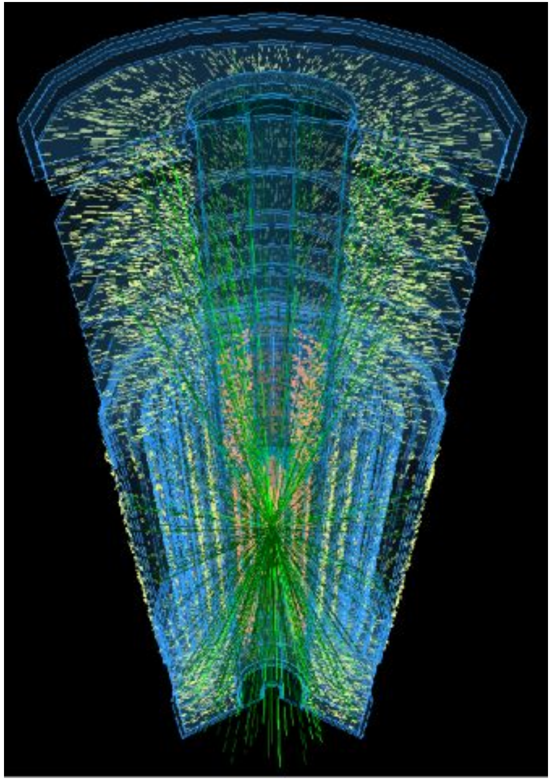


Quantum-assisted Deep Generative Calorimeter Surrogate

Hao Jia
TRIUMF & UBC
May.30rd, 2024

ATLAS EXPERIMENT

Background

- Detector simulation used almost 40% of the computing resources of the ATLAS experiment for LHC Run 2 analysis.
- Current techniques for Calorimeter shower simulation are computationally expensive

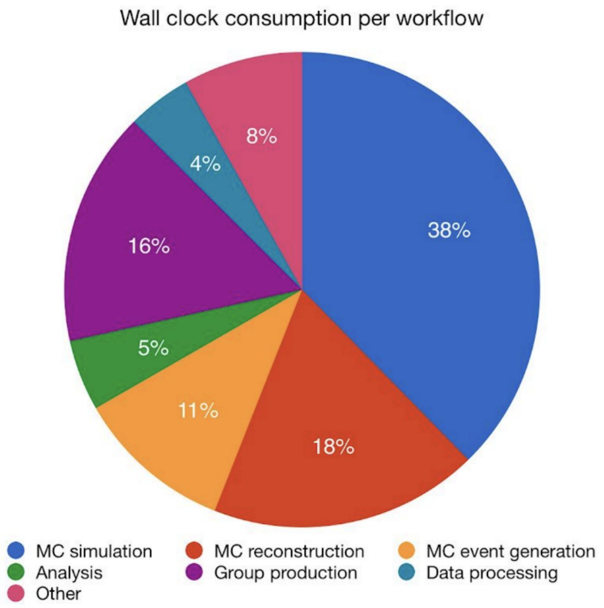
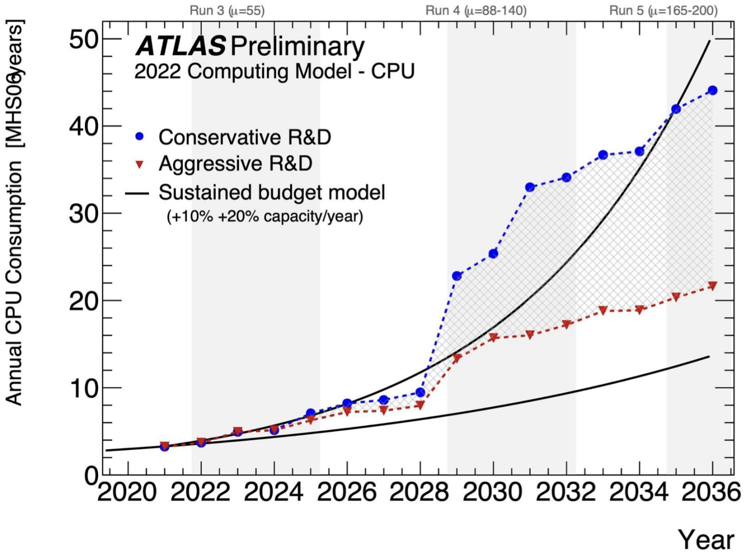


Figure 1: ATLAS CPU hours used by various activities in 2018

[1] P. Calafiura et al. ATLAS HL-LHC Computing Conceptual Design Report. Technical report, CERN, Geneva, Sep 2020 [2] AtlFast3, https://arxiv.org/abs/2109.02551

Background

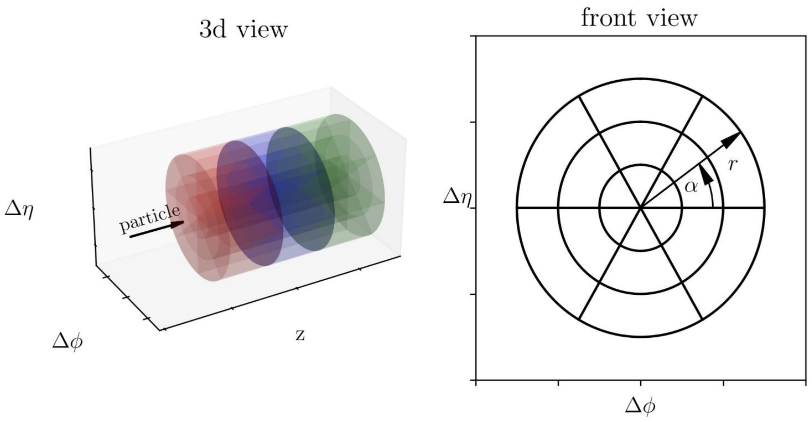
- Detector simulation used almost 40% of the computing resources of the ATLAS experiment for LHC Run 2 analysis.
- Current techniques for Calorimeter shower simulation are computationally expensive.
- Need to develop a faster, computationally cheaper detector simulation techniques for HL-LHC.



[1] P. Calafiura et al. ATLAS HL-LHC Computing Conceptual Design Report. Technical report, CERN, Geneva, Sep 2020 [2] AtlFast3, https://arxiv.org/abs/2109.02551

Dataset

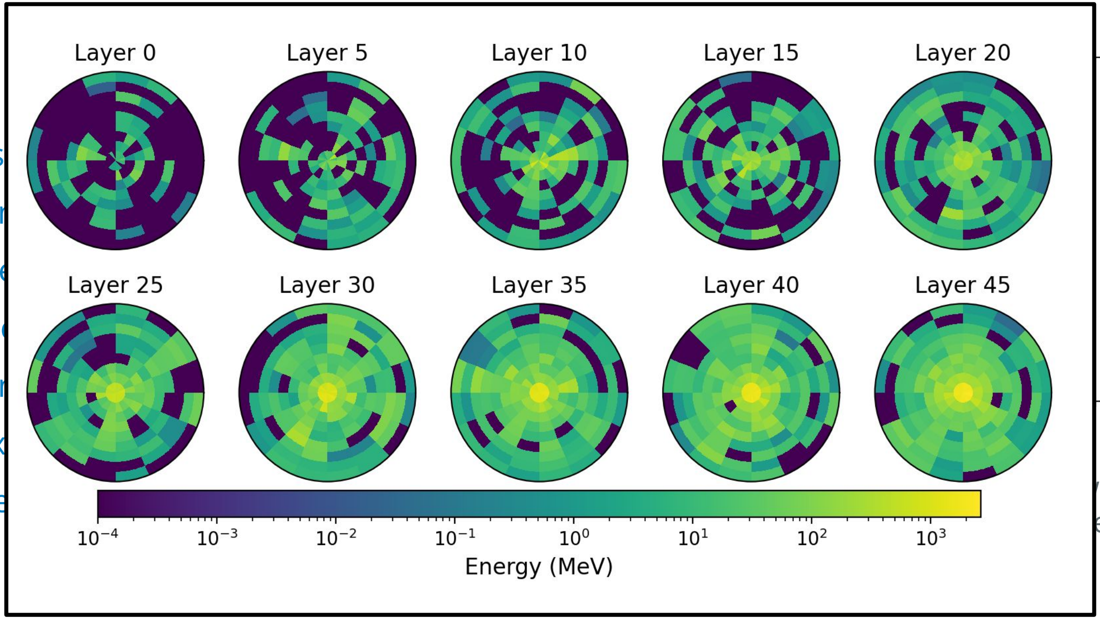
- 100,000 GEANT4-simulated electron showers (1 GeV to 1 TeV)
- The geometry features a concentric cylinder structure with 45 layers
- Each layer has 144 readout cells, 9 in radial and 16 in angular direction, yielding a total of 9x16x45 = 6480 voxels
- Each event: {input_energy: 1x6480 tensor}

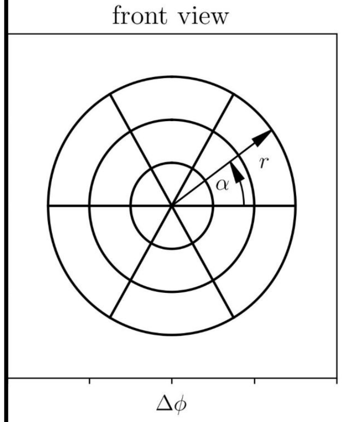


The image shows a 3d view of a geometry with 3 layers, with each layer having 3 bins in radial and 6 bins in angular direction.

Dataset

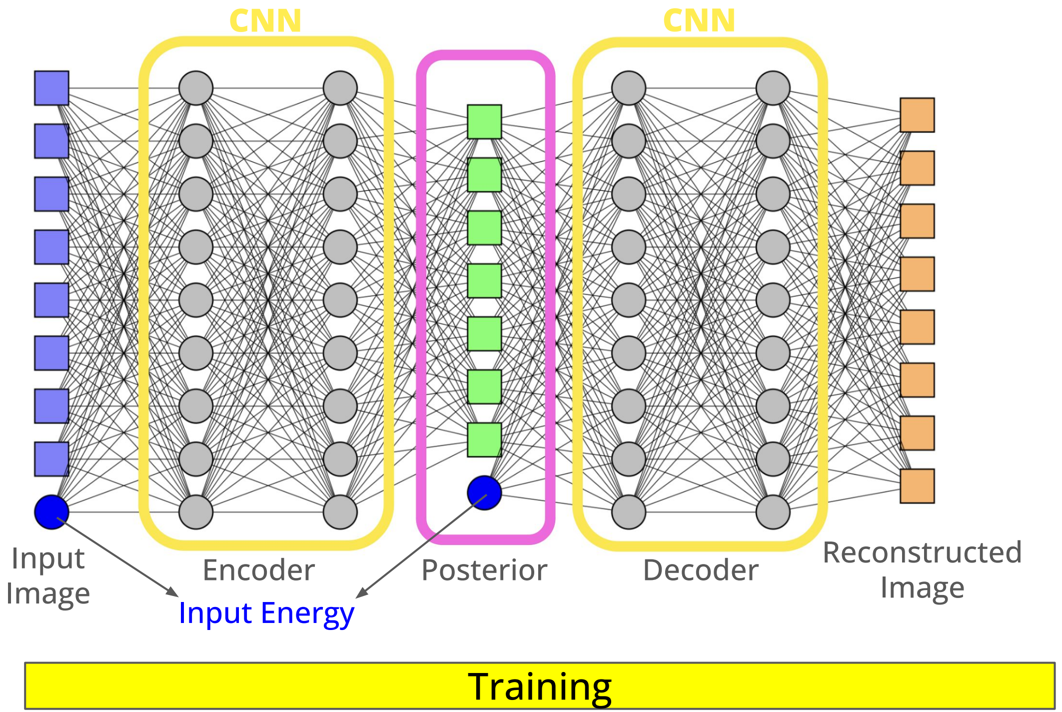
- 100,000
  showers
- The geo
  structure
- Each lay
  and 16 i
  of 9x16x
- Each eve



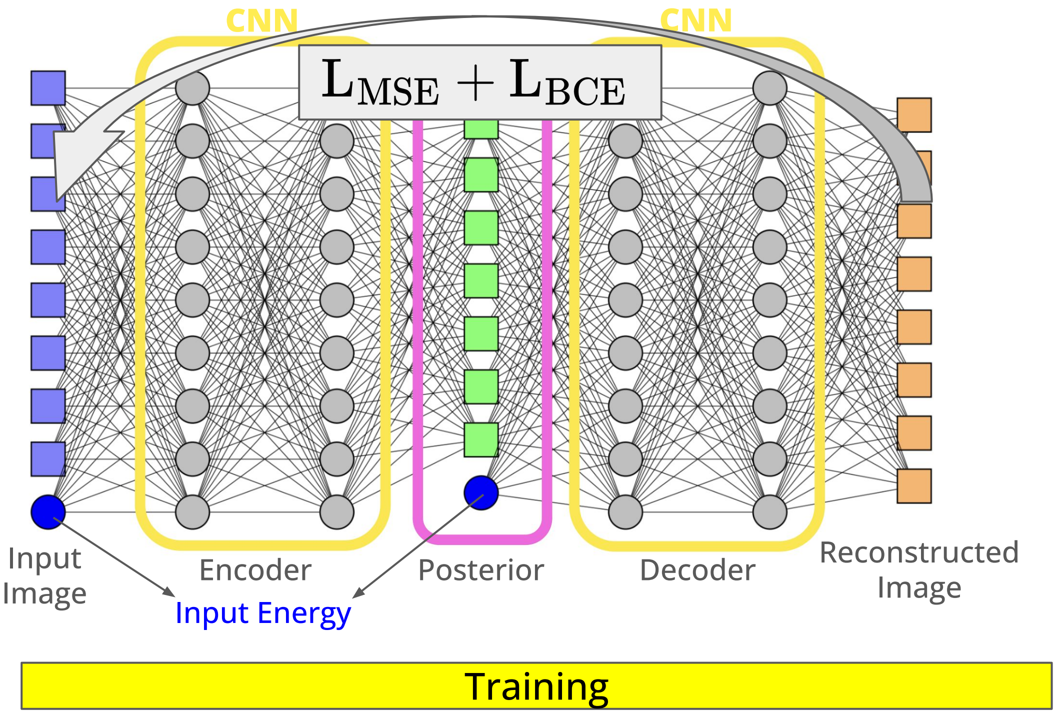


of a geometry
er having 3 bins
lar direction.

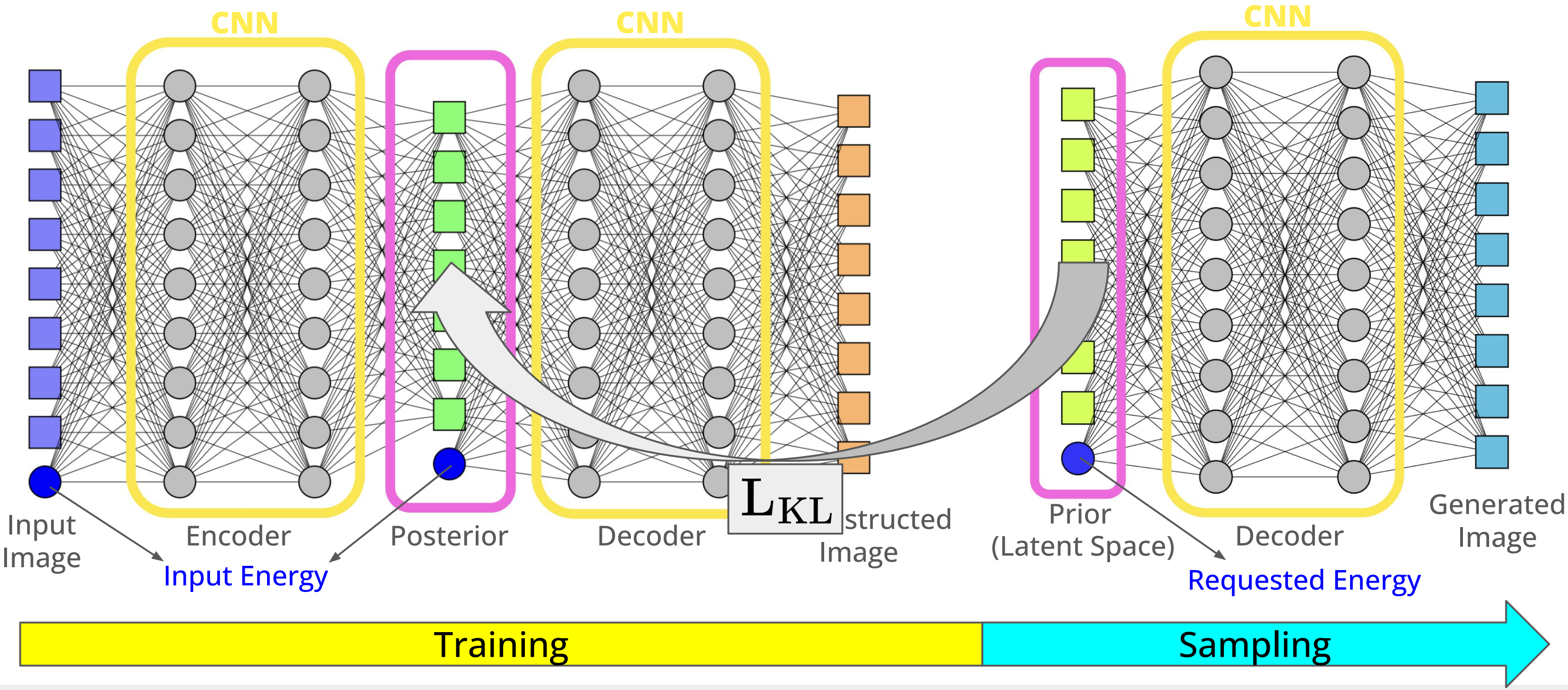
Variational Autoencoder



Variational Autoencoder

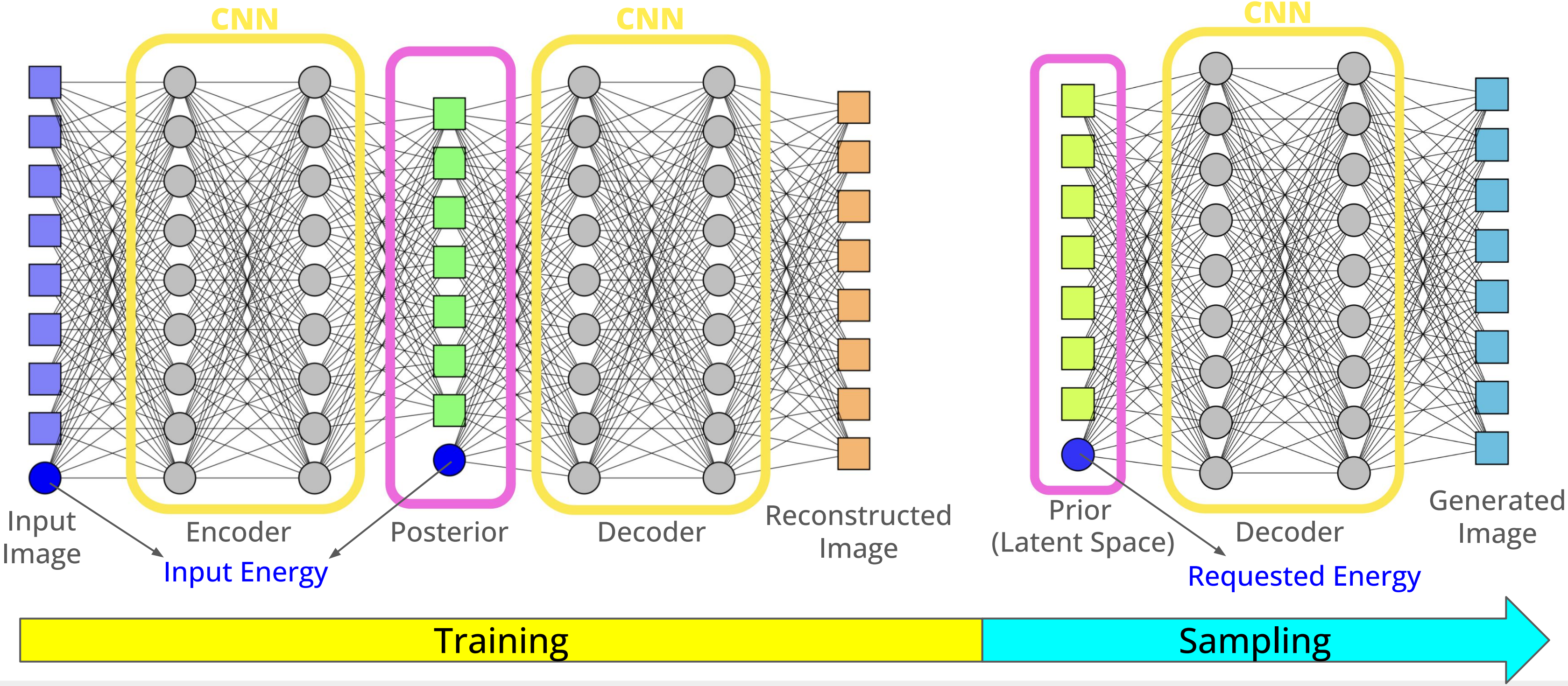


Variational Autoencoder

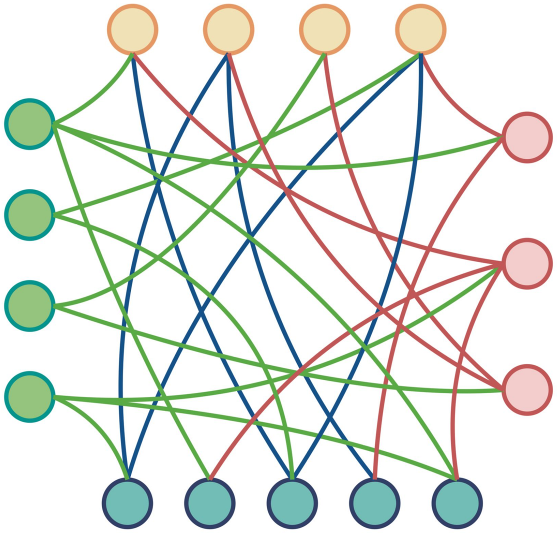


Variational Autoencoder

$$L = L_{MSE} + L_{BCE} + L_{KL}$$



Prior: Restricted Boltzmann Machine



4-Partite RBM based on
D-Wave's Pegasus Topology

For $\mathbf{x} = (1, 0, 1, 1, 0, 1, ..., 0, 1)$

$$P(\mathbf{x}) = \frac{1}{Z} e^{-E(\mathbf{x})}, \quad \text{where } Z = \sum_{\mathbf{x}} e^{-E(\mathbf{x})}$$

$$E(\mathbf{x}) = - \sum_{\substack{\rho,\sigma \in \{a,b,c\} \\ \rho \neq \sigma}} \sum_{i,j} w_{ij}^{\rho\sigma} x_i^\rho x_j^\sigma - \sum_{\rho \in \{a,b,c\}} \sum_i b_i^\rho x_i^\rho$$

where $x_i \in \{0, 1\}$, w_i, b_i are trainable weights and biases.

- Energy Based Model
- More expressive than traditional Gaussian prior.
- Classically, we use Markov-chain to get samples.

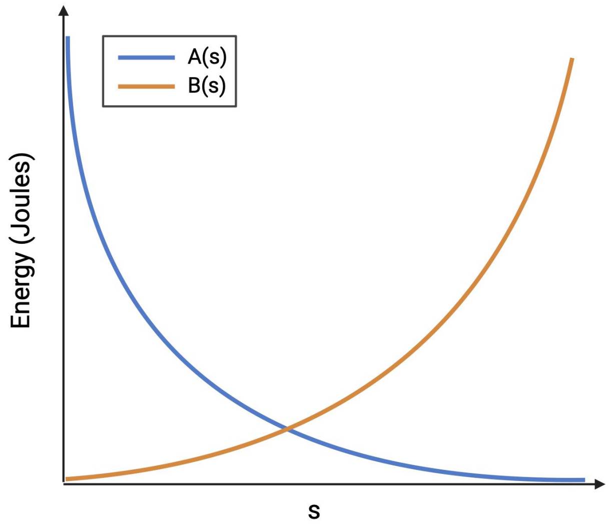
Move to QPU: Quantum Annealing

$$\mathcal{H}(s) = A(s)\sum_{l}\sigma_l^x + B(s)\left[\sum_{l}\sigma_l^z h_l + \sum_{l<m} J_{lm}\sigma_l^z\sigma_m^z\right]$$

h_l is the magnetic field acting on spin l
J_{lm} is the interaction strength between spins l and m
σ_l^z is the spin variables, which can take values of +1 or -1

Quantum Annealing:

- Start with A(0) >>B(0) end up with A(1) << B(1)
- start in quantum superposition state and end up in a classical state
- Fast! One anneal = 1 sample
- Independent samples each time!



Annealing functions $A(s)$, $B(s)$ in 1 QA cycle

Move to QPU: Quantum Annealing

$\mathcal{H}(s) = A(s)\sum$

h_l is the magn

J_{lm} is the inte

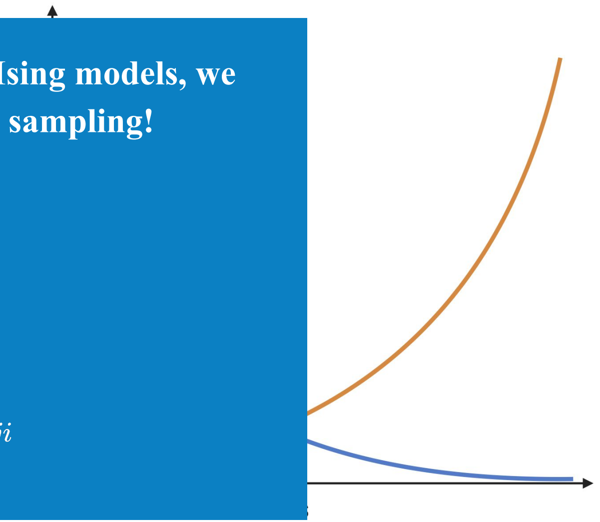
σ_l^z is the spin

Quantum An

- Start wit
  A(1) << B
- start in c
  end up i
- Fast! One anneal = 1 sample
- Independent samples each time!

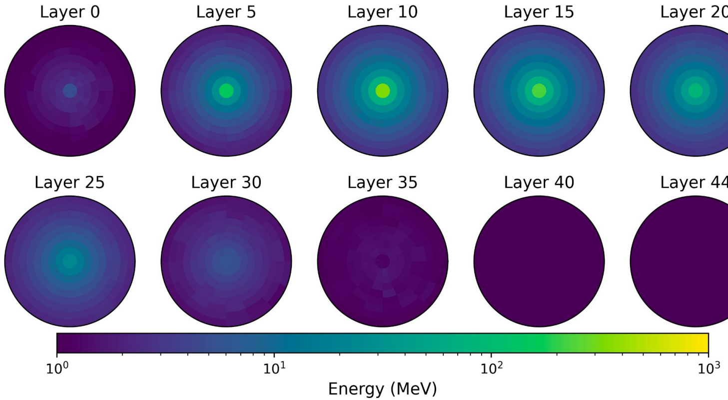
If we can build a mapping between RBM and Ising models, we can potentially use D-WAVE to do latent space sampling!

$$\begin{cases} J' = -\frac{1}{4}W \\ h'_i = -\frac{1}{2}b_i - \frac{1}{4}\sum_j W_{ji} \end{cases}$$

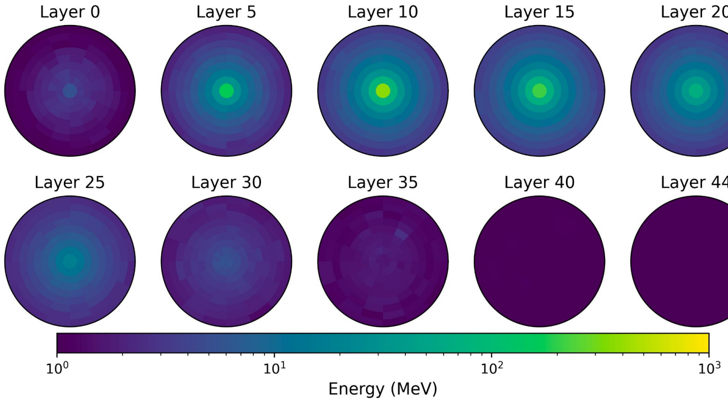


Annealing functions $A(s)$, $B(s)$ in 1 QA cycle

Results



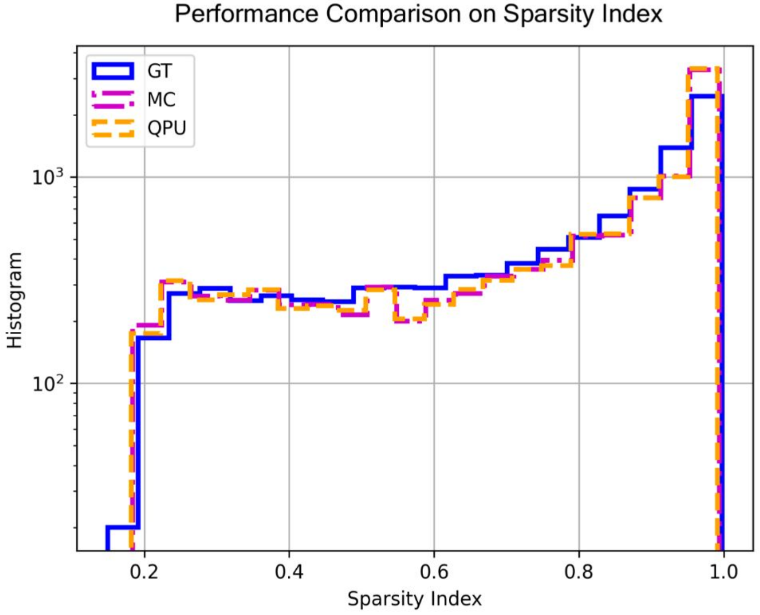
Averaged Energy Images (GEANT Target)



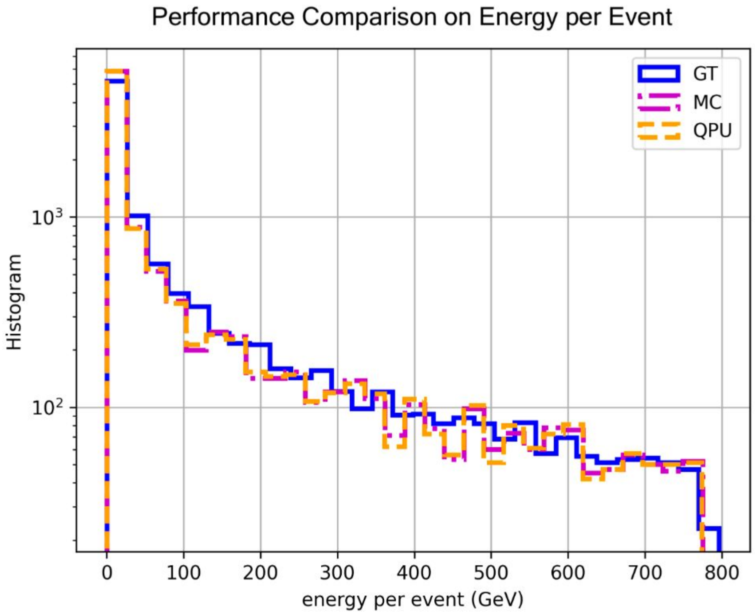
Averaged Energy Images (QPU-generated)

Synthetic Images Generation Rates Comparison				
Type	GEANT4	A100 GPU	Total QPU Access	QPU Annealing
Time per sample	~1s	~2ms	~0.2ms	~0.02ms

Performance Evaluation



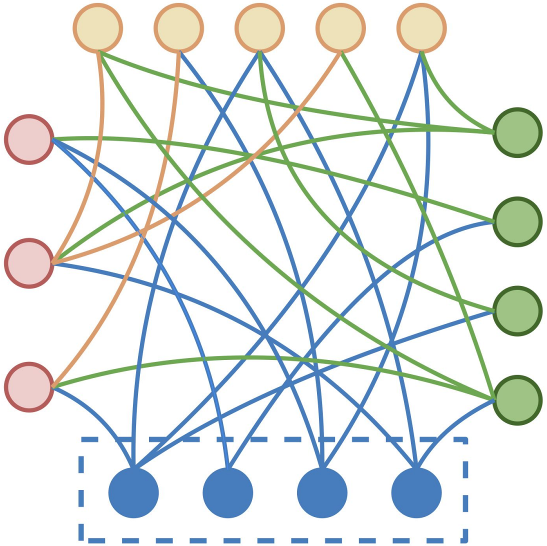
$$N(E=0)/N_{\text{tot}}$$



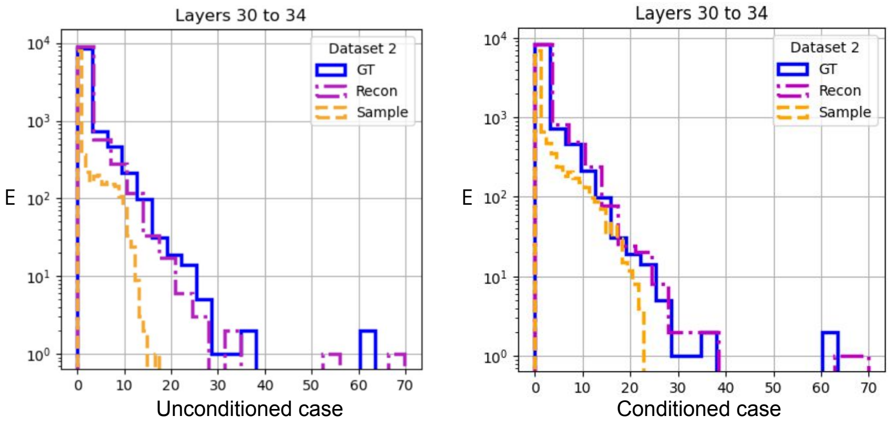
$$\sum E$$

ATLAS EXPERIMENT

Ongoing: Energy Conditioned Prior



Binary Encoded Input Energy



- For better sampling quality
- Have finished the classical training stage
- Use strong MF to configure D-Wave states
- Currently slow, need to cooperate with D-Wave

Summary

- We have shown that it is possible to utilize the Quantum Processing Unit for generating Restricted Boltzmann Machine samples, which facilitate the generation of particle showers.
- Quantum Processing Unit sampling is significantly faster than traditional Monte Carlo methods, maintaining high-quality shower image generation.
- Energy conditioned prior turns out to perform better, but more work needs to be done on the D-Wave end.

The Team

Supervisors:

- ➢ Wojciech T. Fedorko
- ➢ Maximilian Swiatlowski
- ➢ Colin Gay
- ➢ Alison Lister
- ➢ Geoffrey Fox
- ➢ Eric Paquet
- ➢ Roger G. Melko

Students & Postdocs:

- ➢ Javier Q. Toledo-Marín
- ➢ Hao Jia
- ➢ Abhishek Abhishek
- ➢ Sebastian Gonzalez
- ➢ Deniz Sogutlu
- ➢ Soren Andersen
- ➢ Sehmimul Hoque
- ➢ Lan Liu

The Team

Supervisors:

- Wojciech T. Fedorko
- Maximilian Swiatlowski
- Colin Gay
- Alison Lister
- Geoffrey Fox
- Eric Paquet
- Roger G. Melko

Students & Postdocs:

- Javier Q. Toledo-Marín
- Hao Jia
- Abhishek Abhishek
- Sebastian Gonzalez
- Deniz Sogutlu
- Soren Andersen
- Sehmimul Hoque
- Lan Liu

Thanks! / Questions?

Backup

Restricted Boltzmann Machine: Why?

Theoretical Base:
Le Roux N, Bengio Y. Representational power of restricted boltzmann machines and deep belief networks. Neural Comput. 2008 Jun;20(6):1631-49. doi: 10.1162/neco.2008.04-07-510. PMID: 18254699.

- ❖ Increasing the number of hidden units in RBMs leads to enhanced modeling power.
- ❖ RBMs are universal approximators of discrete distributions. (RBMs are theoretically capable of representing any discrete probability distribution given enough hidden units)

Pros:

- More expressive latent space
- Better Data Adaption
- Low-energy states are more probable
- Parameters jointly trained with VAE parameters

Cons:

- Computationally expensive: block Gibbs sampling
- Slower than traditional method
- Quality: block gibbs steps
- Limited GPU memory
- Correlations among samples?

What to learn: $\text{Loss} = \text{Loss}_{MSE} + \text{Loss}_{KL} + \text{Loss}_{BCE}$

- $\text{Loss}_{MSE}(\mathbf{x}, \mathbf{x}') = \frac{1}{n}\Sigma_{i=1}^{n}(\mathbf{x_i} - \mathbf{x_i'})^2$

  Also called autoencoding loss, reconstruction loss. It is used to measure the difference between the original input data and the reconstructed data.

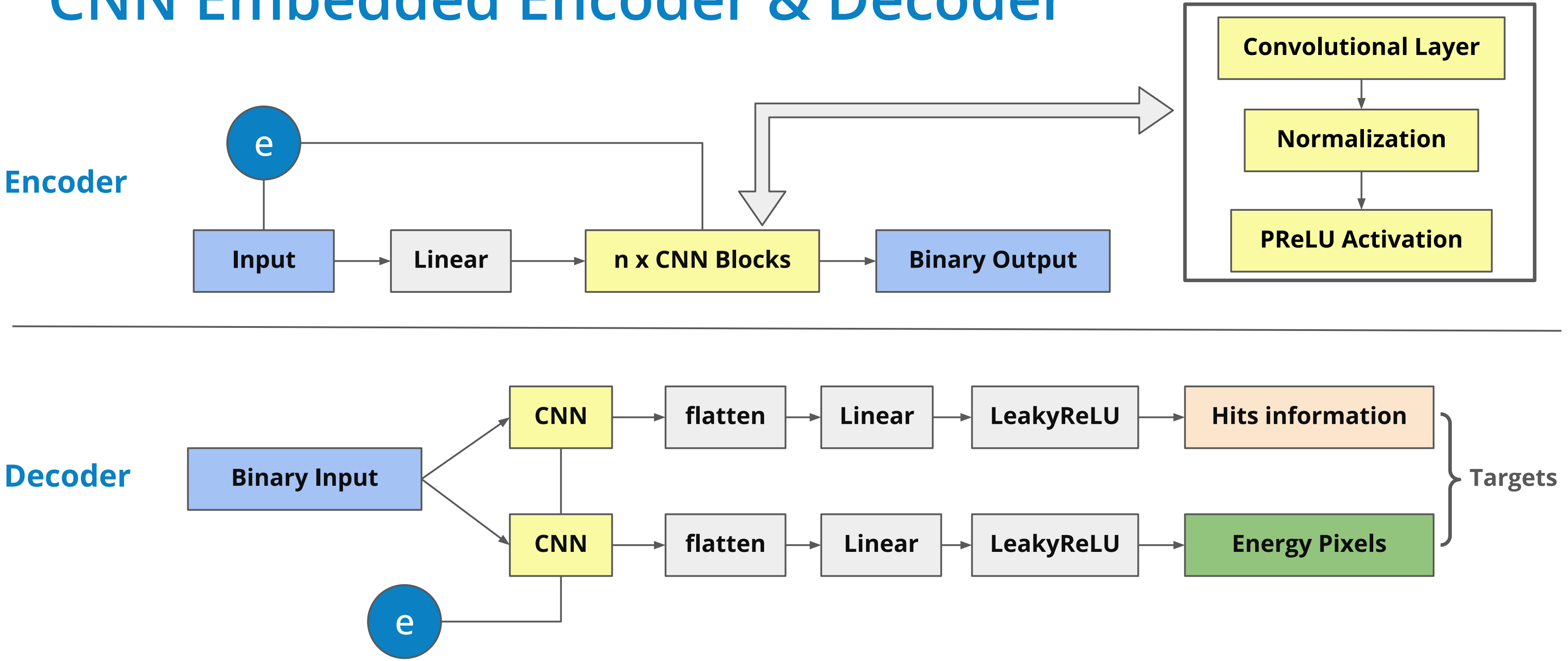
- $\text{Loss}_{KL}(q(\mathbf{z}|\mathbf{x}, e))||p(\mathbf{z})) = \mathbb{E}_{q(\mathbf{z}|x)}[\log q(\mathbf{z}|\mathbf{x}, e)) - \log p(\mathbf{z})]$

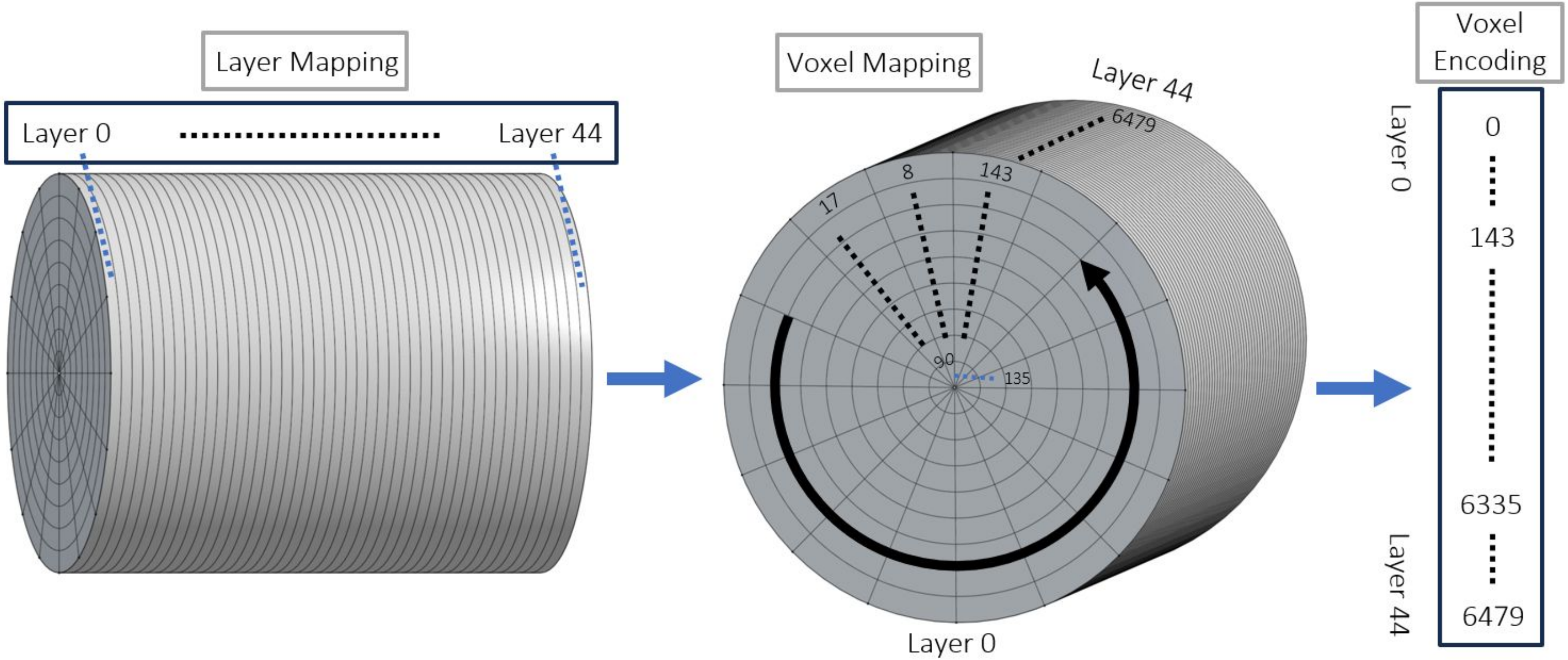
  Use the KL divergence as part of the loss function to measure the difference between the encoder output distribution (approximate posterior) q(z|x,e) and the prior distribution p(z).

- $\text{Loss}_{BCE}(\mathbf{y}, \mathbf{y}') = \frac{1}{N}\sum_{i=1}^{N}[y_i' \cdot (-\log(\sigma(y_i))) + (1 - y_i') \cdot (-\log(1 - \sigma(y_i)))]$

  Hit loss: We build the input labels (y) and reconstructed labels (y') by making each zero energy pixel label be 0 and non-zero pixel be 1. It is used to learn and normalize the output hit pattern.

CNN Embedded Encoder & Decoder




Layer Mapping
Layer 0
Layer 44
Voxel Mapping
Layer 44
6479
17
8
143
90
135
Layer 0
Voxel Encoding
Layer 0
0
143
6335
Layer 44
6479