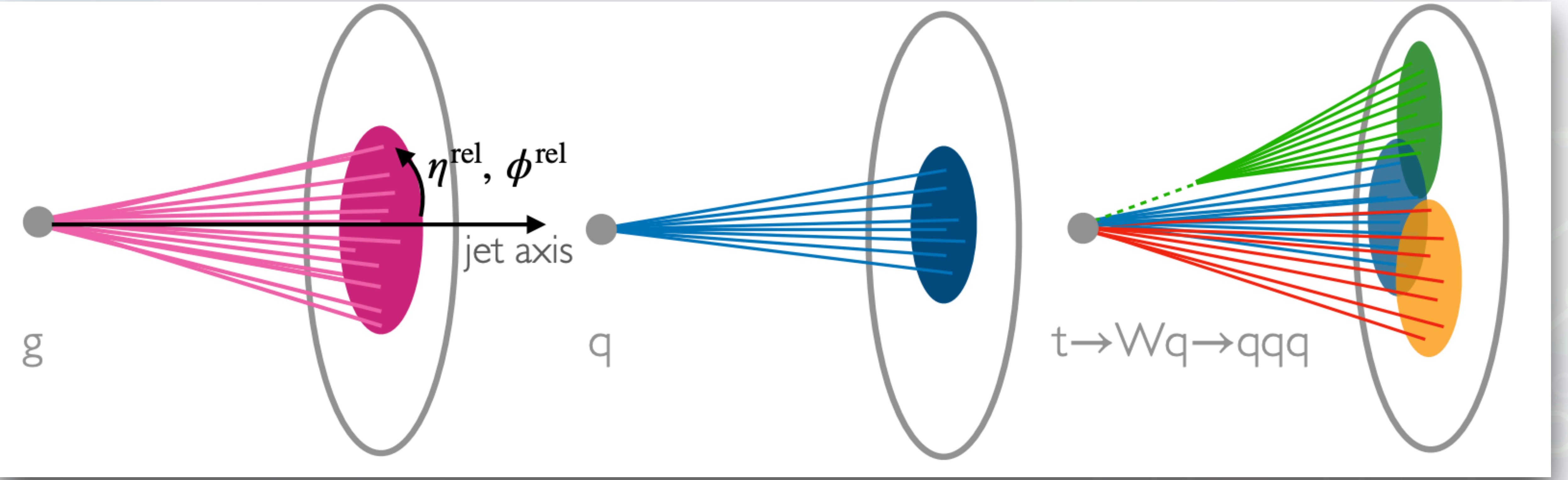
# Jet discrimination with a quantum complete graph neural network

**Yi-An Chen**, **Kai-Feng Chen**

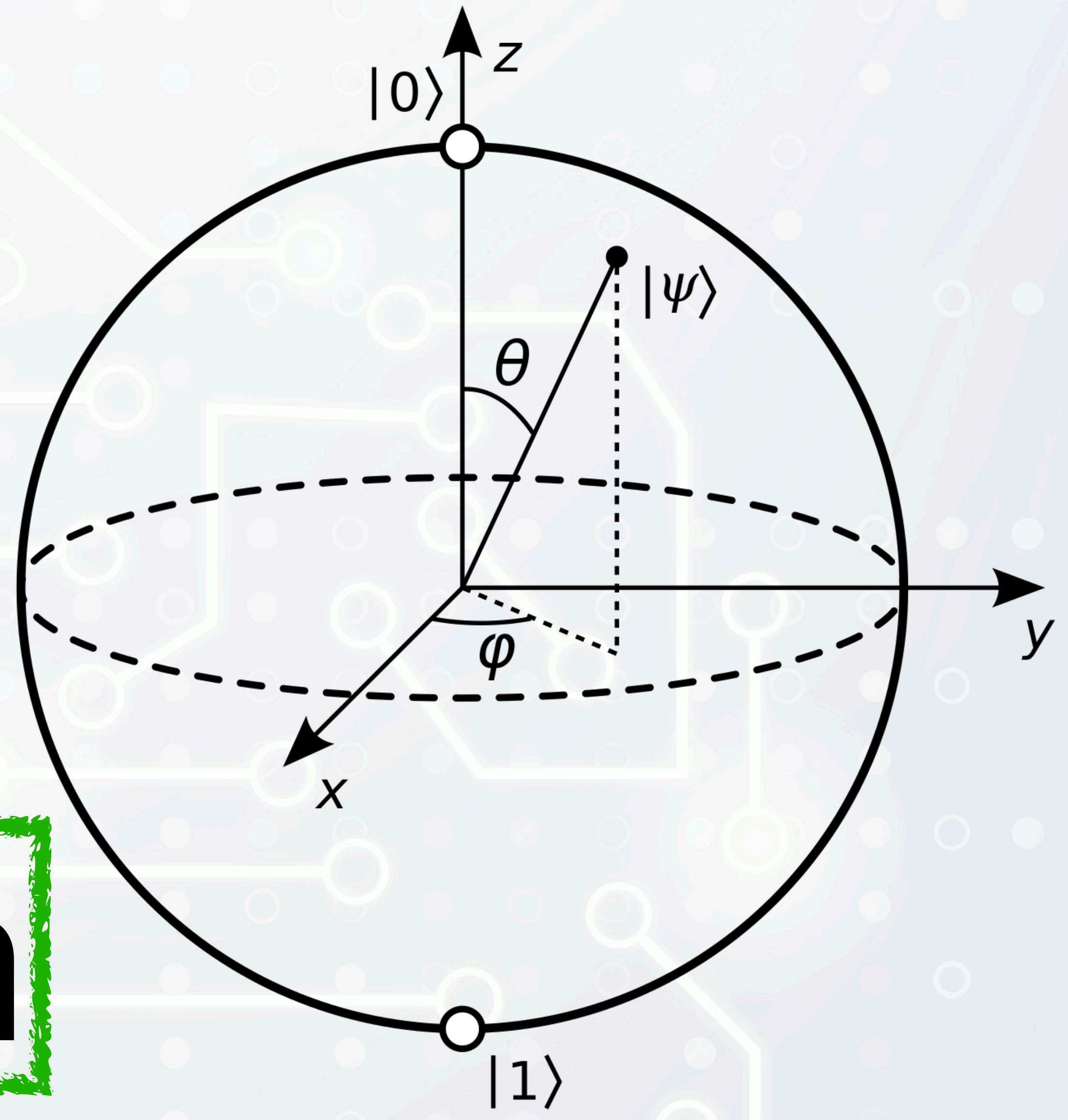
*Department of Physics, National Taiwan University, Taipei, Taiwan*

*arXiv:2403.04990*
*Phys. Rev. D 111, 016020*

March 18, 2025 ISGC @ Academia Sinica

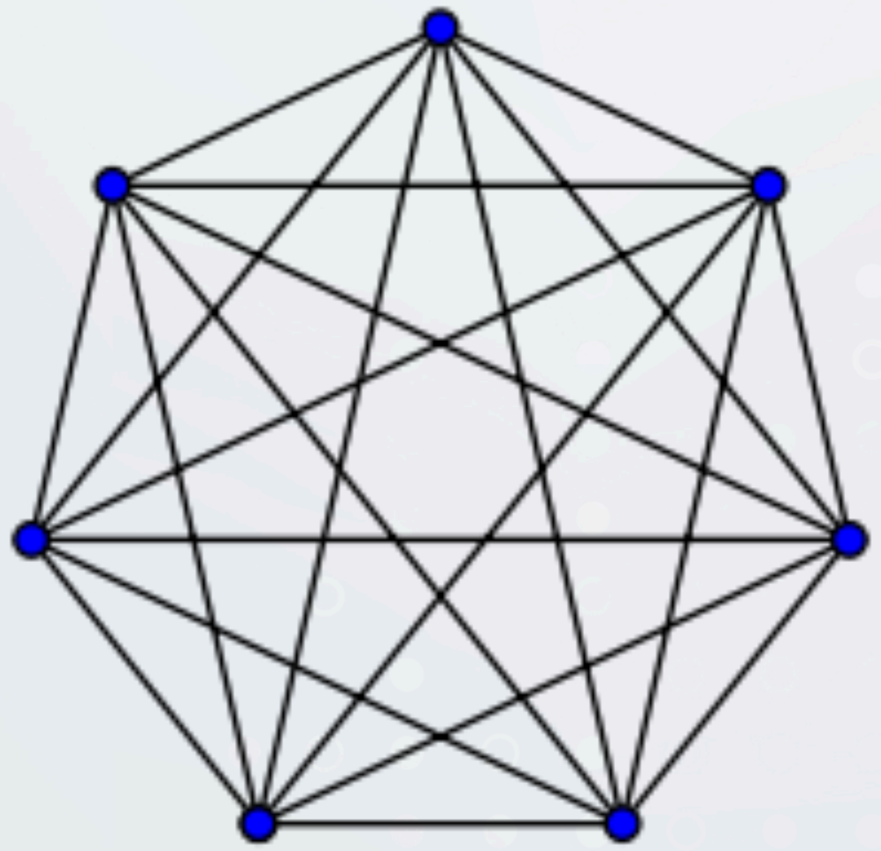



Jets, adapted from (arxiv 2106.11535)

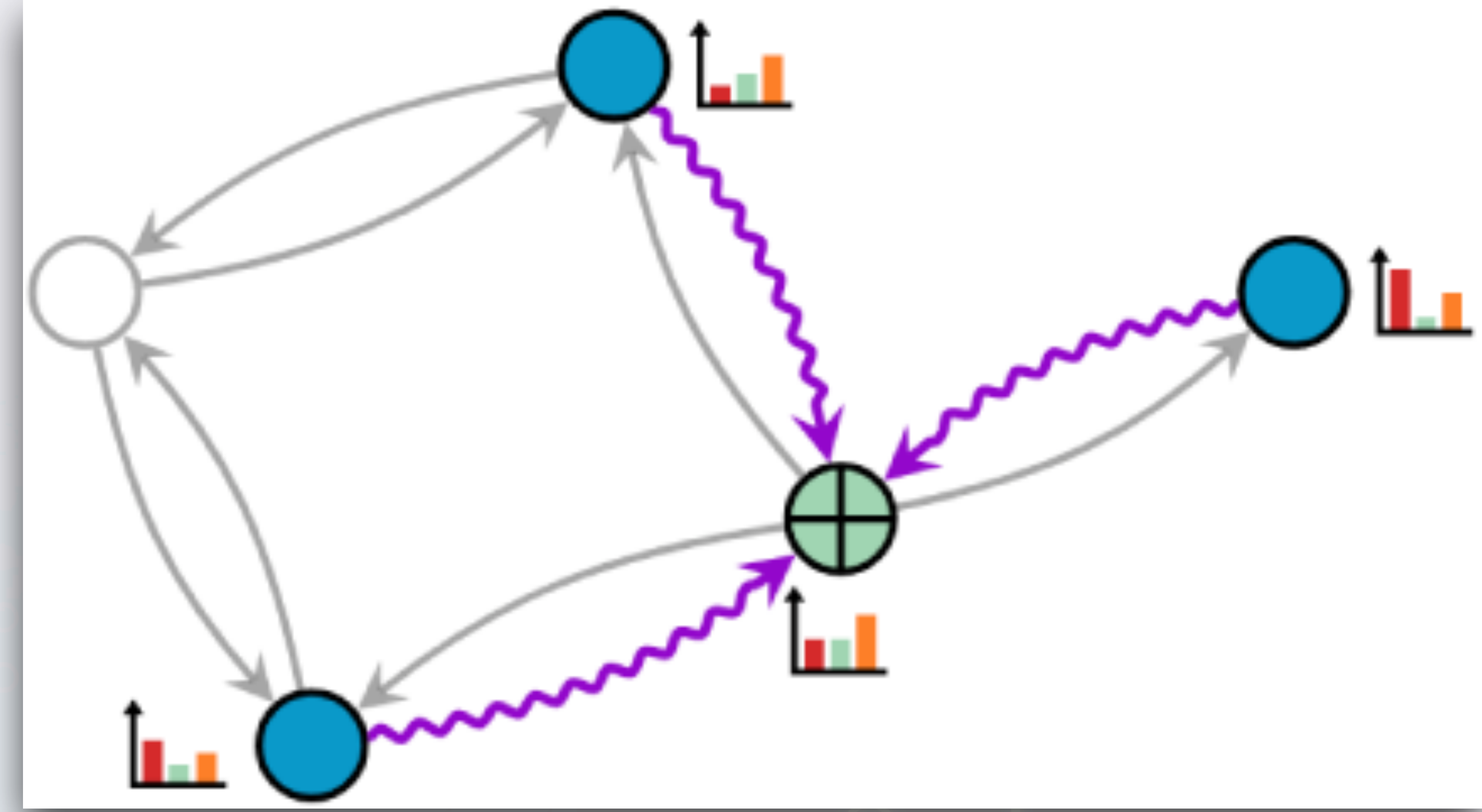



Bloch Sphere
(Adapted from Wikipedia)

# Jet discrimination with a quantum complete graph neural network



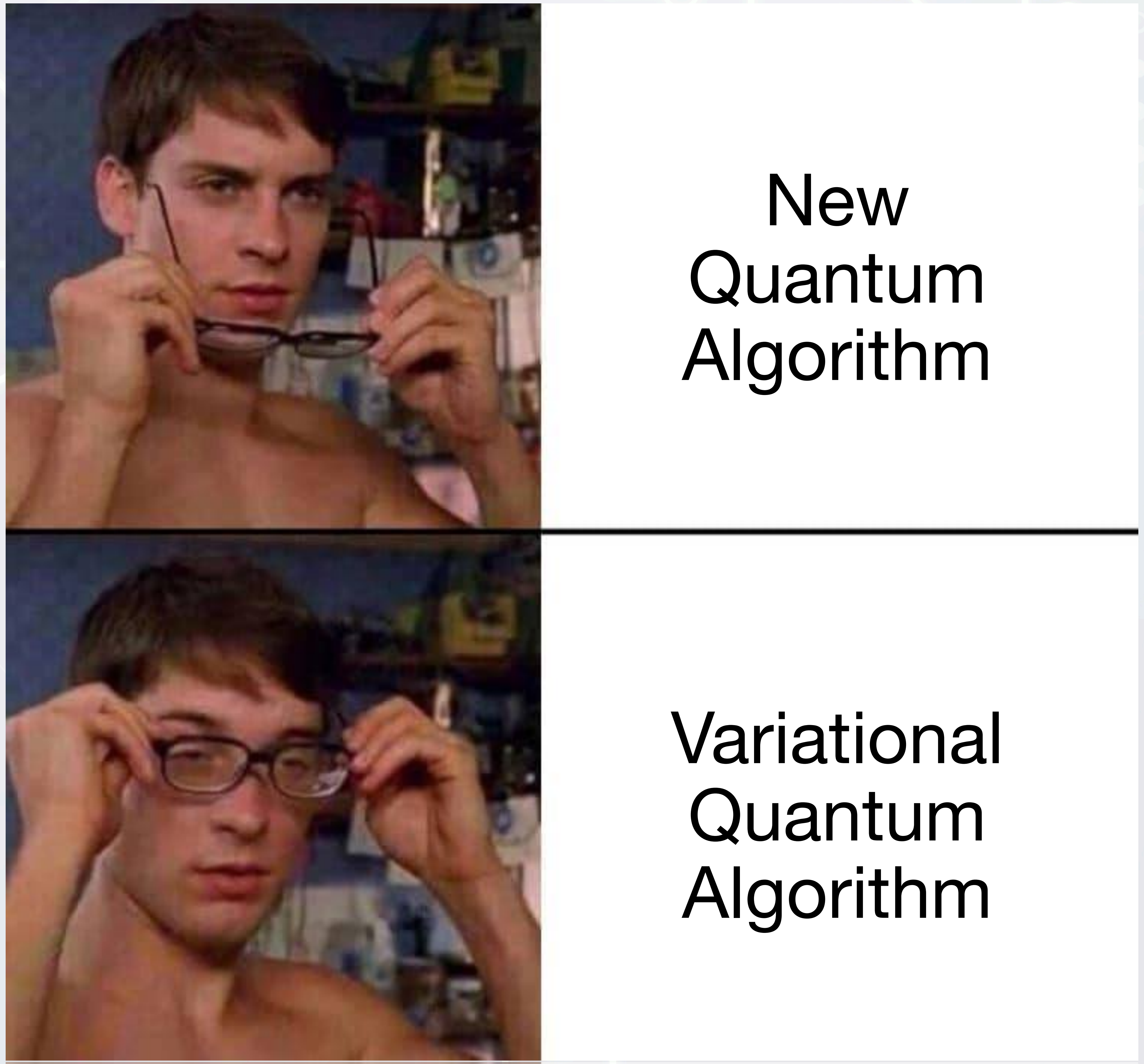
Complete Graph
(Adapted from Wikipedia)



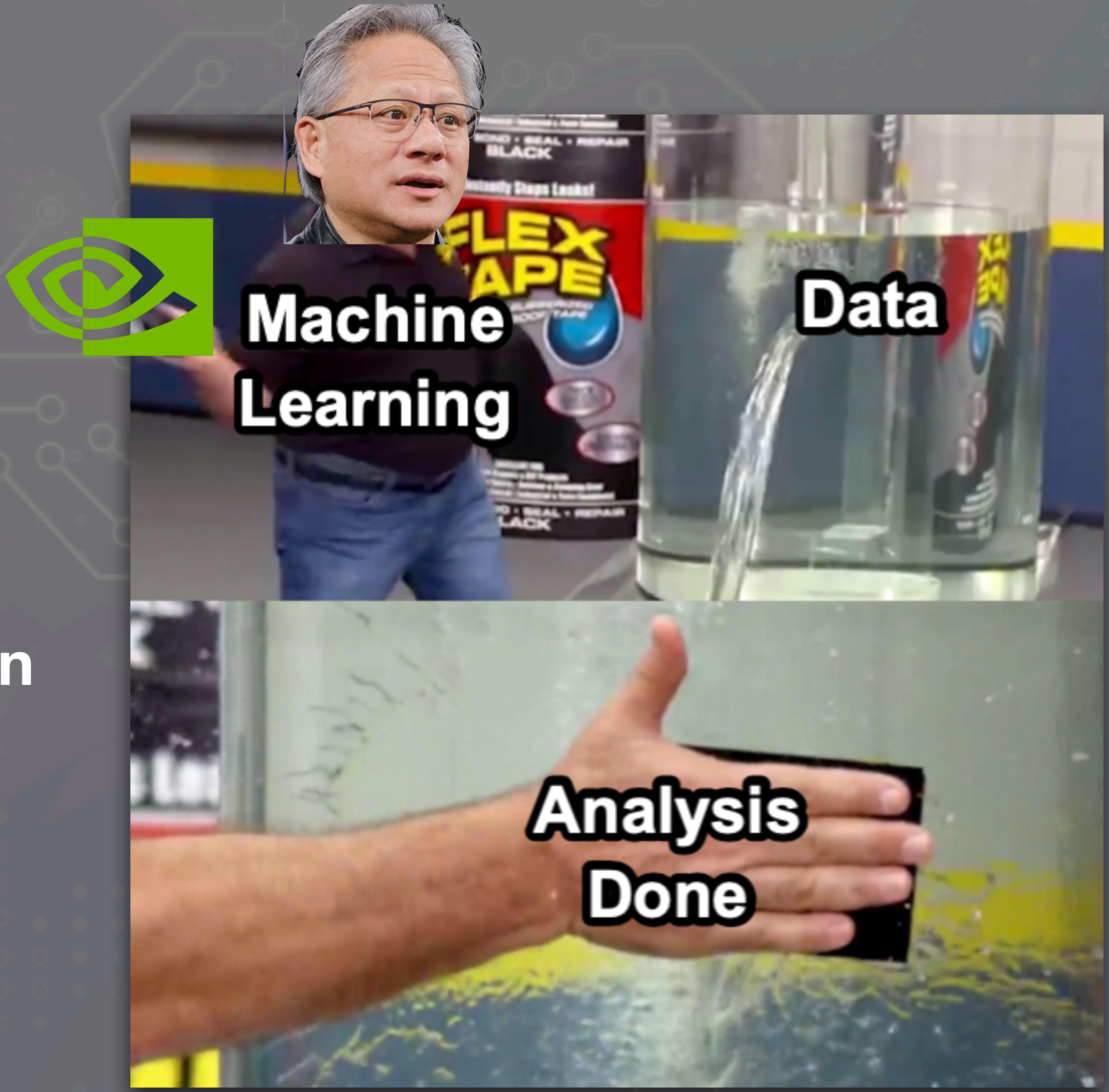
GNN (adapted from PyTorch-Geometric)

# Outline

- **Classical Machine Learning**
  - Message-Passing Graph Neural Network
- **Quantum Machine Learning**
  - Variational Quantum Circuit
  - Quantum Complete Graph Neural Network
- **Training X IBMQ X Summary**
  - Public Monte Carlo Simulated Jet Data
  - Running on IBMQ



# Classical
# Neural Networks
## Models for Jet Discrimination



# Jet Dataset

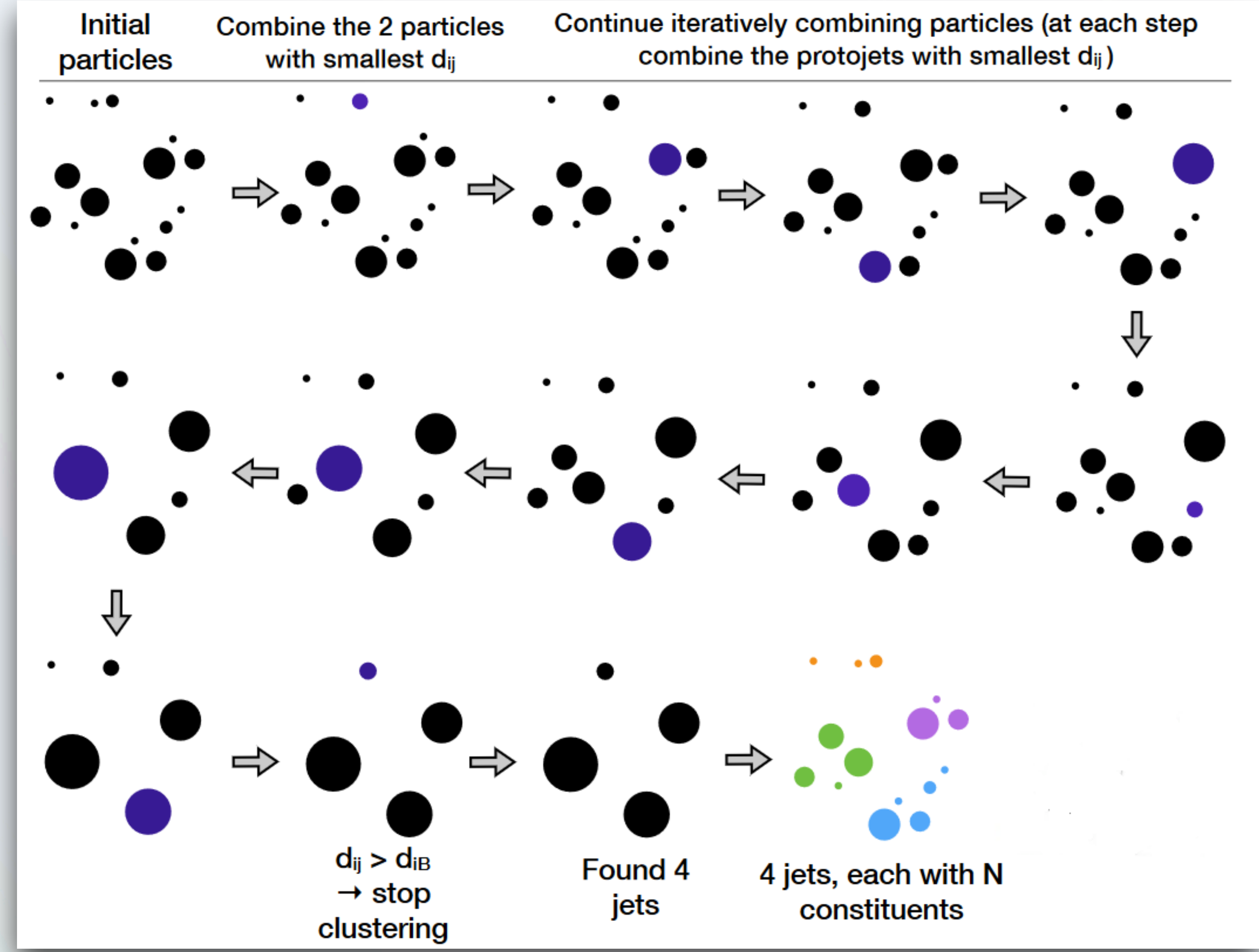
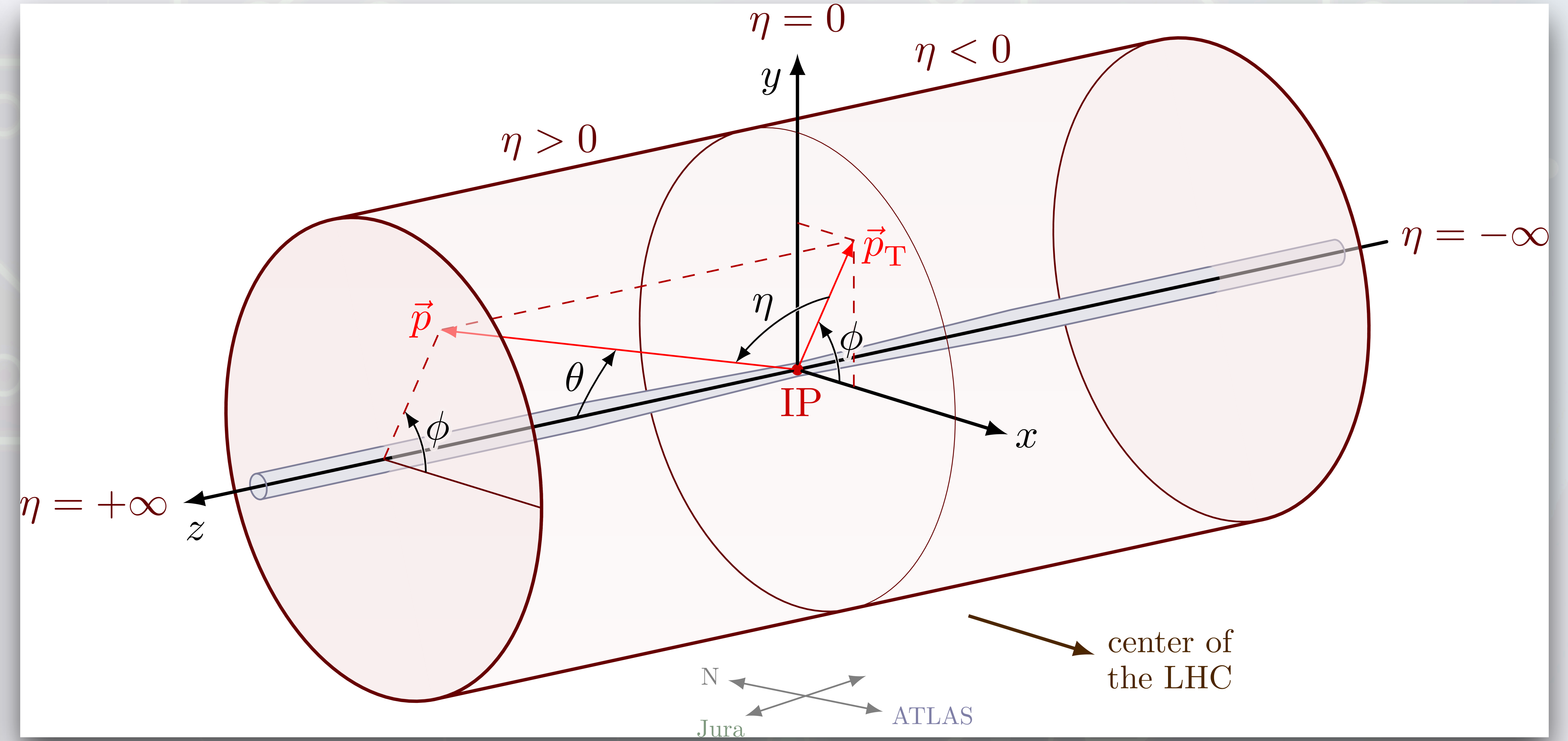



Figure adapted from 2021 CMS Open Data Workshop

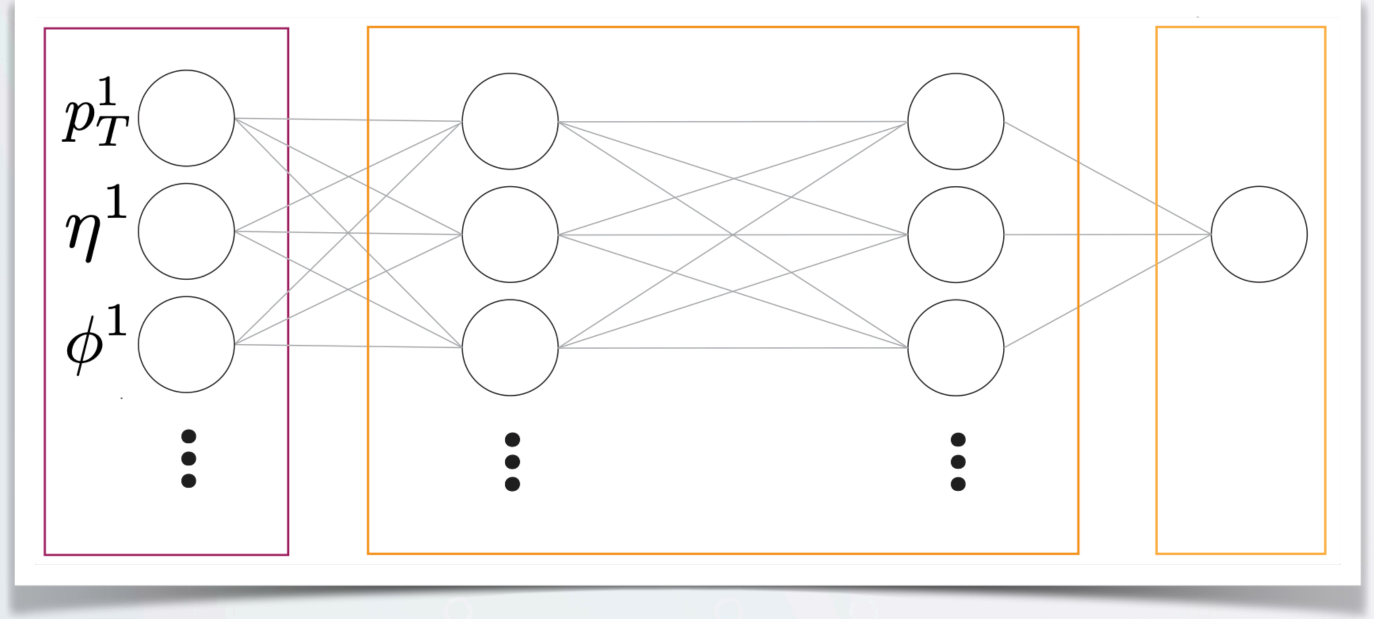



CMS Coordinate System by Izaak Neutelings

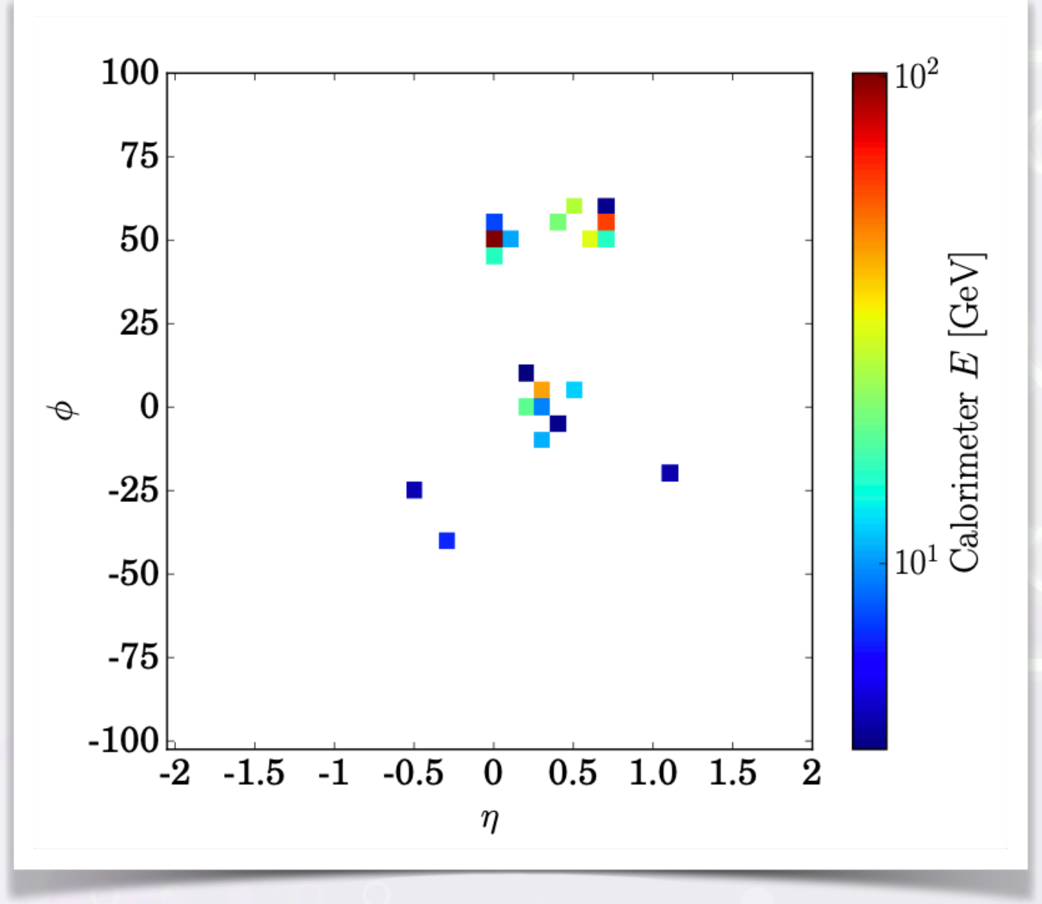
$$\overrightarrow{x} = (p_T, \Delta\eta, \Delta\phi)$$

# Classical ML for Jets

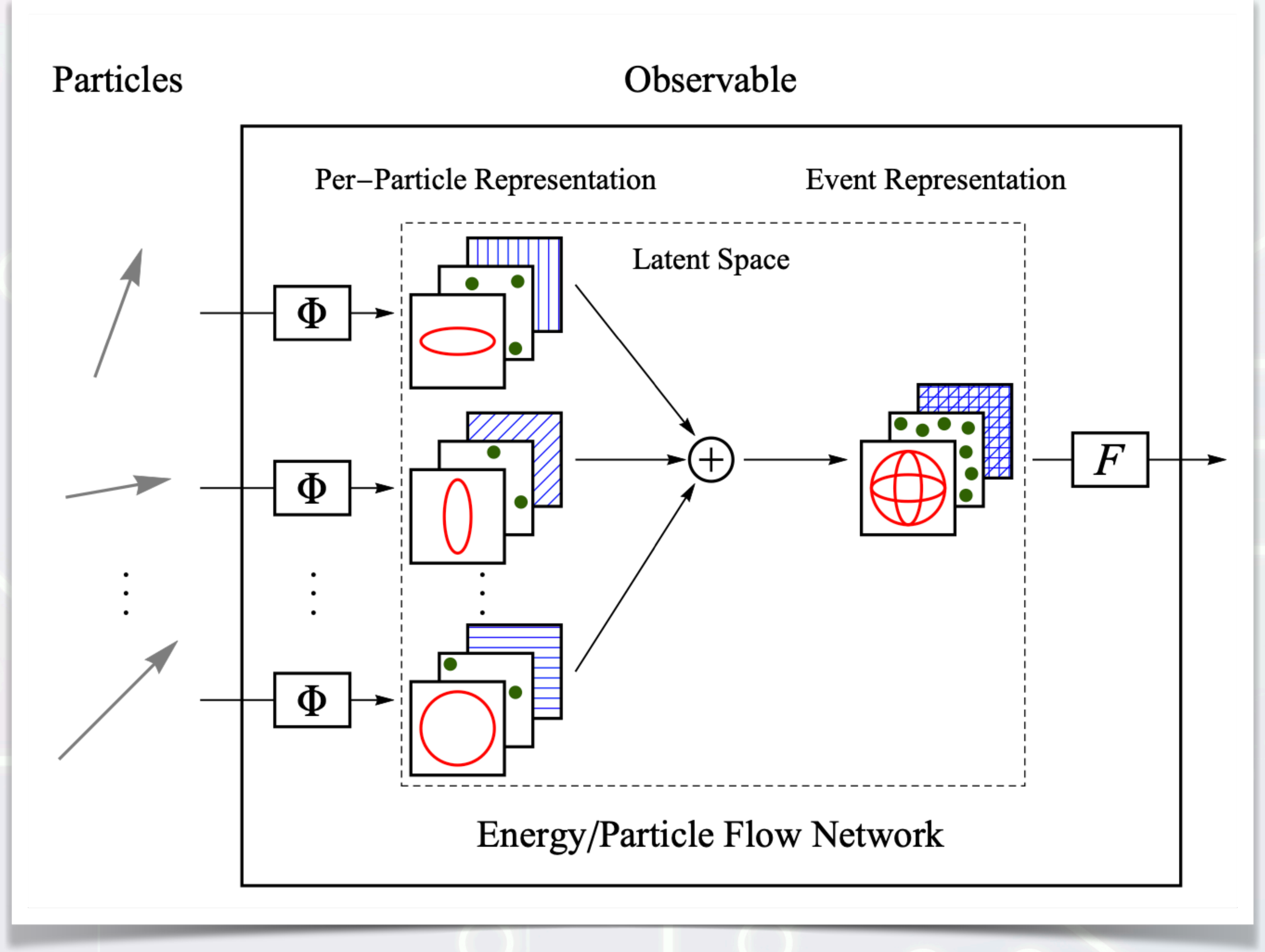
## Overview



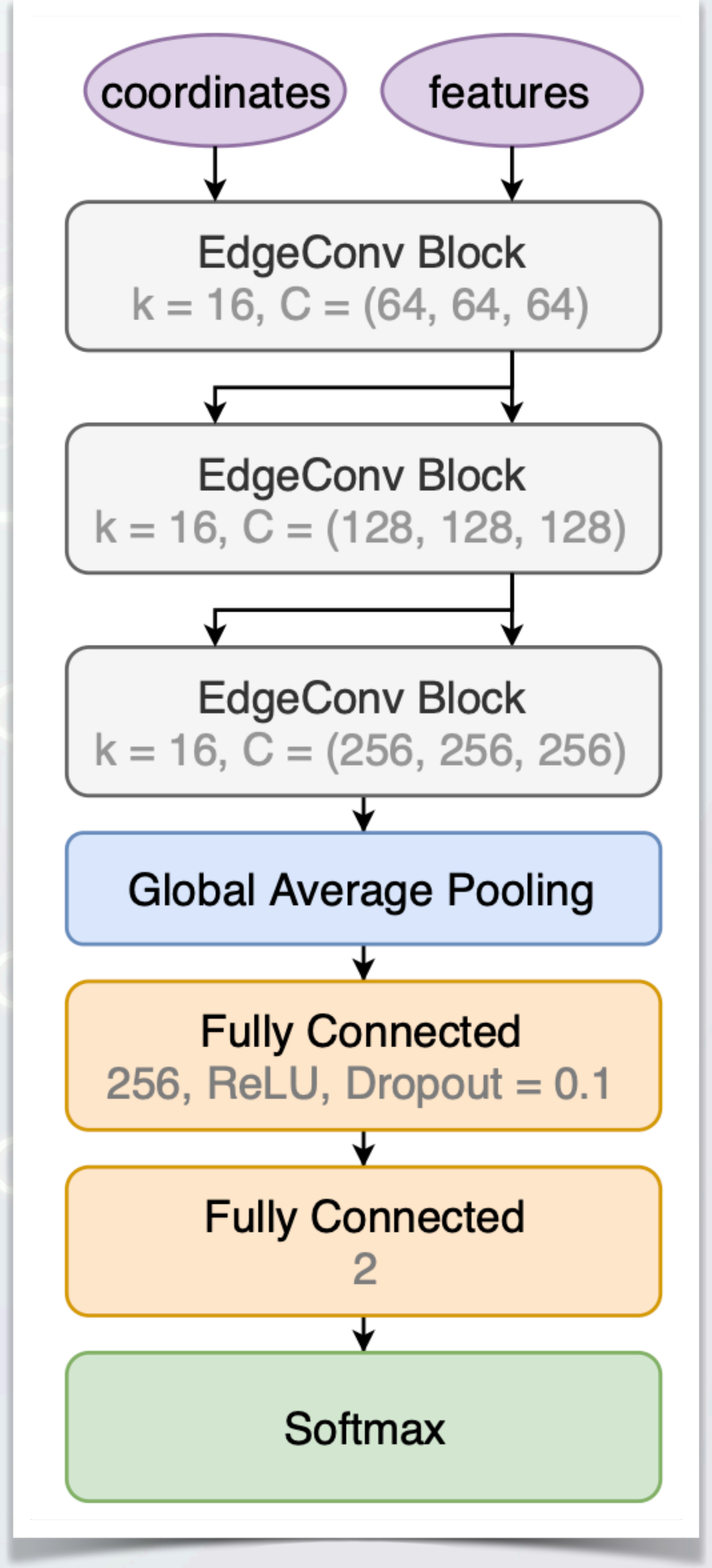

DNN (arXiv 1704.02124)



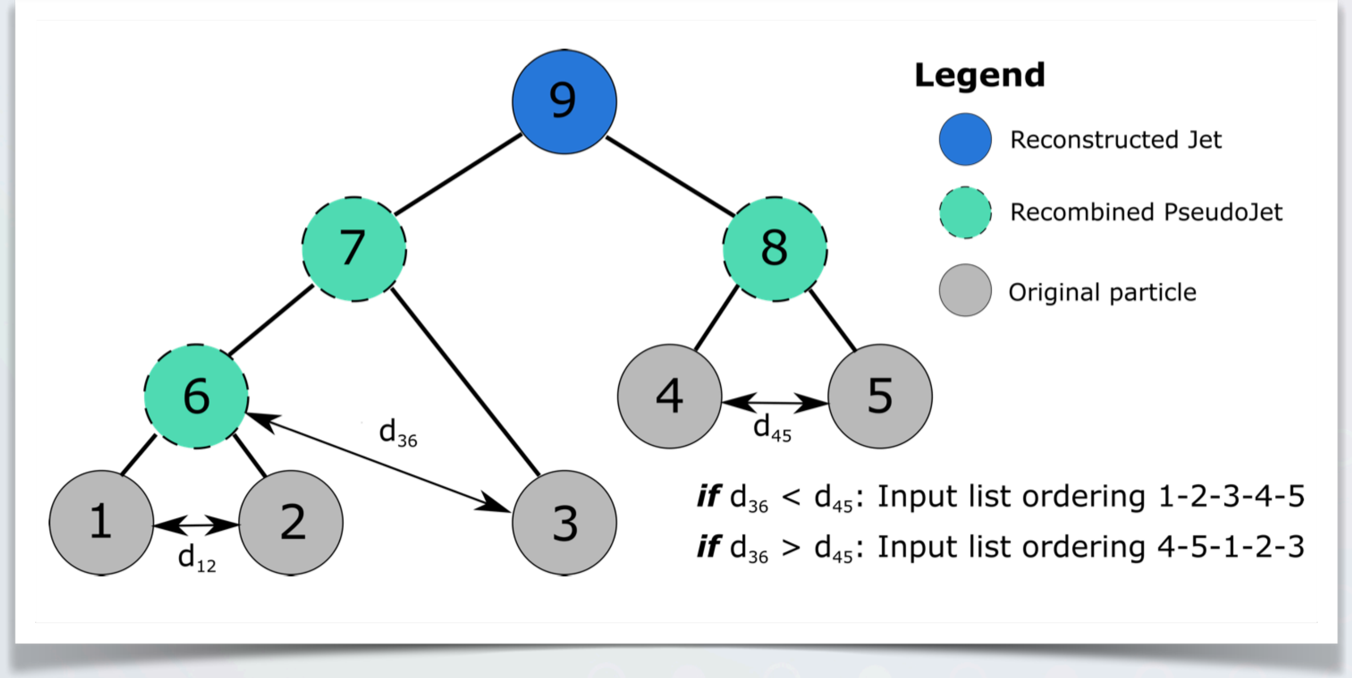

CNN (arXiv 1707.08966)



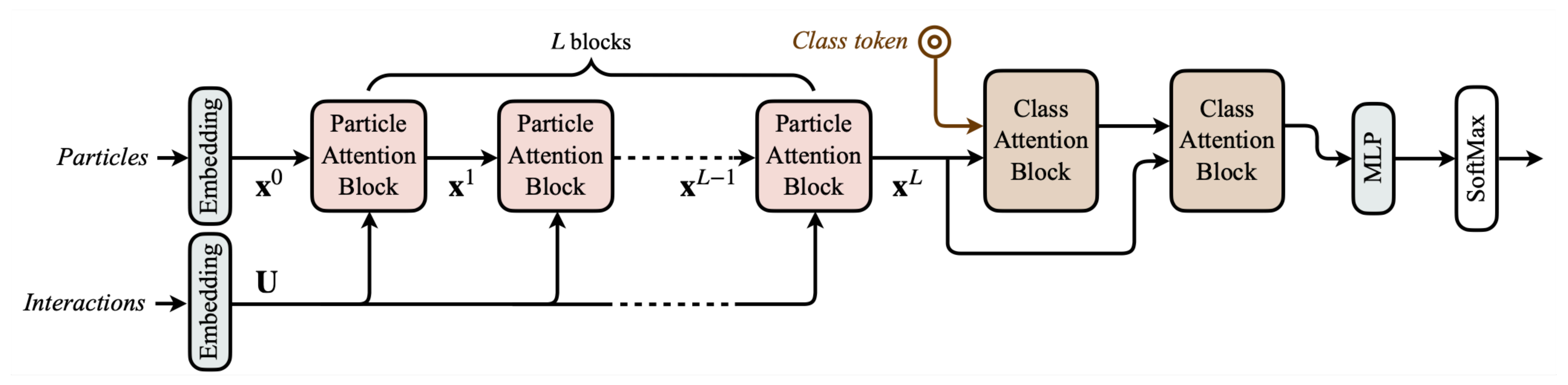

Deep Sets: EFN/PFN
(arXiv 1810.05165)




Particle Net
(arXiv 1902.08570)




LSTM (arXiv 1711.09059)




Particle Transformer (arXiv 2202.03772)

# Classical Machine Learning

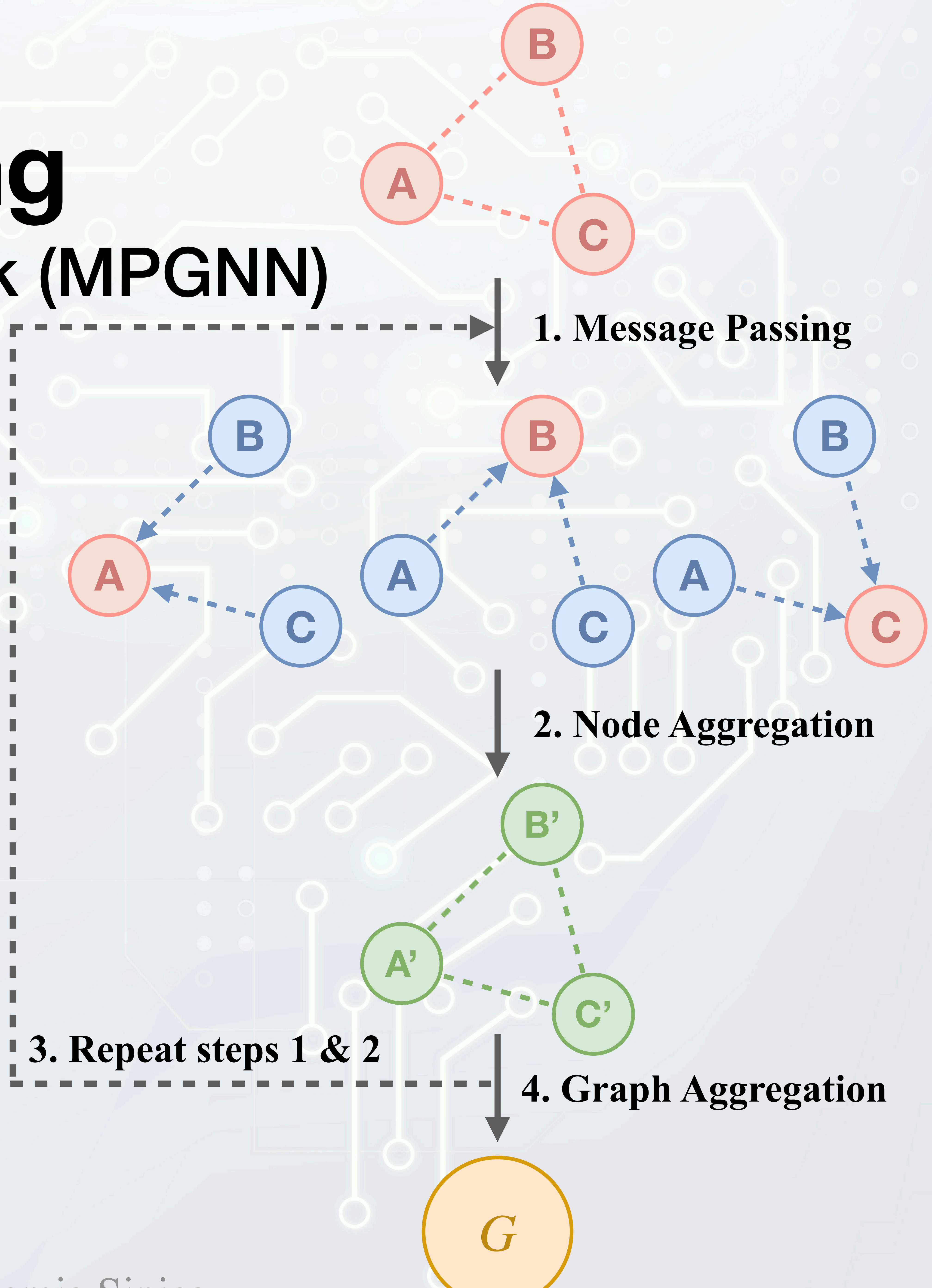
## Message-Passing Graph Neural Network (MPGNN)

1. Message Passing: Compute the information for each particle pair through some parametrized transformation.

2. Node Aggregation: Aggregate the transformed information for each particle. Typically element-wise **summation** $\Longleftrightarrow$ **permutation-invariant**

3. Repeat step 1 & 2 for several times (optional).

4. Graph Aggregation: Aggregate the information of all particles.



# Classical Machine Learning

## Deep Sets Theorem and MPGNN

- Deep Sets Theorem (arXiv 1703.06114) : A function (model) $f$ is permutation-invariant over a set $X$ (particles) if and only if $f(X) = g\left(\sum_{\mathbf{x}_i \in X} \boxed{h(\mathbf{x}_i)}\right)$ for some suitable transformations $g$ and $h$.

- The Message-Passing Graph Neural Network (MPGNN) obeys the Deep Sets Theorem, and is usually written as:

$$\mathbf{x}_i^{(k)} = \gamma^{(k)}\left(\mathbf{x}_i^{(k-1)}, \bigoplus_{j \in \mathcal{N}(i)} \boxed{\phi^{(k)}(\mathbf{x}_i^{(k-1)}, \mathbf{x}_j^{(k-1)}, \mathbf{e}_{ij})}\right)$$

Aggregation function (MEAN, SUM, MAX, etc.)

# QCGNN

## Quantum Complete Graph Neural Network

# Quantum Machine Learning

## Variational Quantum Circuit

# QCGNN

## Quantum Complete Graph Neural Network

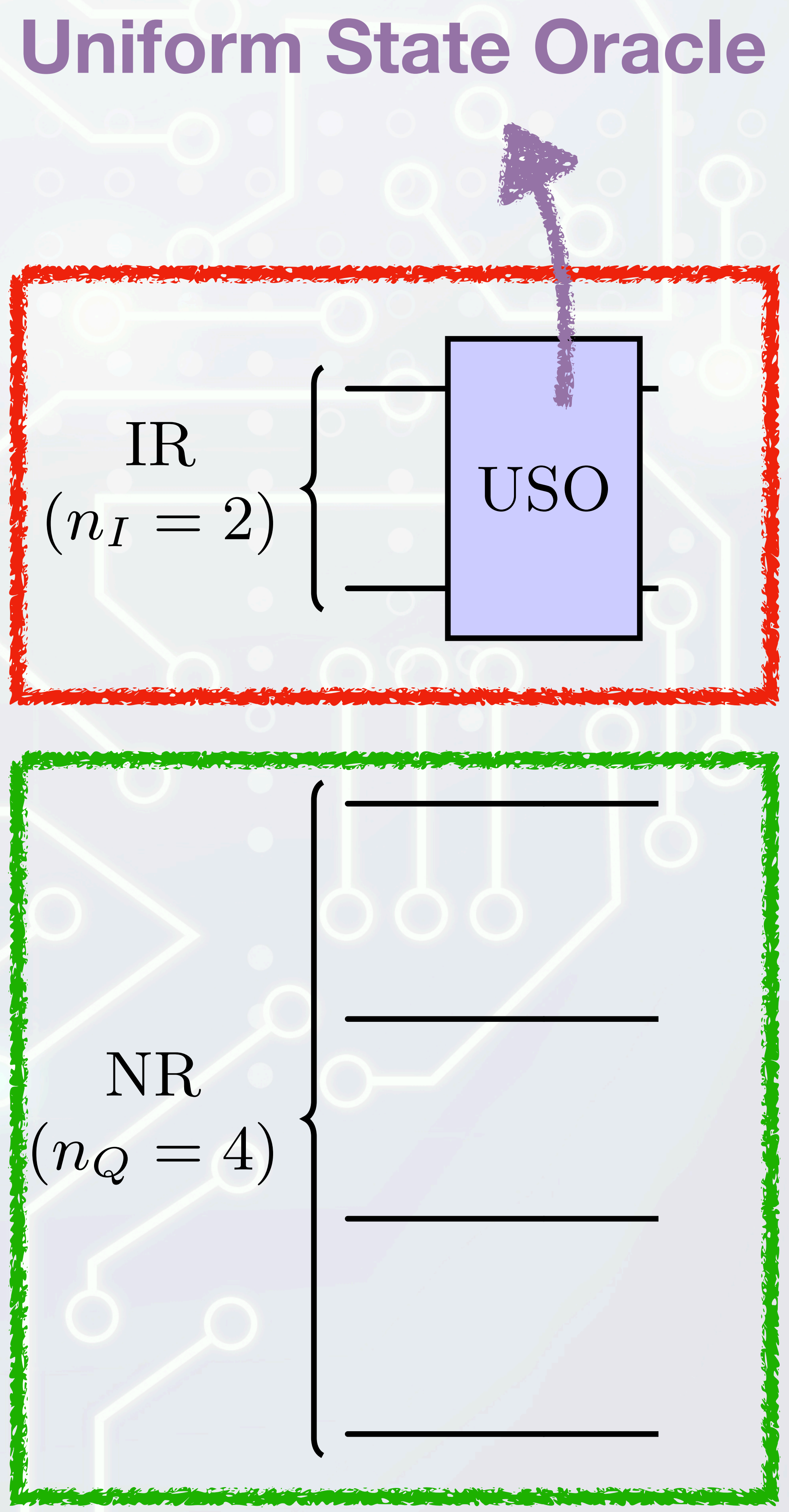
Suppose we have $N$ particles with features $\{\mathbf{x}_i \mid 0 \leq i \leq N-1\}$. We prepare a quantum circuit with $n_I + n_Q$ qubits where

- $n_I = \lceil \log_2 N \rceil$ is the number of qubits in the **index register (*IR*)**
- $n_Q$ is the number of qubits in the **network register (*NR*)**

**The initial quantum state is initialized as**

$$|\psi_0\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle |0\rangle^{\otimes n_Q}$$

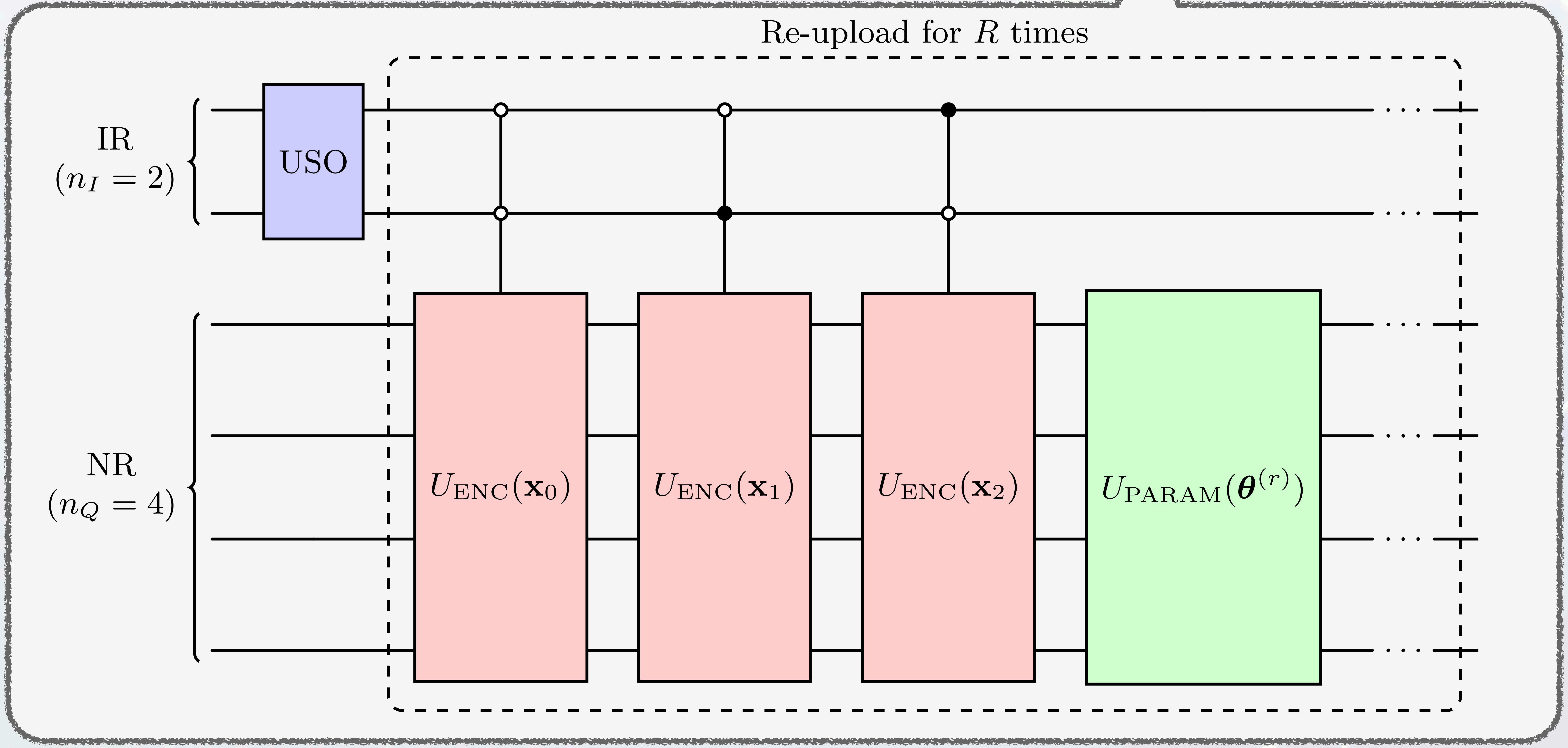
If $N = 2^{n_I}$, then we can simply use Hadamard gates. Otherwise, one should use some *Uniform State Oracle* (USO) to prepare the state.



# QCGNN

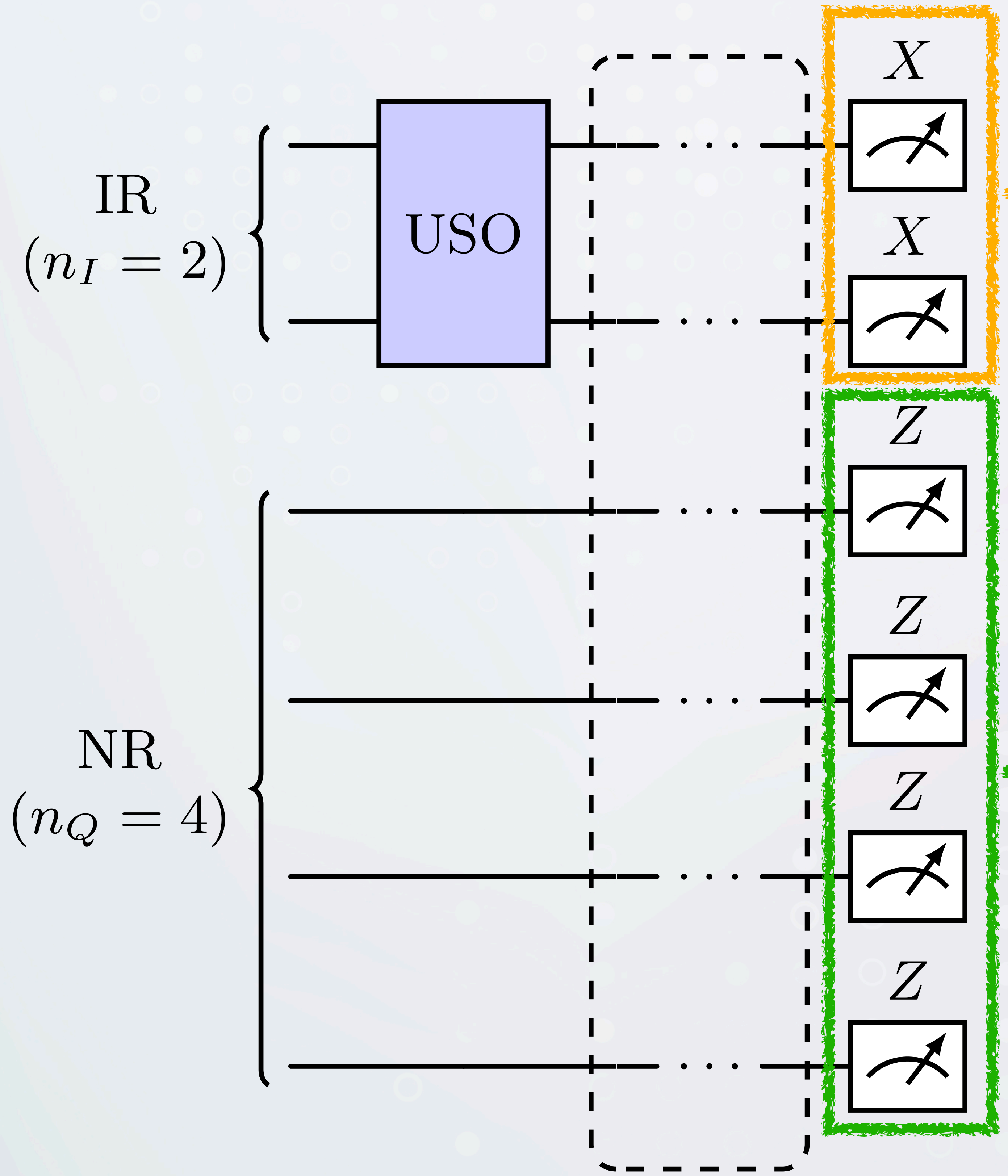
## Quantum Circuit

$$|\psi\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |\mathbf{x}_i, \theta\rangle$$




The QCGNN quantum circuit of a 3-particle jet, with $n_I = 2$ and $n_Q = 4$.

# QCGNN
## Measurement




The quantum state before measurement is $|\psi\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |\mathbf{x}_i, \theta\rangle$

Consider a Hermitian matrix $J$ with dimension $2^{n_I} \times 2^{n_I}$ full of ones, i.e.,

$$J = \begin{bmatrix} 1 & \dots & 1 \\ \vdots & & \vdots \\ 1 & \dots & 1 \end{bmatrix} = (I + X)^{\otimes n_I}.$$

Denote the observables on NR as P, then

$$\langle\psi|J \otimes \boxed{P}|\psi\rangle = \boxed{\frac{1}{N} \sum_{i<N} \sum_{j<N} \langle \mathbf{x}_i; \theta | P | \mathbf{x}_j; \theta \rangle}$$

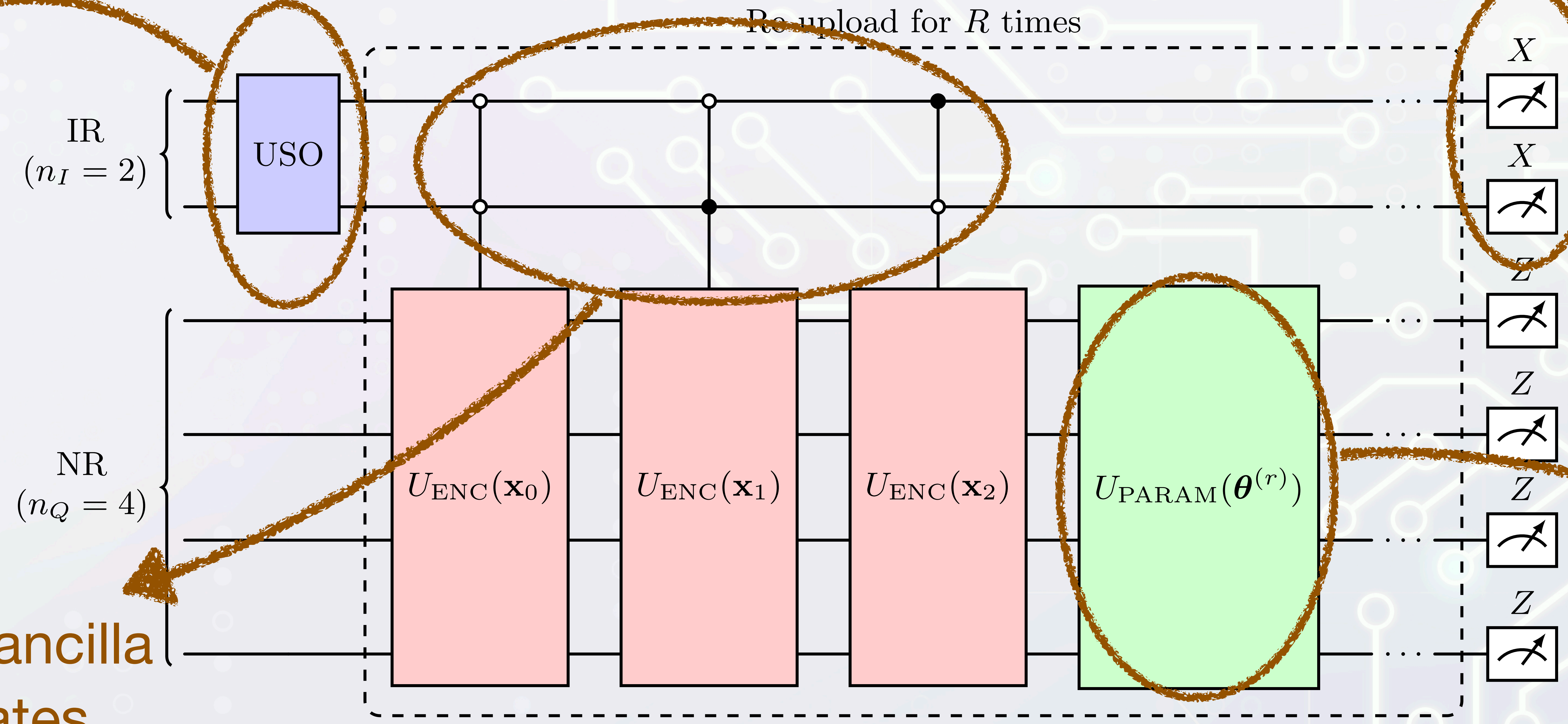
**Similar to MPGNN, with automatic aggregation.**

# QCGNN

## Gate and Computational Complexity

$O(N)$ Pauli-string observables

$O(\log_2 N)$ gates and circuit depth




Assuming Deep VQC

Additional $O(\log_2 N)$ ancilla qubits and Toffoli gates

When $N$ is large and the parametrized gates are deep comparing to encoding gates, **QCGNN only needs $O(N)$ computations**, while classical MPGNN requires $O(N^2)$ computations!

# Experiments Results

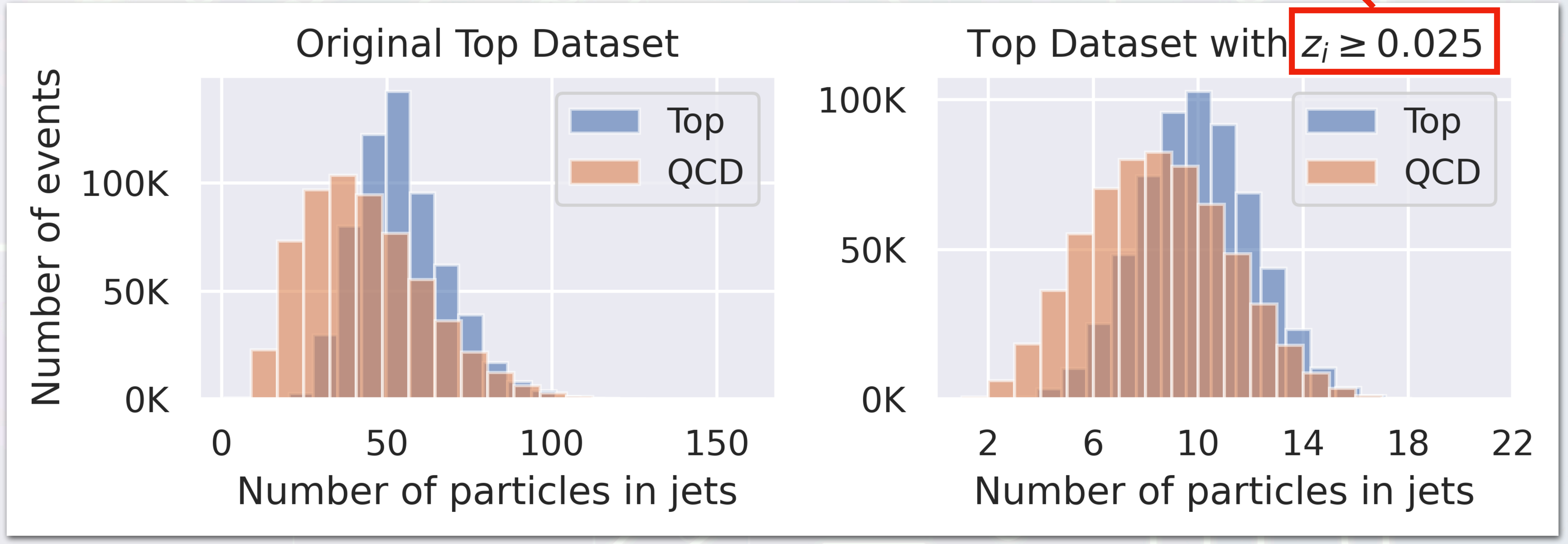
## Quantum Simulators & IBMQ


Jet Substructure (CMS 2022 Open Data Workshop)

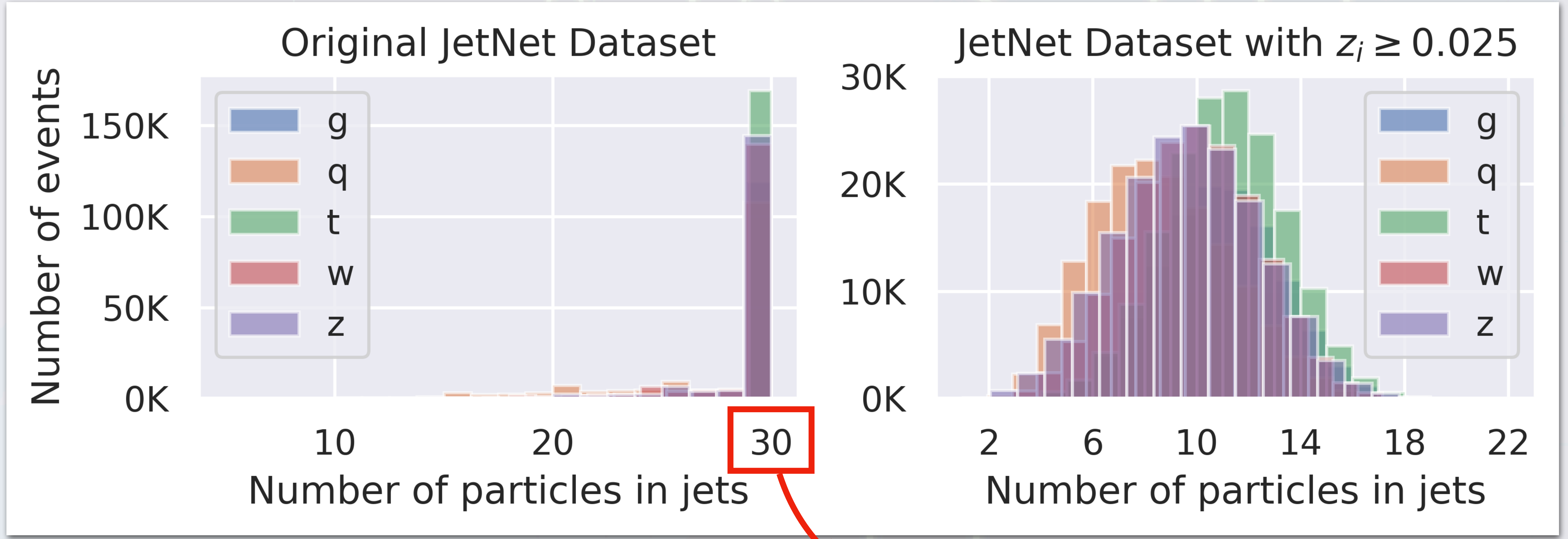
# Jet Dataset

$$z_i \equiv \frac{p_{T_i}}{p_{T_{jet}}}$$

- **Top-Taggers (arXiv 1902.09914)**
  - Top v.s. Gluon / Light Quarks
  - Jet $p_T \in [550{,}650]GeV$
  - Pythia / Delphes (ATLAS)
  - FastJet with $R = 0.8$




- **JetNet (arXiv 2106.11535)**
  - Multi-class $\{g, q, t, W, Z\}$
  - Jet $p_T \sim 1TeV$
  - MadGraph / Pythia
  - FastJet with $R = 0.8$




Only the 30 highest $p_T$ particles are provided

# Training Results
## AUC and Accuracy

- Each training process was conducted with 5 different random seeds and 30 epochs.
- Each class has 25K training samples, 2.5K validation samples, and 2.5K testing samples.
- The number of particles of jets lies between 4~16 $\Longrightarrow$ At most $n_I = 4$ qubits for IR is needed.
- The performance of the state-of-the-art classical models is also presented.

State-of-the-art classical models

Classical models for benchmarking

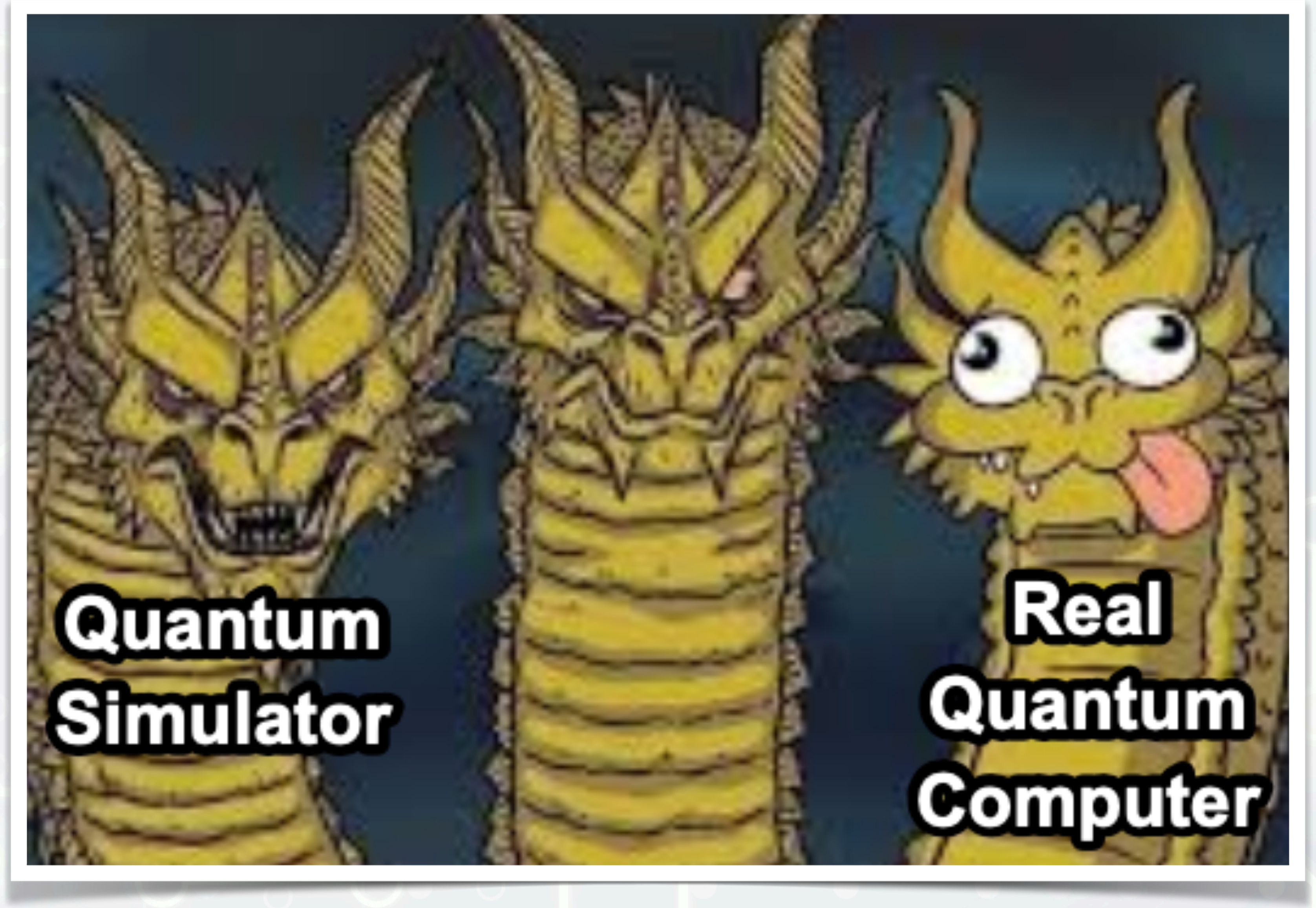
QCGNN with $n_Q = 3$ and $n_Q = 6$ (On simulators)

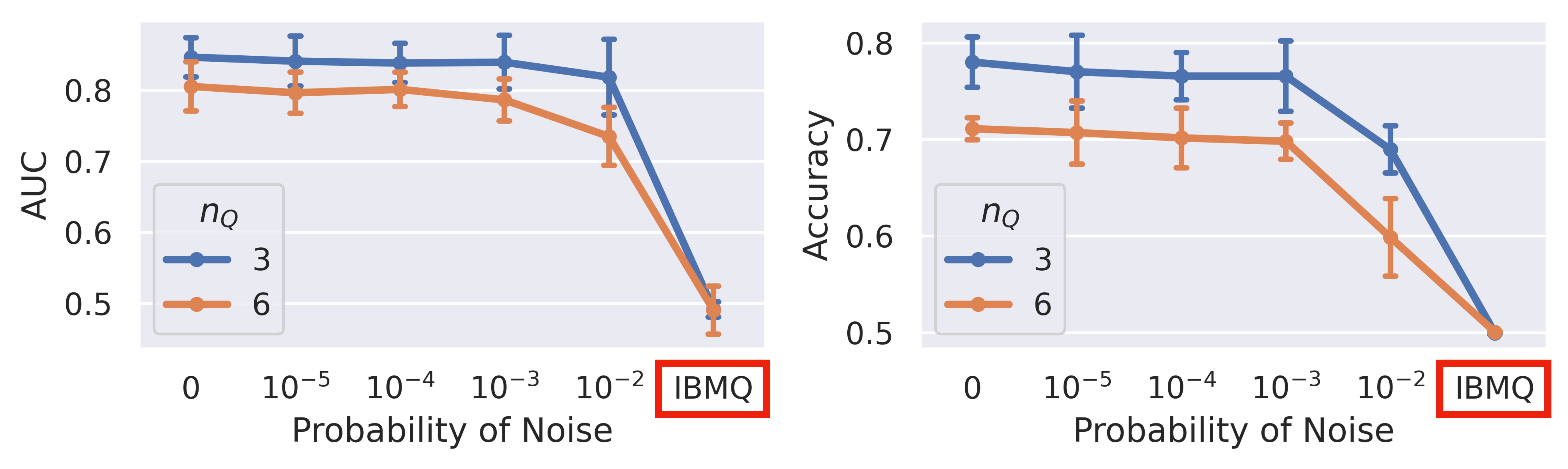
| Model | Top Dataset (2 classes) | | | JetNet Dataset (5 classes) | | |
|---|---|---|---|---|---|---|
| | # params | AUC | Accuracy | # params | AUC | Accuracy |
| Particle Transformer | 2.2M | 0.946±0.005 | 0.868±0.009 | 2.2M | 0.889±0.002 | 0.656±0.006 |
| Particle Net | 177K | 0.953±0.003 | 0.885±0.006 | 178K | 0.896±0.003 | 0.669±0.004 |
| Particle Flow Network | 72.3K | 0.954±0.004 | 0.885±0.005 | 72.7K | 0.900±0.003 | 0.675±0.005 |
| MPGNN - $n_M = 64$ | 13K | 0.961±0.003 | 0.896±0.003 | 13.3K | 0.903±0.002 | 0.683±0.007 |
| MPGNN - $n_M = 6$ | 255 | 0.924±0.006 | 0.866±0.006 | 323 | 0.865±0.004 | 0.615±0.010 |
| MPGNN - $n_M = 3$ | 126 | 0.922±0.005 | 0.864±0.006 | 194 | 0.757±0.110 | 0.475±0.141 |
| QCGNN - $n_Q = 6$ | 201 | 0.932±0.004 | 0.868±0.005 | 269 | 0.822±0.003 | 0.543±0.006 |
| QCGNN - $n_Q = 3$ | 99 | 0.919±0.006 | 0.864±0.005 | 167 | 0.796±0.009 | 0.505±0.014 |

# IBMQ Results

## Noise Extrapolation

- The training of QCGNN is done with simulators.
- The pre-trained QCGNN is tested on ***ibm_brussels***.



# IBMQ Results

## Runtime of Quantum Gates

- The gate runtime experiment is conducted with two different IBMQ backends for 10 times.
- $T_{ENC}$ and $T_{PARAM}$ are the time for encoding and parametrized gates respectively.

| IBMQ Backend | N | $T_{\text{ENC}}$ | $T_{\text{PARAM}}$ |
|---|---|---|---|
| ibm_nazca | 2 | 2.567 | 0.209 |
| | 4 | 5.352 | 0.197 |
| | 8 | 10.551 | 0.219 |
| ibm_strasbourg | 2 | 2.595 | 0.217 |
| | 4 | 5.416 | 0.197 |
| | 8 | 11.085 | 0.211 |

Scales as $O(N)$

Constant time $O(1)$

# Summary

- In the task of jet discrimination, graph is one of the natural representation. To design permutation-invariant models, graph neural networks have become a popular architecture.
- **We introduce a new quantum model, the Quantum Complete Graph Neural Network. If the parametrized gates are deep enough, the cost of QCGNN only scales as $O(N)$, while classical MPGNN requires $O(N^2)$.**
- QCGNN has also been tested on IBMQ real quantum devices. However, due to noise in the quantum circuits, information transmission was unsuccessful.
- As the quantum computers becoming more robust in the future, the potential for quantum advantage of the QCGNN can be studied.

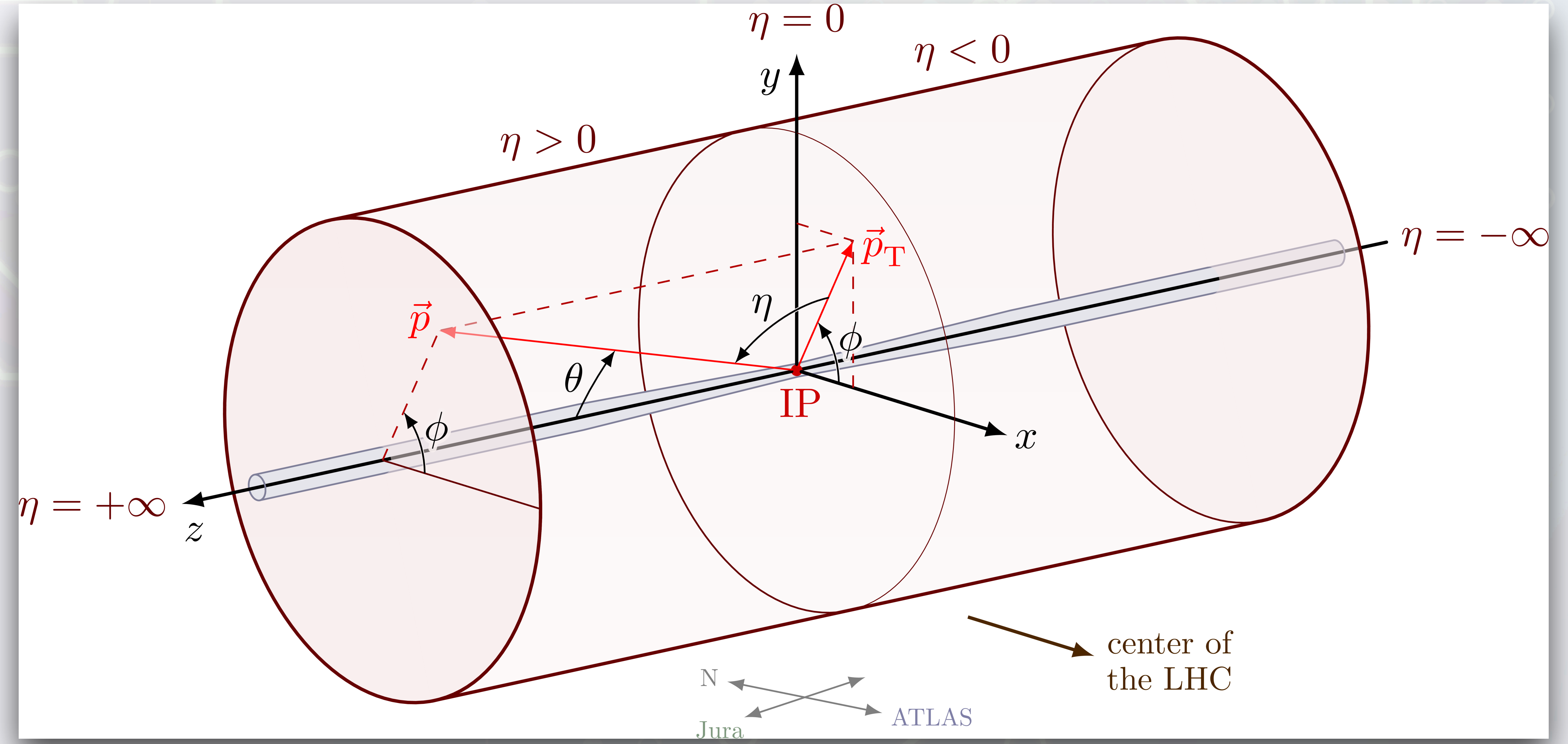

Thanks for your attention!
Hypergeometric functions
functions
Special functions
contour
Contour fincctions
Conctal
Contour

# Backup Slides

# CMS Coordinate System

The particle flow is defined as

- The transverse momentum $p_T = \sqrt{p_x^2 + p_y^2}$
- The azimuthal angle $\phi = \tan^{-1}(\frac{p_y}{p_x})$
- The pseudo-rapidity $\eta = -\ln\left[\tan(\frac{\theta}{2})\right]$

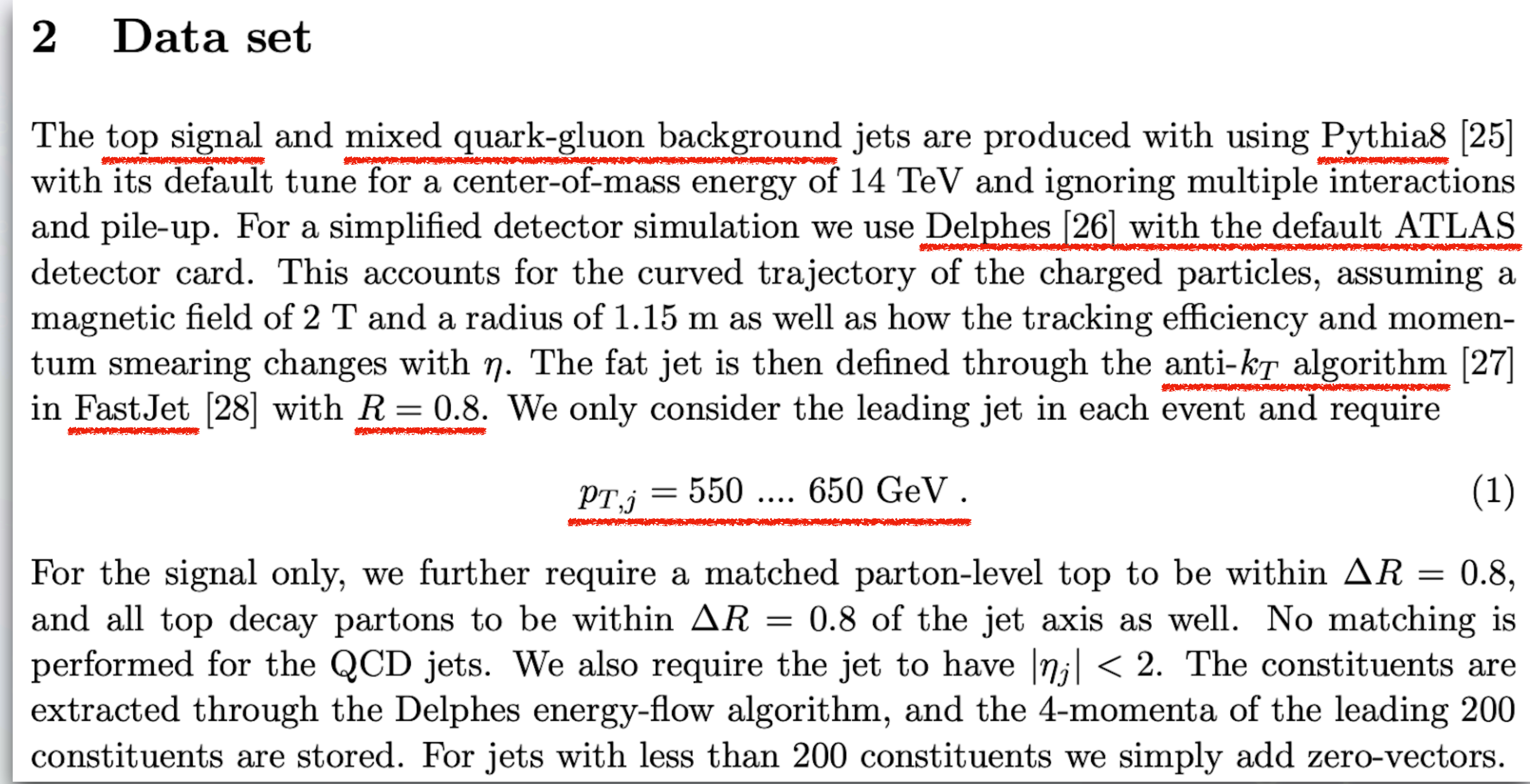


CMS Coordinate System by Izaak Neutelings

In jet analysis, the differences $\Delta\phi$ and $\Delta\eta$ relative to the jets is adopted, since they are Lorentz invariant under boosts in $z$-direction.

# Dataset

## Top Tagging (arXiv 1902.09914)



### 2 Data set

The top signal and mixed quark-gluon background jets are produced with using Pythia8 [25] with its default tune for a center-of-mass energy of 14 TeV and ignoring multiple interactions and pile-up. For a simplified detector simulation we use Delphes [26] with the default ATLAS detector card. This accounts for the curved trajectory of the charged particles, assuming a magnetic field of 2 T and a radius of 1.15 m as well as how the tracking efficiency and momentum smearing changes with $\eta$. The fat jet is then defined through the anti-$k_T$ algorithm [27] in FastJet [28] with $R = 0.8$. We only consider the leading jet in each event and require

$$p_{T,j} = 550 \; .... \; 650 \text{ GeV} \; . \tag{1}$$

For the signal only, we further require a matched parton-level top to be within $\Delta R = 0.8$, and all top decay partons to be within $\Delta R = 0.8$ of the jet axis as well. No matching is performed for the QCD jets. We also require the jet to have $|\eta_j| < 2$. The constituents are extracted through the Delphes energy-flow algorithm, and the 4-momenta of the leading 200 constituents are stored. For jets with less than 200 constituents we simply add zero-vectors.

# Dataset

## JetNet (arXiv 2106.11535)

### B JetNet Generation

The so-called parton-level events are first produced at leading-order using MADGRAPH5_aMCATNLO 2.3.1 [51] with the NNPDF 2.3LO1 parton distribution functions [52]. To focus on a relatively narrow kinematic range, the transverse momenta of the partons and undecayed gauge bosons are generated in a window with energy spread given by $\Delta p_{\mathrm{T}}/p_{\mathrm{T}} = 0.01$, centered at 1 TeV. These parton-level events are then decayed and showered in PYTHIA 8.212 [5] with the Monash 2013 tune [53], including the contribution from the underlying event. For each original particle type, 200,000 events are generated. Jets are clustered using the anti-$k_{\mathrm{T}}$ algorithm [54], with a distance parameter of $R = 0.8$ using the FASTJET 3.1.3 and FASTJET CONTRIB 1.027 packages [55, 56]. Even though the parton-level $p_{\mathrm{T}}$ distribution is narrow, the jet $p_{\mathrm{T}}$ spectrum is significantly broadened by kinematic recoil from the parton shower and energy migration in and out of the jet cone. We apply a restriction on the measured jet $p_{\mathrm{T}}$ to remove extreme events outside of a window of $0.8\,\mathrm{TeV} < p_{\mathrm{T}} < 1.6\,\mathrm{TeV}$ for the $p_{\mathrm{T}} = 1\,\mathrm{TeV}$ bin. This generation is a significantly simplified version of the official simulation and reconstruction steps used for real detectors at the LHC, so as to remain experiment-independent and allow public access to the dataset.

# Classical ML for Jets

## Message Passing GNN

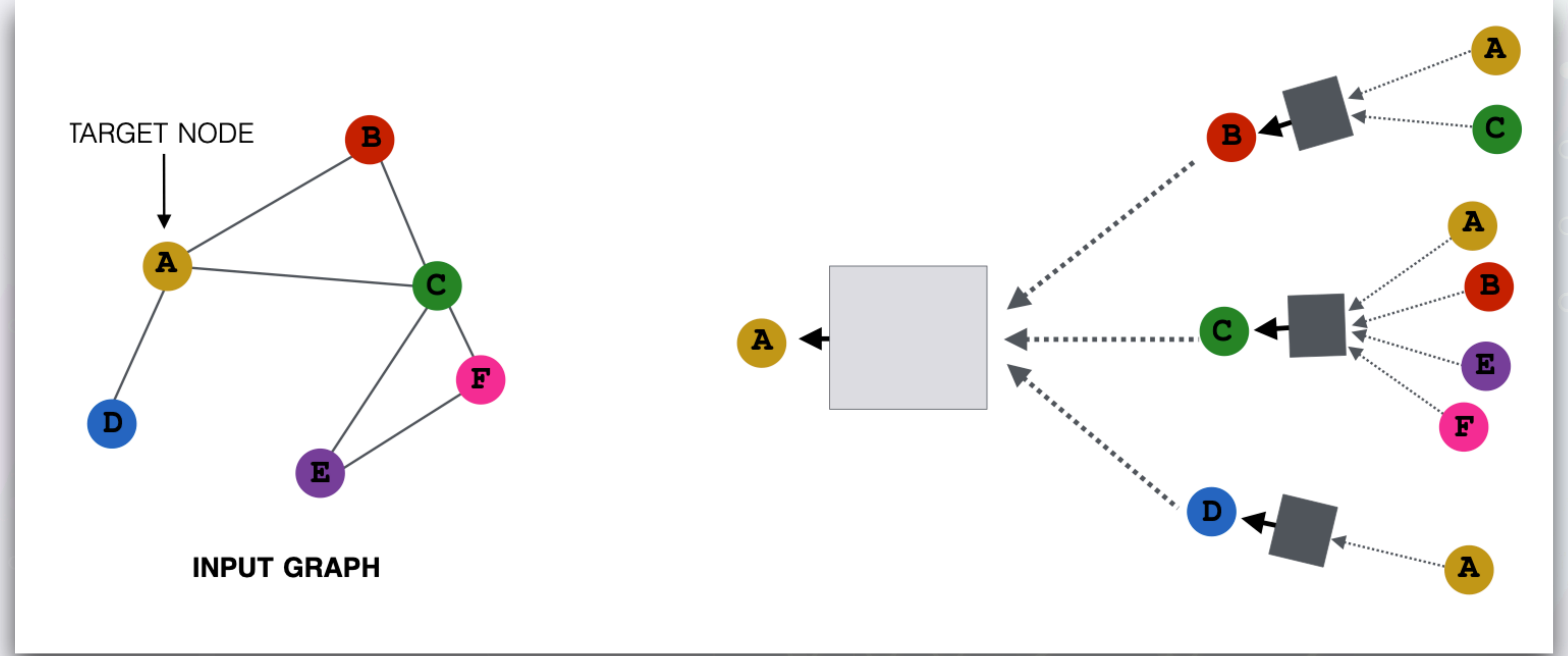



Figure adapted from "Stanford CS224W: Machine Learning with Graphs" by J.Leskovec.

$$f(X) = g\left(\sum_{\vec{x}_i \in X} \boxed{h(\vec{x}_i)}\right)$$

$$\vec{x}_i^{(k)} = \gamma^{(k)}\left(\vec{x}_i^{(k-1)}, \bigoplus_{j \in \mathcal{N}(i)} \boxed{\phi^{(k)}\left(\vec{x}_i^{(k-1)}, \vec{x}_j^{(k-1)}, \mathbf{e}_{ij}\right)}\right)$$

$k$-th update

Aggregation function (MEAN, SUM, MAX, etc.)

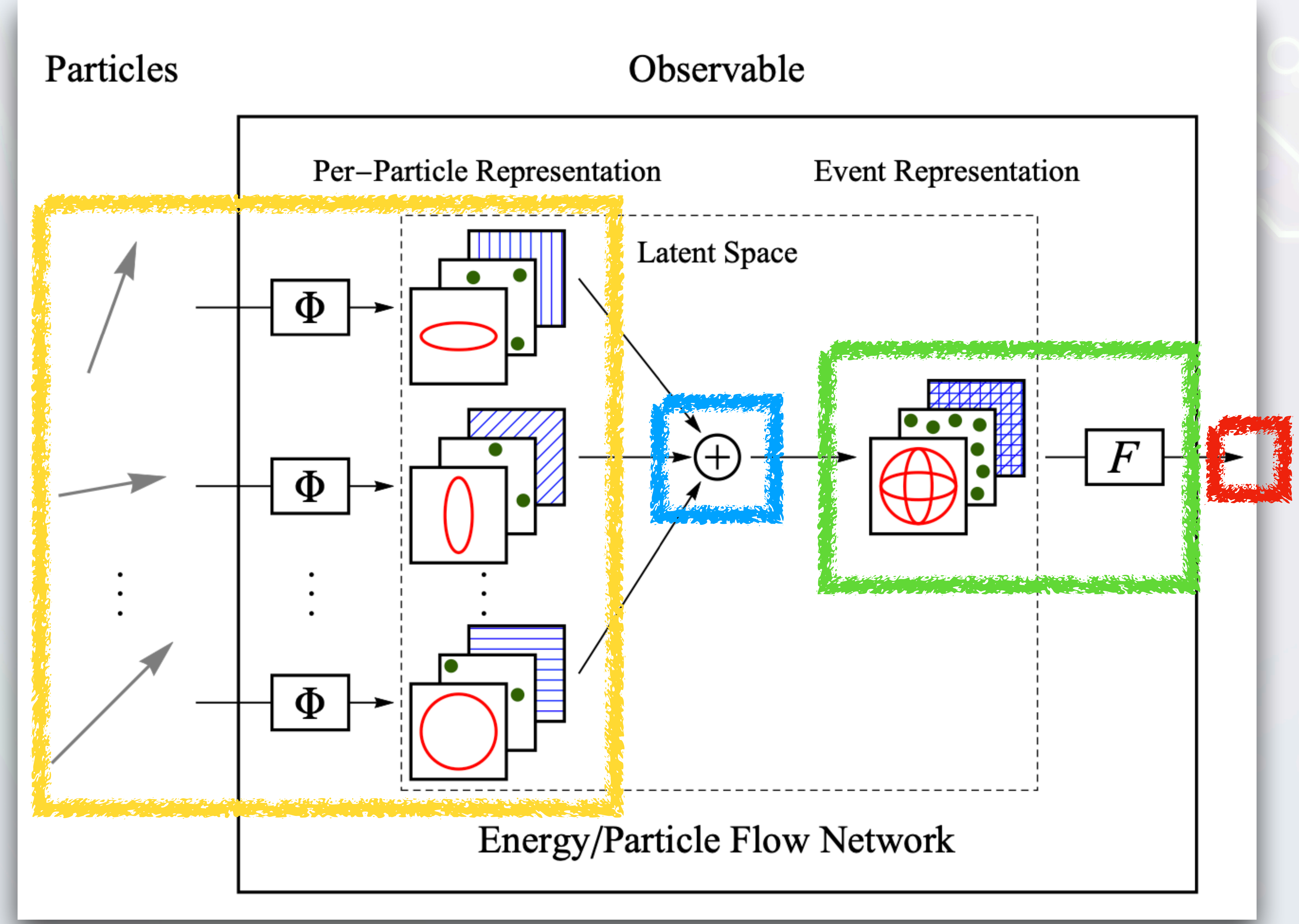
- GCN (arXiv 1609.02907)
- GraphSAGE (arXiv 1706.02216)
- GAT (arXiv 1710.10903)
- EdgeConv (arXiv 1801.07829)

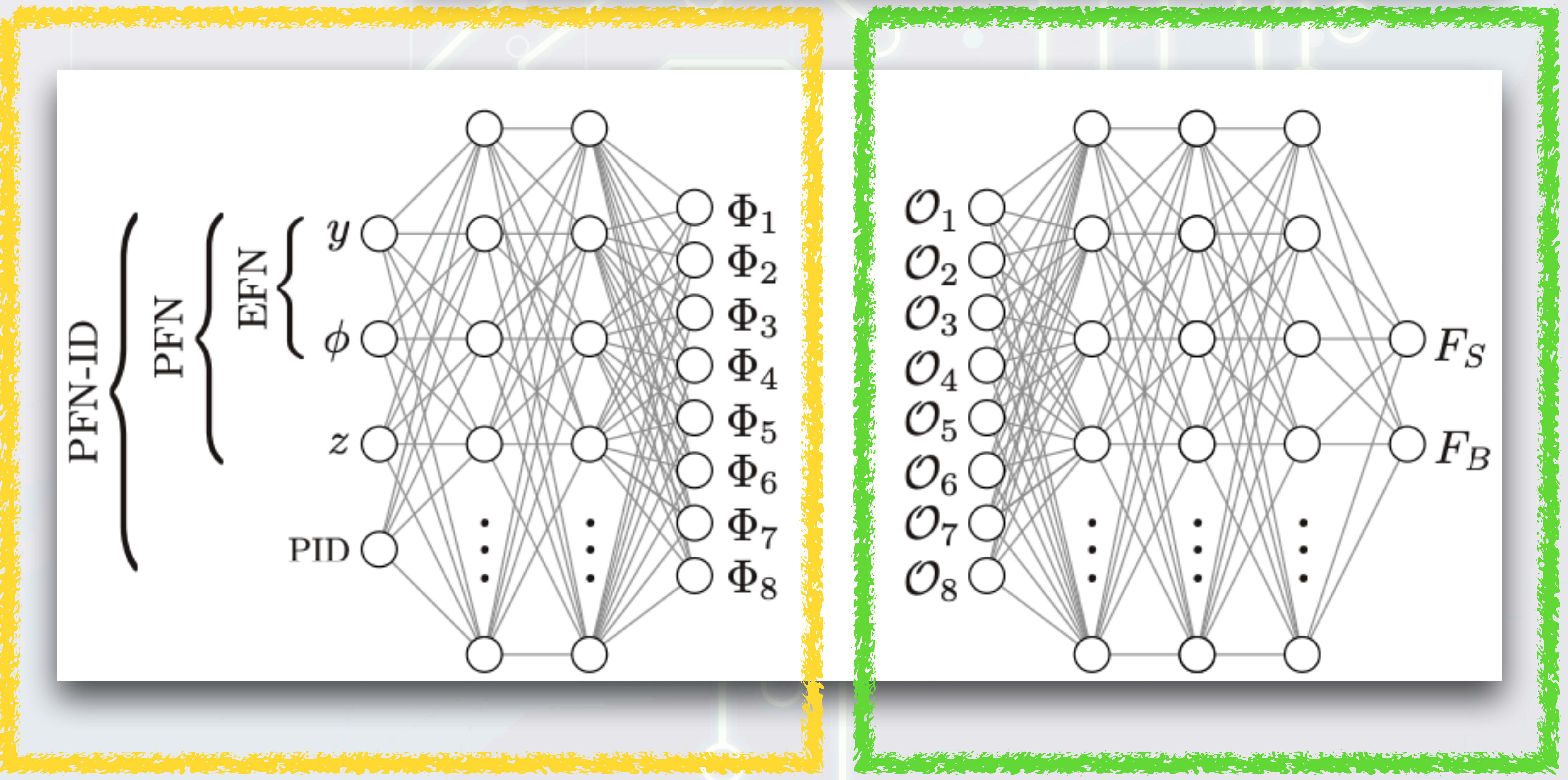
# Particle Flow Network

## arXiv 1810.05165

Motivated by the *Deep Set Theorem*

$$\mathcal{O}(\{p_1, \ldots, p_M\}) = F\left(\sum_{i=1}^{M} \Phi(p_i)\right)$$



# Single Qubit Gates

## Rotation Gates

*Theorem 4.1*: (**$Z$-$Y$ decomposition for a single qubit**) Suppose $U$ is a unitary operation on a single qubit. Then there exist real numbers $\alpha, \beta, \gamma$ and $\delta$ such that

$$U = e^{i\alpha} R_z(\beta) R_y(\gamma) R_z(\delta). \tag{4.11}$$

$$R_x(\theta) \equiv e^{-i\theta X/2} = \cos\frac{\theta}{2} I - i \sin\frac{\theta}{2} X = \begin{bmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix} \tag{4.4}$$

$$R_y(\theta) \equiv e^{-i\theta Y/2} = \cos\frac{\theta}{2} I - i \sin\frac{\theta}{2} Y = \begin{bmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix} \tag{4.5}$$

$$R_z(\theta) \equiv e^{-i\theta Z/2} = \cos\frac{\theta}{2} I - i \sin\frac{\theta}{2} Z = \begin{bmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{bmatrix}. \tag{4.6}$$

# Multi-Qubit Gates

## Decomposition of Multi-Controlled Gates

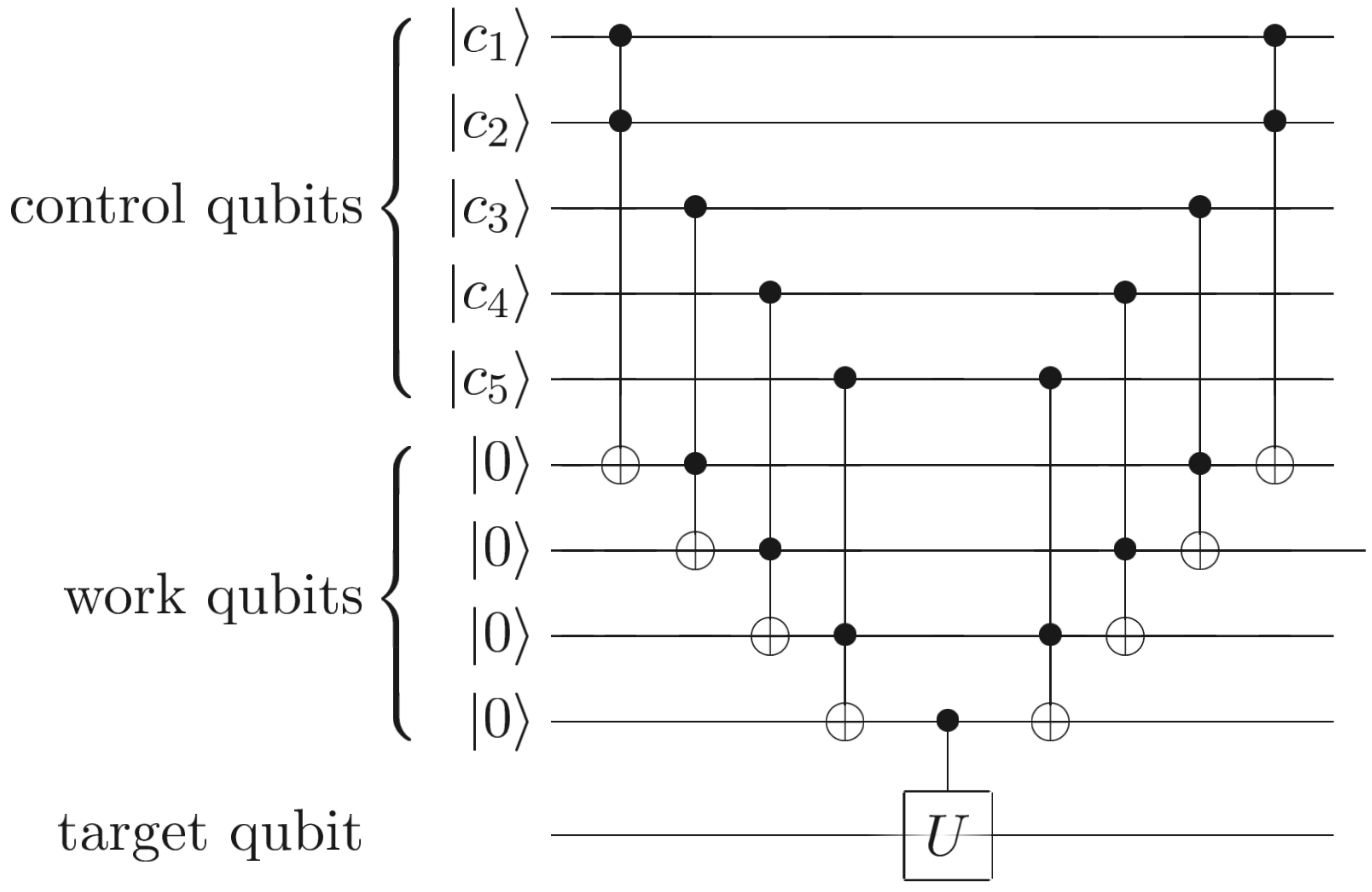



Figure 4.10. Network implementing the $C^n(U)$ operation, for the case $n = 5$.

Adopted from "Quantum Computation and Quantum Information" by N & C.

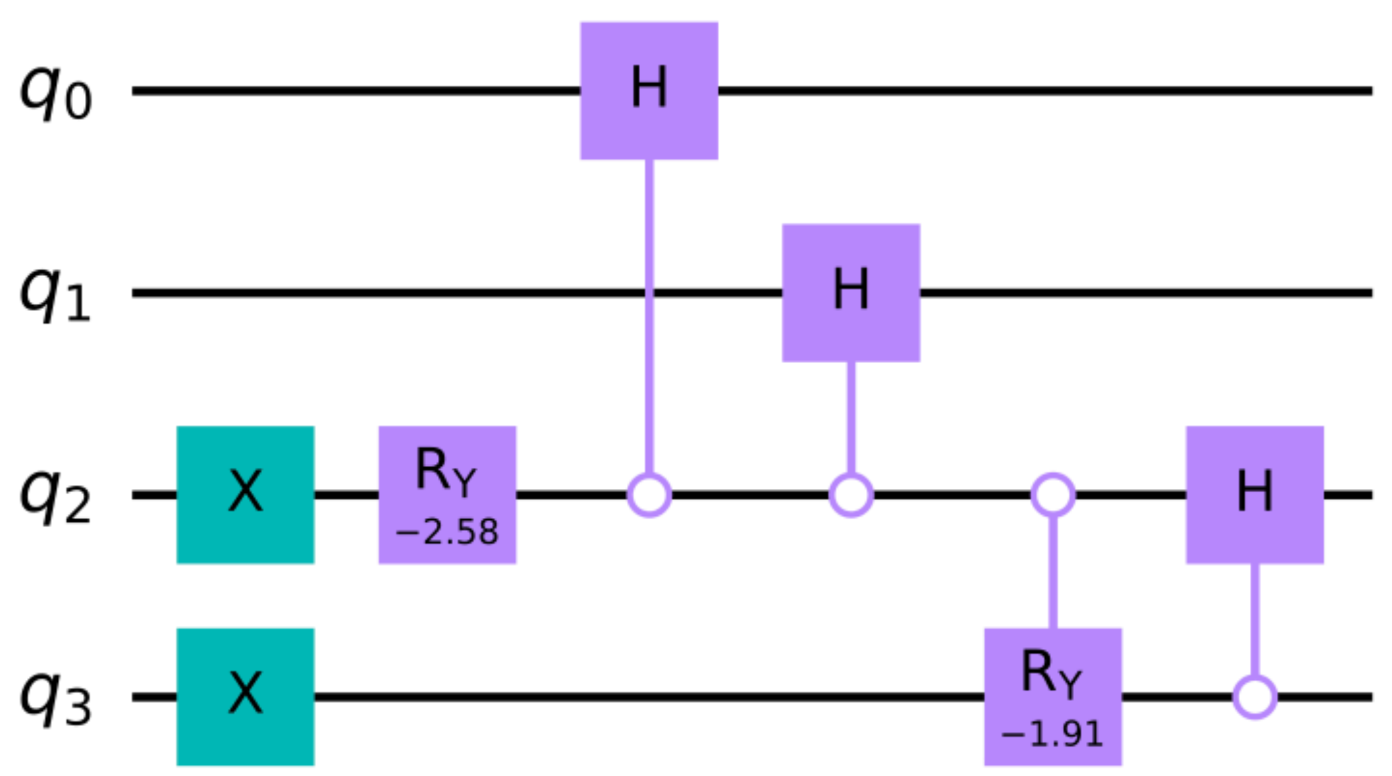
# Uniform State Oracle

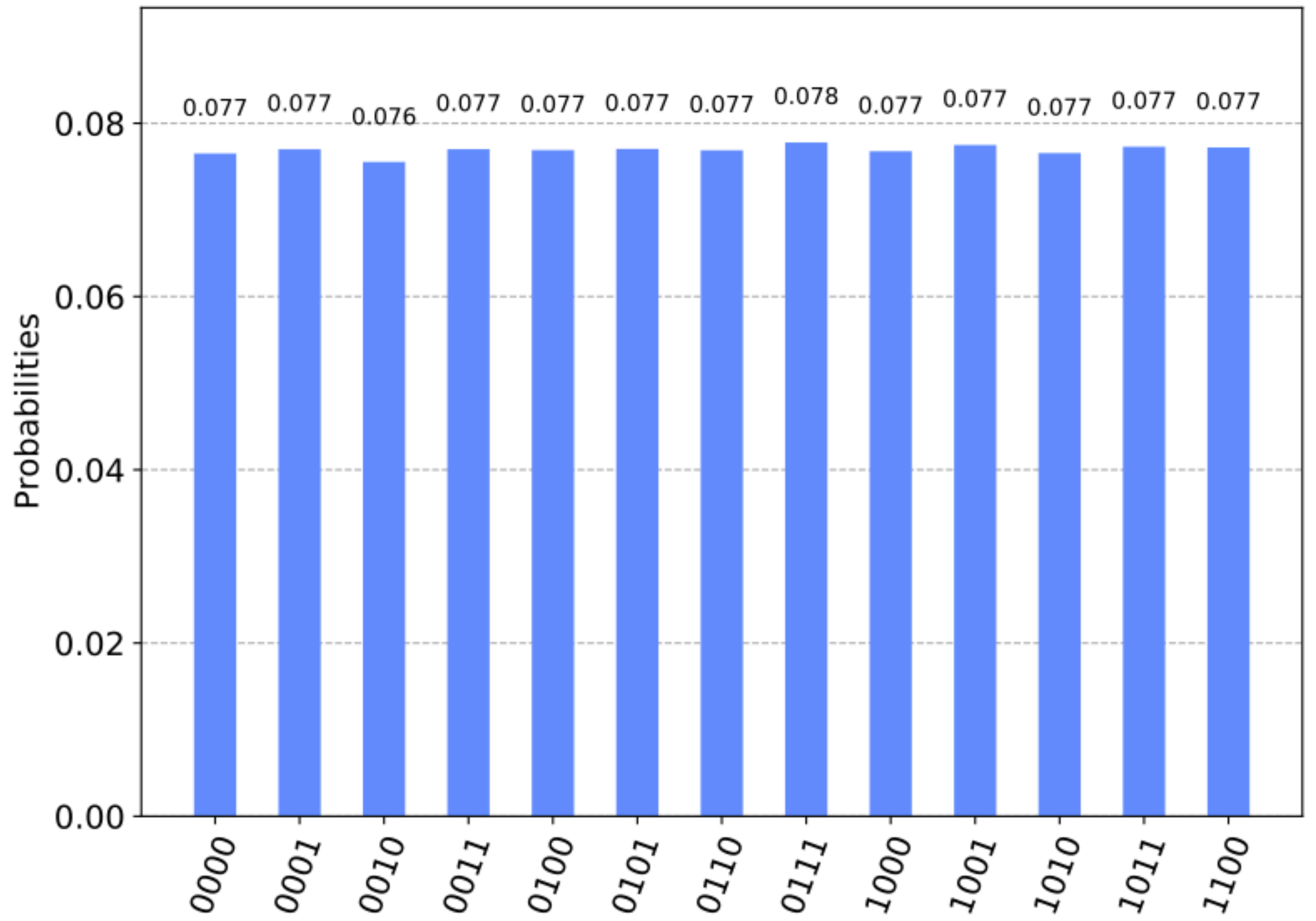
## arXiv 2306.11747

In this paper, we propose an efficient approach for quantum state preparation of uniform superposition state $|\Psi\rangle = \frac{1}{\sqrt{M}} \sum_{j=0}^{M-1} |j\rangle$ that offers a significant (exponential) reduction in gate complexity and circuit depth without the use of ancillary qubits. We show that using only $n = \lceil \log_2 M \rceil$ qubits, the uniform superposition state $|\Psi\rangle$ can be prepared for arbitrary $M$ with a gate complexity and circuit depth of $O(\log_2 M)$.

Quantum circuit for generating a 13-basis uniform state:

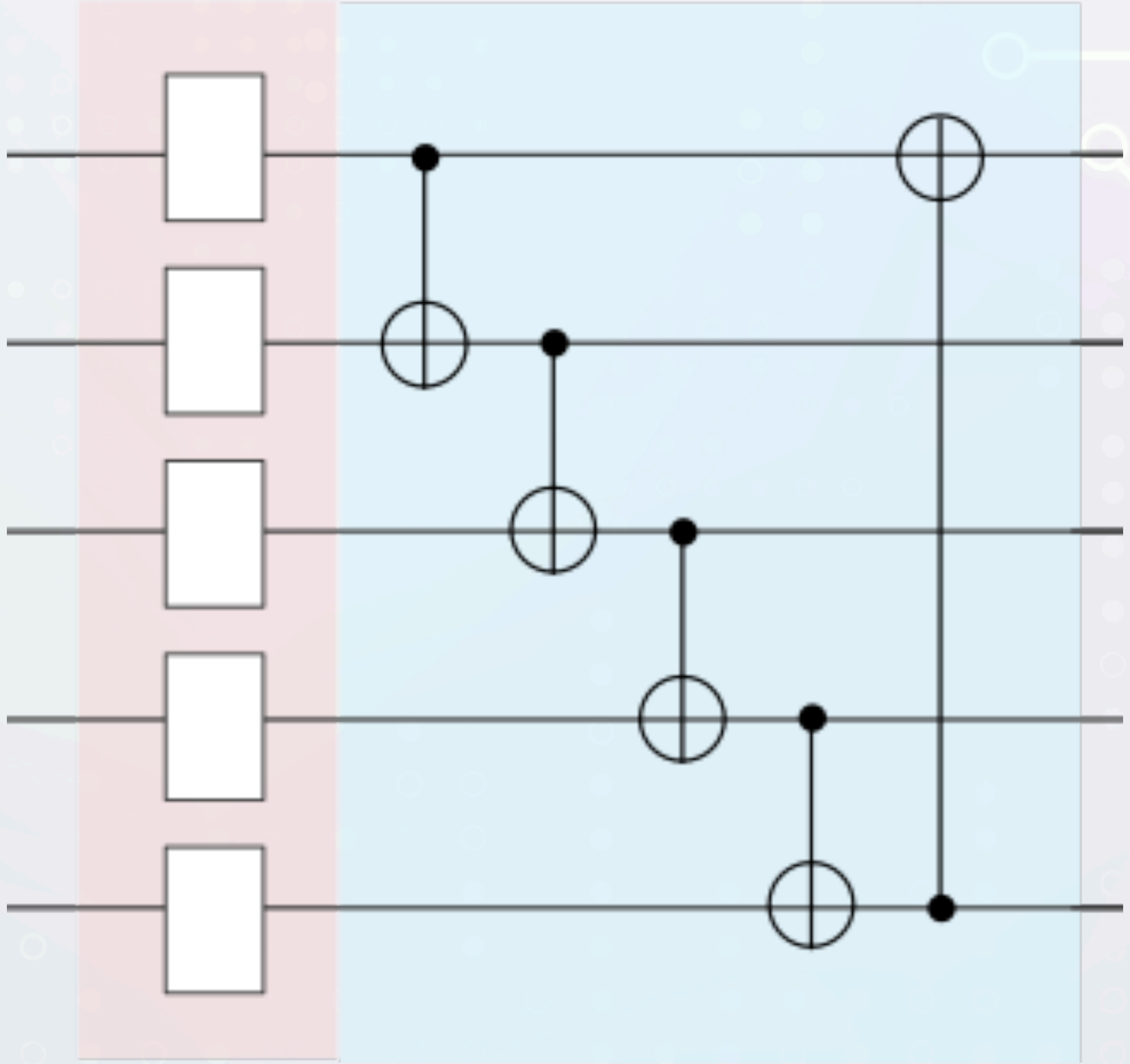
$$|\psi\rangle = \frac{1}{\sqrt{13}} \sum_{i=0}^{12} |i\rangle$$



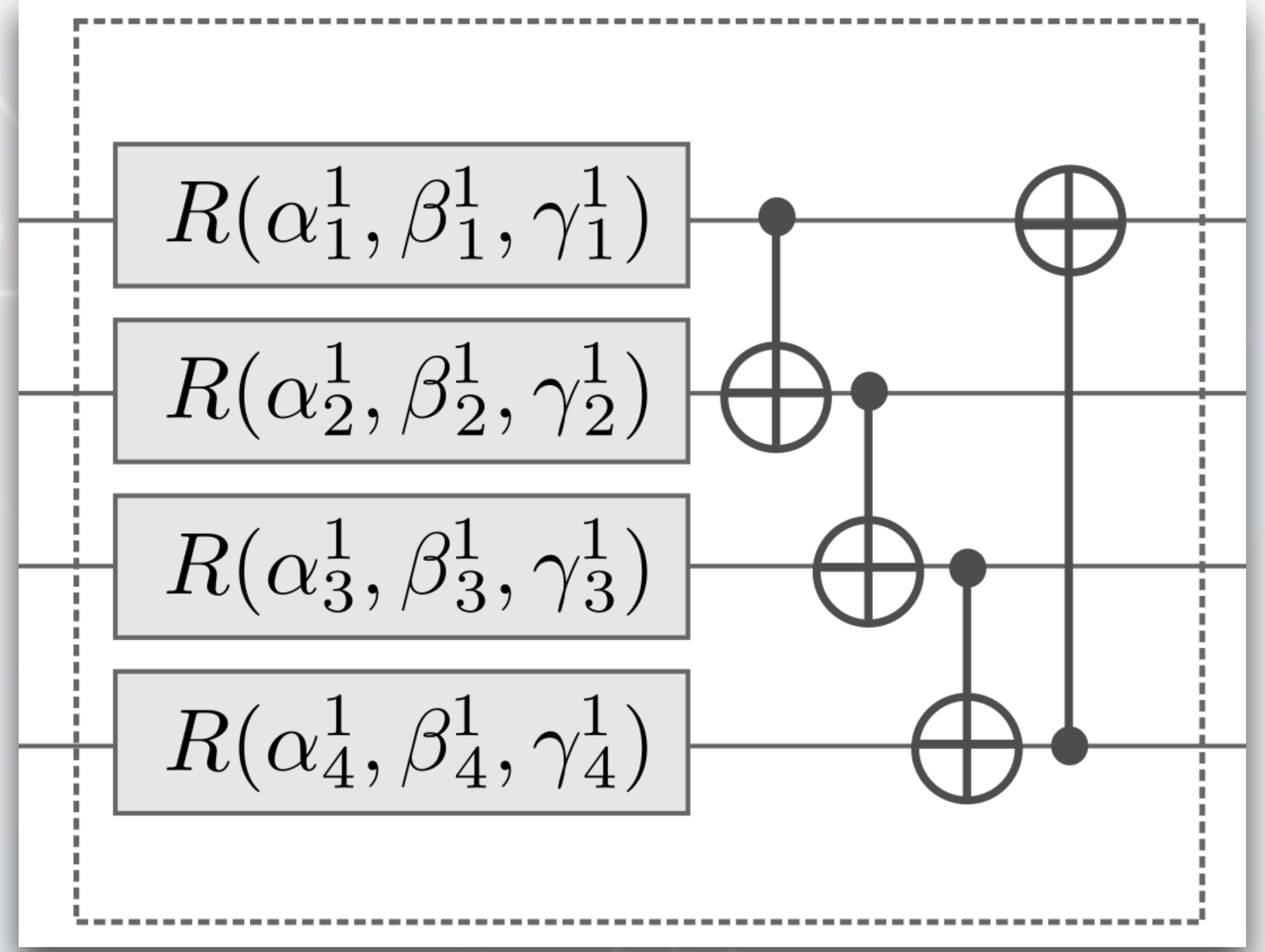



# VQC Ansatz

## PennyLane

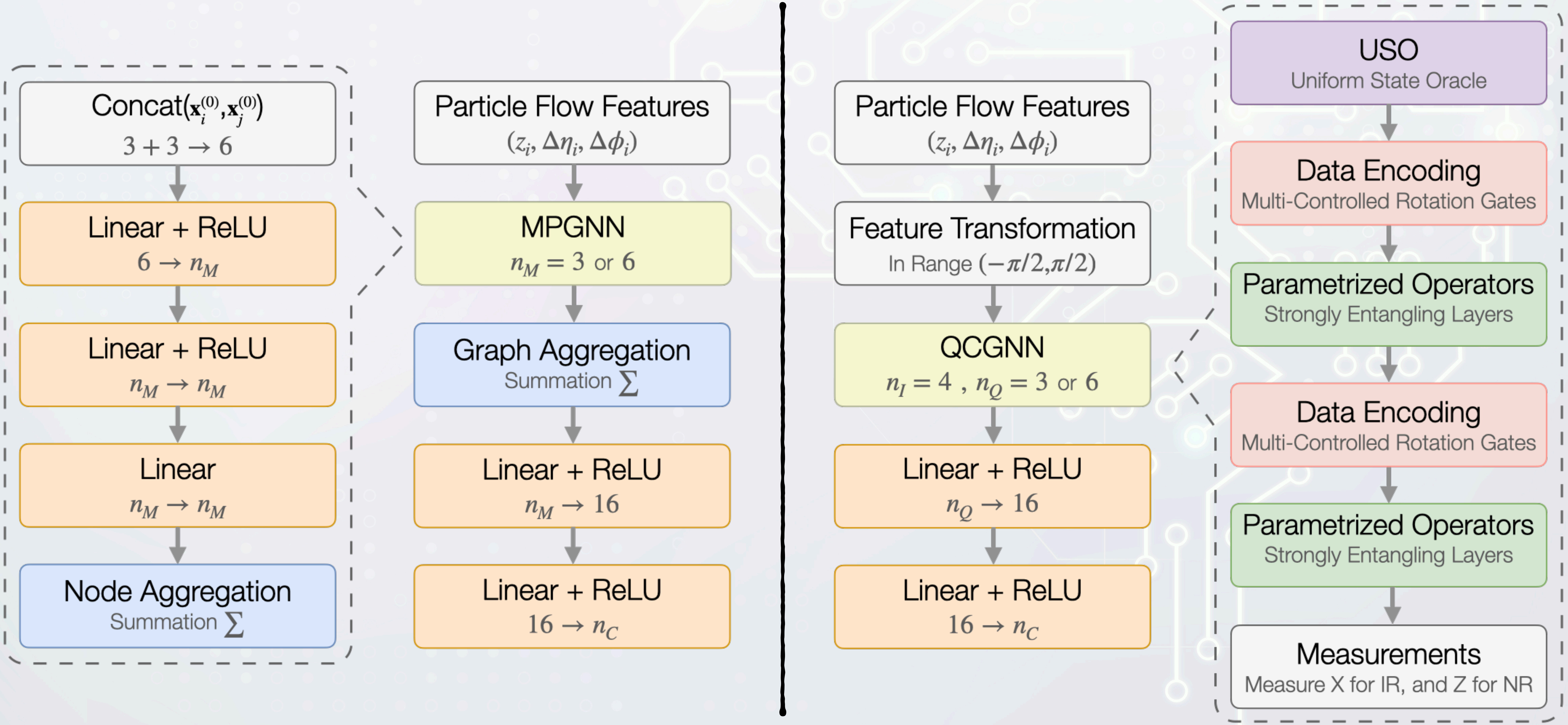


qml.BasicEntanglerLayers



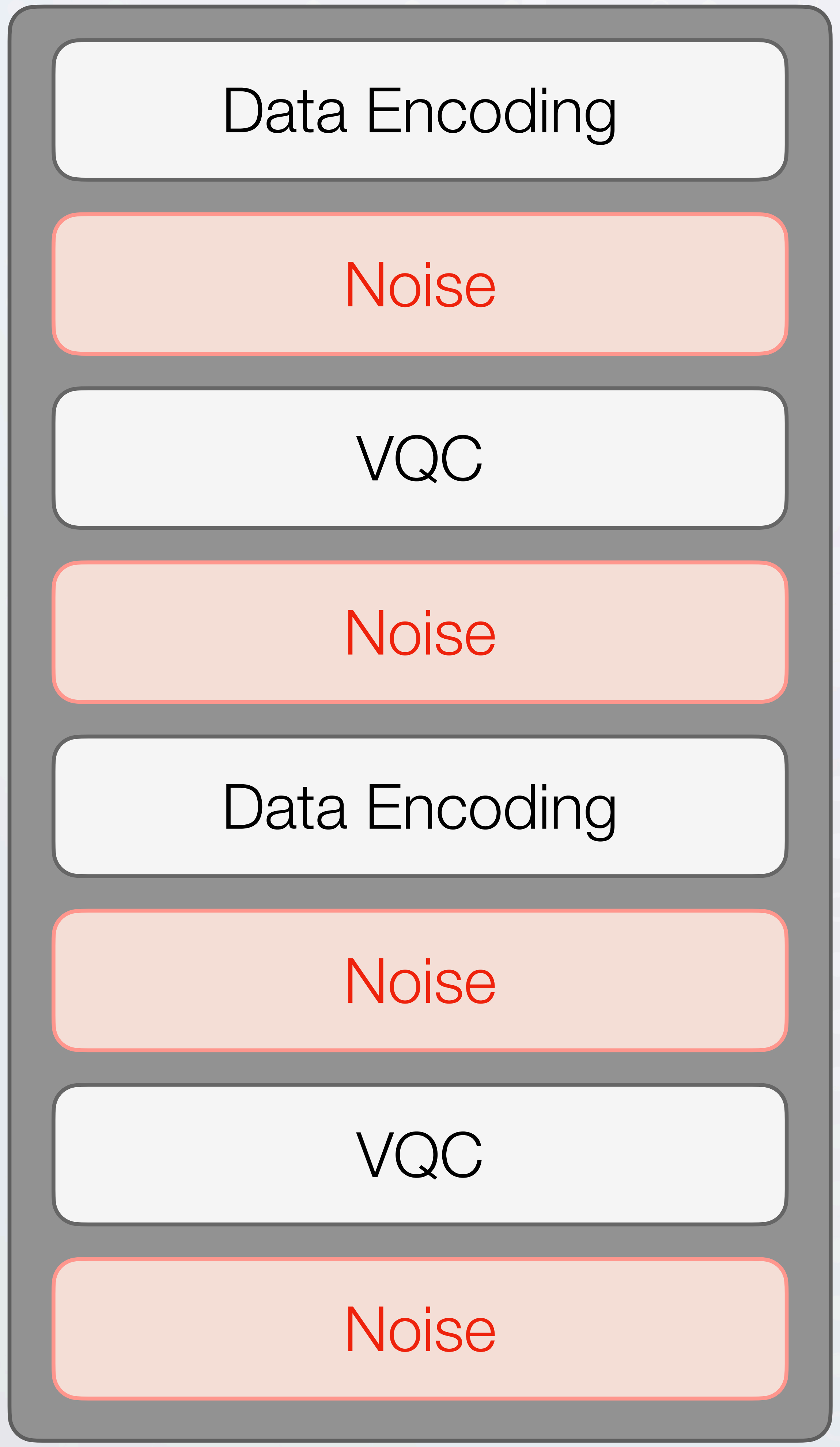

qml.StronglyEntanglingLayers

# Model Setup


Concat($\mathbf{x}_i^{(0)}, \mathbf{x}_j^{(0)}$)
$3+3 \to 6$
Linear + ReLU
$6 \to n_M$
Linear + ReLU
$n_M \to n_M$
Linear
$n_M \to n_M$
Node Aggregation
Summation $\sum$
Particle Flow Features
$(z_i, \Delta\eta_i, \Delta\phi_i)$
MPGNN
$n_M = 3$ or $6$
Graph Aggregation
Summation $\sum$
Linear + ReLU
$n_M \to 16$
Linear + ReLU
$16 \to n_C$
Particle Flow Features
$(z_i, \Delta\eta_i, \Delta\phi_i)$
Feature Transformation
In Range $(-\pi/2, \pi/2)$
QCGNN
$n_I = 4$ , $n_Q = 3$ or $6$
Linear + ReLU
$n_Q \to 16$
Linear + ReLU
$16 \to n_C$
USO
Uniform State Oracle
Data Encoding
Multi-Controlled Rotation Gates
Parametrized Operators
Strongly Entangling Layers
Data Encoding
Multi-Controlled Rotation Gates
Parametrized Operators
Strongly Entangling Layers
Measurements
Measure X for IR, and Z for NR

# Noise

## Simulated with PennyLane




**qml.DepolarizingChannel**

$$K_0 = \sqrt{1-p}\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$$

$$K_1 = \sqrt{p/3}\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}$$

$$K_2 = \sqrt{p/3}\begin{bmatrix}0 & -i\\ i & 0\end{bmatrix}$$

$$K_3 = \sqrt{p/3}\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}$$

**qml.GeneralizedAmplitudeDamping**

$$K_0 = \sqrt{p}\begin{bmatrix}1 & 0\\ 0 & \sqrt{1-\gamma}\end{bmatrix}$$

$$K_1 = \sqrt{p}\begin{bmatrix}0 & \sqrt{\gamma}\\ 0 & 0\end{bmatrix}$$

$$K_2 = \sqrt{1-p}\begin{bmatrix}\sqrt{1-\gamma} & 0\\ 0 & 1\end{bmatrix}$$

$$K_3 = \sqrt{1-p}\begin{bmatrix}0 & 0\\ \sqrt{\gamma} & 0\end{bmatrix}$$

# Parameter Shift Rule

## arXiv 1905.13311

**Gradients of parameterized quantum gates using the parameter-shift rule and gate decomposition**

Gavin E. Crooks*
*California Institute of Technology, Pasadena, CA 91125, USA and*
*Berkeley Institute for Theoretical Sciences, Berkeley, CA 94706, USA*

- Consider a VQC output $f(\theta) = \langle \psi | U_G^\dagger(\theta) A U_G(\theta) | \psi \rangle$, where $A$ is some Hermitian operator of observable and $U_G(\theta) = e^{-ia\theta G}$ with some Hermitian operator $G$.

- If $G$ has two unique eigenvalues $e_0$ and $e_1$, the gradient can be calculated by

$$\frac{d}{dx} f(x) = r \left[ f(\theta + \frac{\pi}{4r}) - f(\theta - \frac{\pi}{4r}) \right] \qquad \text{with} \qquad r = \frac{a}{2}(e_1 - e_0)$$

# Barren Plateau

arXiv 2309.09342

$$\mathrm{Var}_{\boldsymbol{\theta}}[\ell_{\boldsymbol{\theta}}(\rho, O)] = \frac{\mathcal{P}_{\mathfrak{g}}(\rho)\mathcal{P}_{\mathfrak{g}}(O)}{\dim(\mathfrak{g})}$$