

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

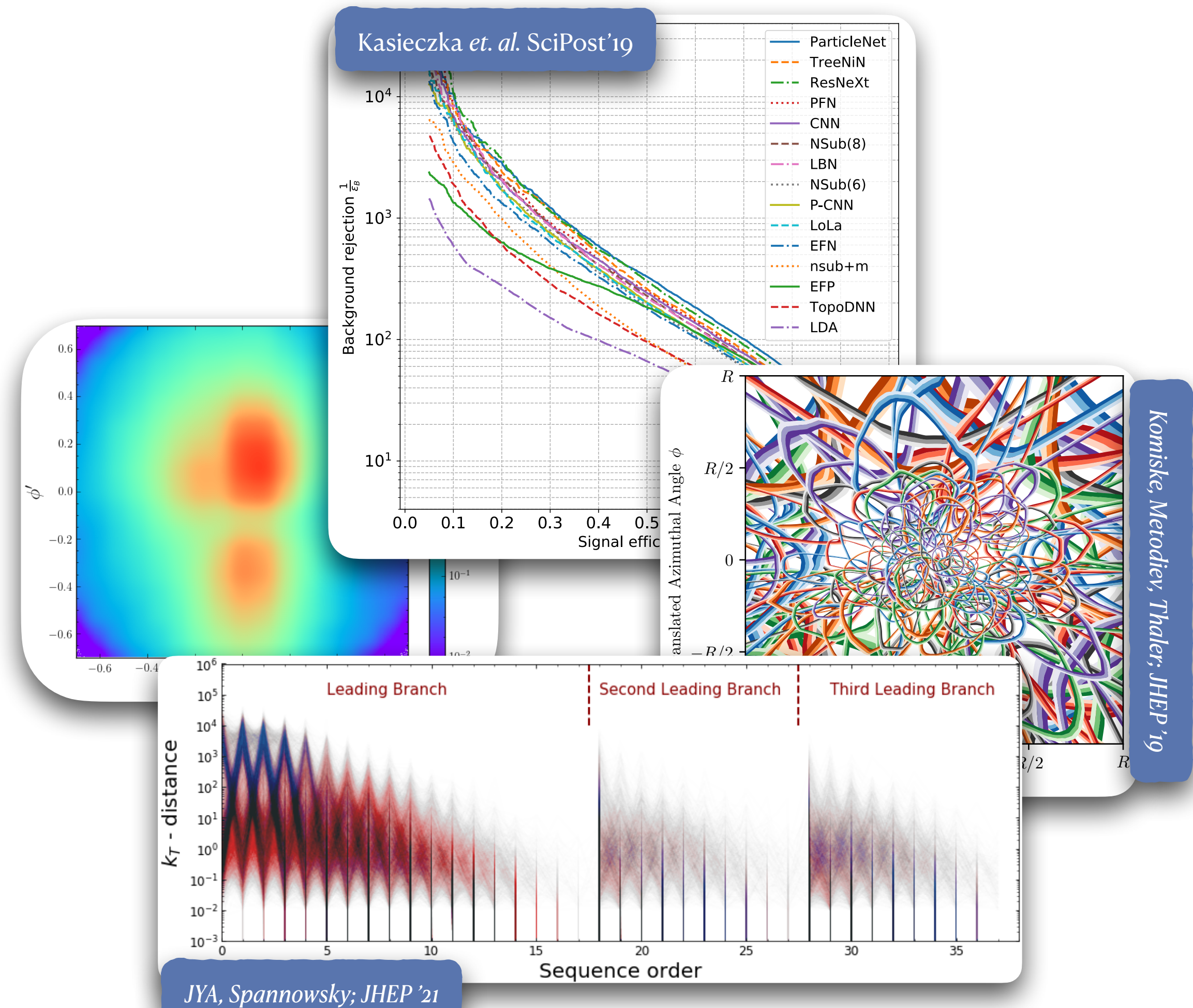
ACAT, Bari, IT.

October 25th, 2022



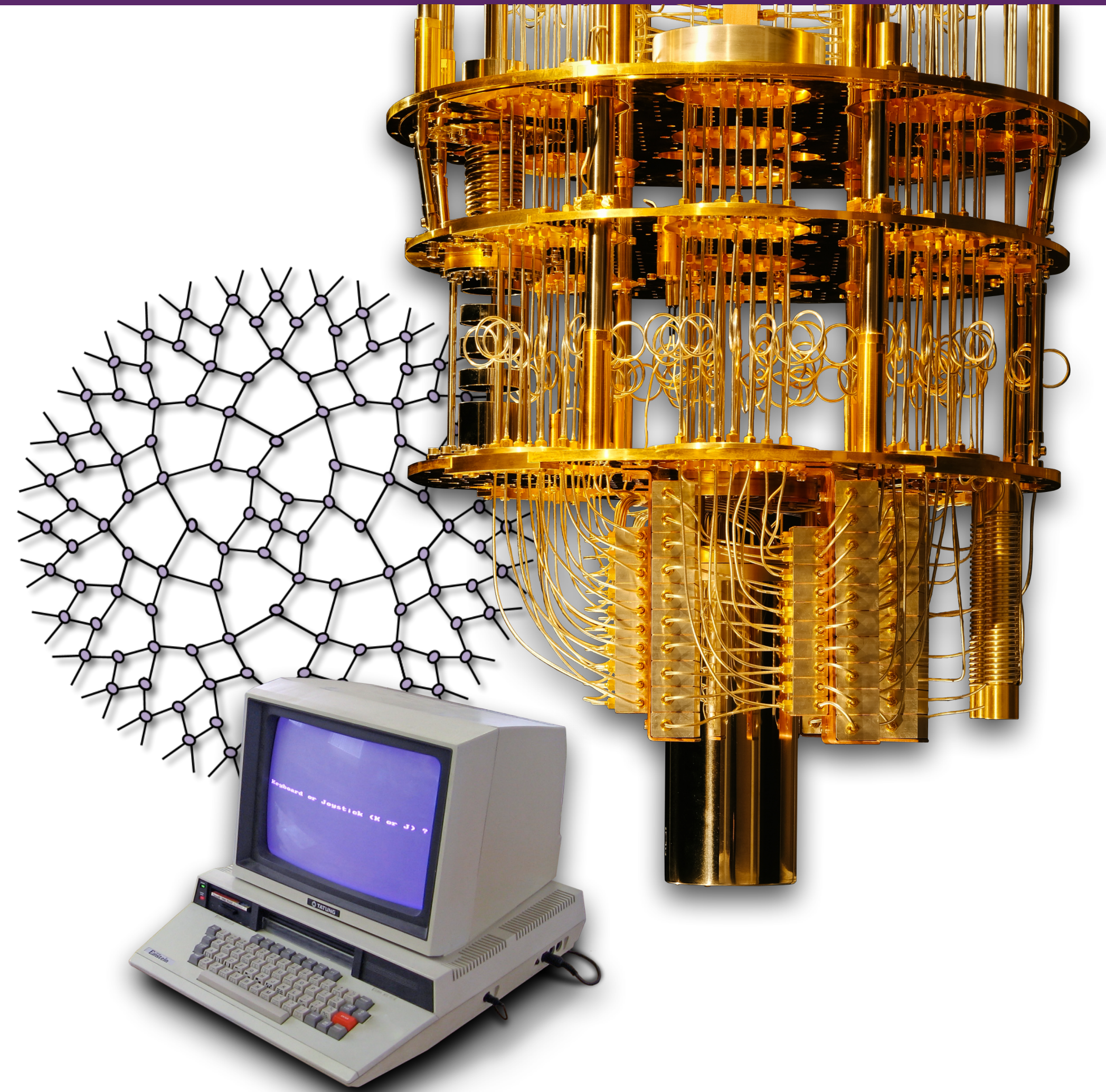
Sales pitch of the talk!

- 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!



Outline

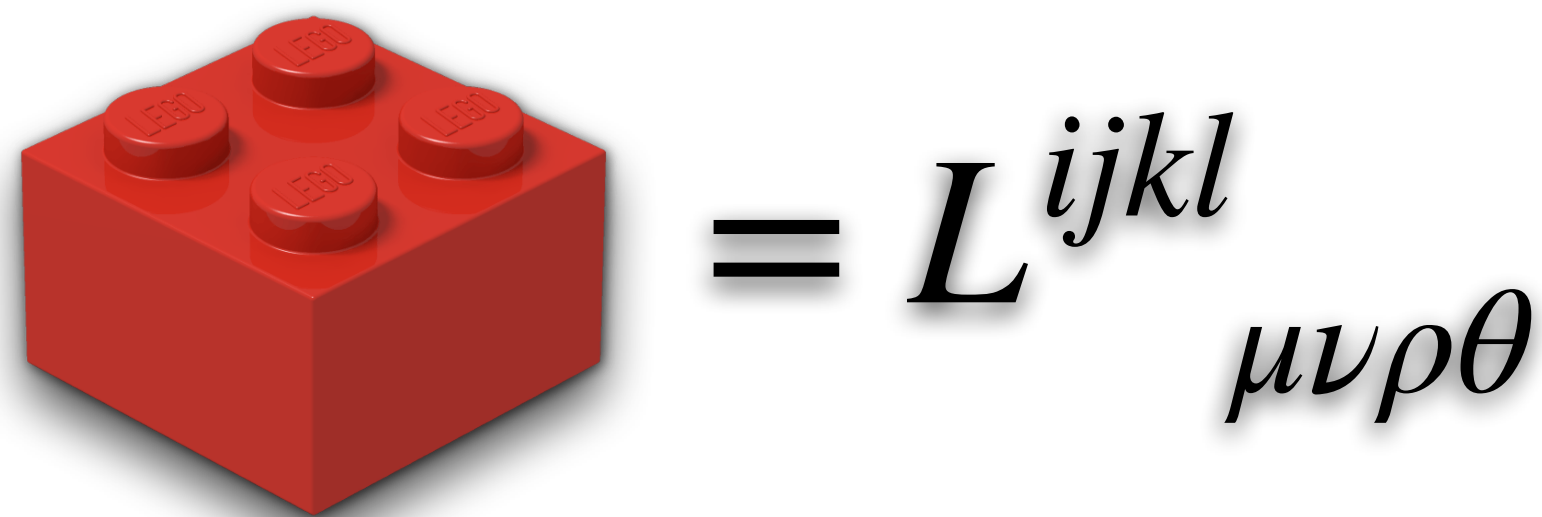
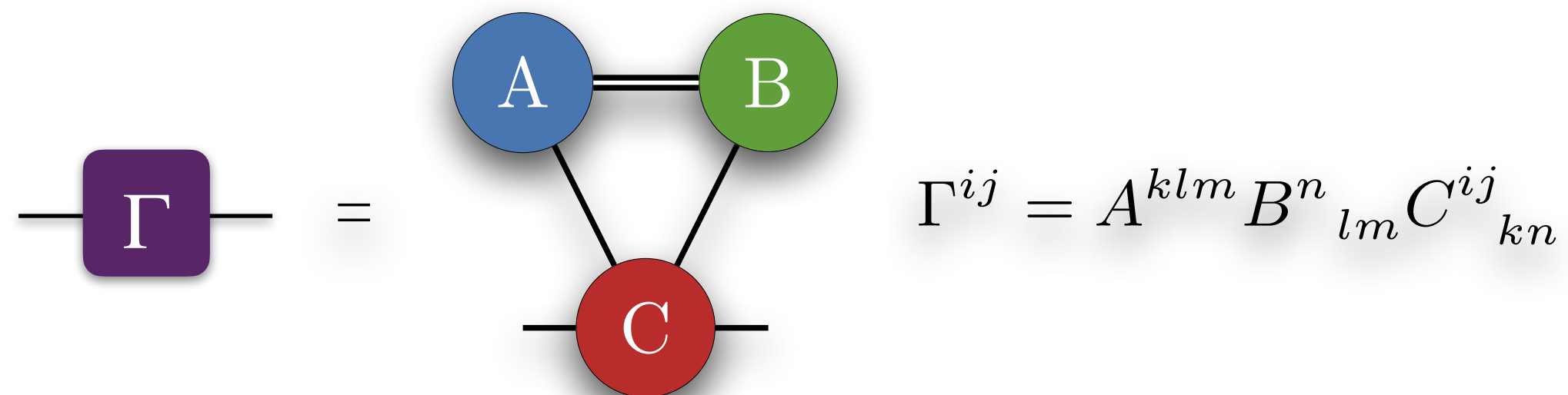
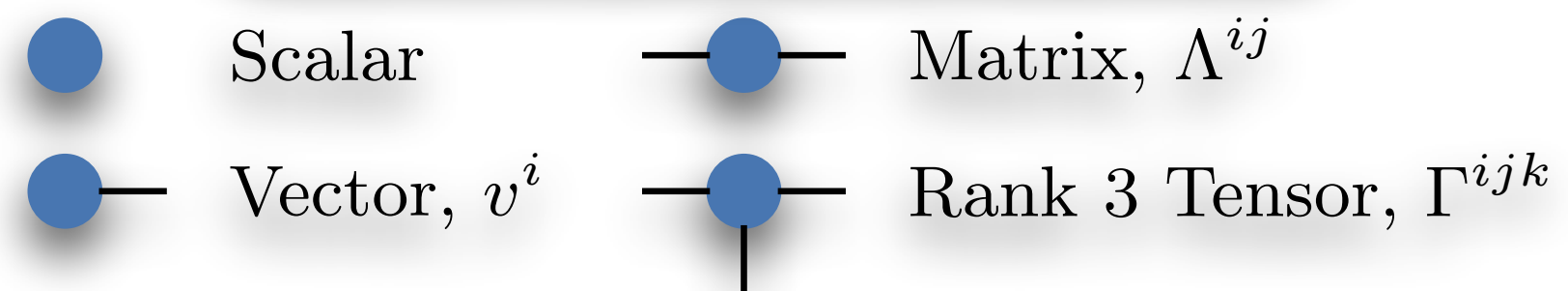
- ❖ Introduction
 - Representing a problem as a quantum many-body system
- ❖ Hello world of HEP-ML: Top Tagging
- ❖ Quantum Machine Learning on a Quantum device
- ❖ Conclusion



Introduction

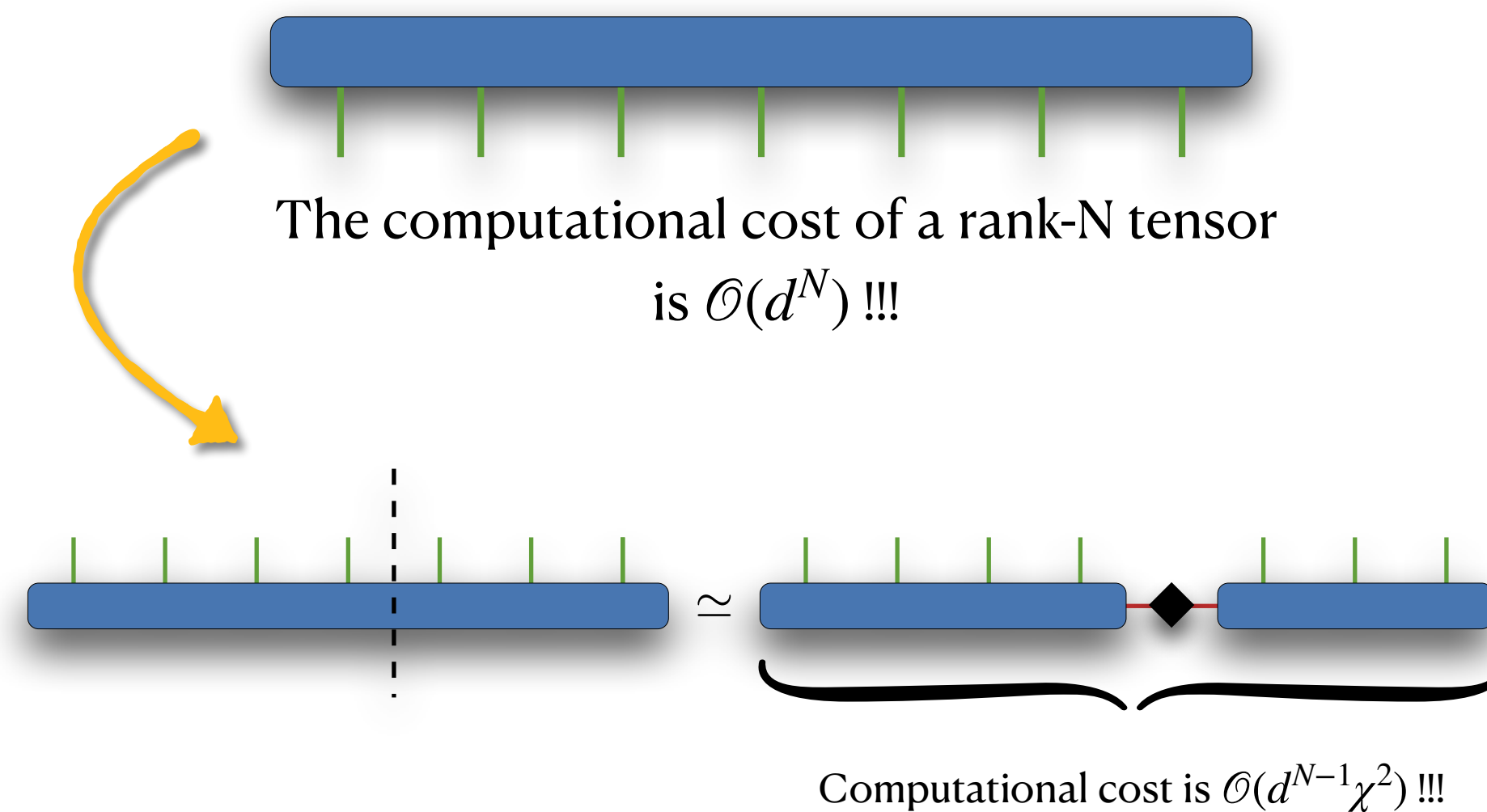
Tensor Networks: Origins

Tensor Diagram Notation

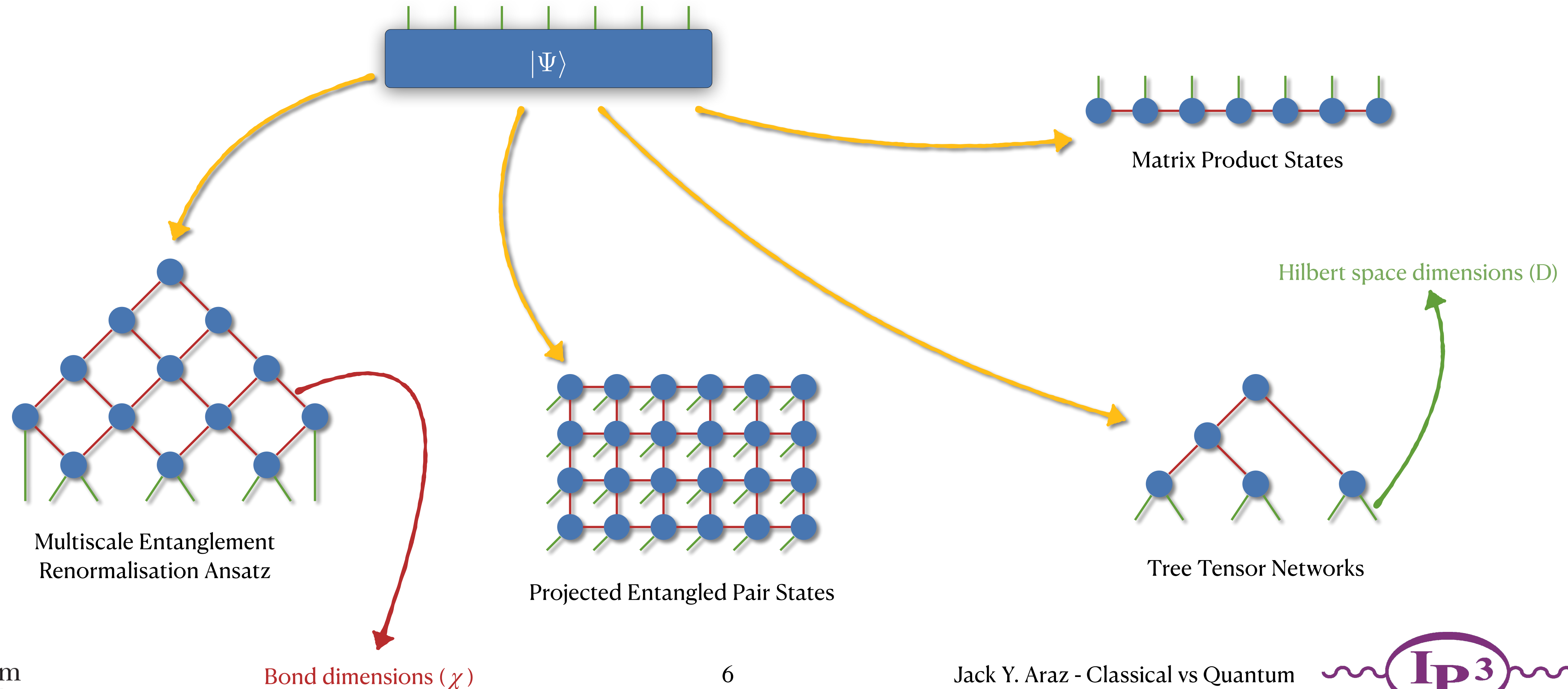


$$|\Psi\rangle = \sum_{\phi_1, \dots, \phi_n=0} \mathcal{W}_{\phi_1 \dots \phi_n} |\phi_1\rangle \otimes |\phi_2\rangle \otimes \dots \otimes |\phi_n\rangle$$

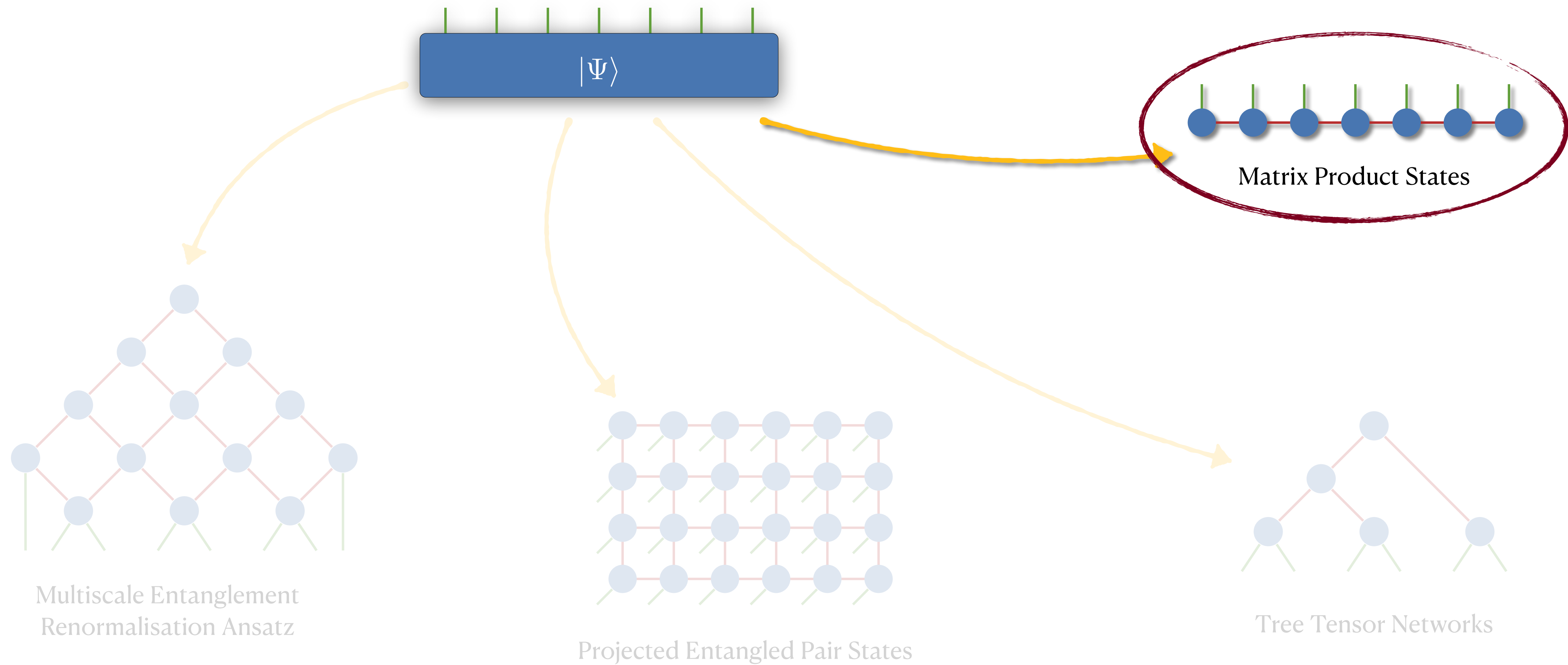
$$\forall |\phi_i\rangle \in \mathcal{H}^{\otimes 2^N} \rightarrow |\phi_i\rangle \in \{|\uparrow\rangle, |\downarrow\rangle\}$$



Types of Tensor Networks (some of them)



Types of Tensor Networks (some of them)



Matrix Product States for Classification

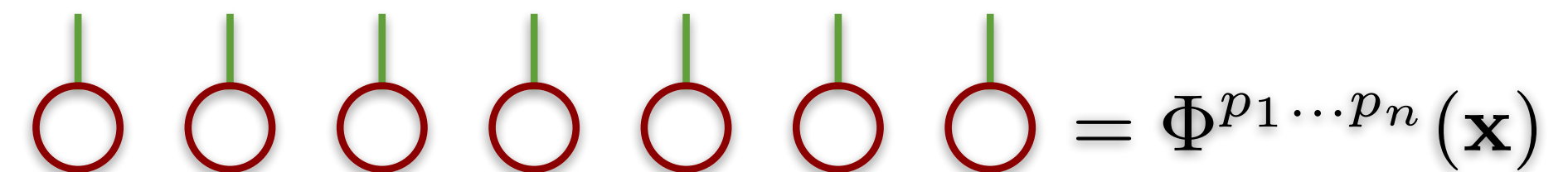
Sub-Outline

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

Data Embedding

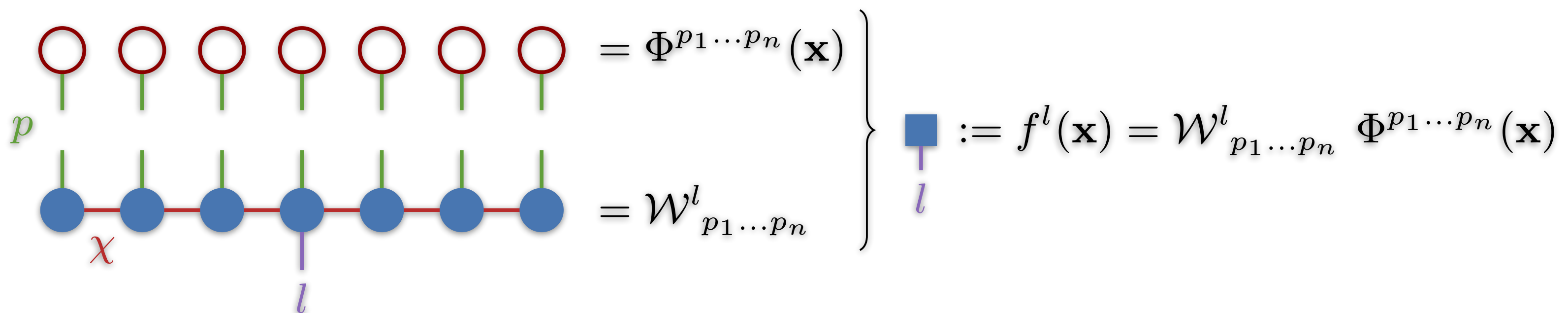
$$\mathbf{X} = \{x_1, x_2, \dots, x_n\} \in \mathbb{R} \quad , \quad \phi(x) := \forall x \in \mathbb{R} \rightarrow \mathbb{C}^m$$

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



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

Little modification

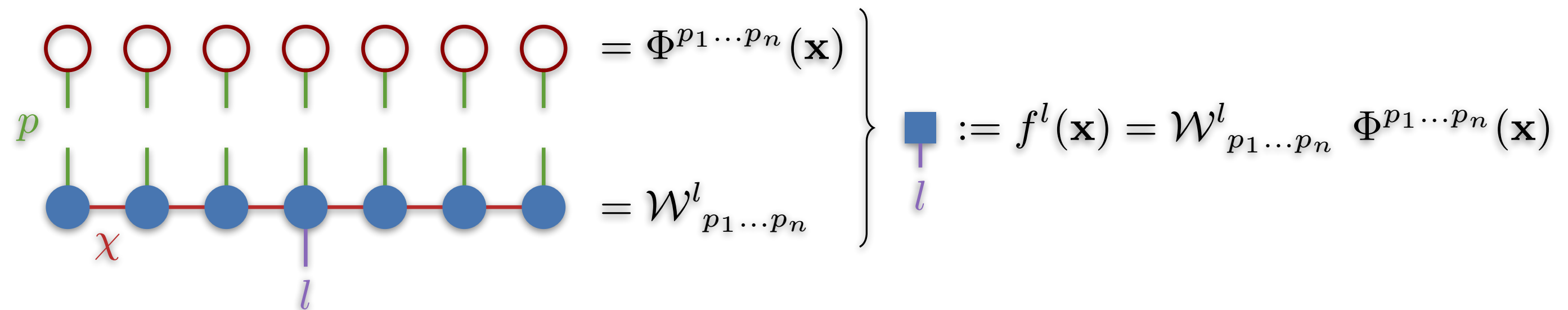


Trainable tensors: $\mathcal{W}(\theta_i)$

Matrix Product States for Classification

Sub-Outline

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



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

Or anything else you like to minimise...

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

**No activation function!!
Everything is linear!!**

TNs intrinsically have "topological nonlinearity"

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

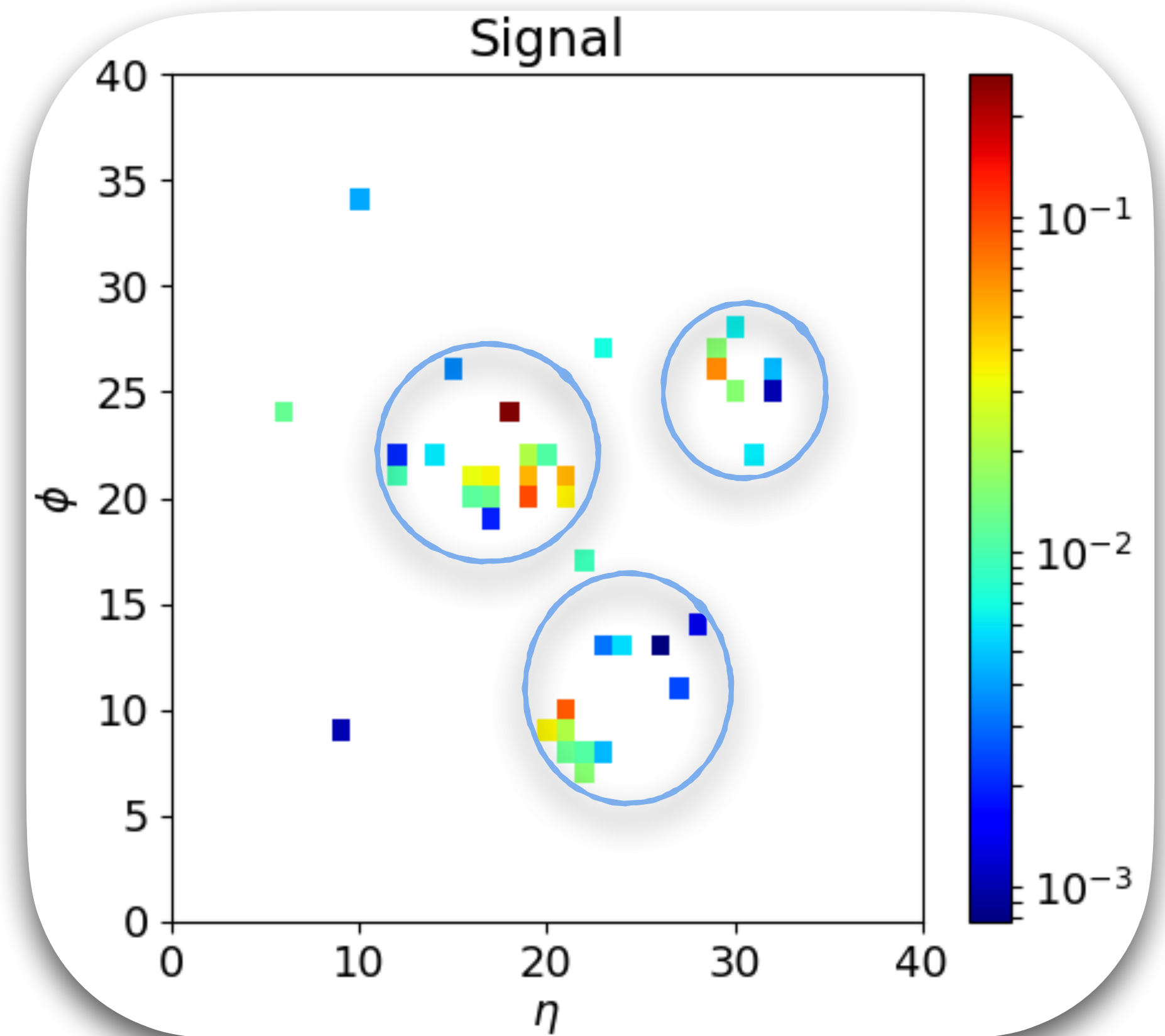
$$\arg \min_{\theta_i \in \mathcal{W}} \mathcal{L}(q(x^{(i)}), p(x^{(i)}; \theta))$$

Hello World of HEP-ML: Top Tagging

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

Data from: [Kasieczka et. al. SciPost'19](#)

- ❖ Leading FatJet Definition: anti- k_T algorithm with $R = 0.8, p_T \in [550, 650] \text{ GeV}, |\eta| < 2$
- ❖ 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.

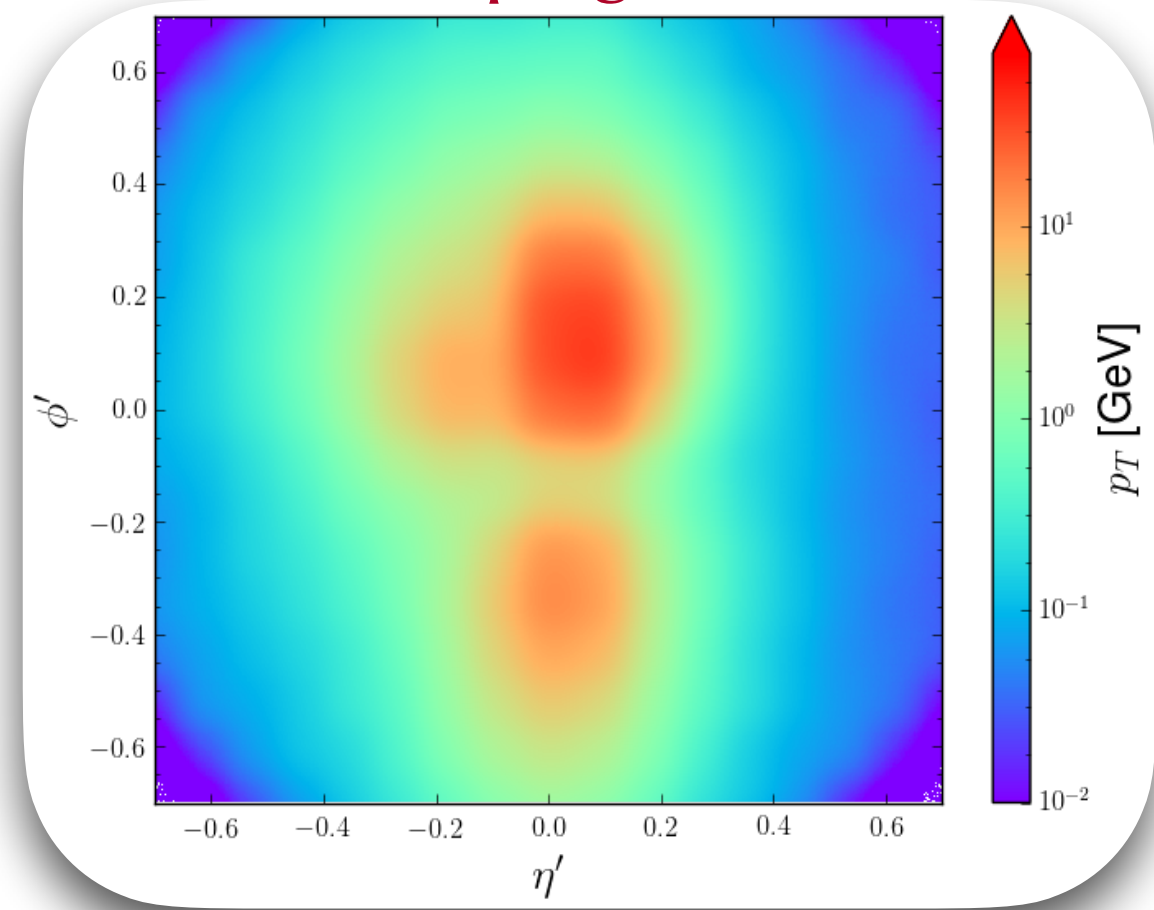
Similar preprocess, based on CNN:

[Kasieczka, Plehn, Russell, Schell; JHEP '17](#)

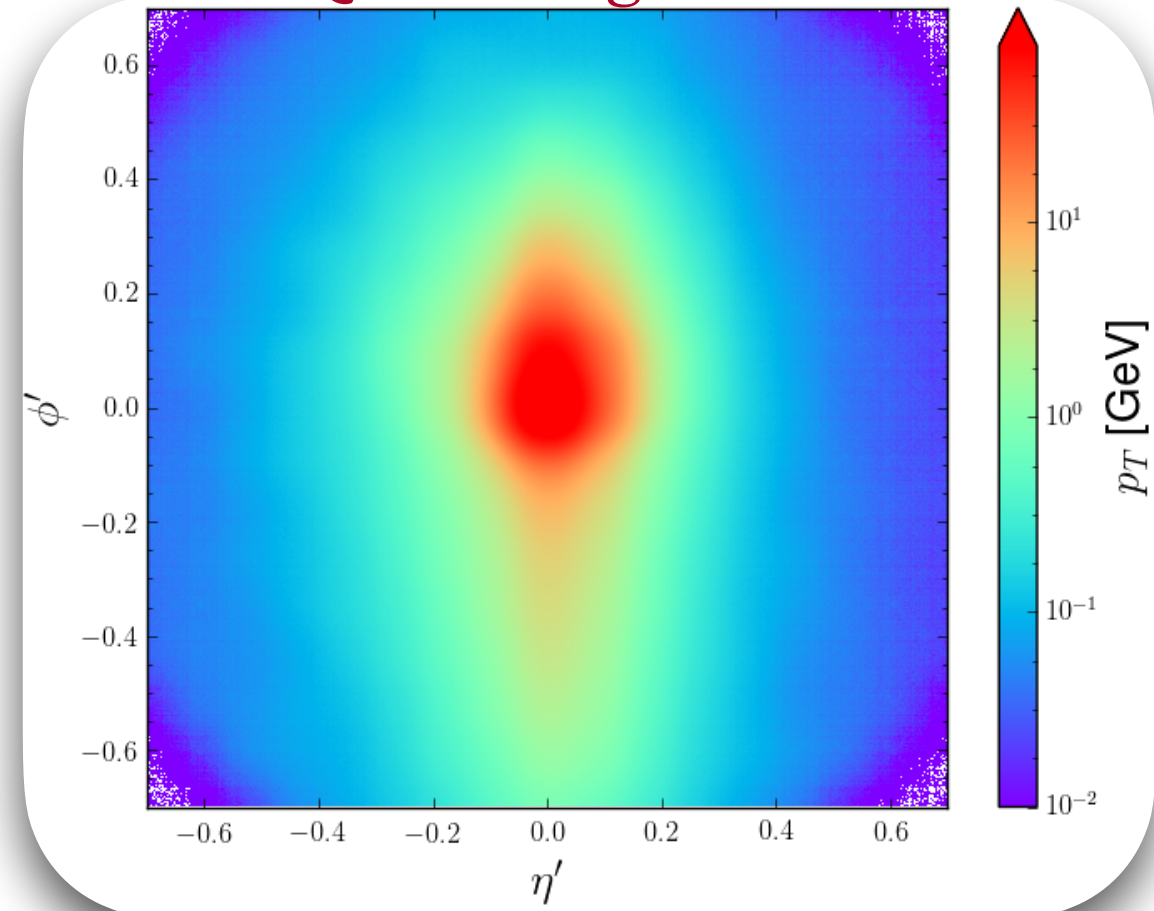
[Macaluso, Shih; JHEP '18](#)

[JYA, Spannowsky; JHEP '21](#)

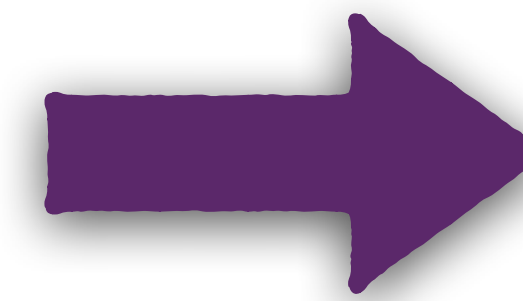
Top Signal



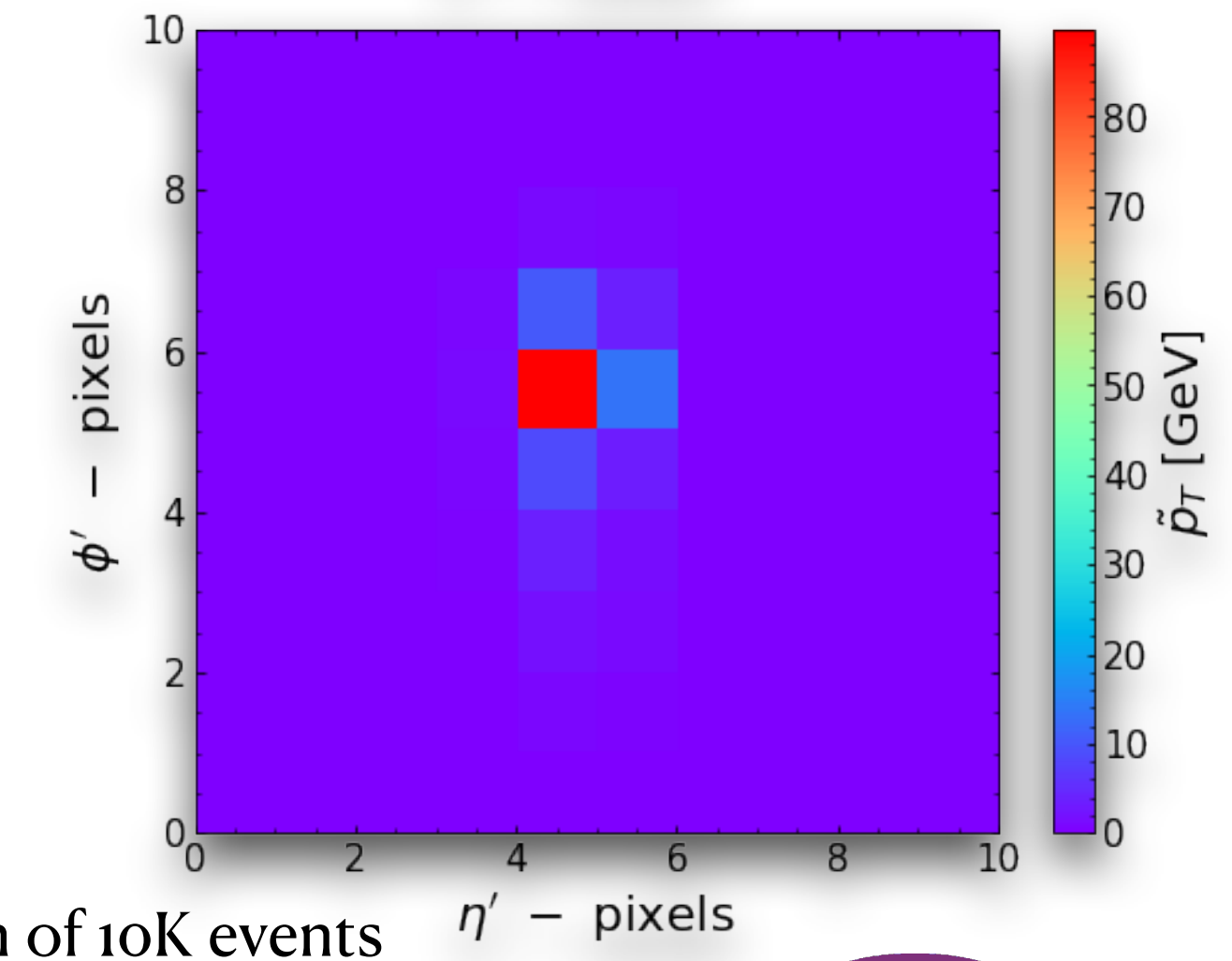
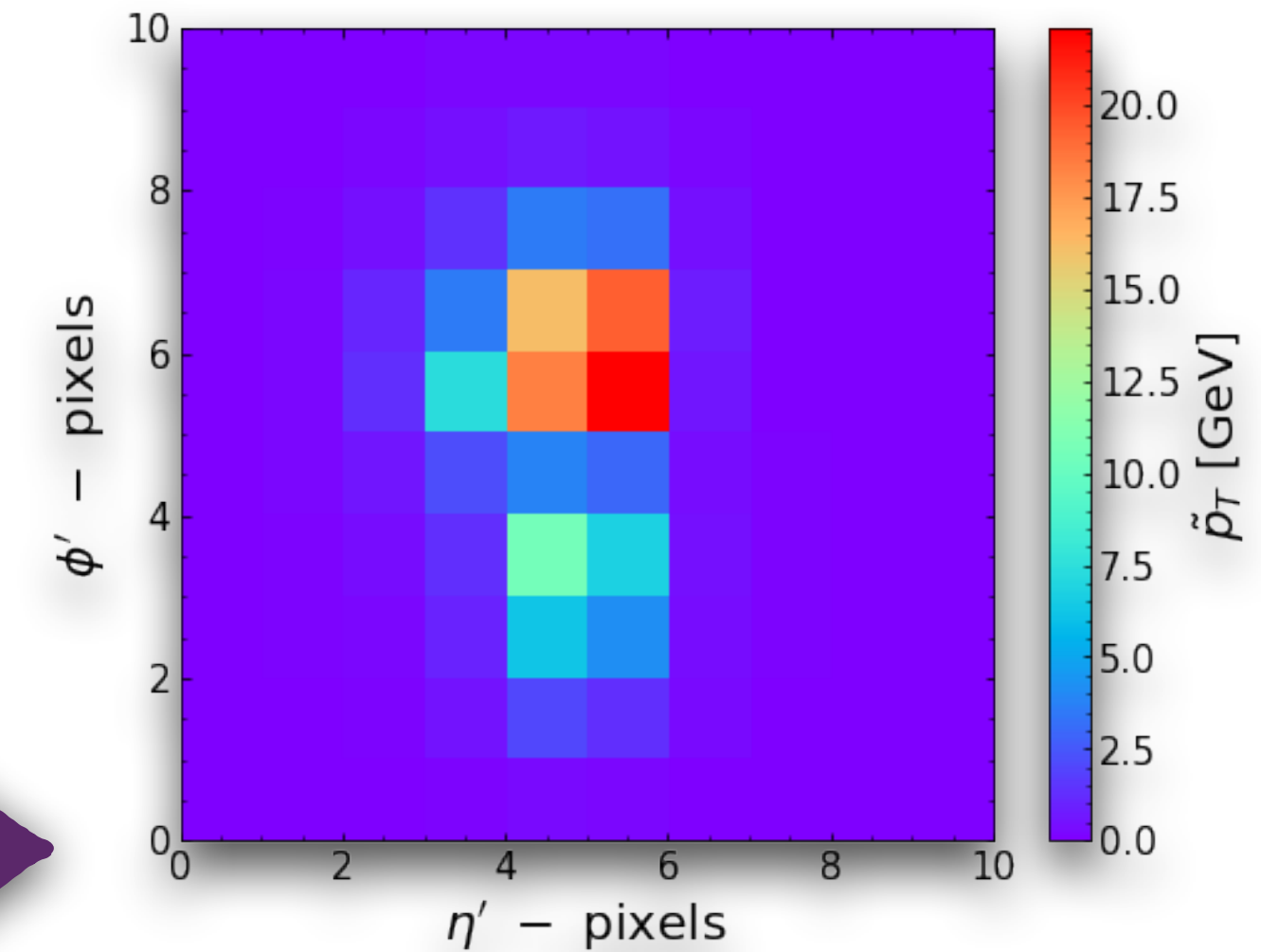
QCD Background



Mean of 10K events

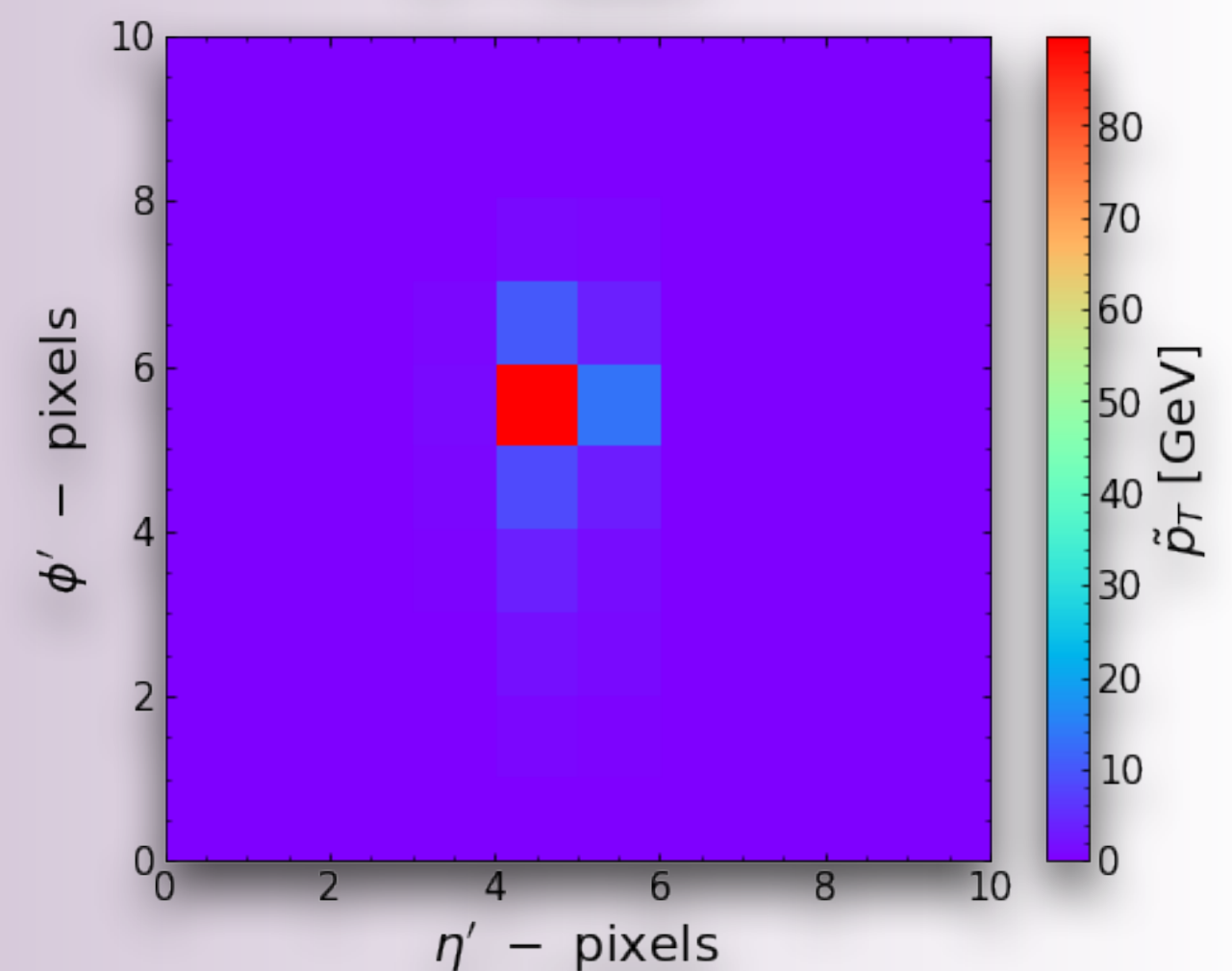
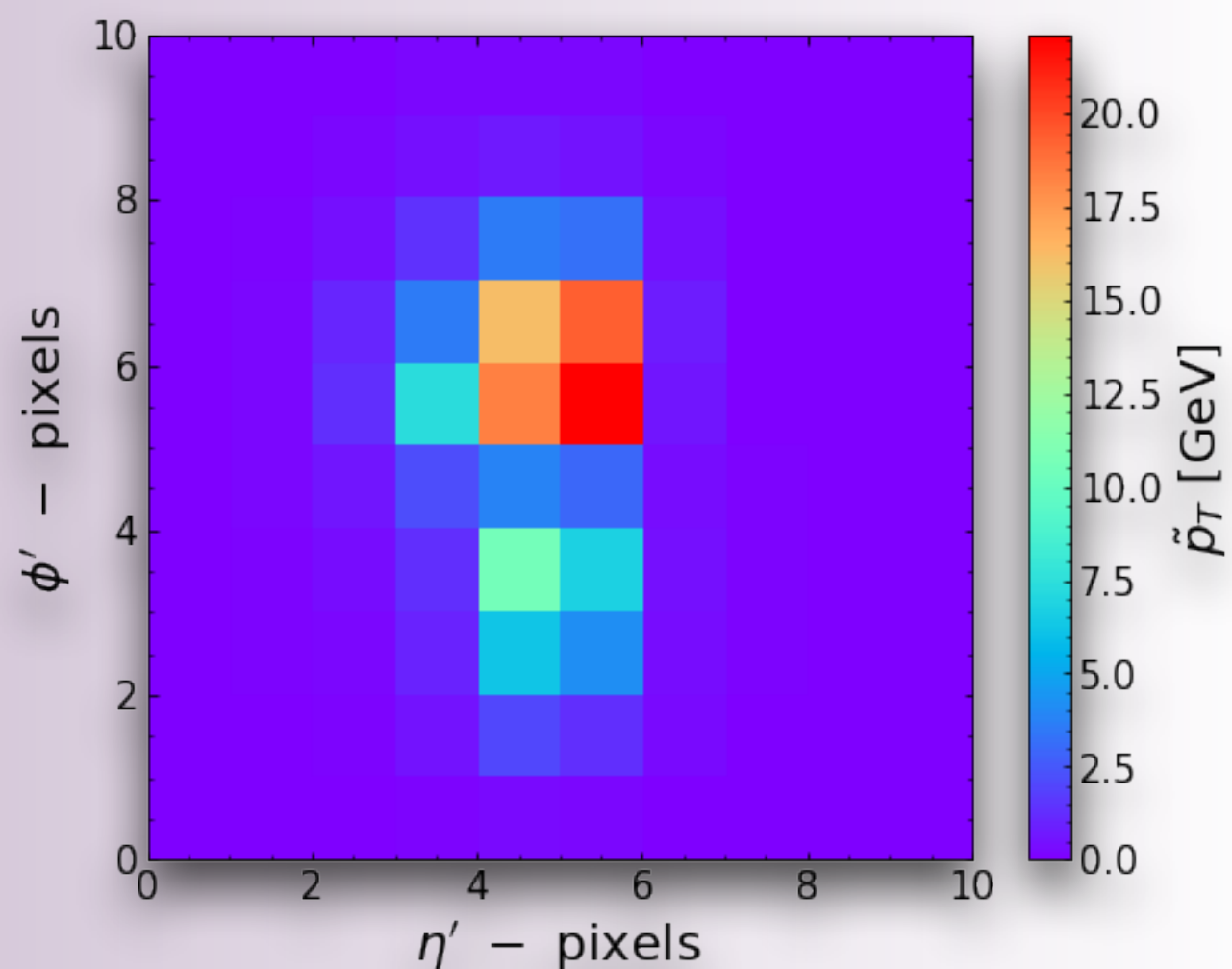


Crop & downsample

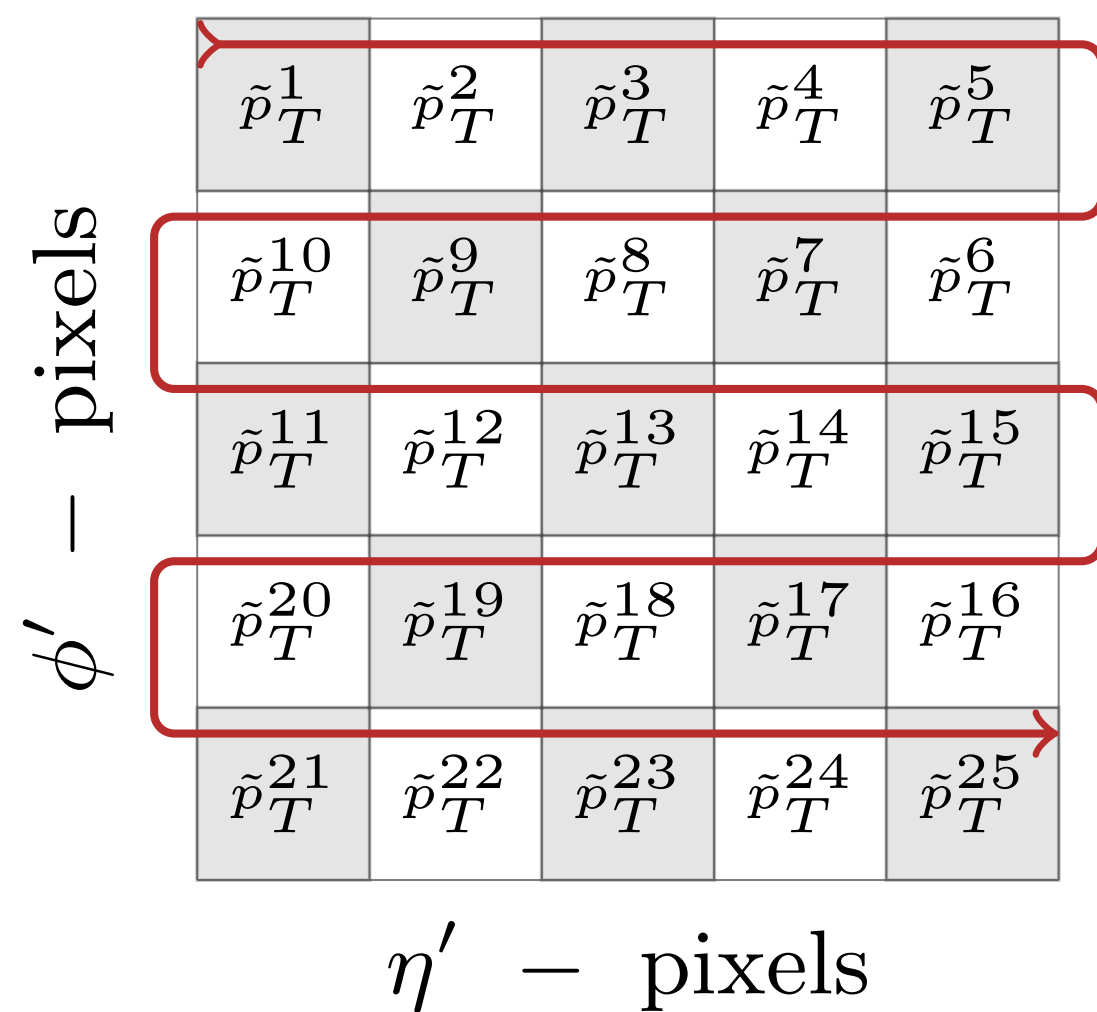


Mean of 10K events

Top Tagging through MPS



η - based ordering



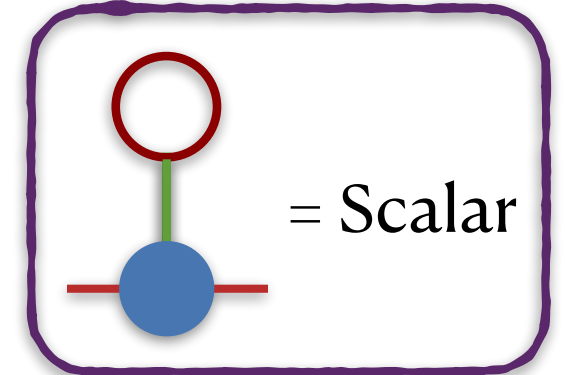
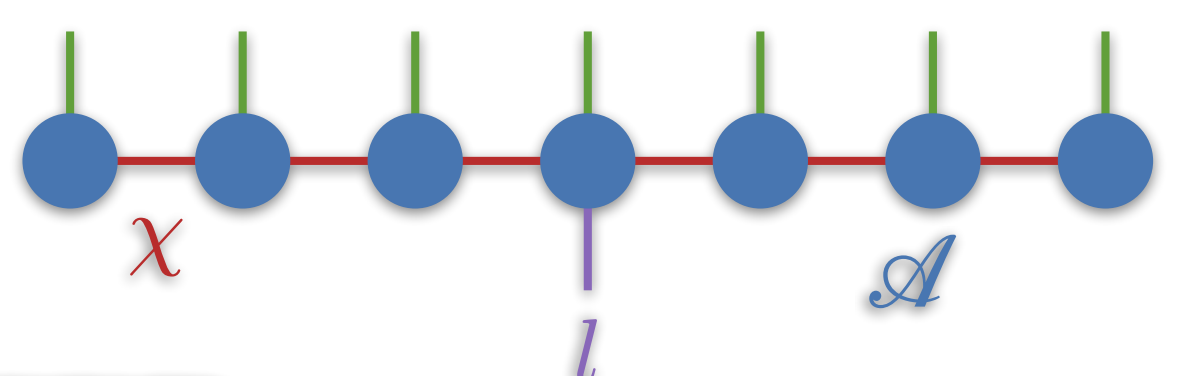
Data Embedding

$$\Phi^{p_1 \dots p_n}(\mathbf{x}) = \bigotimes_{i=1}^N \phi^{p_i}(\tilde{p}_T^i)$$

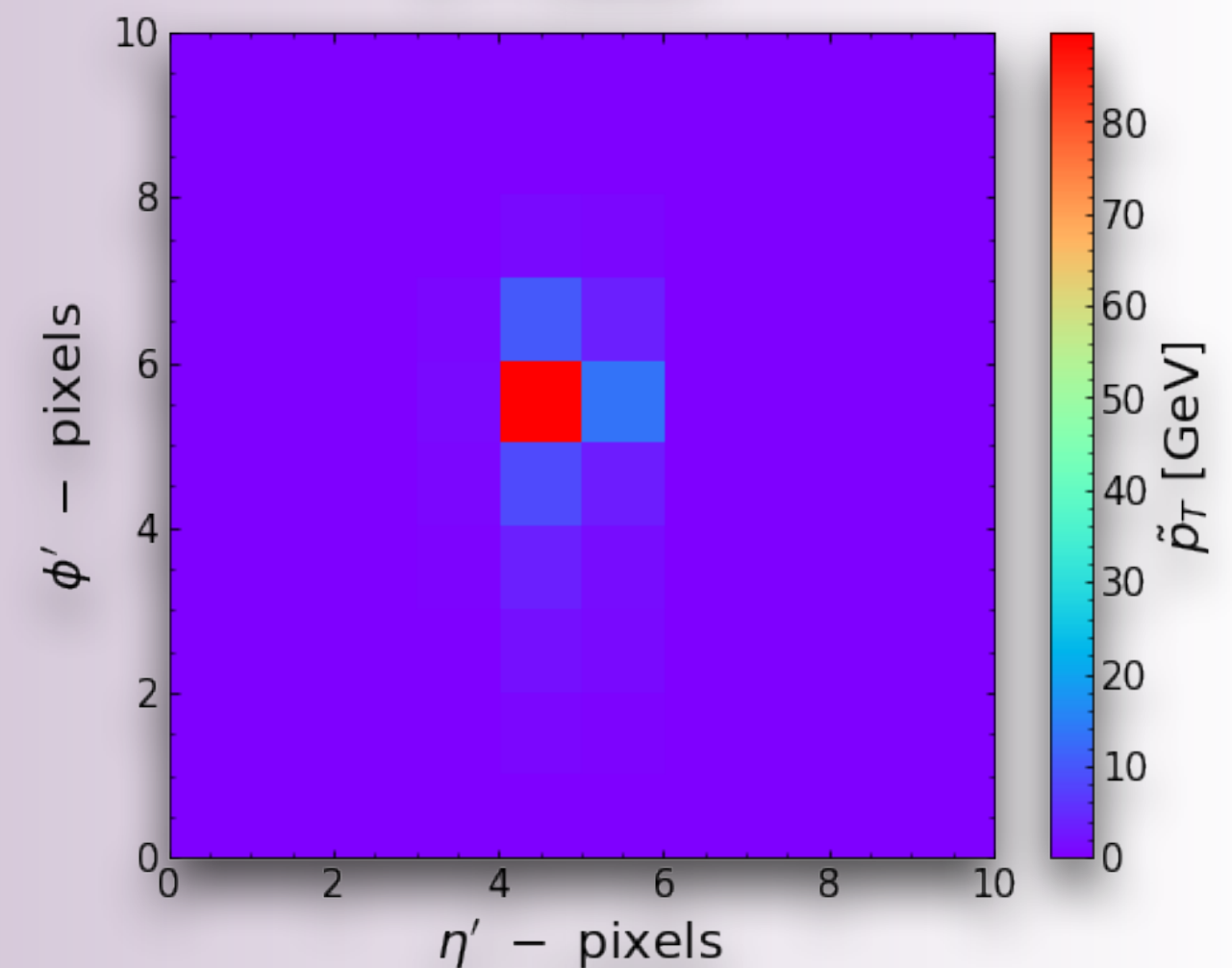
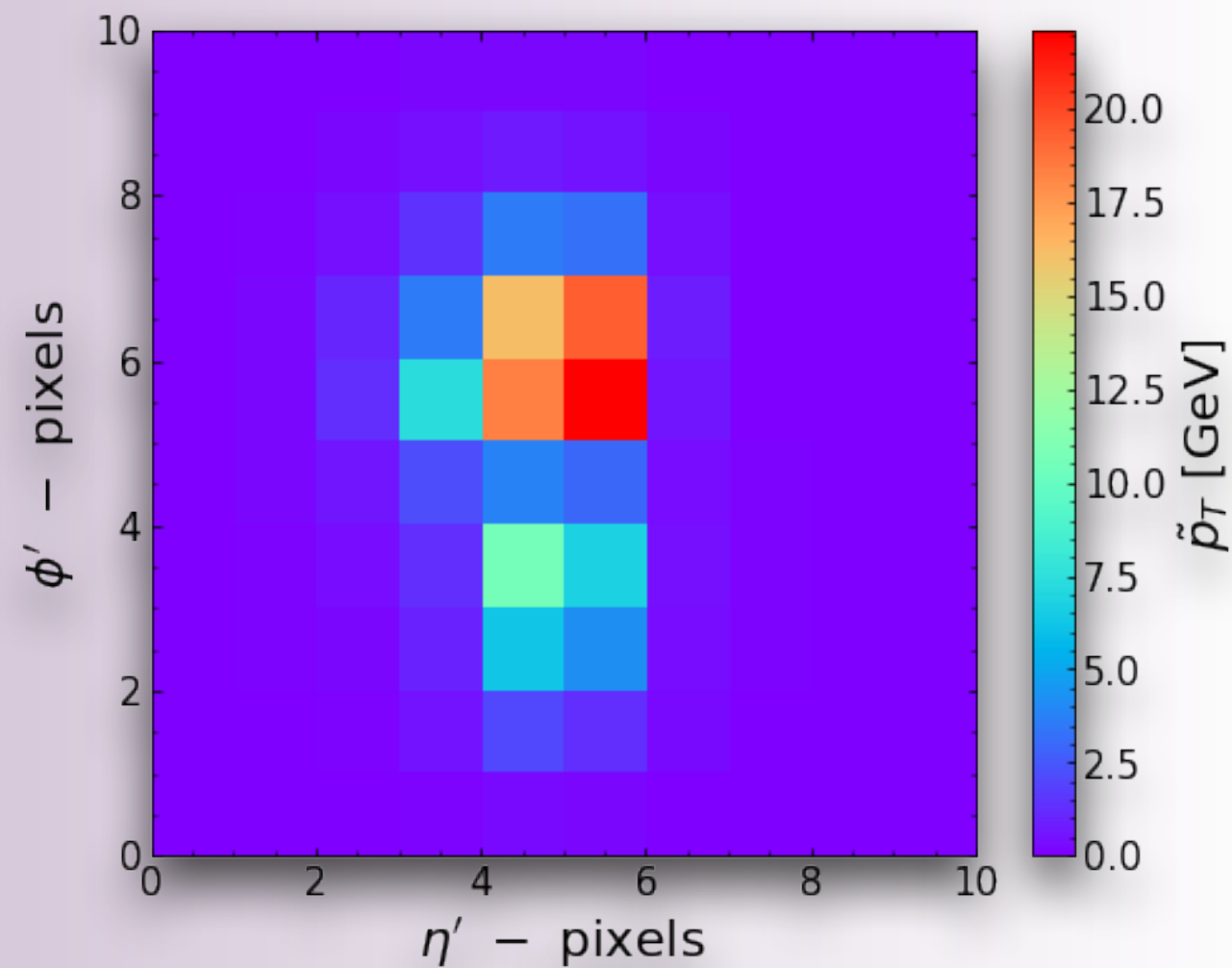
$$\phi^{p_i}(\tilde{p}_T^i) := \forall \tilde{p}_T \in \mathbb{R} \rightarrow \mathbb{C}^{10} = \sqrt{\binom{D-1}{d_i-1}} \cos^{D-d_i} \left(\tilde{p}_T^i \frac{\pi}{2} \right) \sin^{d_i-1} \left(\tilde{p}_T^i \frac{\pi}{2} \right)$$

Network Initial Condition

Each feature assumed to be uncorrelated

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


Top Tagging through MPS

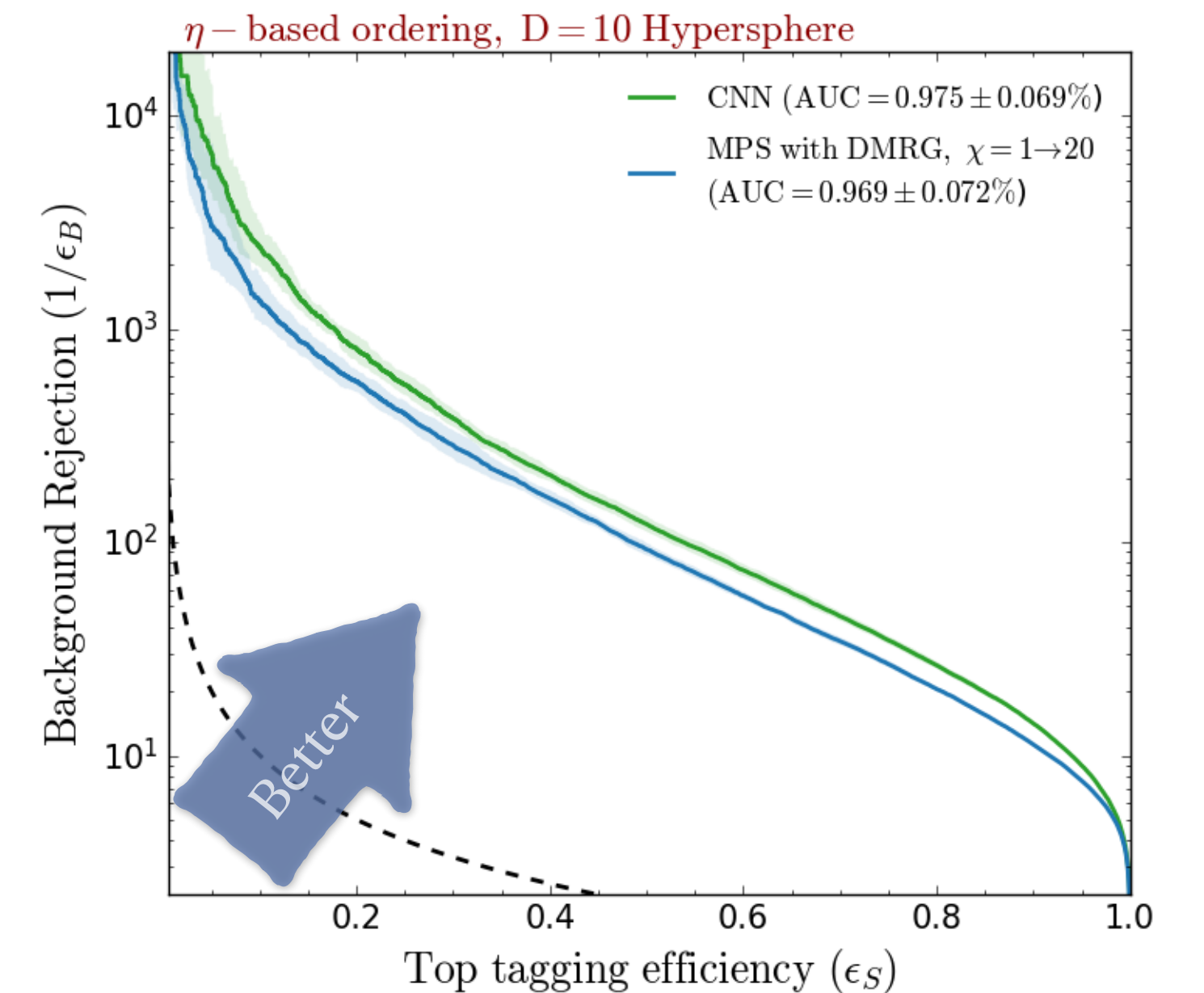
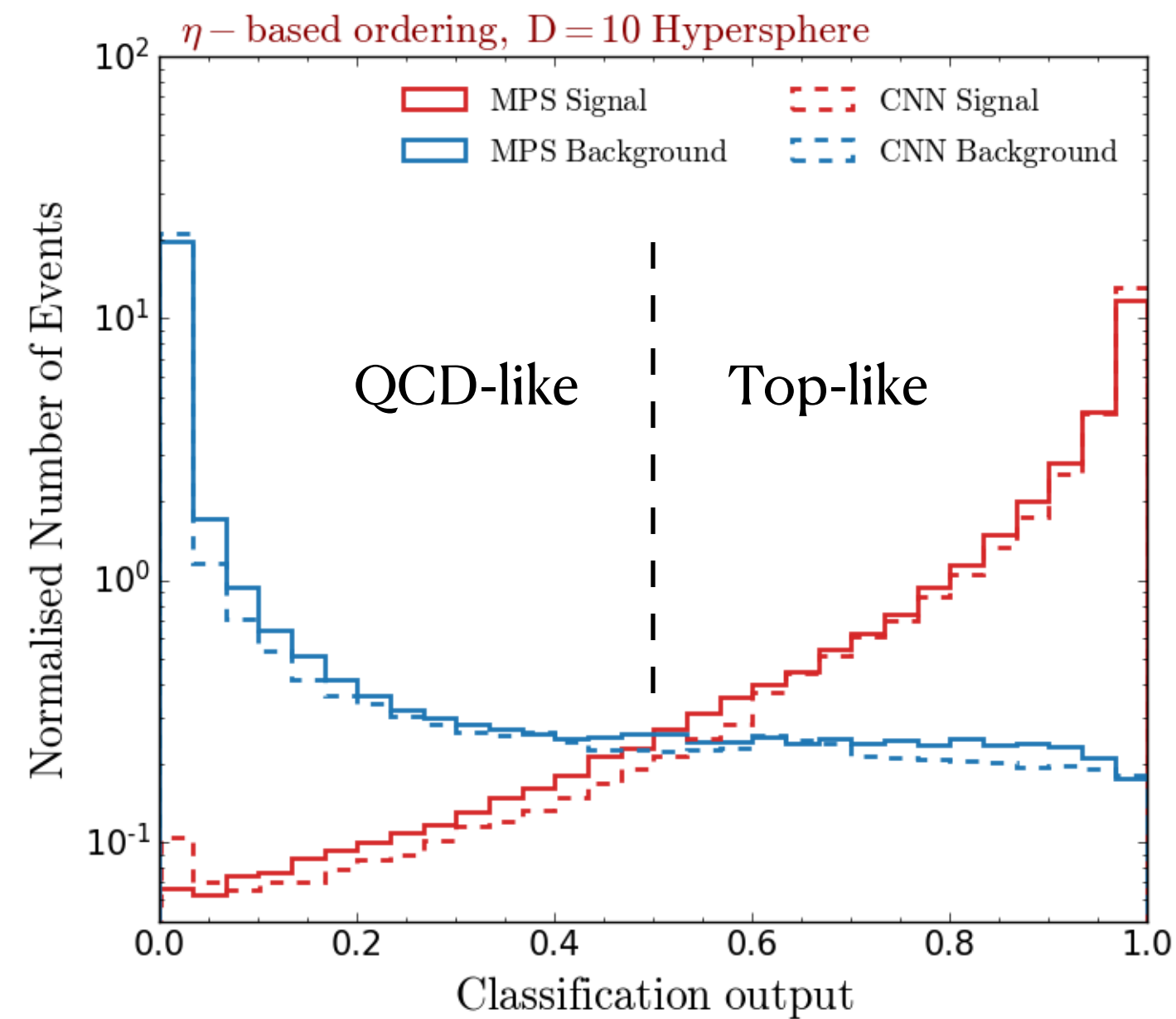


Assumptions & Requirements

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

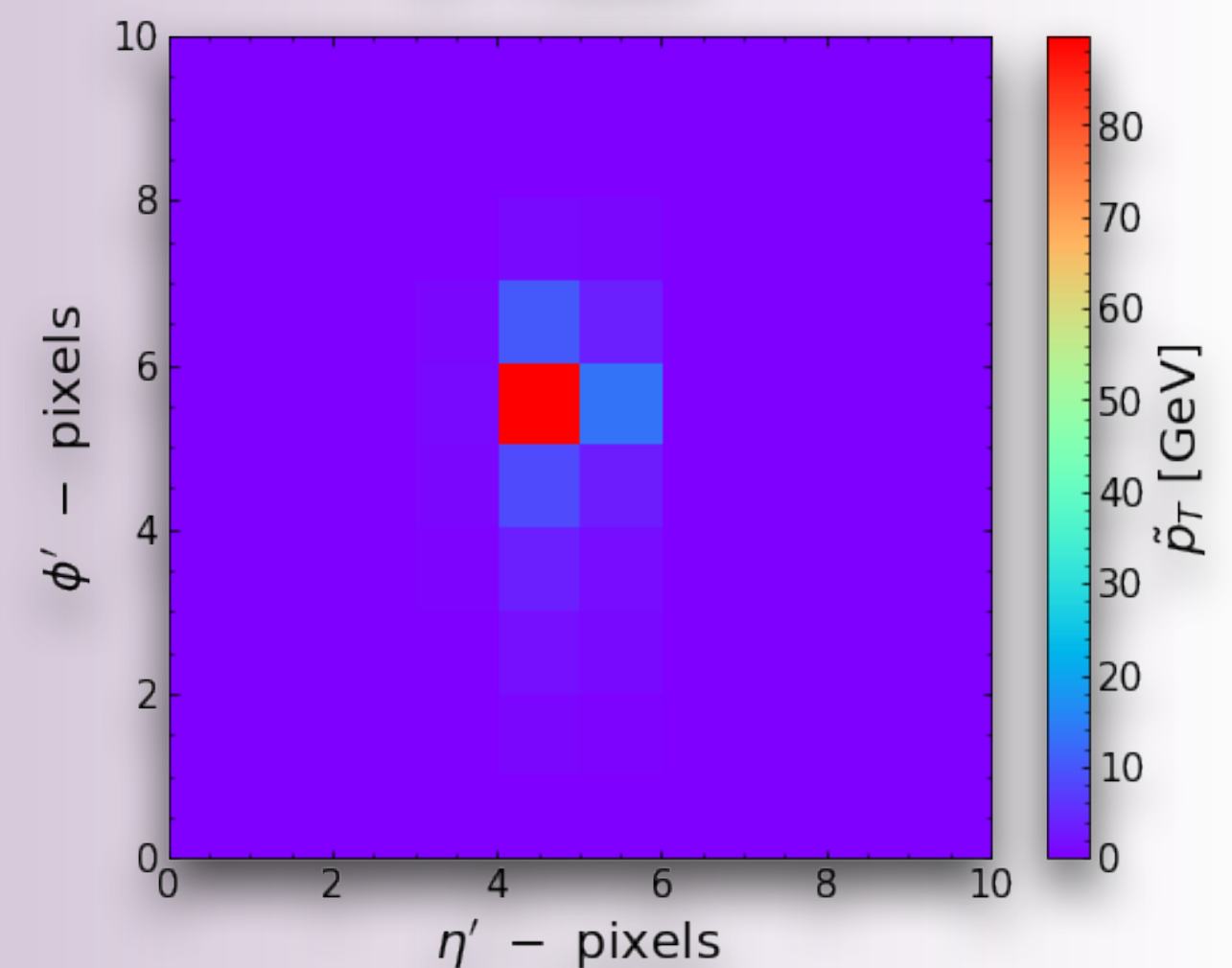
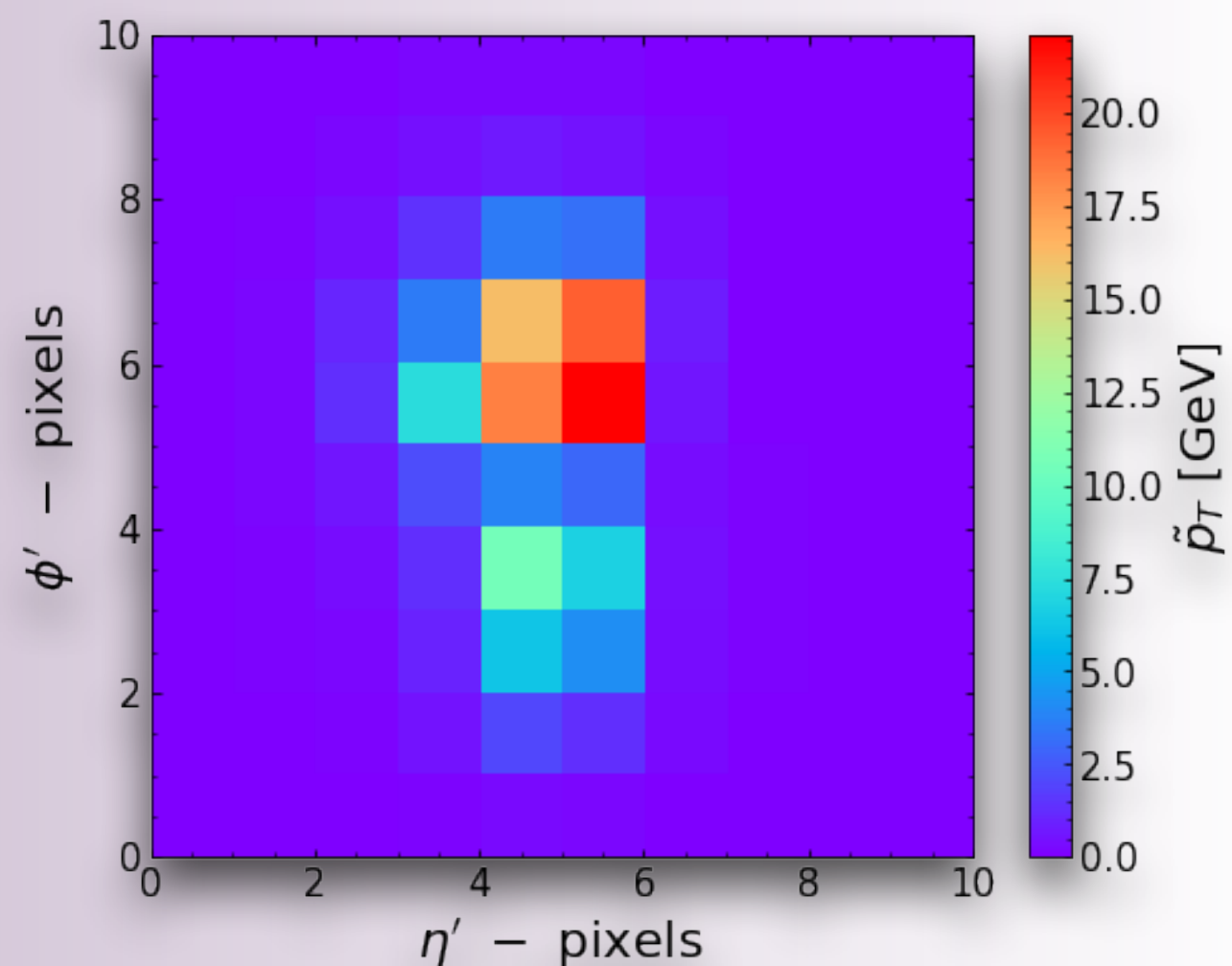
CNN architecture from:

JYA, Spannowsky; JHEP '21



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

Top Tagging through MPS



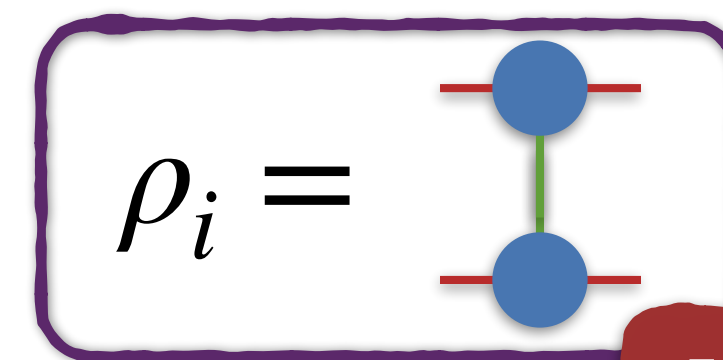
η - based ordering

\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

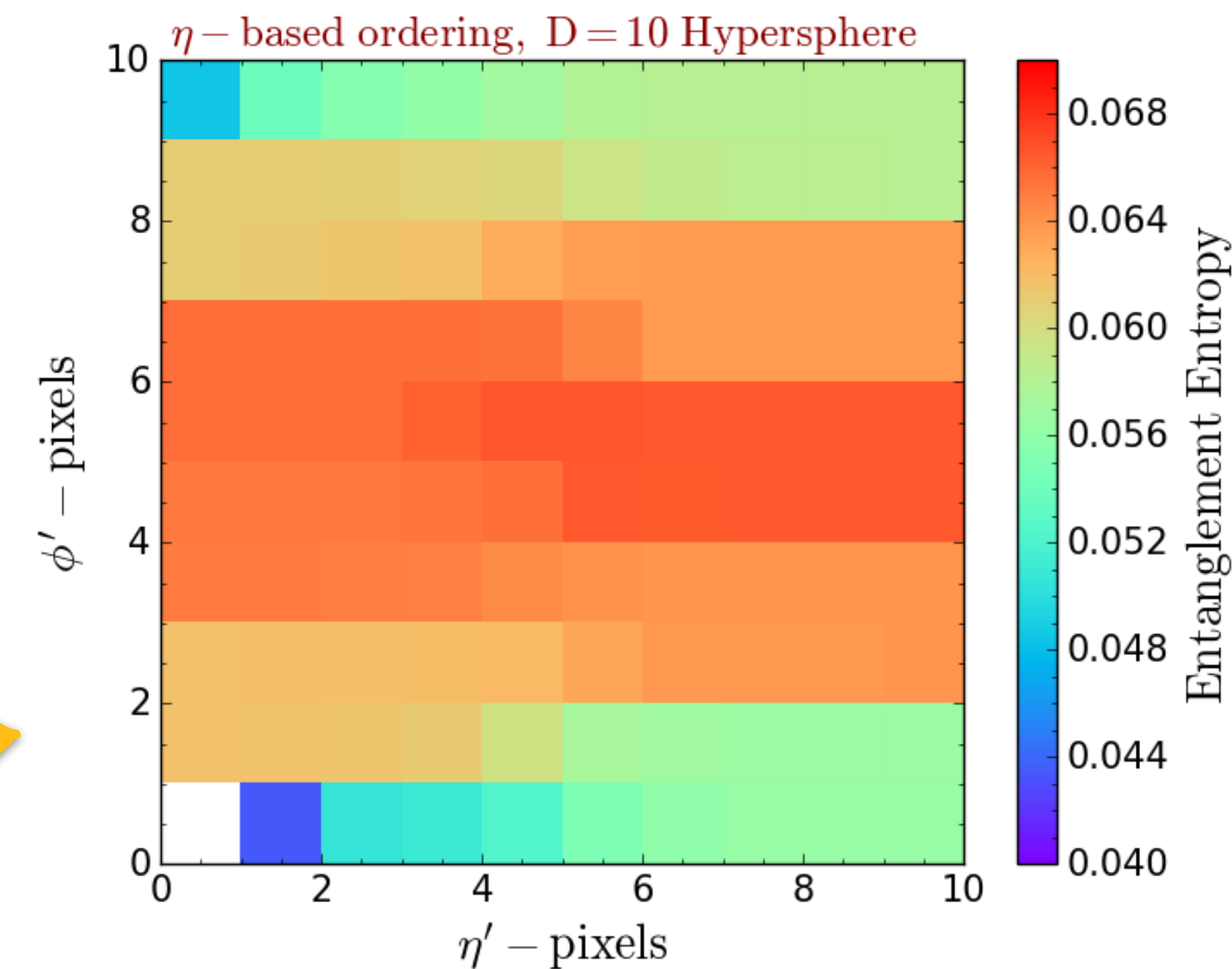
$\mathcal{S}(\rho)$ is a quantifiable measure to understand which pixels are not crucial for the process without the data!!!



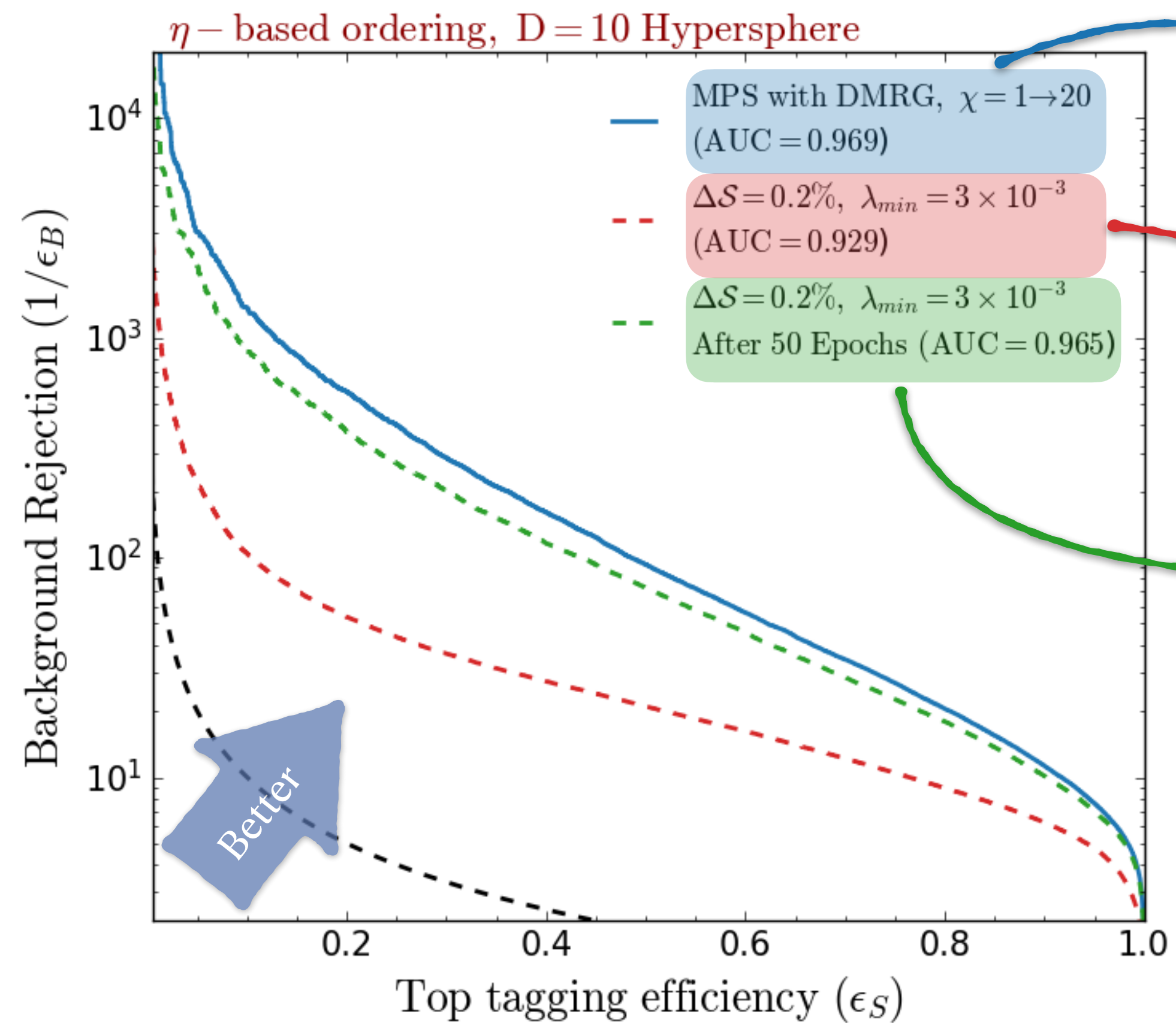
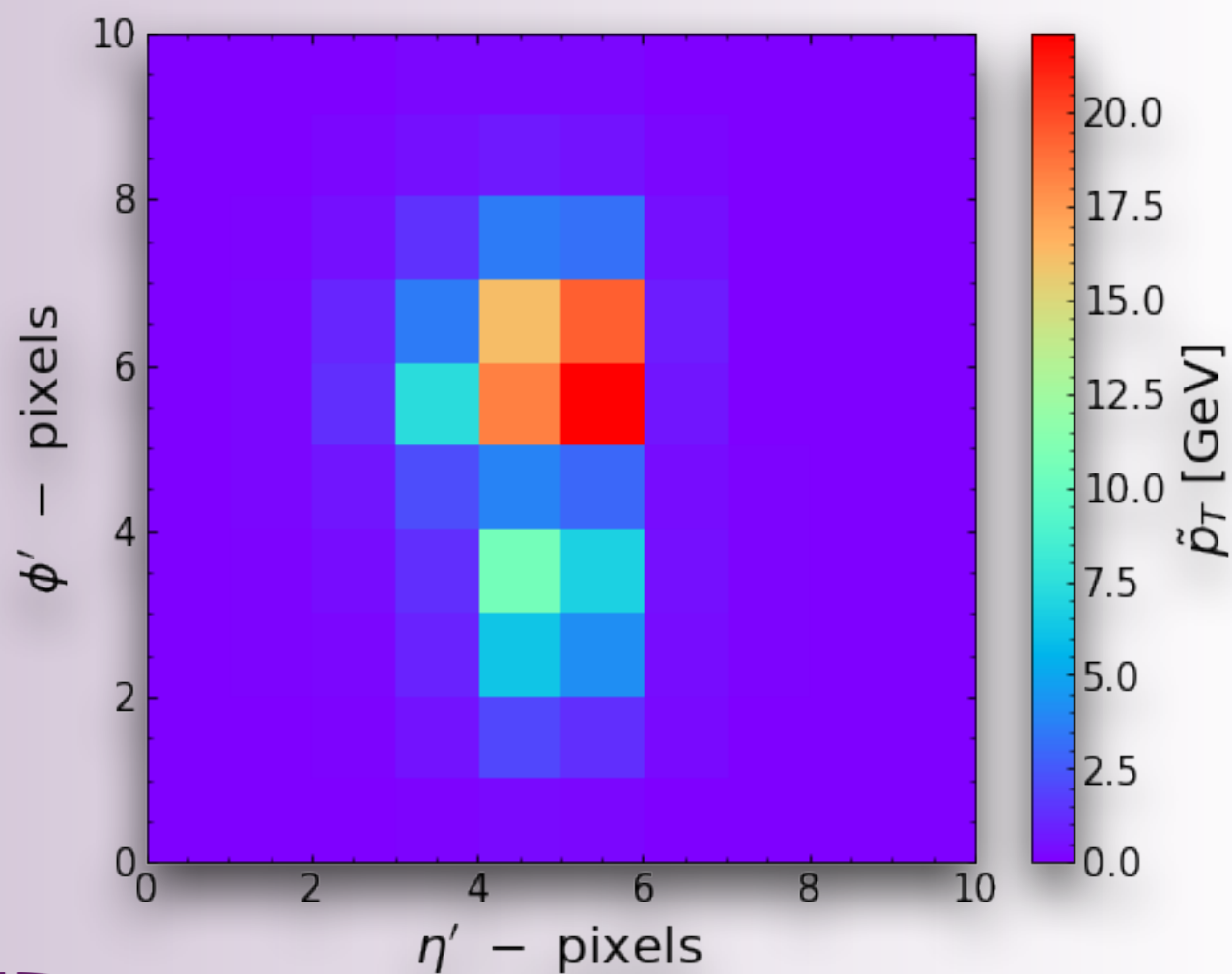
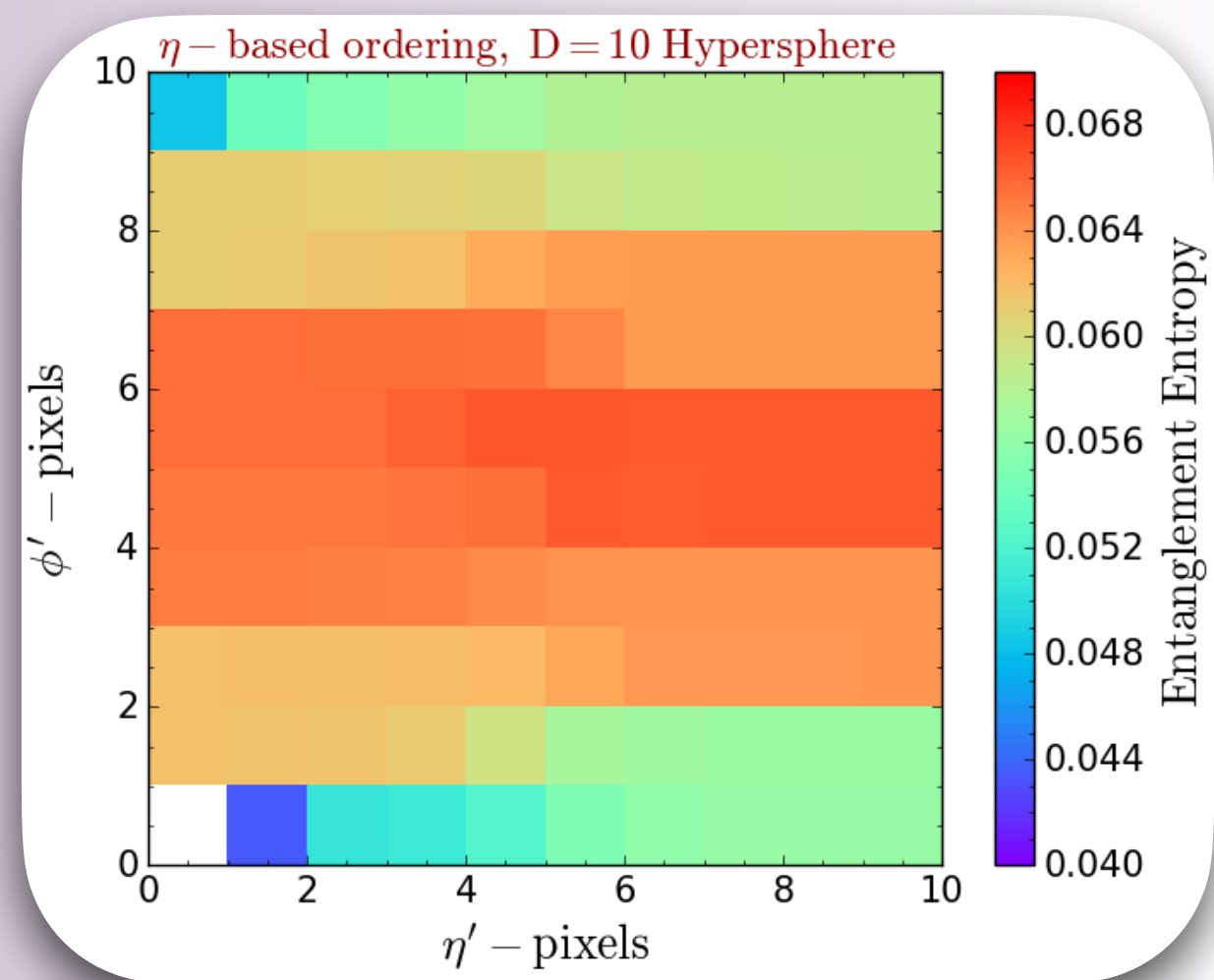
Reduced density matrix

Entanglement Entropy

$$\mathcal{S}(\rho) = -\text{Tr}[\rho \log \rho] \quad ; \quad \rho := |\phi\rangle\langle\phi|$$



Top Tagging through MPS



100 pixels, 390500 trainable parameters

54 pixels, 43410 trainable parameters

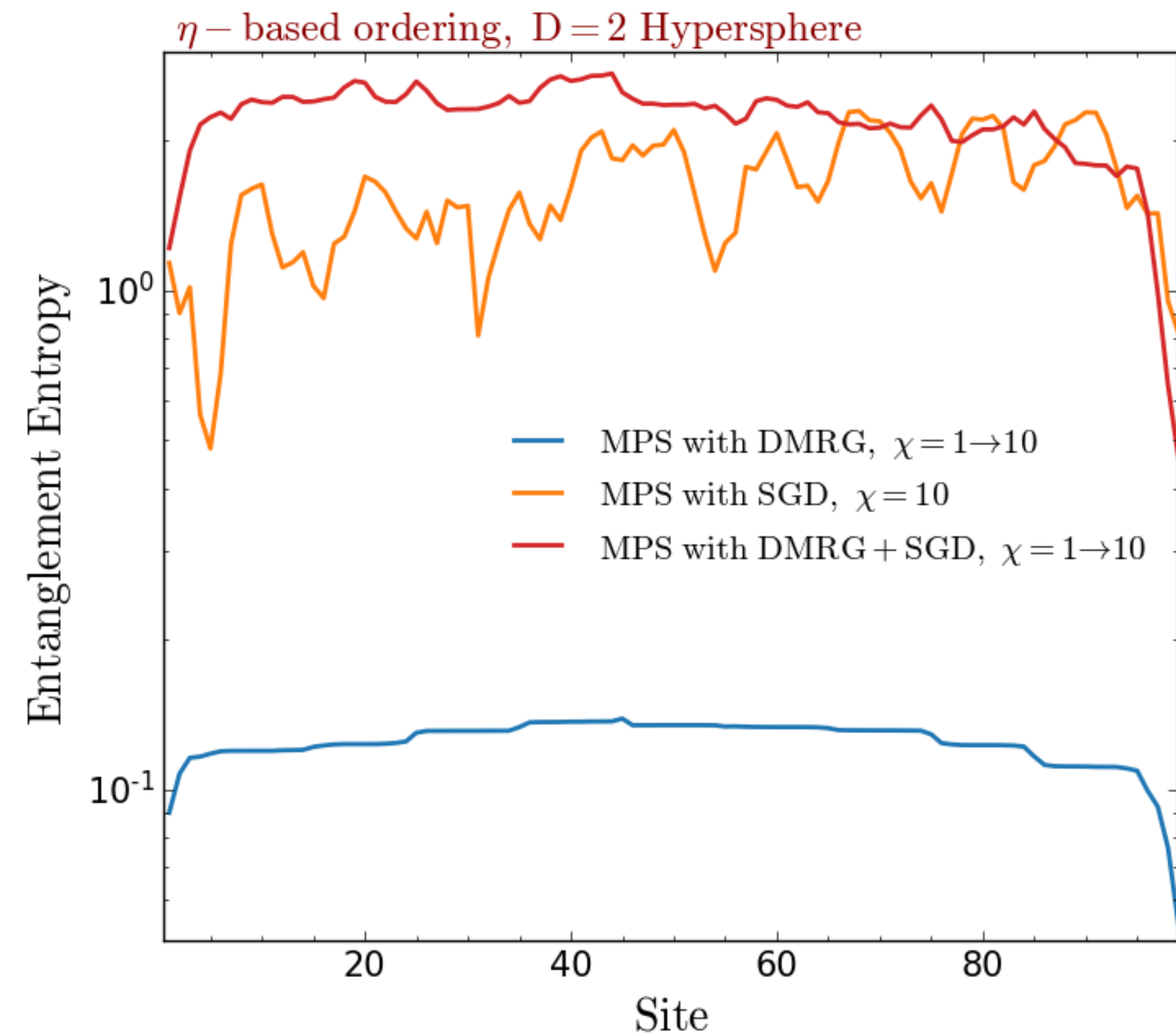
54 pixels, 34160 trainable parameters

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)!



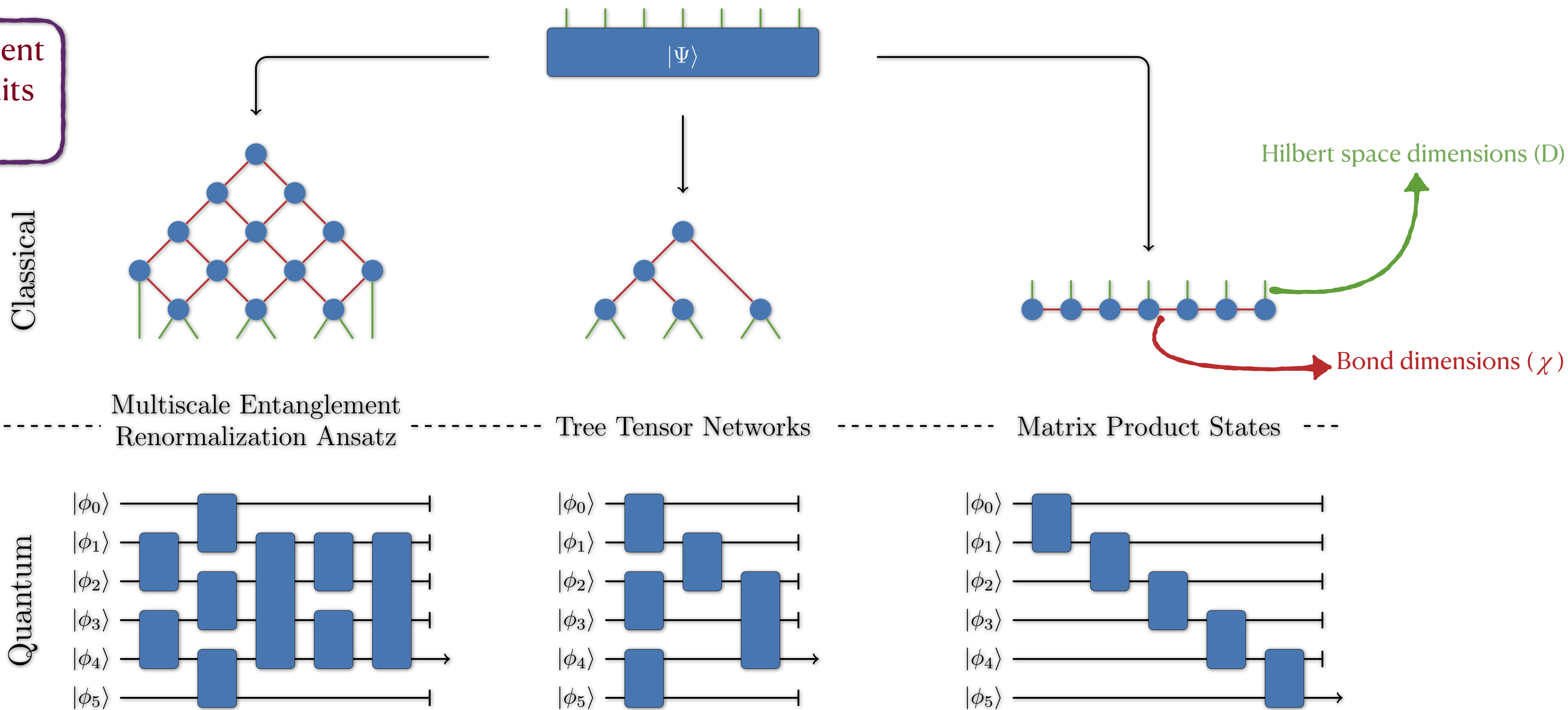


Quantum Machine Learning

Yes, up to now, it was “classical”...

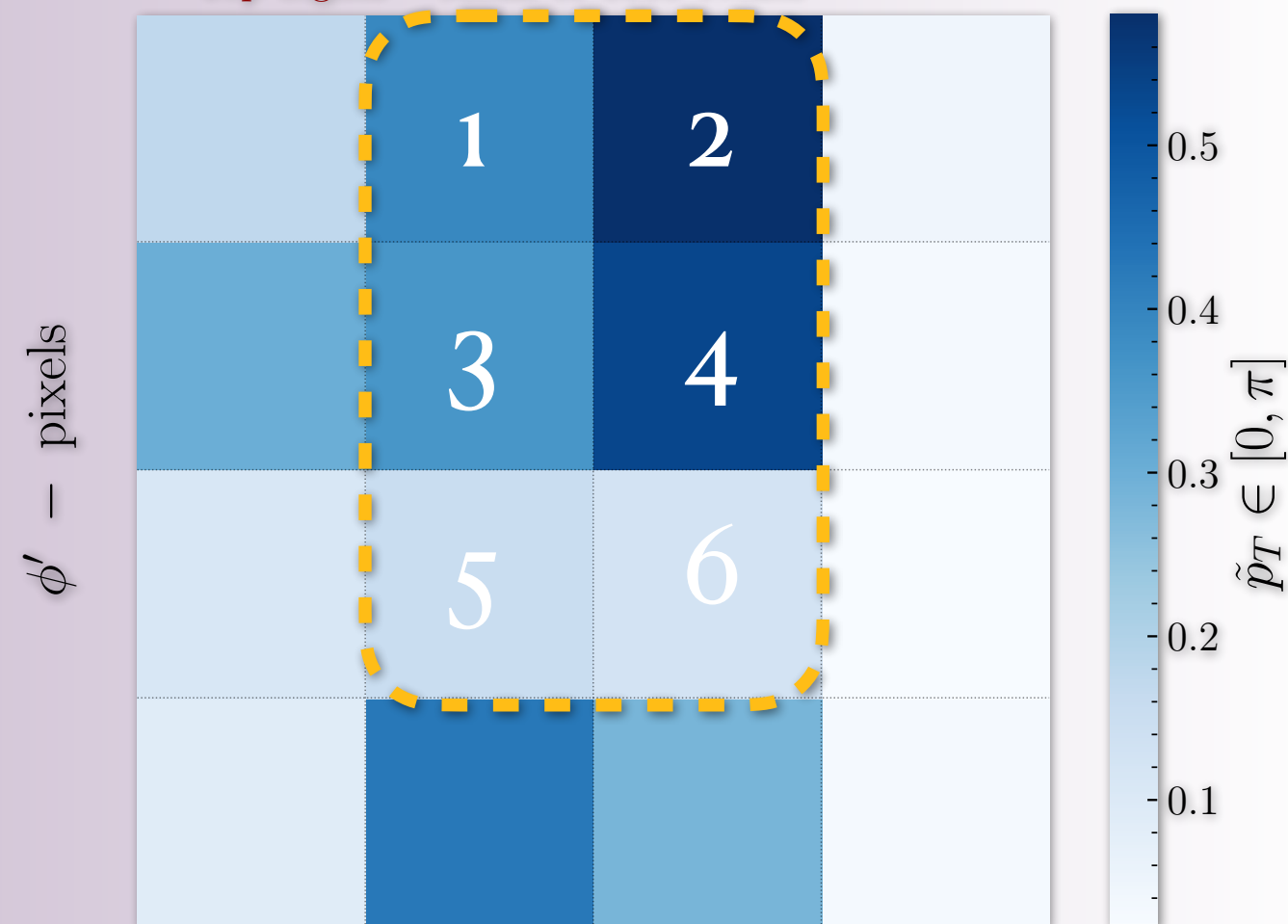
Quantum Tensor Networks

TNs can represent quantum circuits classically

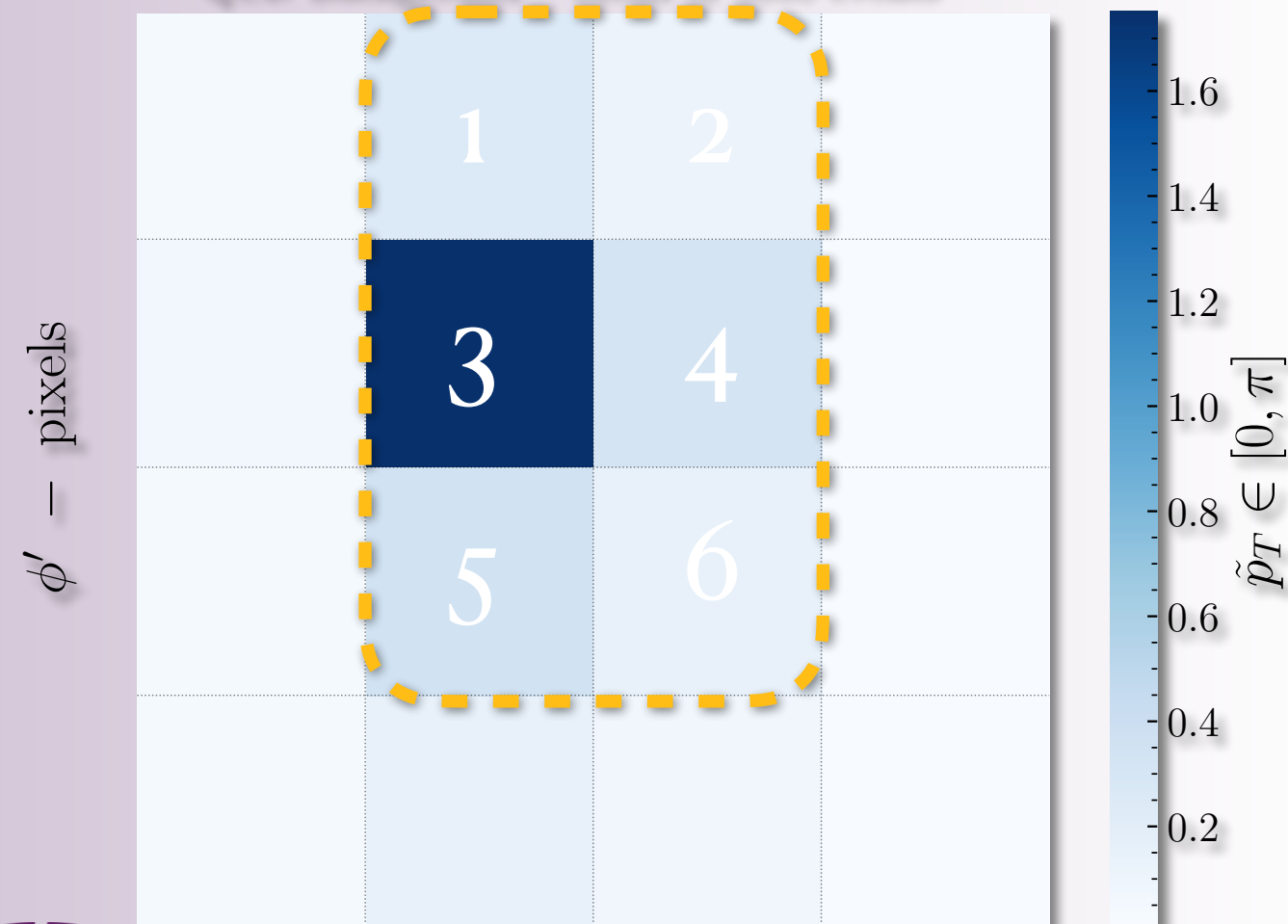


Experimenting with 6-Qubits

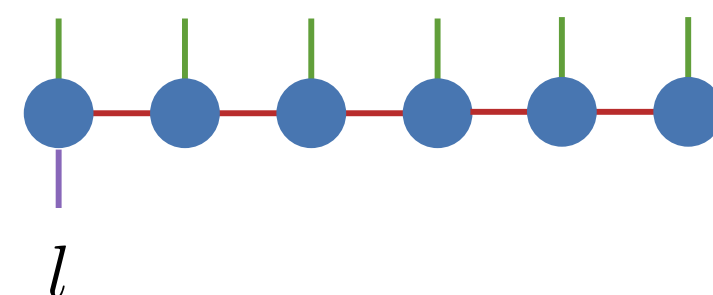
Top Signal – Mean of 5000 events



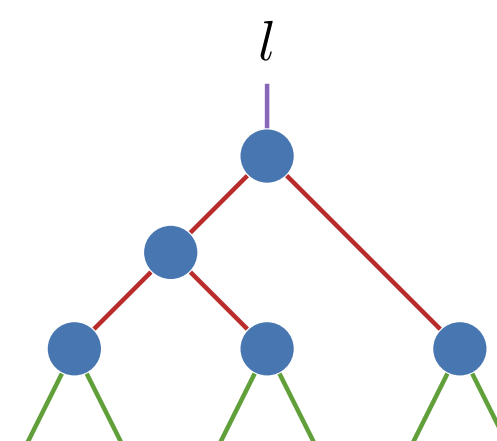
QCD Background – Mean of 5000 events



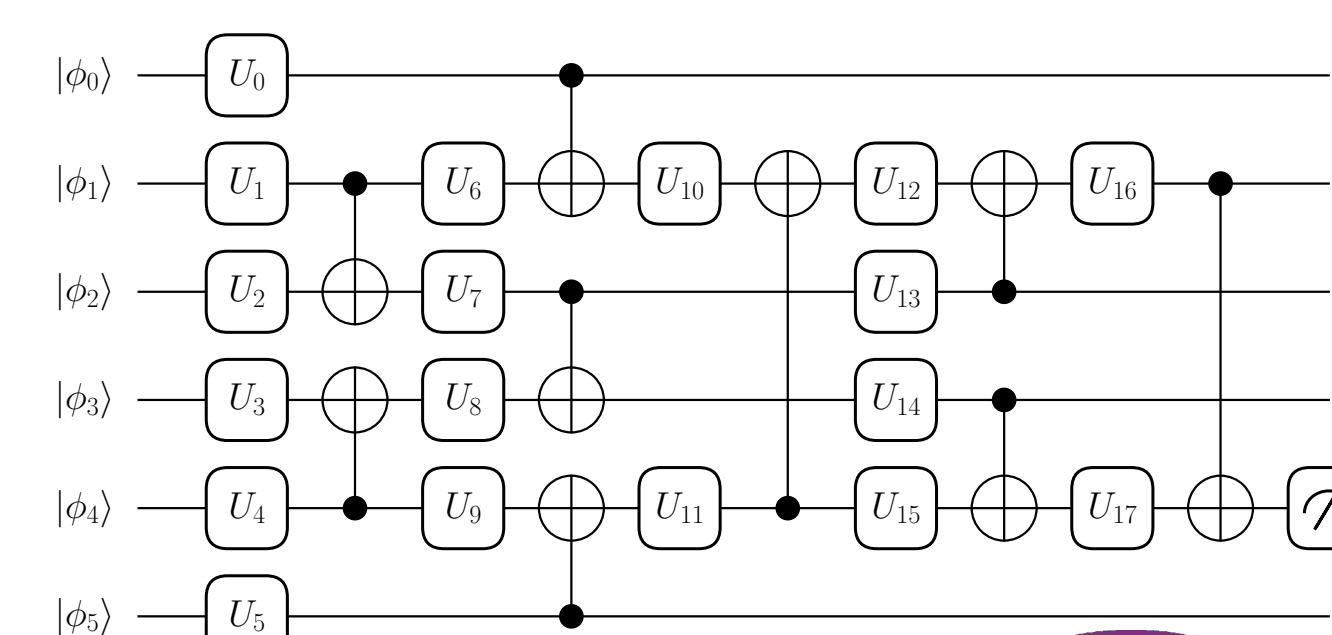
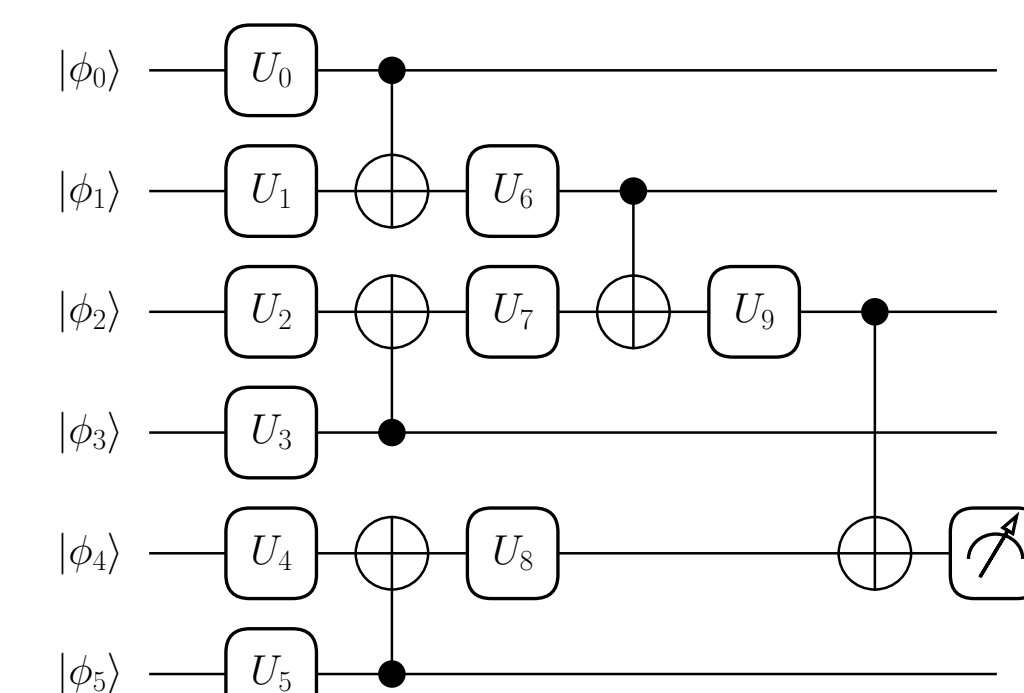
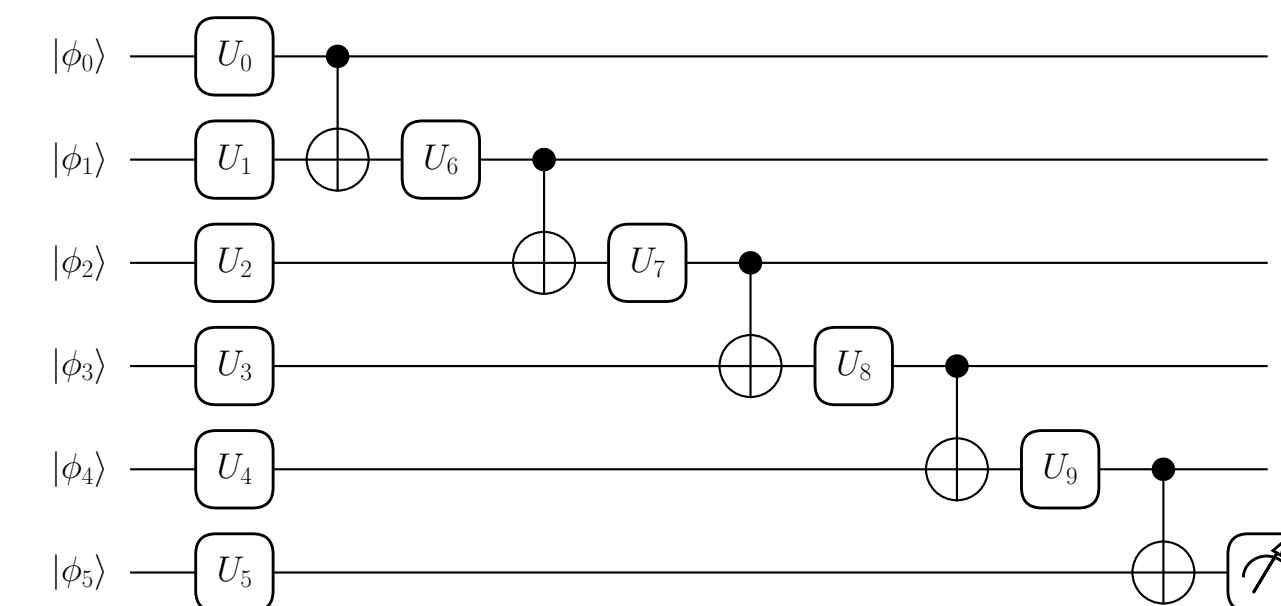
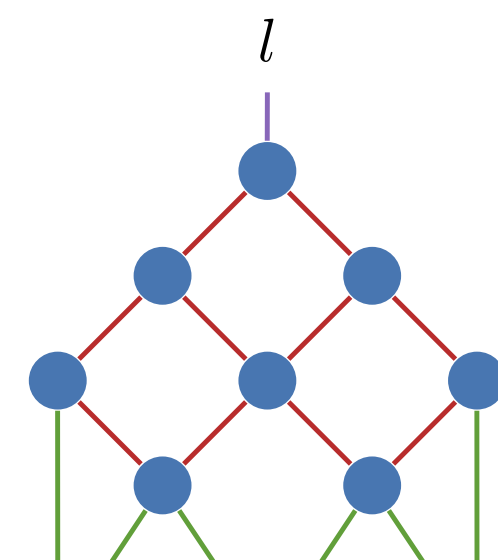
Matrix Product States



Tree Tensor Networks

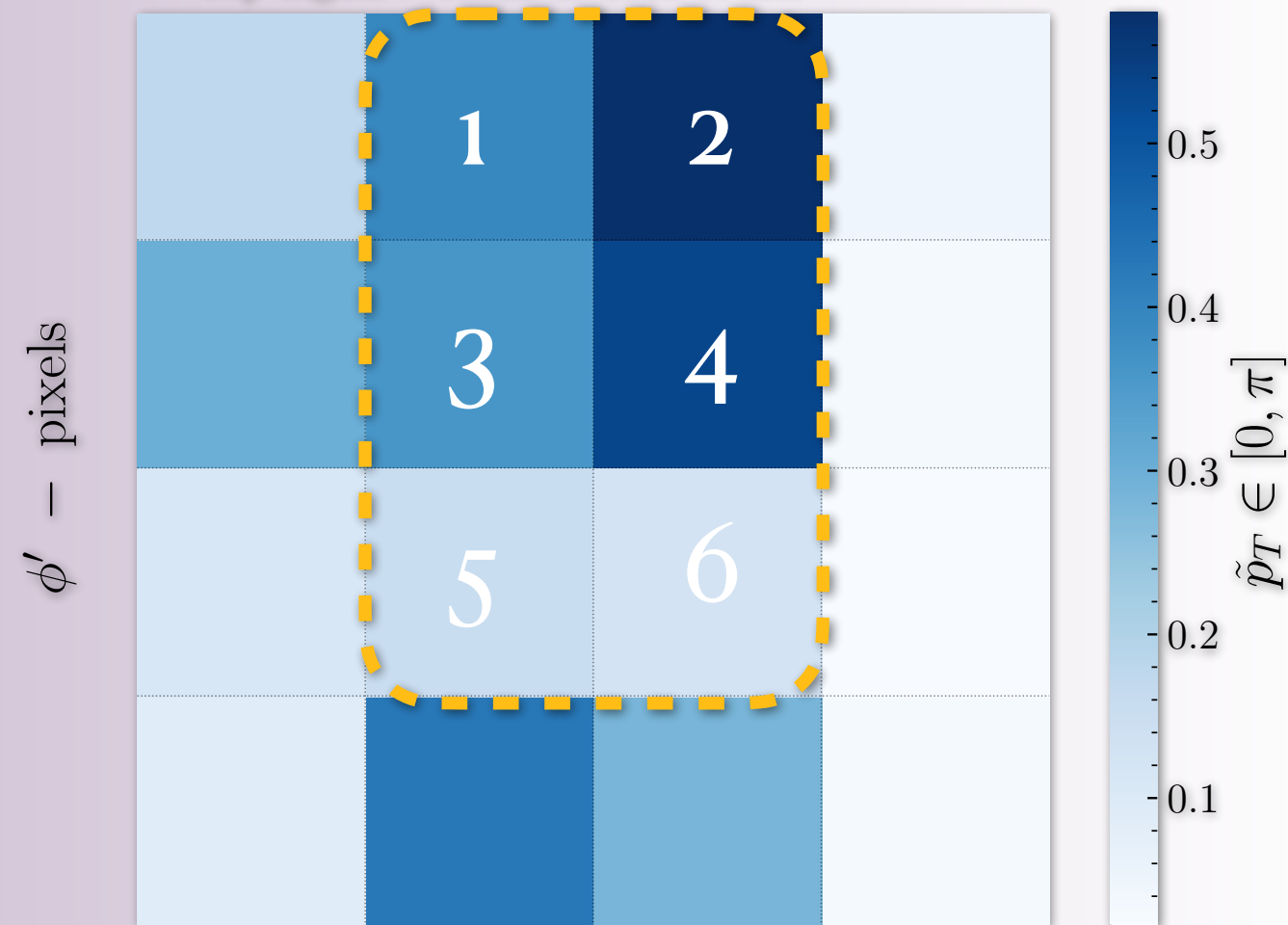


Multiscale Entanglement Renormalisation Ansatz



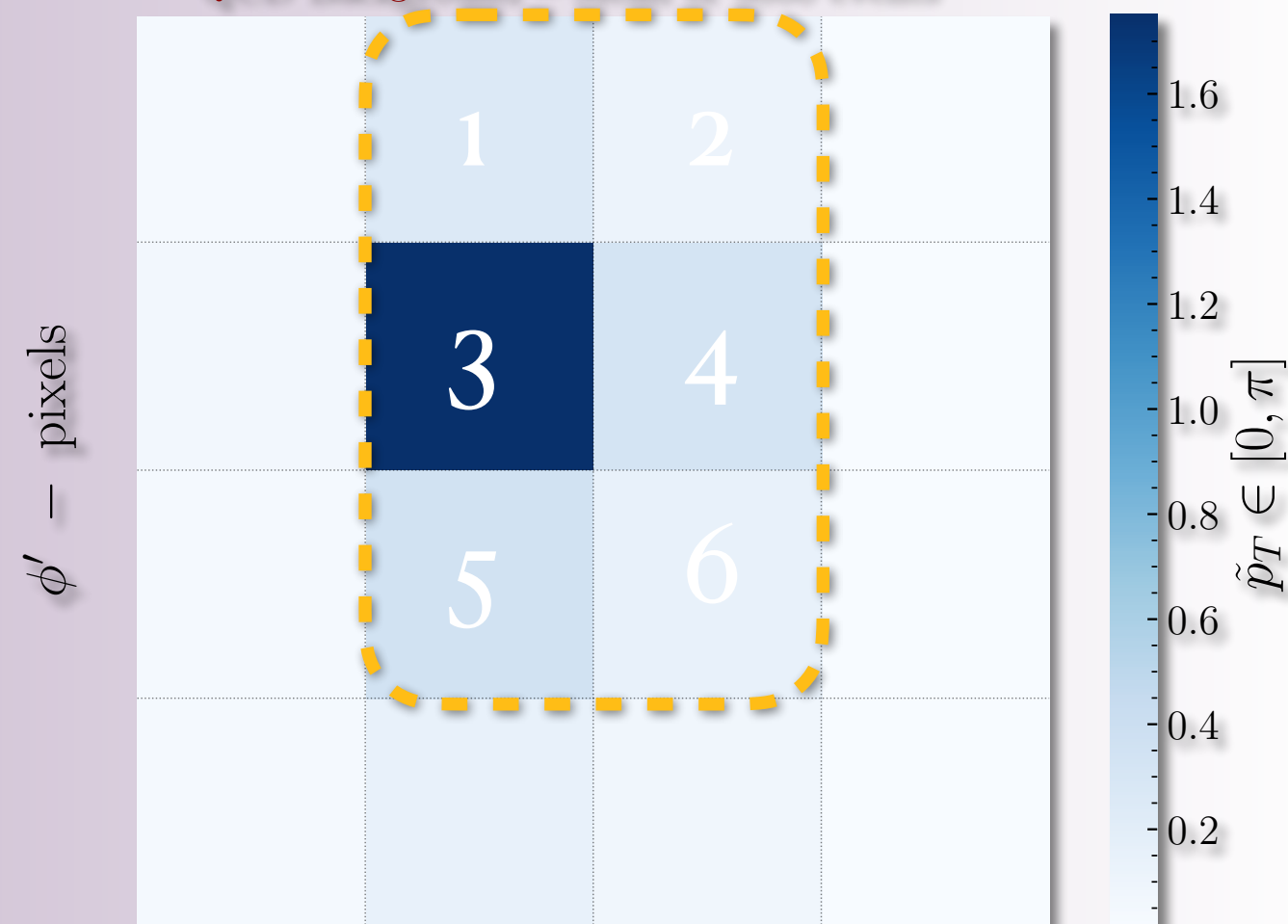
Experimenting with 6-Qubits

Top Signal – Mean of 5000 events



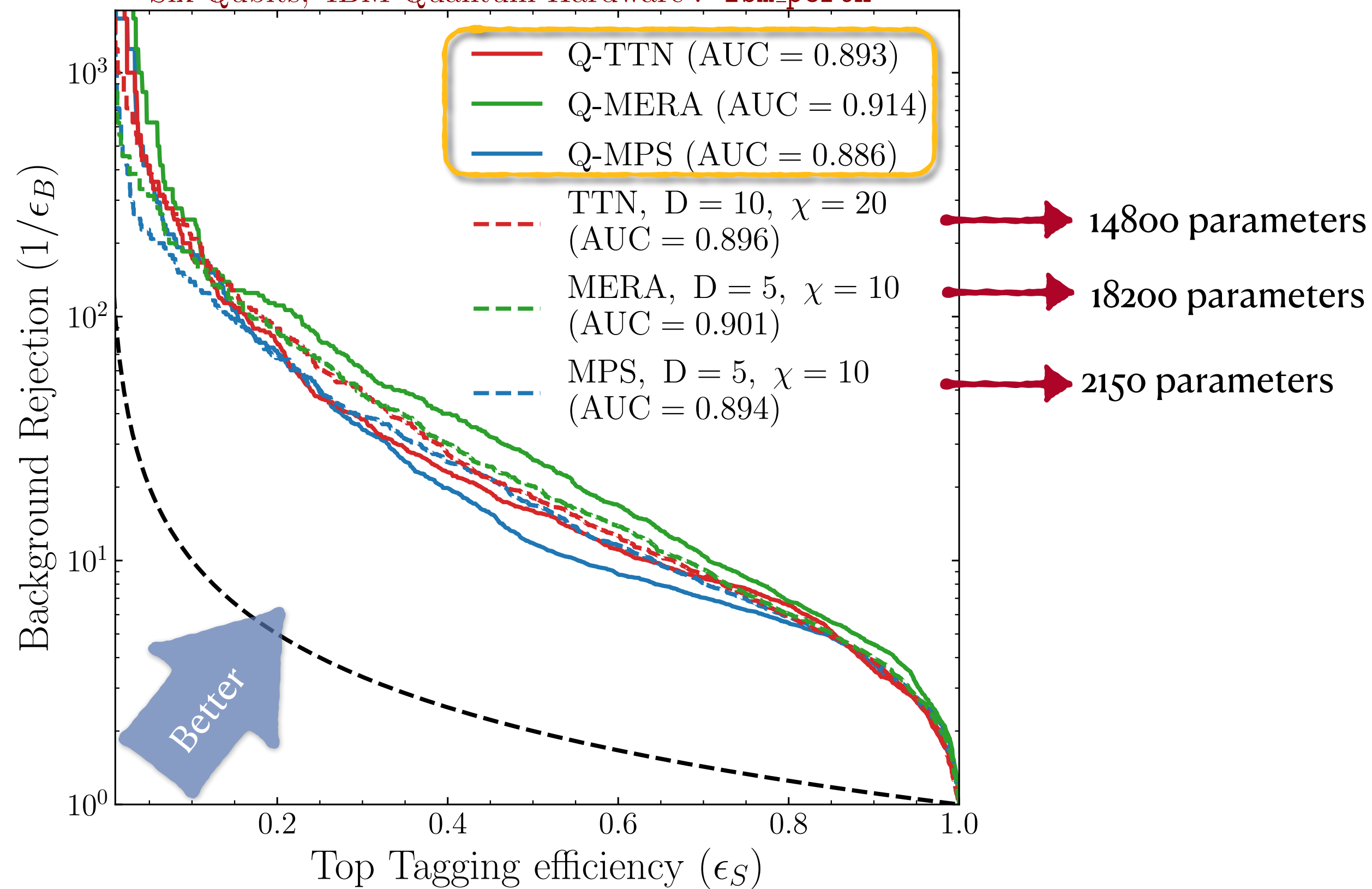
η' – pixels

QCD Background – Mean of 5000 events



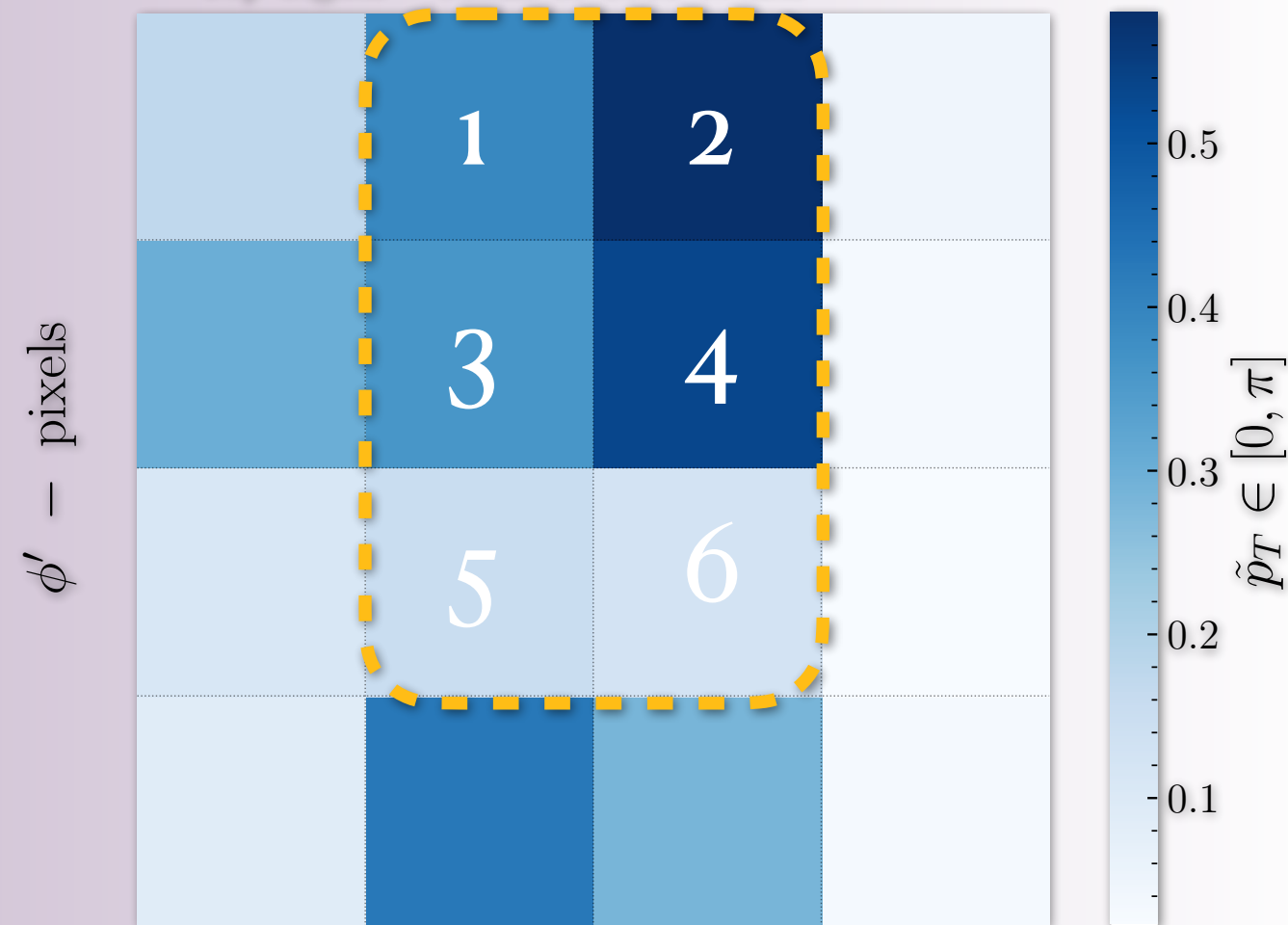
η' – pixels

Six Qubits, IBM Quantum Hardware : `ibm_perth`



Experimenting with 6-Qubits

Top Signal – Mean of 5000 events



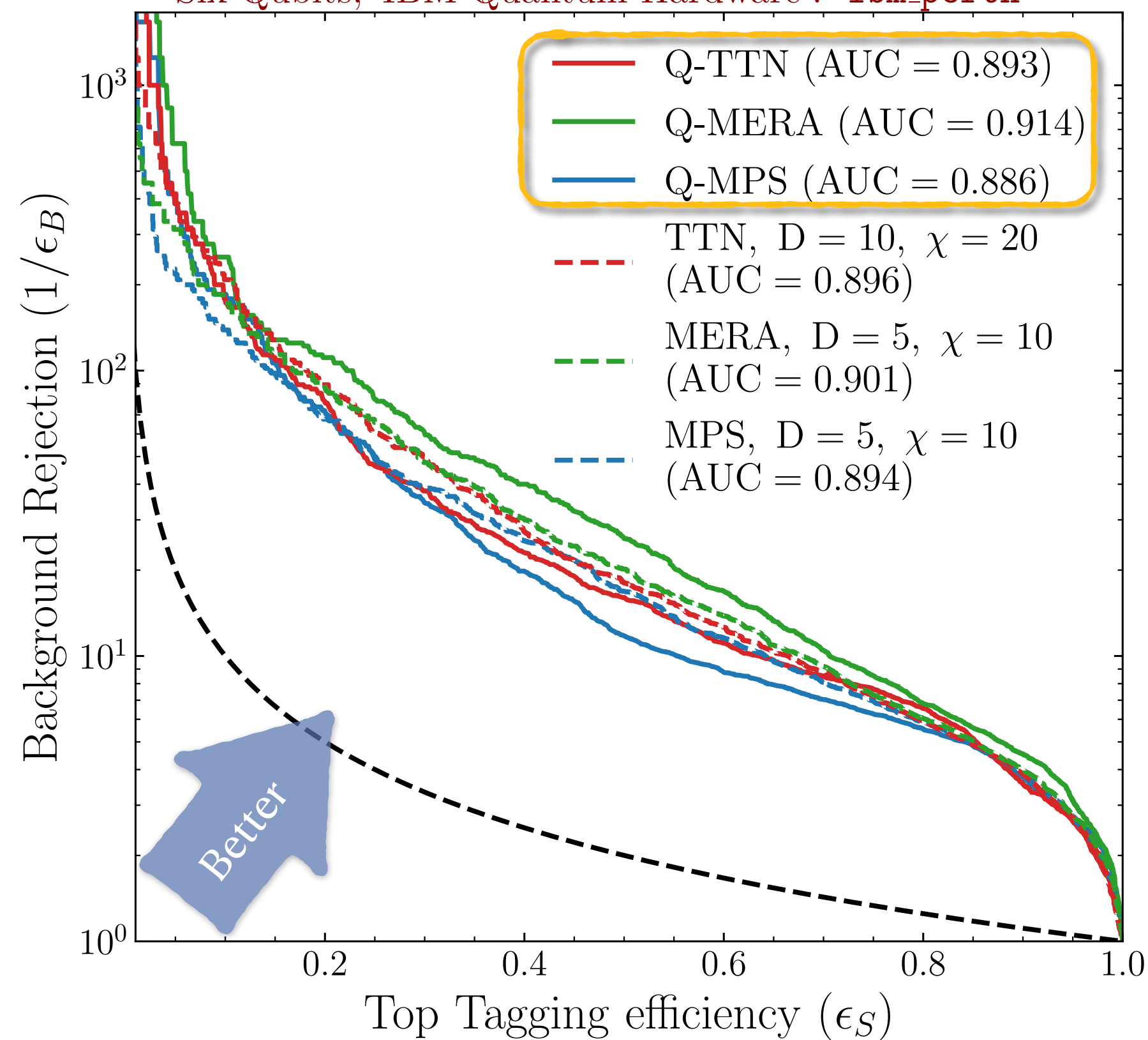
η' – pixels

QCD Background – Mean of 5000 events



η' – pixels

Six Qubits, IBM Quantum Hardware : *ibm_perth*



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
	5	10	18200	0.901
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 optimisation** compared to classical networks.
- **BUT** near term quantum devices are still very much limited to a few qubits.

BACKUP

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. $\dots p_n(\mathbf{x})$
- ❖ The cost for **TNs scales at most linearly** in the number of training examples, where quadratically for most kernel methods.
- ❖ 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.

Novikov, Trofimov, Oseledets; arXiv: 1605.03795

Lu, May, et. al. arXiv: 1411.4000

Stoudenmire, Schwab; arXiv: 1605.05775

Why finding a quantitative measure is important?

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

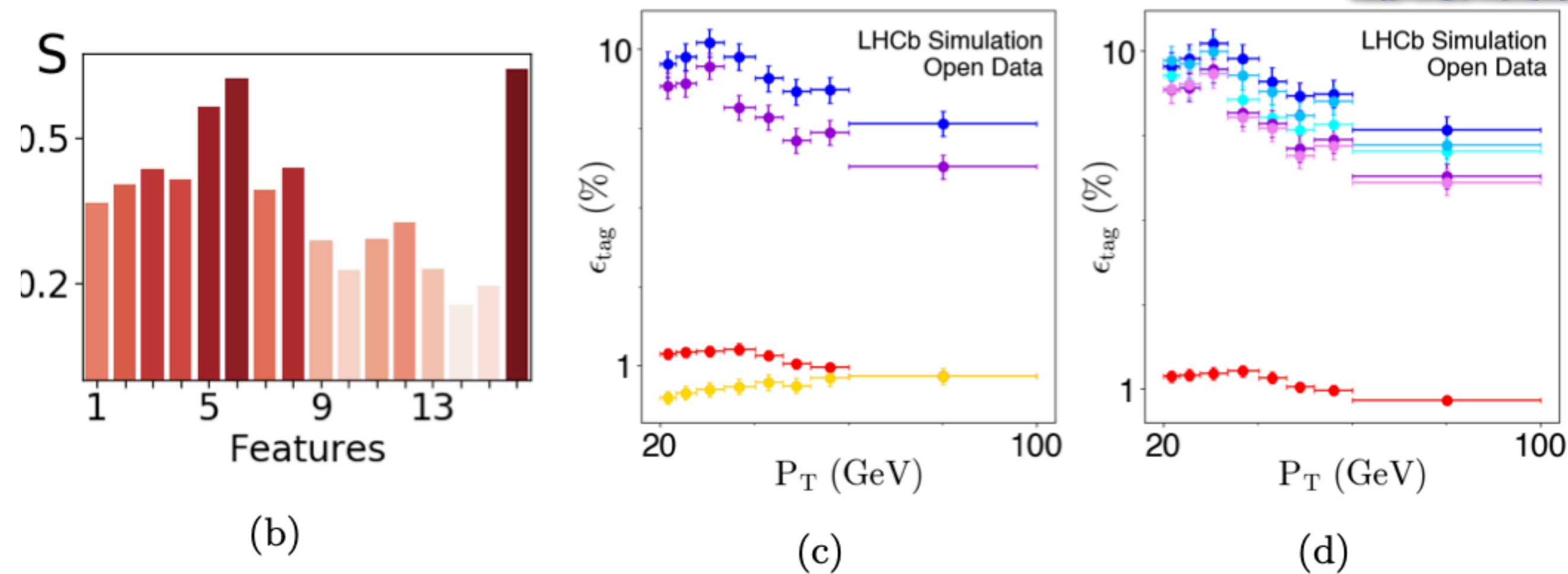
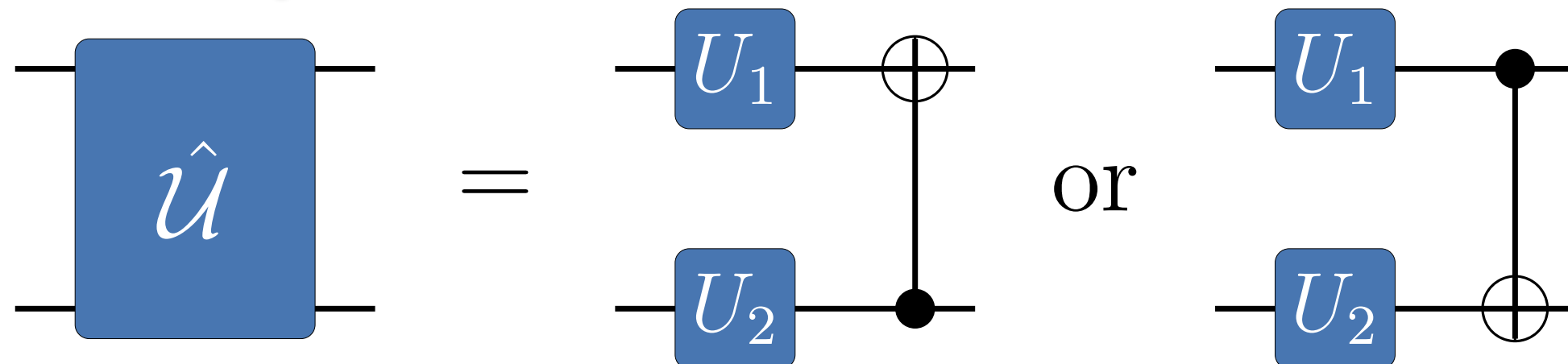
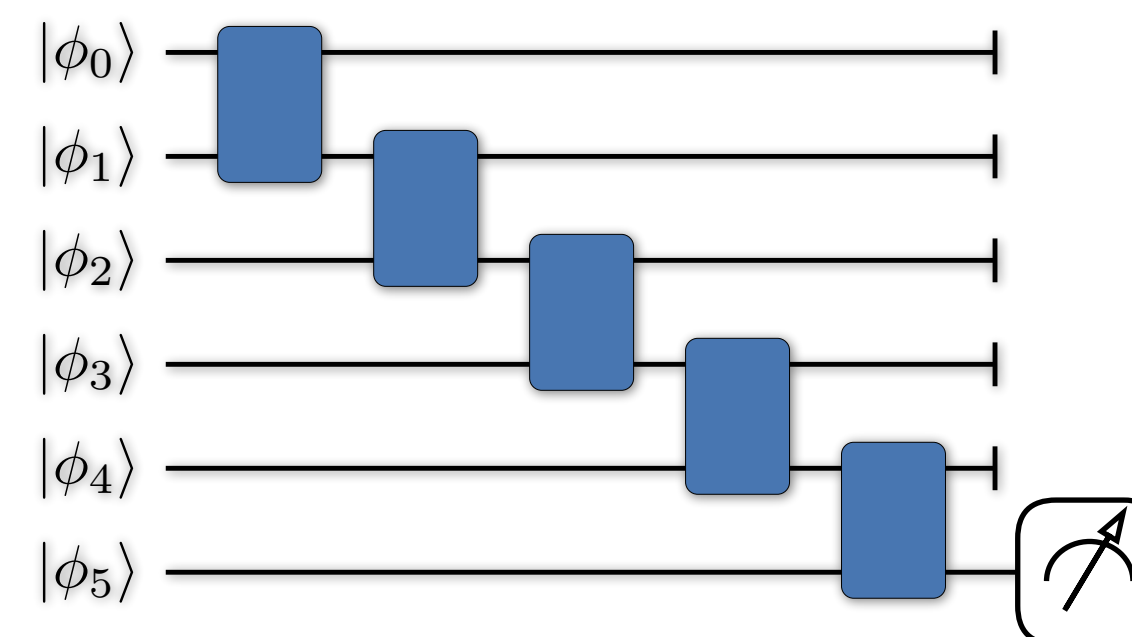
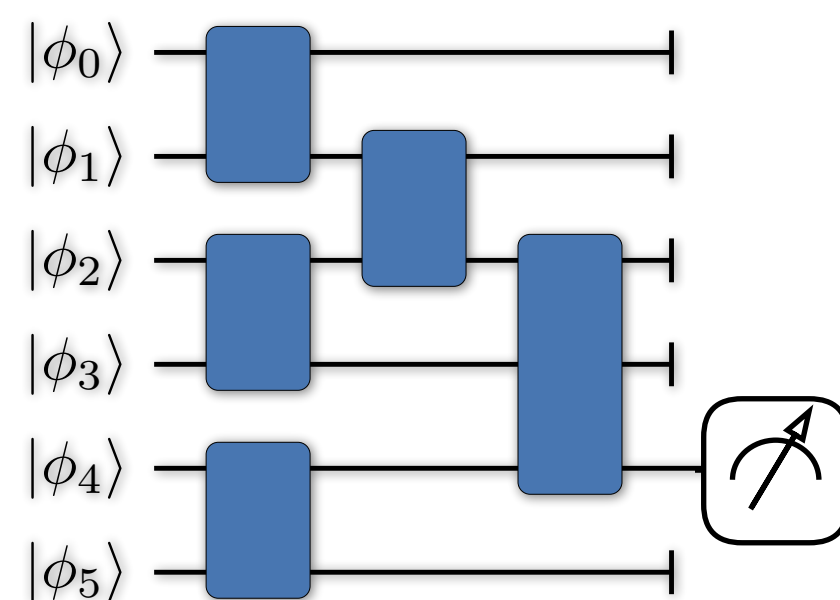
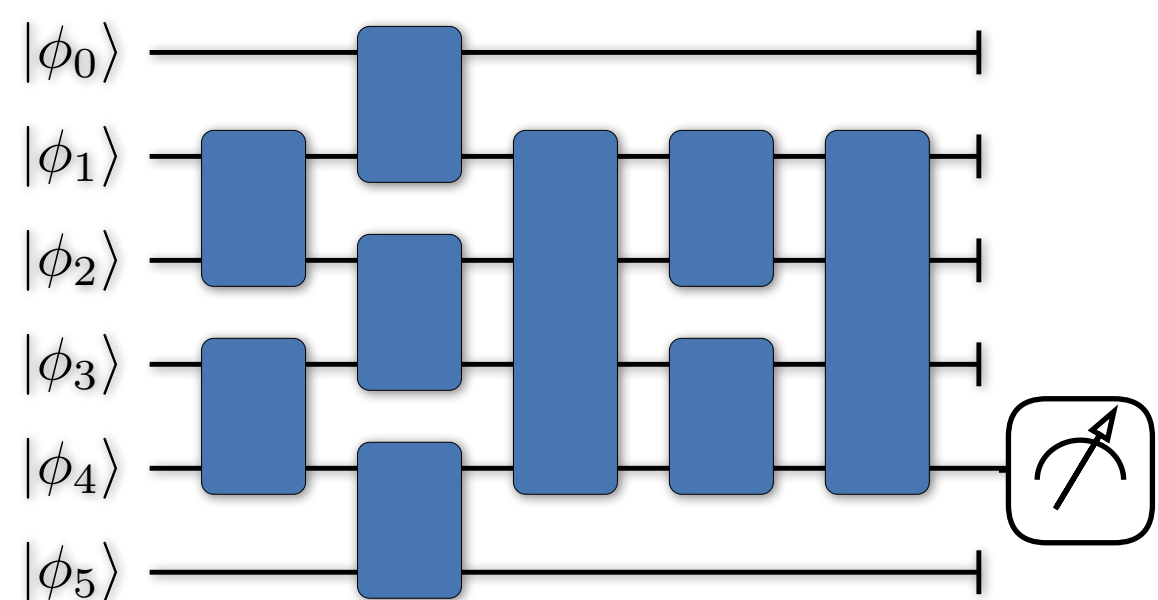


FIG. 3: Exploiting the information provided by the learned **TTN classifier**: (a) Correlations between the 16 input features (blue for anti-correlated, white for uncorrelated, red for correlated). The numbers indicate q , p_T^{rel} , ΔR of the muon (1-3), kaon (4-6), pion (7-9), electron (10-12), proton (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).

- Under the hood of the algorithm
- Scientific knowledge from the amount of data
- Support mitigation

Quantum Tensor Networks

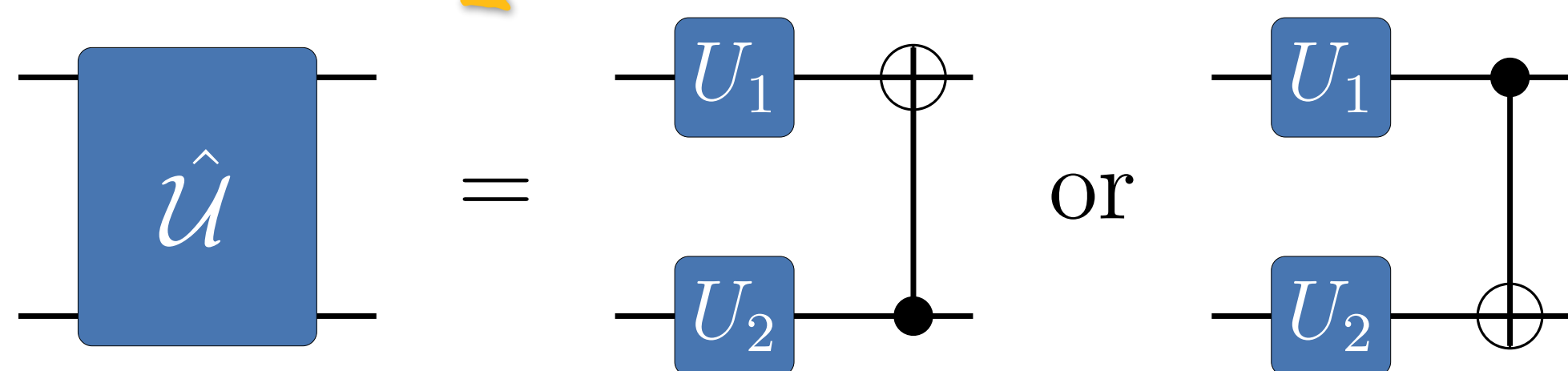
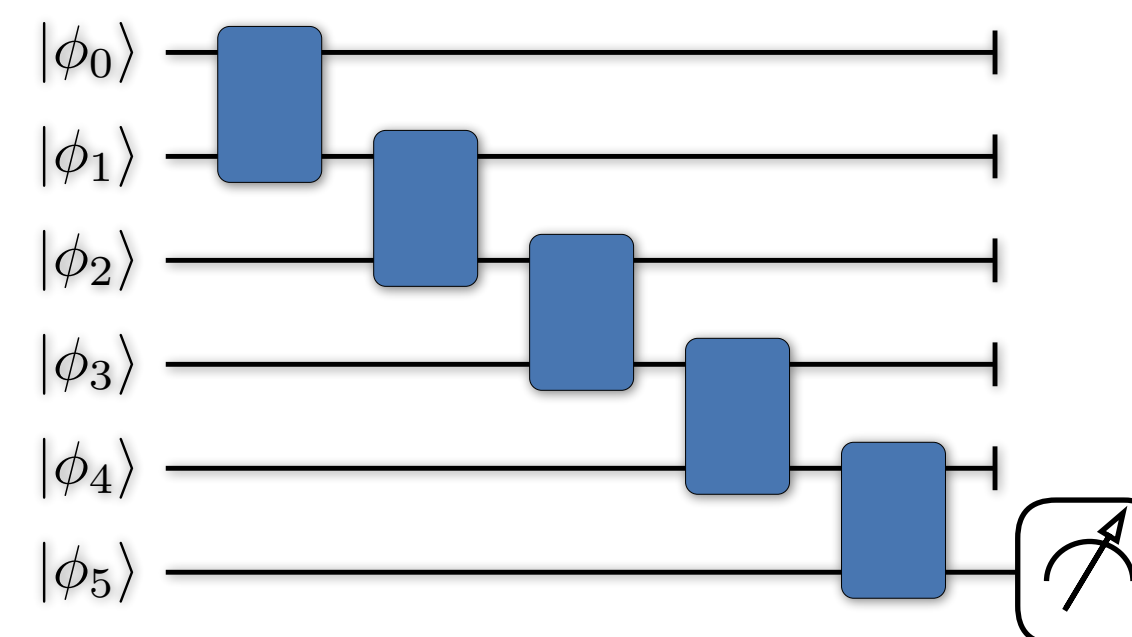
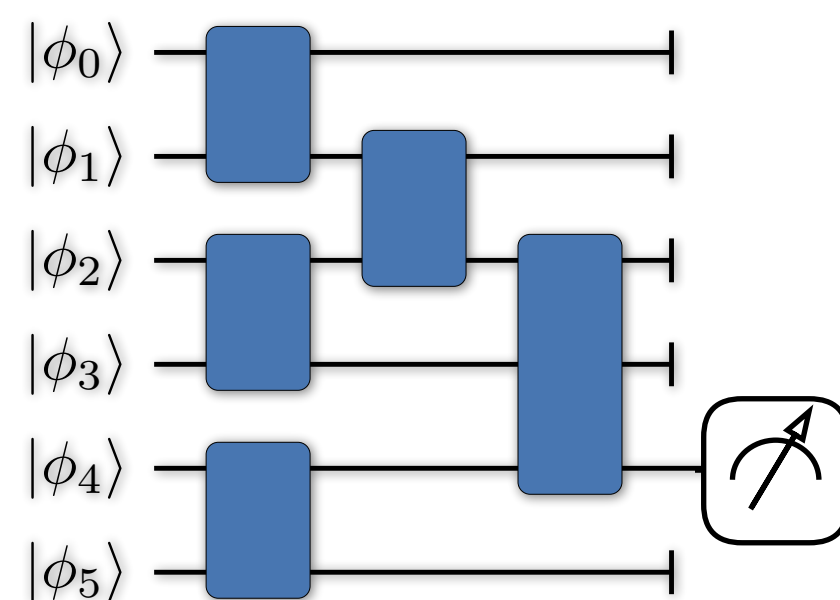
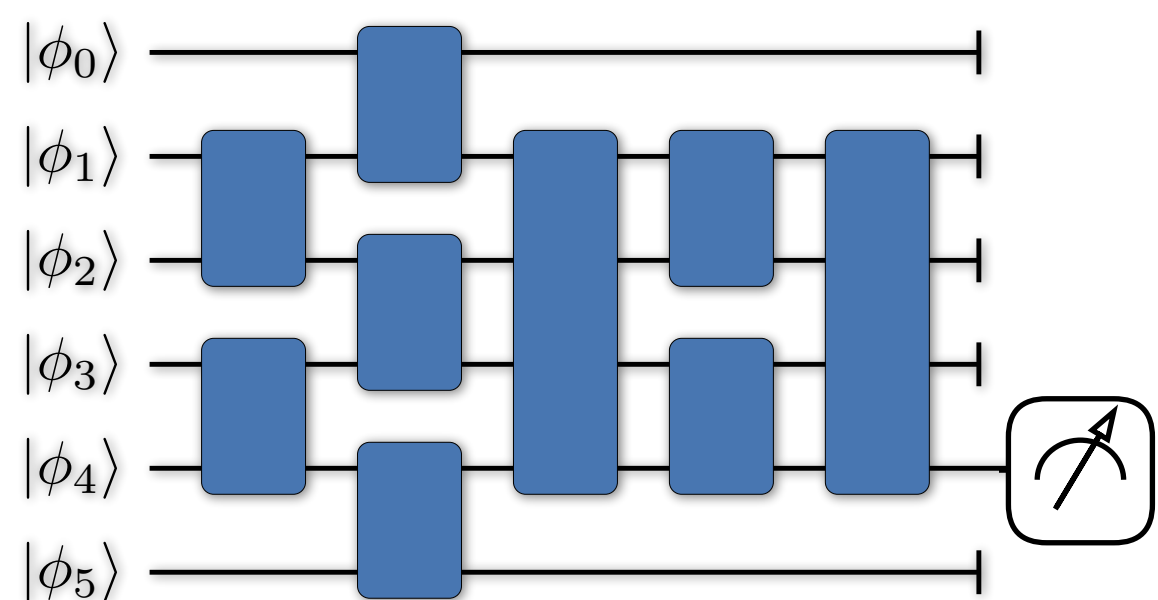


$$U(\theta, \varphi, \lambda) = \begin{pmatrix} \cos(\theta/2) & -e^{i\lambda} \sin(\theta/2) \\ e^{i\varphi} \sin(\theta/2) & e^{i(\lambda+\varphi)} \cos(\theta/2) \end{pmatrix}$$

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

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

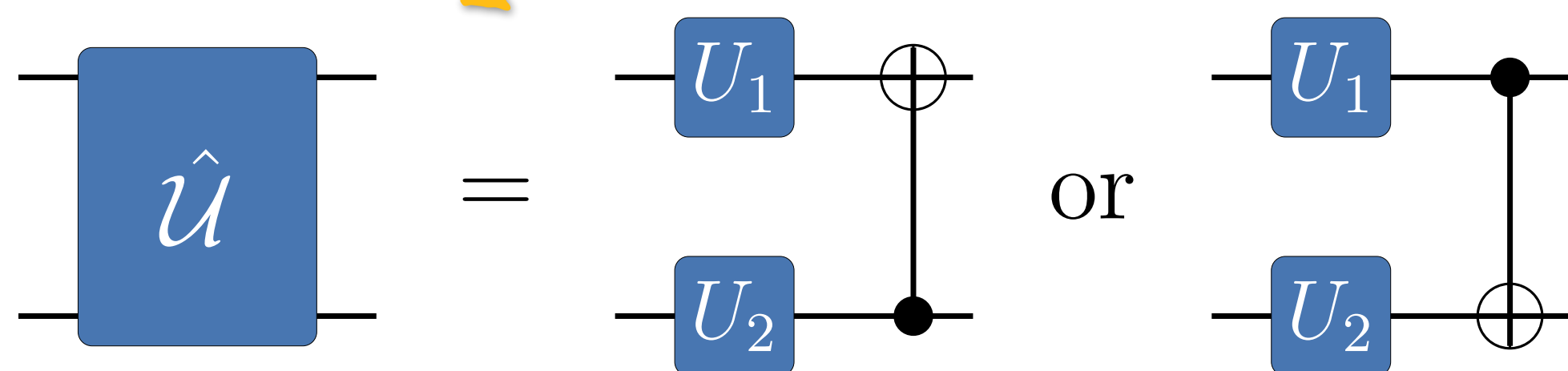
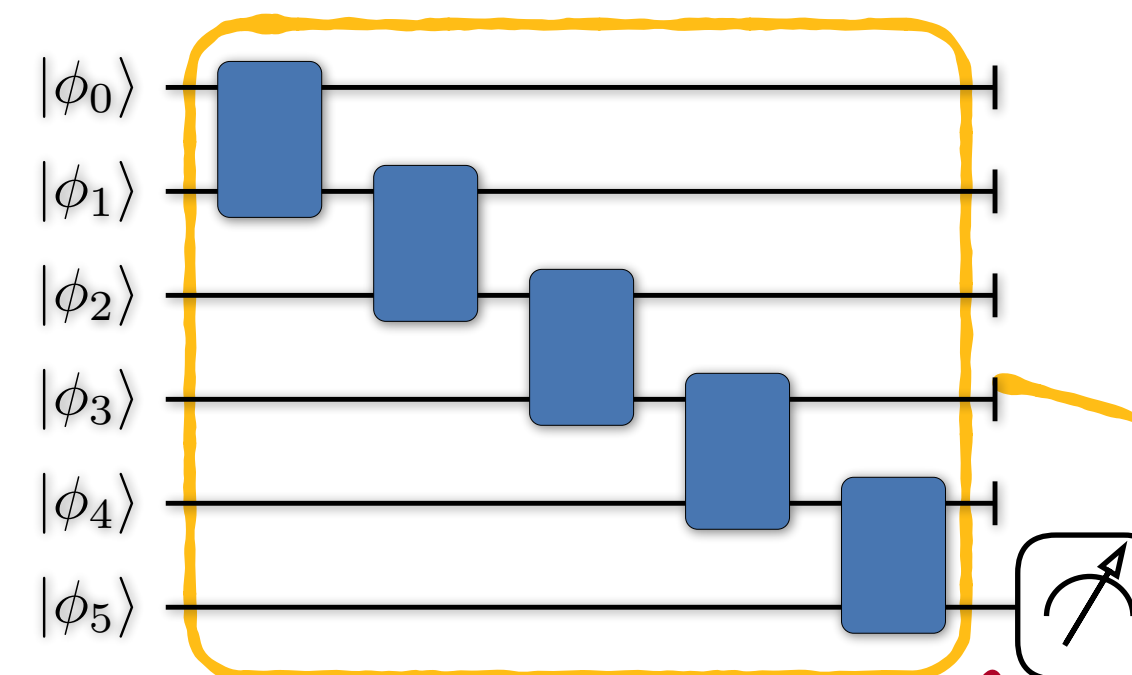
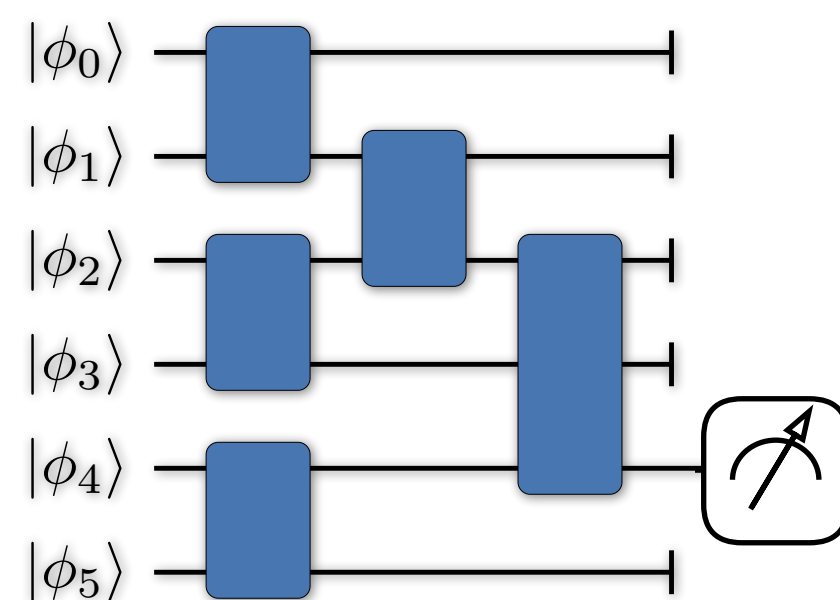
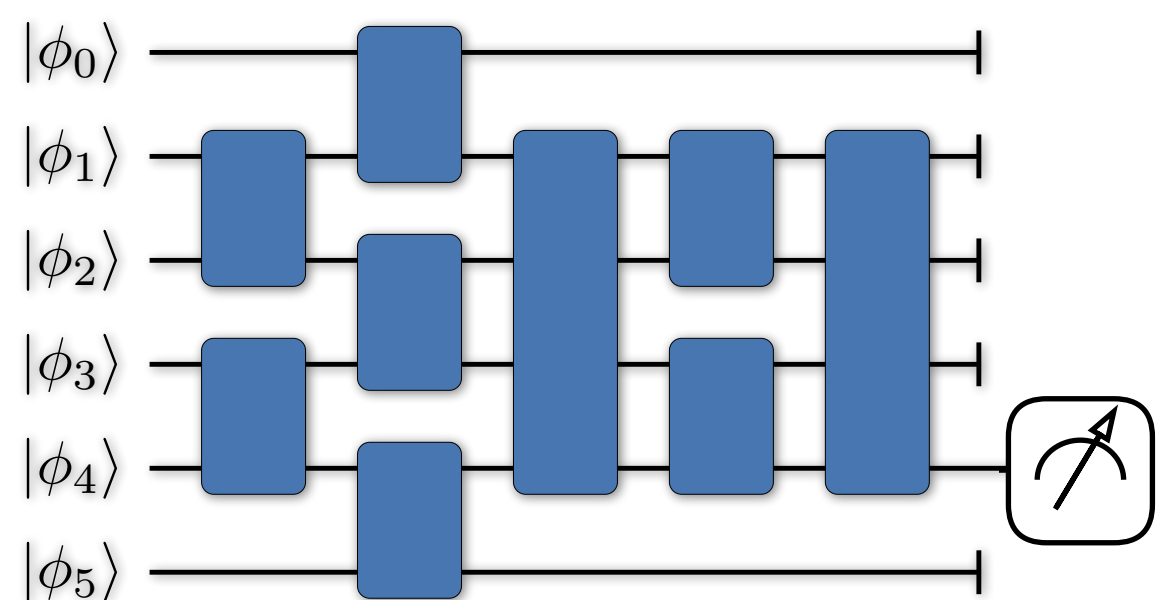
Quantum Tensor Networks



$$M_{\theta}(\Phi^{\beta_1 \dots \beta_n}(\mathbf{x})) = \langle \Phi | \hat{\mathcal{U}}_{\text{QC}}^{\dagger}(U_i(\theta_j)) \hat{M} \hat{\mathcal{U}}_{\text{QC}}(U_i(\theta_j)) | \Phi \rangle$$

$$p(\mathbf{x}^{(i)}; \theta) = \left| M_{\theta}(\Phi^{\beta_1 \dots \beta_n}(\mathbf{x}^{(i)})) \right|^2$$

Quantum Tensor Networks



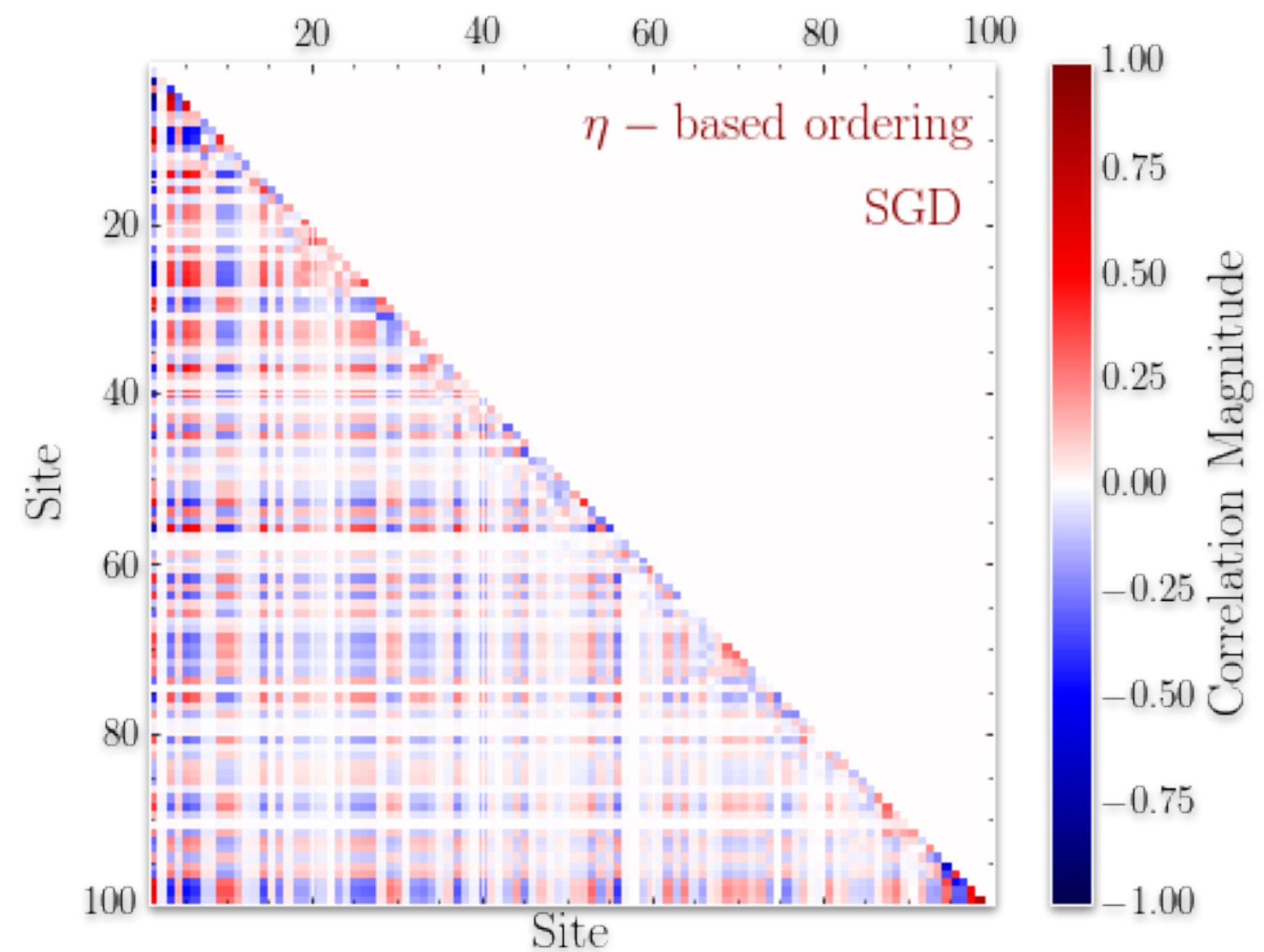
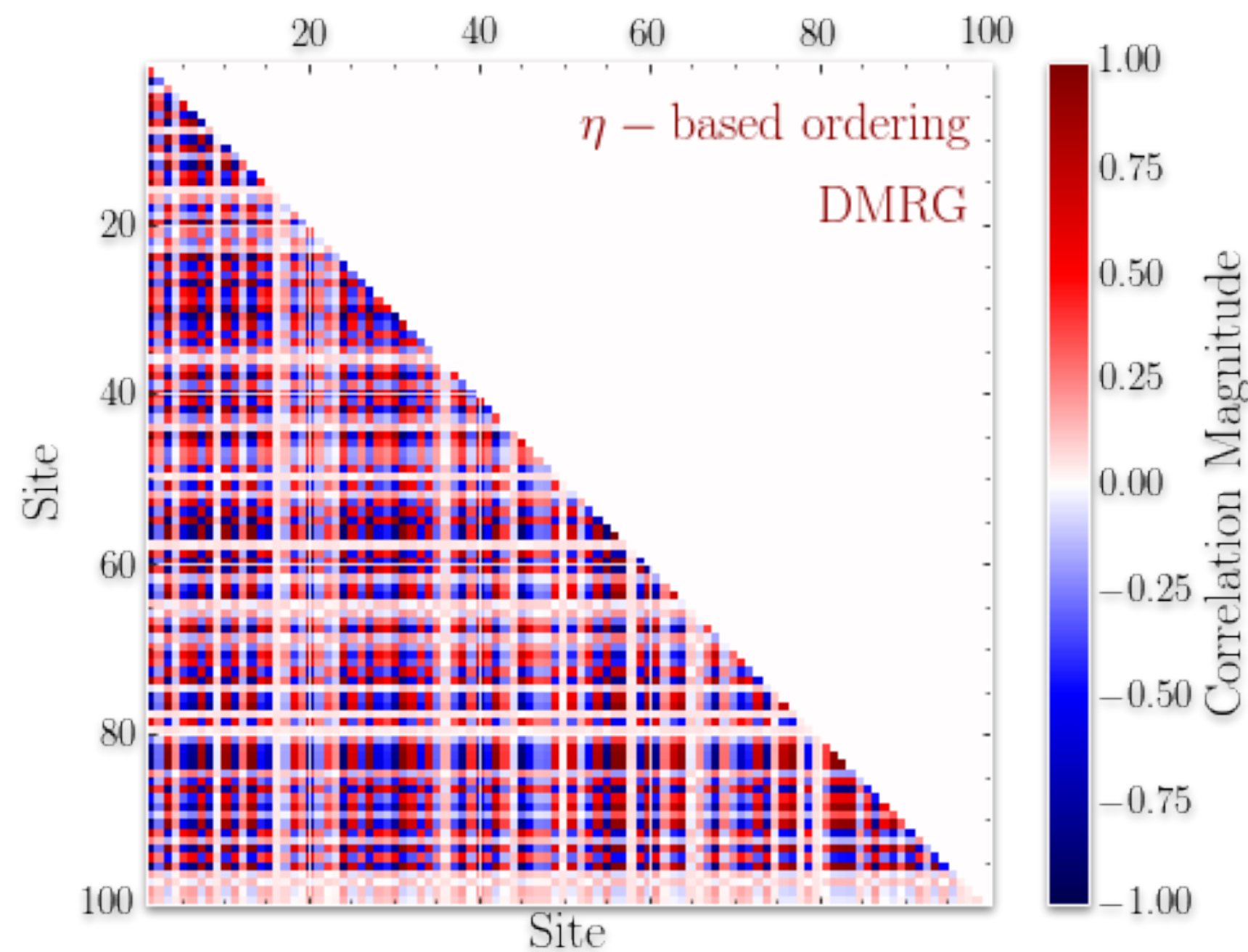
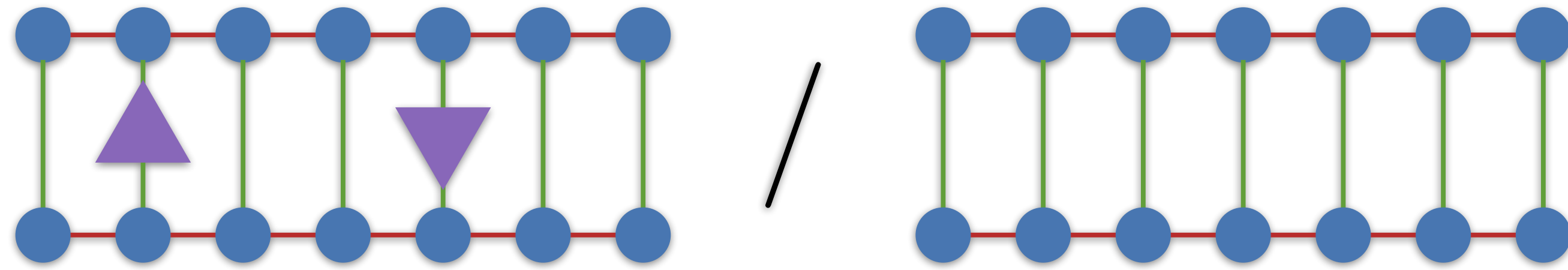
$$M_\theta(\Phi^{\beta_1 \dots \beta_n}(\mathbf{x})) = \langle \Phi | \hat{U}_{QC}^\dagger(U_i(\theta_j)) \hat{M} \hat{U}_{QC}(U_i(\theta_j)) | \Phi \rangle$$

$$p(\mathbf{x}^{(i)}; \theta) = \left| M_\theta(\Phi^{\beta_1 \dots \beta_n}(\mathbf{x}^{(i)})) \right|^2$$

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

Correlations by SU(2) generators

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

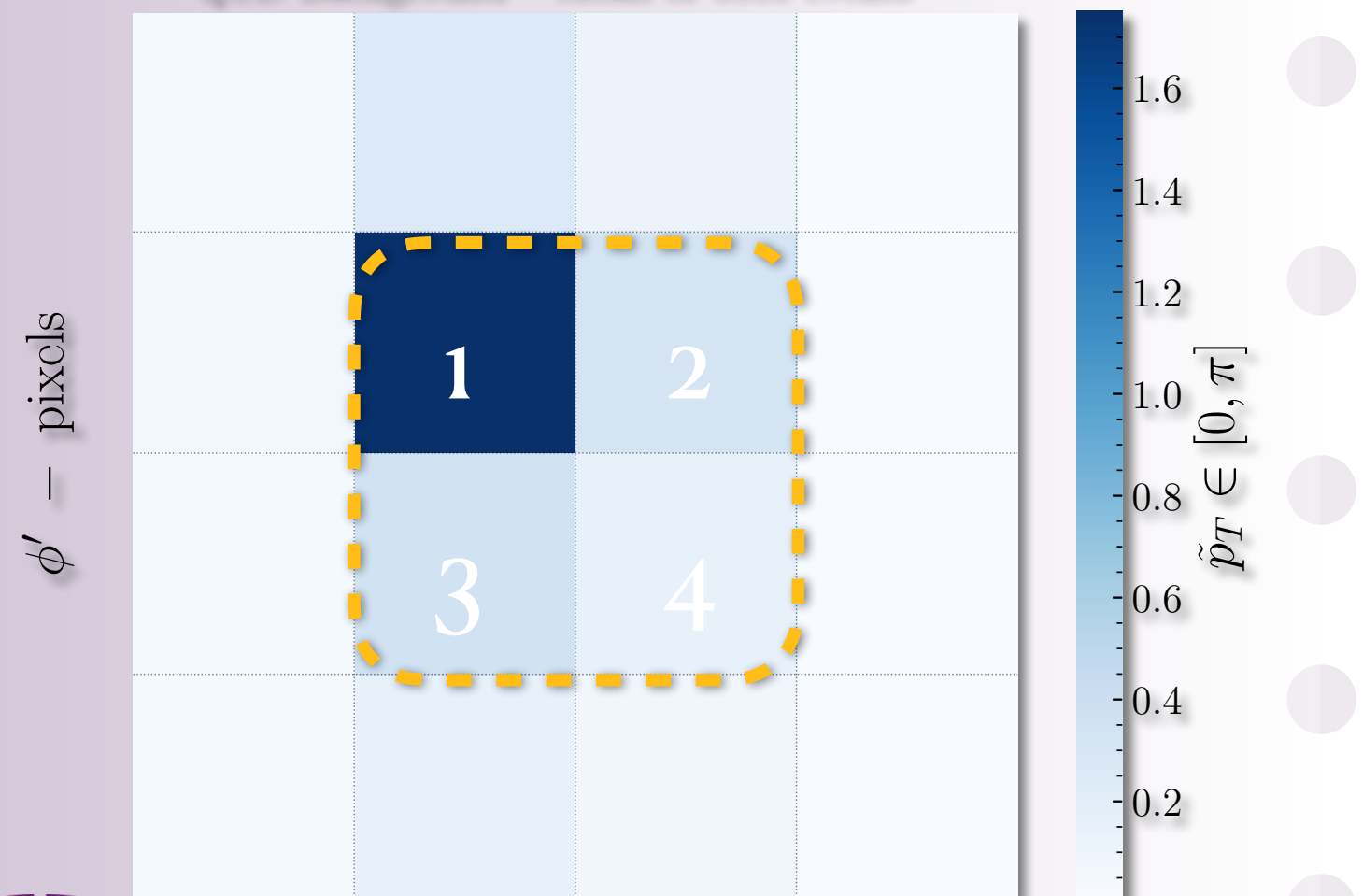


Experimenting with 4-Qubits

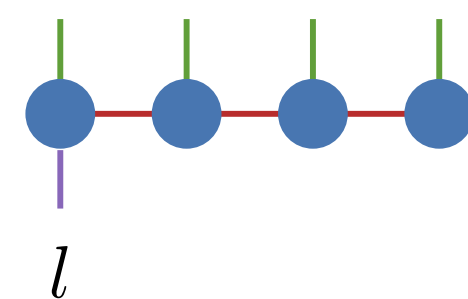
Top Signal – Mean of 5000 events



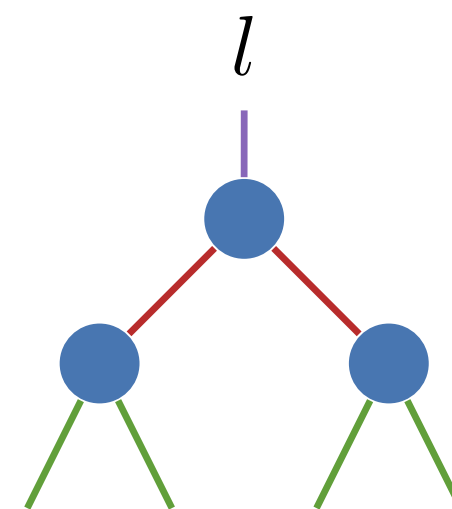
QCD Background – Mean of 5000 events



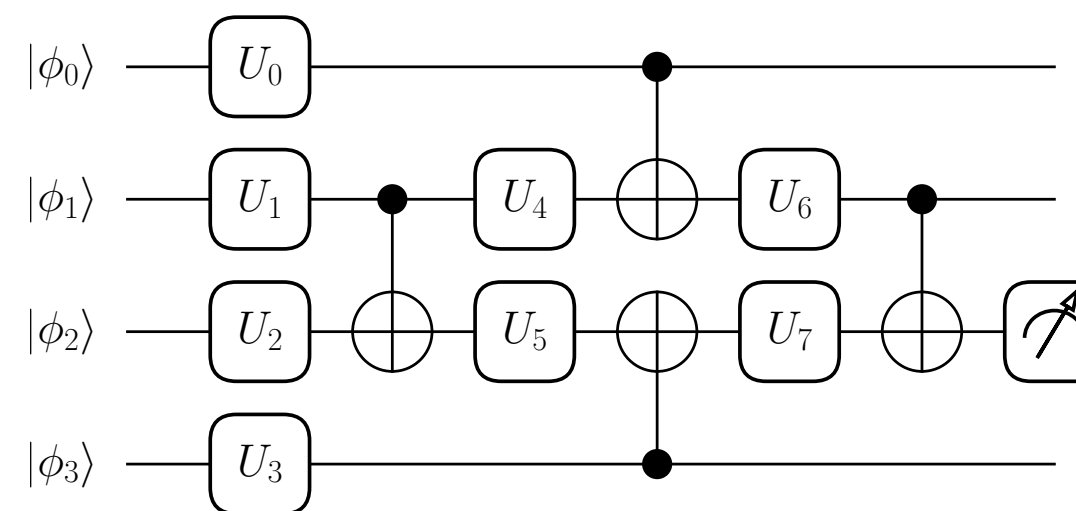
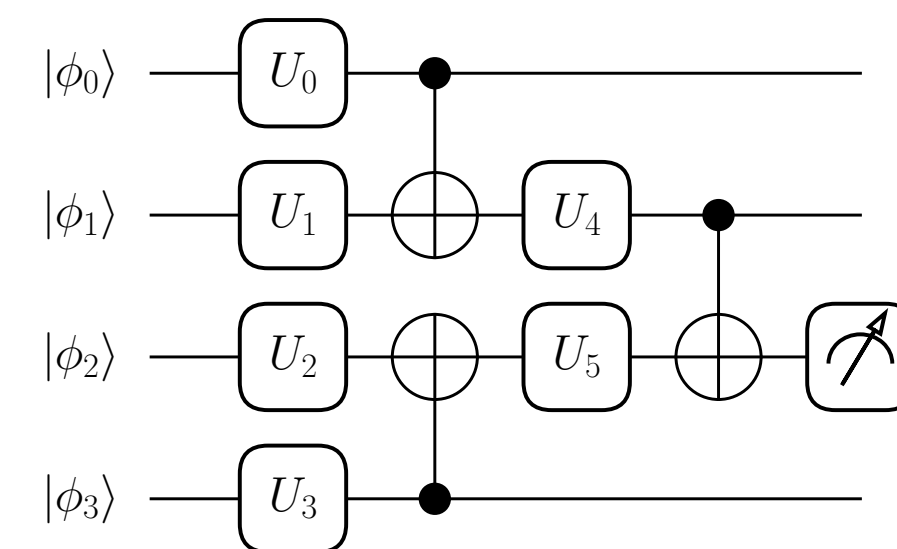
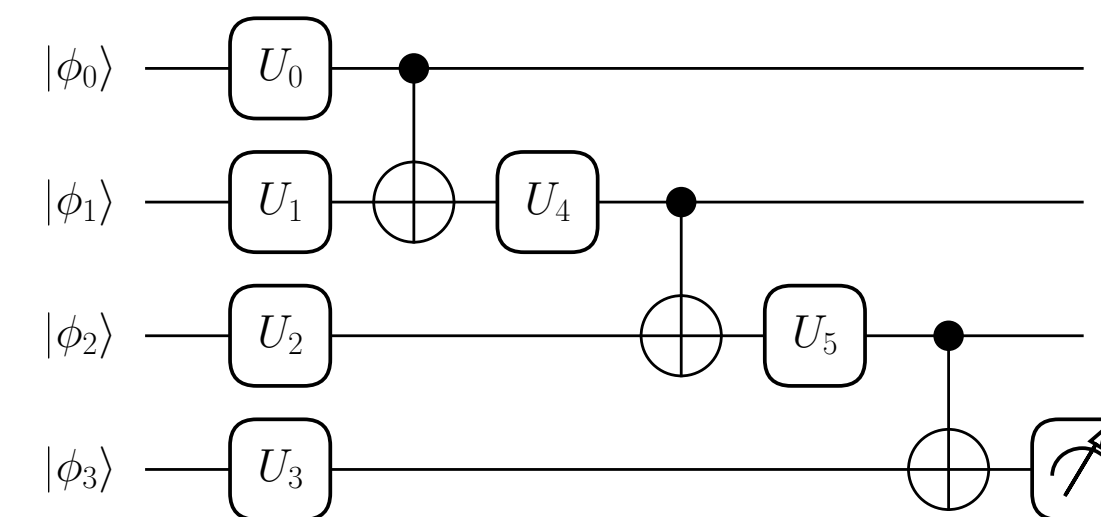
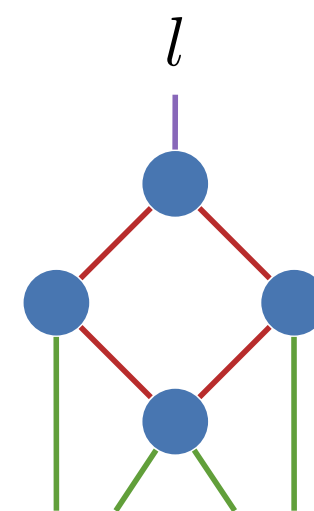
Matrix Product States



Tree Tensor Networks



Multiscale Entanglement Renormalisation Ansatz



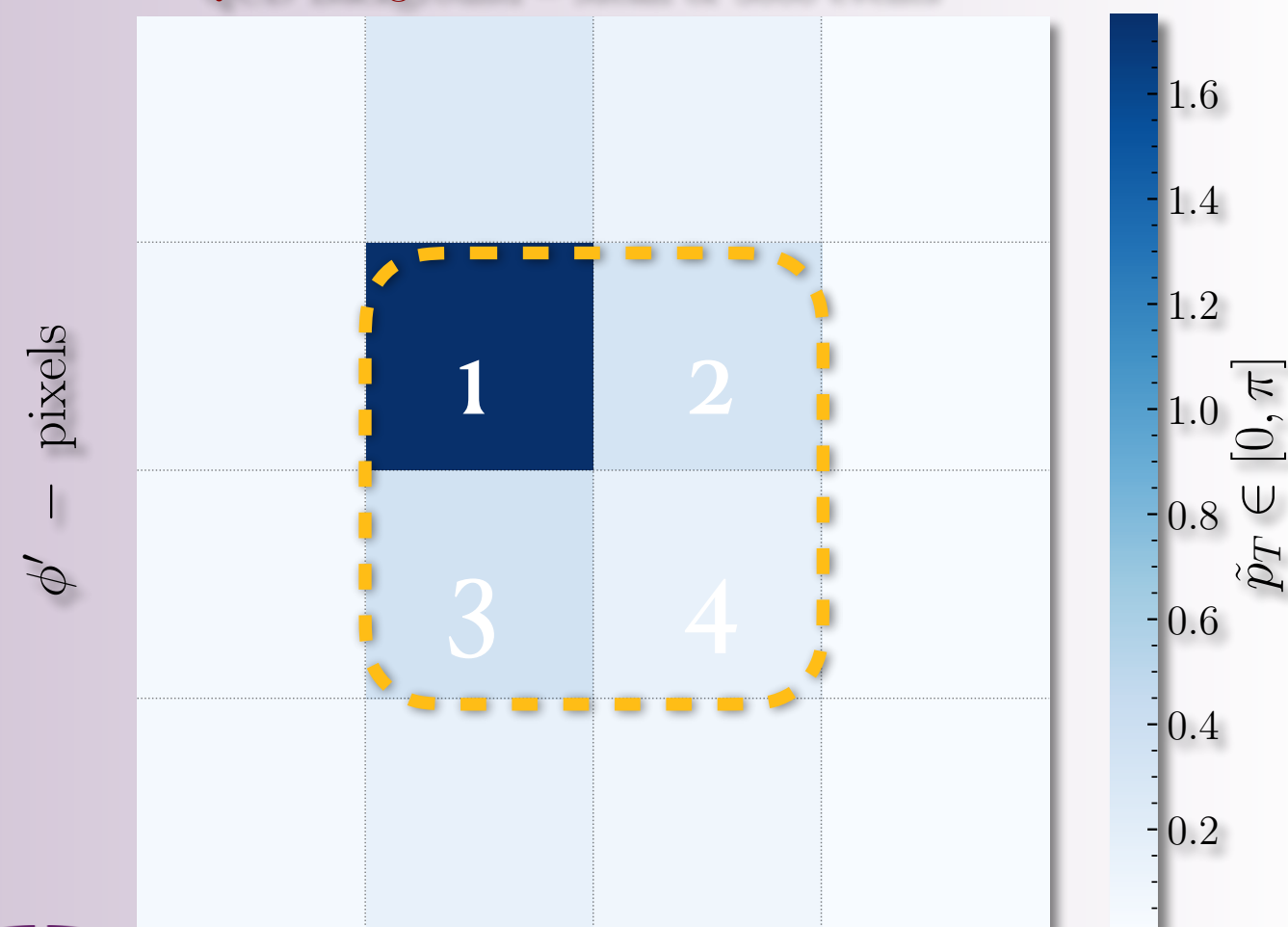
Experimenting with 4-Qubits

Top Signal – Mean of 5000 events

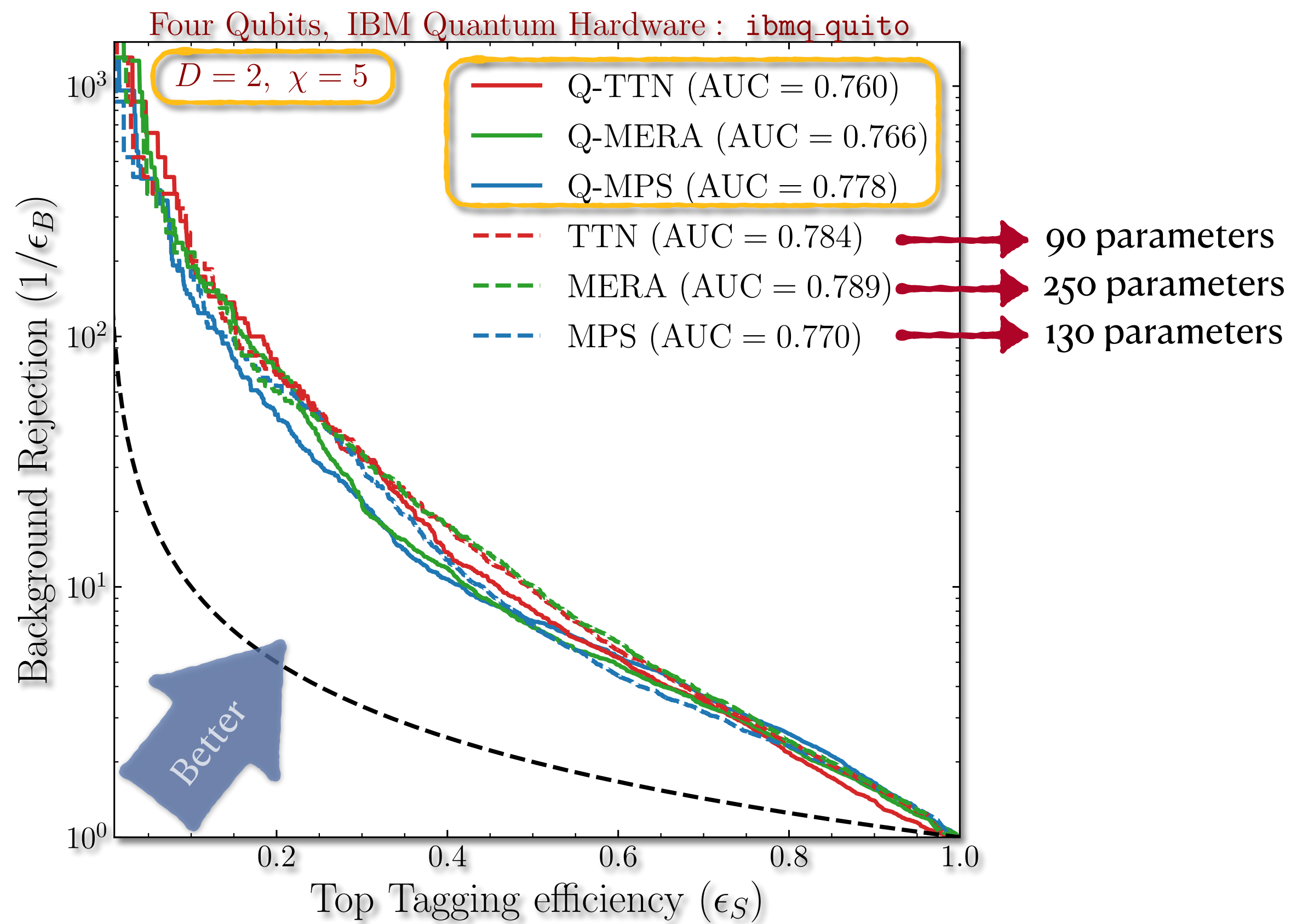


η' – pixels

QCD Background – Mean of 5000 events



η' – pixels



Singular Value Decomposition

$$\begin{array}{ccccccc}
 & m \times n & & m \times k & & k \times l & & l \times n \\
 \left[\begin{array}{cccc} \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \end{array} \right] & = & \left[\begin{array}{cccc} \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \end{array} \right] & \left[\begin{array}{ccc} \bullet & & \\ & \bullet & \\ & & \bullet \end{array} \right] & \left[\begin{array}{cccc} \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \end{array} \right] \\
 \mathbf{M} & = & \mathbf{U} & \mathbf{S} & \mathbf{V}^\dagger
 \end{array}$$

Orthogonal singular column vectors

Positive definite singular values in descending order

Orthogonal singular row vectors

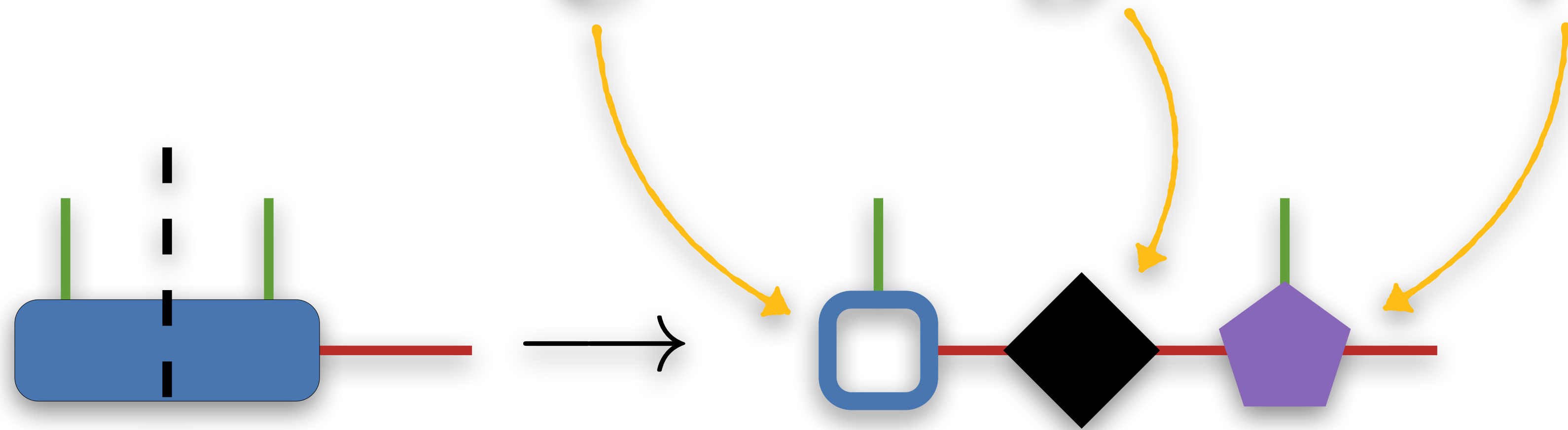
By changing the number of singular values one can change the accuracy of the decomposition!!!

$$\mathbf{S} = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_n)$$

λ_i also known as Schmidt values

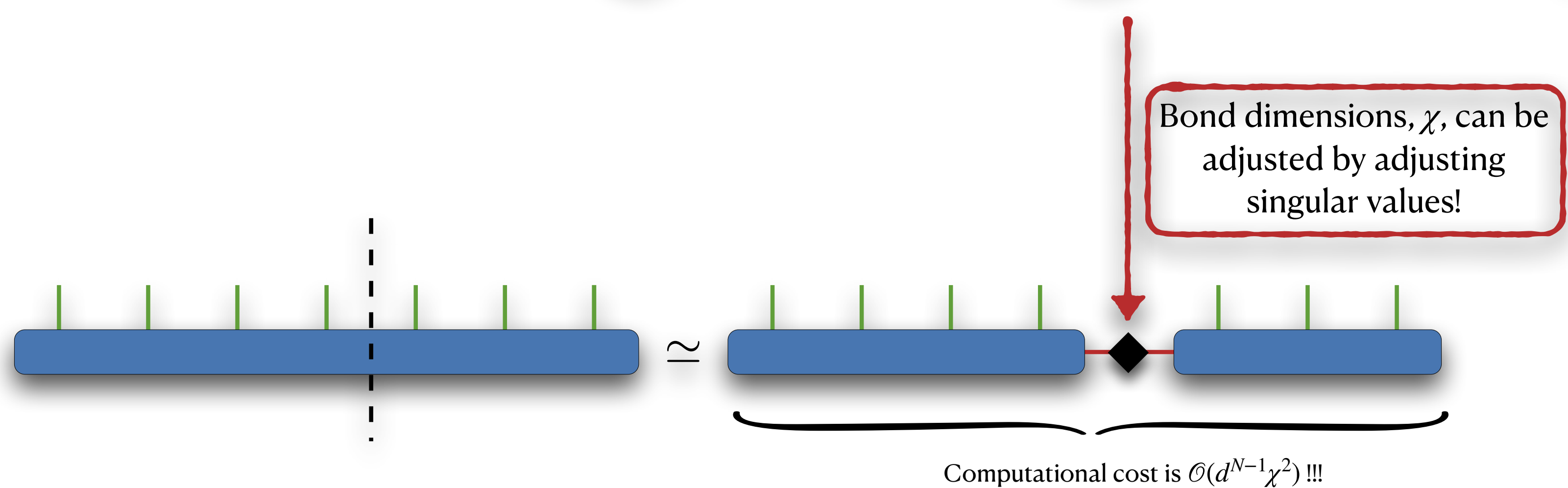
Singular Value Decomposition

$$\begin{matrix}
 m \times n & & m \times k & & k \times l & & l \times n \\
 \left[\begin{array}{cccc} \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \end{array} \right] & = & \left[\begin{array}{cccc} \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \end{array} \right] & \left[\begin{array}{cc} \bullet & \\ & \bullet \\ & \\ & \bullet \end{array} \right] & \left[\begin{array}{cccc} \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \end{array} \right] \\
 \mathbf{M} & = & \mathbf{U} & \mathbf{S} & \mathbf{V}^\dagger
 \end{matrix}$$



Singular Value Decomposition

$$\begin{matrix}
 m \times n & & m \times k & & k \times l & & l \times n \\
 \left[\begin{array}{cccc} \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \end{array} \right] & = & \left[\begin{array}{cccc} \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \end{array} \right] & \left[\begin{array}{cc} \bullet & \\ & \bullet \end{array} \right] & \left[\begin{array}{cccc} \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \end{array} \right] \\
 \mathbf{M} & = & \mathbf{U} & \mathbf{S} & \mathbf{V}^\dagger
 \end{matrix}$$



Matrix Product States for Classification

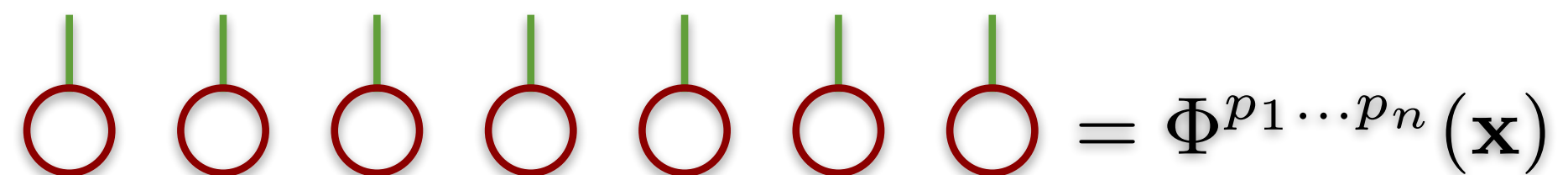
Sub-Outline

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

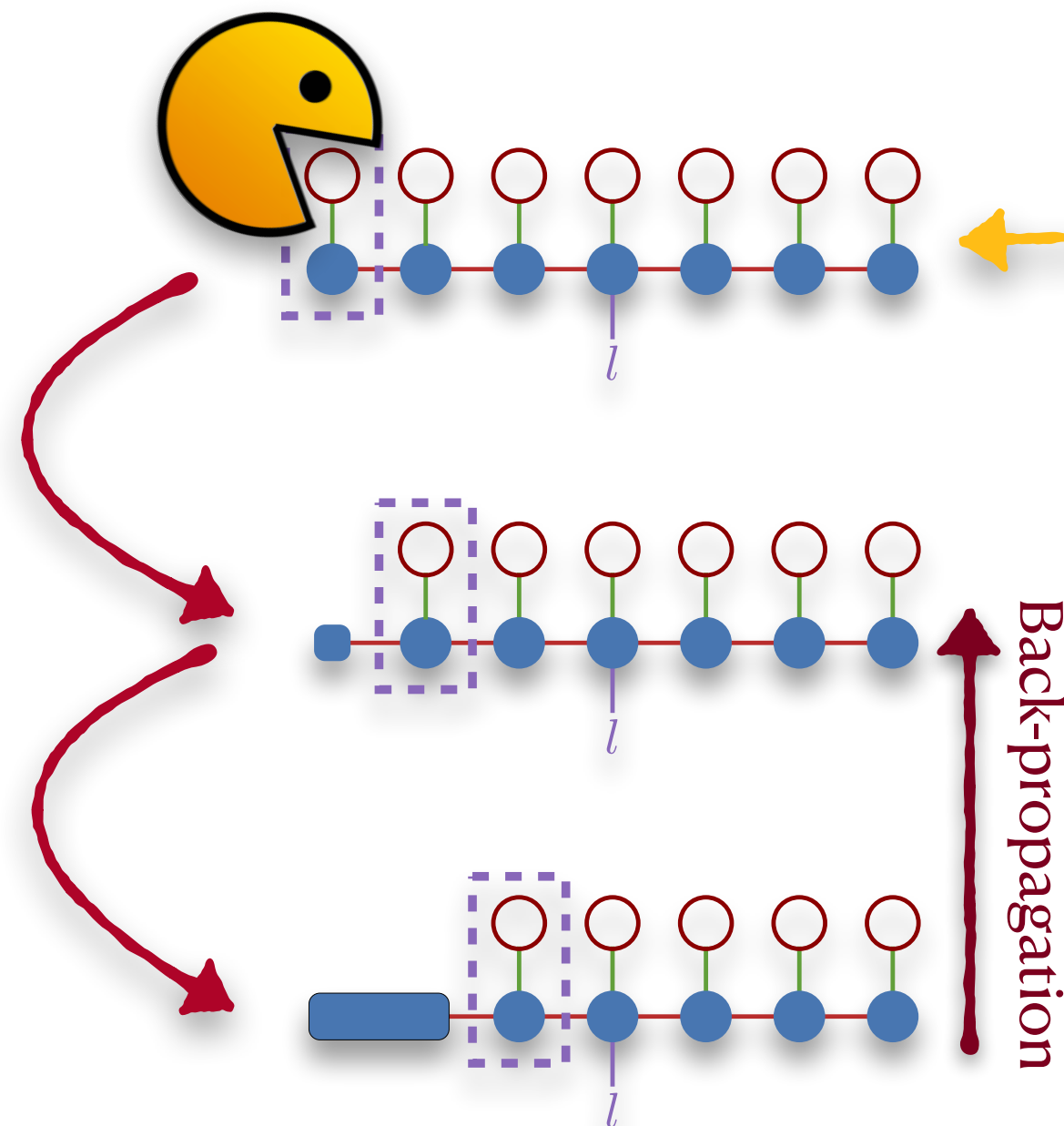
Data Embedding

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

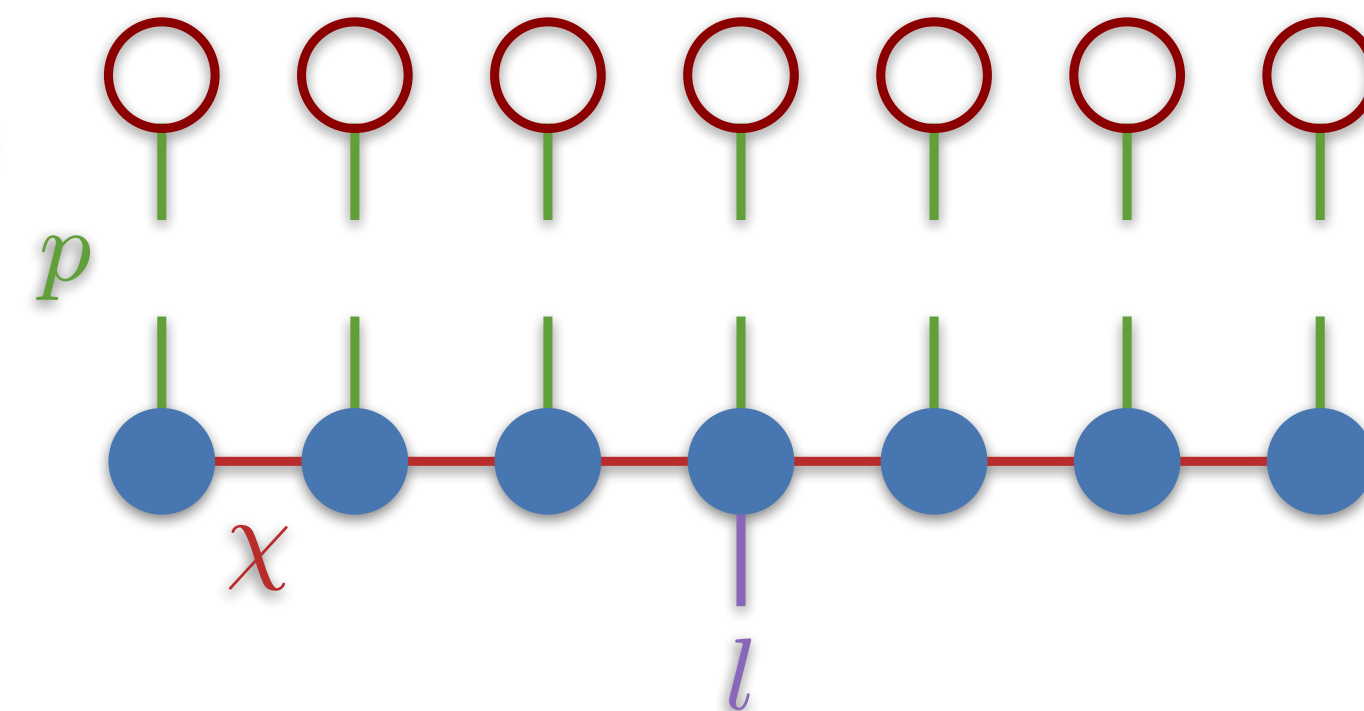
$$\phi^{p_i}(x_i) = \begin{bmatrix} \cos(x_i \pi/2) \\ \sin(x_i \pi/2) \end{bmatrix} \text{ or } \phi^{p_i}(x_i) = \begin{bmatrix} 1 \\ x_i \\ x_i^2 \end{bmatrix} \text{ or } \dots$$



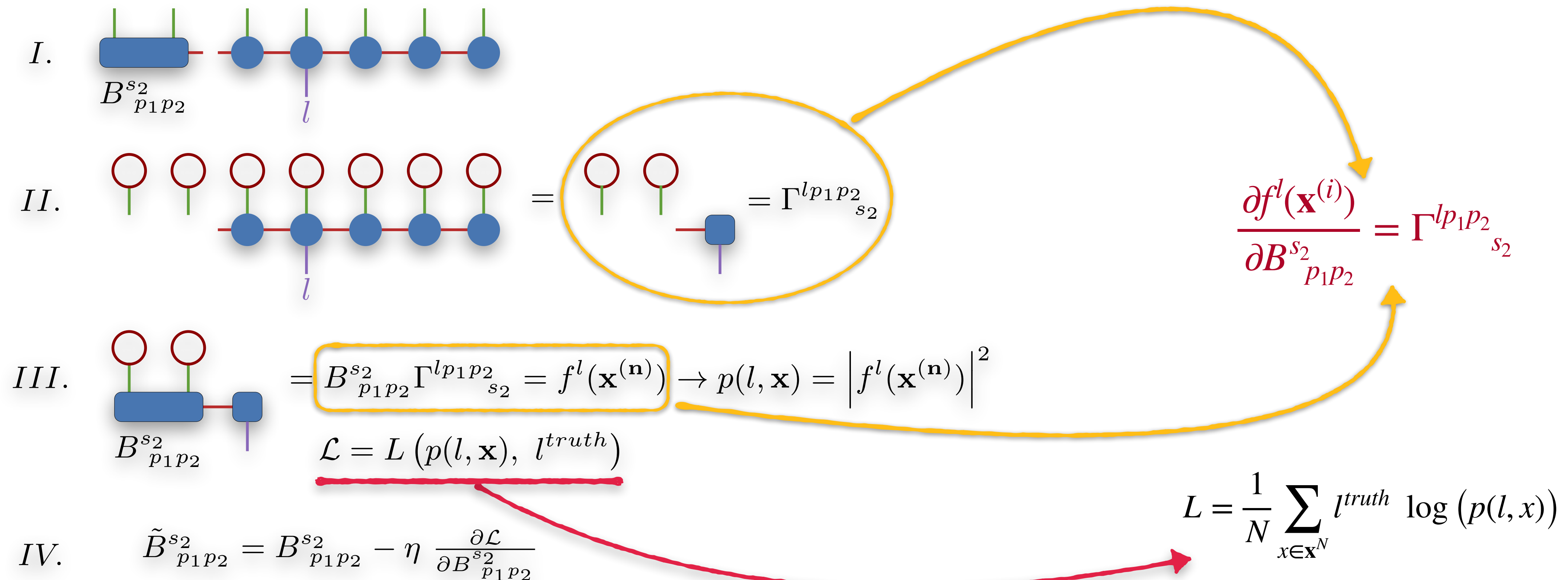
$$\text{Seven red circles} = \Phi^{p_1 \dots p_n}(\mathbf{x})$$



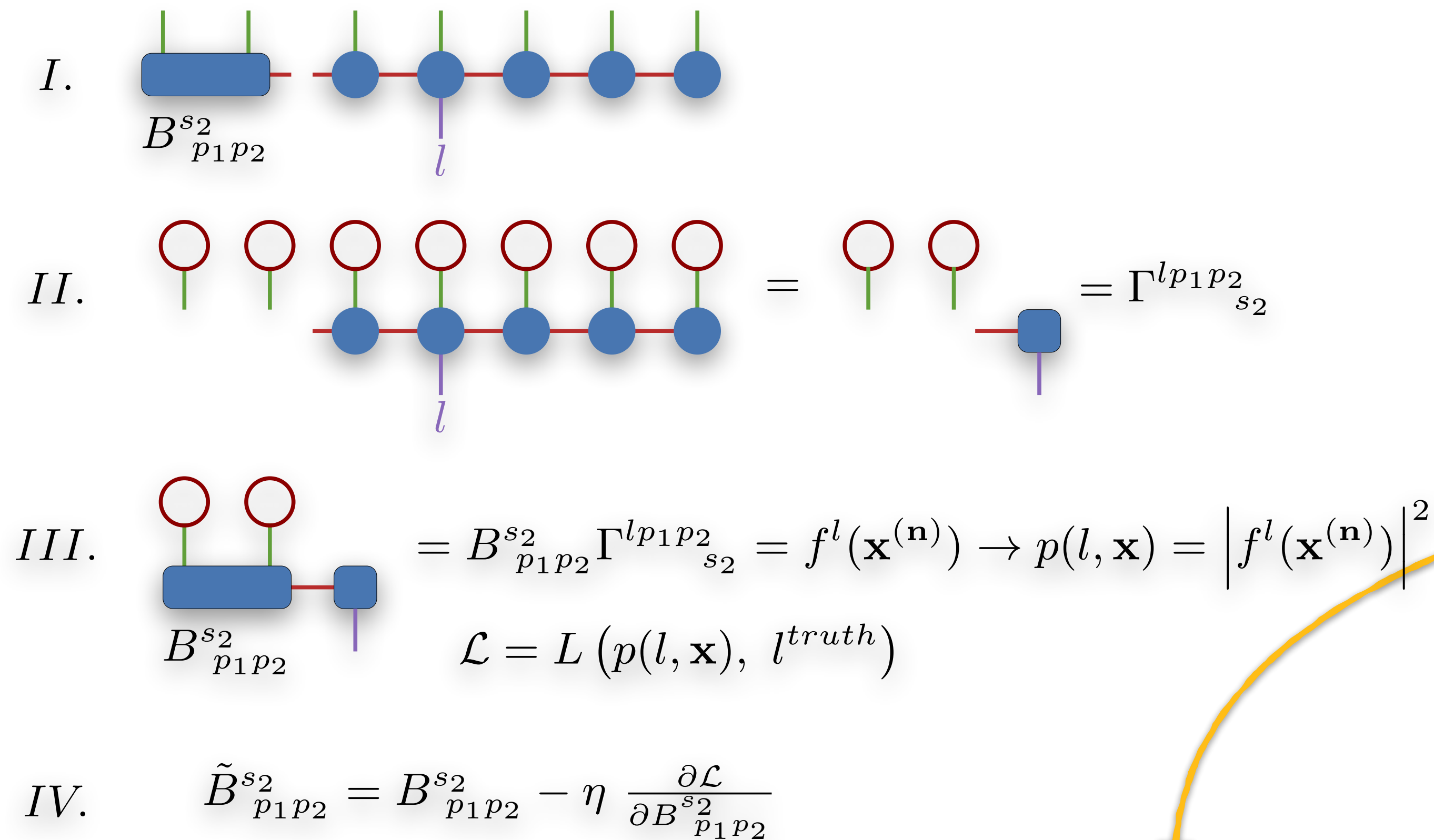
An efficient contraction algorithm is essential for training!



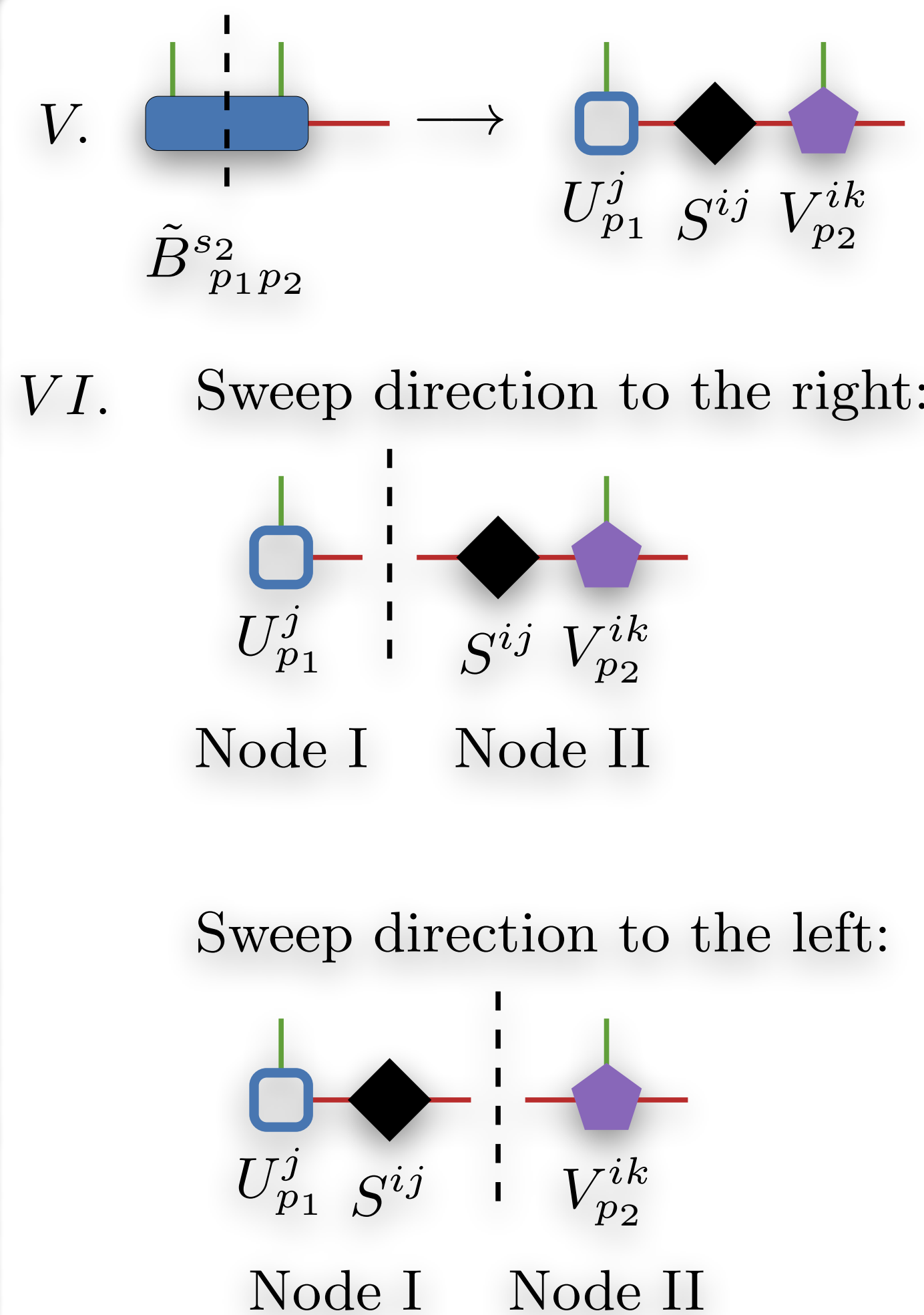
Density Matrix Renormalization Group Algorithm



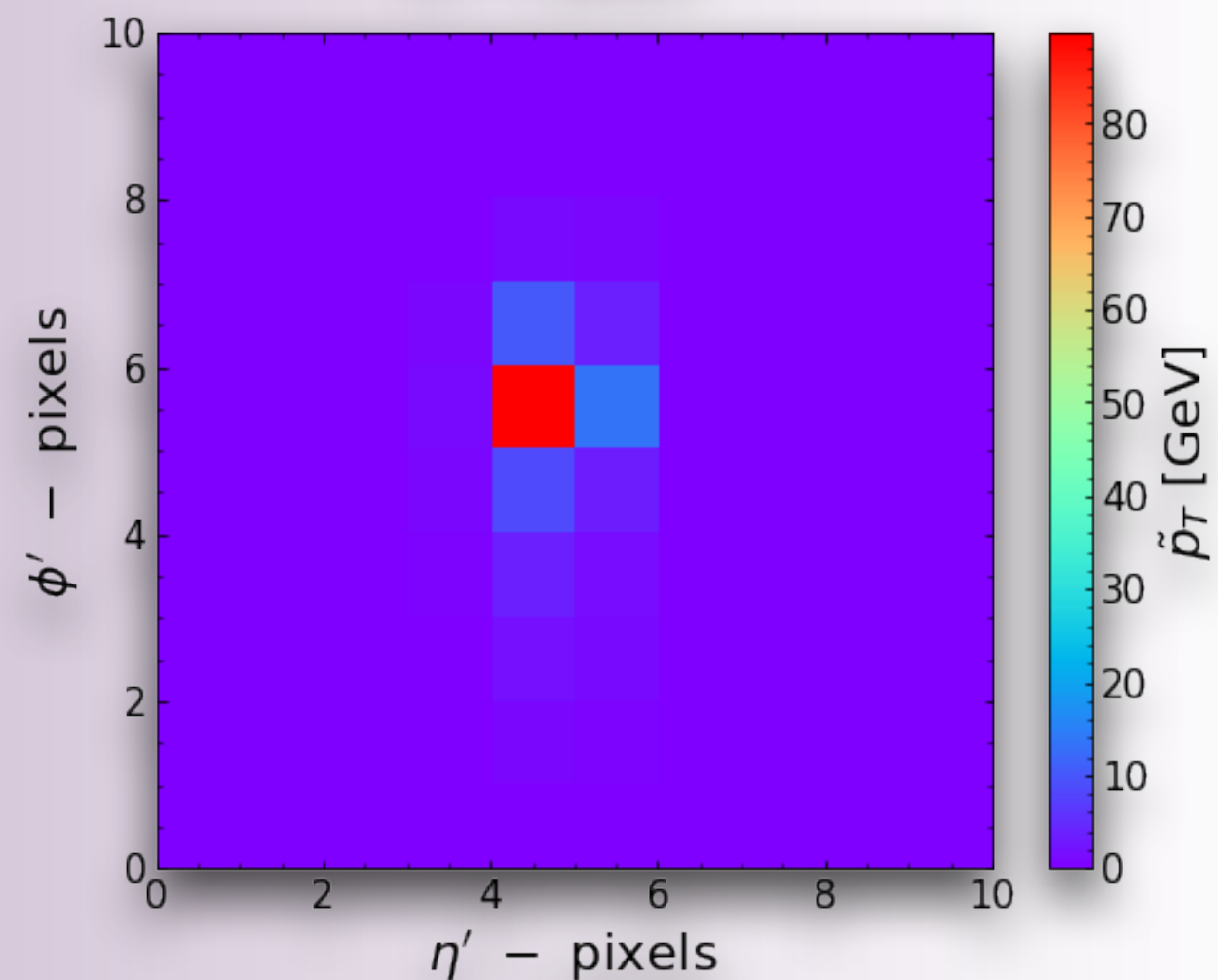
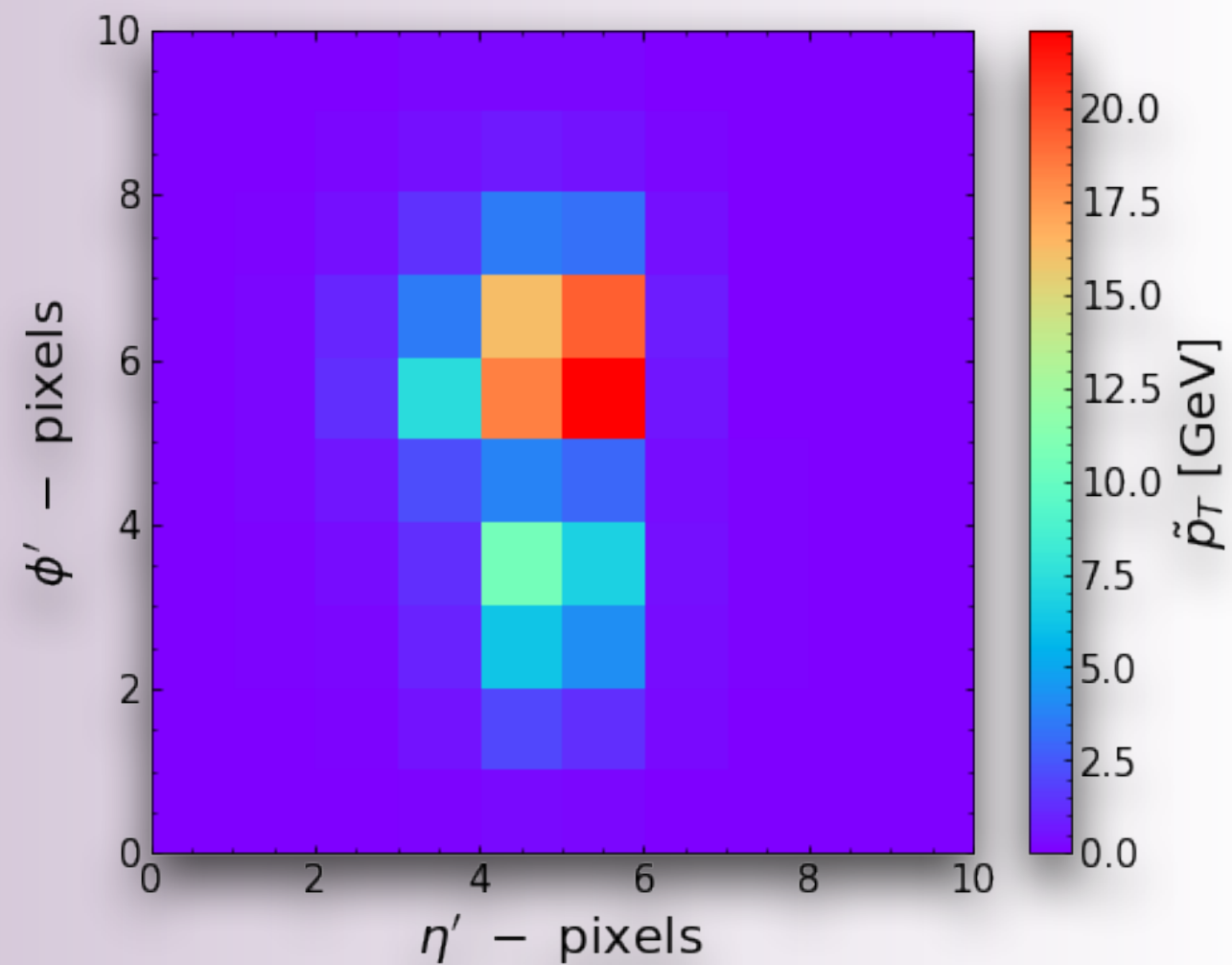
Density Matrix Renormalization Group Algorithm



Adjust the bond dimension via SVD!

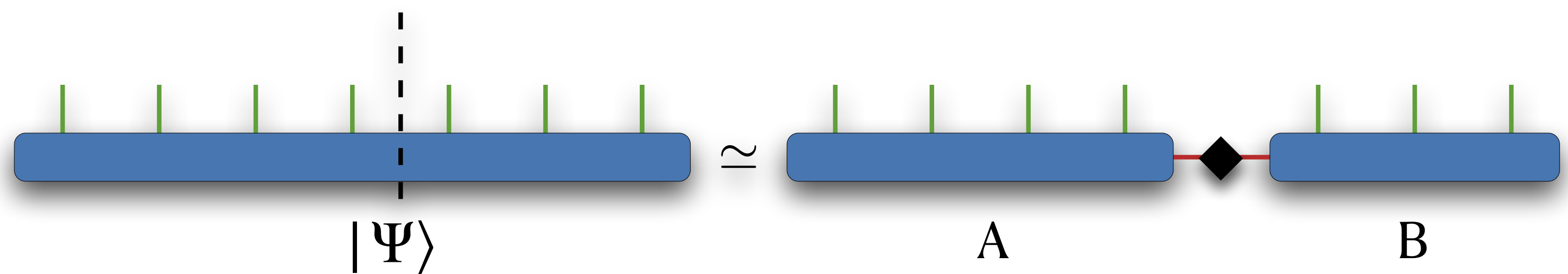


Top Tagging through MPS



Assumptions & Requirements

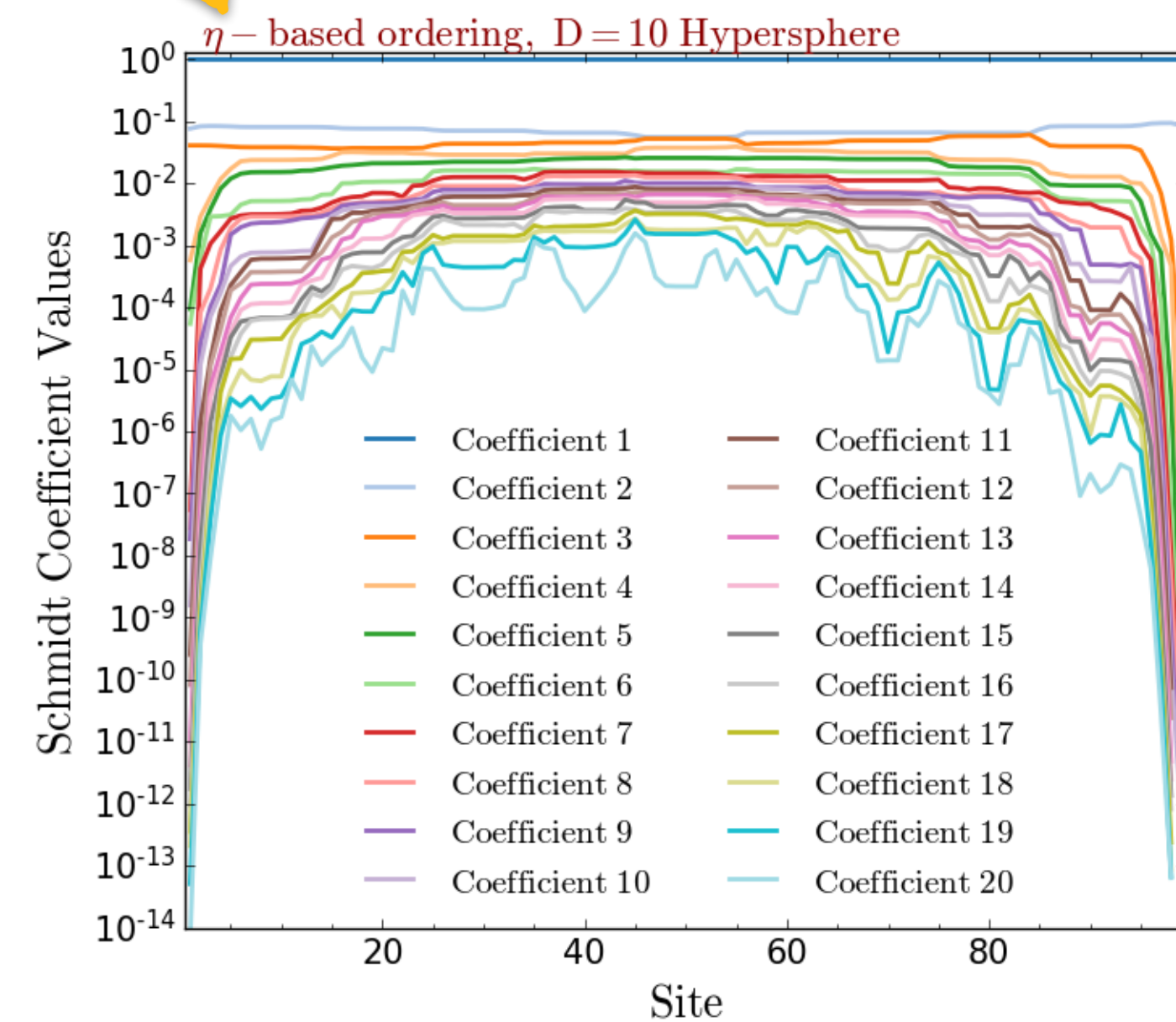
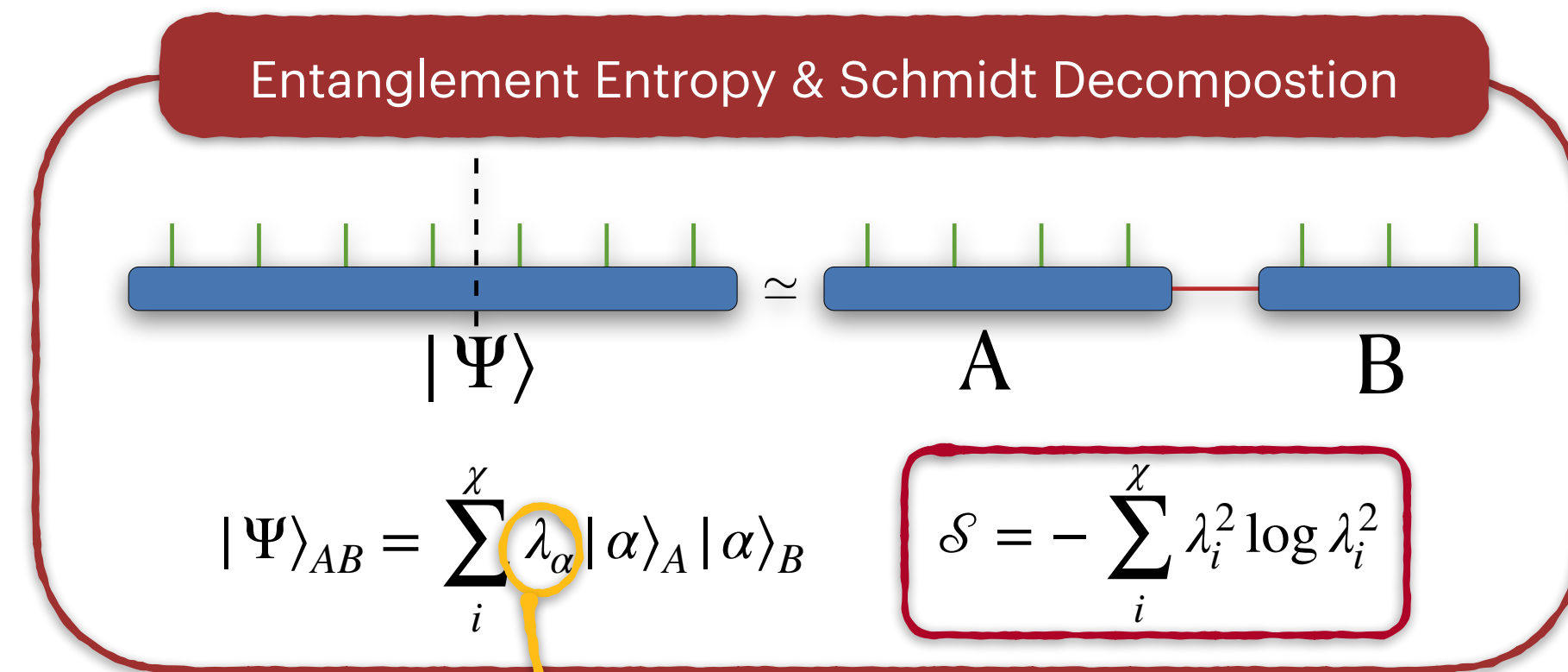
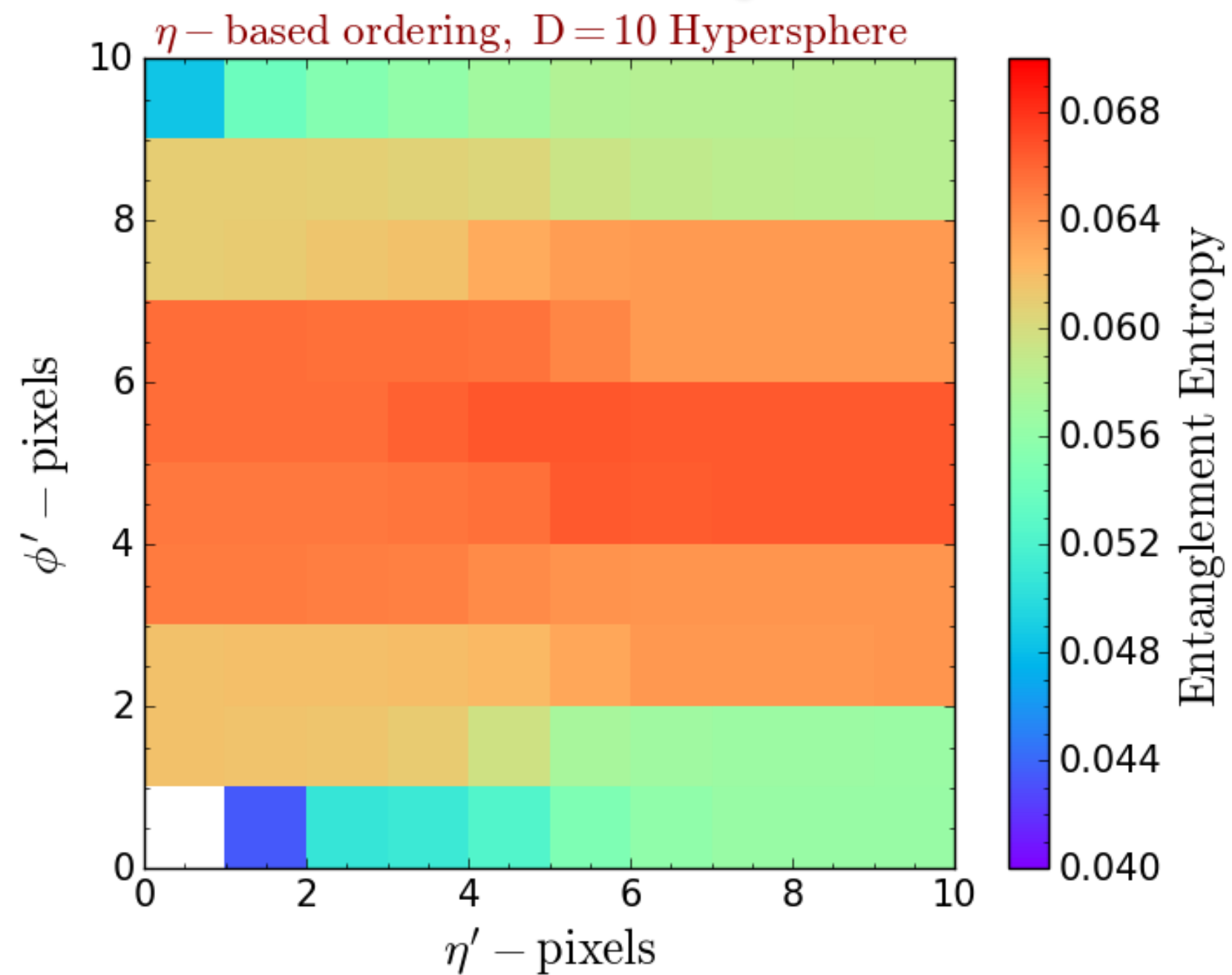
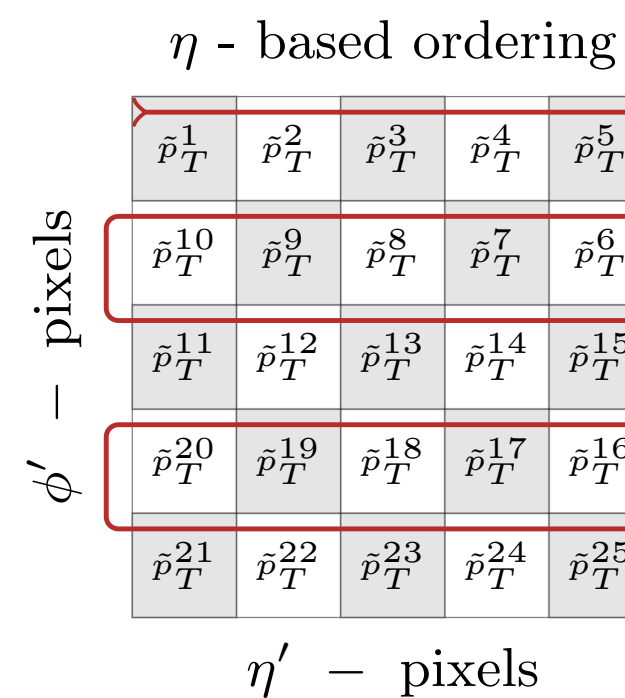
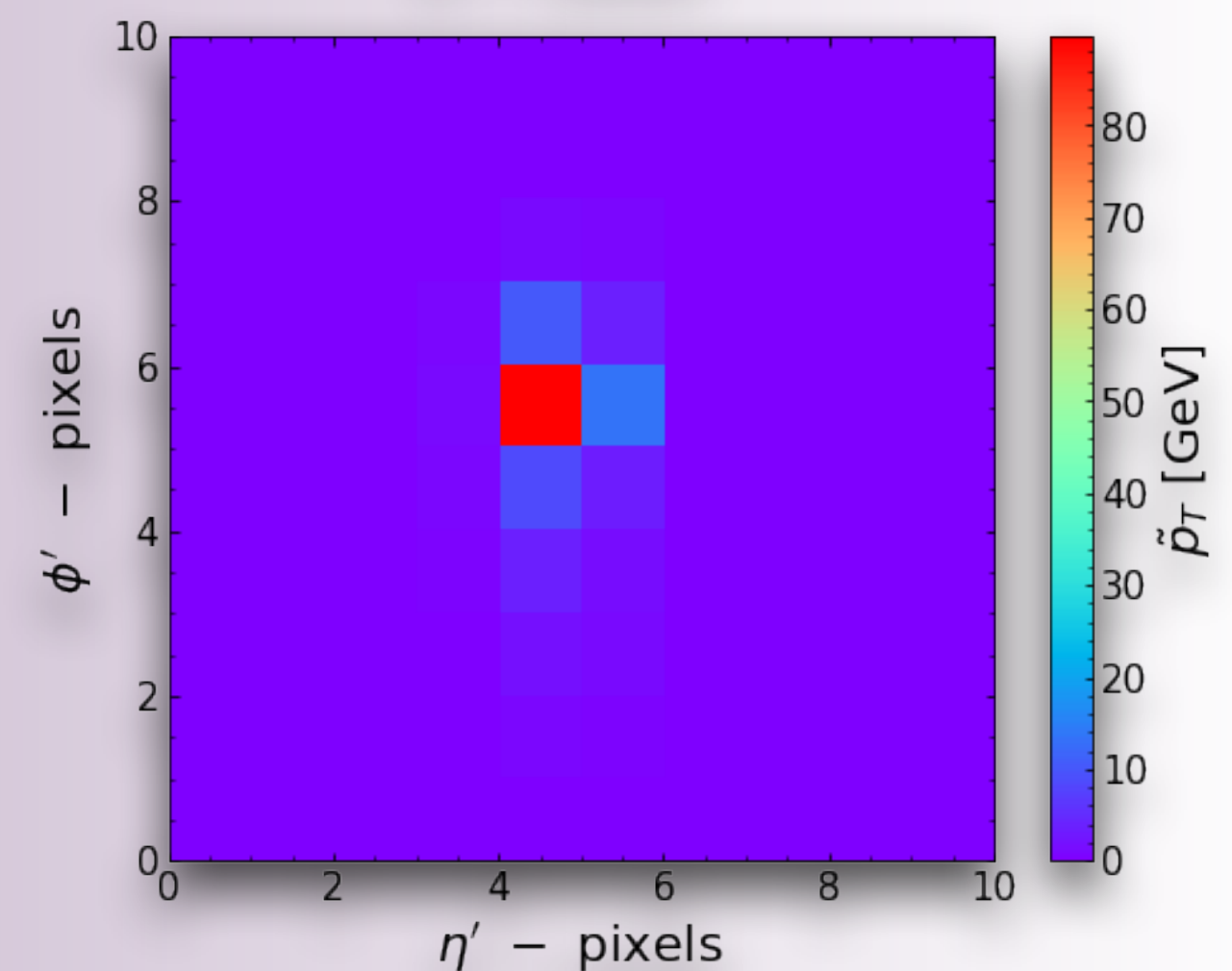
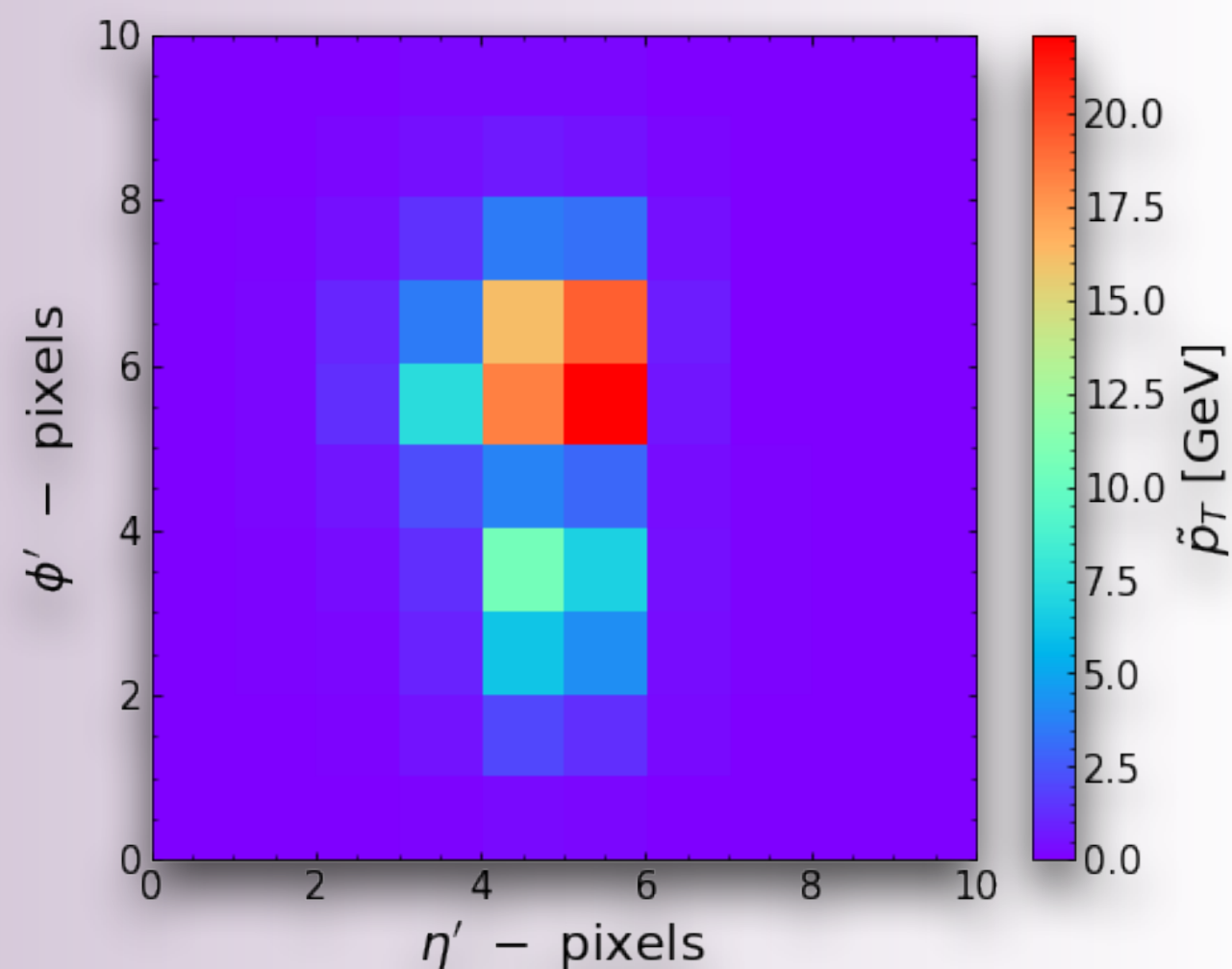
Entanglement Entropy & Schmidt Decomposition



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

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

Top Tagging through MPS



Fisher Information & Effective Dimensions

