CaloShowerGAN

A GAN model for fast calorimeter shower simulate in HEP

Michele Faucci Giannelli (INFN - Roma2)

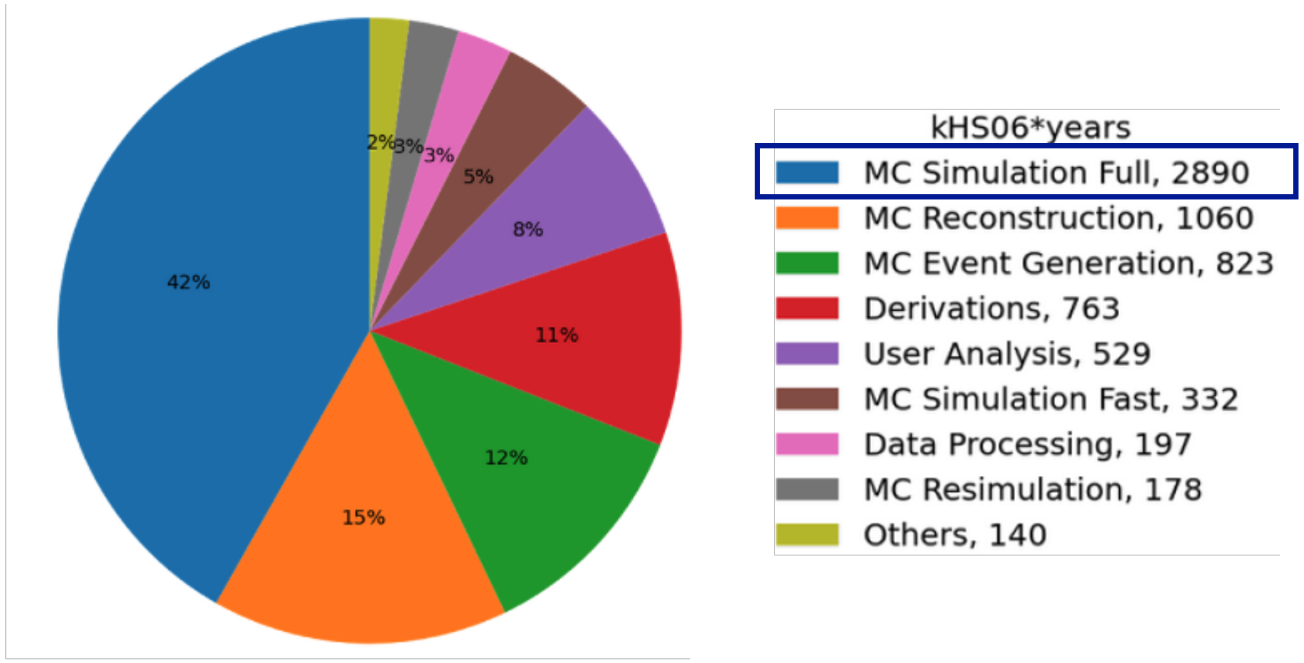
Rui Zhang (Wisconsin)

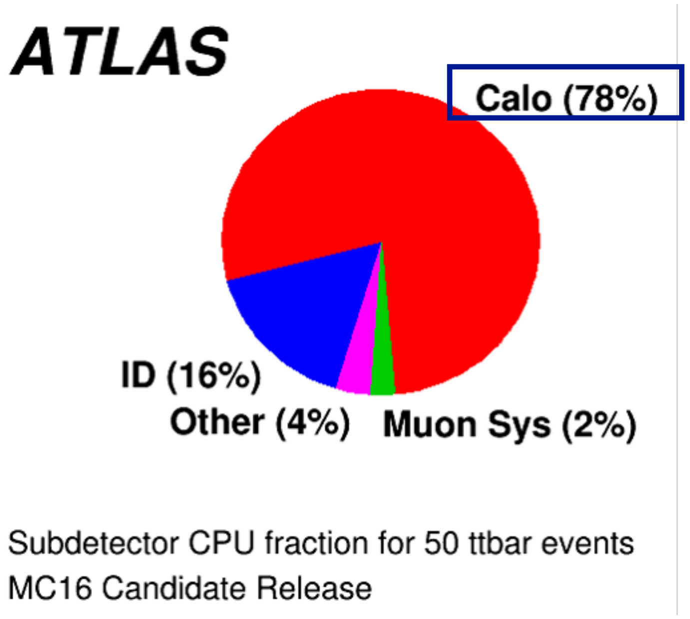
CaloChallenge Workshop 2023

30–31 May 2023, Frascati, Italy

Simulation in HEP

- Monte-Carlo simulation is crucial in understanding and analysing data
  - Simulate interactions happening in calorimeter is time & resource intensive





  - Reducing time in calorimeter simulation is the first task to speed up the MC production
- Fast Calorimeter simulation becomes an increasingly essential requirement
  - Generative models assisted calorimeter simulation would be much faster
  - In the meantime, required the generated showers to have high quality and to be as close as with Geant 4

Generative Adversarial Networks (GAN)

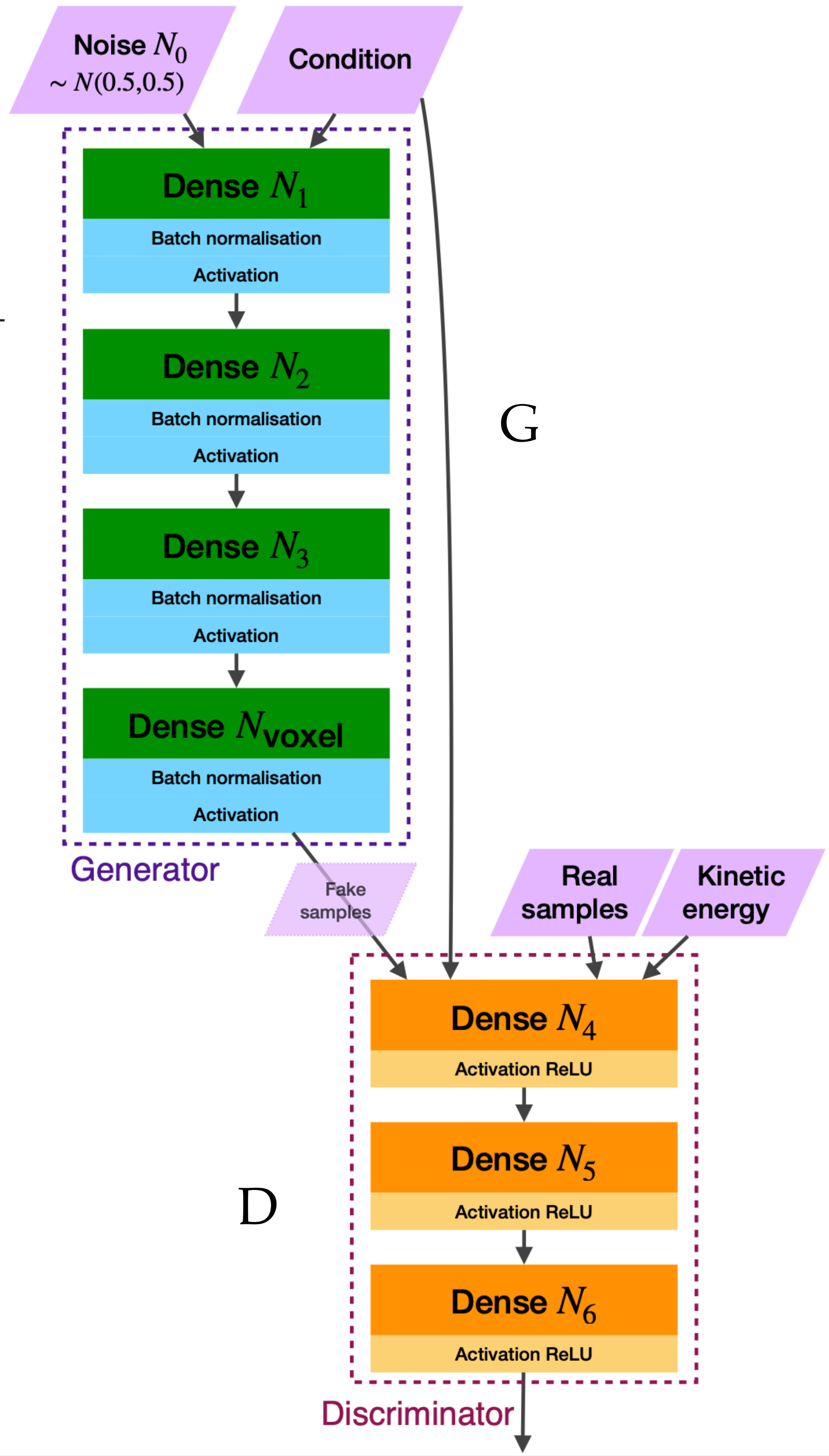
- GANs are an known approach for generative models using deep learning techniques
  - An unsupervised learning task, learn patterns in input dataset and generate new examples that could plausibly have been drawn from the input
- Training of GANs is framed as supervised learning
  - Two sub-models: the generator model is trained to generate new examples and the discriminator model is trained to classify examples as either real (from the input) or fake (generated).
  - The two models are trained together in a zero-sum game, until the discriminator model is fooled about half the time, meaning the generator model is generating plausible examples.
- Wasserstein GAN (WGAN) is a variant that introduces the Wasserstein distance (also known as Earth Mover's distance) as a new metric
  - Improved training stability; better gradient flow and vanishing gradients; Mode collapse mitigation; More meaningful loss and evaluation metrics

GAN solution for shower simulation

- CaloShowerGAN is proposed in this talk to use WGAN technique for calorimeter shower simulation
- It is conditioned by the incident kinetic energy of the particle
- Will use the Dataset1 from the CaloChallenge [link] to benchmark
- Table of Contents:
  - Dataset
  - Model architecture and training
  - Hyperparameter optimisation
  - Model performance
  - Investigation of energy split
  - Summary

CaloShowerGAN

- Generator: 3 hidden layer + 1 output layer
  - Each layer: a dense layer + batch normalisation + activation
  - Consume a noise vector of multi-dimensional Gaussian (mean=0.5, std=0.5)
  - Condition on particle kinetic energy to train/generate
- Discriminator: 3 hidden layer + 1 output layer
  - Each layer: a dense layer + activation (ReLU)
  - Batch normalisation does not help in performance
- Above are common for all particles



Data preprocessing

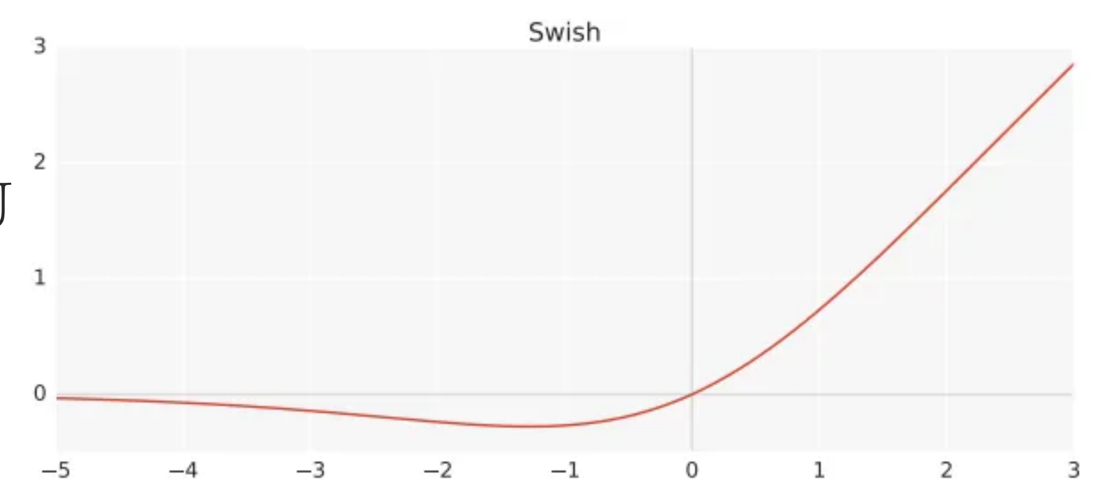
- Input values are energies, normalised by the true incident energy
  - After normalisation, all values in the input are in the same order of magnitude and dimensionless
- Condition label of kinetic energy is normalised to [0, 1] using
  - $\hat{E} = \dfrac{\log \frac{E_{\text{kin}}}{E_{\text{min}}}}{\log \frac{E_{\text{max}}}{E_{\text{min}}}}$
  - Motivated by the expectation that the shower width is logarithmically dependent on the kinetic energy of the incoming particle
- Removing too low values in training vector
  - It is found not improving or undermining the numerical stability, therefore is not used in the results
- Normalisation depending on incident energy
  - It is found not improving the performance, therefore is not used in the results

Training

- Training is done separately for particles (γ, π)
- Optimiser: Adam
- Train 1 million iterations and checkpoint models every 1k
- Evaluated by calculating χ^2 of total energy distributions in all 15 energies between generated and Geant 4 simulated showers
  - A good metric which is non-trivial to produce by output vectors
  - The checkpoint that gives the best is χ^2 will be used

Hyperparameter optimisation

- Optimiser
  - Learning rate and momentum of both generator and discriminator are tested
  - Batch size
  - Cyclic learning rate and variant such as RAdam, LookAhead, AdamW do not help
- Activation
  - Con of Swish: seems to be less stable than ReLU
  - Swish in γ G is helpful while ReLU is good in other situations



  - More powerful when paring with Glorot Normal initialisation of neuron weights
    - ReLU is best to use with He Uniform
- GAN parameters
  - D/G ratio: number of training passes of D for each pass of G (always > 1 since < 1 do not offer advantage)
  - λ that controls the penalty contribution

Final model hyperparameters

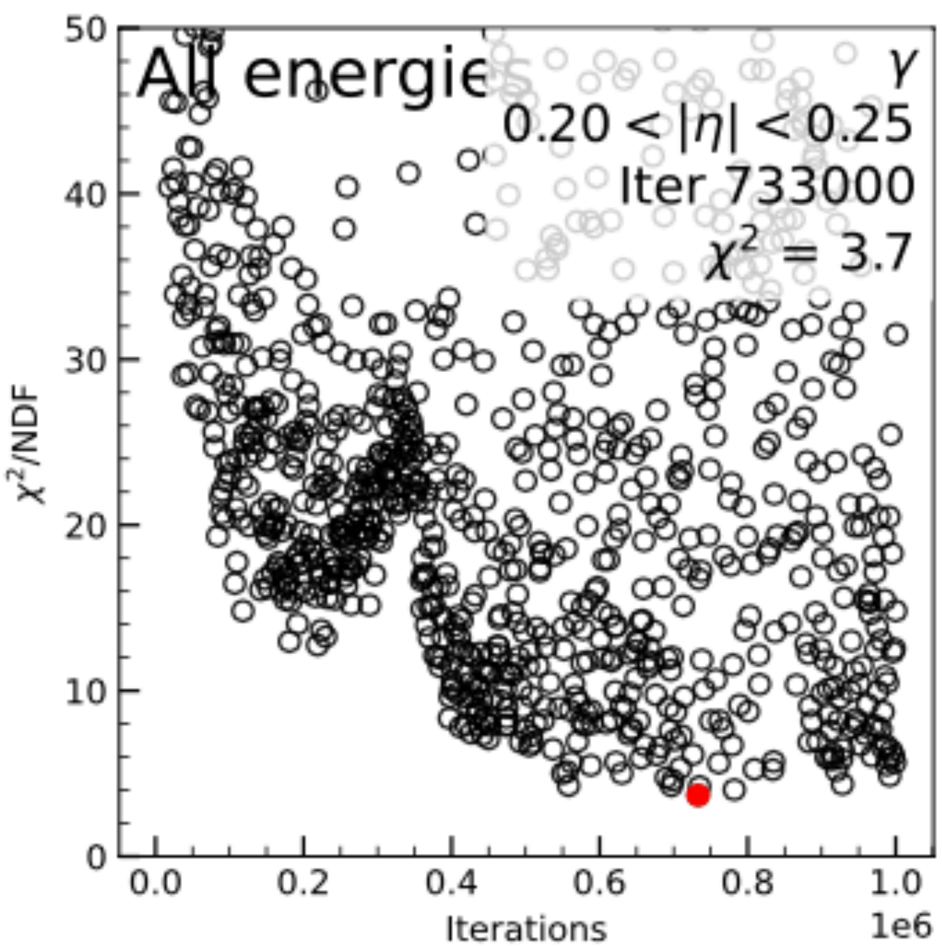
Hyperparameter	Photon	Pion
Generator size (N_0, N_1, N_2, N_3)	100, 100, 200, 400	200, 200, 400, 800
Discriminator size (N_4, N_5, N_6)	368, 368, 368	800, 400, 200
Generator optimiser	Adam	Adam
Learning rate	1×10^{-4}	1×10^{-4}
β_1	0.5	0.5
β_2	0.999	0.999
Discriminator optimiser	Adam	Adam
Learning rate	1×10^{-4}	1×10^{-4}
β_1	0.9	0.5
β_2	0.999	0.999
Batch size	1024	1024
D/G ratio	8	5
λ	3	20
Activation	Swish	ReLU
Neuron weight initialisation (generator)	Glorot Normal	He Uniform
Neuron weight initialisation (discriminator)	He Uniform	He Uniform
Trainable parameters (generator, discriminator)	261k, 408k	871k, 829k

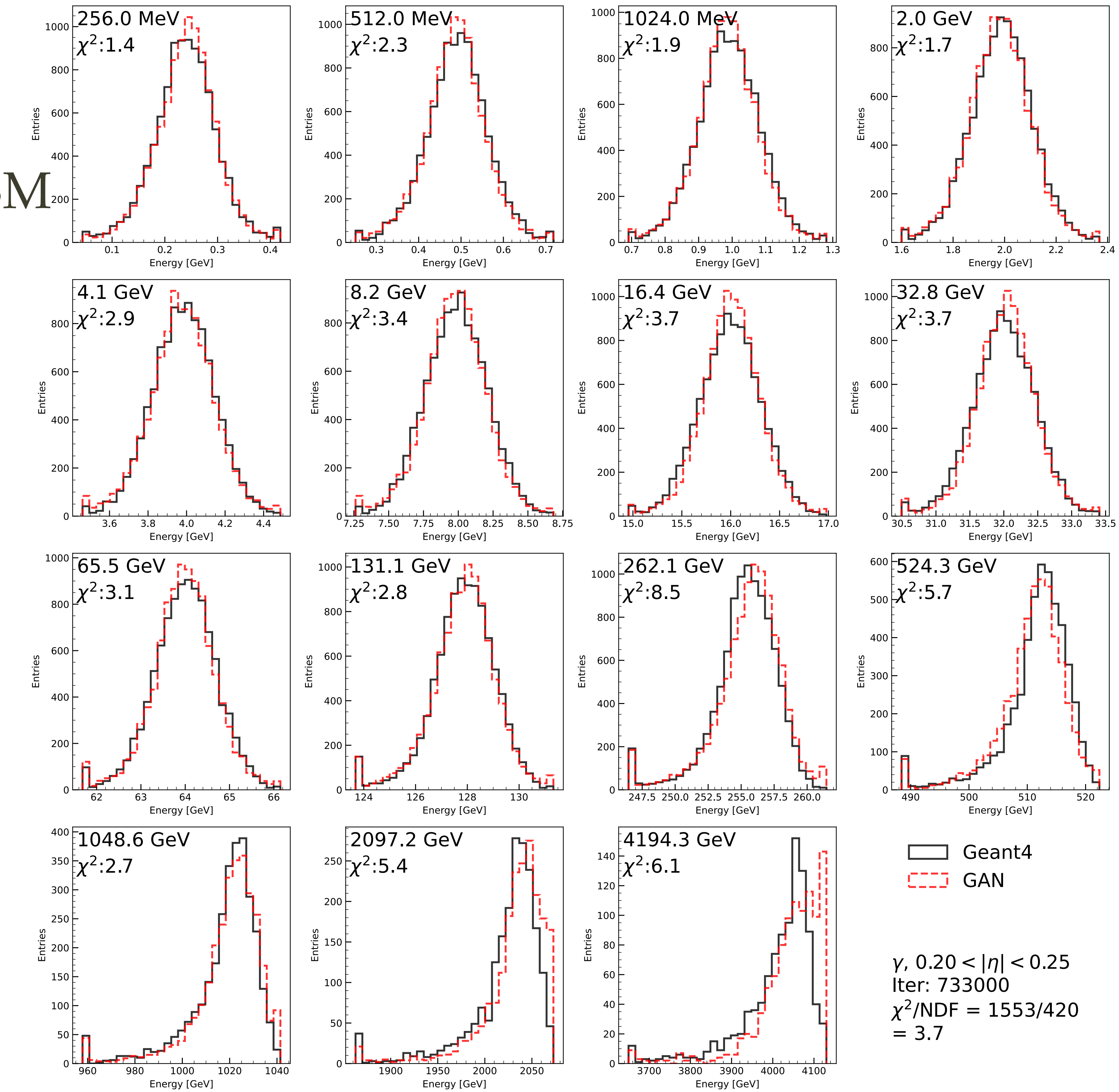
- Pion GAN is larger due to more number of voxels

Photon CaloShowerGAN result

Total energy γ

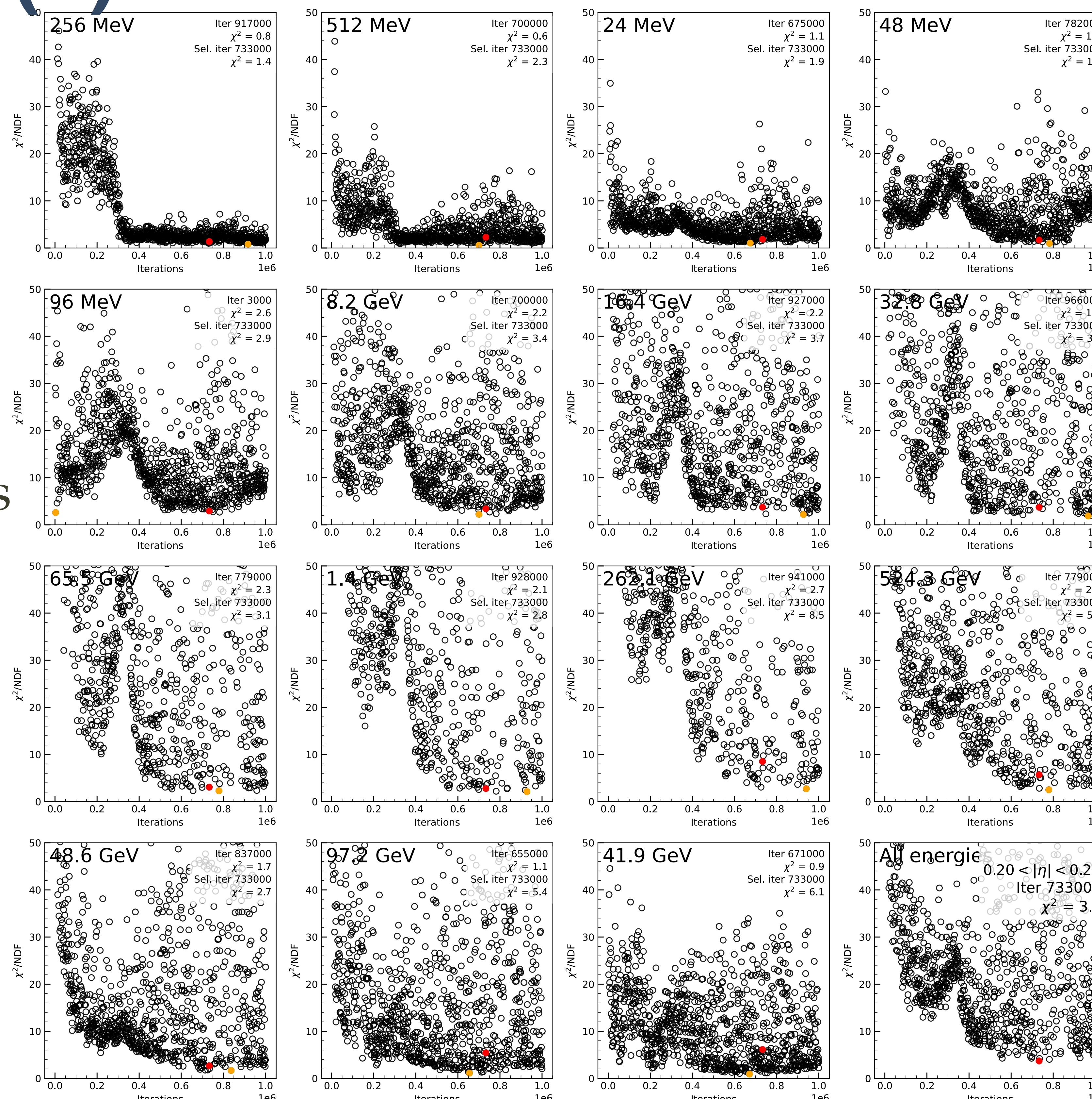
- $\chi^2 = 3.7$
- A structure in χ^2 at ~ 0.3M iteration





Total energy γ (2)

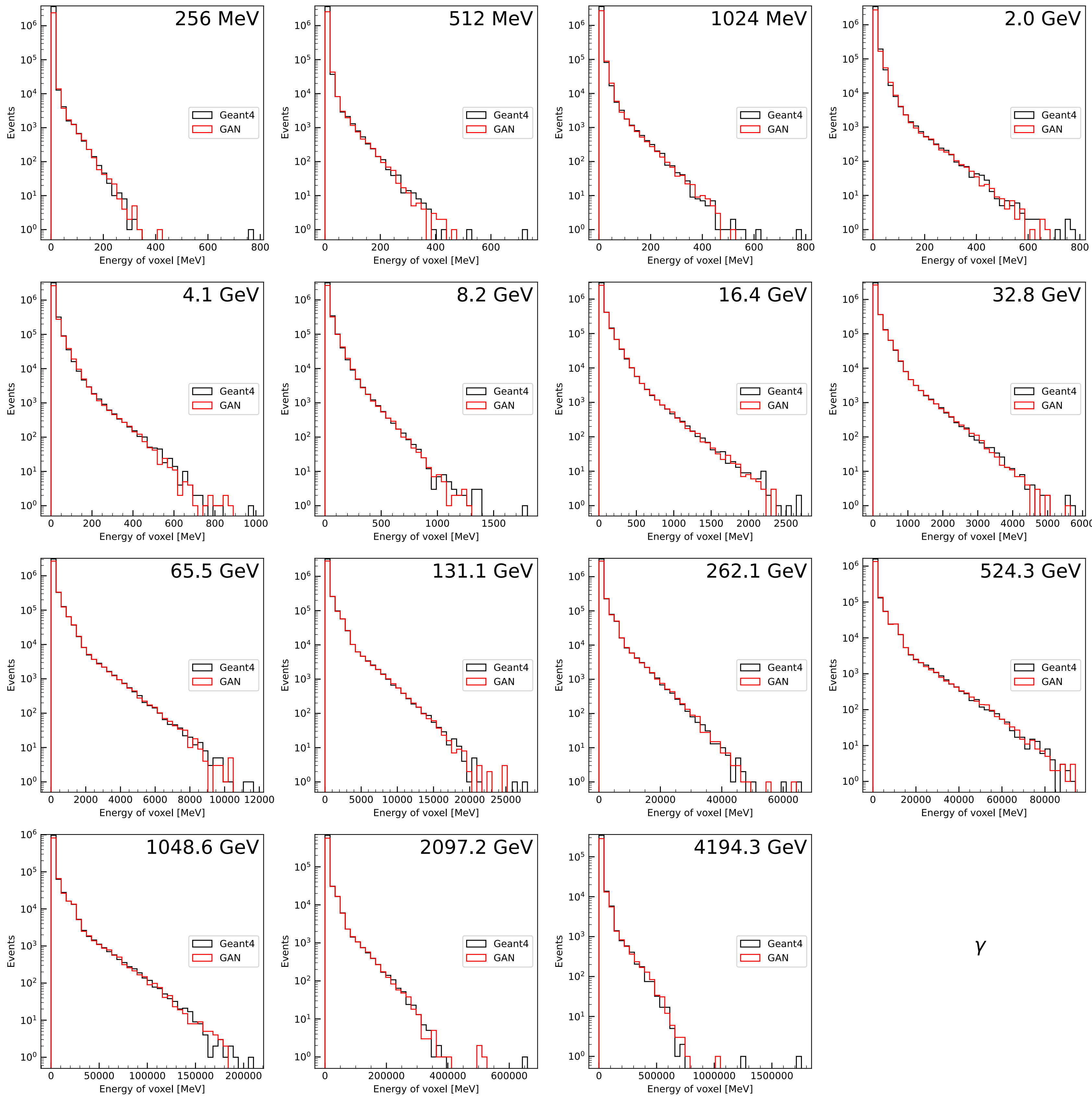
- $\chi^2 = 3.7$
- The global best model performs worse in some energies
- 0.3M iter structure persist in all low energies
- Some energies do not perform well, eg 96 MeV



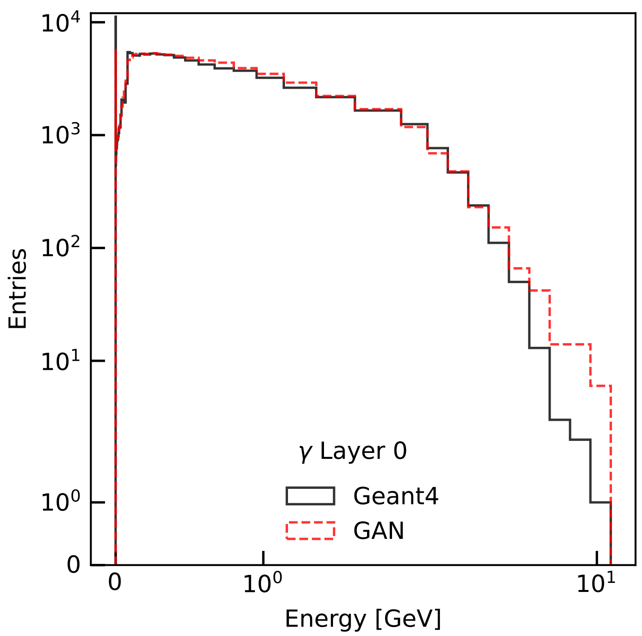
All energies

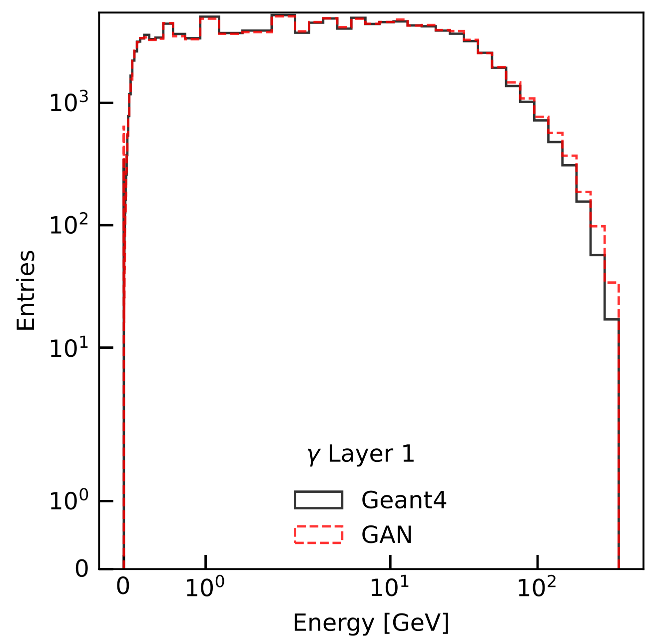
Voxel energy γ

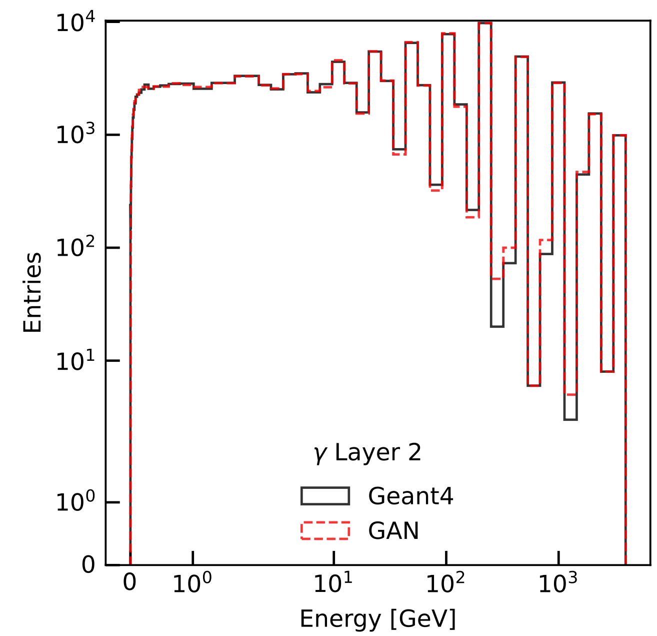
- CaloShowerGAN can reproduce voxel energy distributions

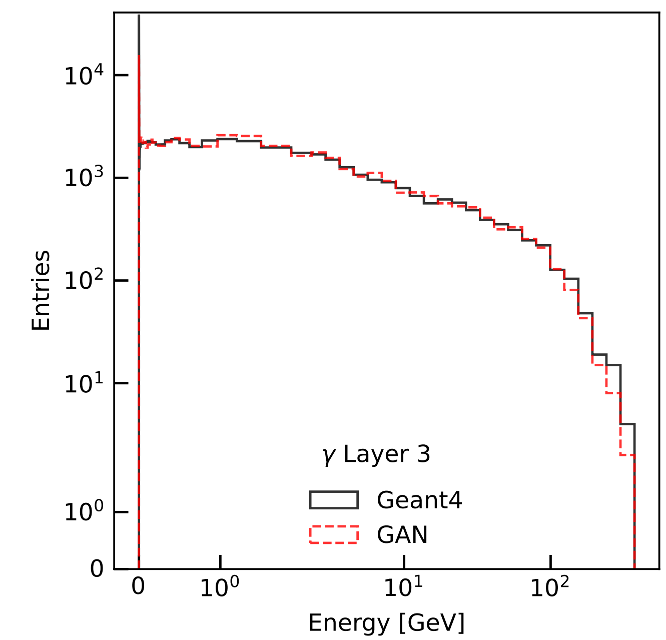


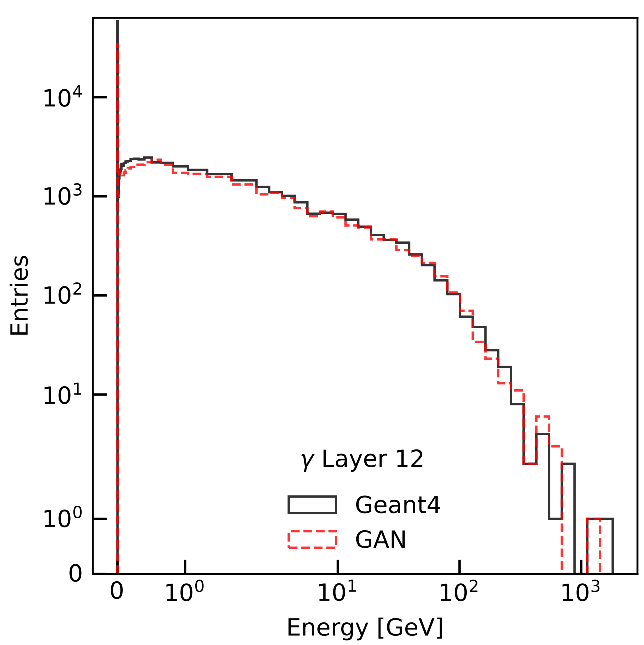
Layer energy γ







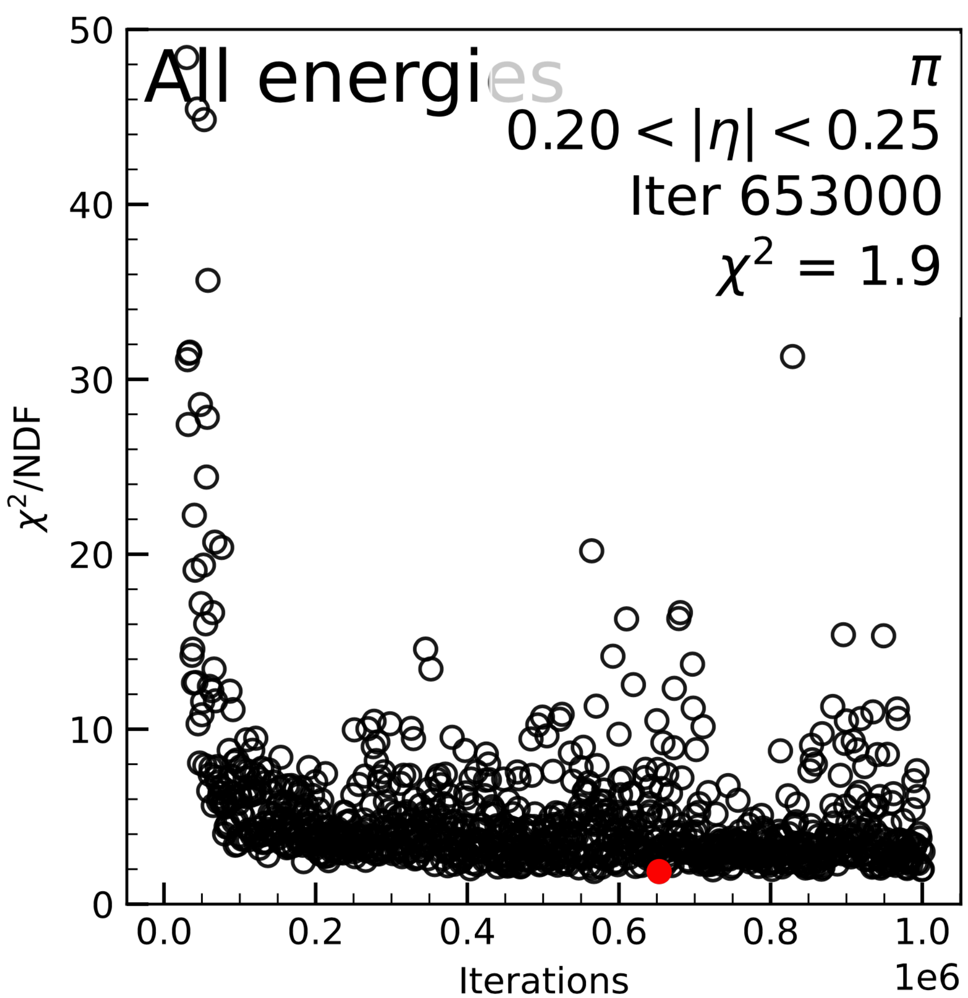


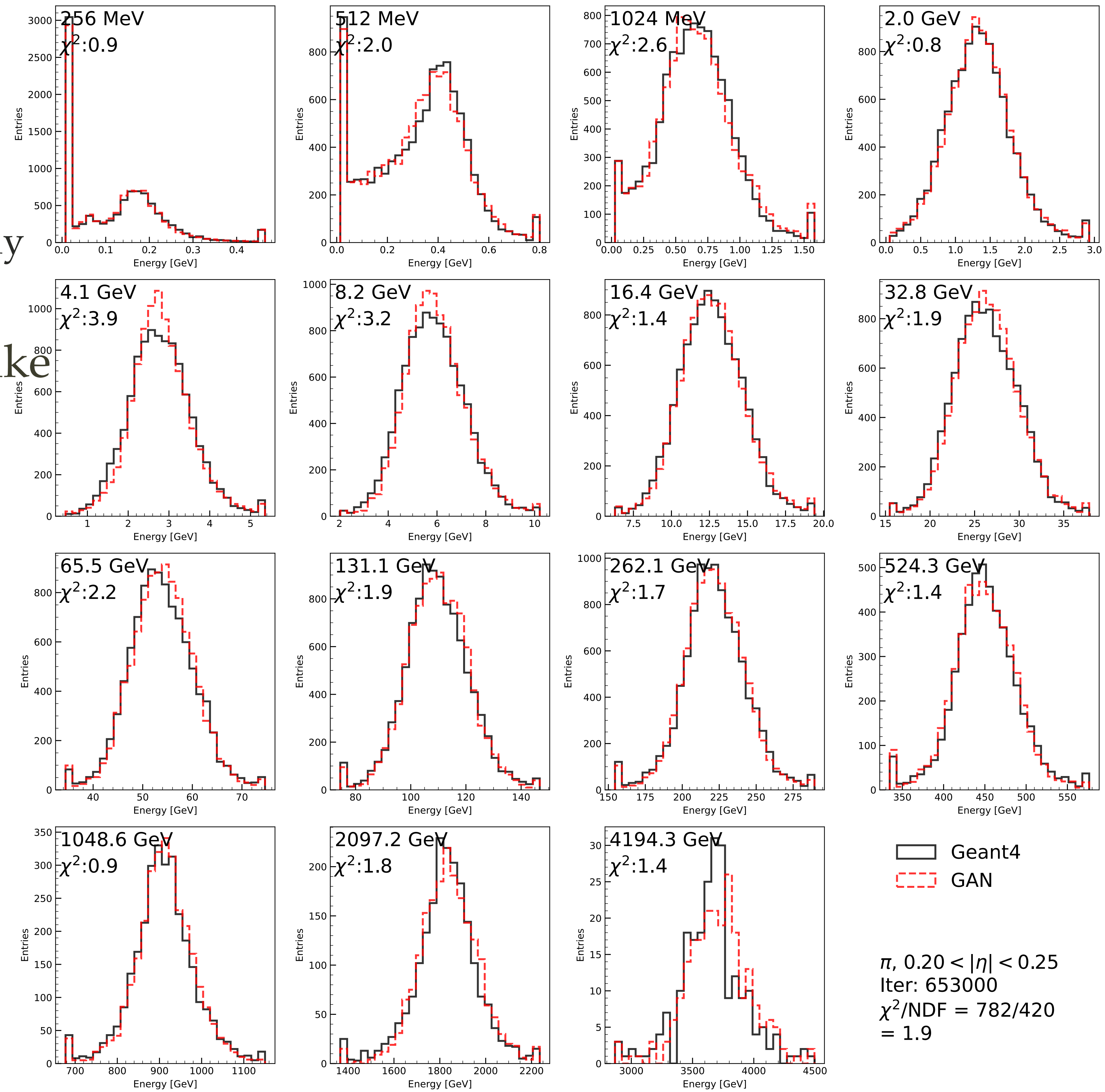


Pion CaloShowerGAN result

Total energy π

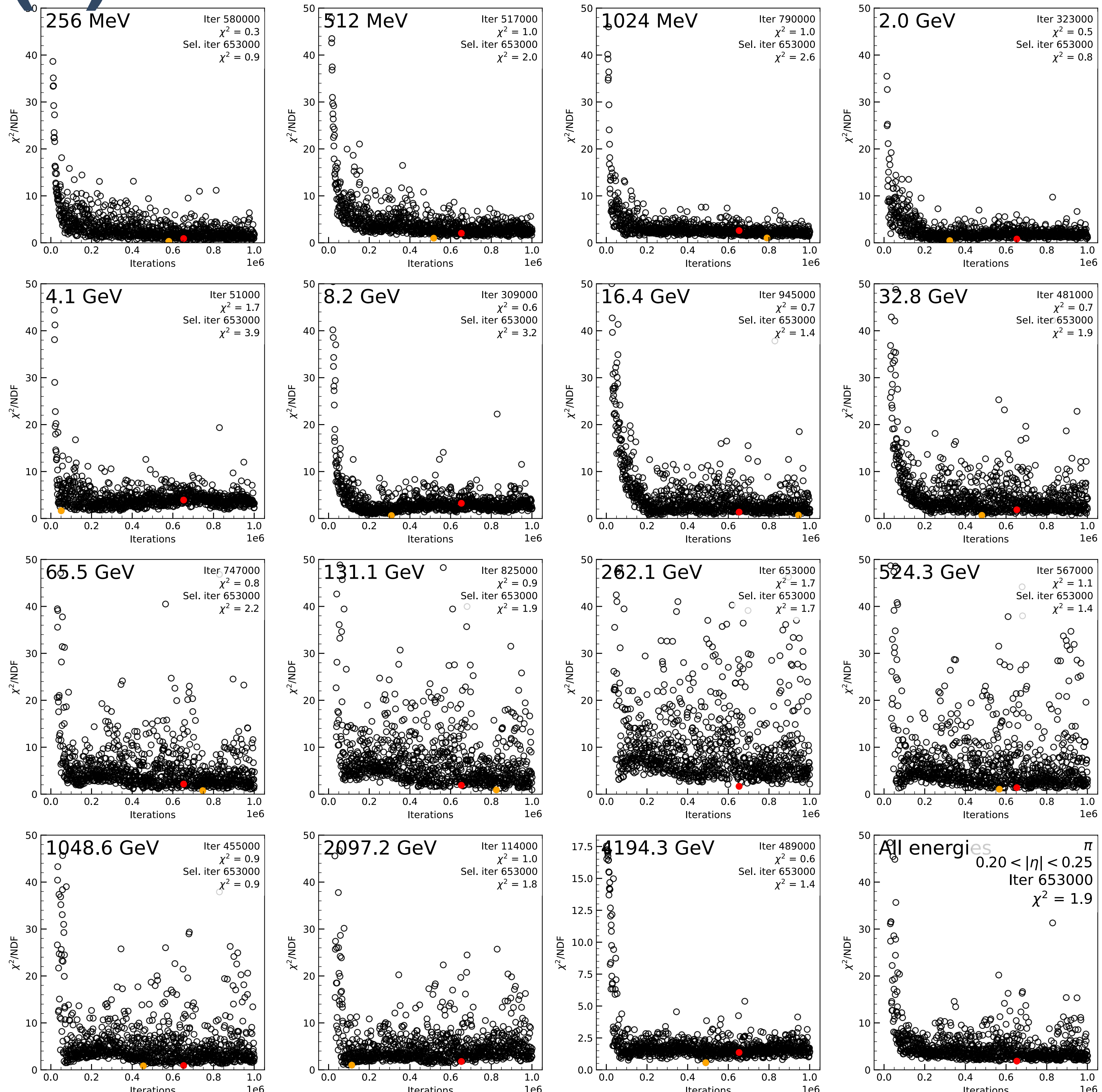
- $\chi^2 = 1.9$
- Training is fast and stable
  - Plateau around 0.2M and slowly improve until 0.6M
- Can produce low energy spike





Total energy π (2)

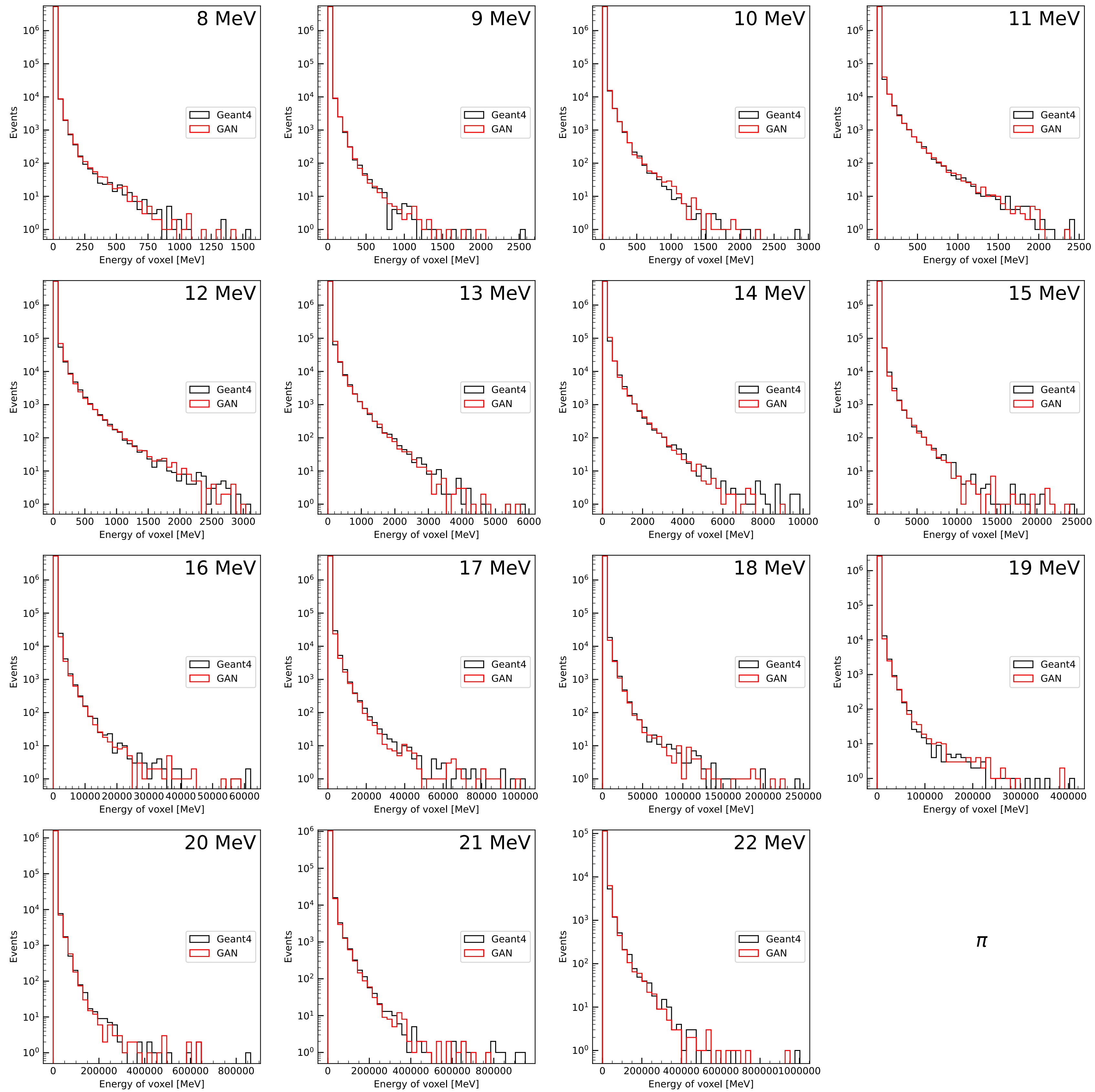
- $\chi^2 = 1.9$
- The global best model performs worse in some energies
- Some energies do not perform well, eg 4.1GeV and 2.1 TeV



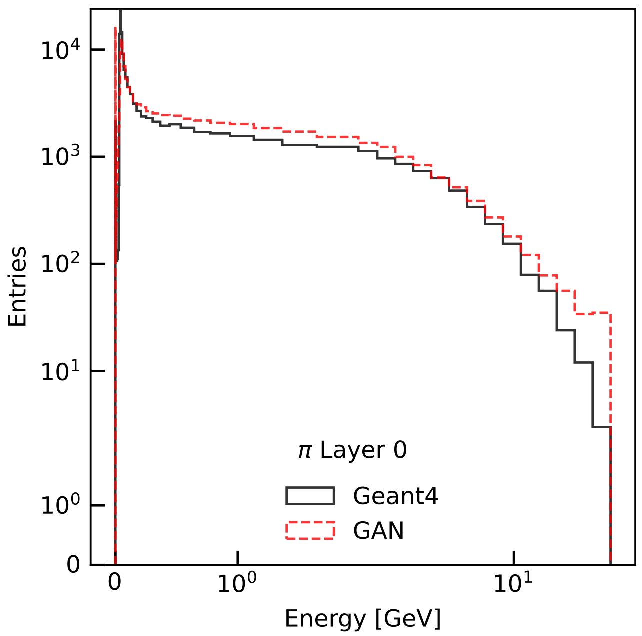
All energies

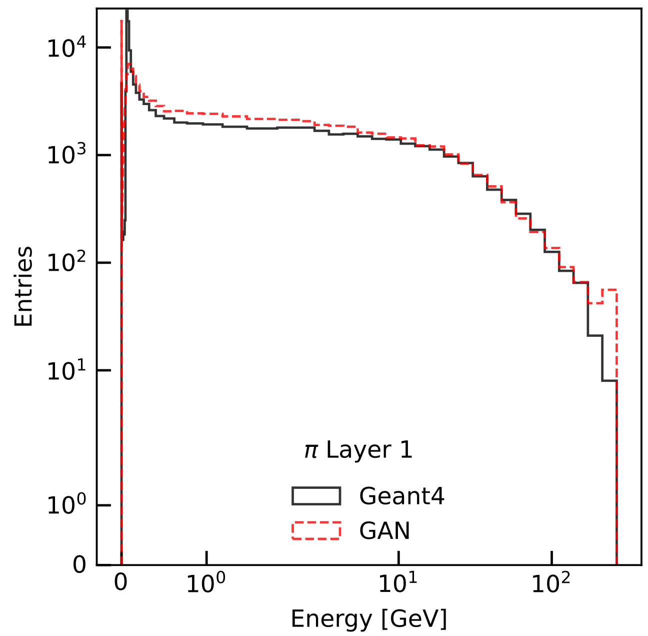
Voxel energy π

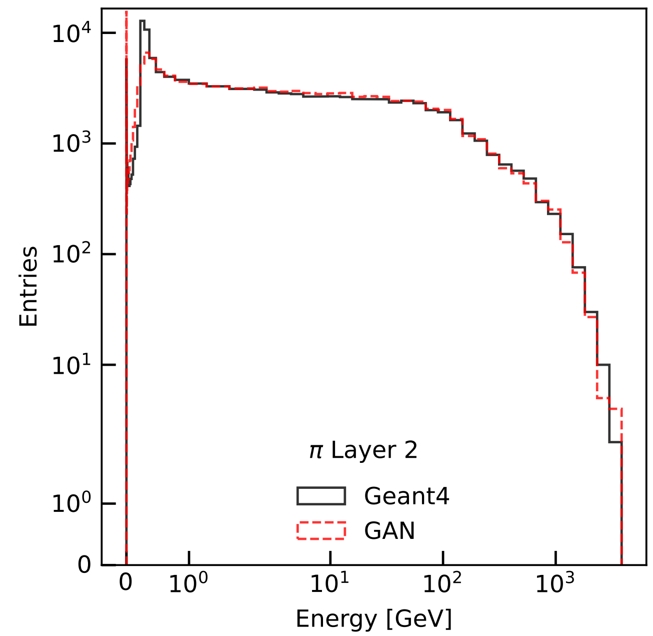
- CaloShowerGAN can reproduce voxel energy distributions

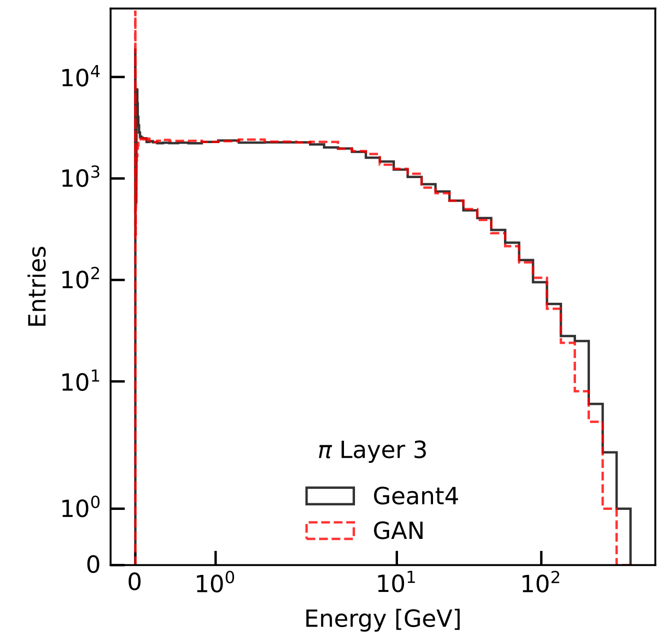


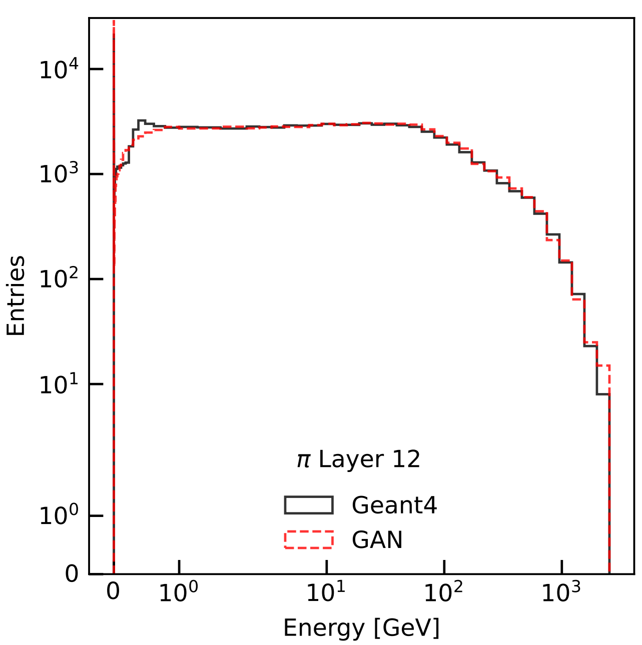
Layer energy π

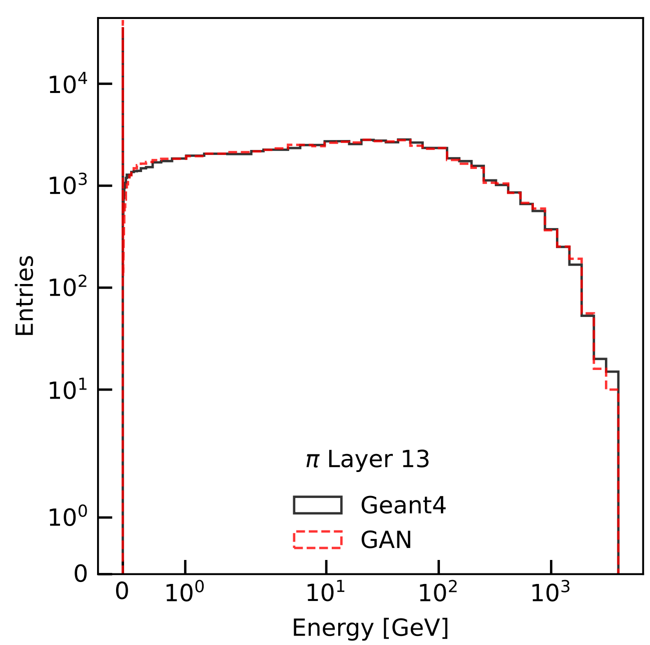


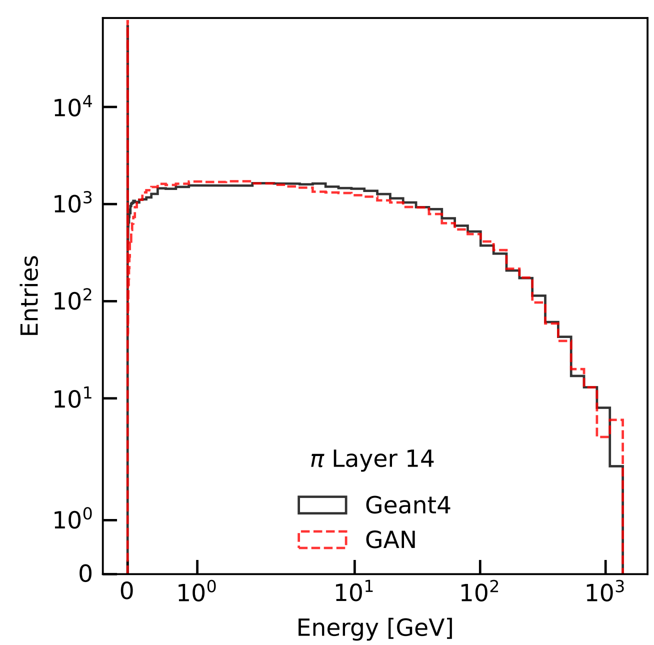








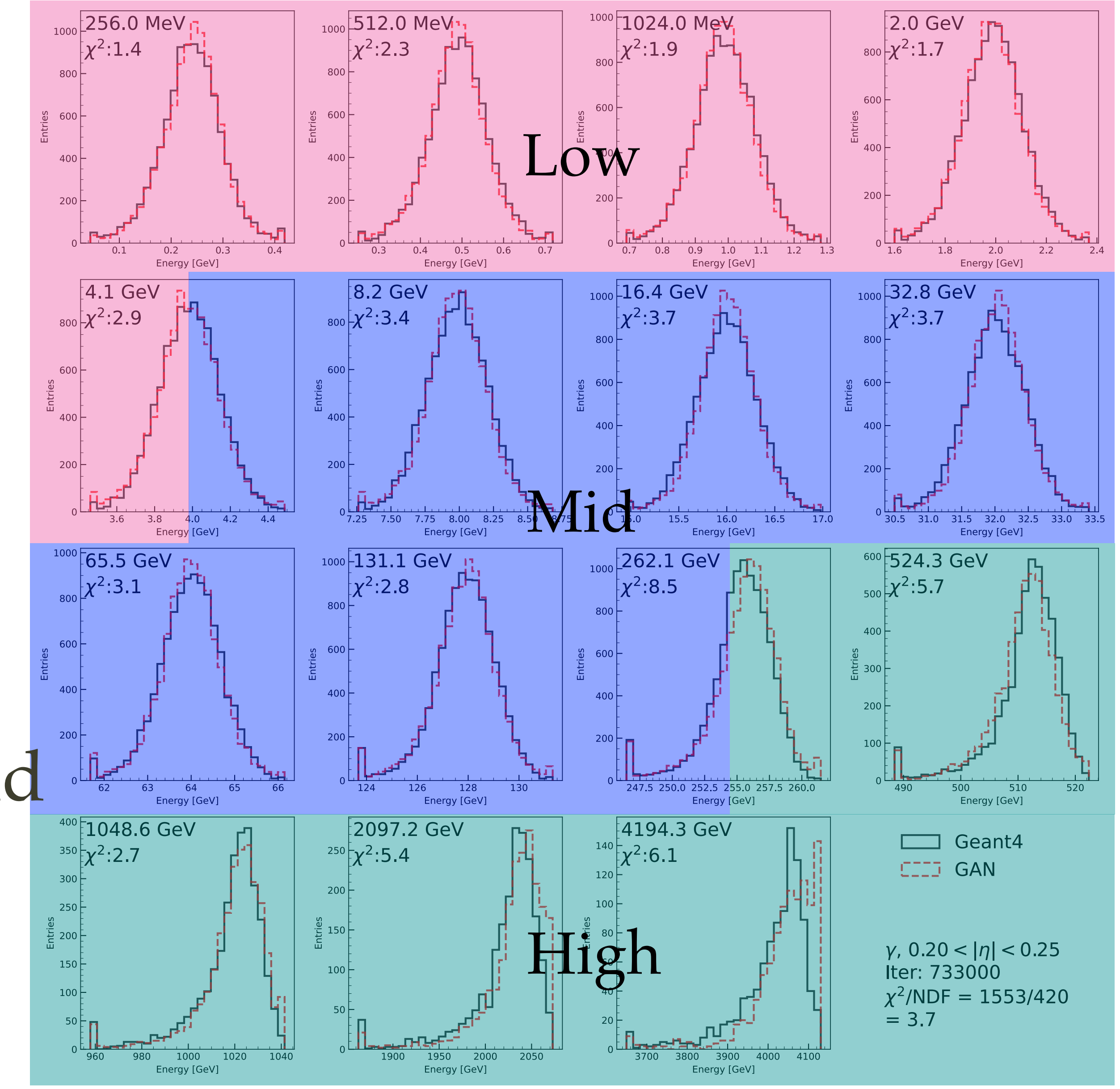




Energy split — Photon

Energy split in photons

- Low energy and high energy photon responses are quite different
  - May make sense to use different GAN models for low and high energies
- Two scenarios are tested
  - Split at 4.1GeV — 2 GANs
  - Split at 4.1GeV and 262GeV — 3 GANs
- Use the same hyperparameter for Mid and High as the previous photon
- Use ReLU for low instead of Swish



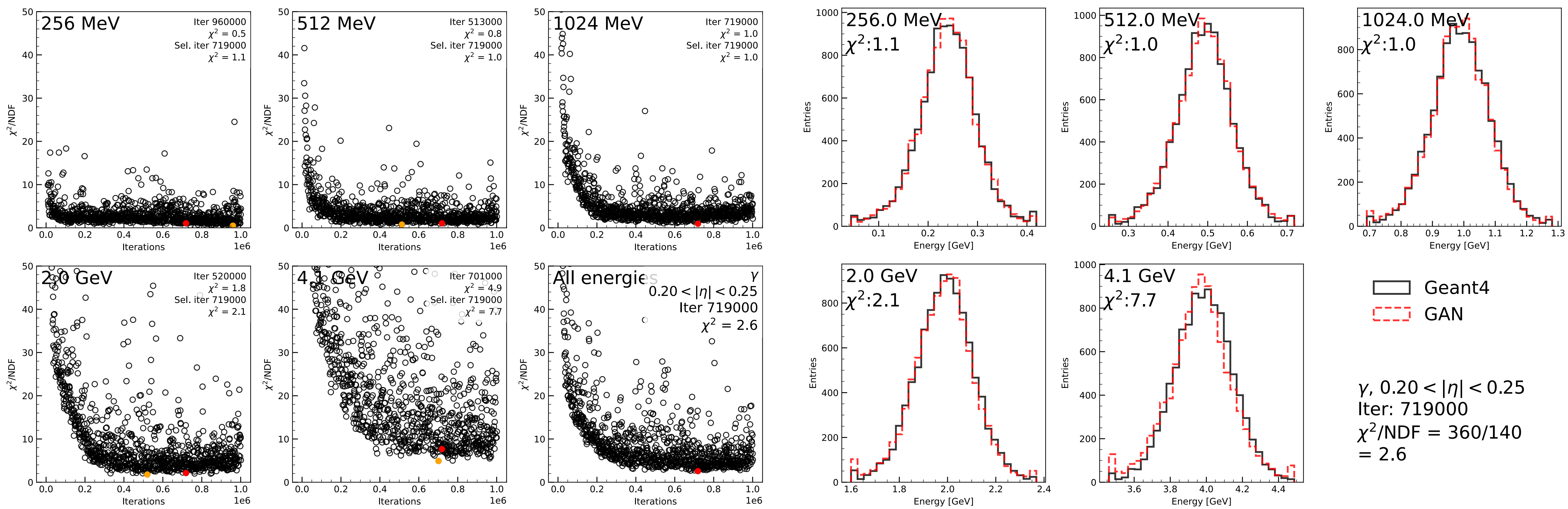
	1 GAN	Low	Mid	High	Mid+Hig
G size	100, 100, 200,	50, 50, 100,	50, 50, 100, 200		
Activation	Swish	ReLU	Swish		

Split energy result γ

- χ^2: 3.7 → 3.1 → 2.5
- If use 3x larger node in G in 1 GAN, $\chi^2 = 8.2$

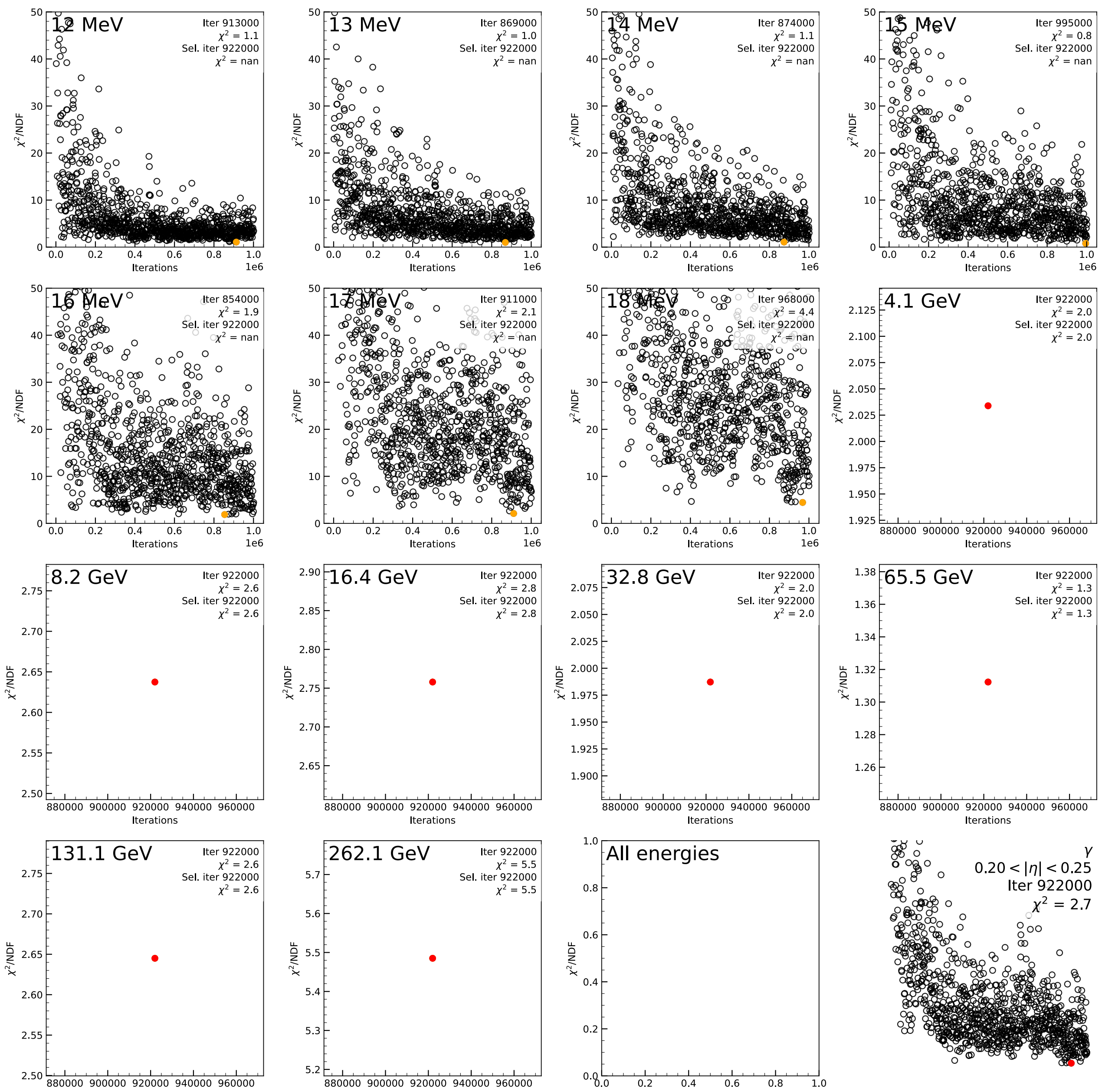
Energy range	Total χ^2	NDF	χ^2/NDF
$\leq 4.1\,\text{GeV}$	360	140	2.6
$\geq 4.1\,\text{GeV}$	1042	308	3.4
4.1 GeV–262.1 GeV	528	196	2.7
$\geq 262.1\,\text{GeV}$	299	140	2.1
$\leq 4.1\,\text{GeV} + \geq 4.1\,\text{GeV}$	1402	448	3.1
$\leq 4.1\,\text{GeV}$ + 4.1 GeV–262.1 GeV + $\geq 262.1\,\text{GeV}$	1187	476	2.5

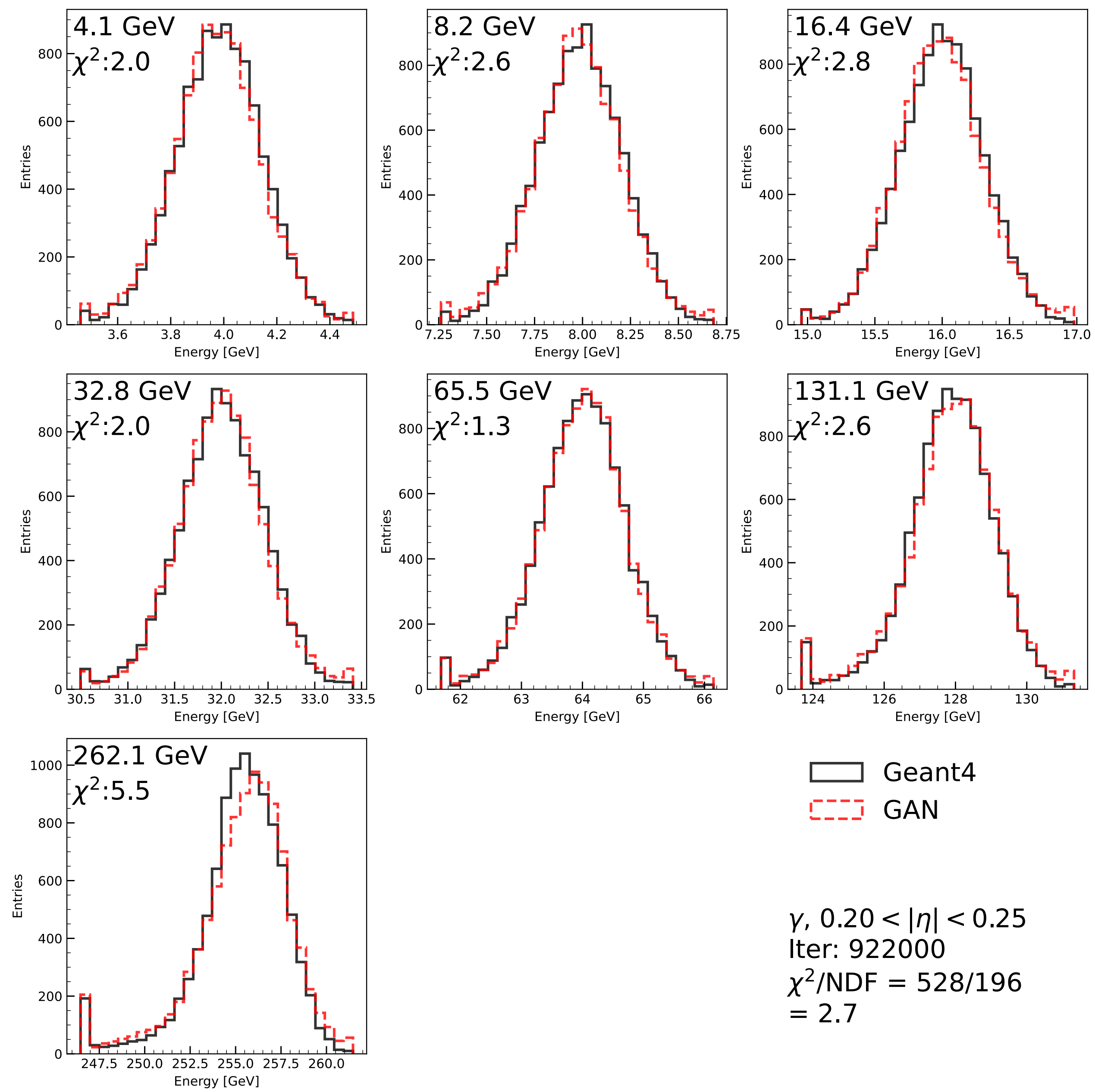
Low energy γ



- Very good low energy model

Mid energy γ

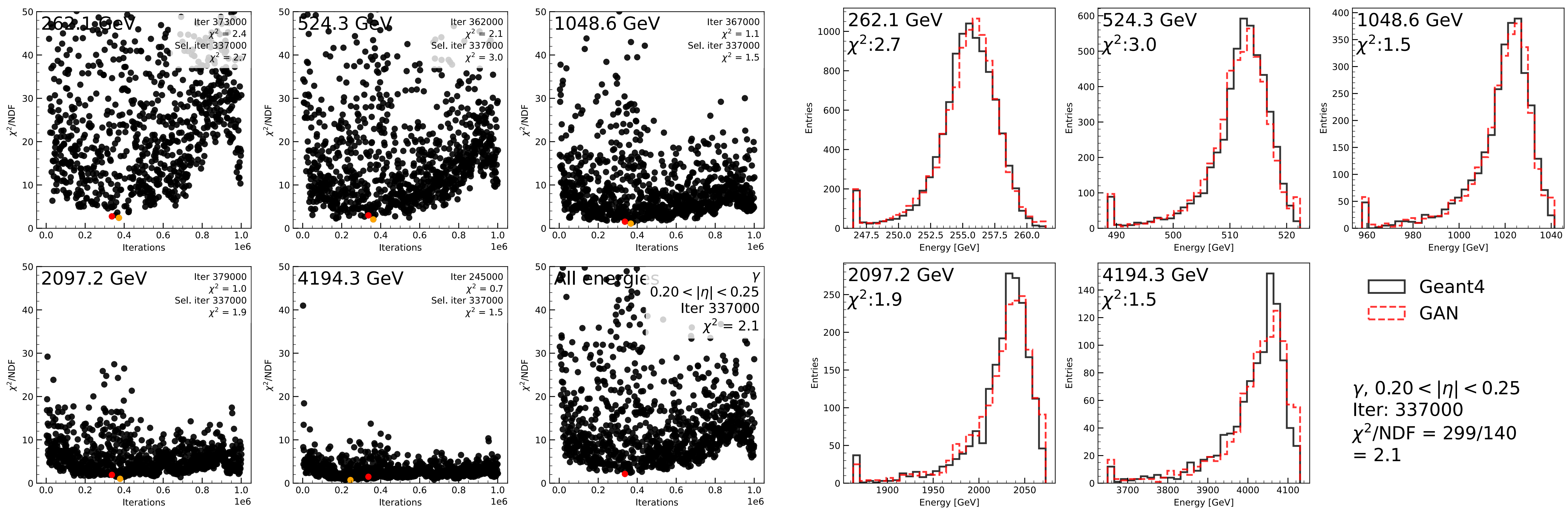




Some plotting issue

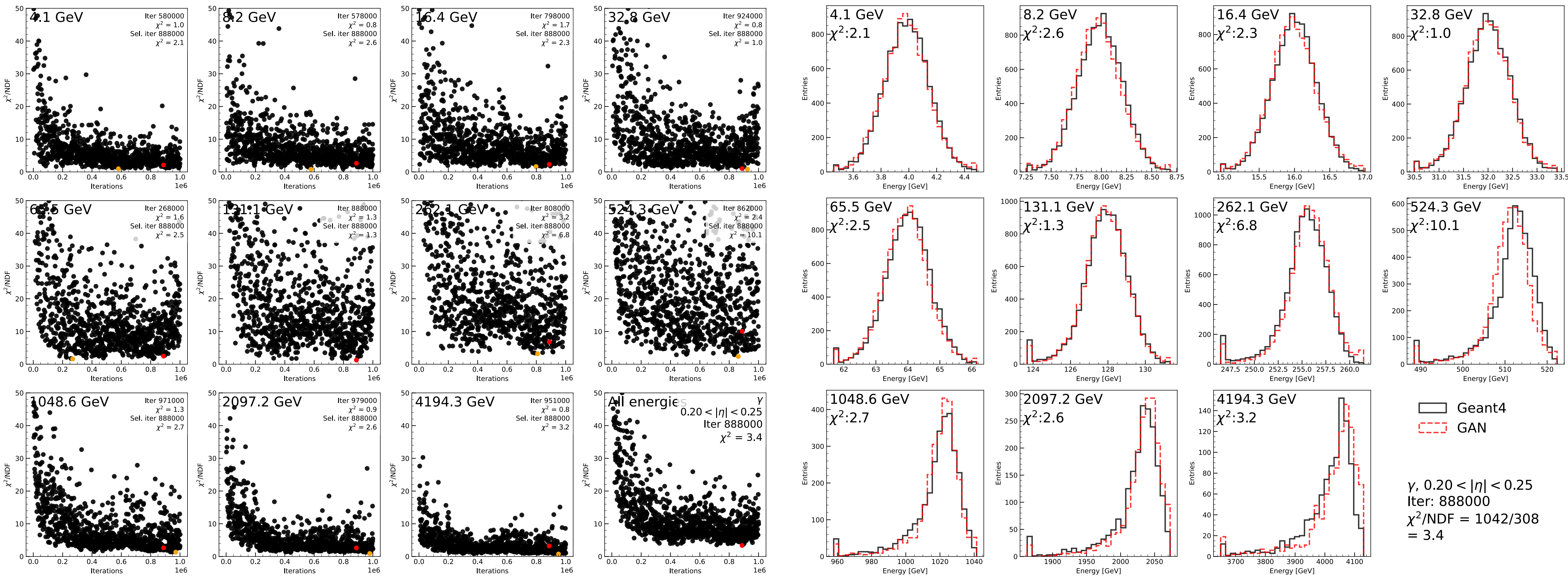
- When energy goes high, training becomes unstable

High energy γ



- High energy is most difficult to train

M+H energy γ



- Good mid energy training, but χ^2 is still large

Time consumption

	Training time V100 GPU [s/1k iter]	Inference time Intel(R) Core(TM) i9-9900K CPU @ 3.60GHz [s/ training events]
γ 1 GAN	38	0.54/121000
π 1 GAN	40	1.2/120230
γ 2 GANs	23+22	0.54
γ 3 GANs	23+23+23	

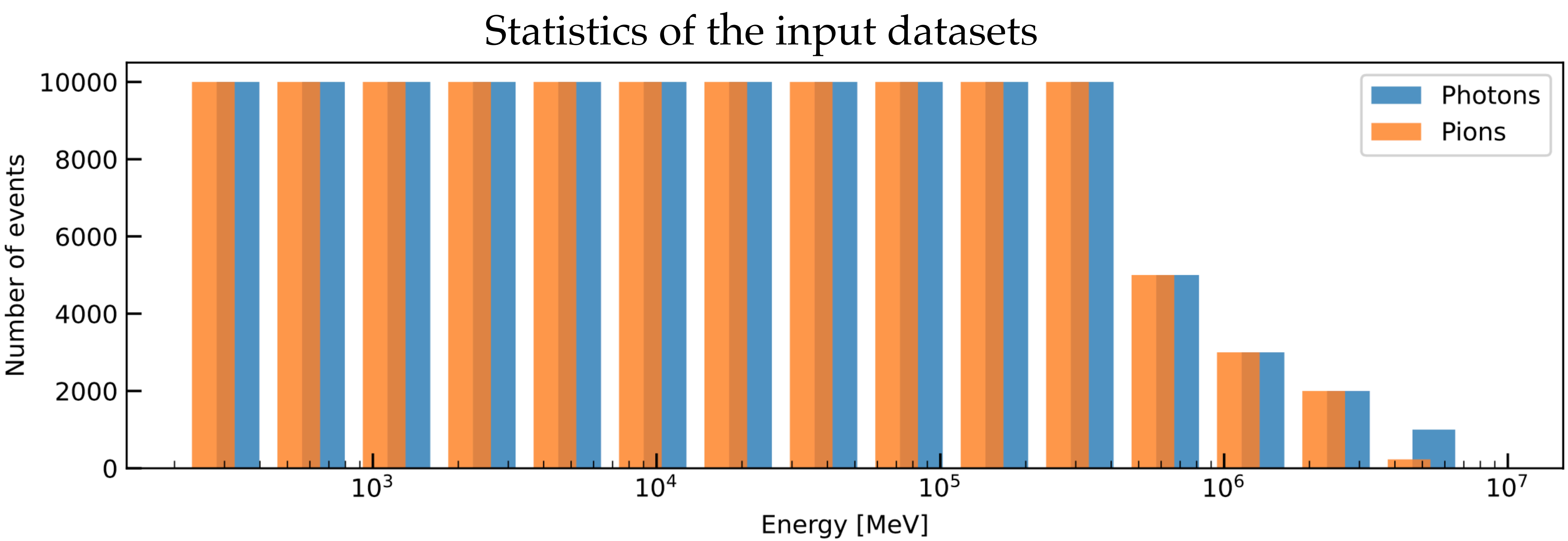
Summary

- CaloShowerGAN is proposed in this talk for fast calorimeter simulation
- Used Dataset1 from CaloChallenge to benchmark the performance
- Great π performance: $\chi^2 = 1.9$
- Good γ performance: $\chi^2 = 3.2$
  - Responses in energies are quite different
  - Train separate GANs for different energy range can further improve the performance $\chi^2 = 3.7 \rightarrow 3.1 \rightarrow 2.5$
  - Train a GAN with 3 times the parameters does not achieve the same results!

Backup

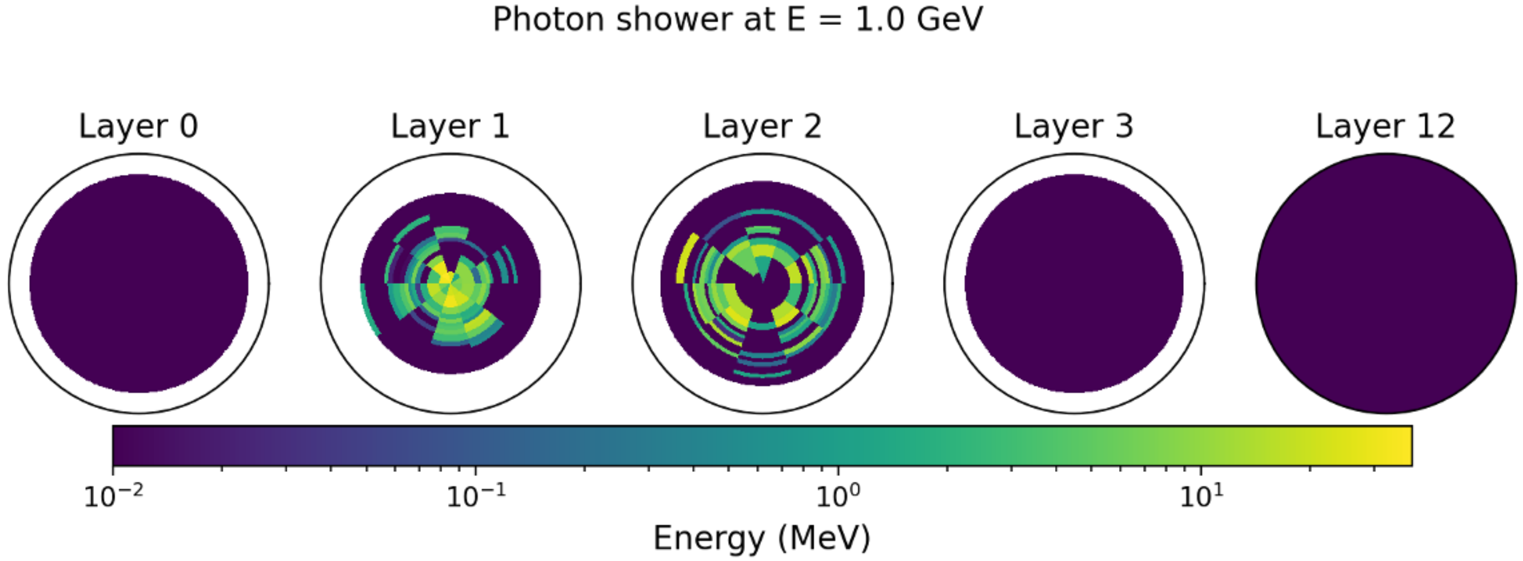
Input datasets

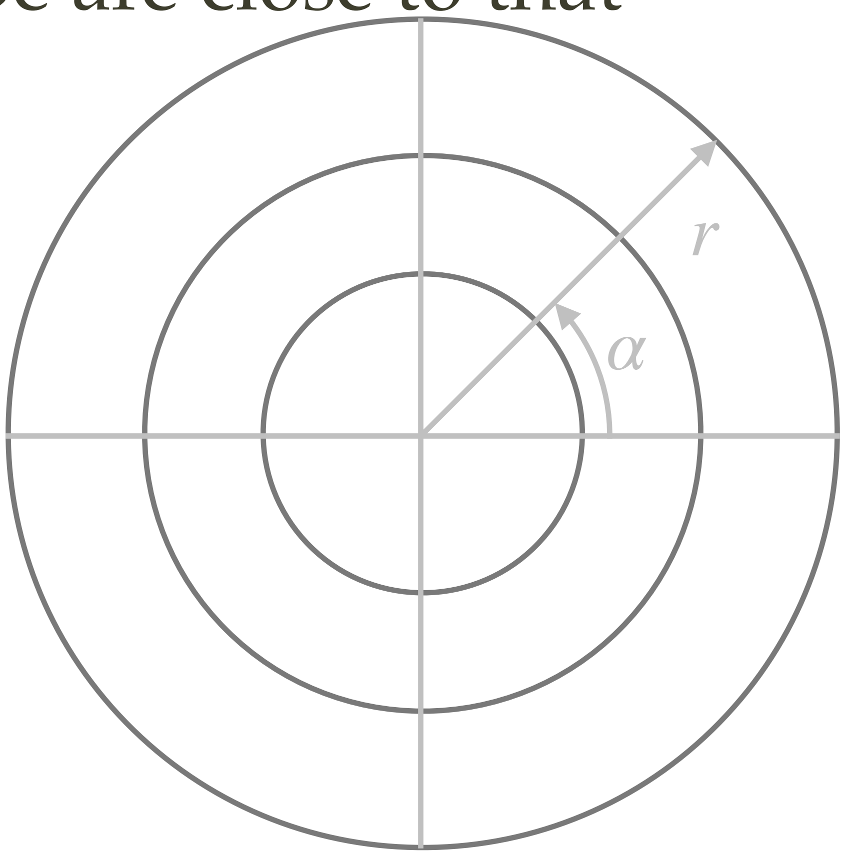
- We used DataSet 1 from CaloChallenge [link]
  - Known as the ATLAS voxelised data used in AtlFast3
  - Separate datasets for single particle photons and pions due to their different calorimeter responses
  - Consists of discrete kinetic energies from 256 MeV to 4 TeV
  - Will condition the GANs on kinetic energies to generate showers with different kinetic energies



Input datasets (2)

- Data structure of the machine learning task is the deposit energies in a group of calorimeter cells called "Voxels"
- Since responses dependent on the incidental pseudo-rapidity (η), the provided dataset is in a slice of $2.0 < \eta < 2.05$
- For a calorimeter layer, shower shape is transformed into r—α plane
- The task of CaloShowerGAN is to generate numbers in each voxel, such that the shower energy and shower shape are close to that simulate by Giant 4





Photons	iter x1000	chi2/ndf	iter x1000	chi2/ndf
swish + he	524000	12.052		
Goroth	601000	11.15	254000	6.195
Goroth_DoubleSize	726000	5.413	758000	3.952
ReLu_DoubleSize	899000	5.566	794000	3.311
Goroth_DoubleSize_Mask1KeV	972000	6.481	808000	3.909
Goroth_DoubleSize_Mask10KeV	867000	6.145	905000	3.587
Goroth_DoubleSize_Mask100eV	912000	5.712	610000	3.609