Weekly Update Nov. 21, 2024

https://github.com/QaloSim/CaloQVAE/tree/haojia

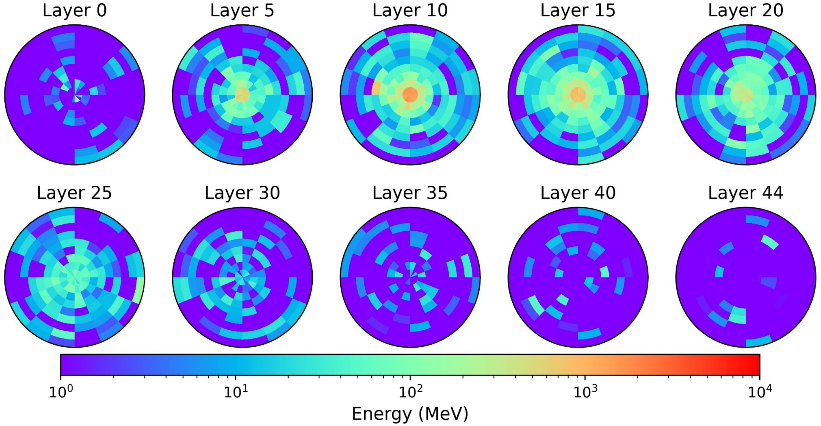
Prograss

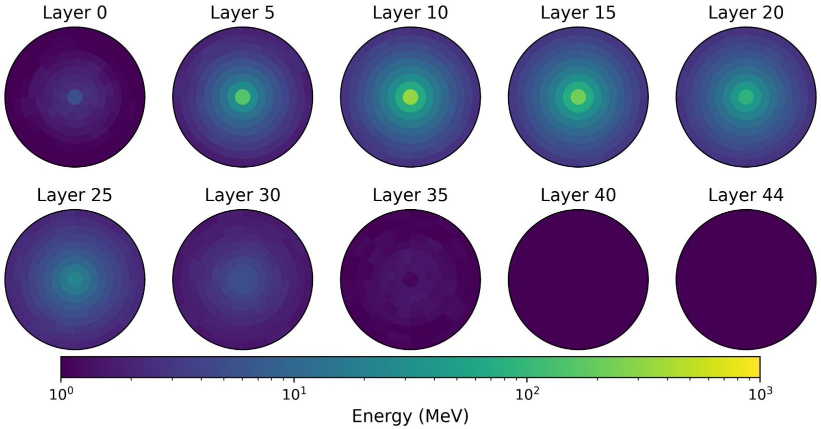
Run:elated-silence-540

- Self Linear Attention Layers
  - Extra Attention layer between incident energy and decoder
  - New scaling: log(1+x)
- Loss
  - AE_loss_per_event = MSE * [exp(a*(input-<input>))+exp(-b*(input-<input>))]
  - Sum_AE_loss = scaled_E_inc ⊗ AE_loss_per_event
- Constrain the NN output by scaled Sigmoid function (0-9) (Reason in later pages)

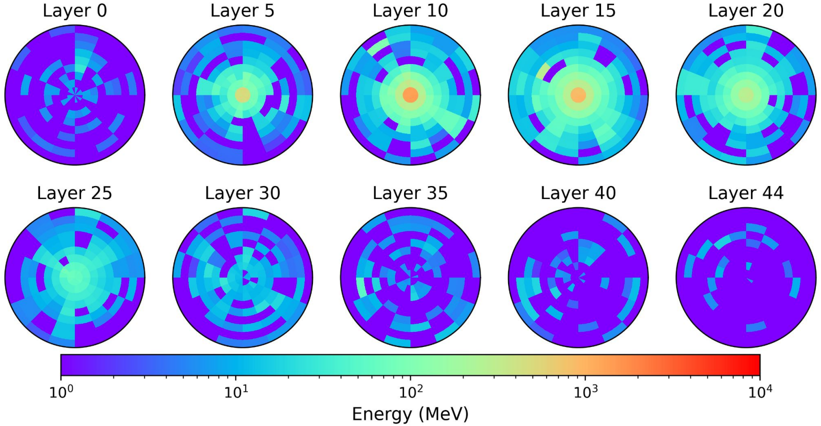
Current: Pegasus, 302 x 4, old KL training, no freezing, potentially better samples next week

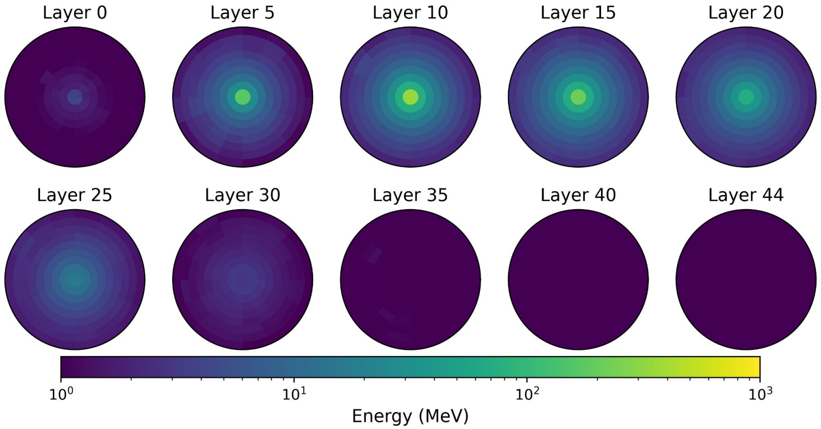
GEANT



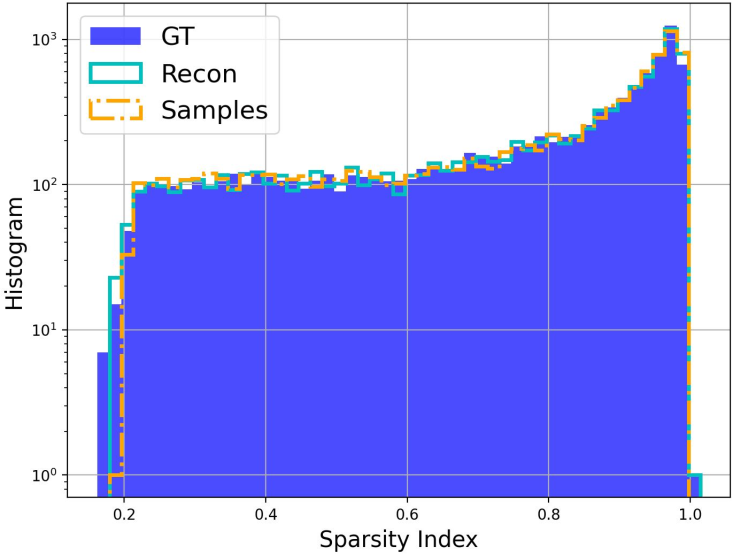


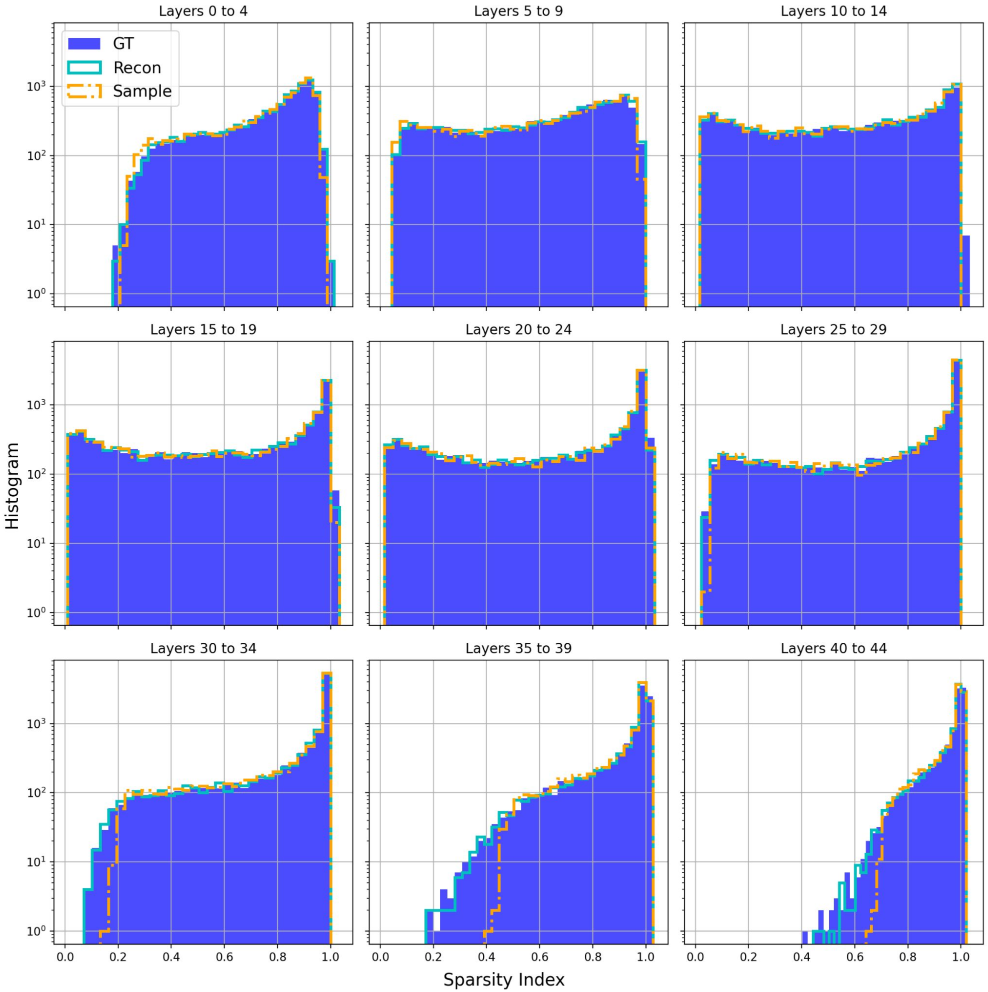
Gen



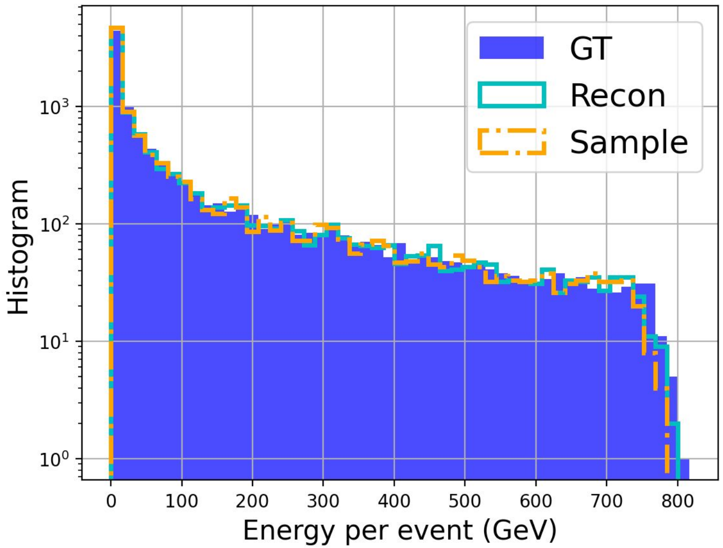


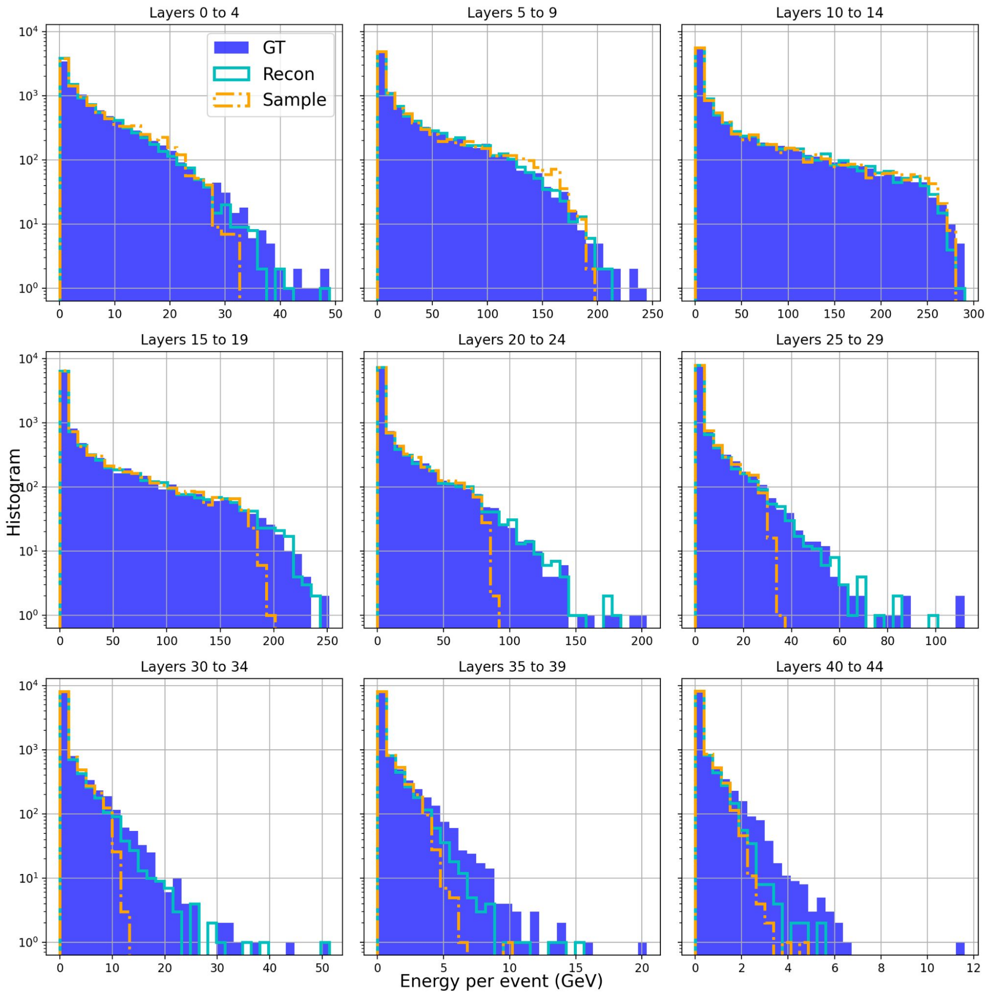
Current: Pegasus, 302 x 4, old KL training, no freezing, potentially better samples next week





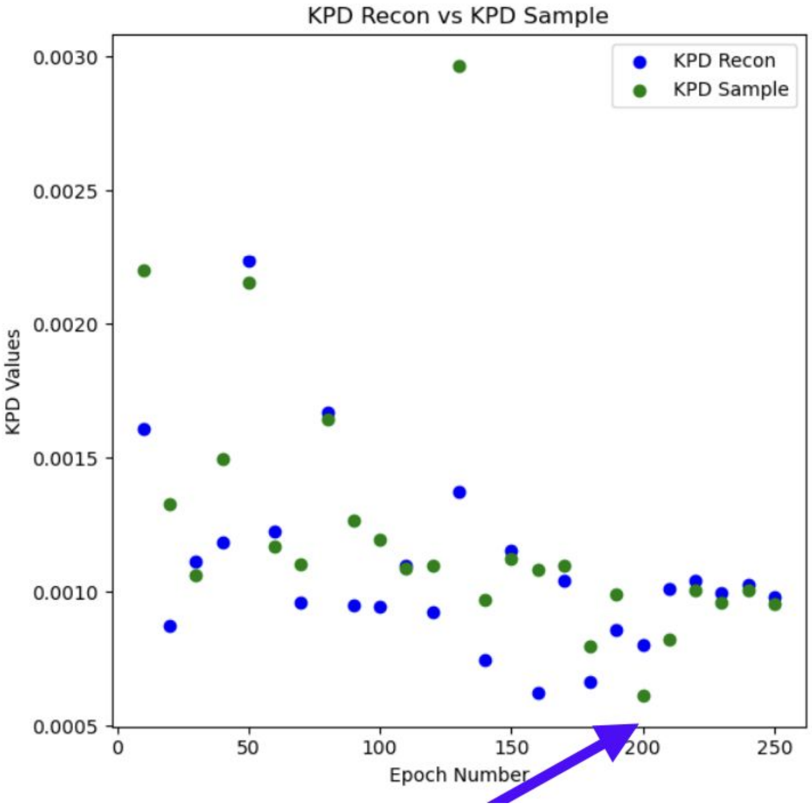
Current: Pegasus, 302 x 4, old KL training, no freezing, potentially better samples next week





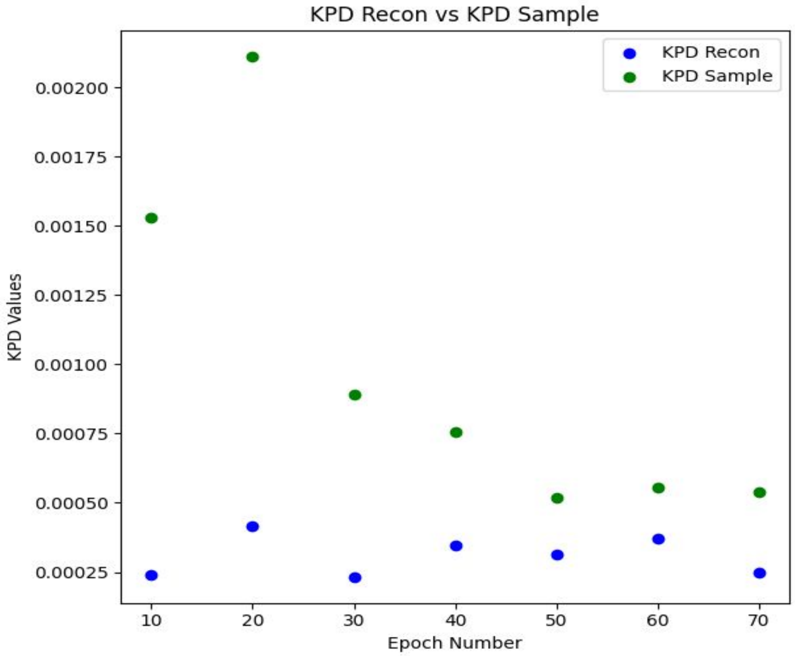
KPD

Current: Pegasus, 302 x 4, old KL training, no freezing, potentially better samples next week



Old

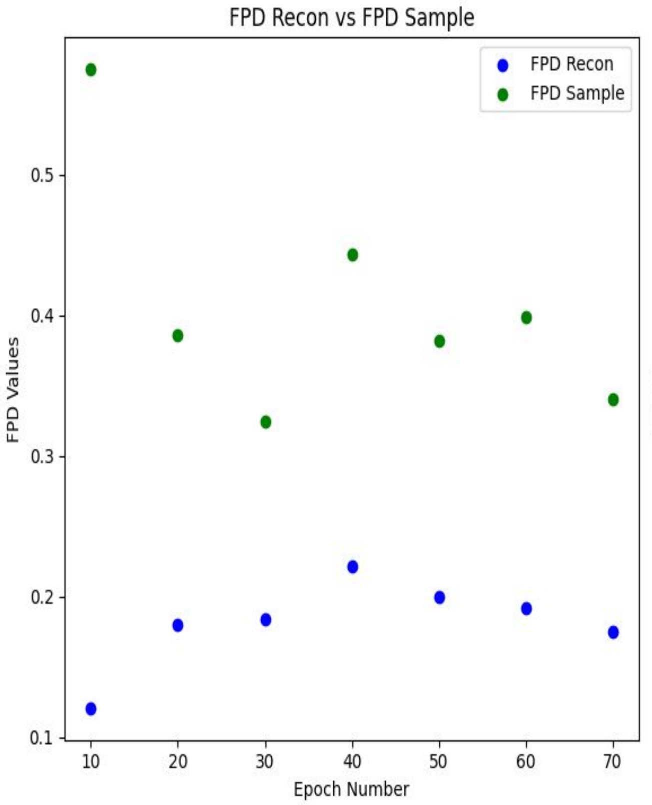
New

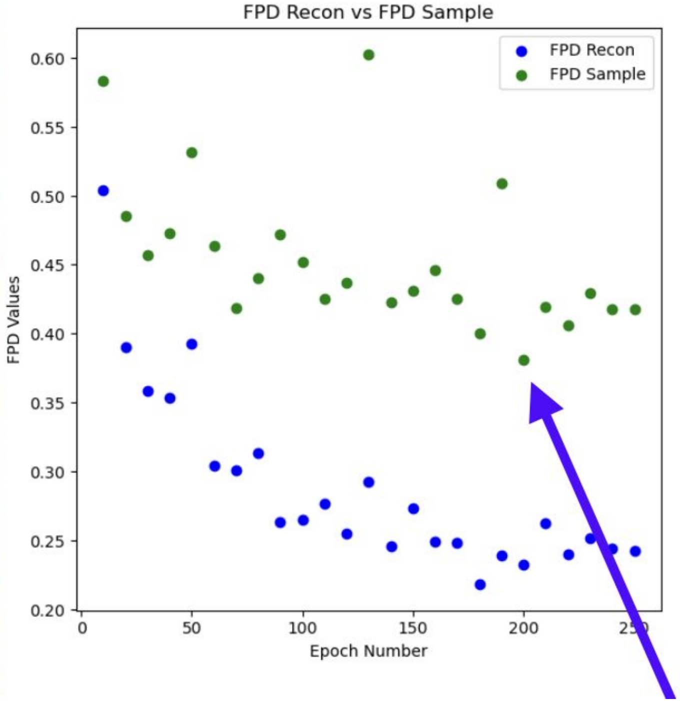


FPD

Current: Pegasus, 302 x 4, old KL training, no freezing, potentially better samples next week

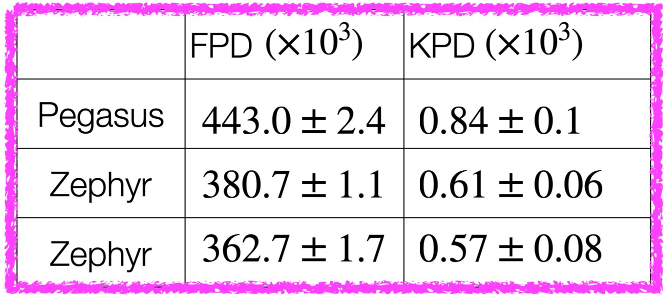
New





Old

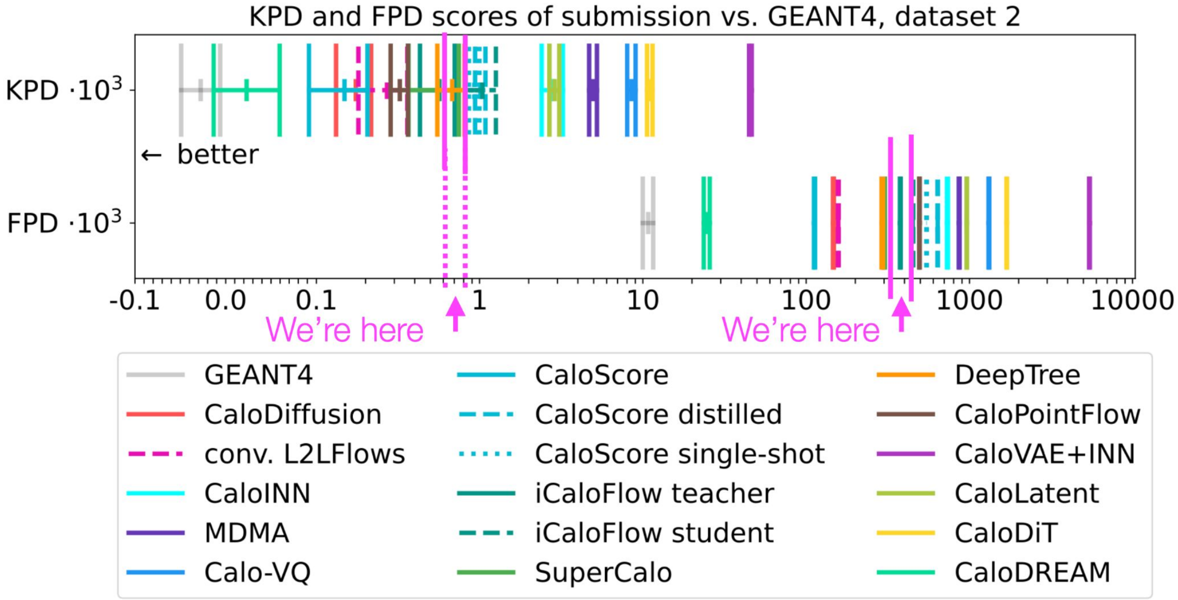
Previous models



	FPD ($\times10^3$)	KPD ($\times10^3$)
Pegasus	443.0 ± 2.4	0.84 ± 0.1
Zephyr	380.7 ± 1.1	0.61 ± 0.06
Zephyr	362.7 ± 1.7	0.57 ± 0.08

New

FPD	KPD
340+- 1.44	0.54+- 0.06



We're here

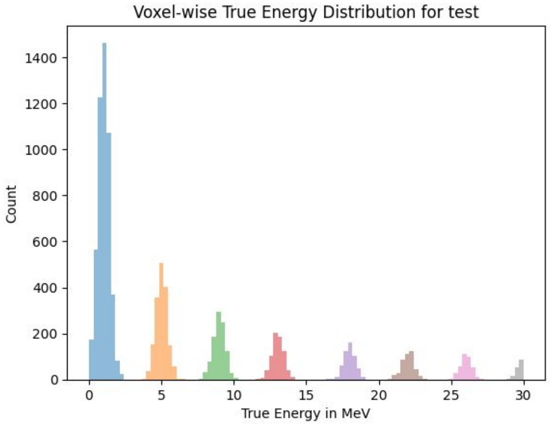
We're here

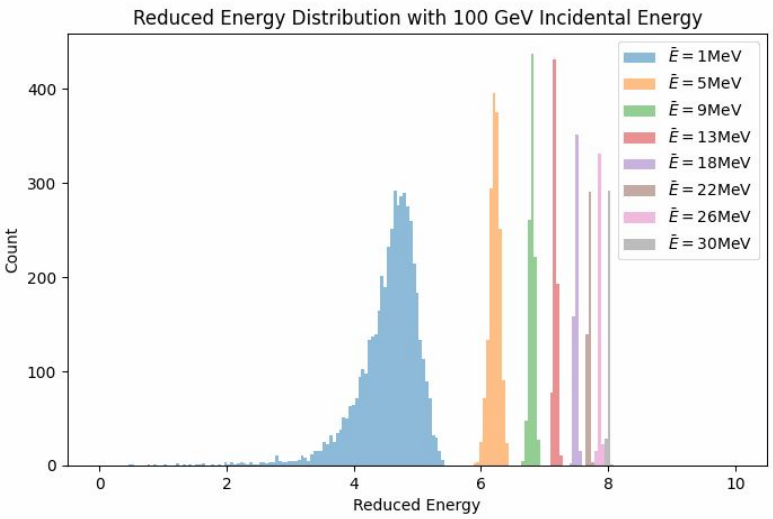
New scaling

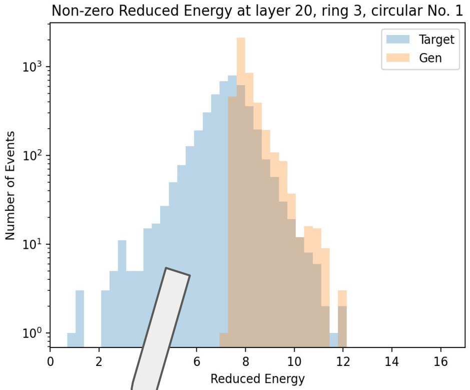
- Better granularity
  - Reason: More adjustable range on b (negative weight)
    - Literally it can control the granularity
    - Why? Next page
- Better performance
  - Higher **b** needs higher **a** for balance.
  - More aggressively high energy region prediction
  - Higher **a**, better e_ratio hist
- Less stable:
  - Rarely, may give high prediction for a single voxel
    - Predict range (0-8), but result in 14….. e^6 order higher than it should.
    - Solved by adding scaled Sigmoid function (0-9) in the last layer of decoder.

Review the old scaling method

```
def _reduce(self, in_data, E_inc, R=1e-7):
    ϵ = in_data / E_inc
    x = R + (1-2*R)*ϵ
    u = torch.log(x/(1-x)) - torch.log(torch.tensor([R/(1-R)]))
    return u
```

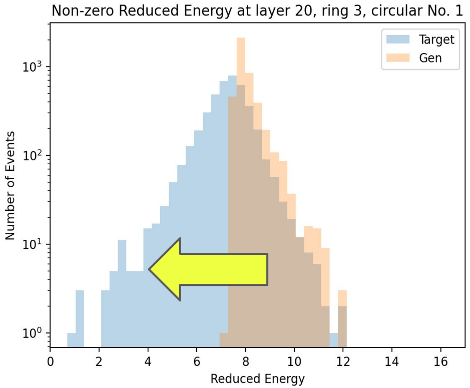






Non-zero Reduced Energy at layer 20, ring 3, circular No. 1

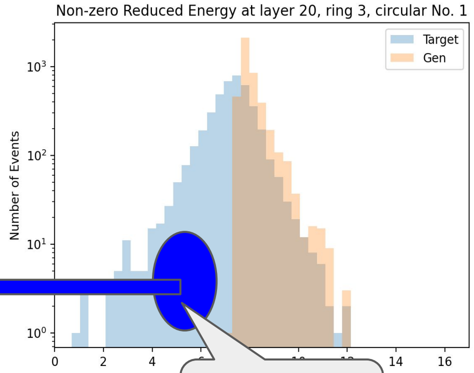
Higher b



Non-zero Reduced Energy at layer 20, ring 3, circular No. 1

Predictions will move left across whole range of E_inc

Lower prediction for high E_inc

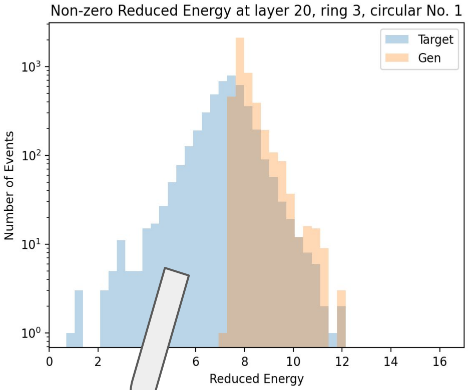


Non-zero Reduced Energy at layer 20, ring 3, circular No. 1

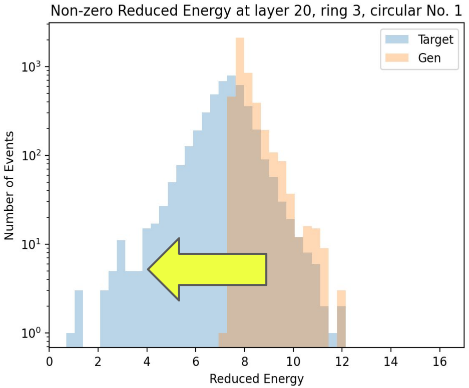
Low prediction for low E_inc

Inverse scale

Tons of extremely low values

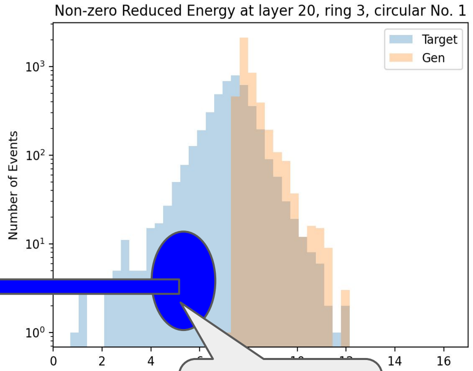

Non-zero Reduced Energy at layer 20, ring 3, circular No. 1

Higher b


Non-zero Reduced Energy at layer 20, ring 3, circular No. 1

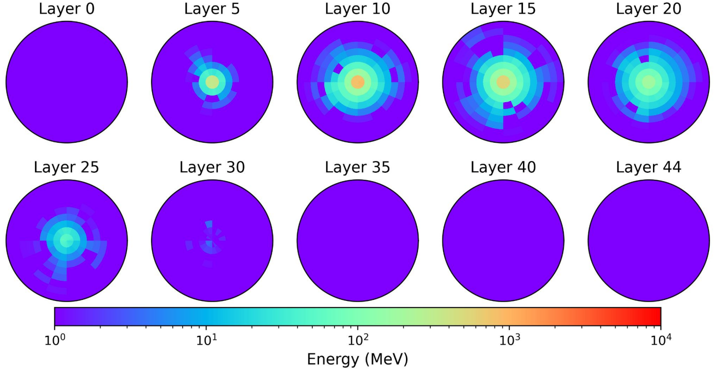
Predictions will move left across whole range of E_inc

Lower prediction for high E_inc


Non-zero Reduced Energy at layer 20, ring 3, circular No. 1

Low prediction for low E_inc

Inverse scale



Next

- Use Zephyr
- New KL Loss training
- Freeze from best FPD/KPD Recon epoch
- More aggressive parameters

Thanks!