Generative quantum machine learning for the hadronization process

Jinghong Yang

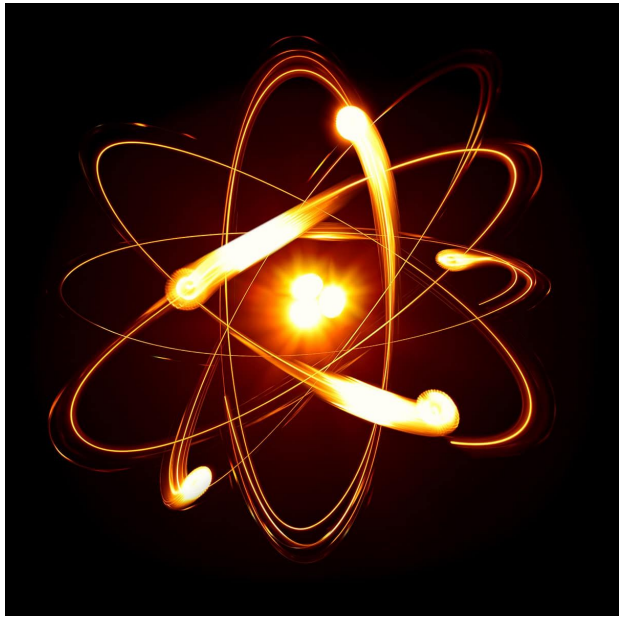
J. Munshi, A.Prasanna, A. Suresh, S. Jabeen

PHENO 2026

University of Pittsburgh

Outline

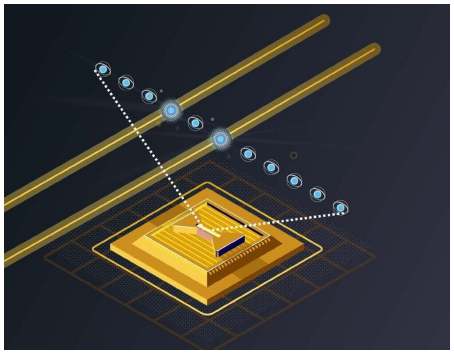
- Introduction
  - Quantum computing
  - Quantum machine learning (QML)
- Hadronization process
- Preliminary results



credit: physicsorld.com

Qubits, Gates, and Circuits

Physical Qubits

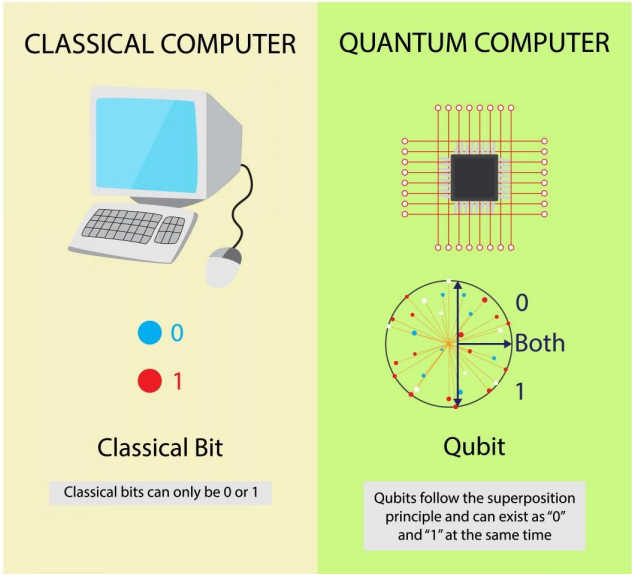


As an example, let's look at an ion trap quantum computer.

Two of the energy levels of the ion are used as the $|0\rangle$ state and $|1\rangle$ state.

Laser are used to control the qubits.

Qubits Representation in Bloch Sphere

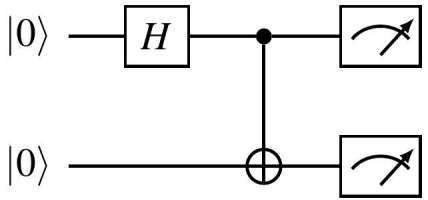


Credit: Nasky / Shutterstock.com

Quantum Computing Operations

- Initialize
- Go through operations called "quantum gates"
- Measure

Hadamard, a single qubit gate



CNOT gate, a type of two-qubit gate

Prospects of quantum computing

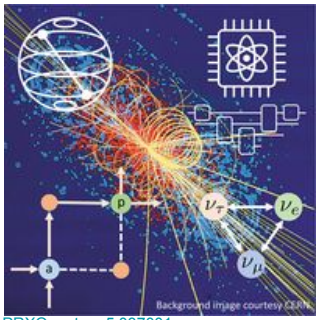
Where can quantum computing offer an advantage?

- cryptography (e.g. factoring)
- simulation of quantum systems.


credit: Q-munity

Other possibilities to explore

- quantum optimizers
- quantum annealing
- quantum machine learning
- ……



PRXQuantum.5.037001

Ref: Quantum Computing in the NISQ era and beyond

Machine Learning

Classical

- Classical neural networks
- Parameters of neural network are varied to minimize loss function

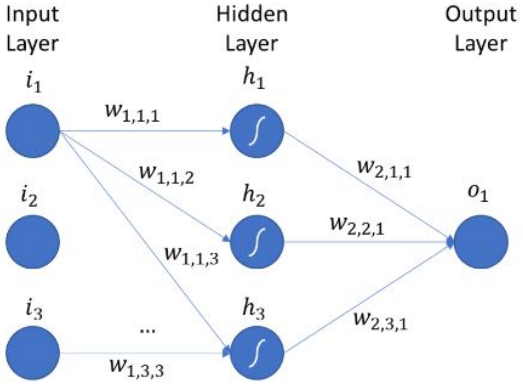
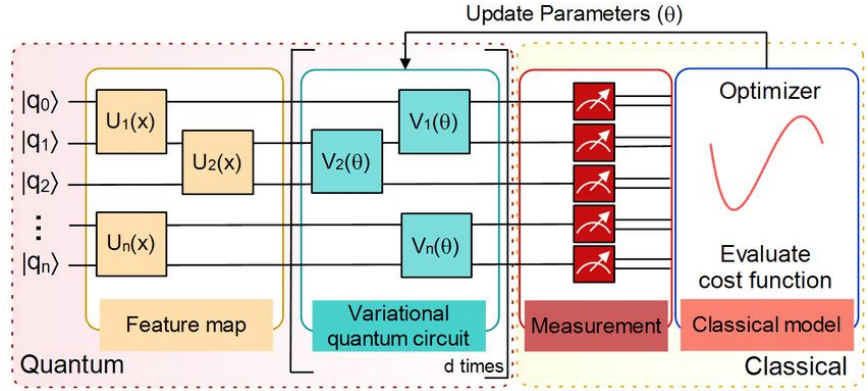


image source

Quantum

- Variational quantum circuit
- Gate parameters are varied to minimize loss function

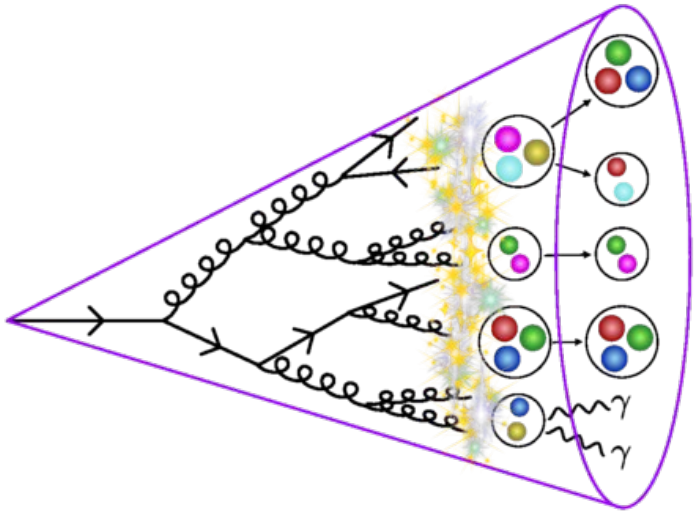


credit: Sen et al.

Hadronization

What is hadronization?

- Quarks and gluons cannot exist on their own.
- They will turn into hadrons.
- This process is called hadronization

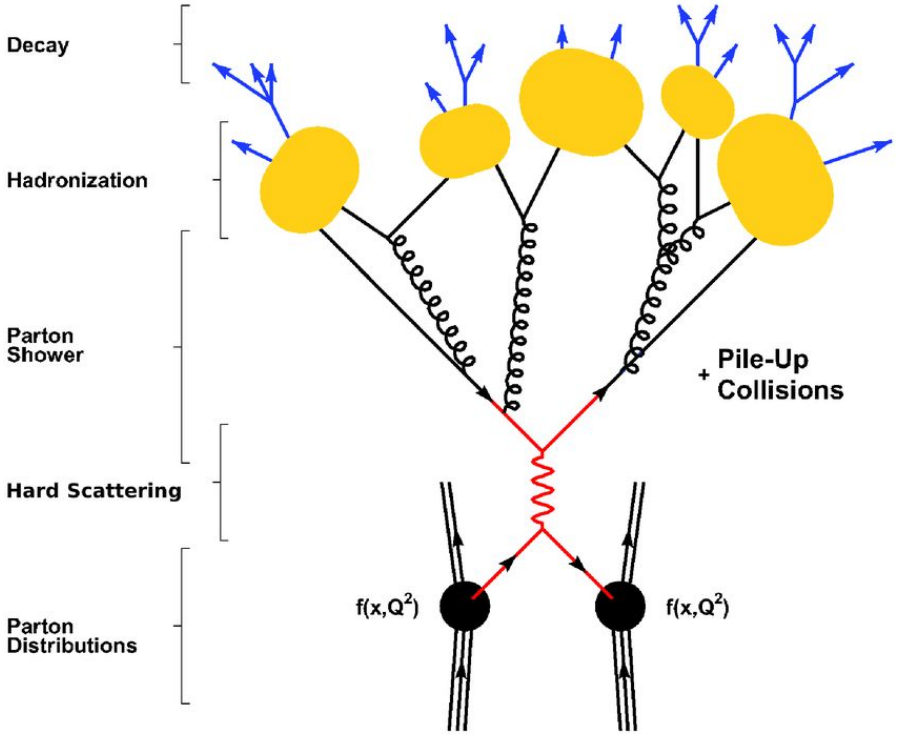


credit: E. Pottebaum

Compared to other processes in particle collisions, understanding of hadronization is relatively limited.

Importance of hadronization

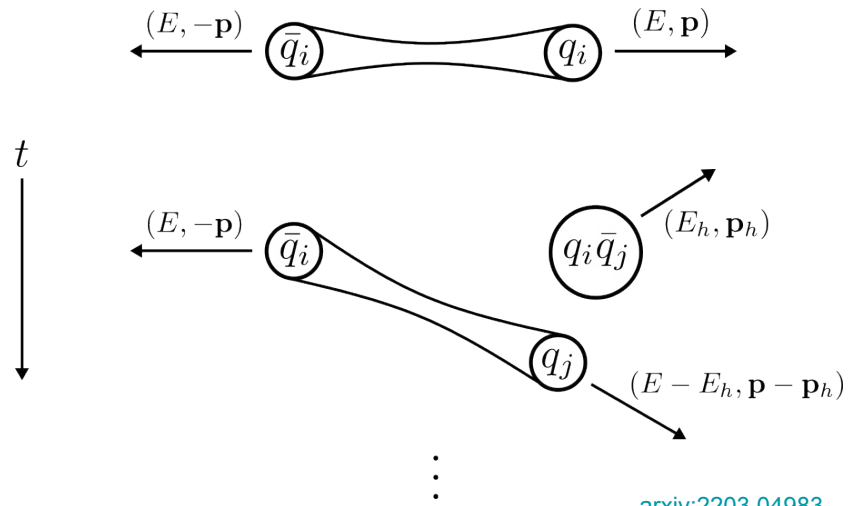
- Needed for particle collision simulation
- Hadronization is not easy to calculate
  - Non-perturbative
  - Involve quantum dynamics
- Existing approaches use phenomenological models
  - Lund string model in Pythia
  - Cluster model in Herwig
- Areas for improvement
  - e.g., arxiv:2004.03540, arxiv:1612.08975



credit: Julia Bauer

Machine learning modeling of hadronization

- Recent efforts to use classical machine learning to model hadronization:
  - HadML ref1, ref2, etc.
  - MLHAD ref1, ref2, etc.



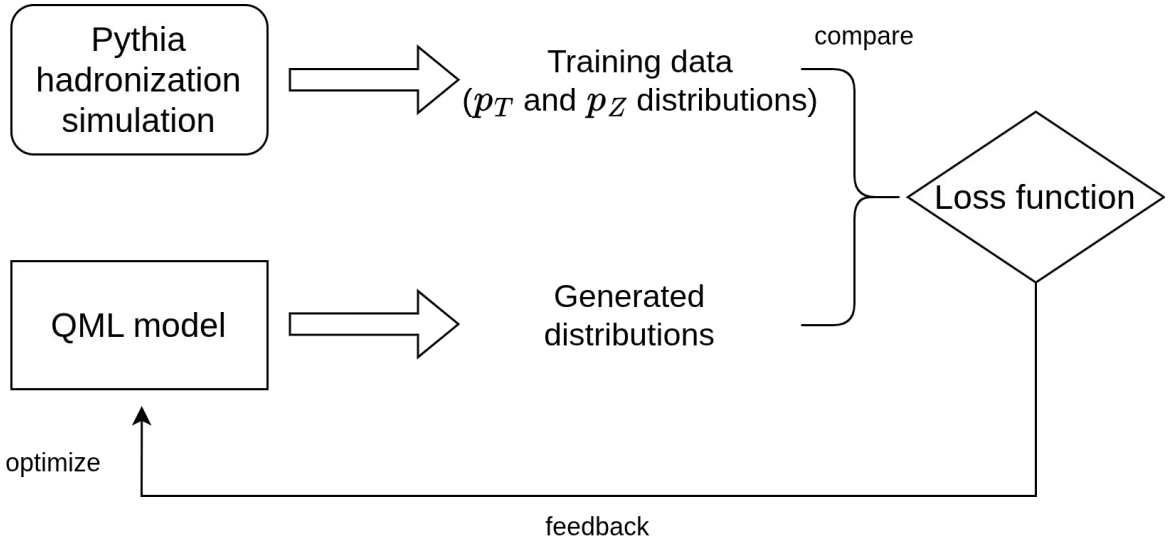
arxiv:2203.04983

E.g., arxiv:2203.04983 uses classical ML to reproduce a simplified version of the lund-string model

- Focus on a single step in hadronization, rather than the whole process
- Studies how a “string” emit a hadron
- Try to reproduce the kinematics of the first-emitted hadron
  - Pt
  - Pz

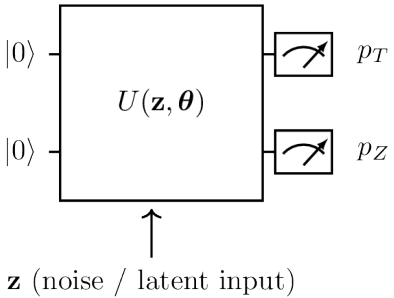
Generative QML for hadronization

- As a first step, use pythia's hadronization component to produce training data
- Use various QML models to reproduce the kinematics distributions of particles



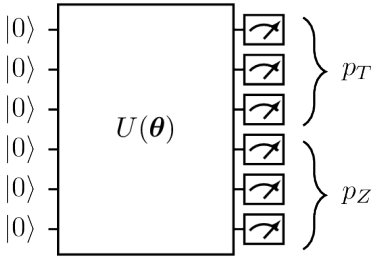
Quantum circuit born machine (QCBM)

Continuous QCBM



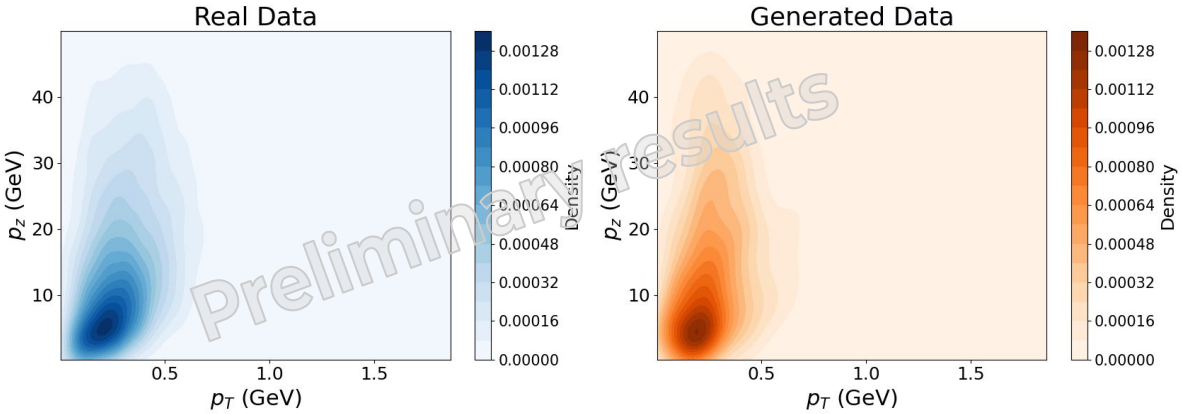
- Output p_T and p_Z from measured observables
- Uses expectation values / shot averages
- Requires latent input $\mathbf{z}$

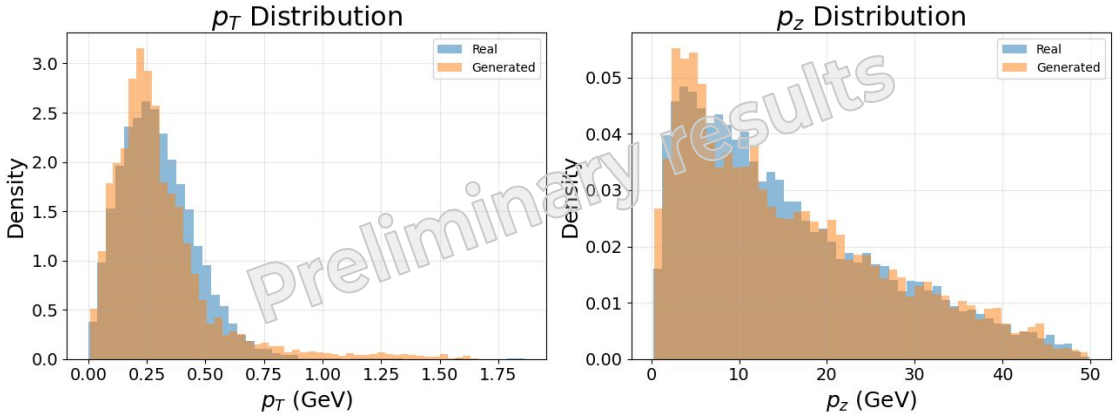
Discrete QCBM



- Samples bit strings from measurement
- Maps bit strings to p_T, p_Z
- No separate latent input needed

Preliminary results: continuous QCBM





Continuous QCBM with Log Min-Max normalization.

16 layer model
96 parameters

Loss function:
MMD loss with extra penalty terms

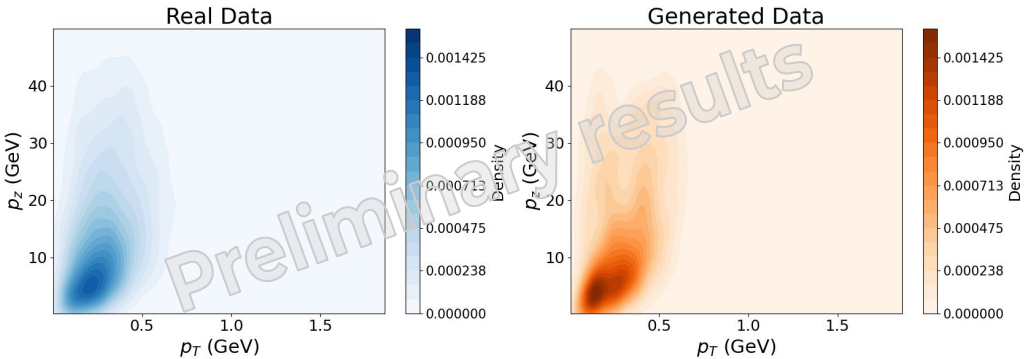
Optimizer: SPSA algorithm

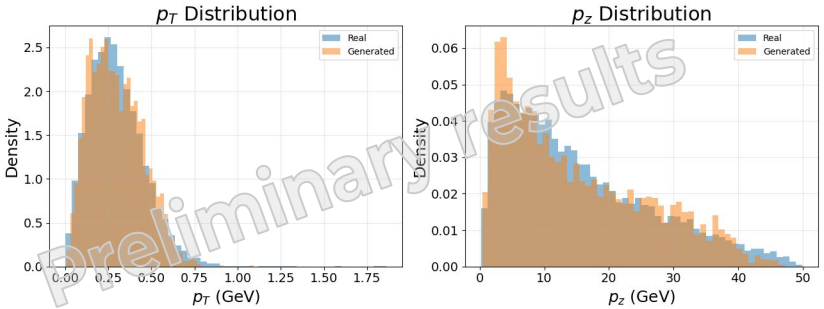
Continuous QCBM with alternative normalization

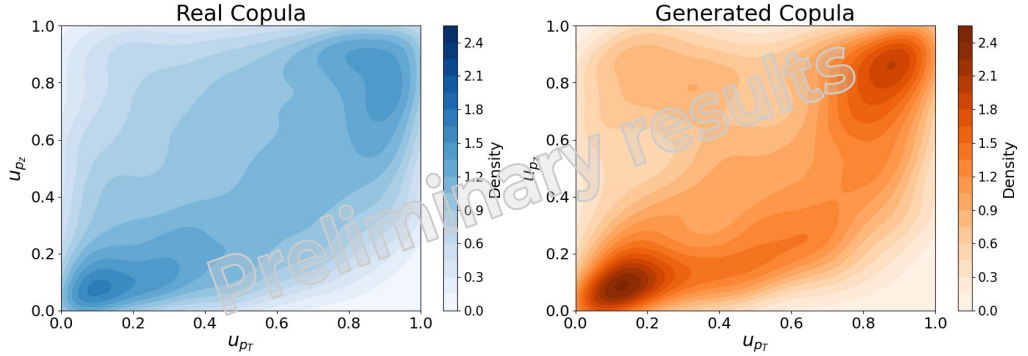
Probability integral transform (PIT)

- After transform, each variable has a uniform marginal distribution

20 layers with 120 parameters
SPSA algorithm





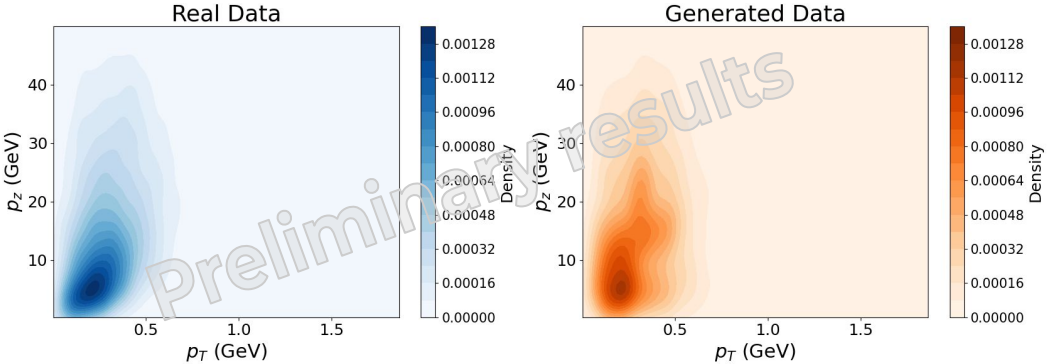


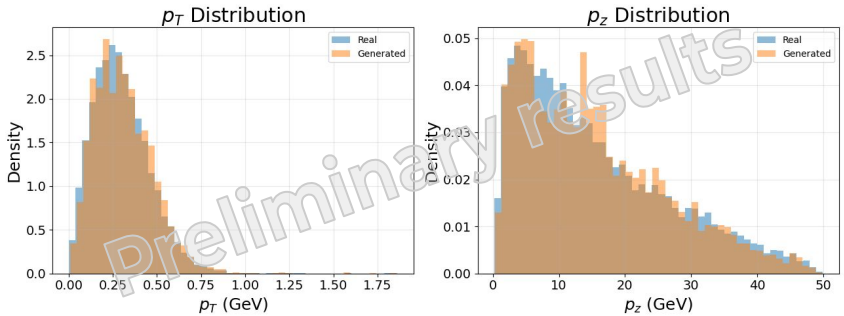
Discrete QCBM

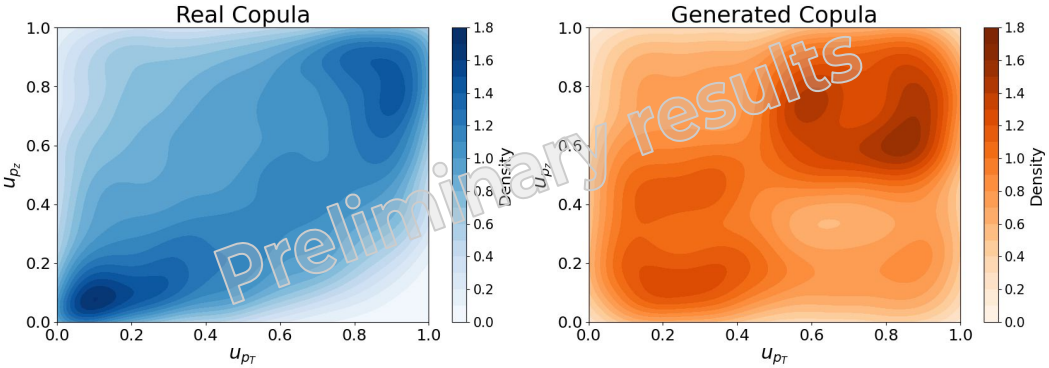
Discrete QCBM with PIT transformation

4 qubits for each variable

4 layers with 96 parameters
SPSA algorithm
Loss function: JSD

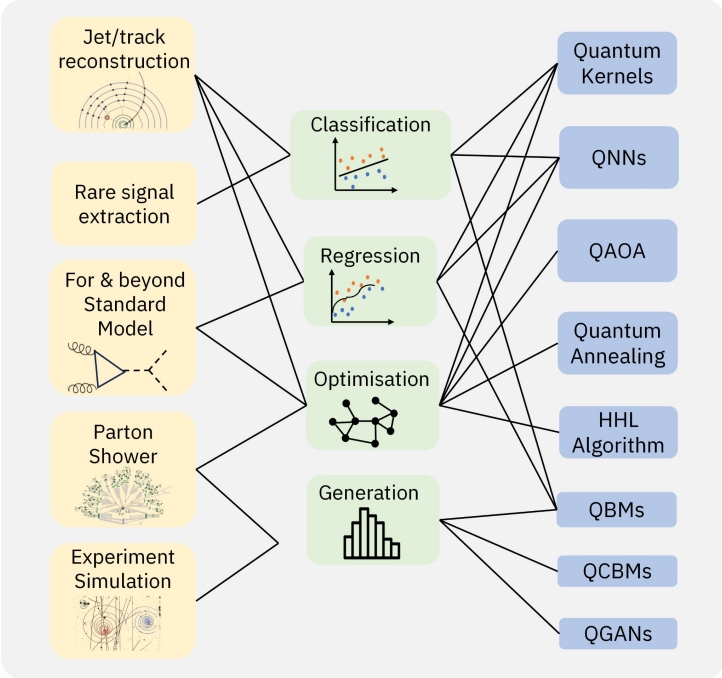






Conclusion

- We explore the use of generative quantum machine learning (QML) for “quantum” problems.
- Hadronization process is a natural candidate.
- Preliminary results show that generative QML has potential for hadronization problem.



credit: Meglio et al.

Thanks for listening!

Comments and suggestions appreciated.

Backup slides

Hadronization

Motivation for exploring alternative strategies

- Measurement of the Lund jet plane using charged particles in 13 TeV proton-proton collisions with the ATLAS detector
- Insight into particle production mechanisms via angular correlations of identified particles in pp collisions at \sqrt{s}=7 TeV
- Thermodynamical String Fragmentation
- Collective properties of hadrons in comparison of models prediction in pp collisions at 7 TeV
- Hadronization mechanisms and strange-particle production in inelastic proton-proton collisions at energies available at the CERN Super Proton Synchrotron

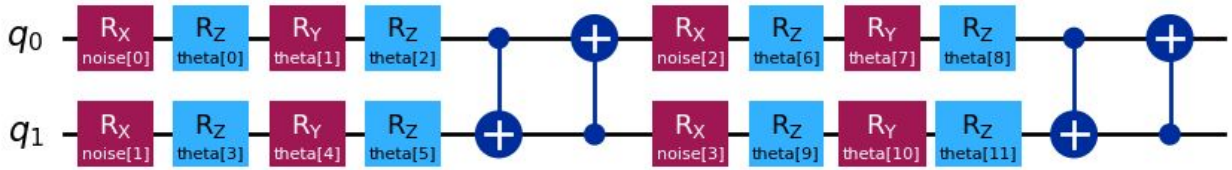
Quantum machine learning

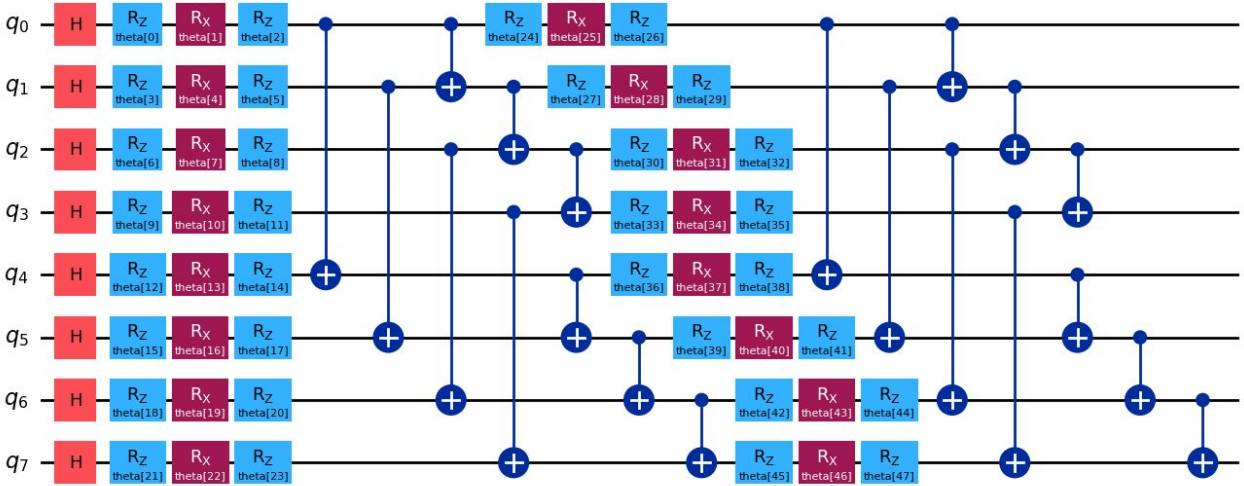
Why quantum generative models may be worth exploring

- The Expressive Power of Parameterized Quantum Circuits
- On the Quantum versus Classical Learnability of Discrete Distributions
- Enhancing Generative Models via Quantum Correlations
- The Born Supremacy: Quantum Advantage and Training of an Ising Born Machine
- Power of data in quantum machine learning
- Information-theoretic bounds on quantum advantage in machine learning
- Quantum advantage in learning from experiments

Hadronization

circuit ansatz





Loss function

Maximum mean discrepancy (MMD loss)

$$\mathcal{L}_{\mathrm{MMD}}(\theta) = \mathbb{E}_{\mathbf{x},\mathbf{y}\sim q_{(\theta)}}[K(\mathbf{x},\mathbf{y})] - 2\mathbb{E}_{\mathbf{x}\sim q_{(\theta)},\mathbf{y}\sim p}[K(\mathbf{x},\mathbf{y})] + \mathbb{E}_{\mathbf{x},\mathbf{y}\sim p}[K(\mathbf{x},\mathbf{y})]$$

Currently, we use the Gaussian kernel

$$K_\sigma(\mathbf{x},\mathbf{y}) = e^{-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2\sigma}}$$

NLL KDE loss is also used for some settings

$$\mathcal{L}_{\mathrm{NLL}} = -\mathbb{E}_{\mathbf{x}\sim\mathrm{real}}[\log(\hat{p}_{\mathrm{gen}}(x))]$$

where $\hat{p}_{\mathrm{gen}}$ can be estimated from generated samples $\{y_j\}$ as

$$\hat{p}_{\mathrm{gen}}(x) = \frac{1}{N}\sum_{j=1}^{N}\frac{1}{(2\pi h^2)^{d/2}}\exp\left(-\frac{\|x-y_j\|^2}{2h^2}\right)$$

PIT normalization

See A performance characterization of quantum generative models

PIT normalization maps variable “x” to “u”, where

$$u = P(X \leq x) = \int_{-\infty}^{x} f_X(t)dt$$

Here, $f_X(t)$ describes the (marginal) probability distribution of x.