Quantum GAN for Fast Shower Simulation

Xiaozhong Huang, Weidong Li

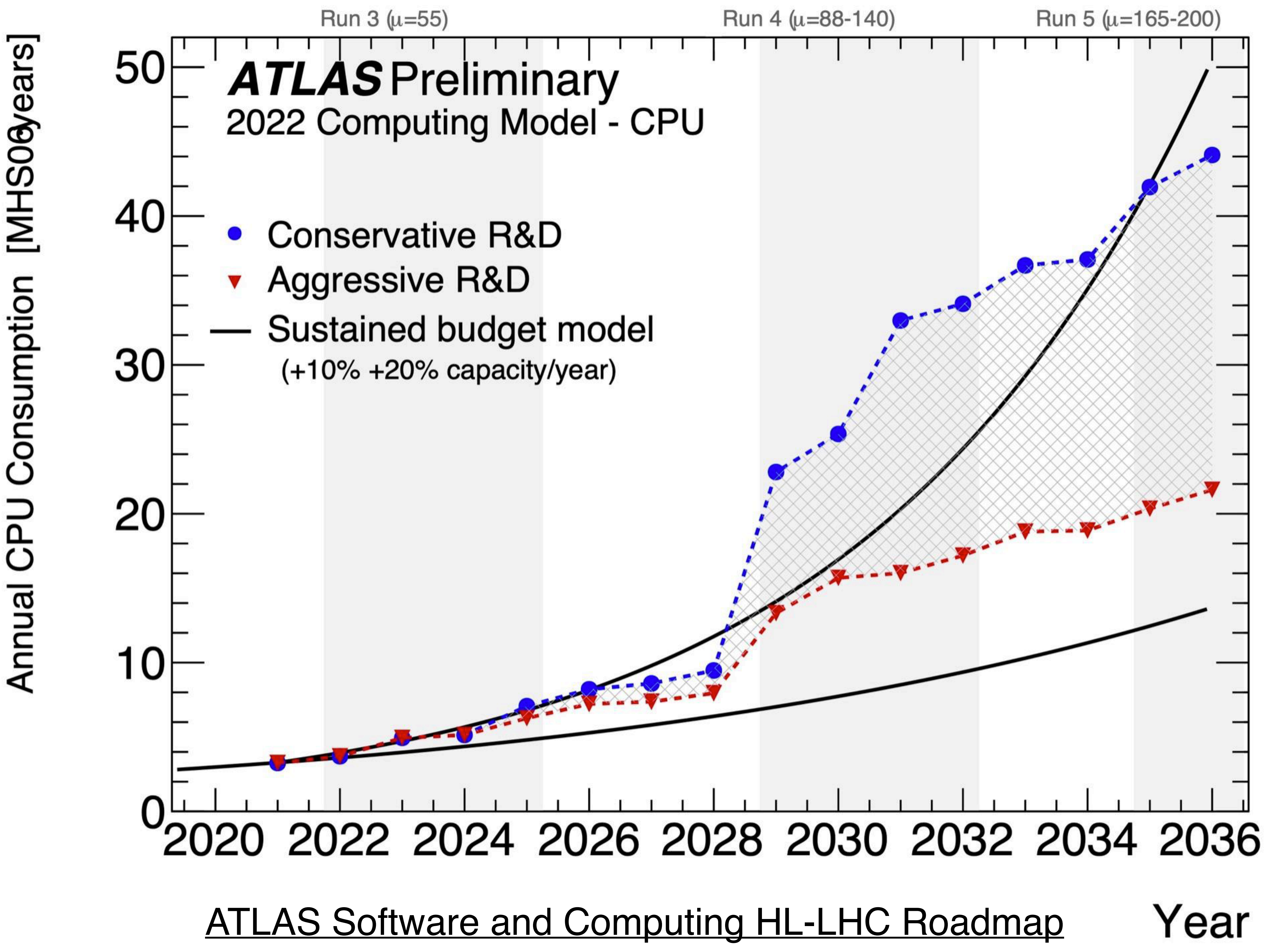
IHEP

Quantum Computing and Machine Learning Workshop

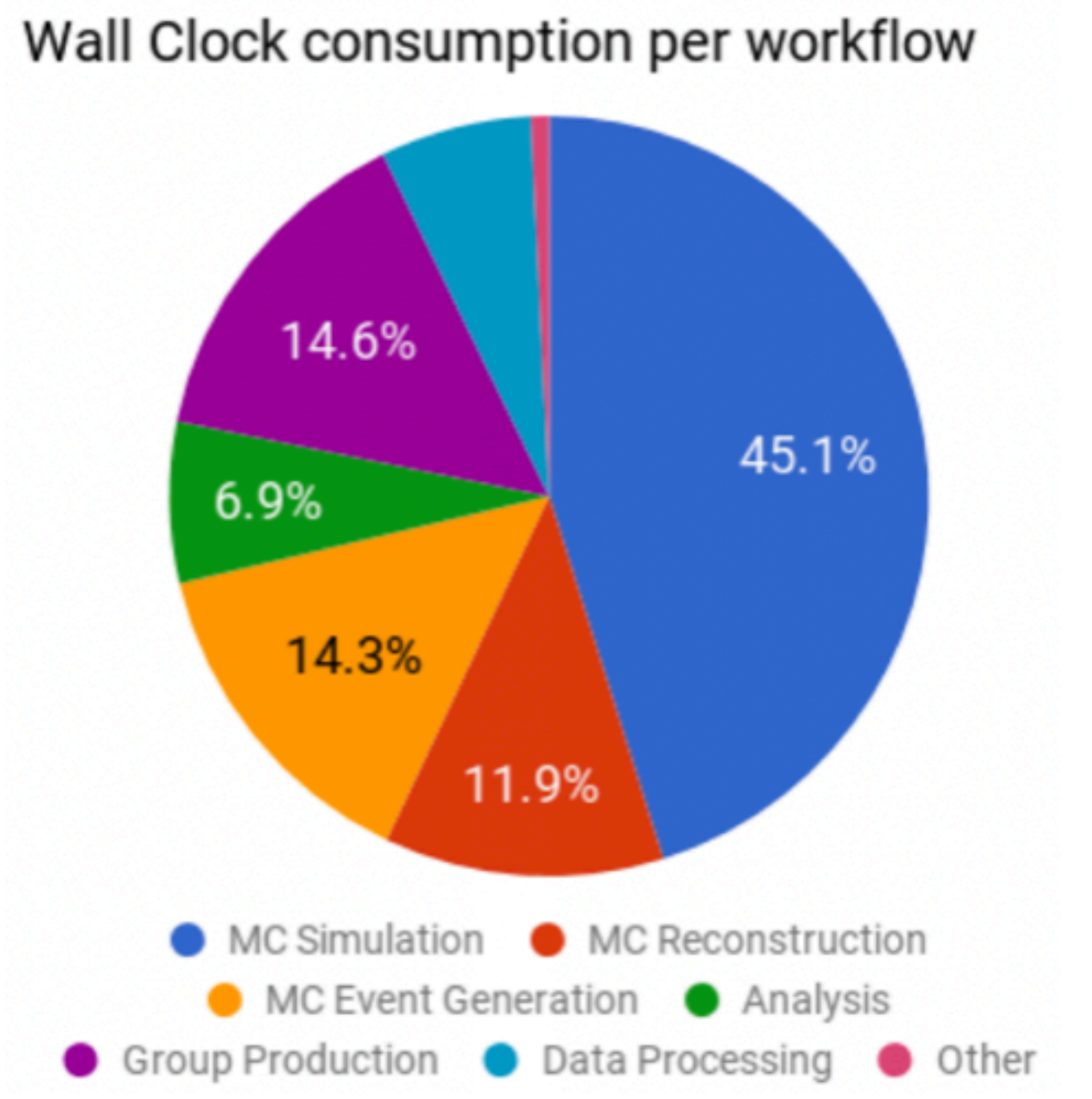
2024/8/7

Why Fast Shower Simulation?

- HL-LHC ⟶ huge computing resources
- MC simulation account for ～50%（dominated by shower simulation)
- Fast shower simulation: help overcome the computational challenge



ATLAS Software and Computing HL-LHC Roadmap



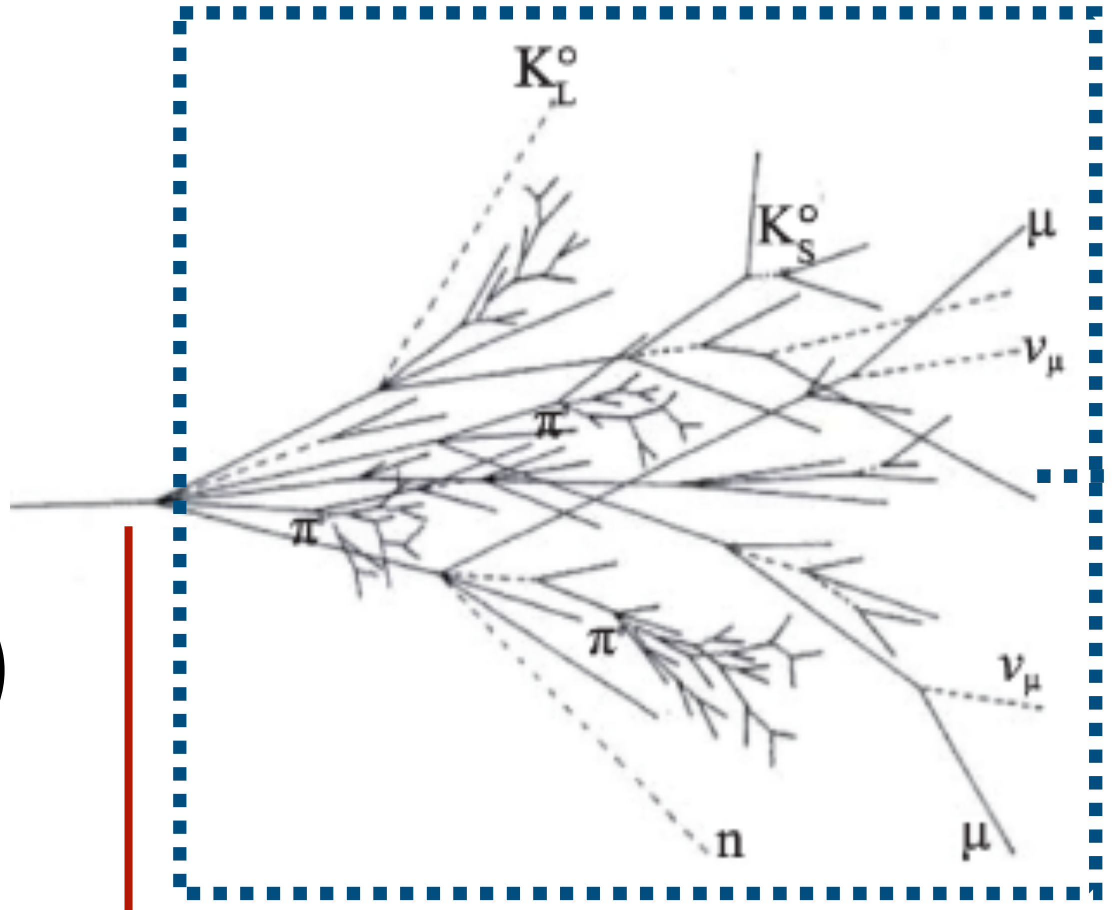
ATLAS 2017 number

Fast Simulation

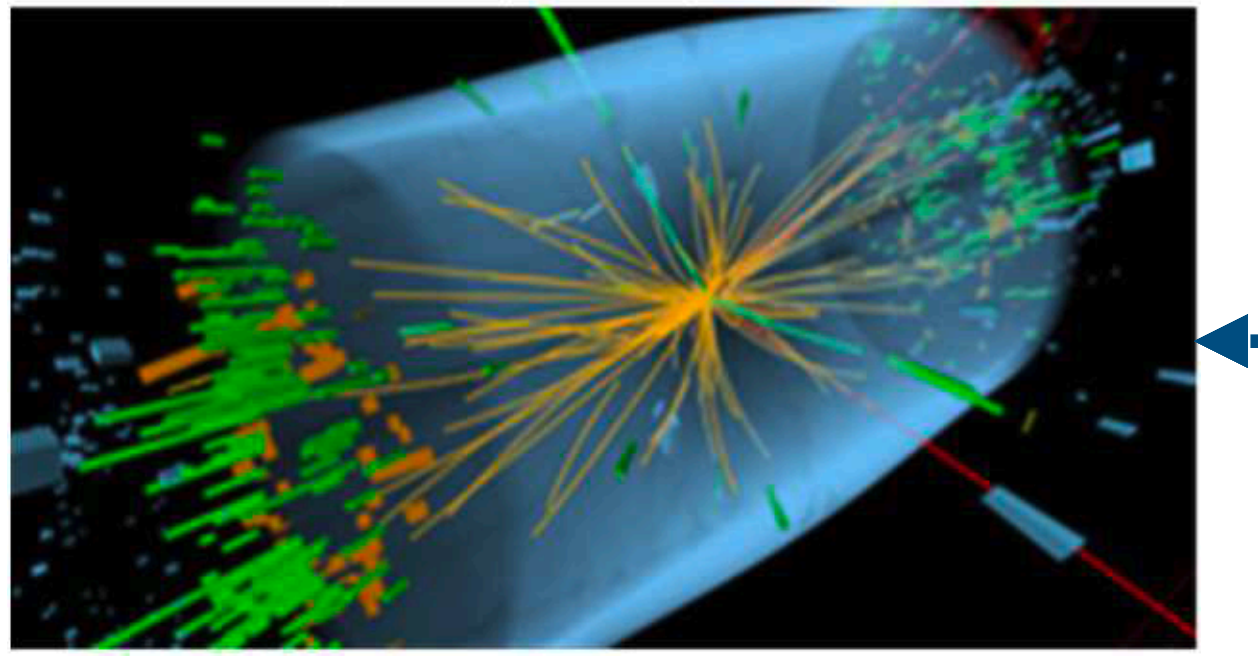
- Geant4: incoming particle → physics process in the detector → energy deposition
  - accurate results, but time-consuming
  - complex geometry
  - number of secondary particles grows quickly

- Fast simulation: incoming particle → energy deposition)
  - parameterization
  - **GAN (<u>ATLAS</u>)**
  - VAE
  - ……

Geant4



fast simulation



QC is an alternative to classical computing

QC + GAN: the potential to outperform classical GAN

Quantum GAN

- Two versions of quantum GAN
  - quantum generator + classical discriminator (choose the hybrid version for our study)
  - quantum generator + quantum discriminator

- NISQ (noisy intermediate-scale quantum era)
  - noisy and unstable qubit
  - number of qubits: [~10, $\sim 10^2$]

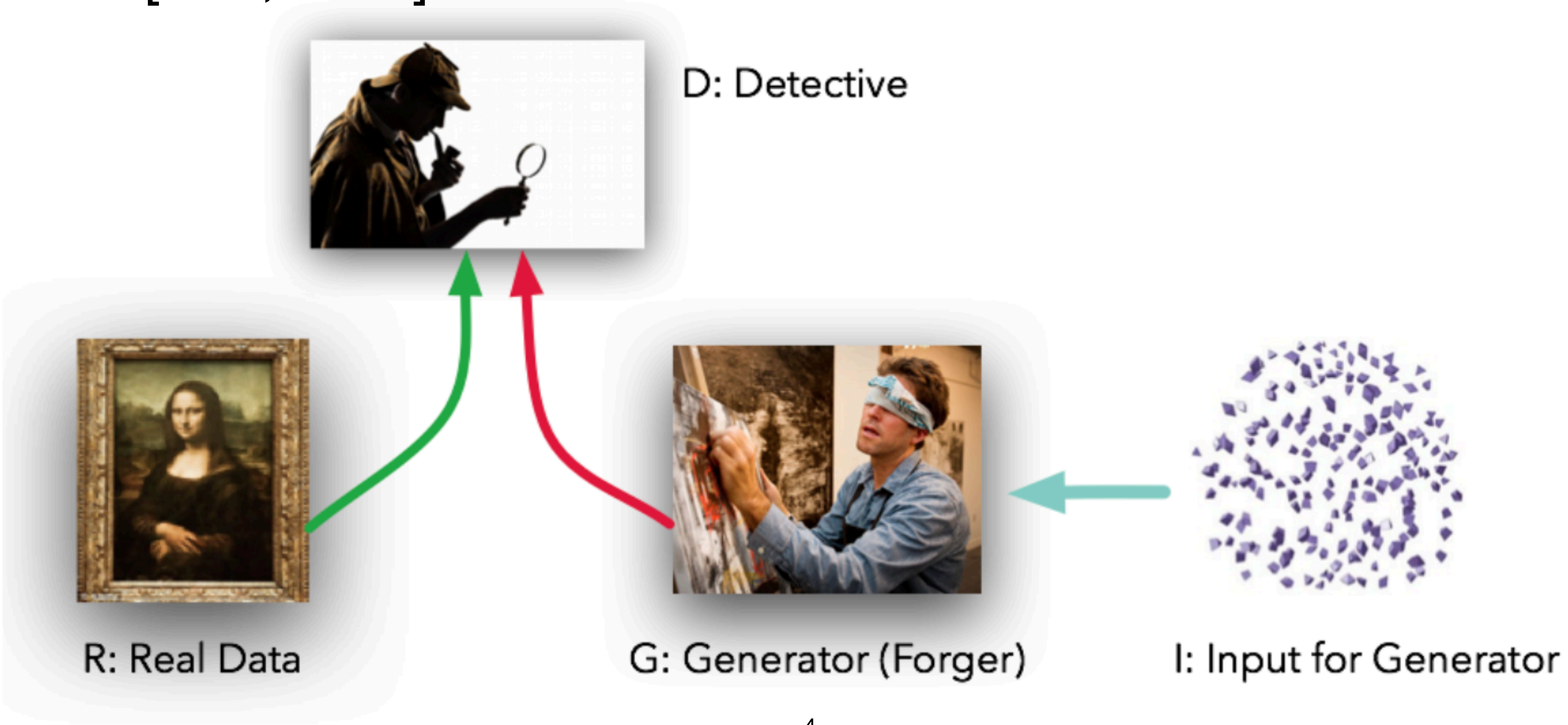
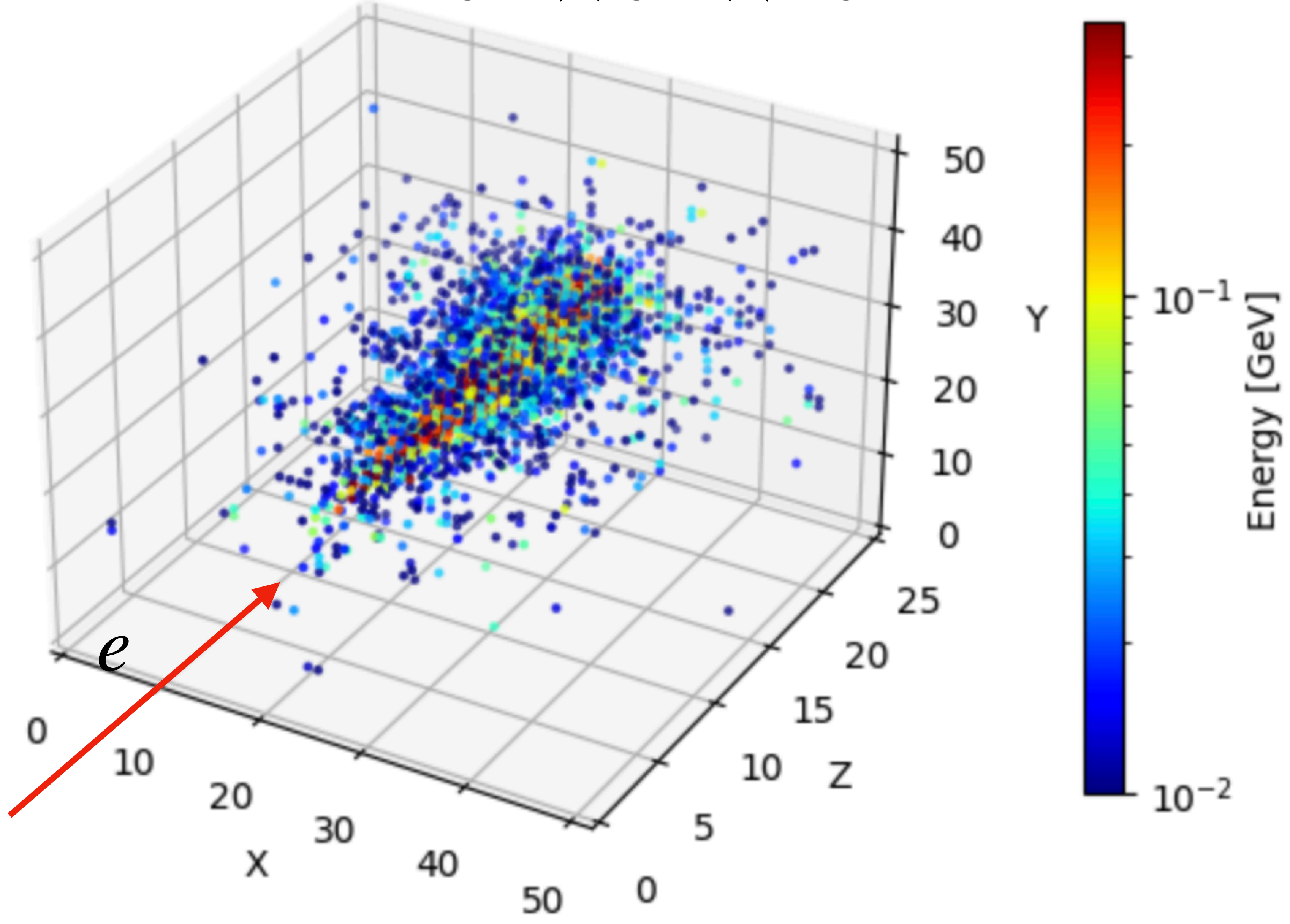


image source

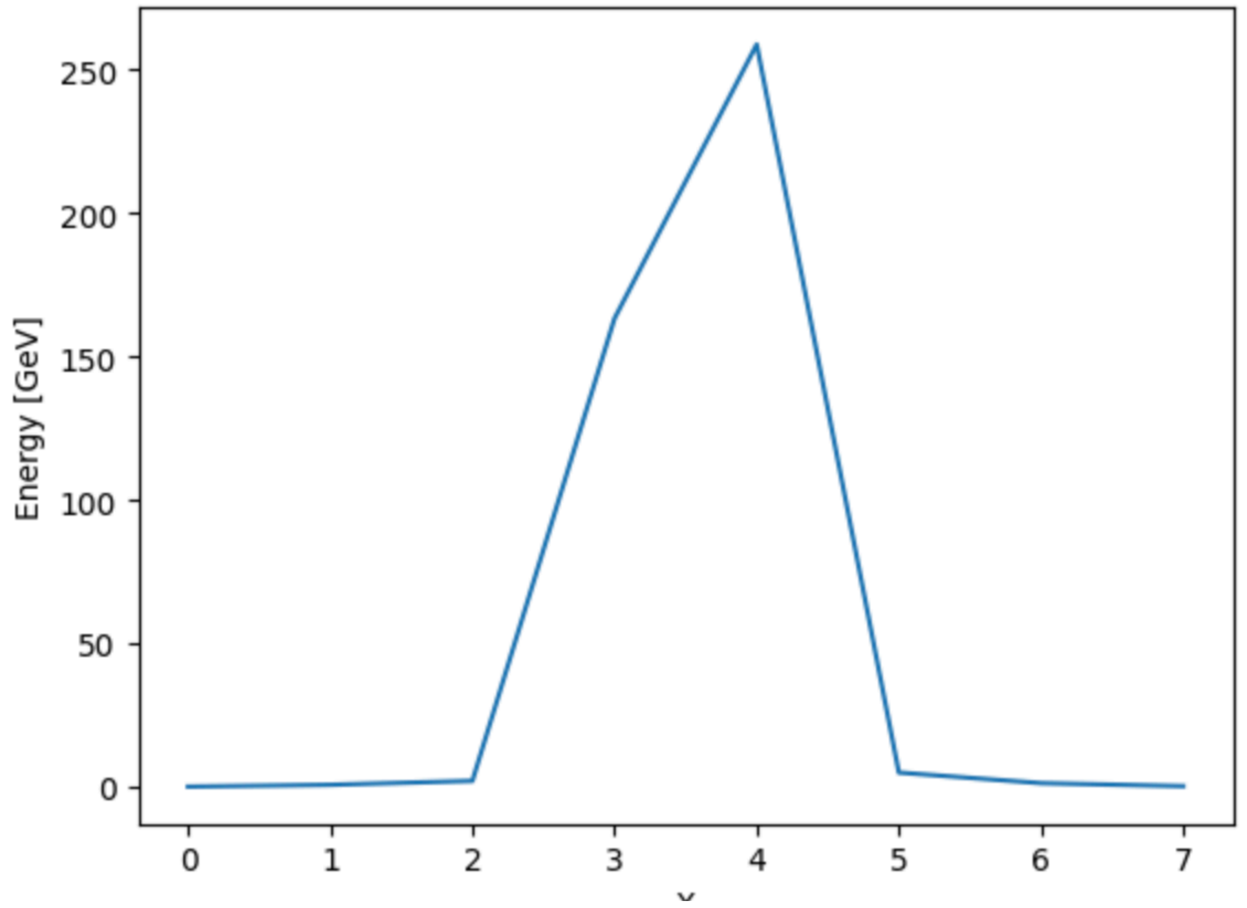
Data Sample

- CLIC Calorimeter images: energy deposits from electrons
- 3D ($51 \times 51 \times 25$): too large for the current quantum device
  - downsampled to 8 pixels
  - downsampled to 64 pixels (8×8)

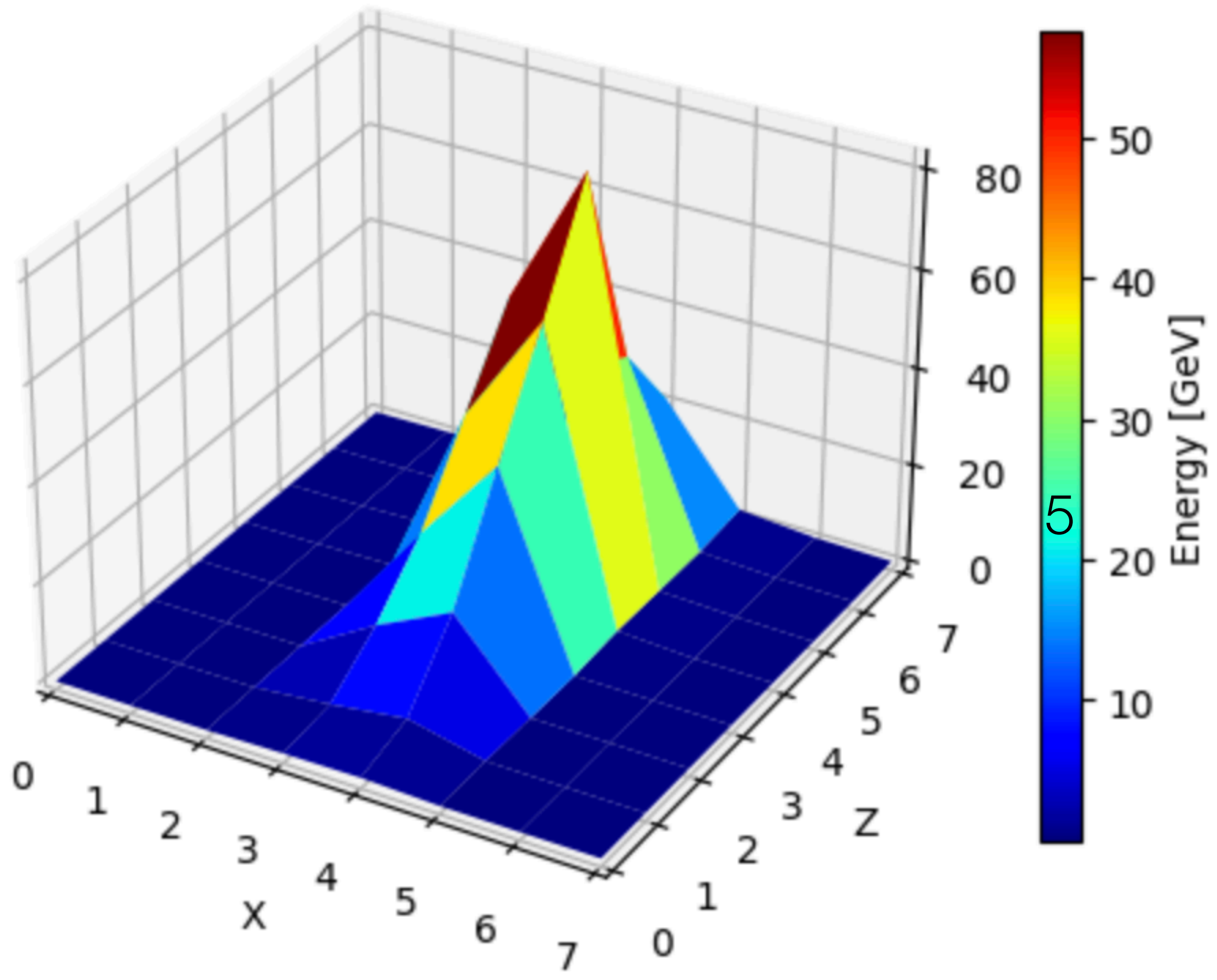
$51 \times 51 \times 25$



8



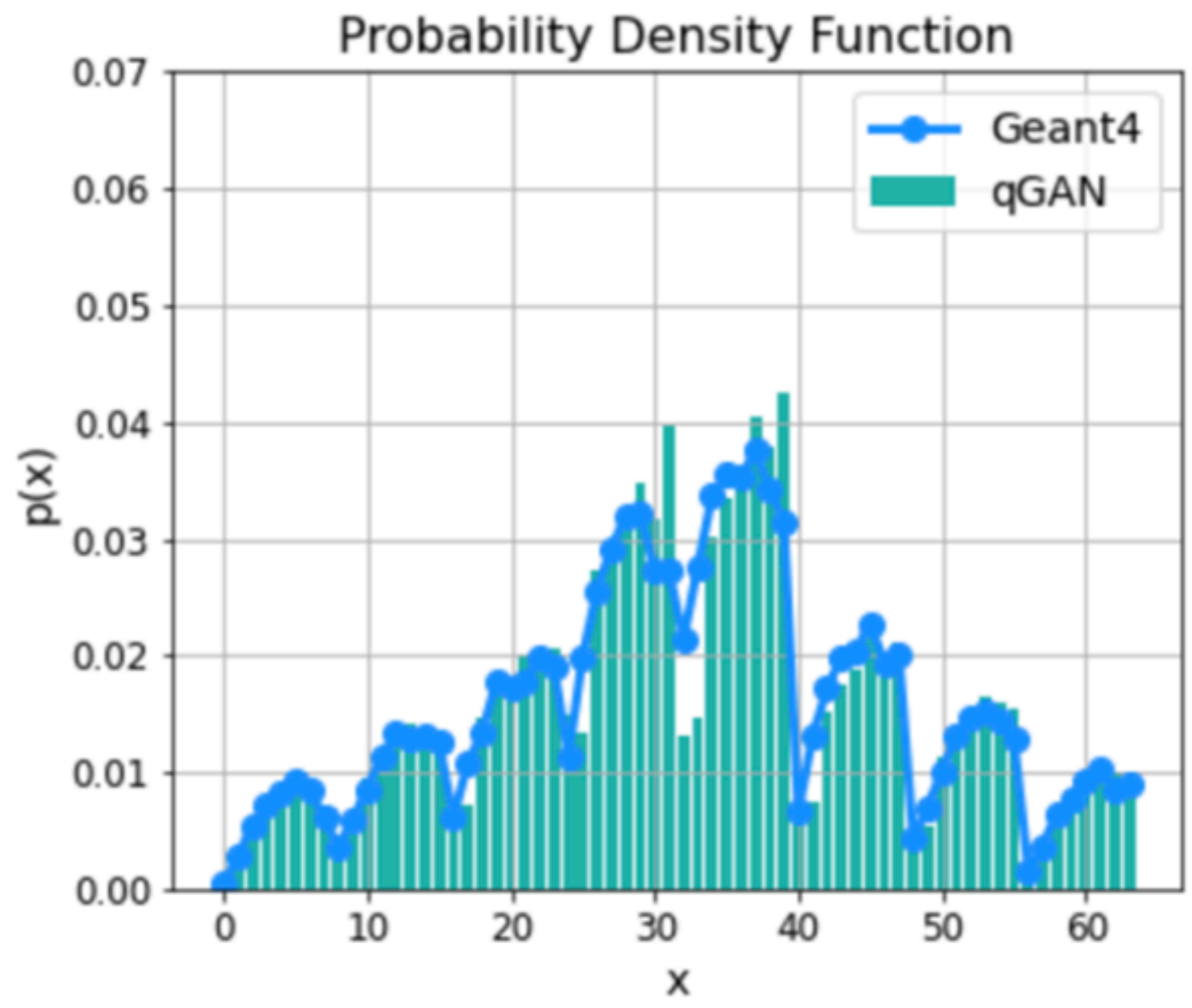
8×8

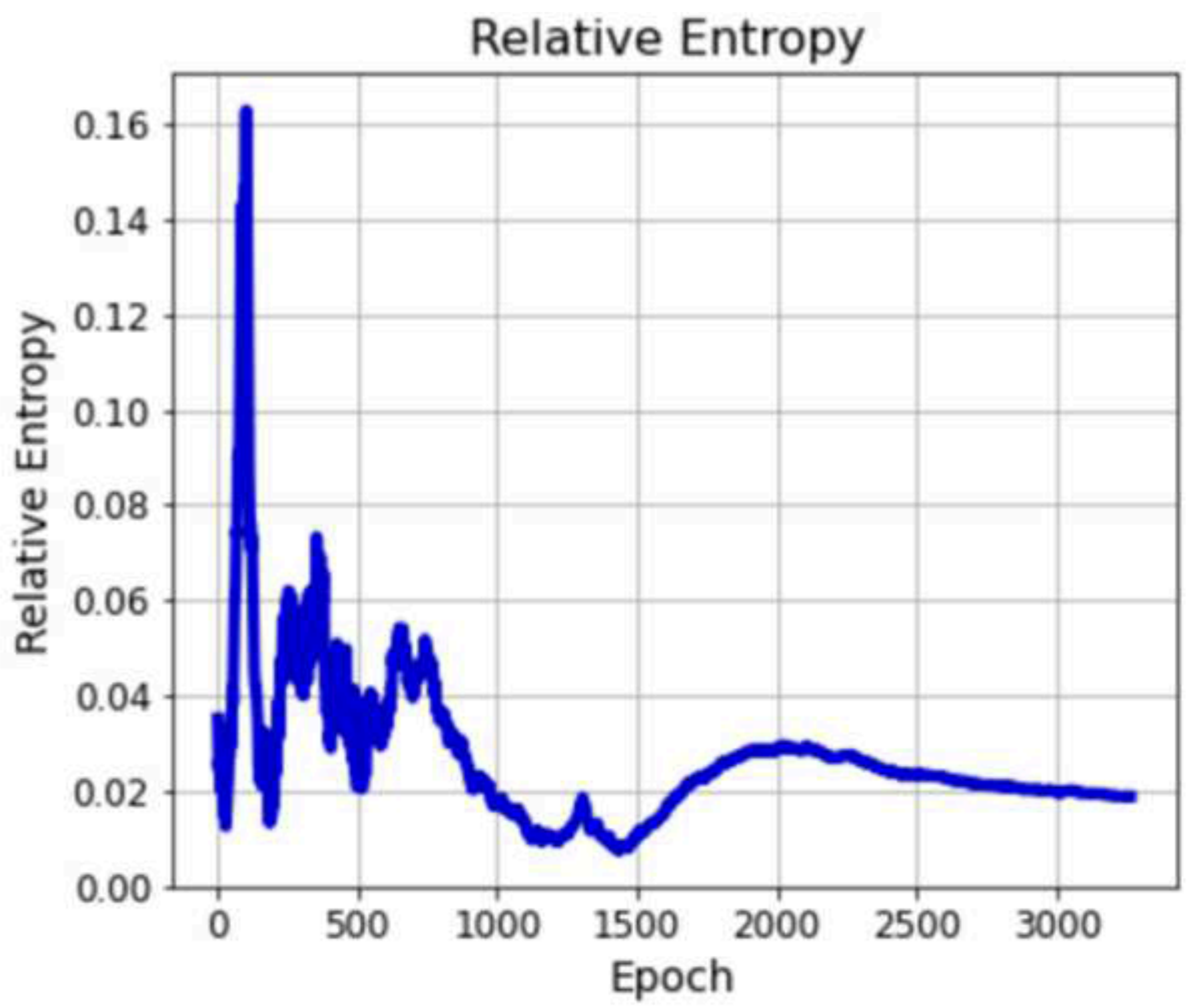


Average Shower Image (PDF)

Research Status

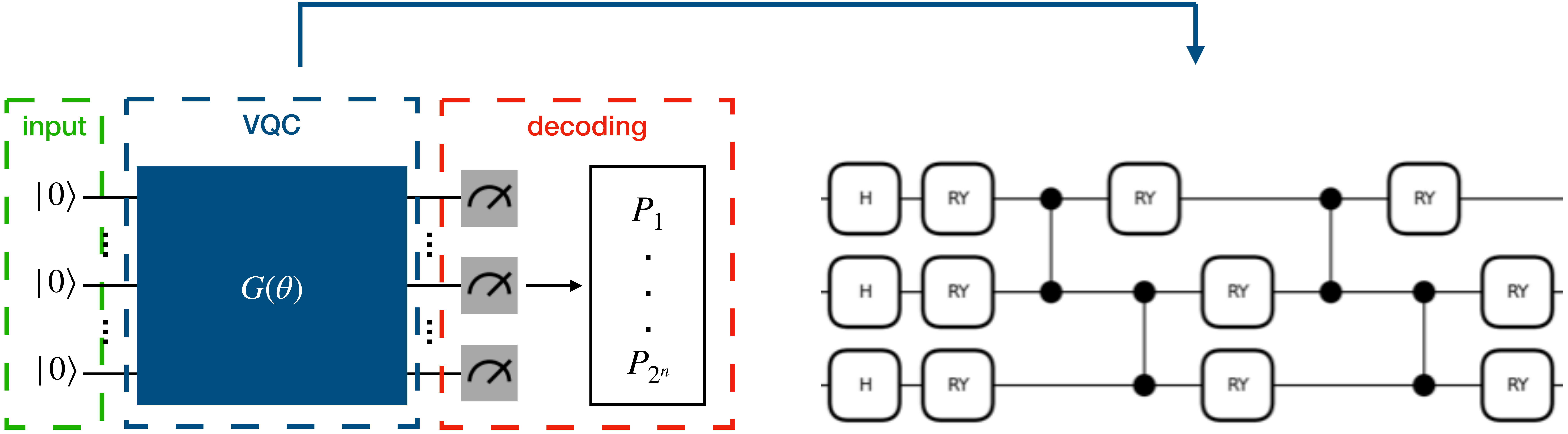
- DESY & CERN started the project about 4 years ago
- Successfully generated the average shower shape
  - 1D image with 8 pixels: good performance
  - 2D image with 64 pixels: training is unstable





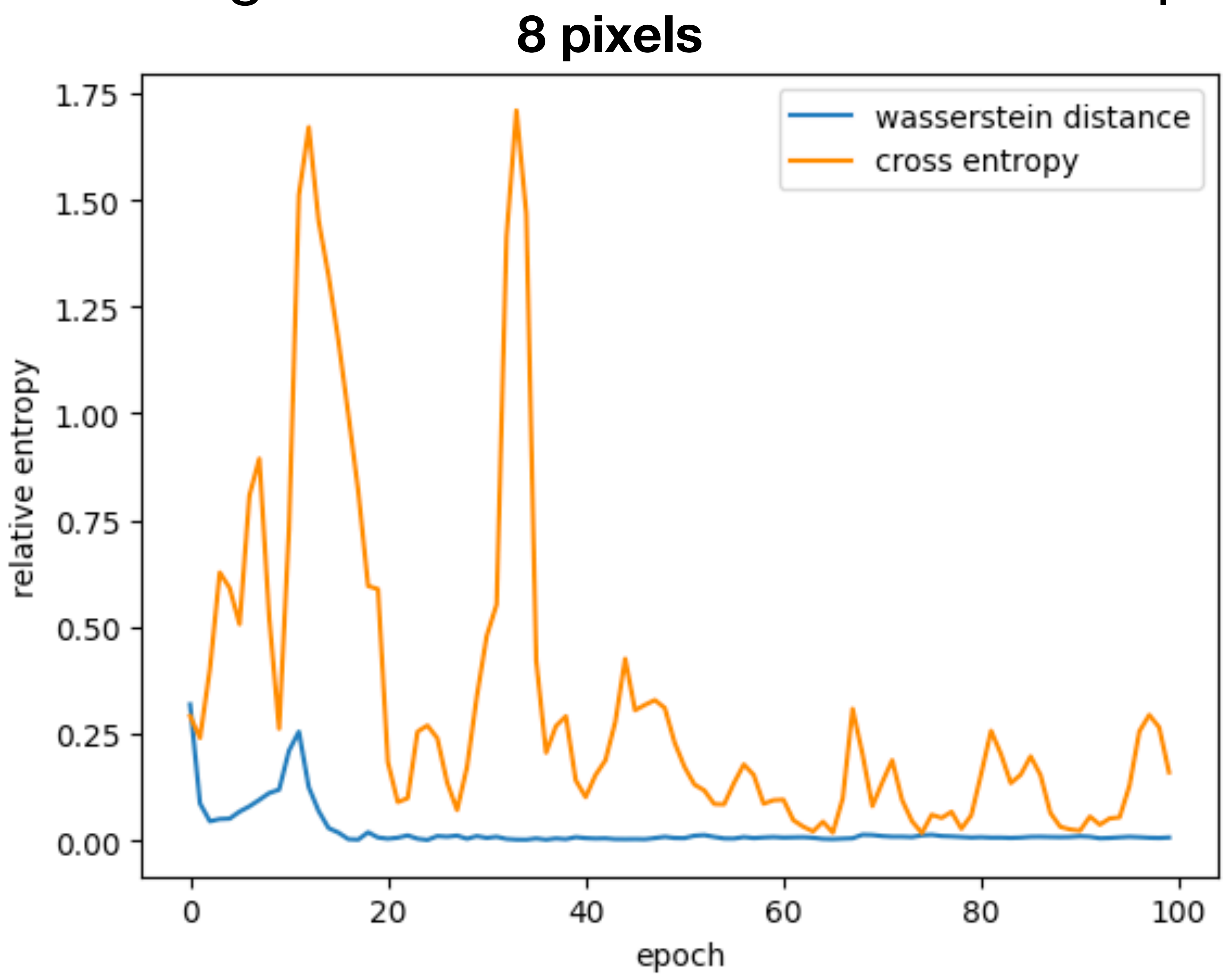
Generator Architecture

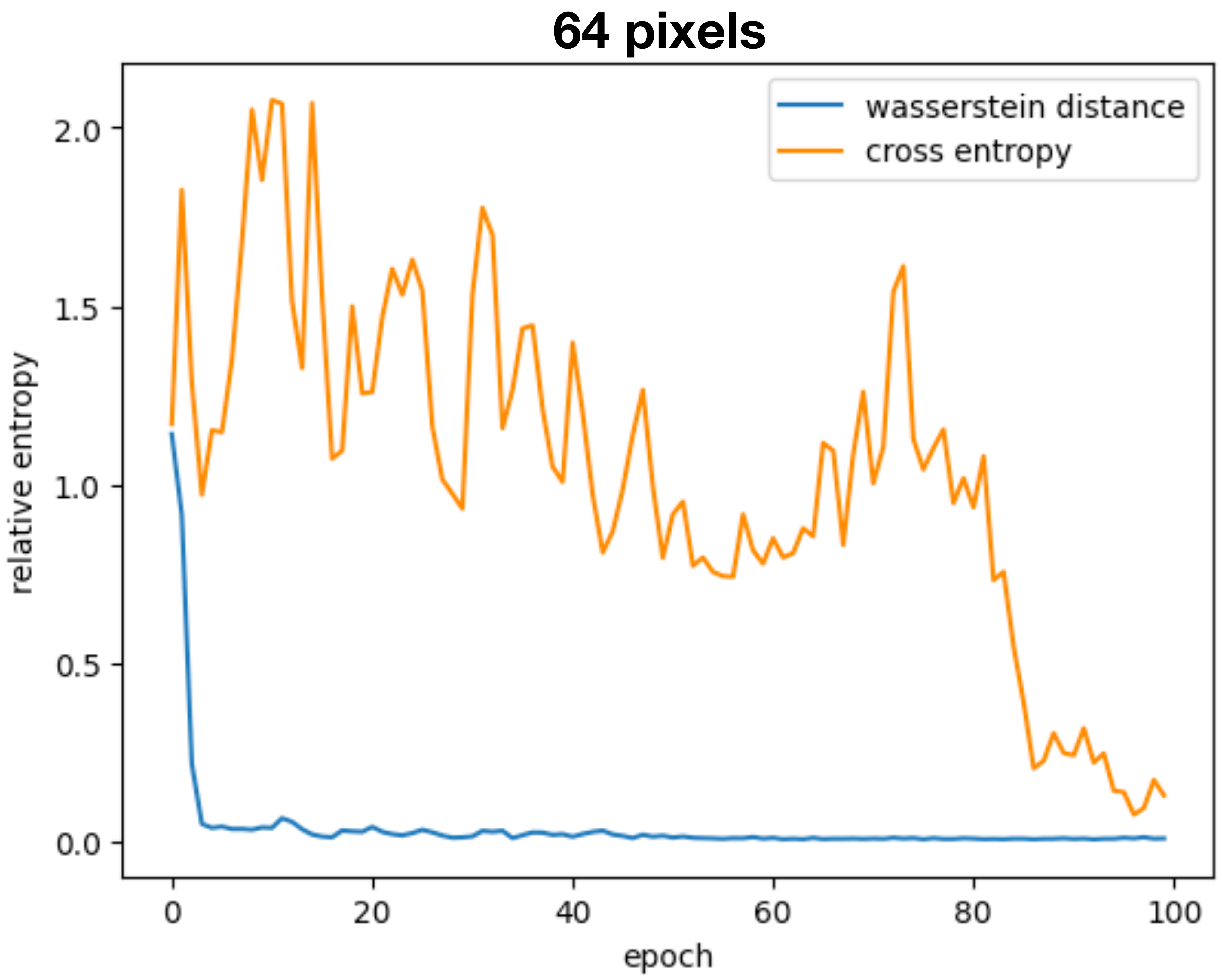
- Input states: $|0\rangle^{\otimes n}$
- Variational quantum circuits: $G(\theta)|0\rangle^{\otimes n} \rightarrow |\psi\rangle$
- Amplitude decoding: n qubits $\rightarrow 2^n$ amplitudes $\rightarrow 2^n$ PDF values
  - 8 pixels: 3 qubits
  - 64 pixels: 6 qubits



Training: Cross Entropy vs Wasserstein Loss

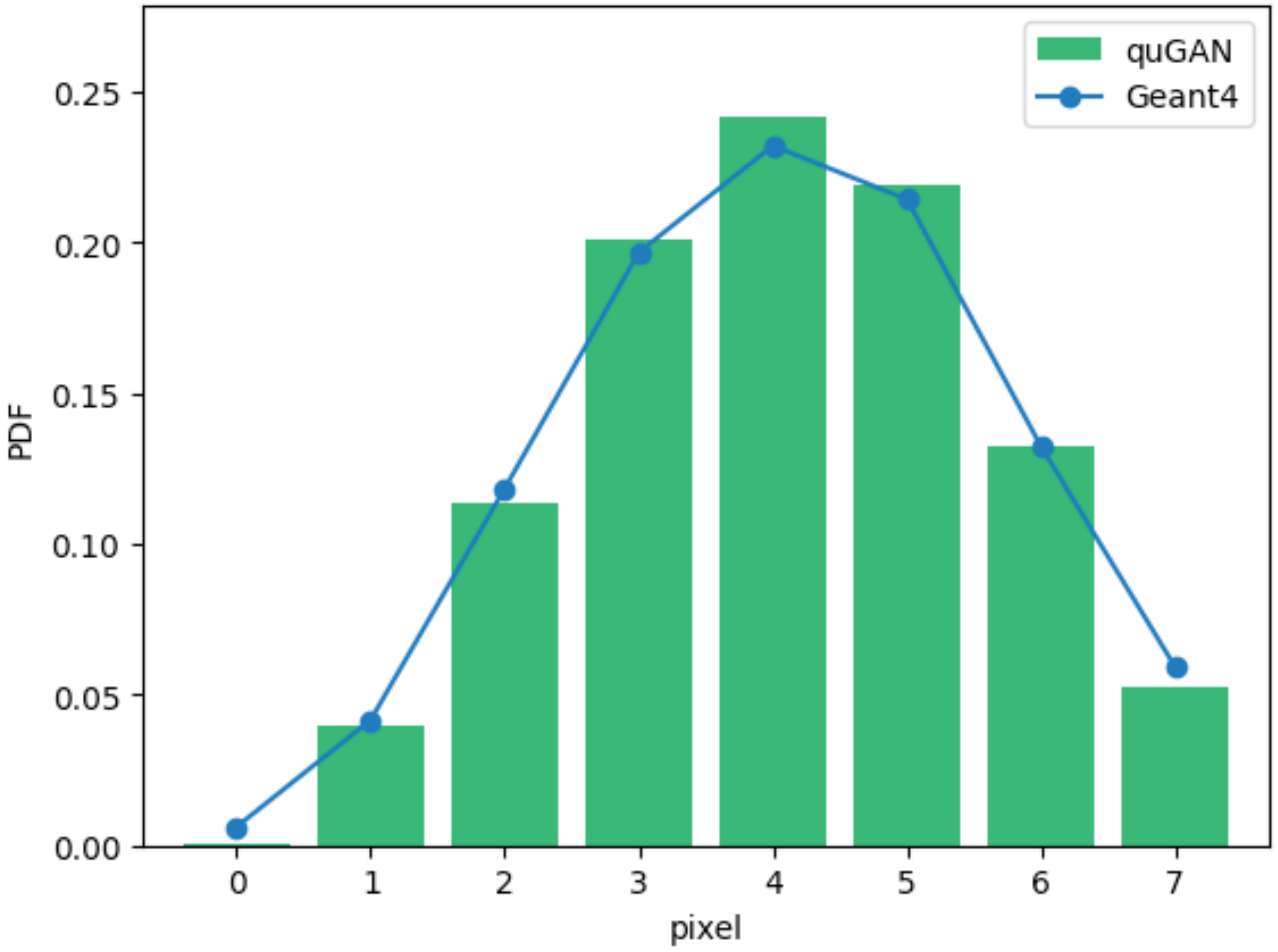
- Training with Wasserstein distance is more stable than cross-entropy loss
- Hyper-parameter optimization could help when using cross-entropy loss
  - time-consuming …
  - training fluctuates for the data with 64 pixels

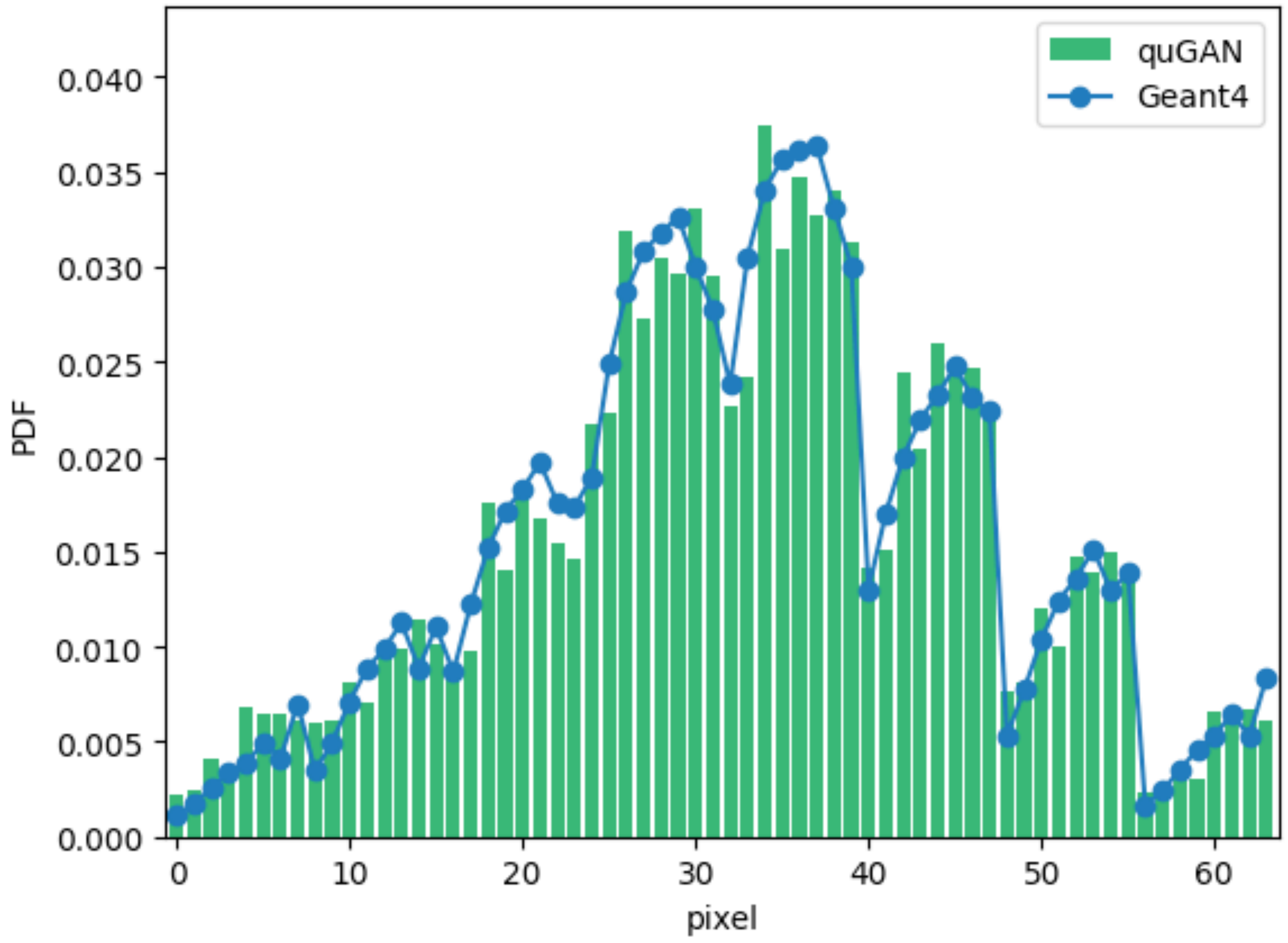




Performance (Ideal Simulator)

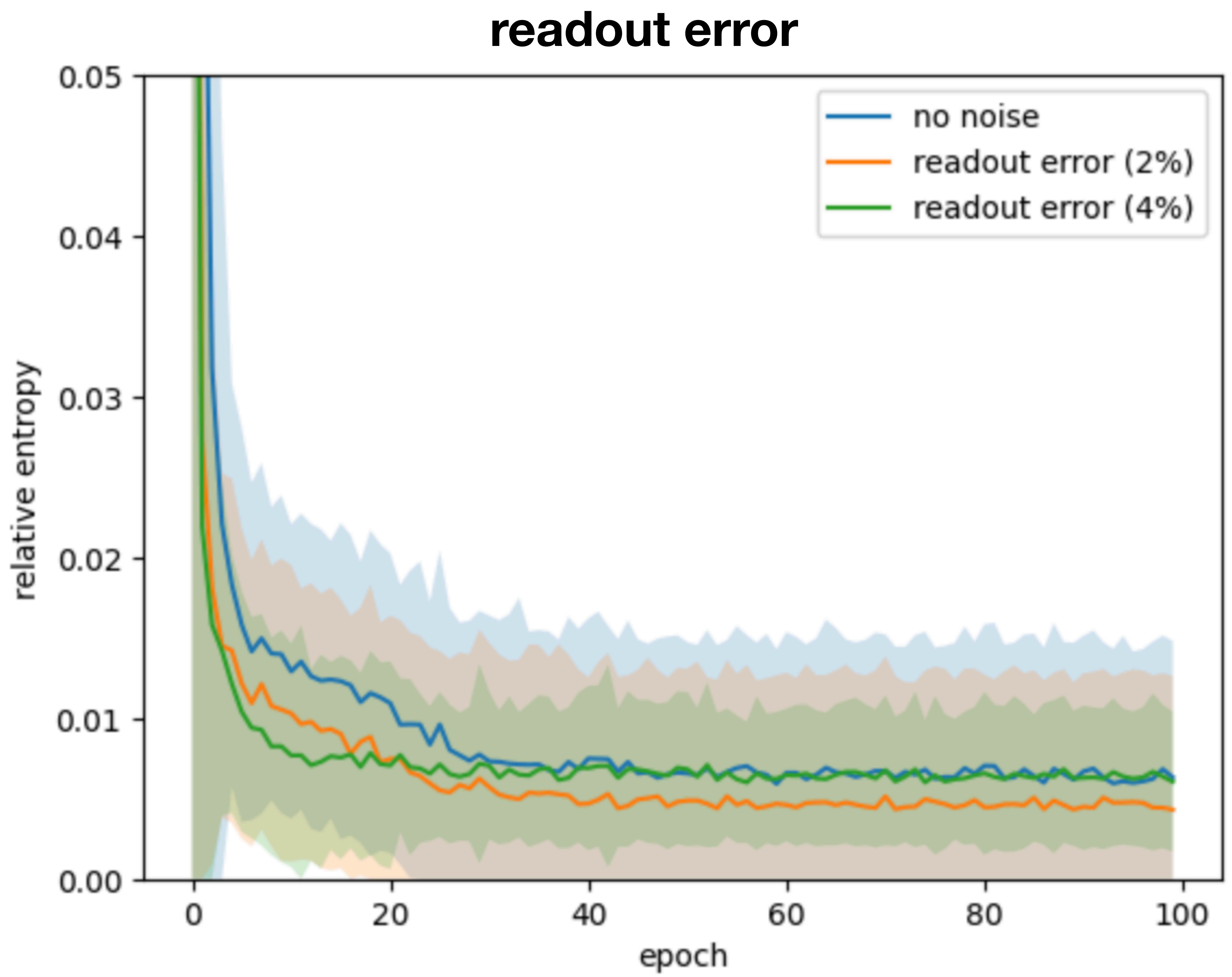
- Generated data are consistent with Geant4.

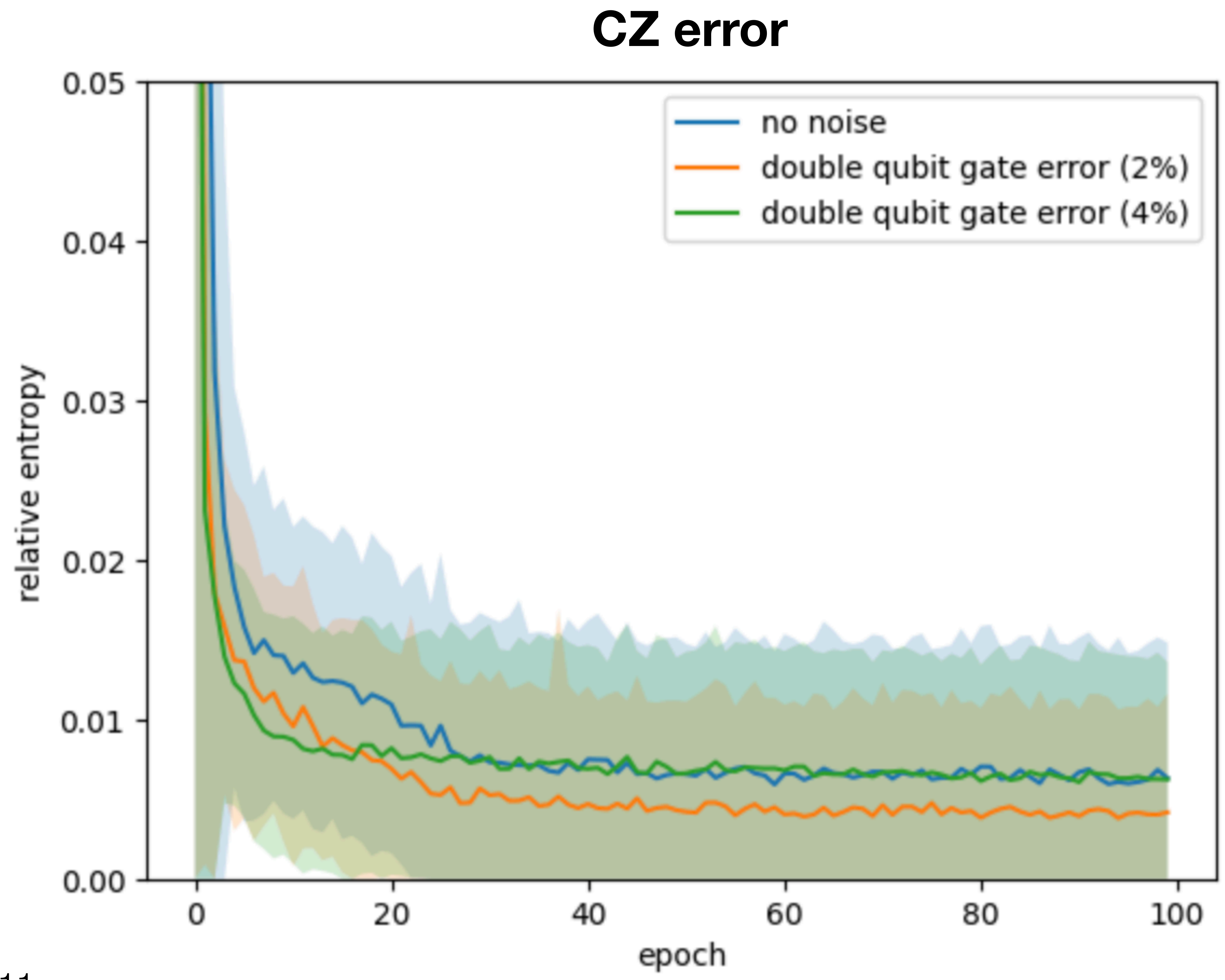




Impact of Noise: Training (8 pixels)

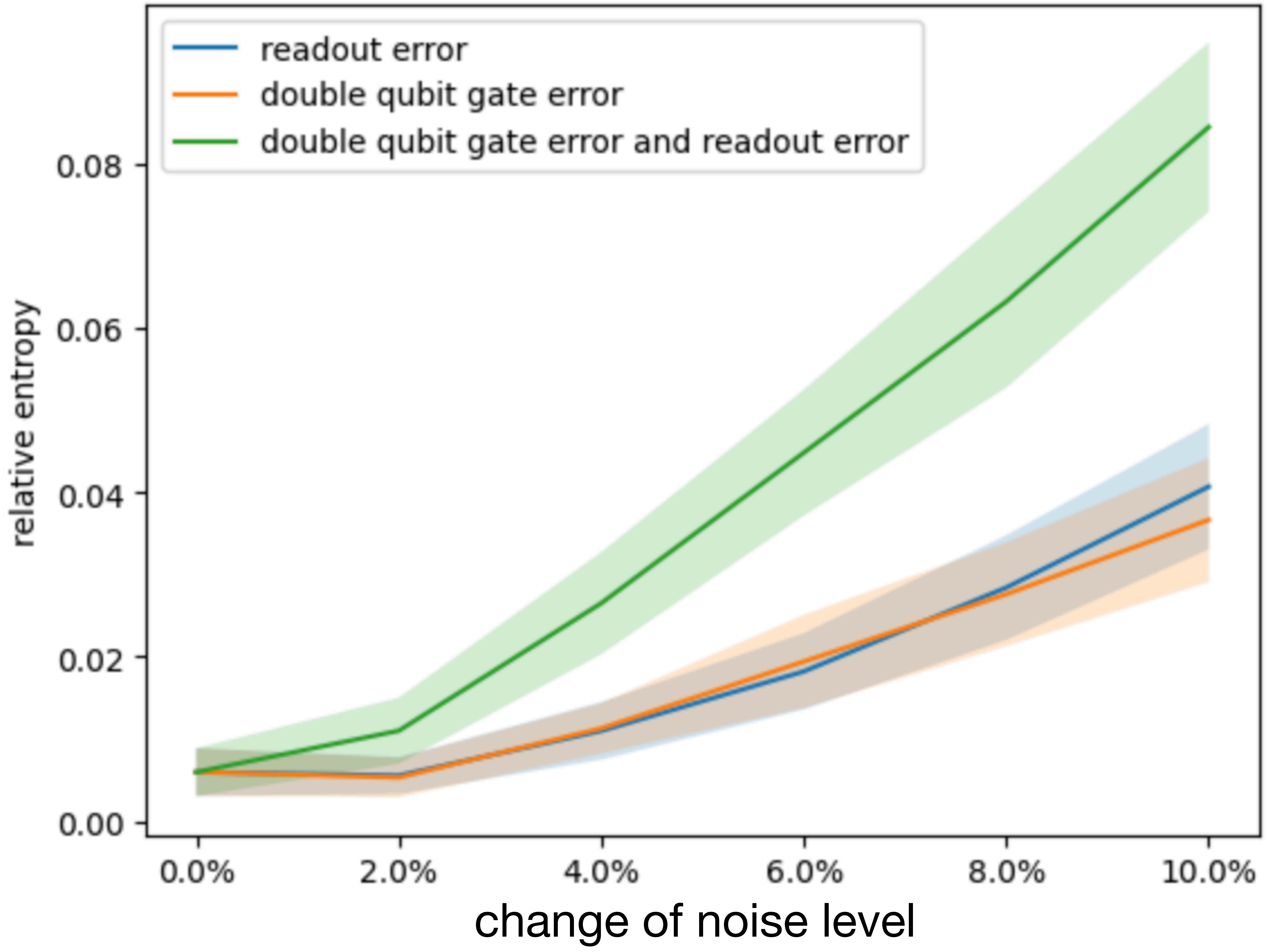
- Consider the impact of readout error and double qubit gate (CZ) error
  - line: mean value
  - band: fluctuation due to the initialization
  - noise (<2%) could improve the training

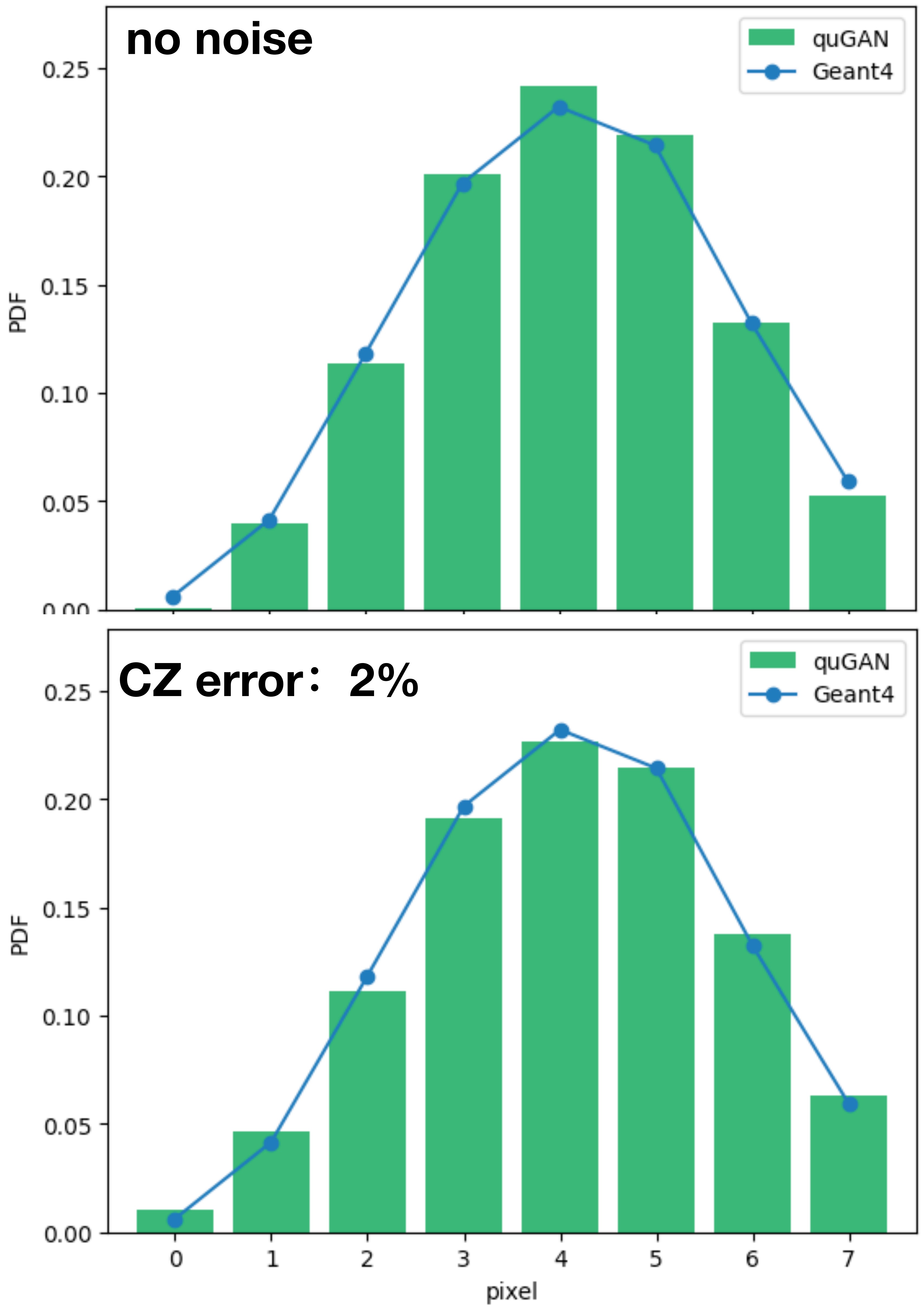




Impact of Noise: Inference (8 pixels)

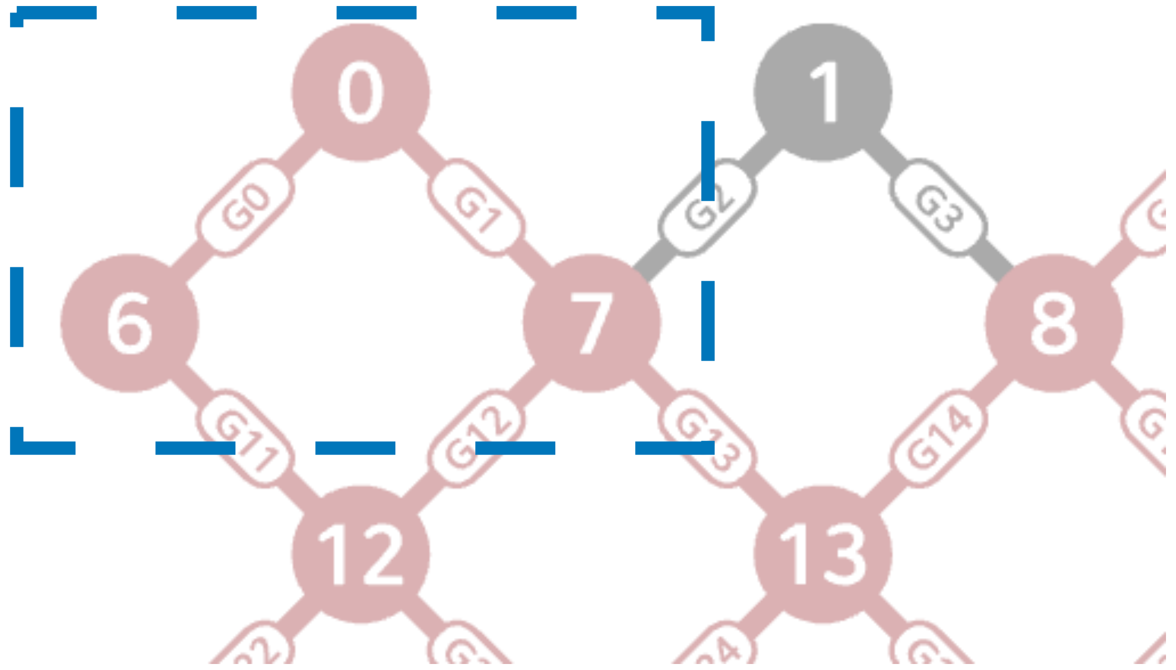
- Noise level at the inference stage may change
  - stable performance when the change is small



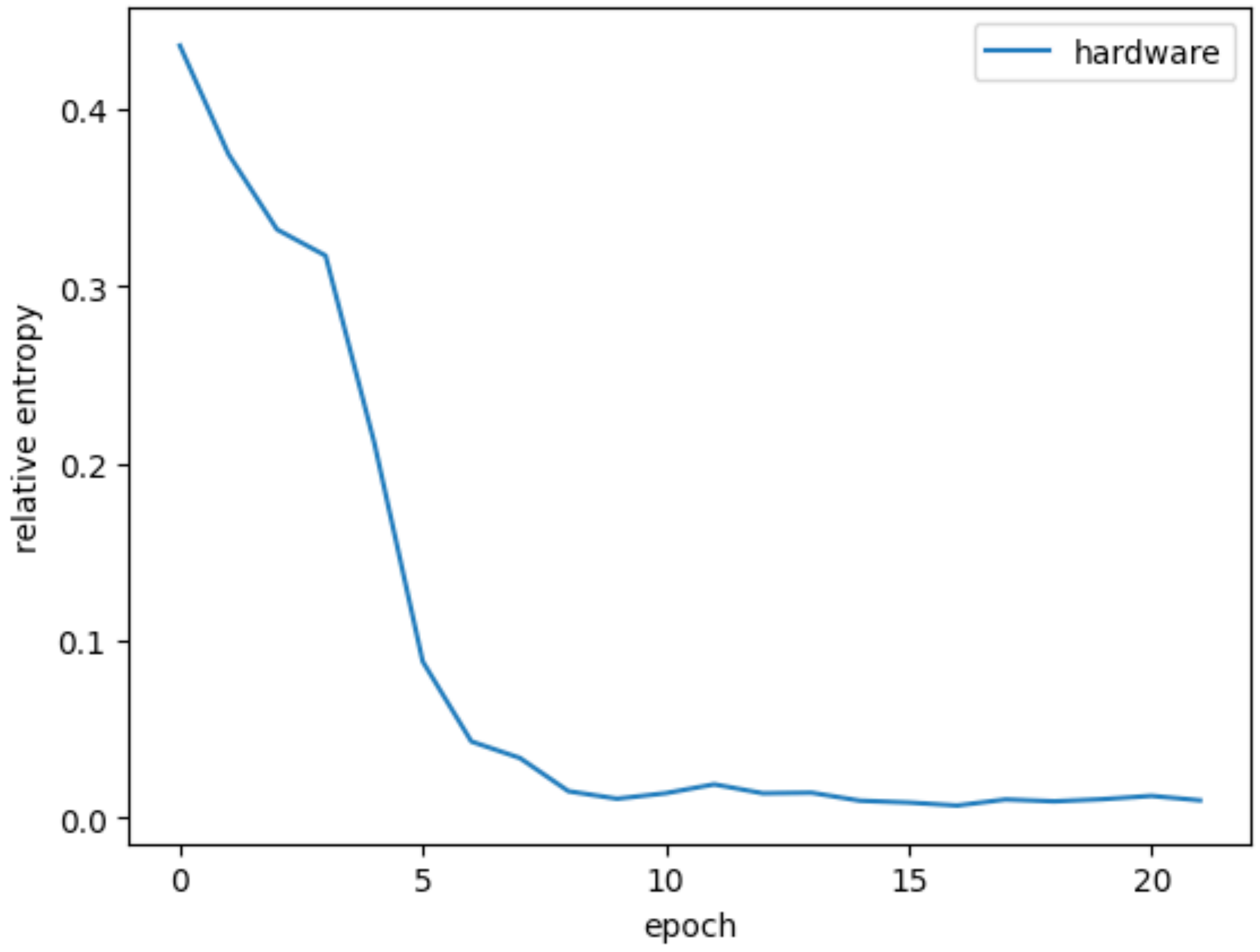


Results on the Hardware (8 pixels)

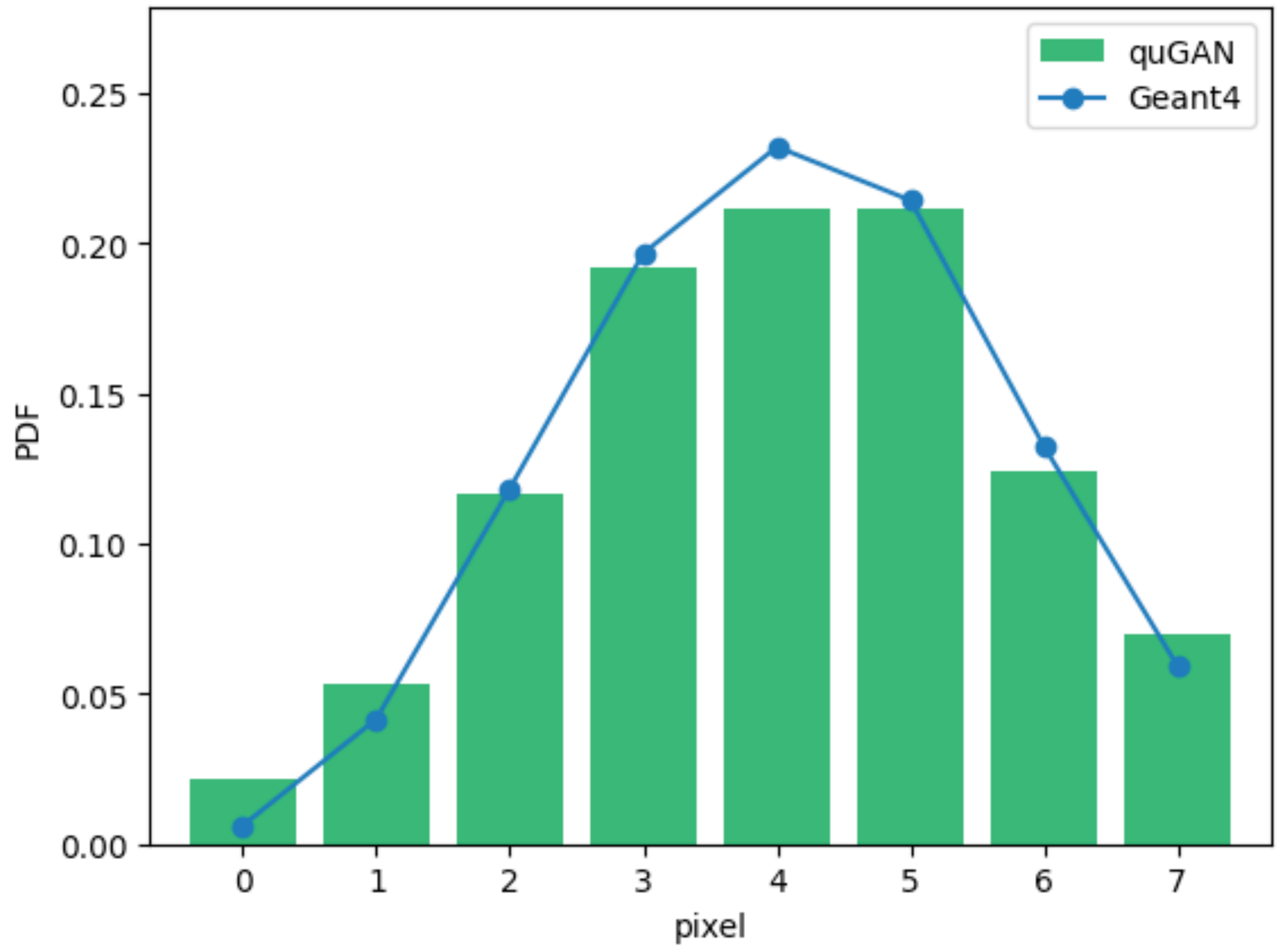
- Test the model on the hardware (Xiaohong: 骁鸿)
  - CZ error: 2%
  - readout error: 2%



training process



performance



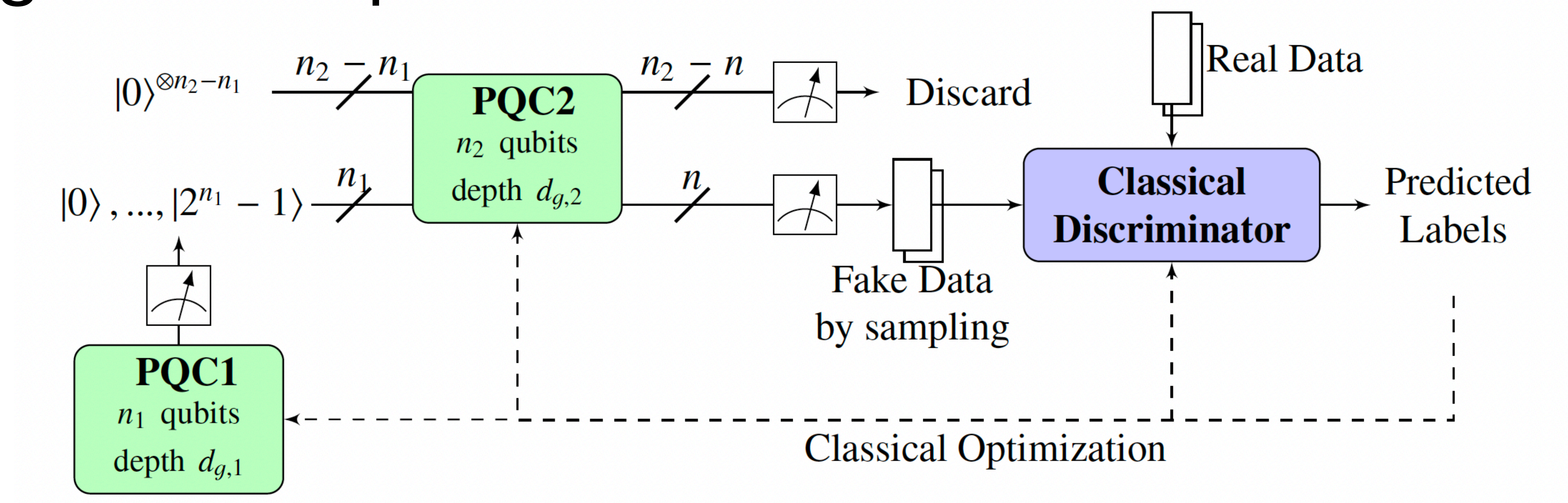
Actual Shower Image

Research Status

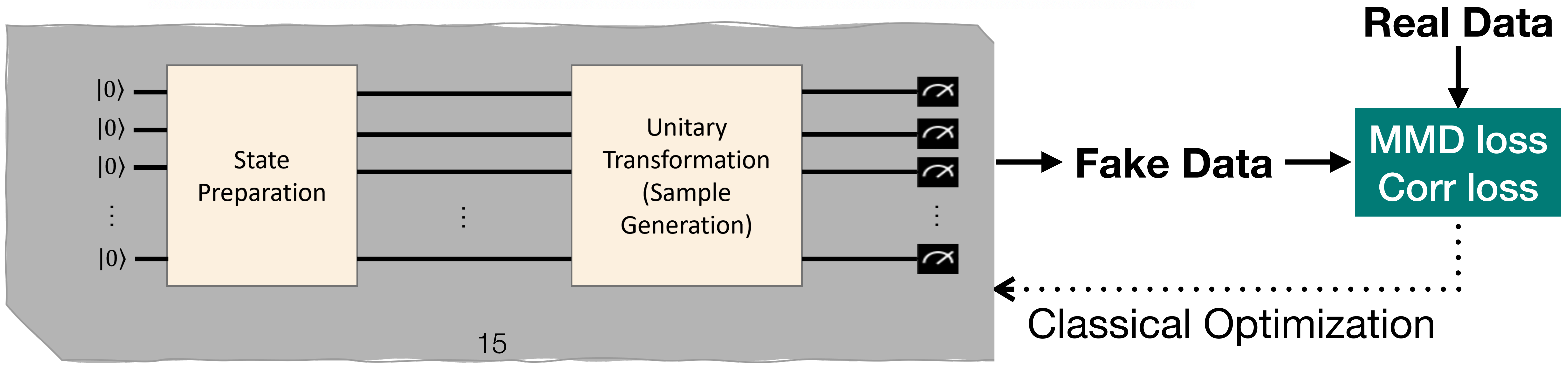
- CERN openlab: dual-PQC GAN
  - hard to train
  - poor results for images with more than 4 pixels

- CERN & DESY: Quantum Angle Generator (QAG)
  - use Maximum Mean Discrepancy (MMD) as a simple discriminator (non-trainable), easy to train
  - relatively good results for images with 8 pixels

<u>dual-PQC GAN</u>
(CERN openlab)

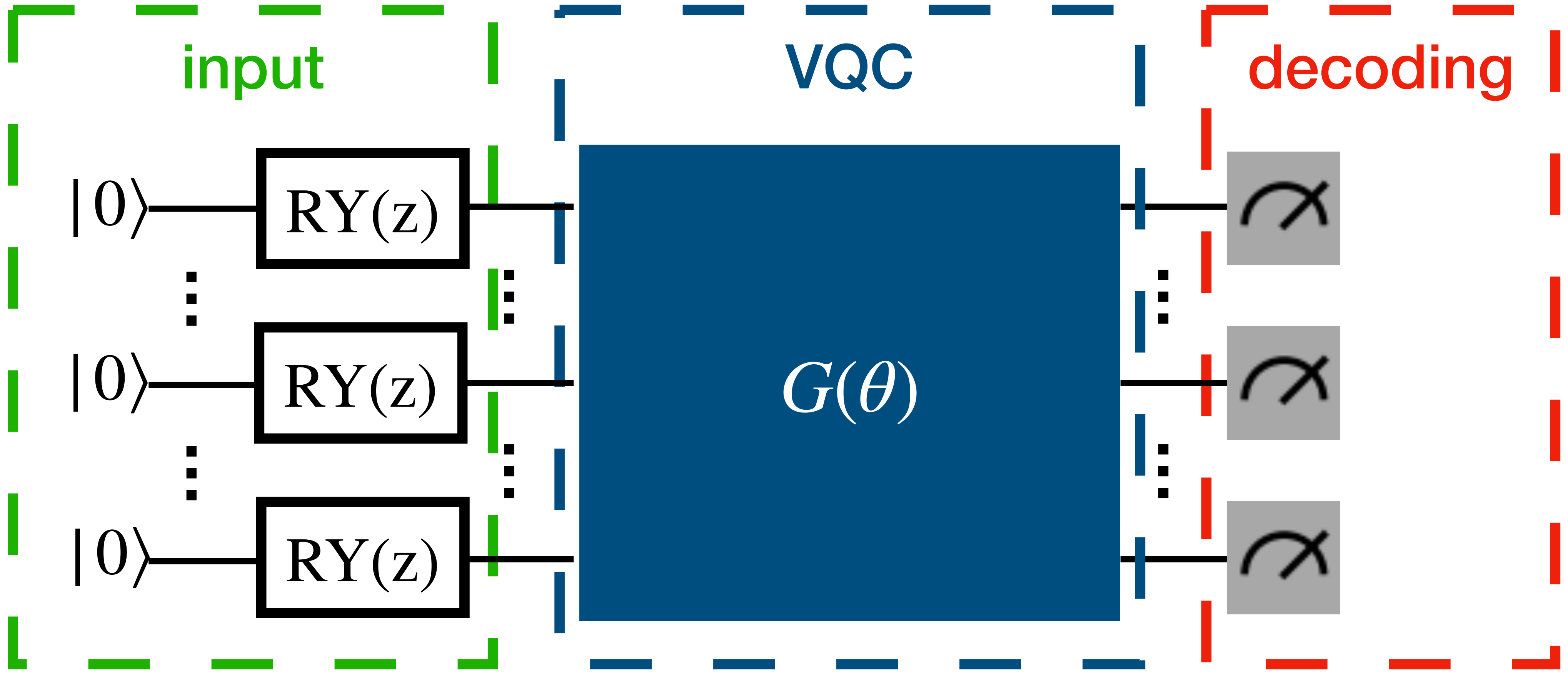


<u>QAG</u>
(CERN & DESY)



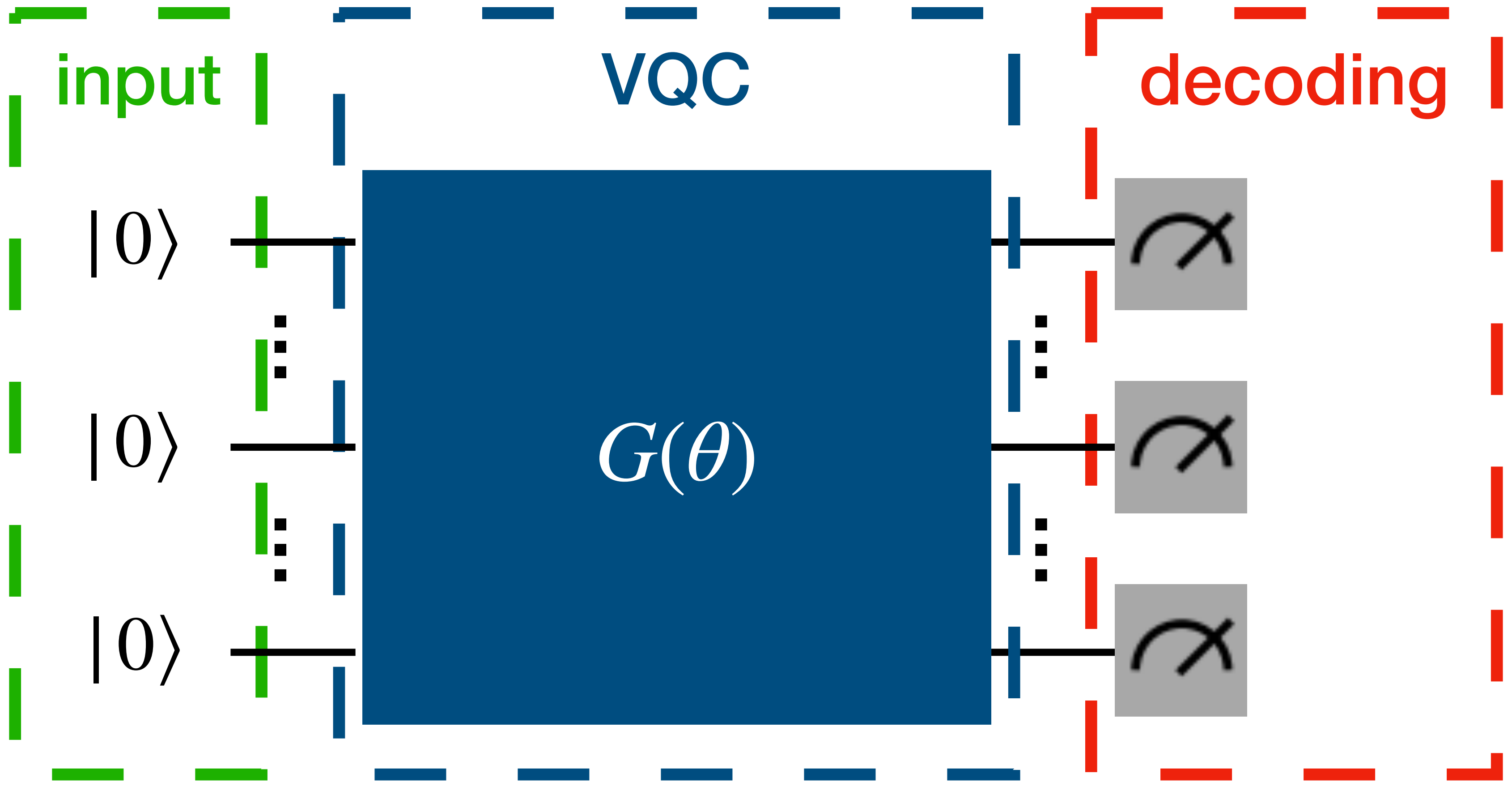
Generator Architecture

- Input states: $R_Y(z)|0\rangle^{\otimes n}$,
  - $z \sim N(0,\ 1)$
- VQC: $G(\theta)R_Y(z)|0\rangle^{\otimes n} \rightarrow |\psi\rangle$
  - θ: trainable parameters
- Decoding: convert $|\psi\rangle$ to energy



Similar to the classical generator, but limited expressibility

- Input states: $|0\rangle^{\otimes n}$
- VQC: $G(\theta)|0\rangle^{\otimes n} \rightarrow |\psi\rangle, \theta = Wz + b$
  - $z \sim N(0,\ 1)$
  - W, b: trainable parameters
- Decoding: convert $|\psi\rangle$ to energy



Embed the latent vectors multiple times to enhance the expressibility

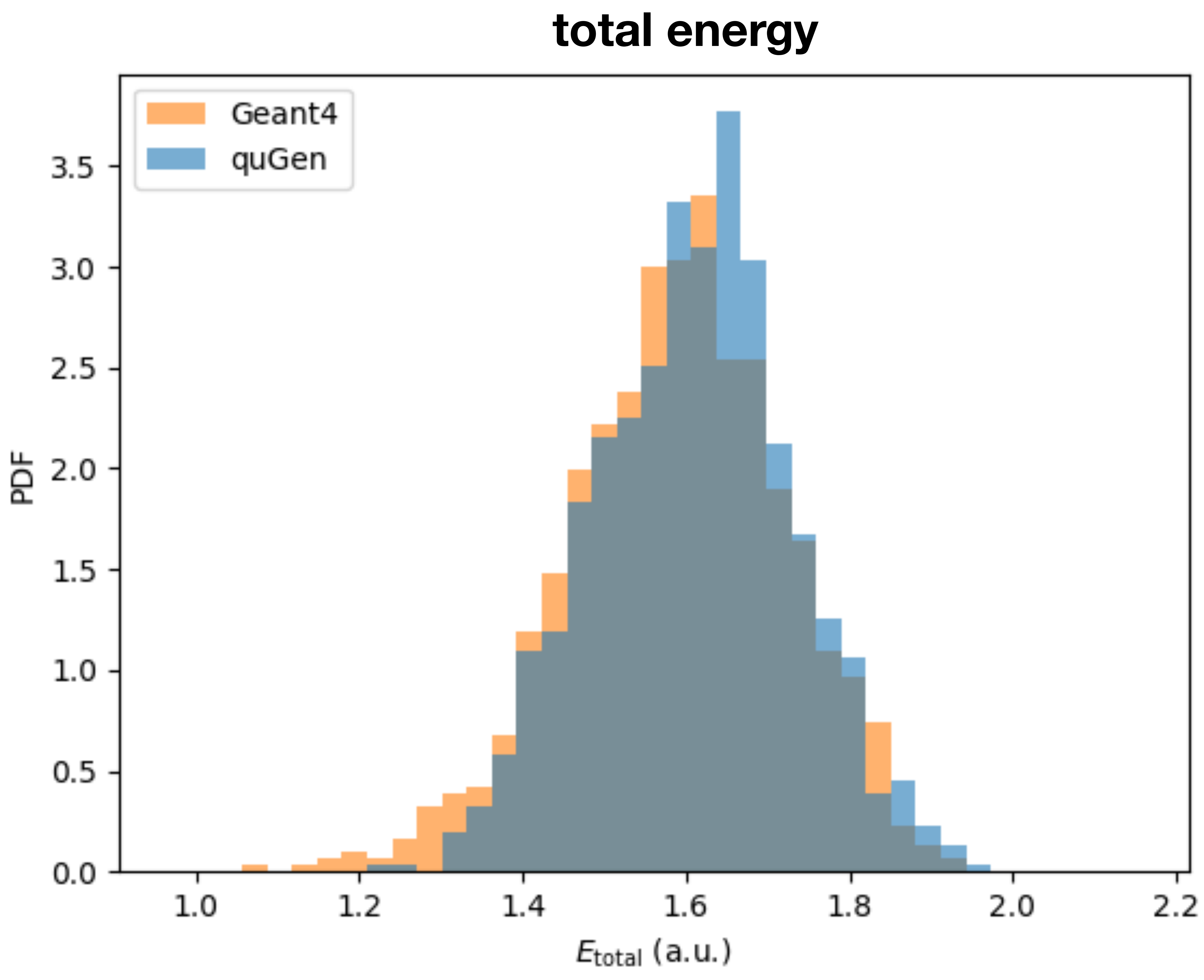
Training Strategy: Warm-up Initialization

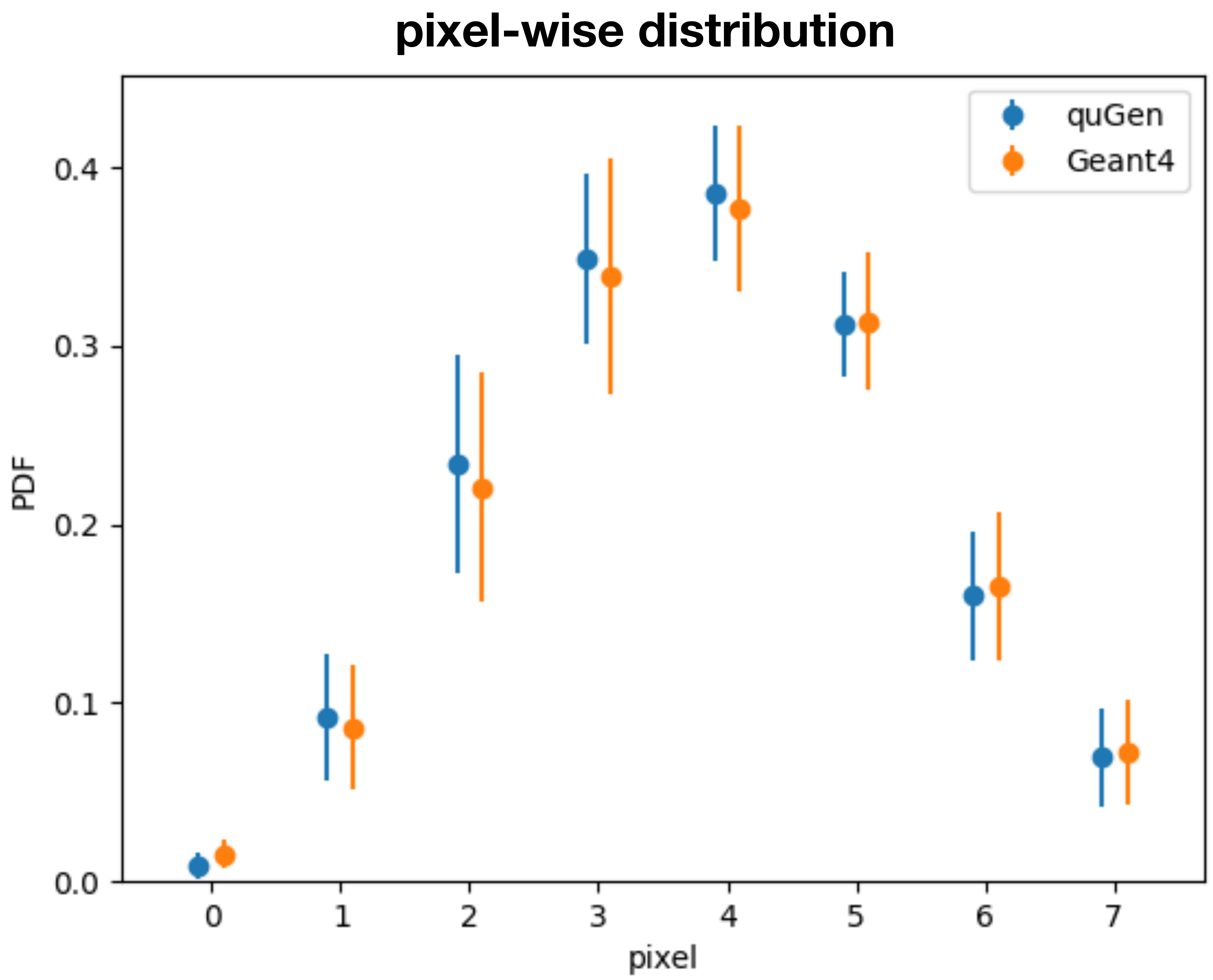
- Warm-up initialization:
  - Step 1: use MMD loss to pre-train the generator
  - Step 2: use the pre-trained generator to pre-train the discriminator (fix the parameters of the generator during the training)
  - Step 3: adversarial training with parameters obtained from previous steps

- Benefits:
  - use the non-trainable discriminator, i.e. MMD loss, to pre-train the model → reduce the training time
  - use the trainable discriminator to improve the performance of the generator

Overall Performance (Ideal Simulator)

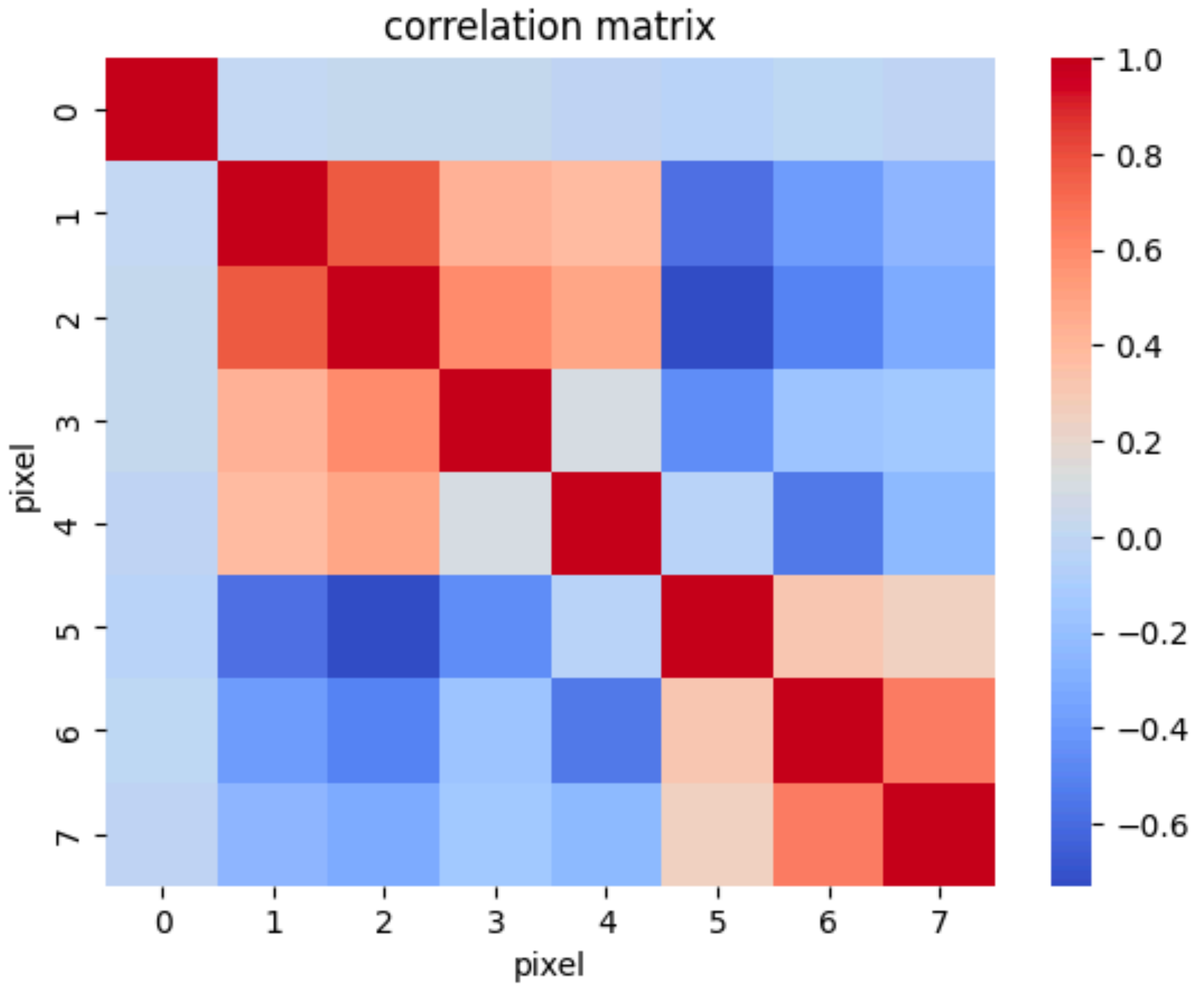
- Consistent distribution between the generated data and Geant4



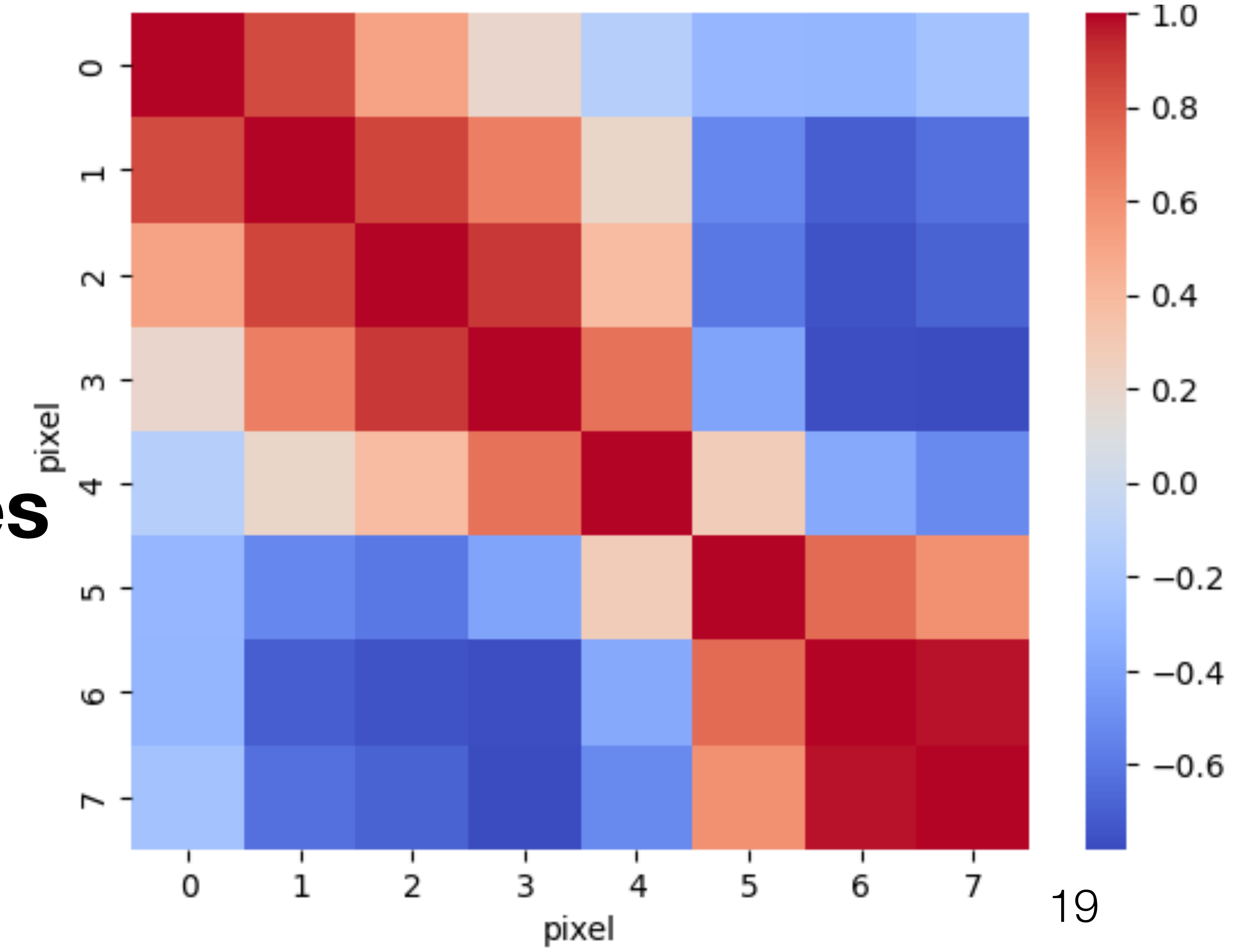


Correlation Matrix (Ideal Simulator)

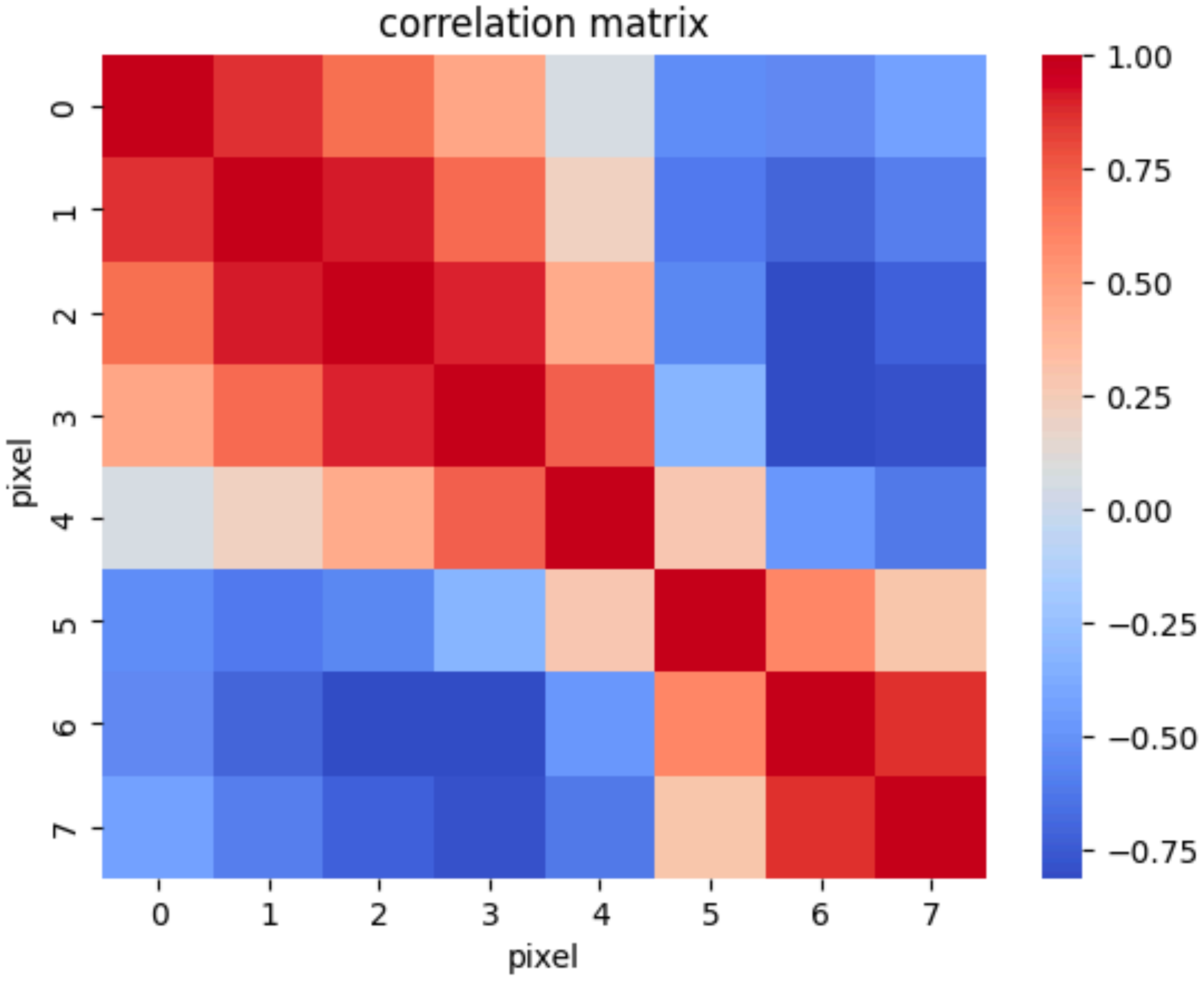
embed the latent vectors one time



embed the latent vectors multiple times



Geant4



Summary

Average shower image:

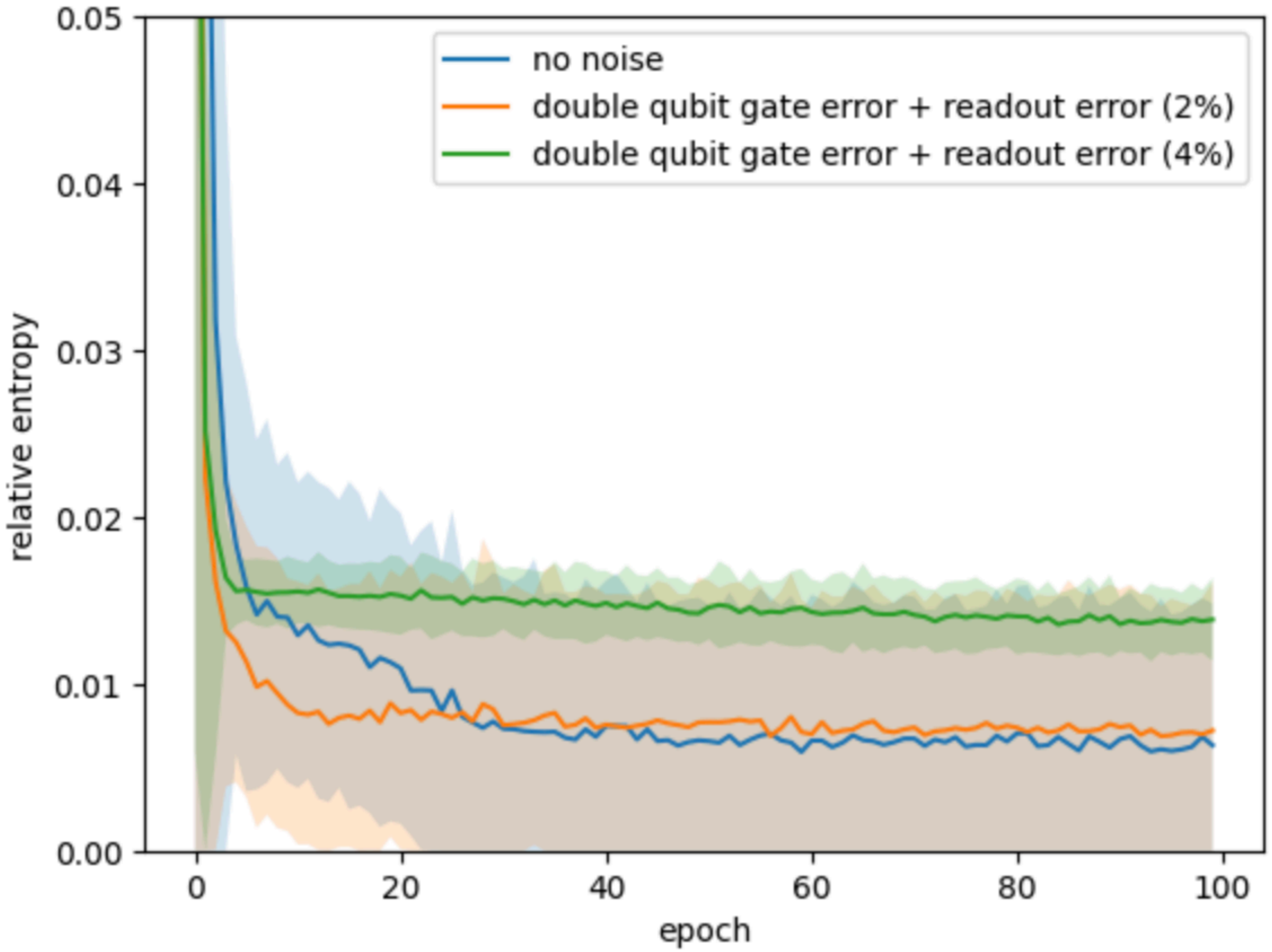
- Quantum GAN could generate images consistent with Geant4
- Training with noise (<2%) improves the performance
- The model inference is stable against noise (<2%)
- Successfully running the model on the hardware (Xiaohong)

Actual shower image:

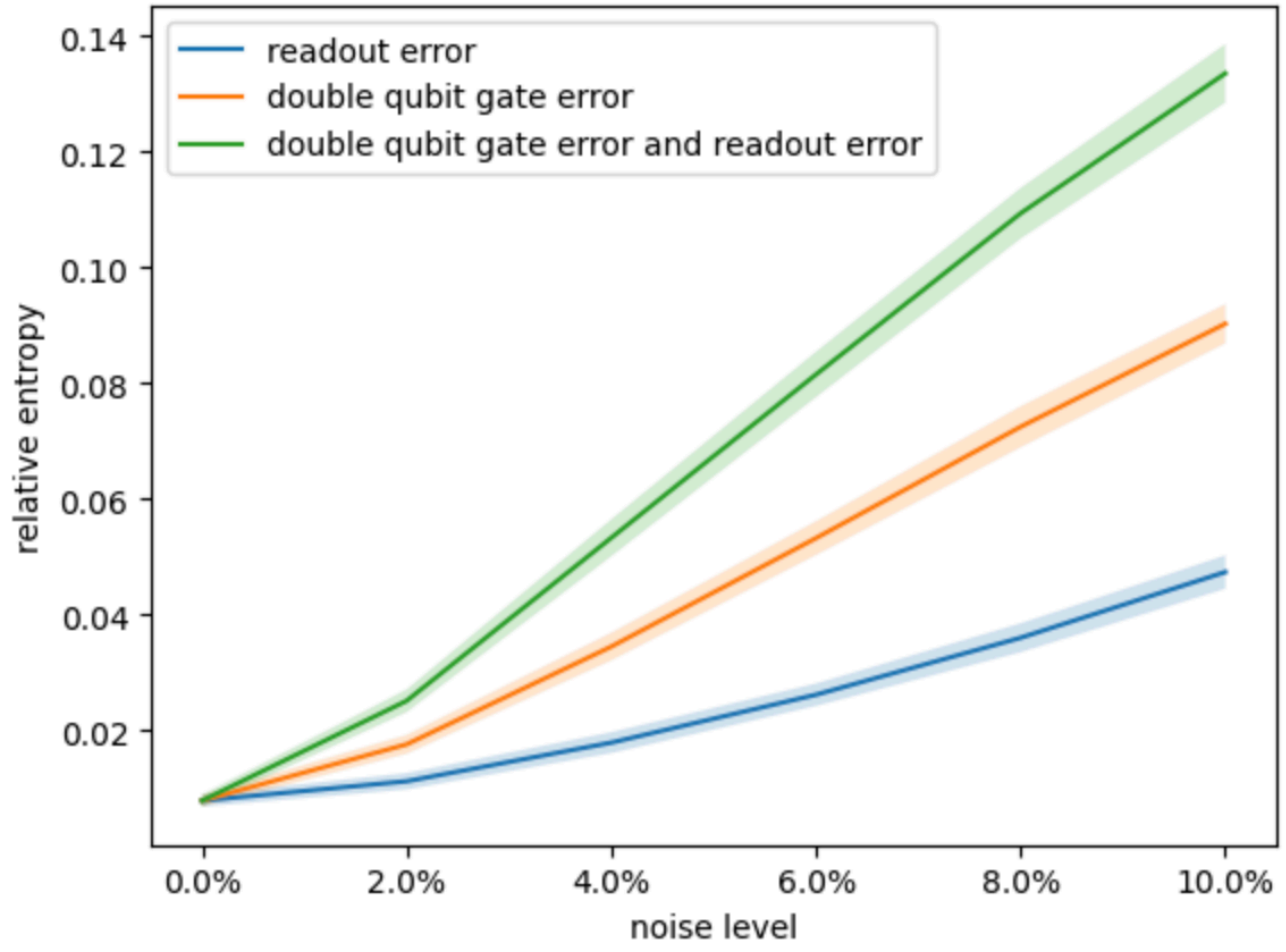
- Reduce the training time via warm-up initialization
- Enhance the expressibility of the generator by embedding the latent vectors into the rotation gates multiple times

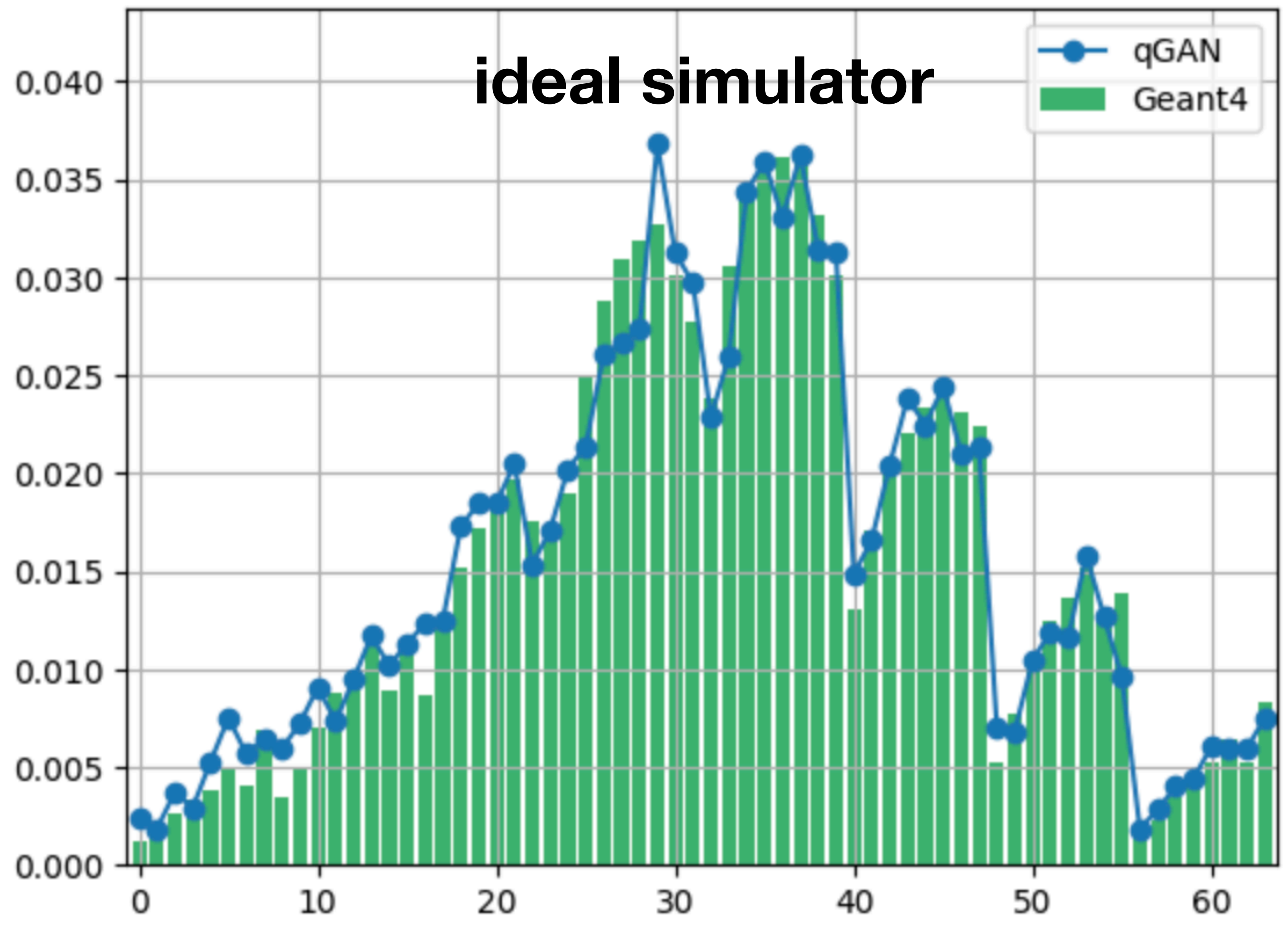
Thank you for your listening!

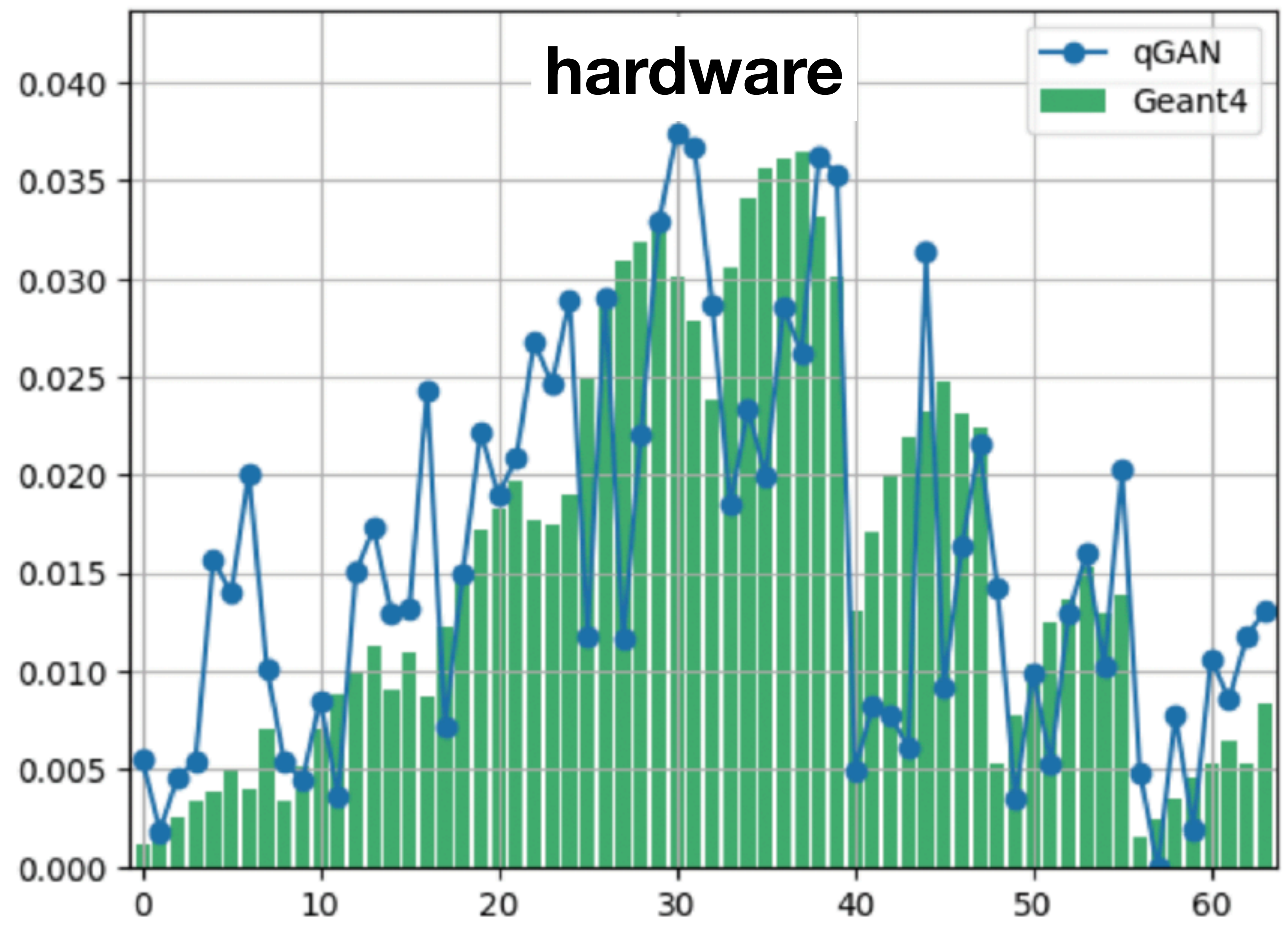
backup



0.05
0.04
0.03
0.02
0.01
0.00
relative entropy
0
20
40
60
80
100
epoch
no noise
double qubit gate error + readout error (2%)
double qubit gate error + readout error (4%)


readout error
double qubit gate error
double qubit gate error and readout error
0.14
0.12
0.10
0.08
0.06
0.04
0.02
relative entropy
0.0%
2.0%
4.0%
6.0%
8.0%
10.0%
noise level


ideal simulator
qGAN
Geant4
0.040
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0
10
20
30
40
50
60


hardware
qGAN
Geant4
0.040
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0
10
20
30
40
50
60