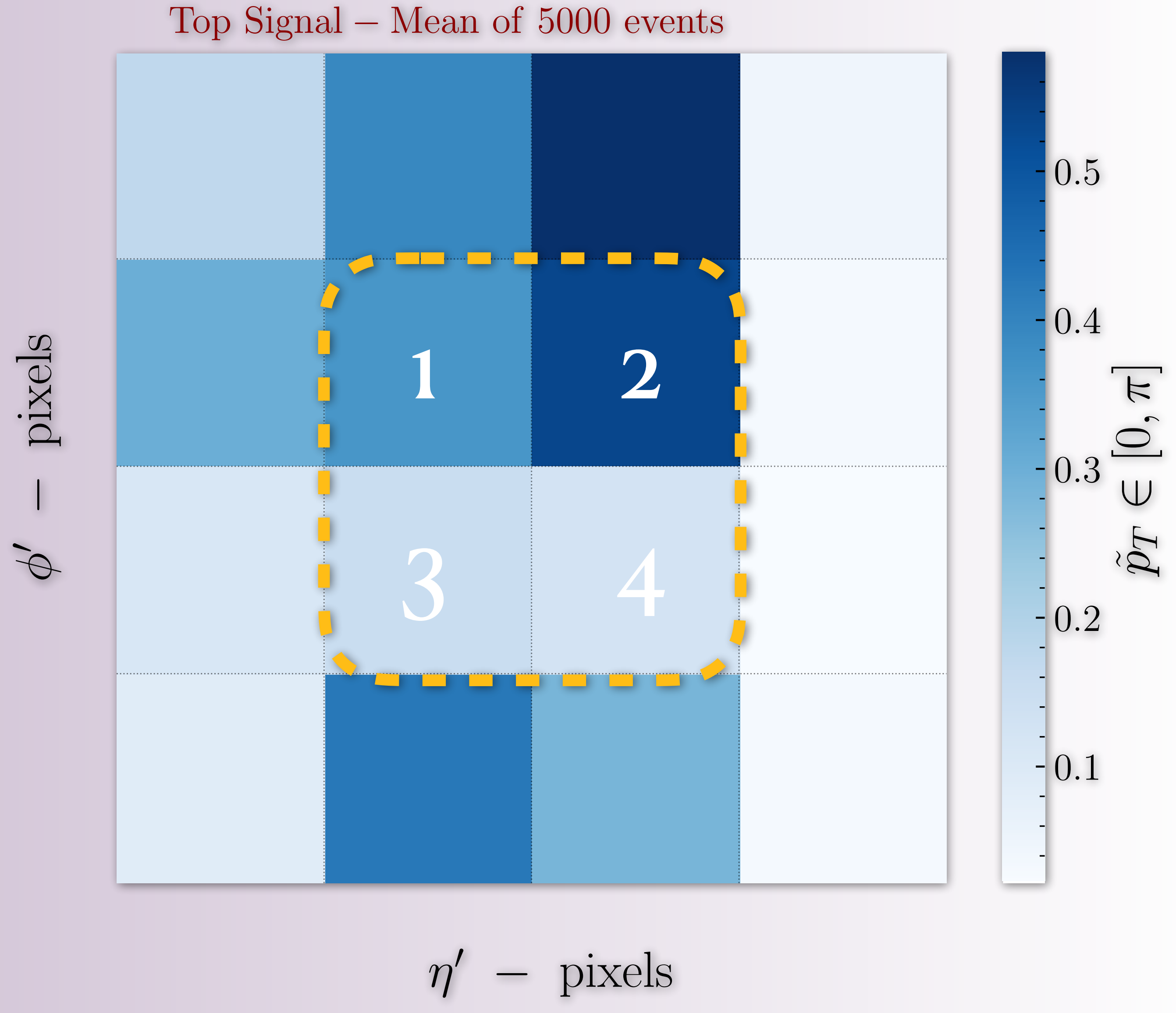


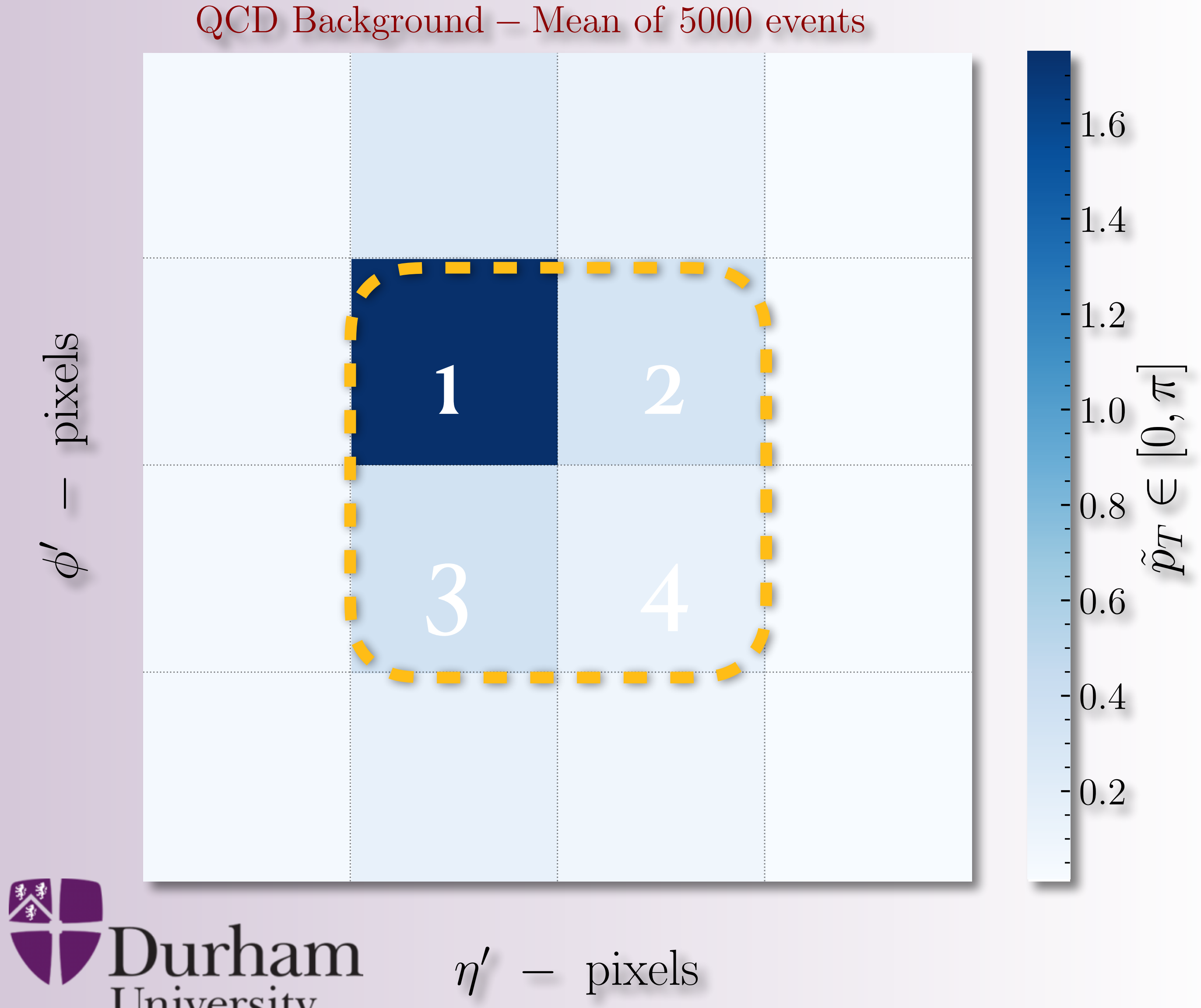
Multiscale Entanglement Renormalisation Ansatz

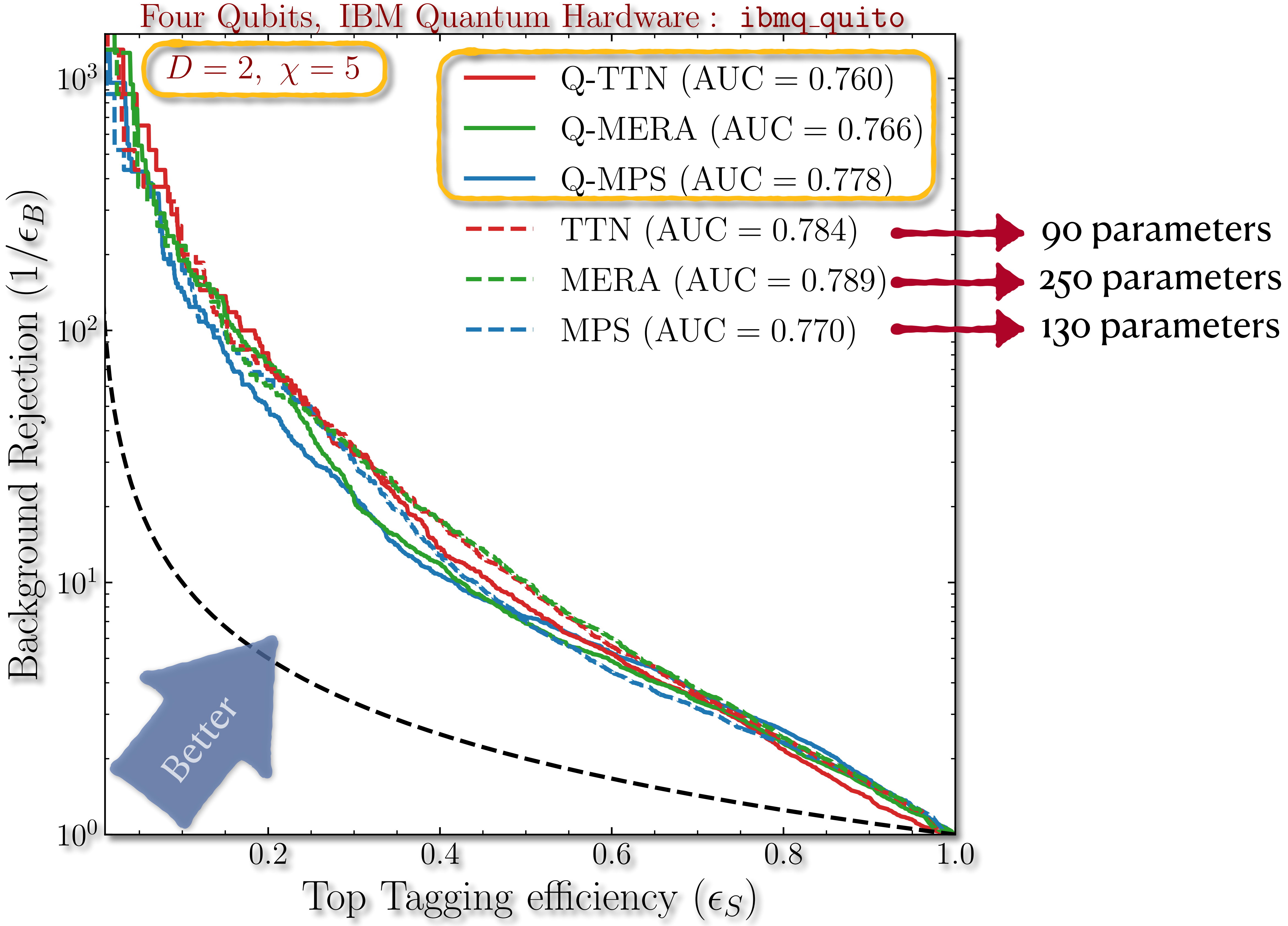




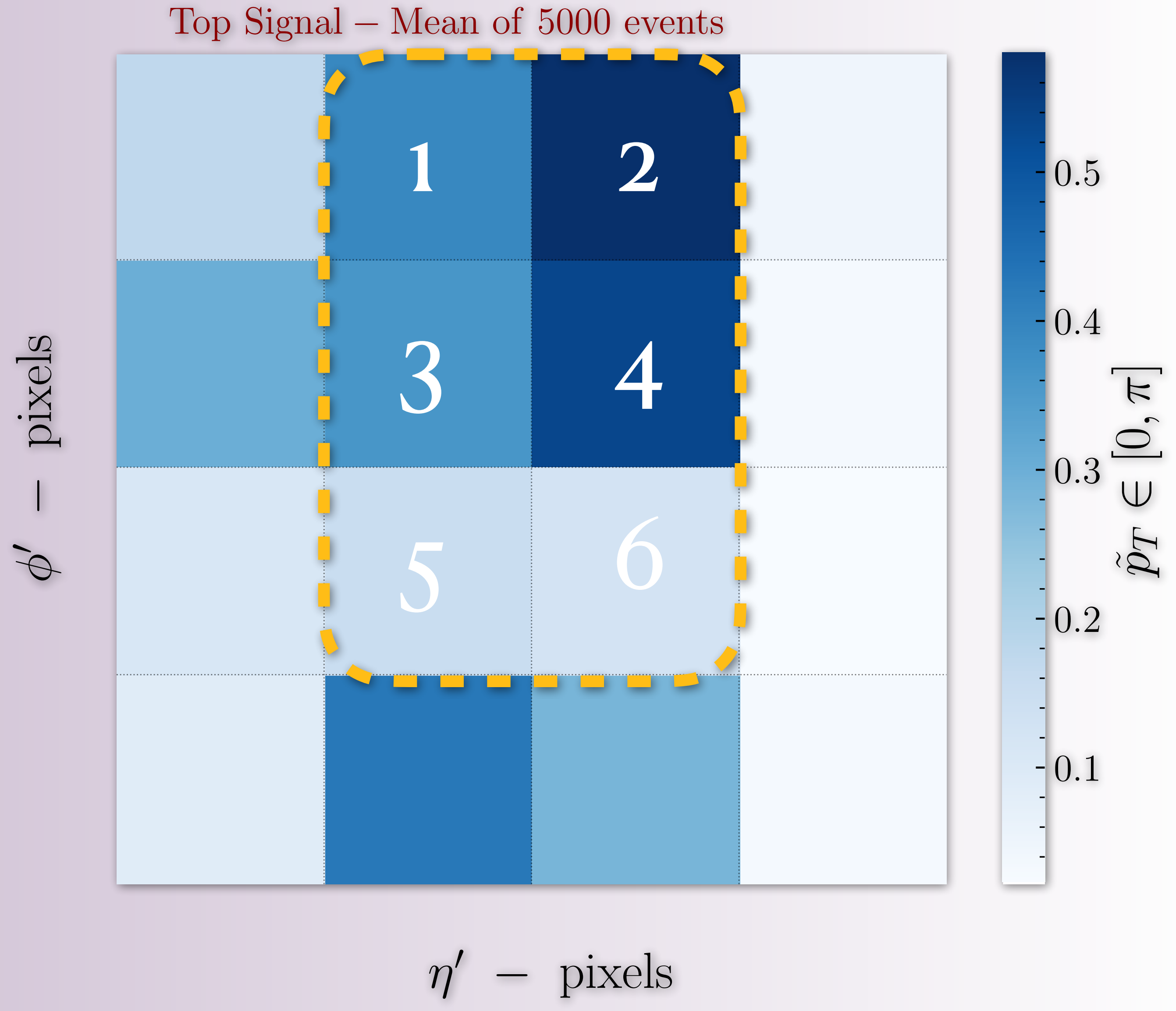
Experimenting with 4-Qubits

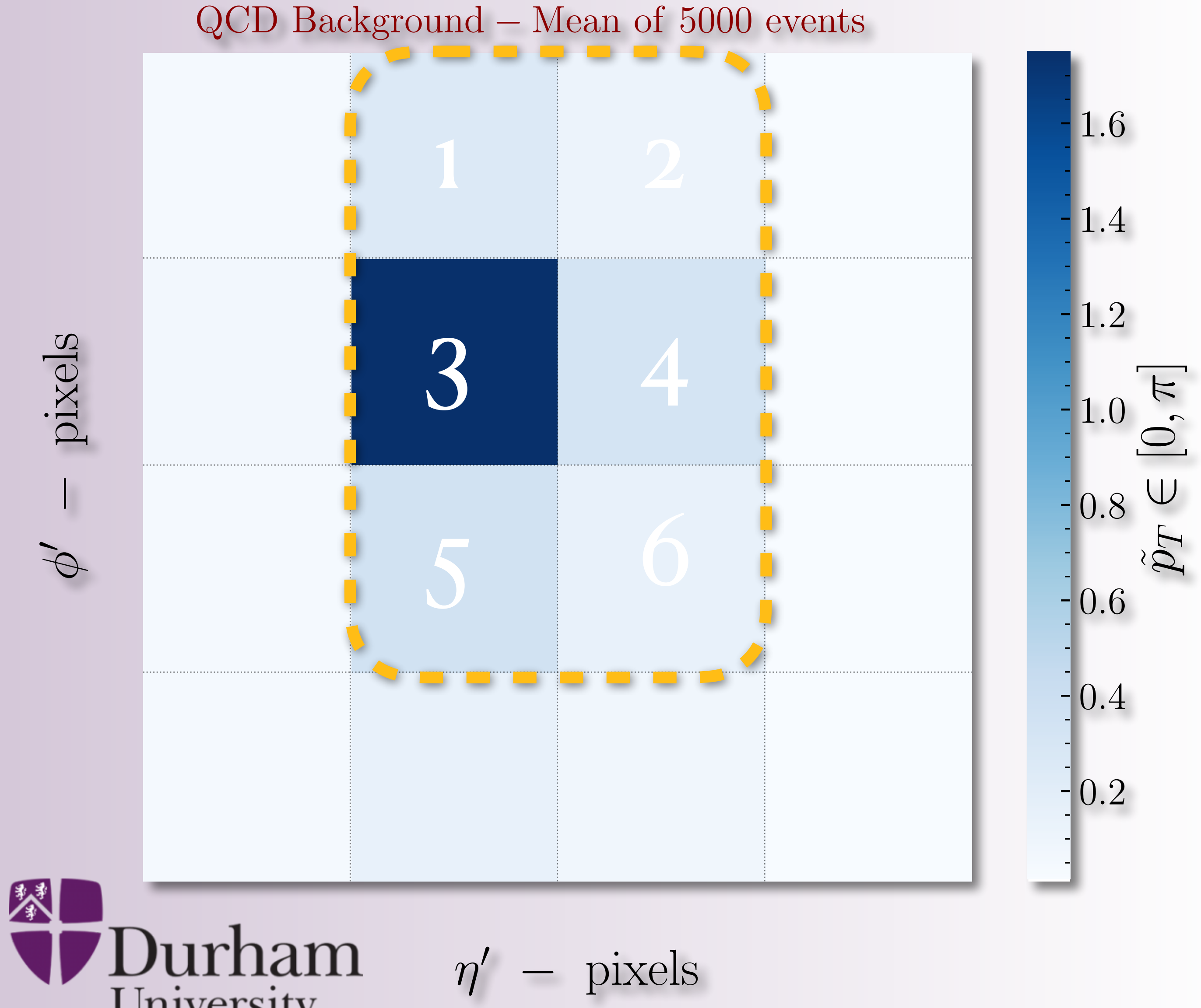




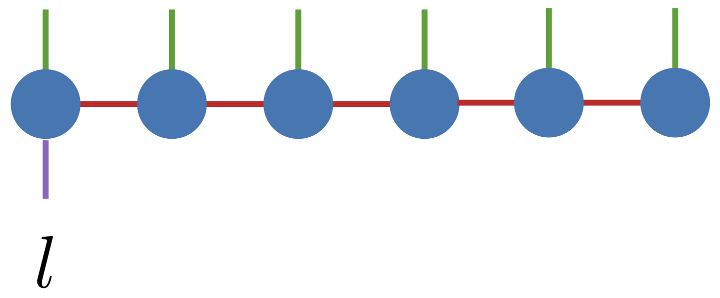


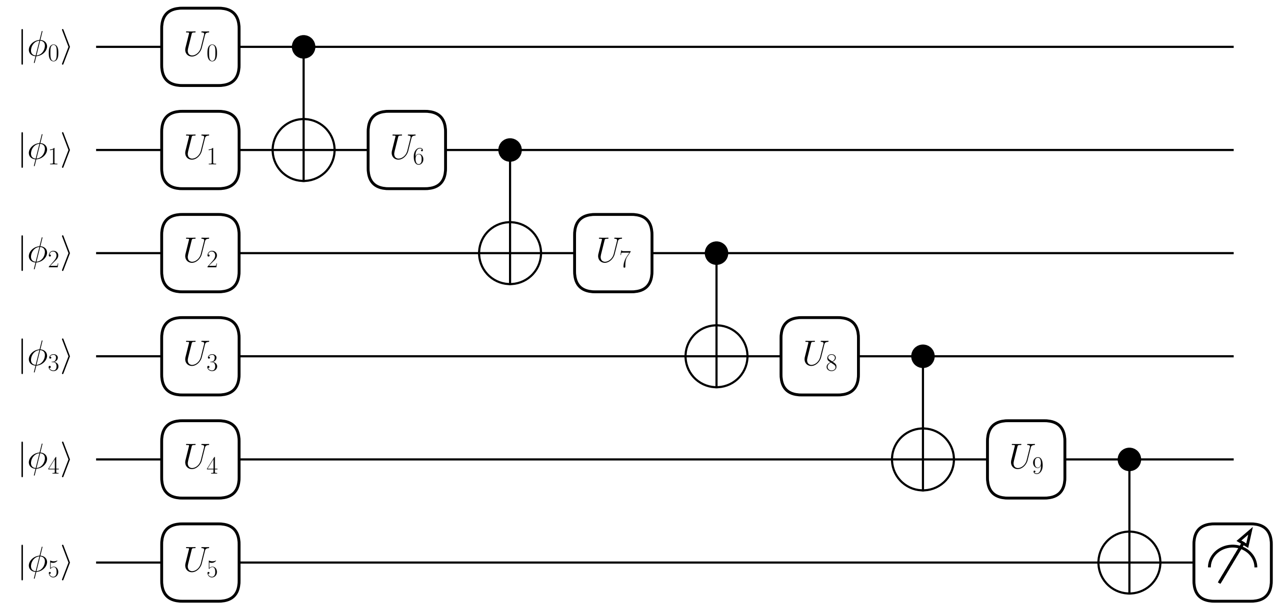
Experimenting with 6-Qubits



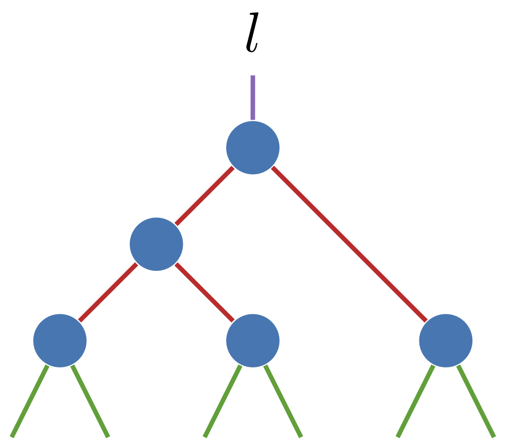


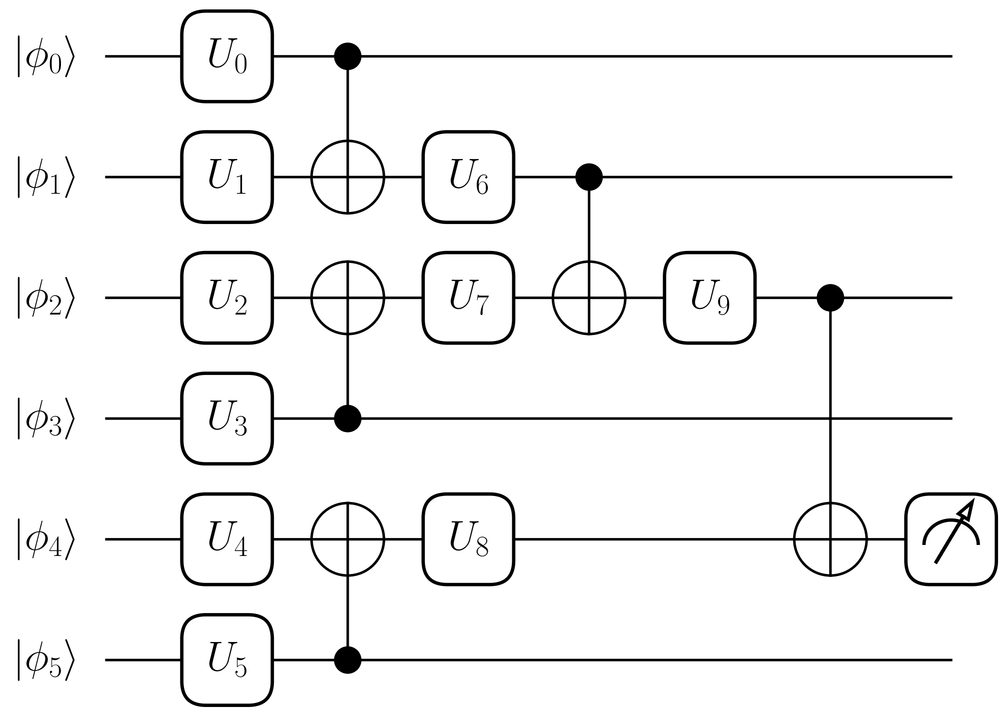
Matrix Product States



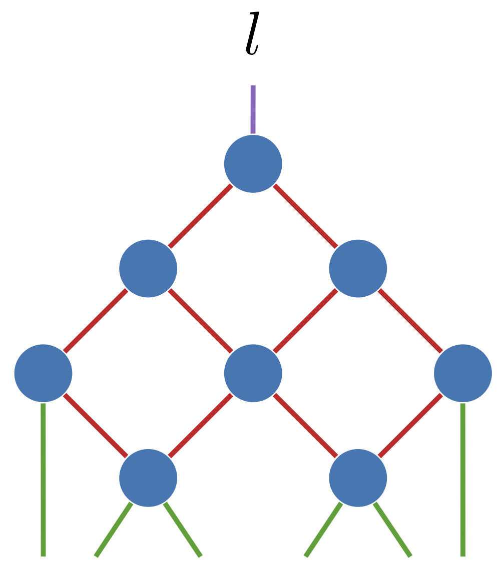


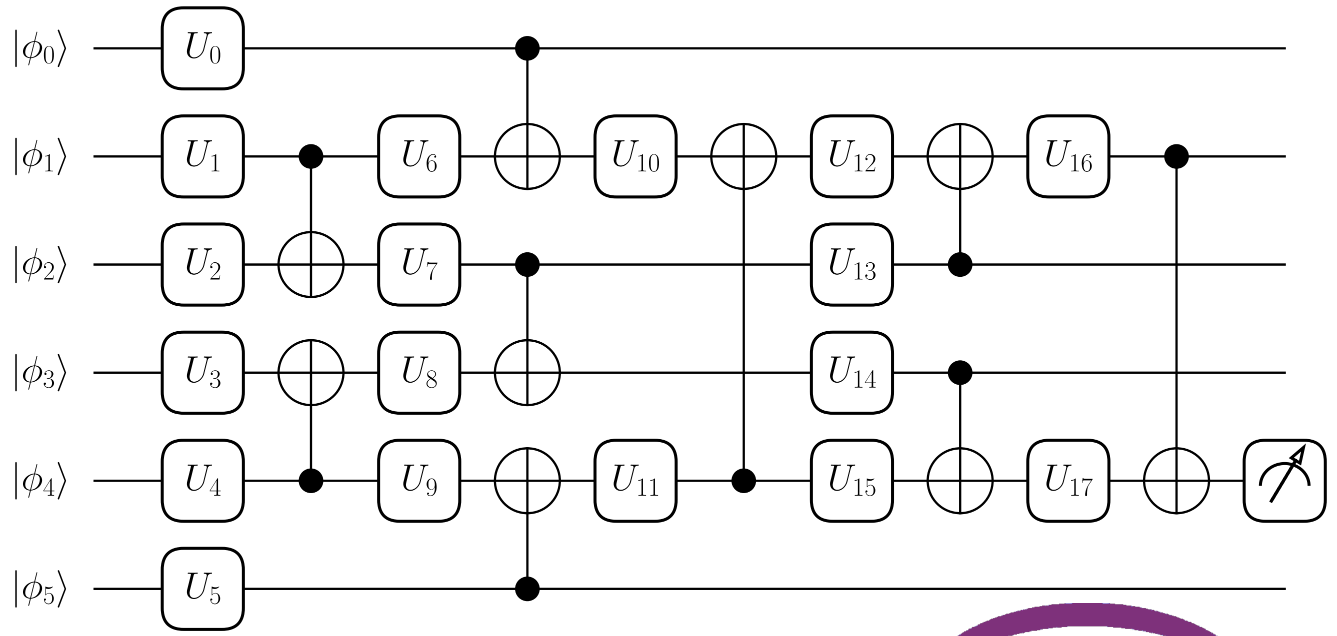
Tree Tensor Networks



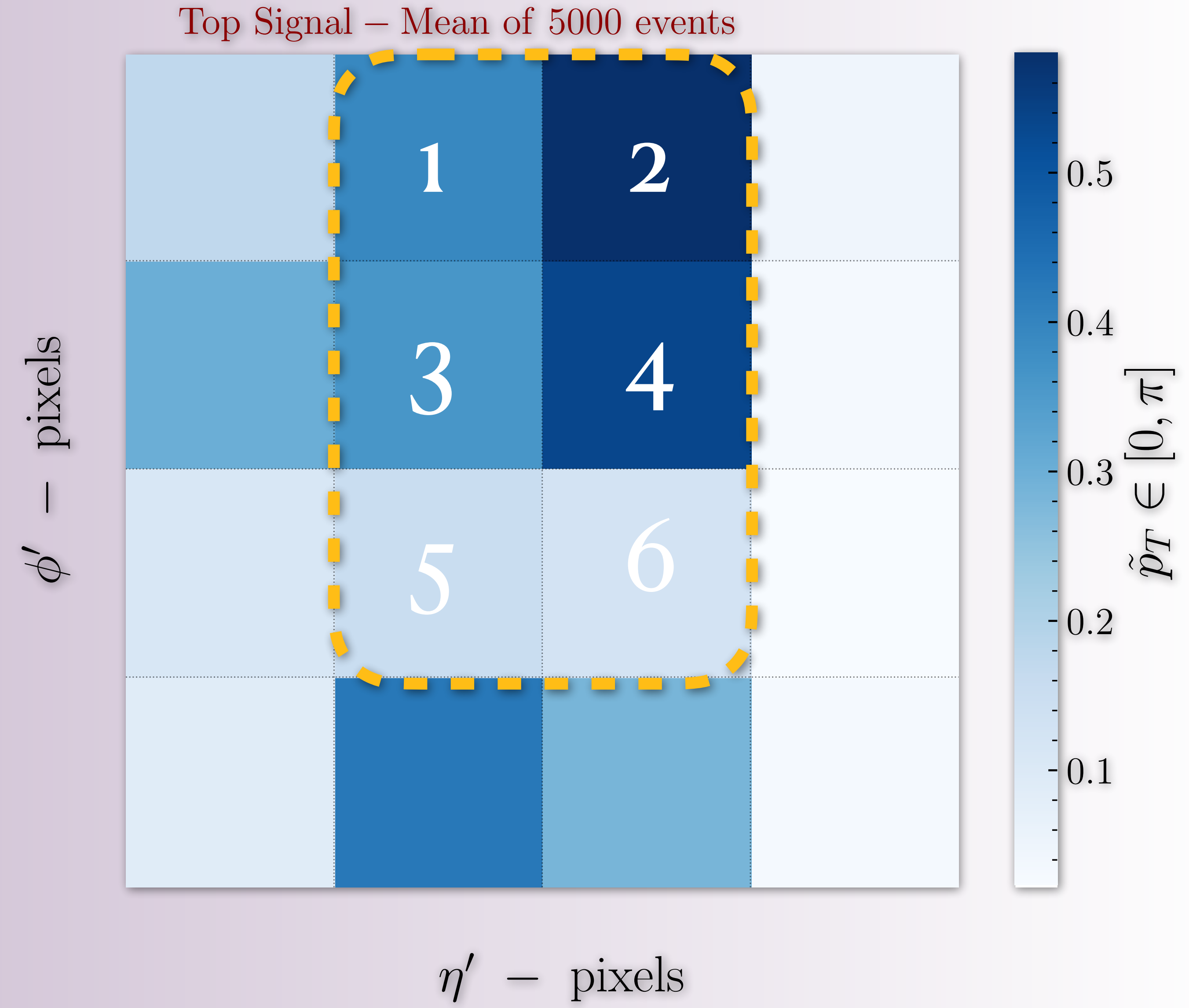


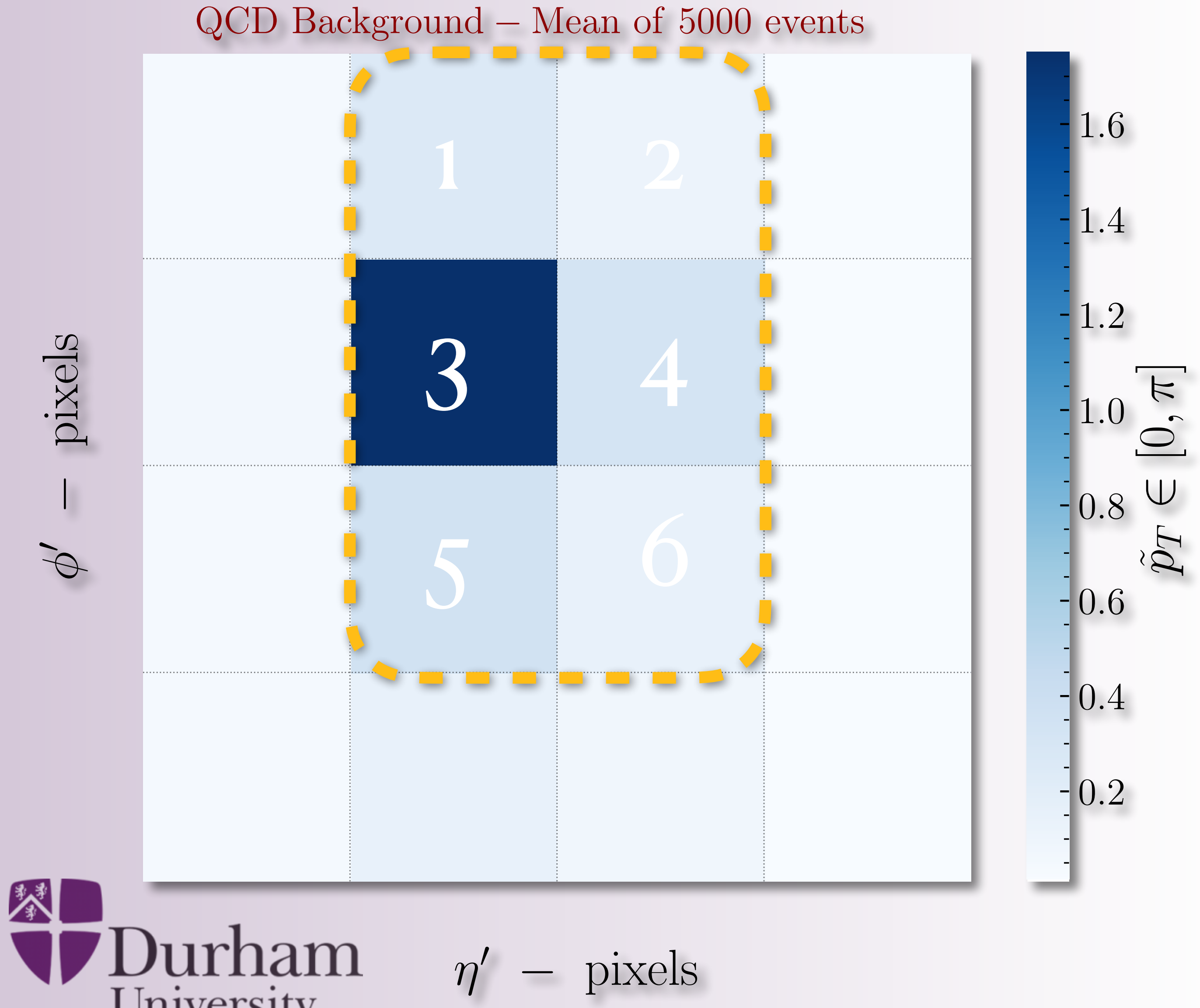
Multiscale Entanglement Renormalisation Ansatz

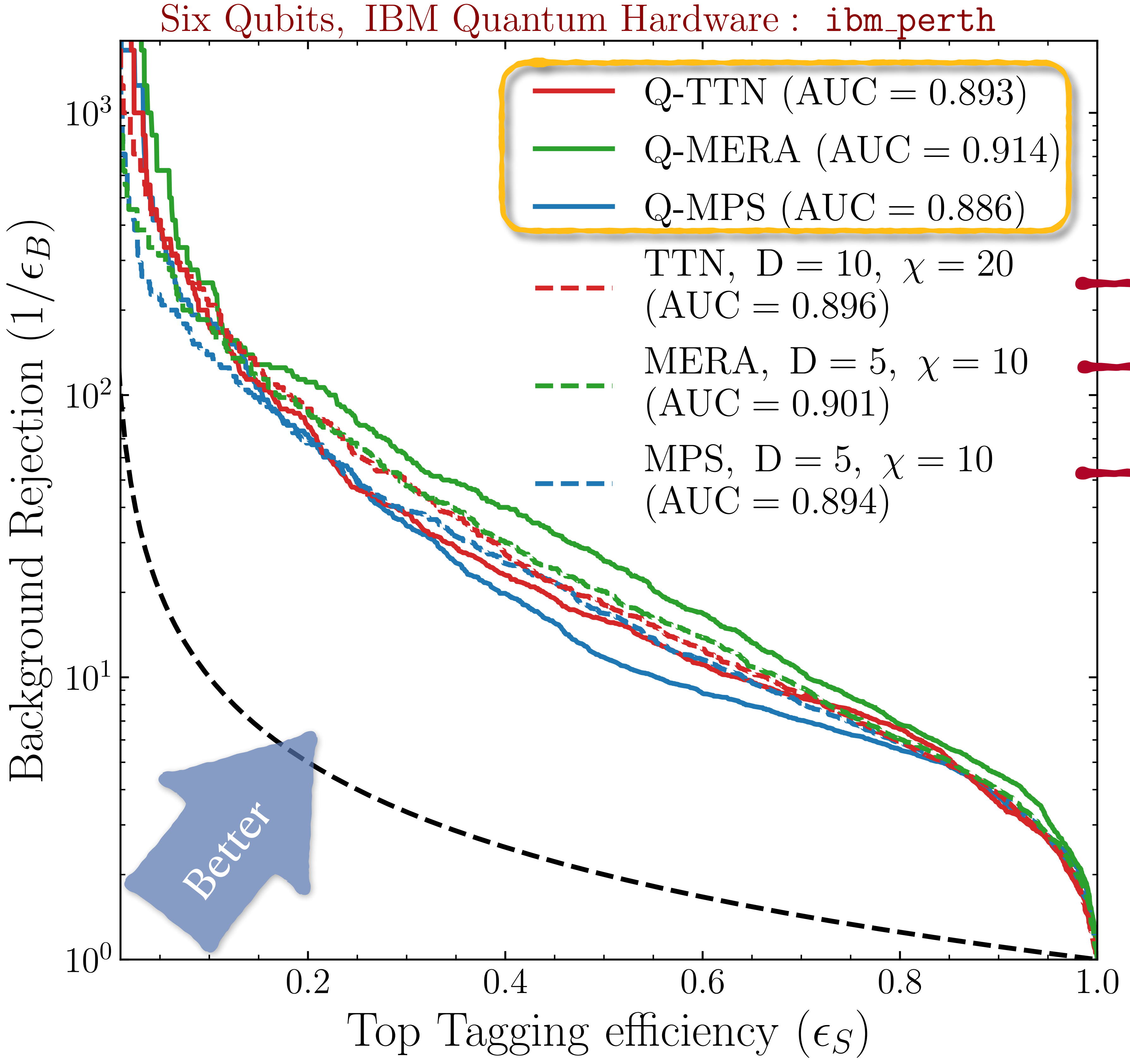




Experimenting with 6-Qubits





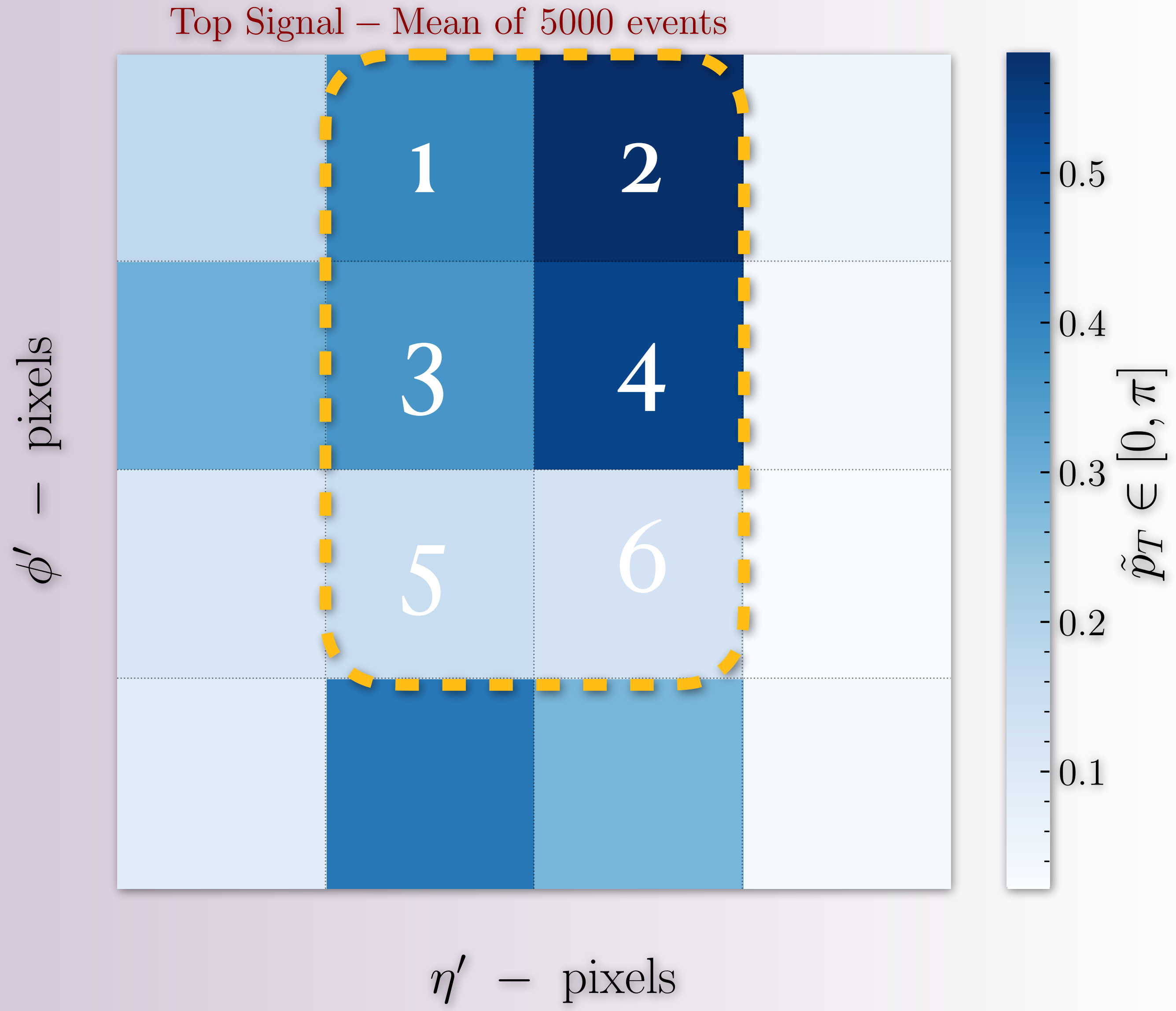


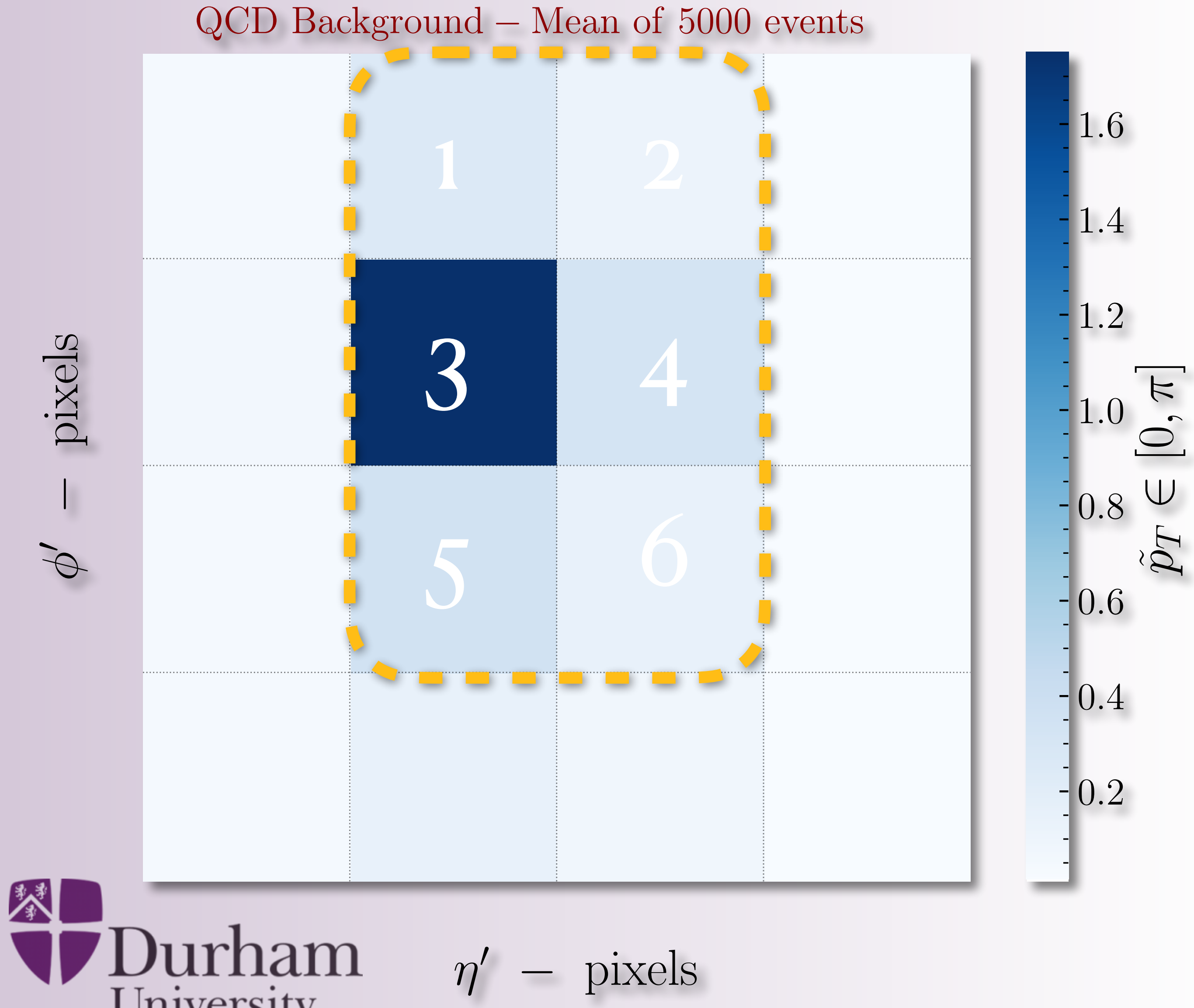
14800 parameters

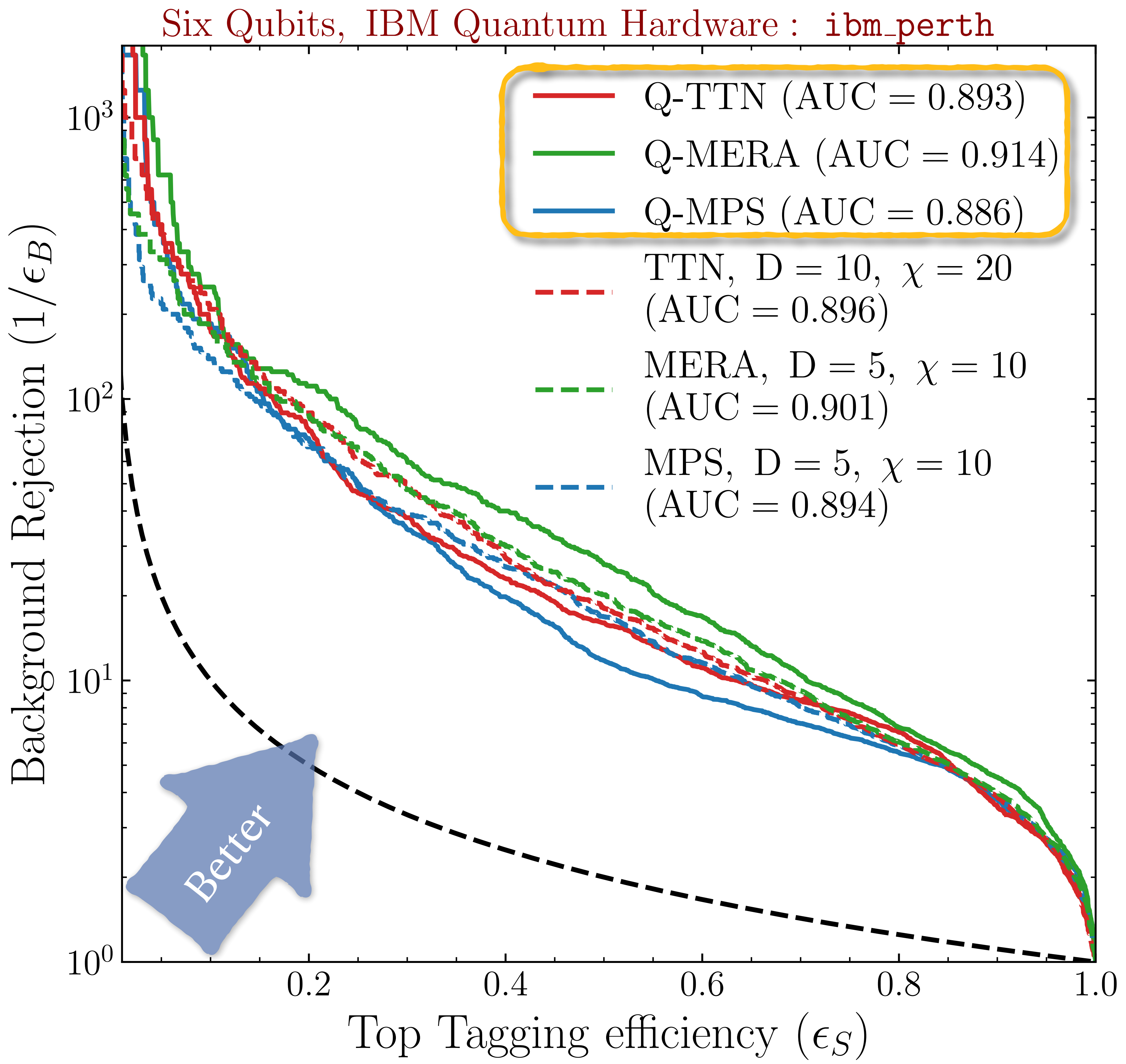
18200 parameters

2150 parameters

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!

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 optimization compared to classical networks.
- BUT near term quantum devices are still very much limited to a few qubits.

BACKUP

Matrix Product States for Classification

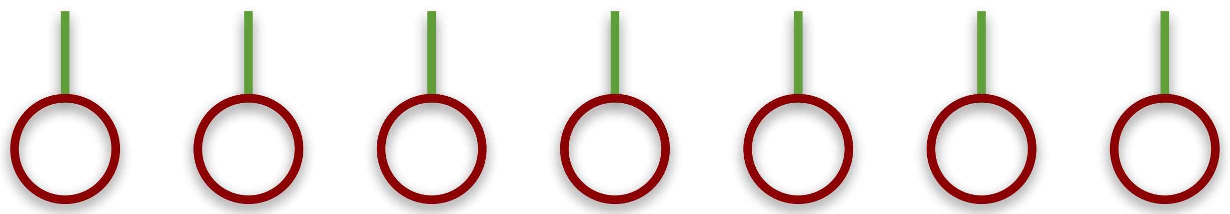
Sub-Outline

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

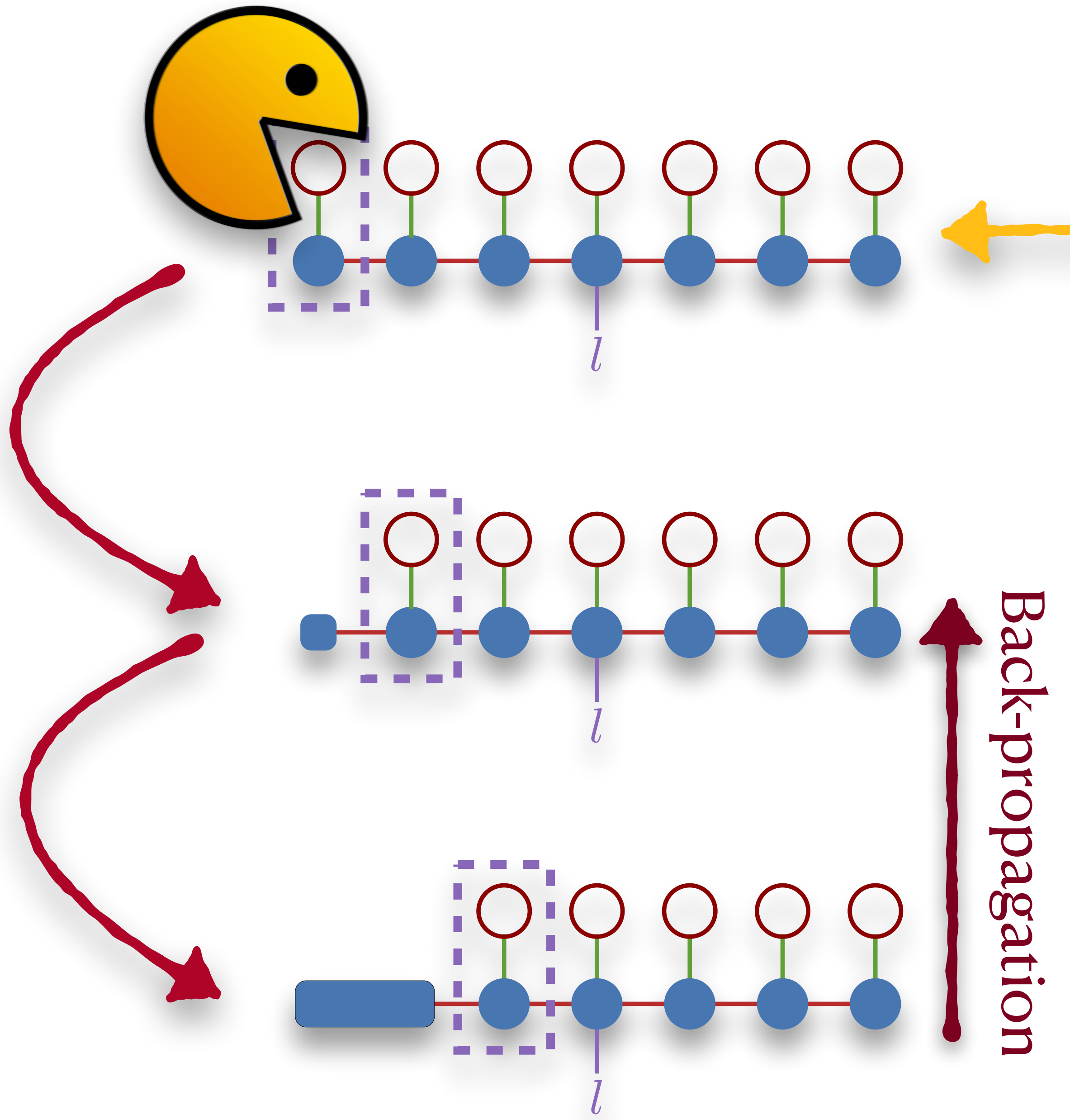
Data Embedding

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

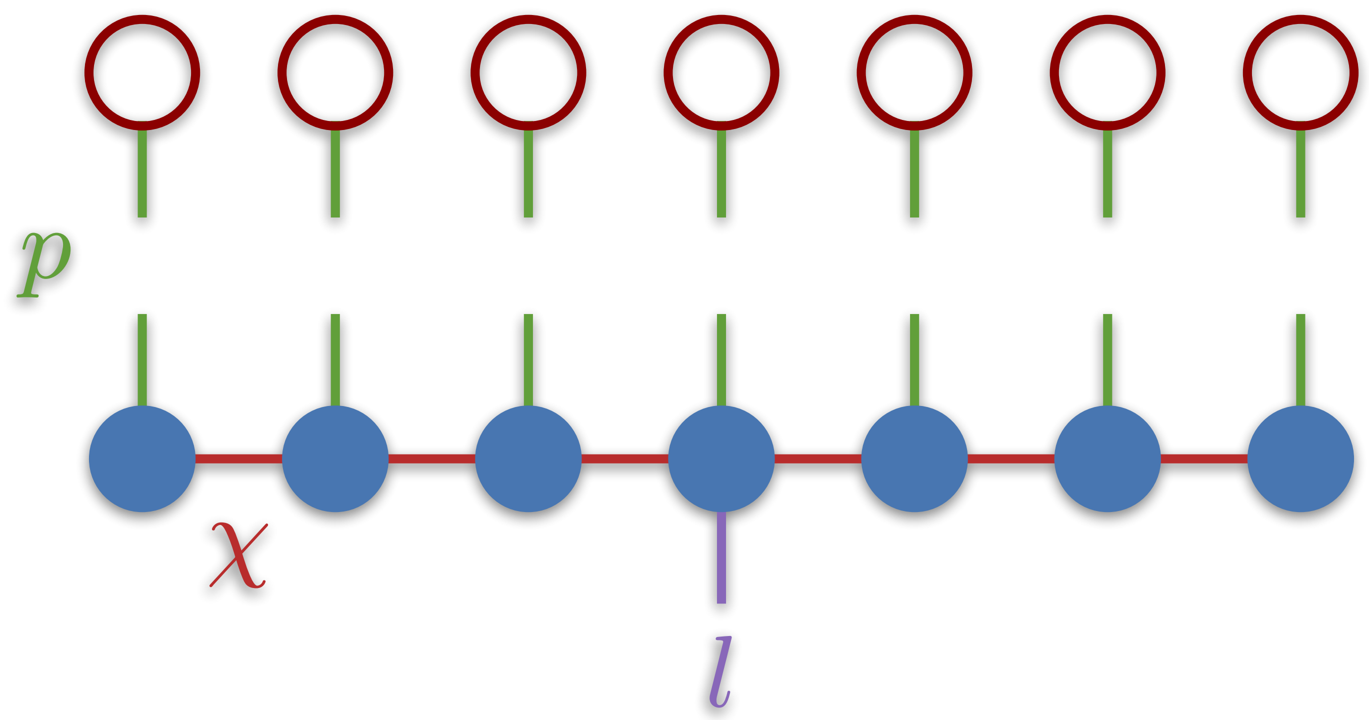
$$\phi^{p_i}(x_i) = \begin{bmatrix} \cos(x_i\ \pi/2) \\ \sin(x_i\ \pi/2) \end{bmatrix} \quad \text{or} \quad \phi^{p_i}(x_i) = \begin{bmatrix} 1 \\ x_i \\ x_i^2 \end{bmatrix} \quad \text{or} \quad \cdots$$



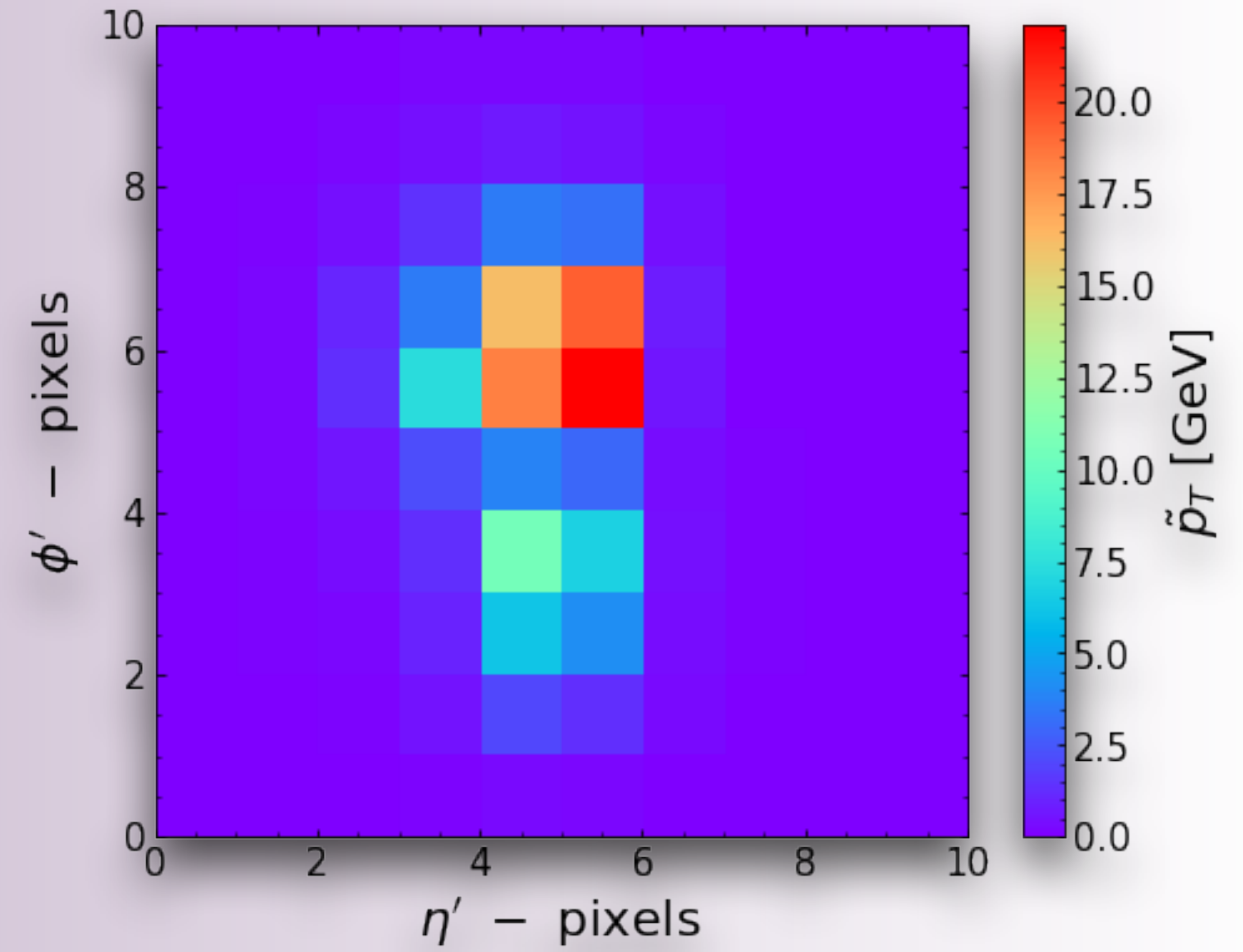
$= \Phi^{p_1\cdots p_n}(\mathbf{x})$

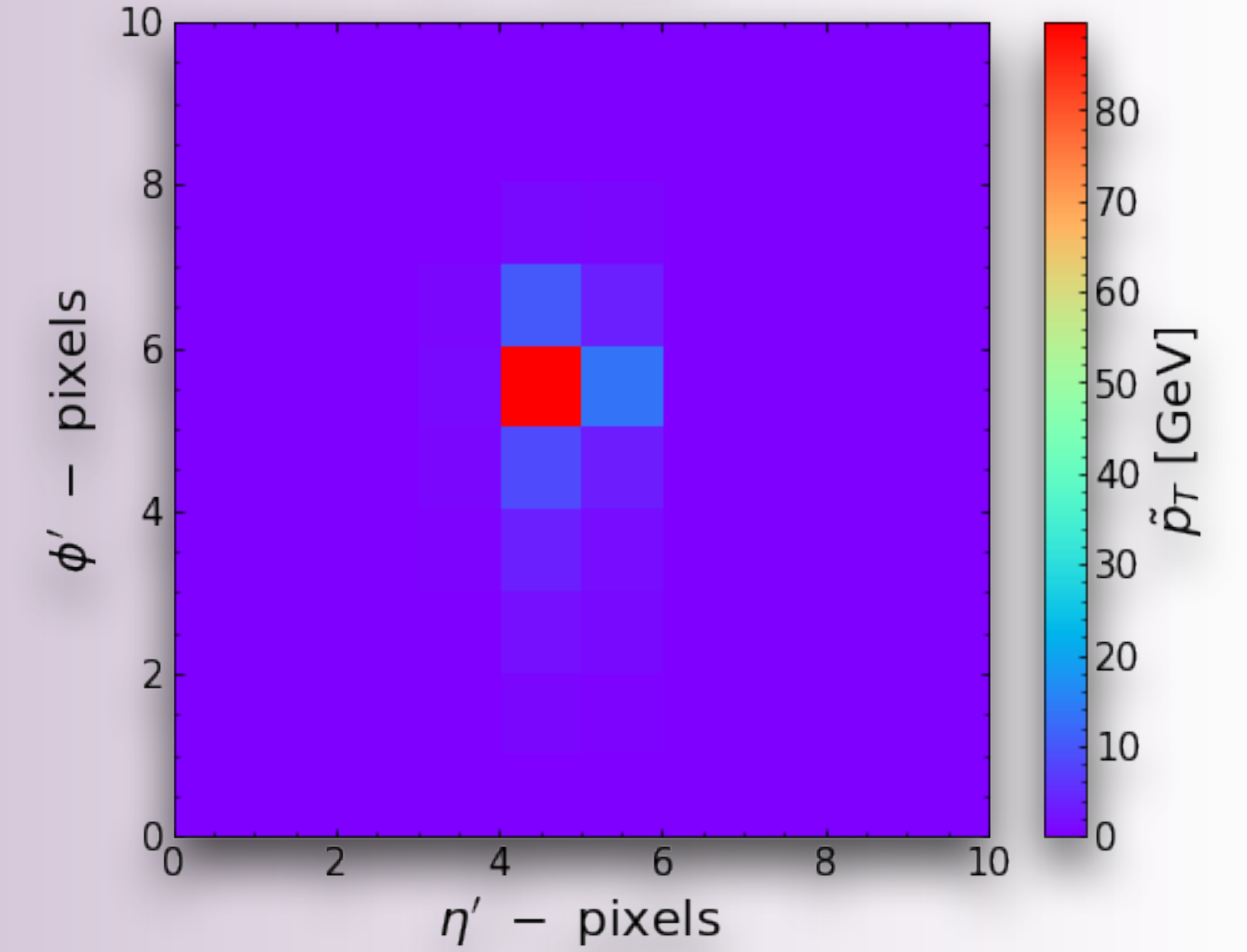


An efficient contraction algorithm is essential for training!

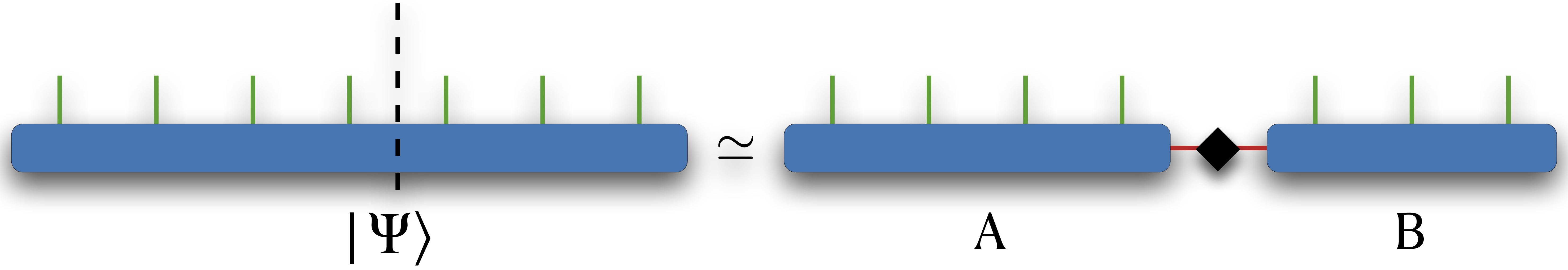


Top Tagging through MPS





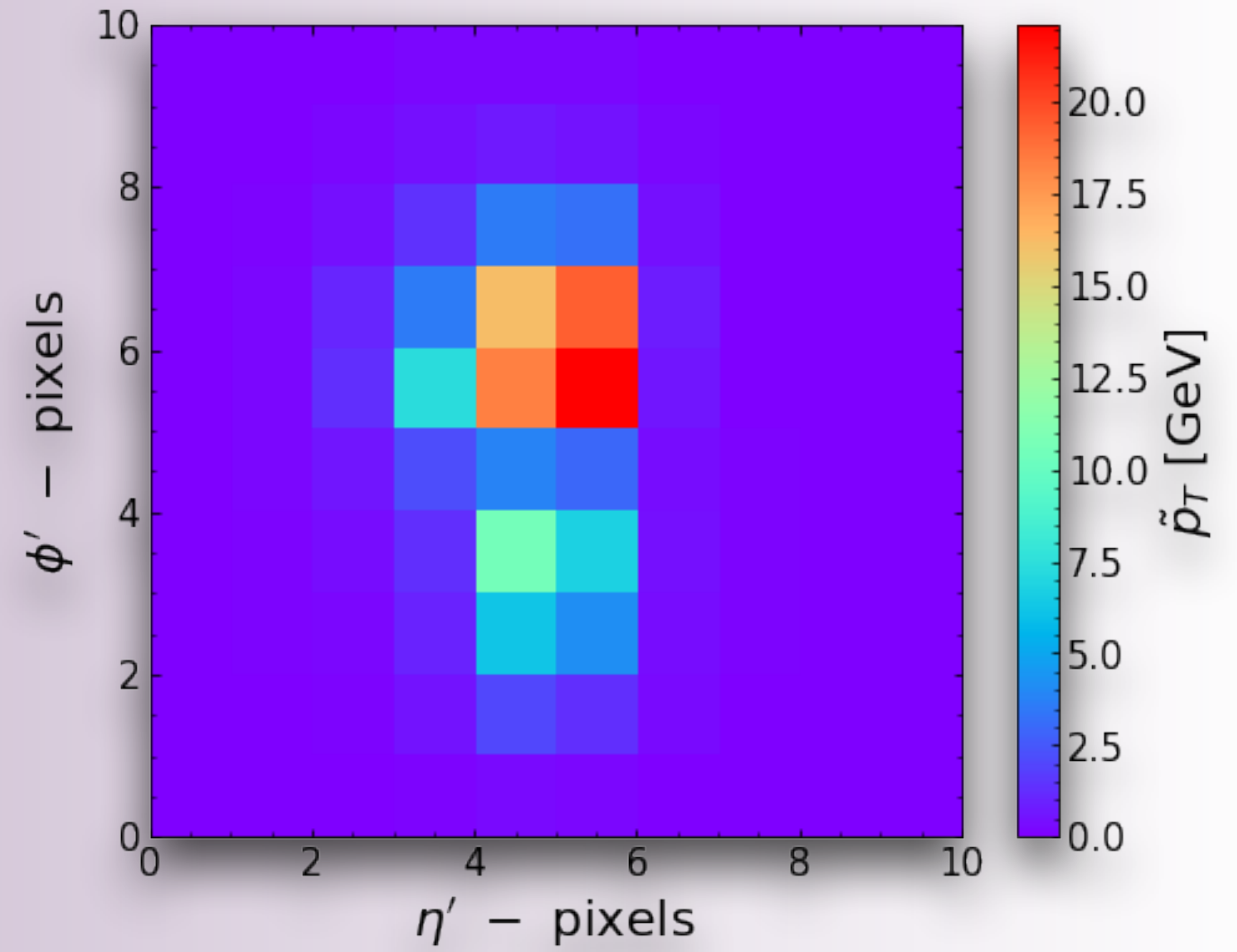
Entanglement Entropy & Schmidt Decompostion

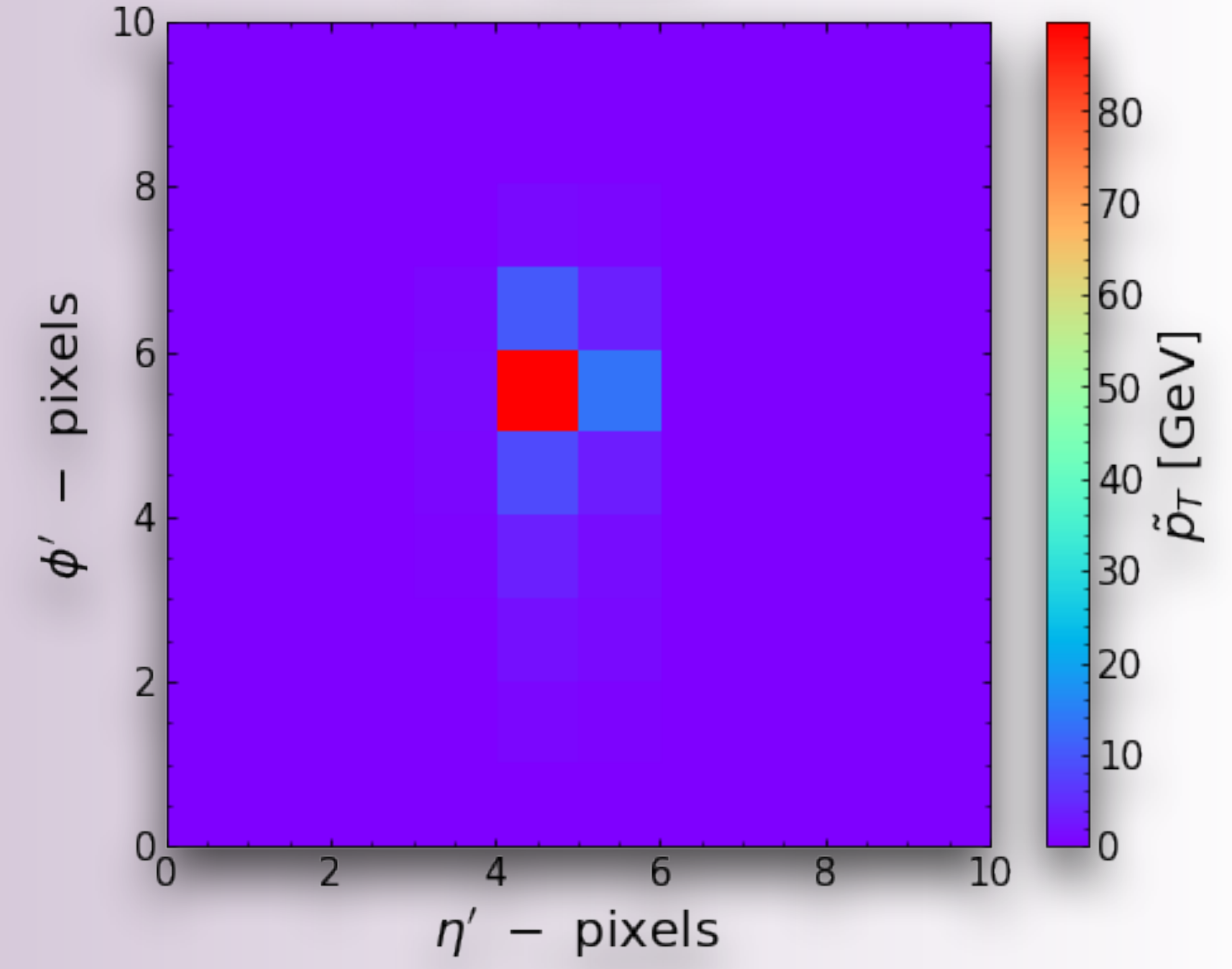


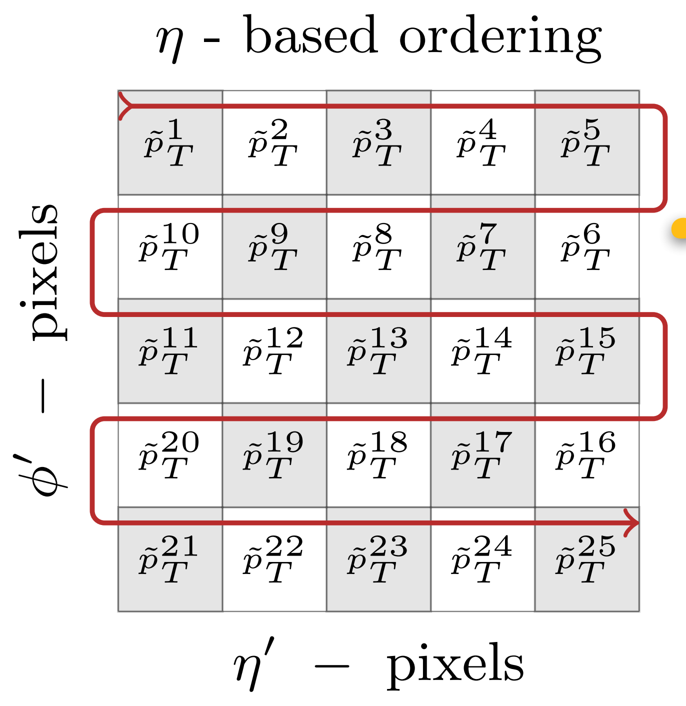
$$|\Psi\rangle_{AB} = \sum_i^{\chi} \lambda_\alpha |\alpha\rangle_A |\alpha\rangle_B \quad \rightarrow \quad \lambda_\alpha := \text{Schmidt values}$$

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

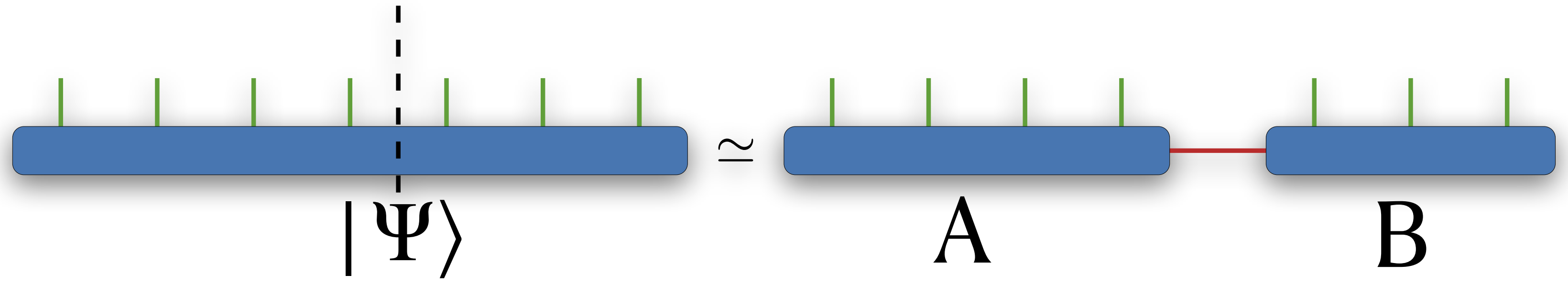
Top Tagging through MPS





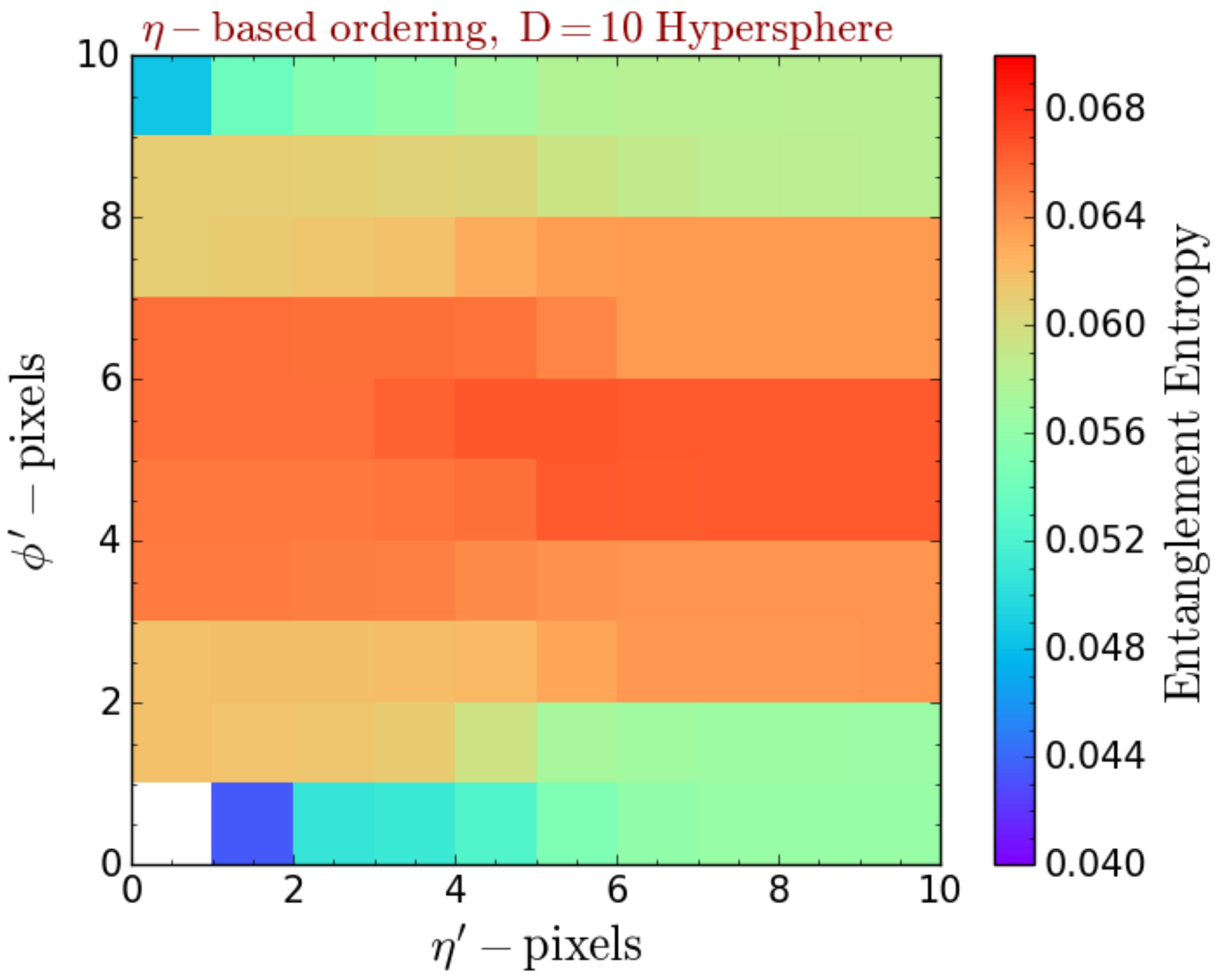


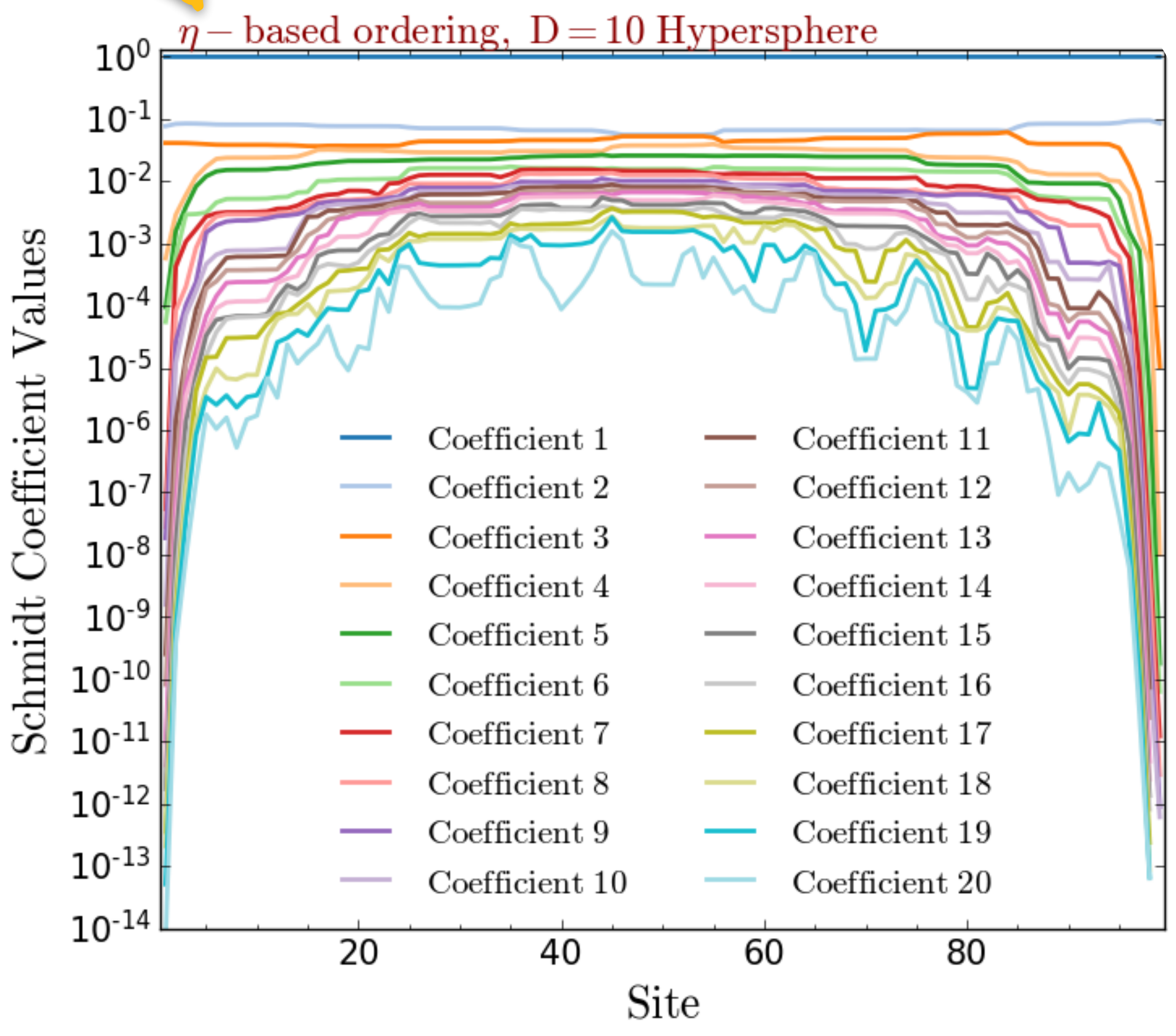
Entanglement Entropy & Schmidt Decompostion



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

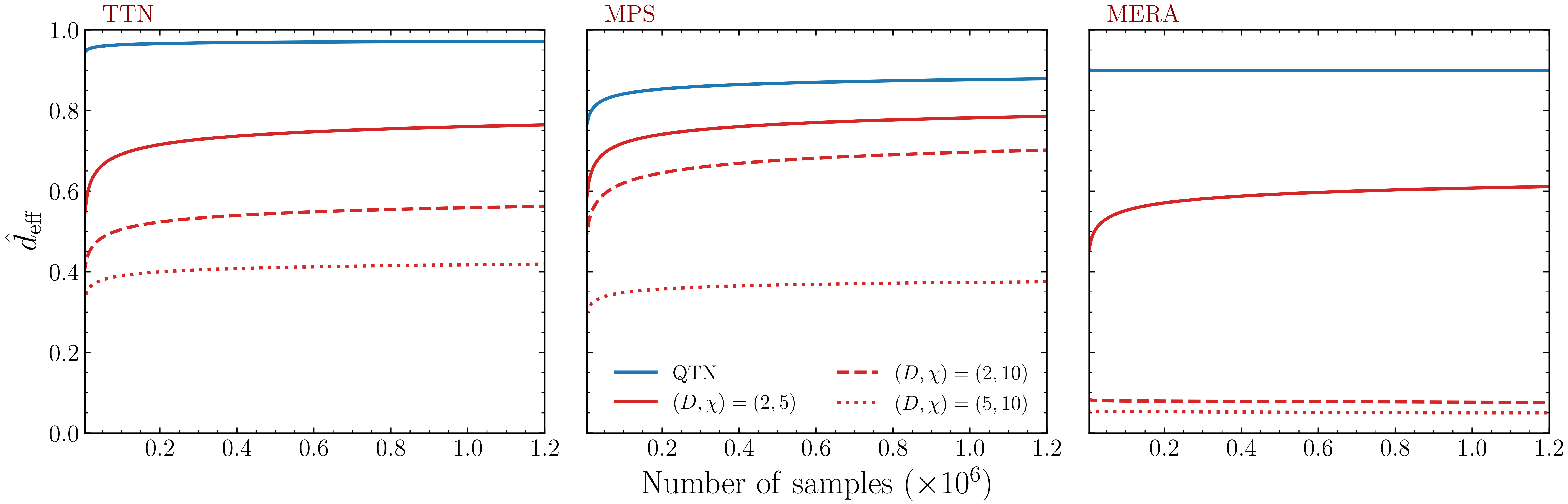
$$\boxed{\mathcal{S} = -\sum_i^{\chi} \lambda_i^2 \log \lambda_i^2}$$





Fisher Information & Effective Dimensions

$$\hat{d}_{\text{eff}} = \frac{2\log\left(\frac{1}{V_\Theta}\int_\Theta \sqrt{\det\left(1+\frac{|\mathbf{X}|}{2\pi\log|\mathbf{X}|}\hat{F}(\theta)\right)}\,d\theta\right)}{d\log\left(\frac{|\mathbf{X}|}{2\pi\log|\mathbf{X}|}\right)}$$



$$\bar{F}(\theta) = \frac{1}{|\mathbf{X}|}\sum_{x\in\mathbf{X}} \partial_\theta \log p(x;\theta)\ \partial_\theta \log p^\top(x;\theta)$$