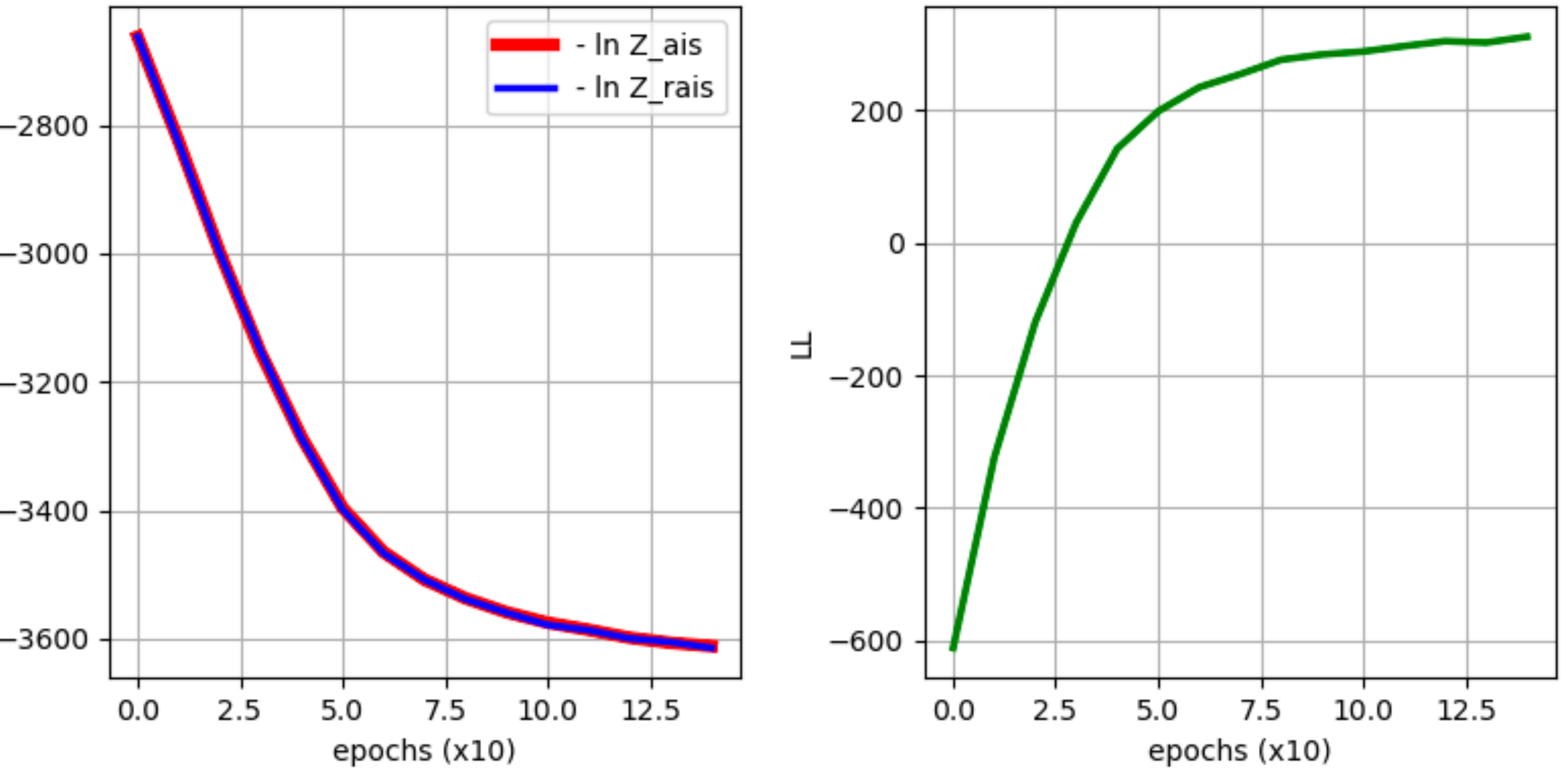
QVAE w/ Pegasus

Jan 22nd

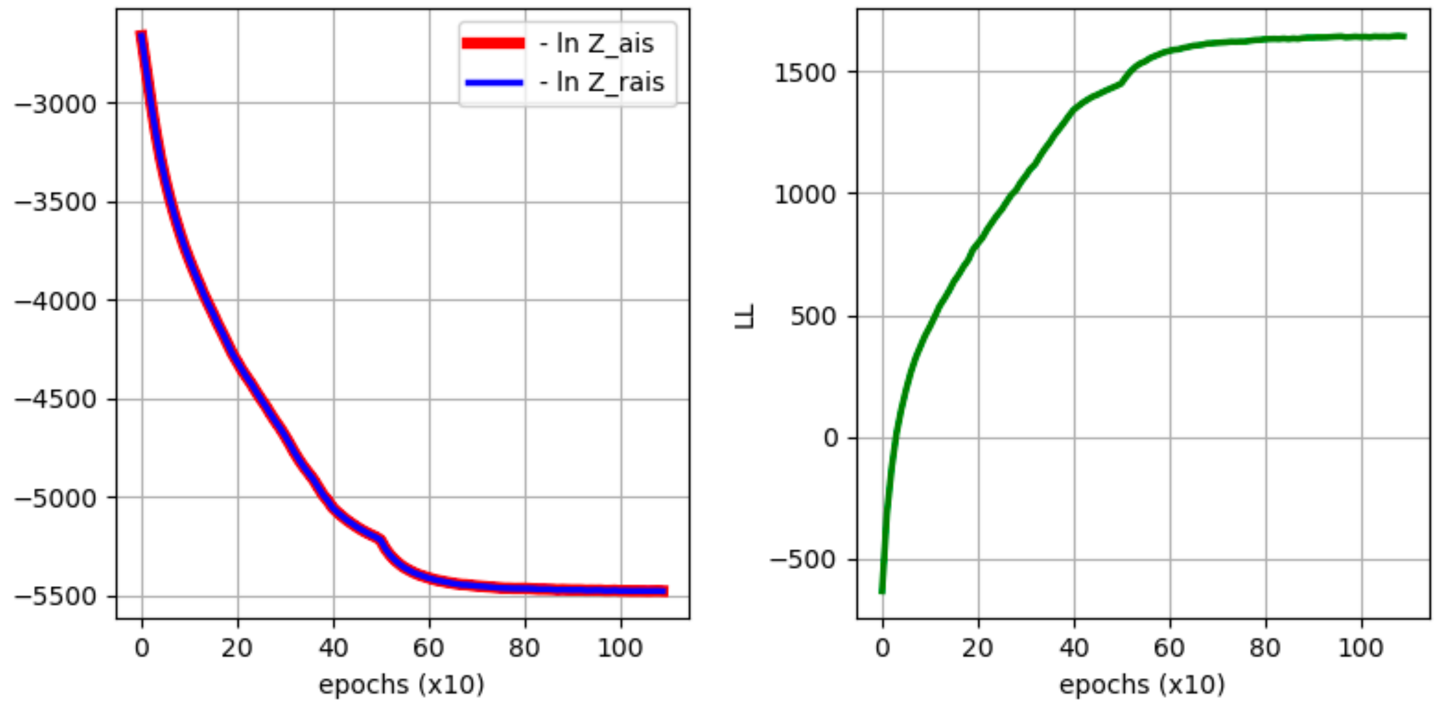
Models

- Drawn-cosmos — Conditionalized via concatenated energy
- Winter-glade — Conditionalized via simple energy addition to voxel array
- Misty-wind — Conditionalized via concatenated energy + voxel positional encoding v2
- Happy-sun — Conditionalized via concatenated energy + voxel positional encoding v1
- Prime-totem — Conditionalized via concatenated energy (150 epochs)

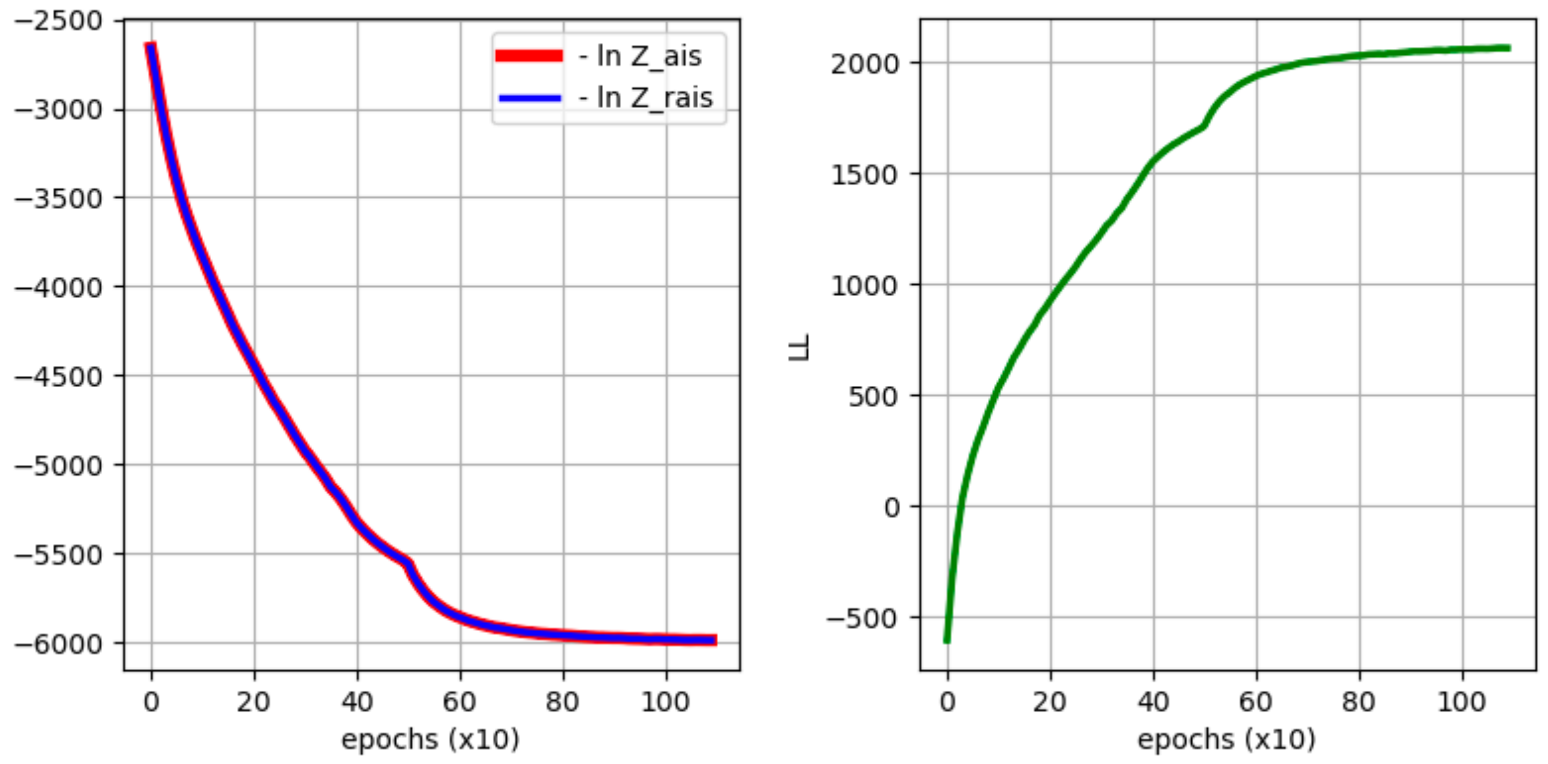
Prime-totem



Happy-sun

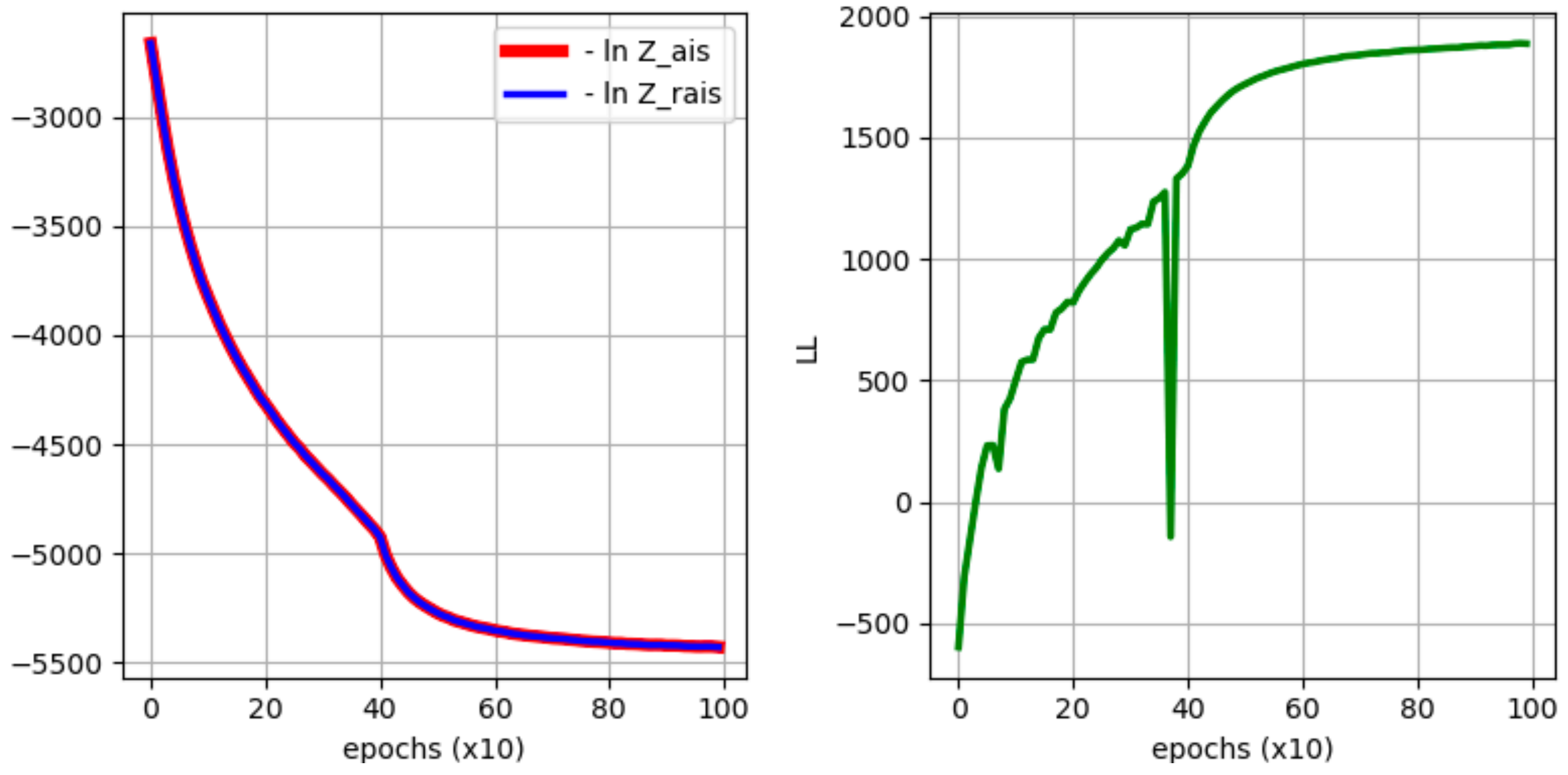


Drawn-cosmos

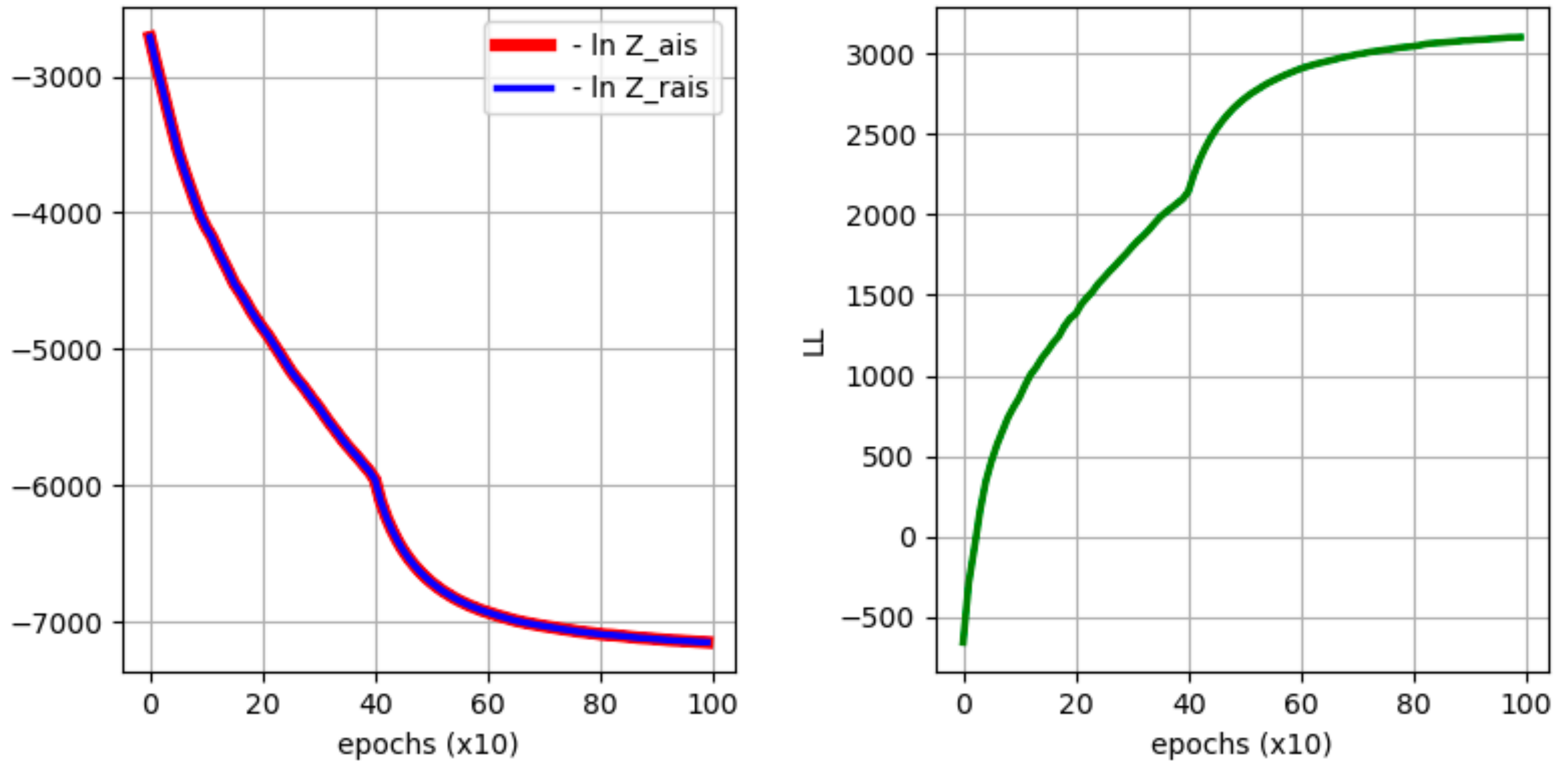


- Partition function via annealed importance sampling and reversed annealed importance sampling vs epochs. We expect both corves to converge.
- Log-likelihood vs epochs. We expect the curve to saturate for a fully-trained RBM.

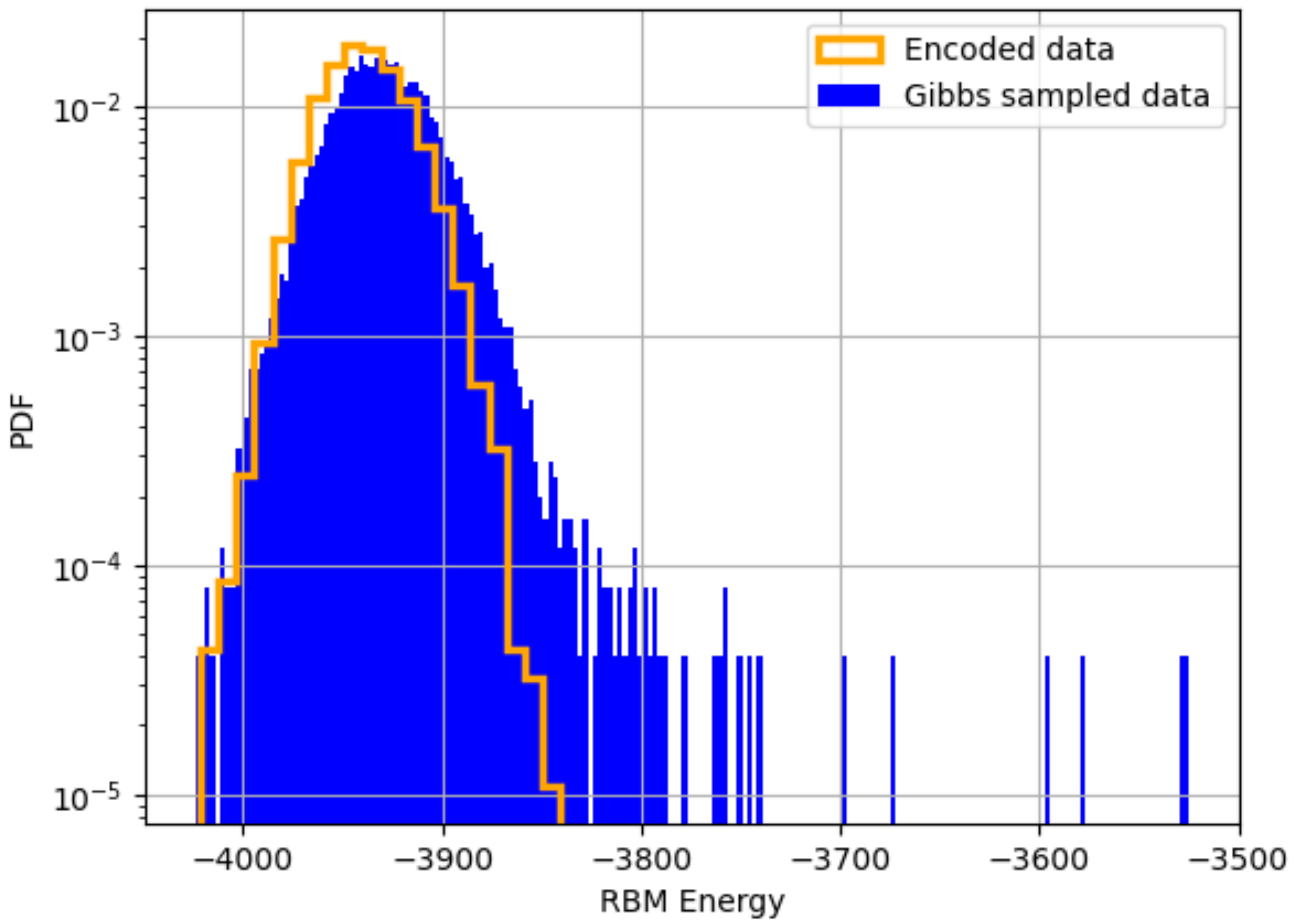
Misty-wind



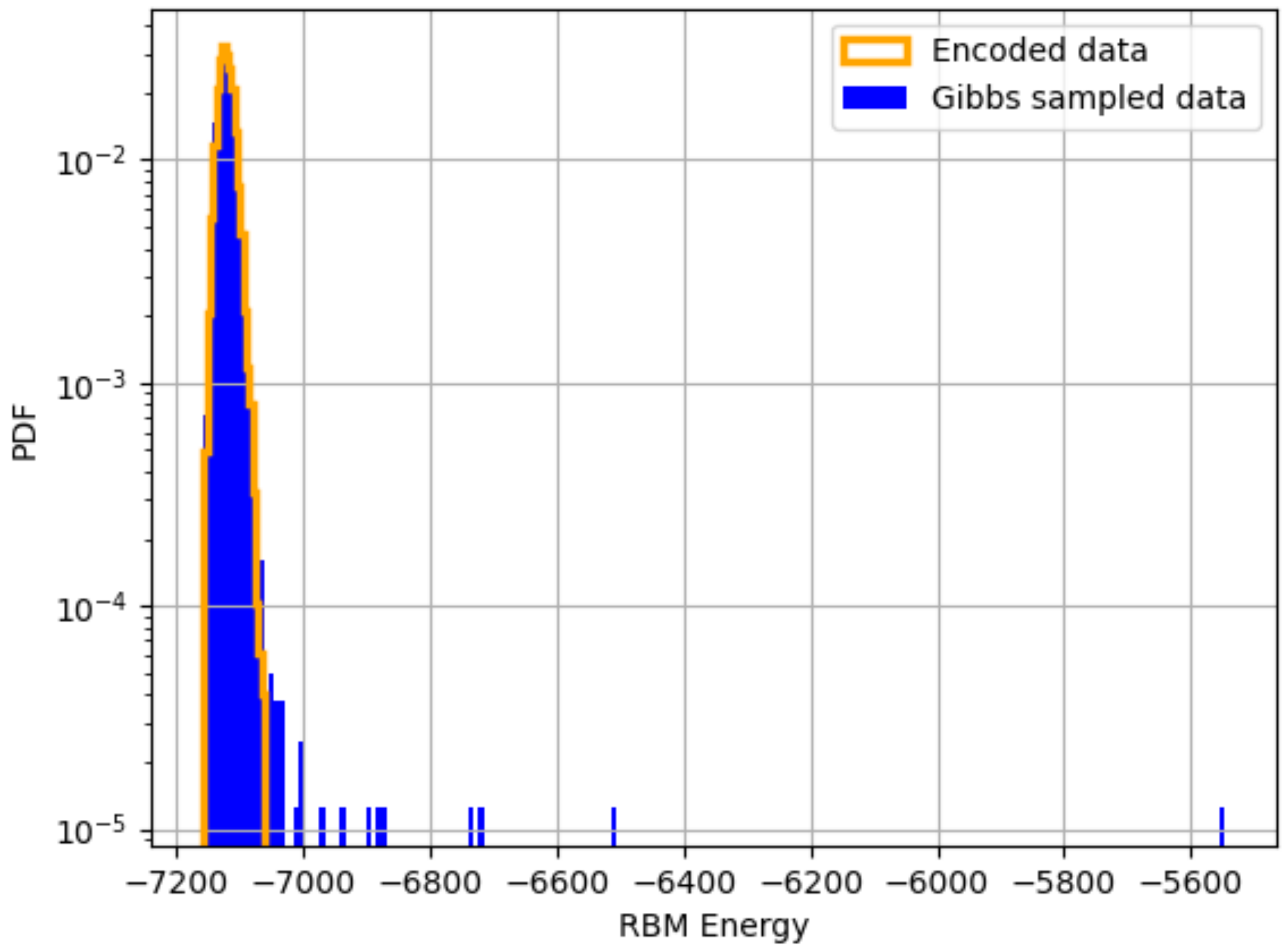
Winter-glade



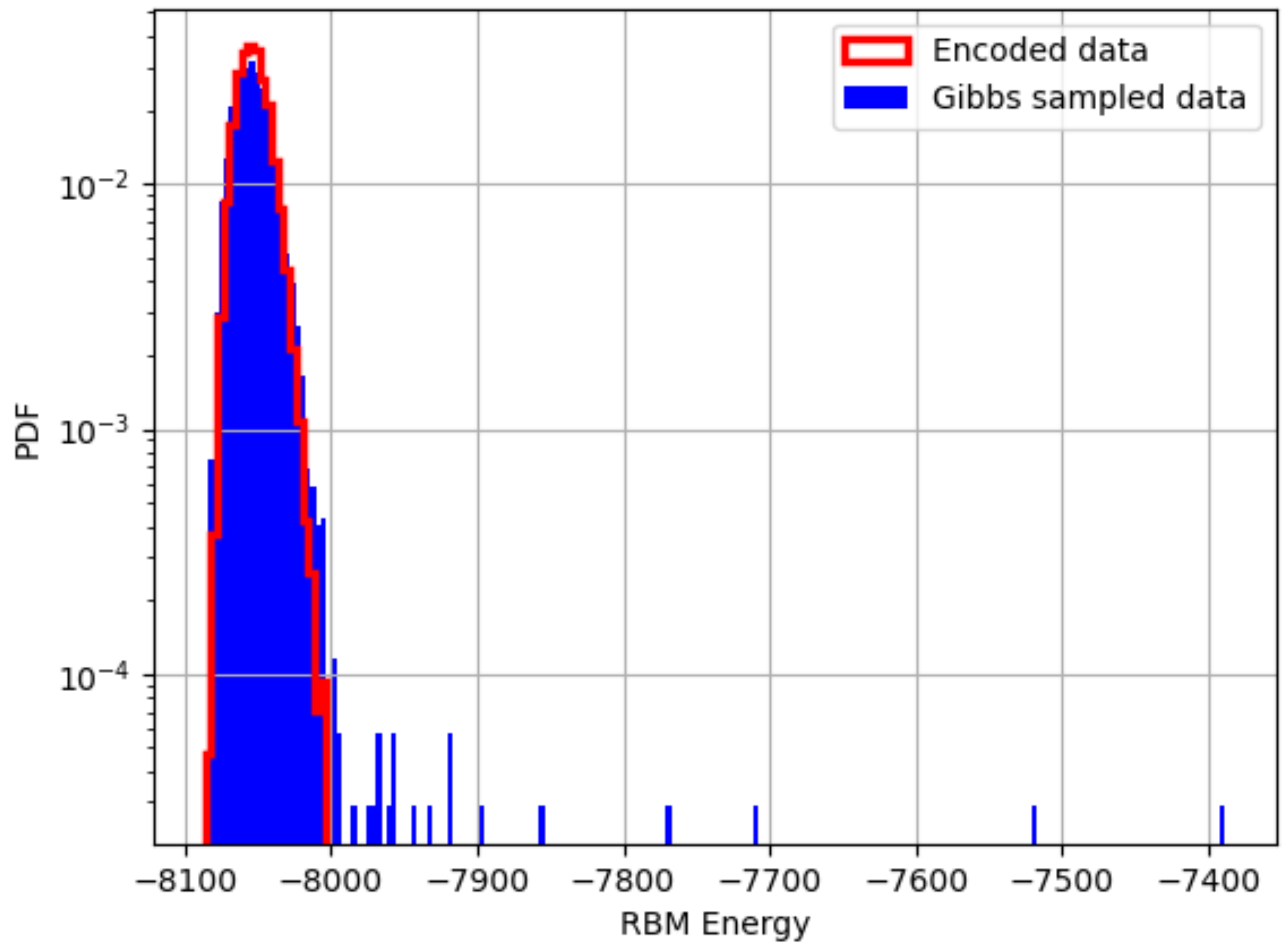
Prime-totem



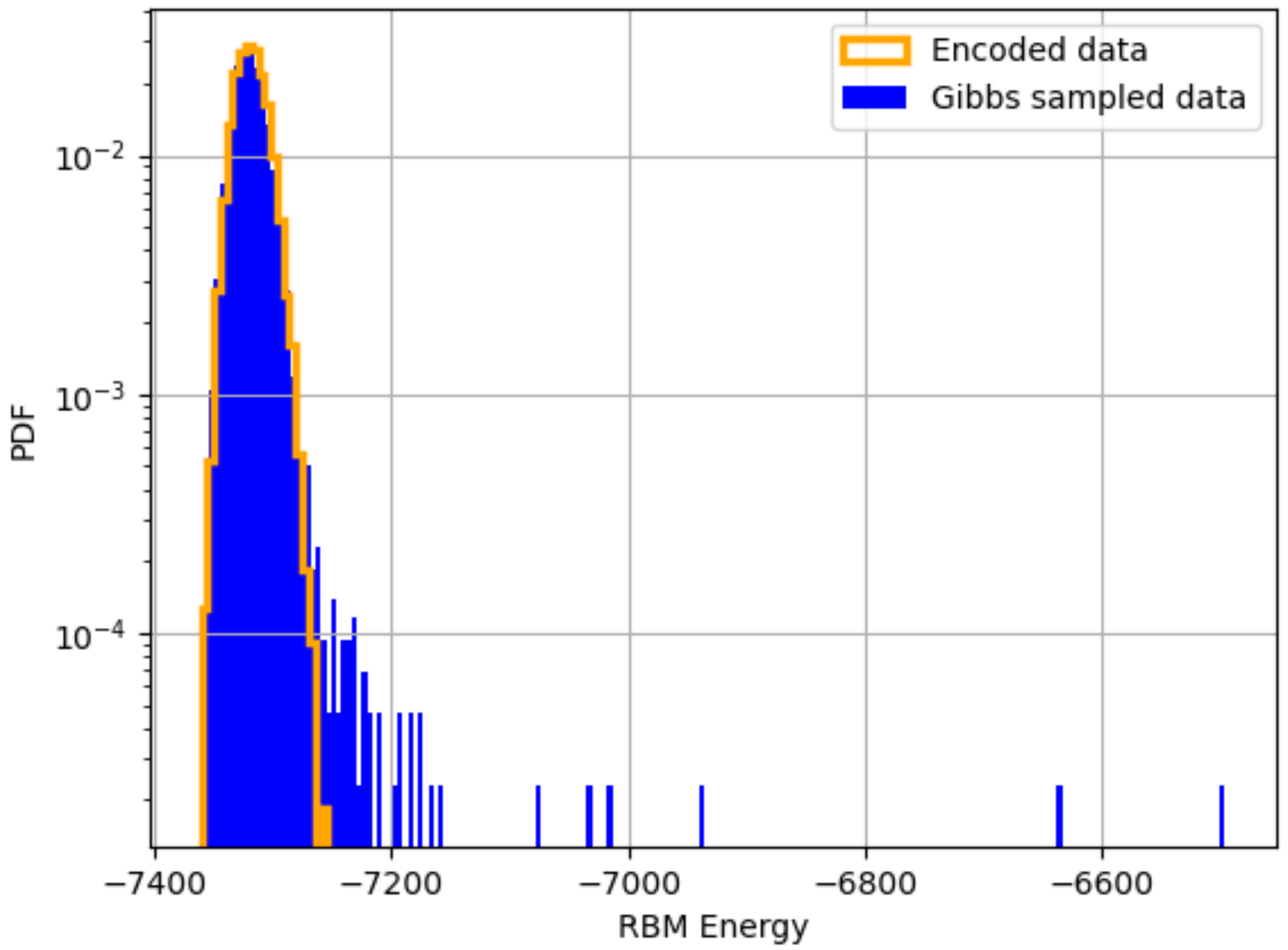
Happy-sun



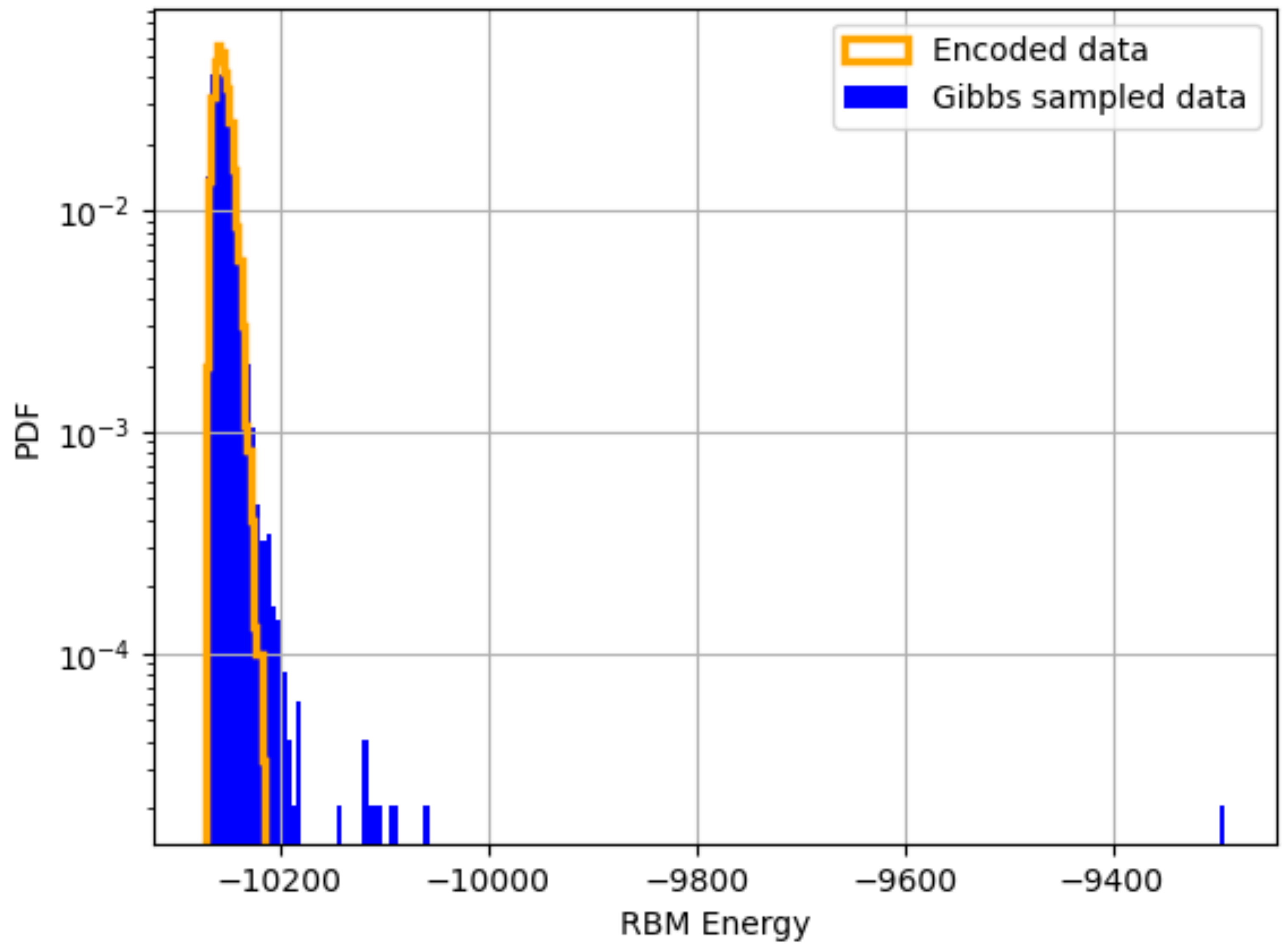
Drawn-cosmos



Misty-wind

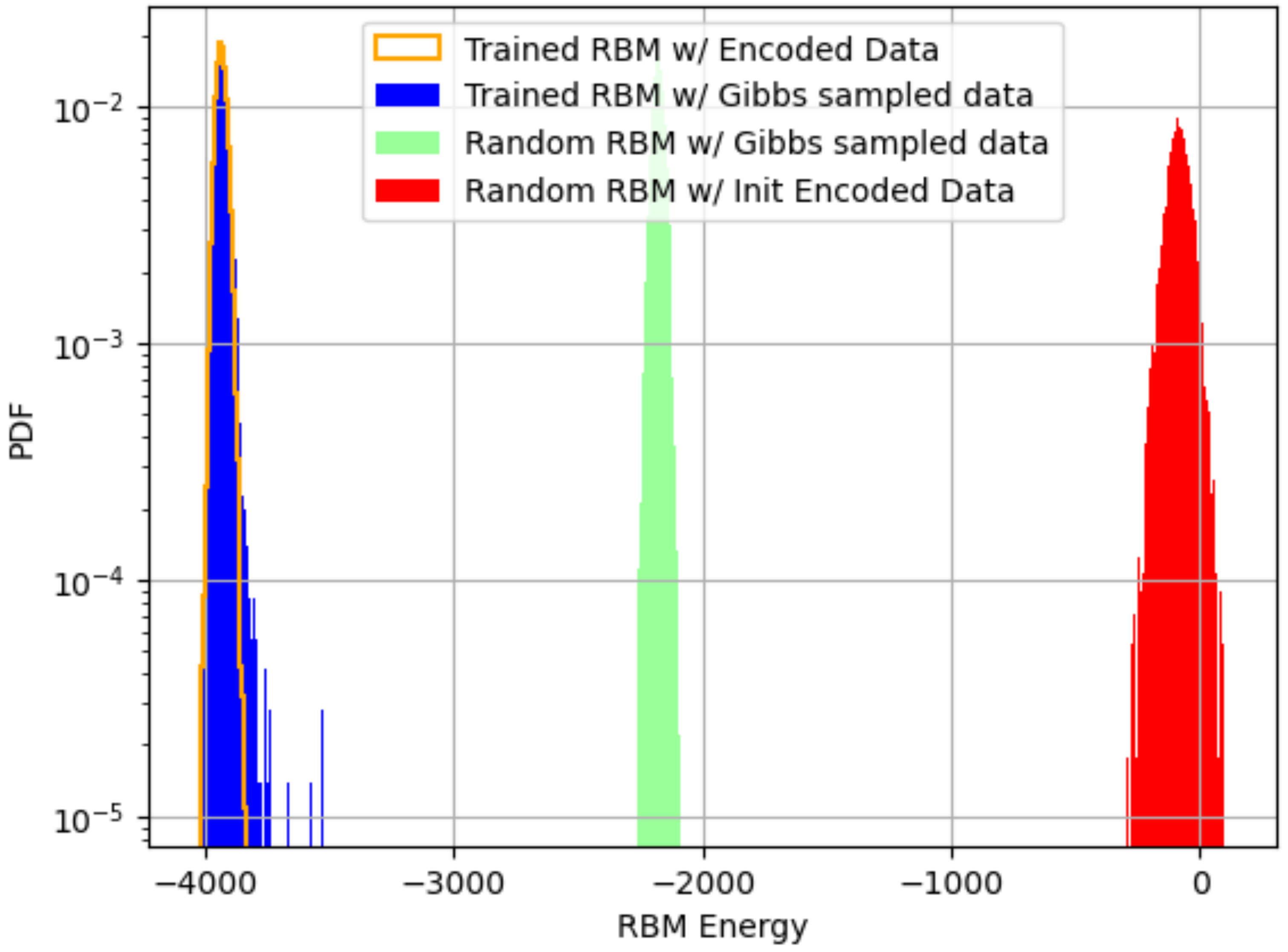


Winter-glade

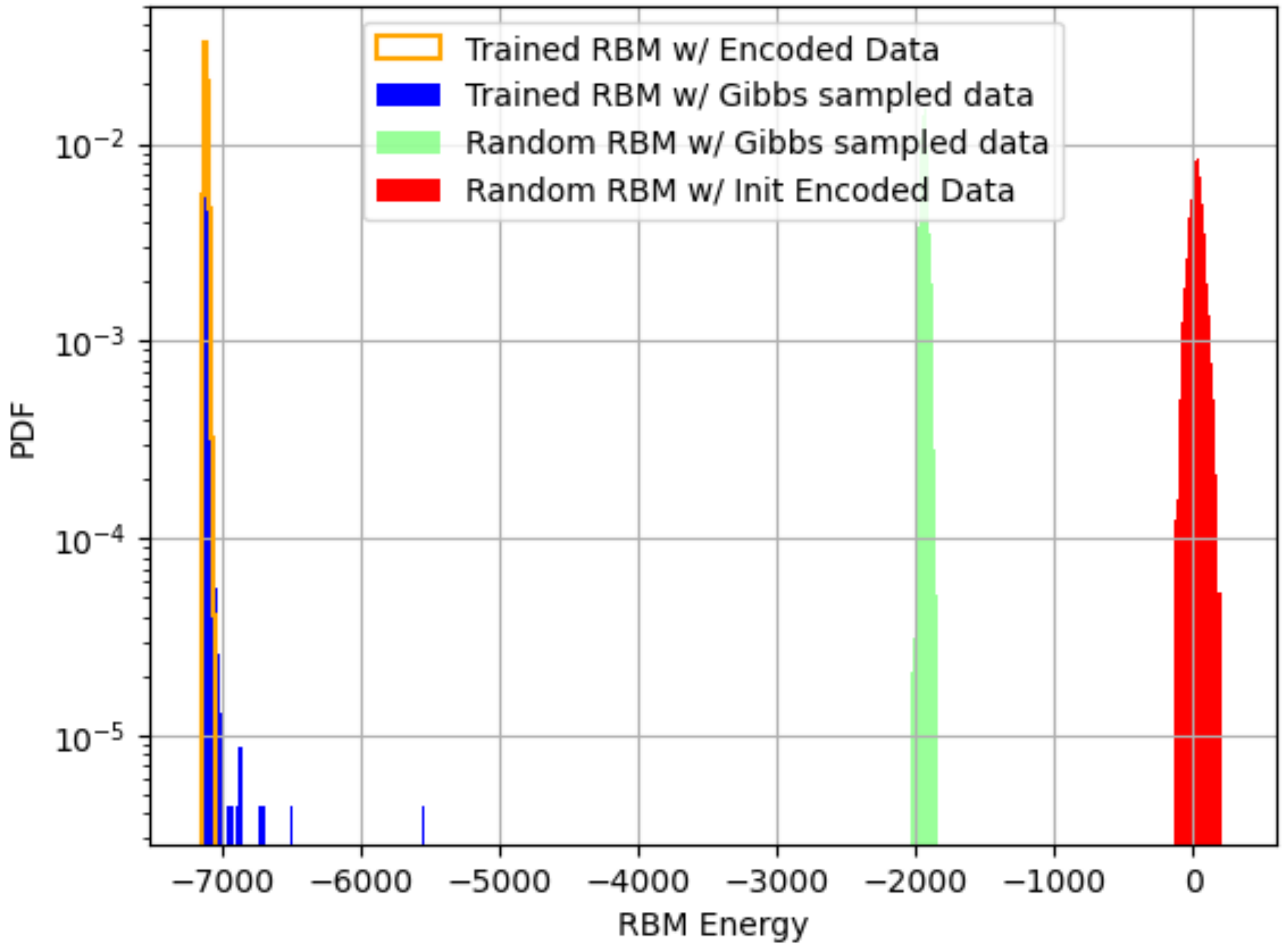


- PDFs of RBM energy of encoded validation data and Gibbs sampled data. We expected overlap between the two PDFs

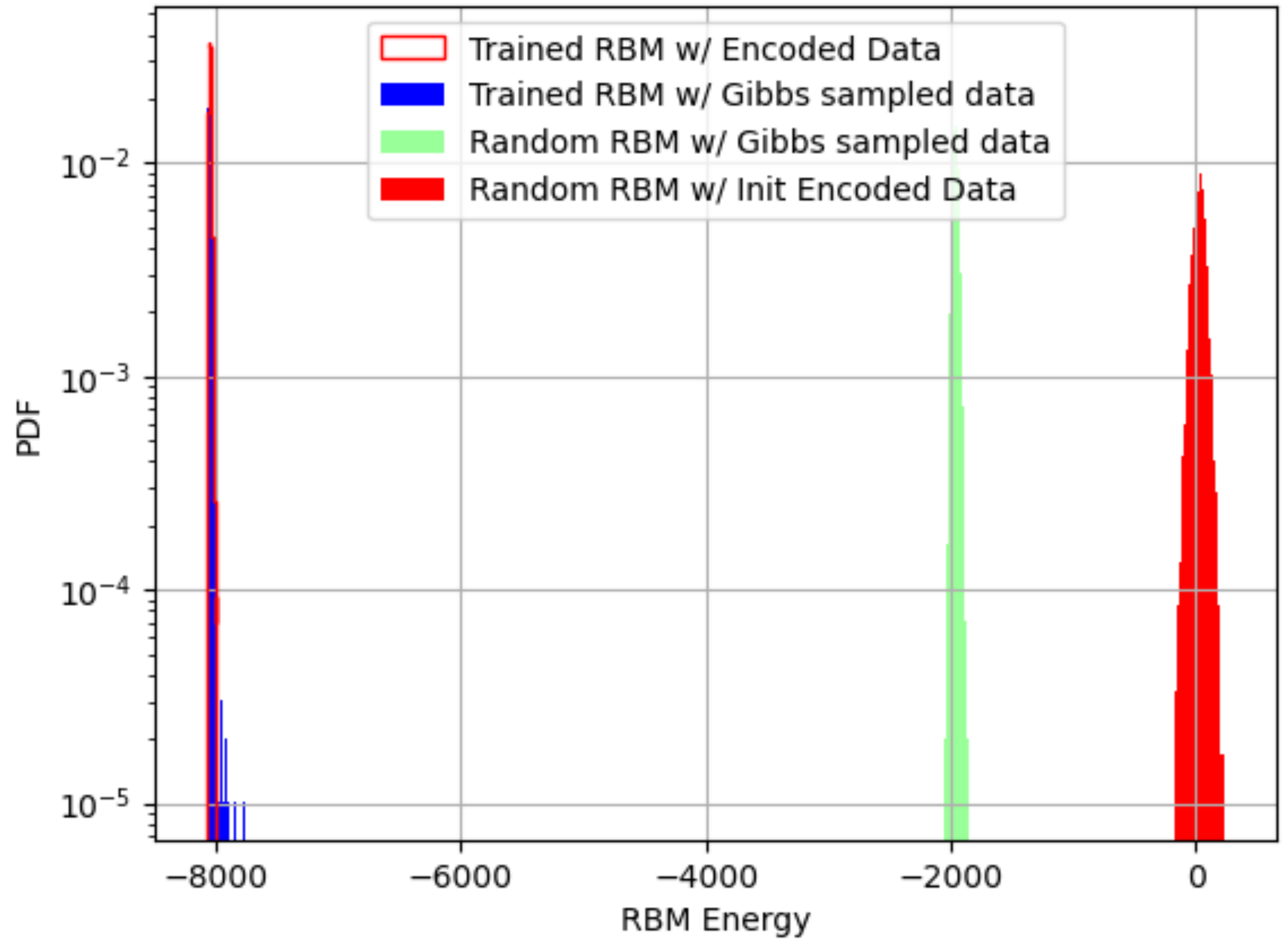
Prime-totem



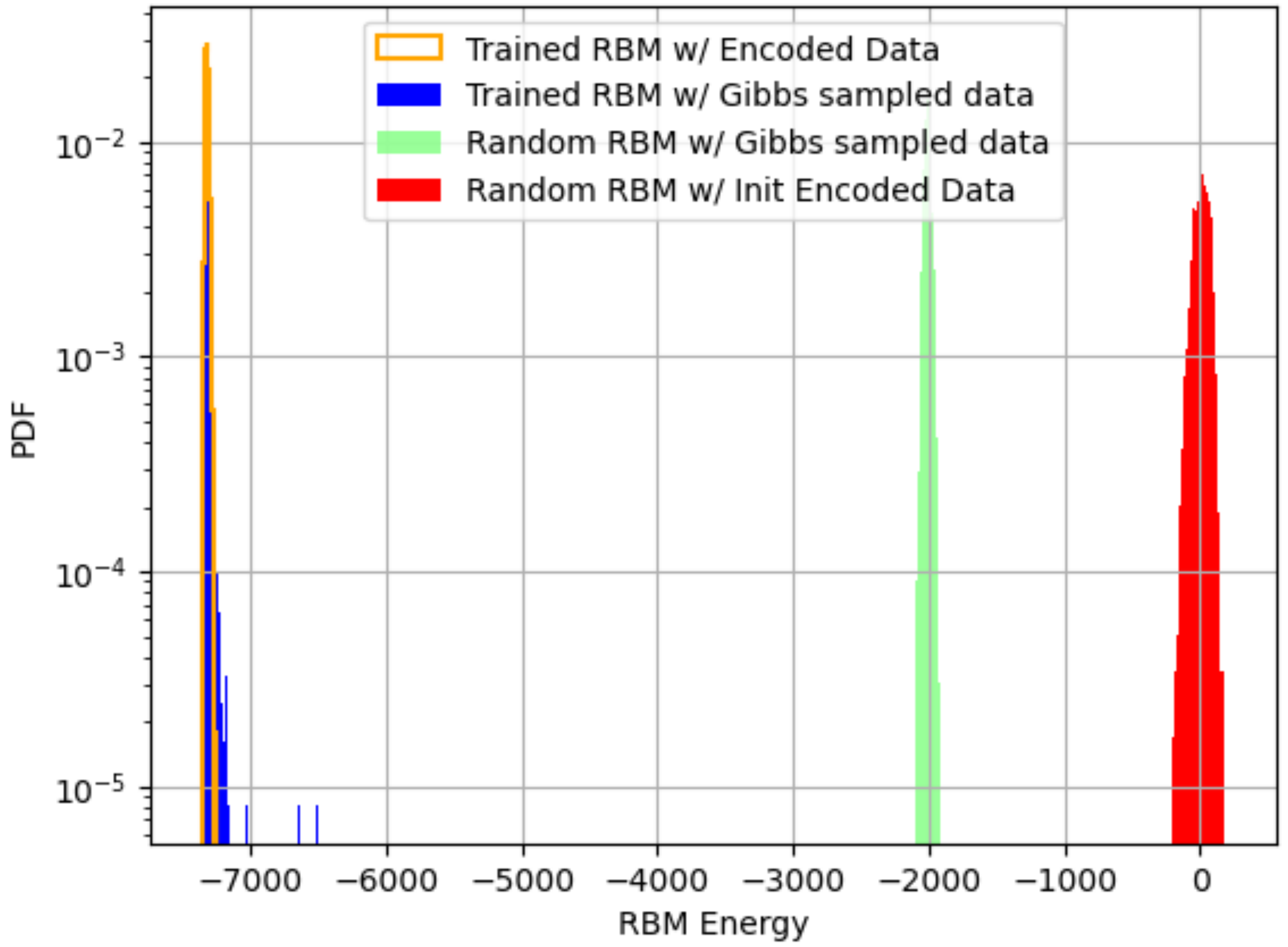
Happy-sun



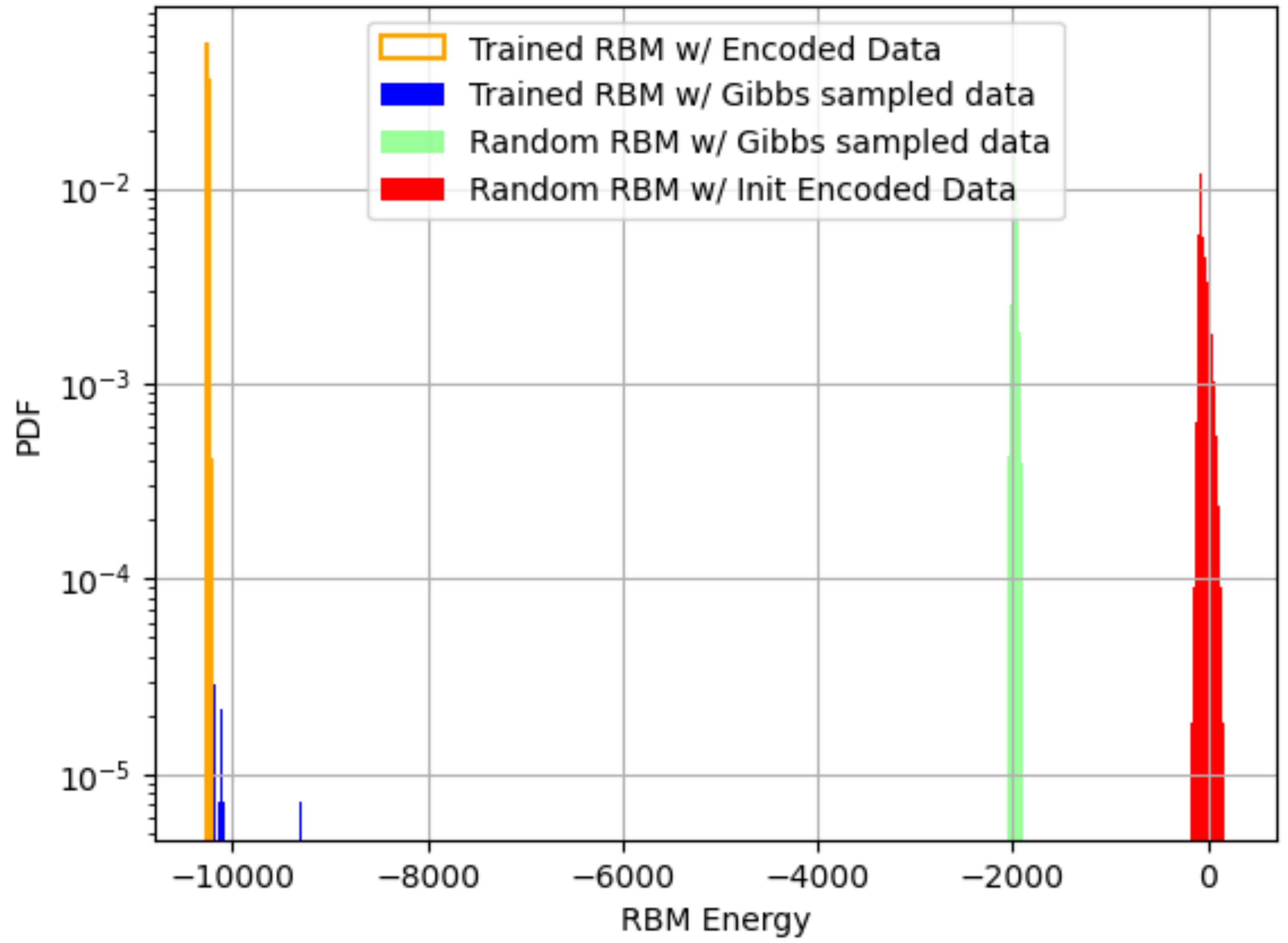
Drawn-cosmos



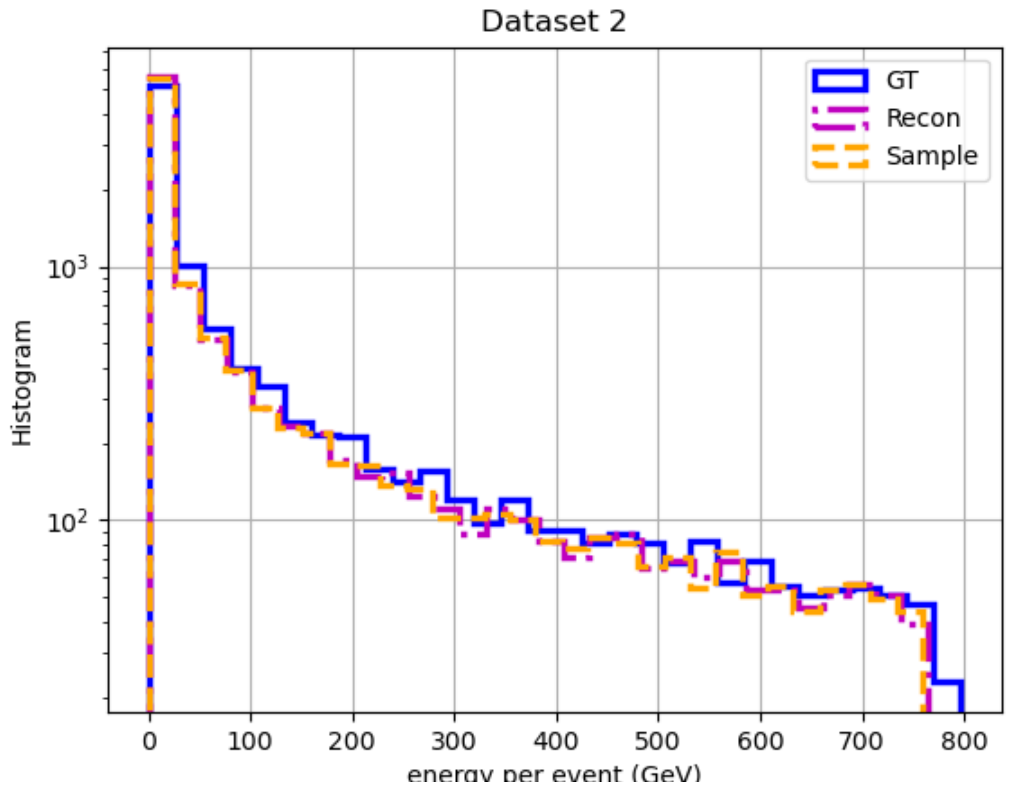
Misty-wind

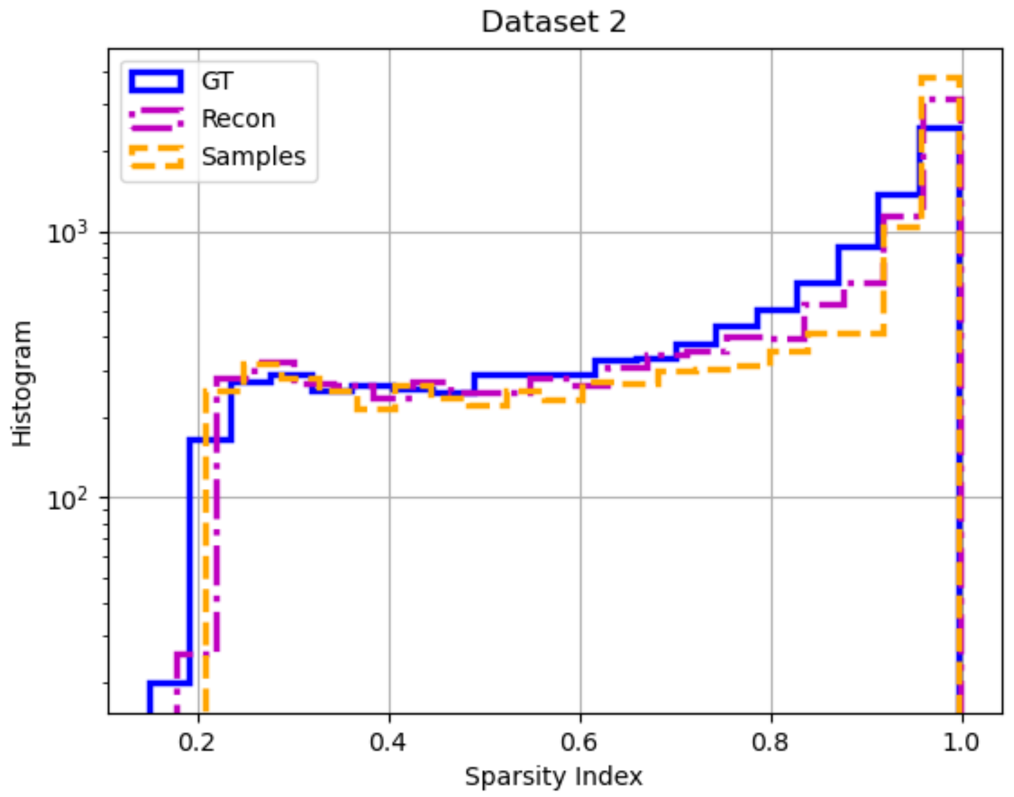


Winter-glade

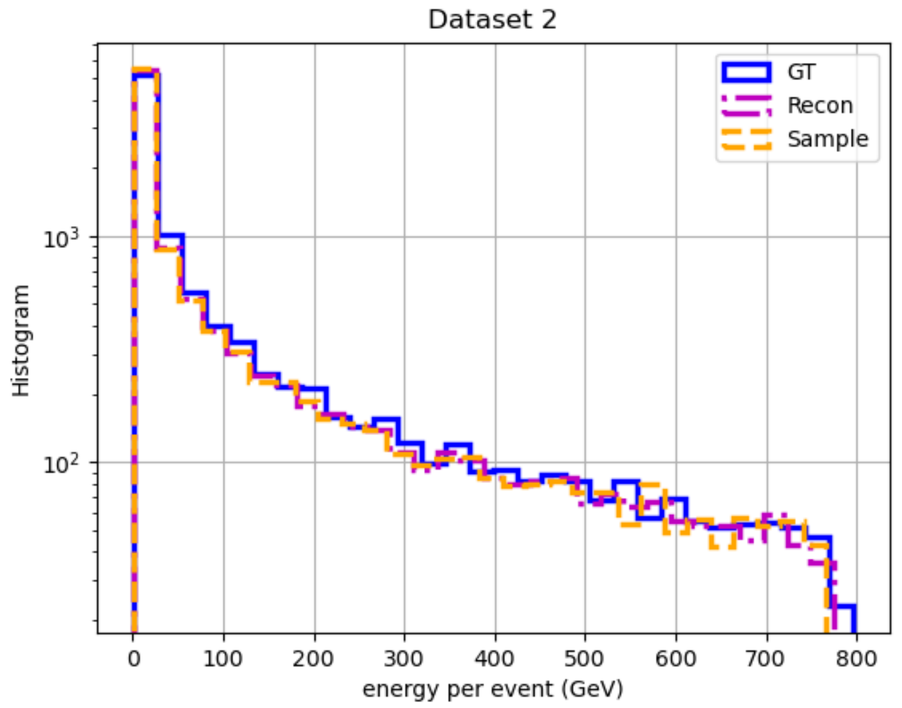


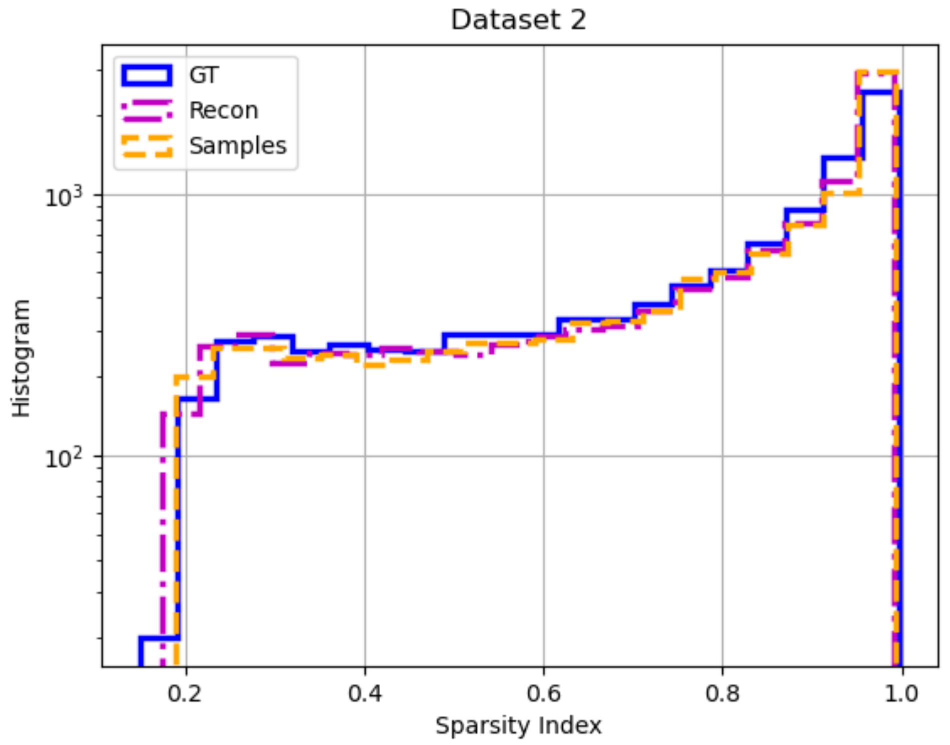
Prime-totem



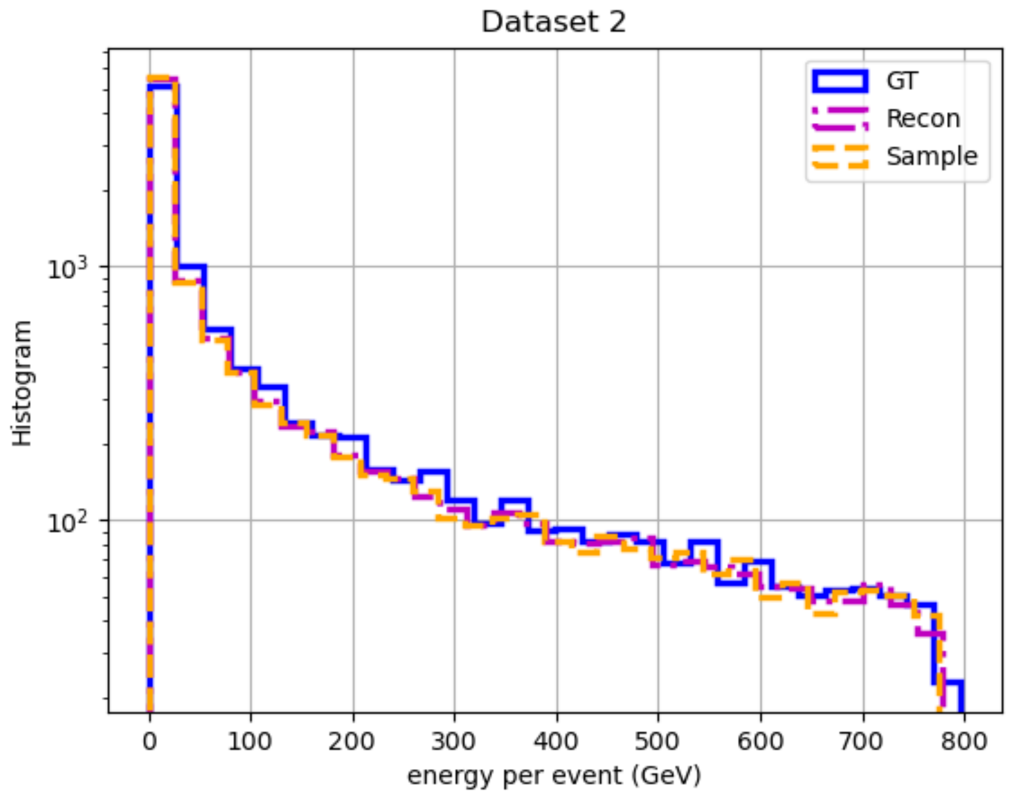


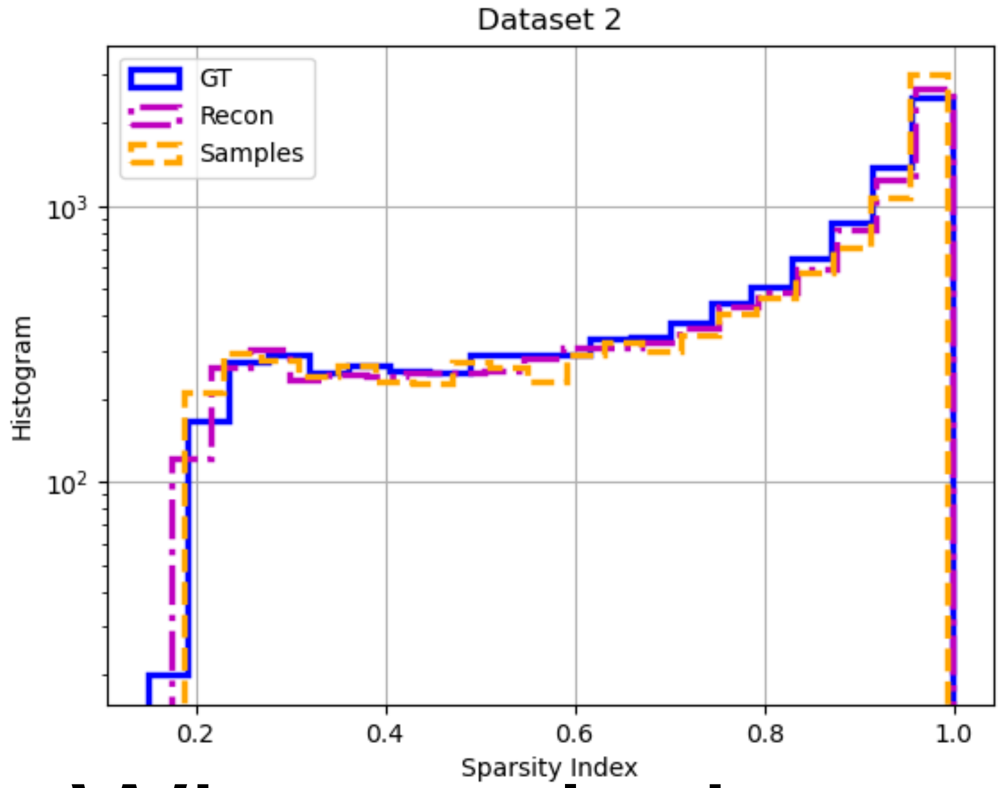
Happy-sun



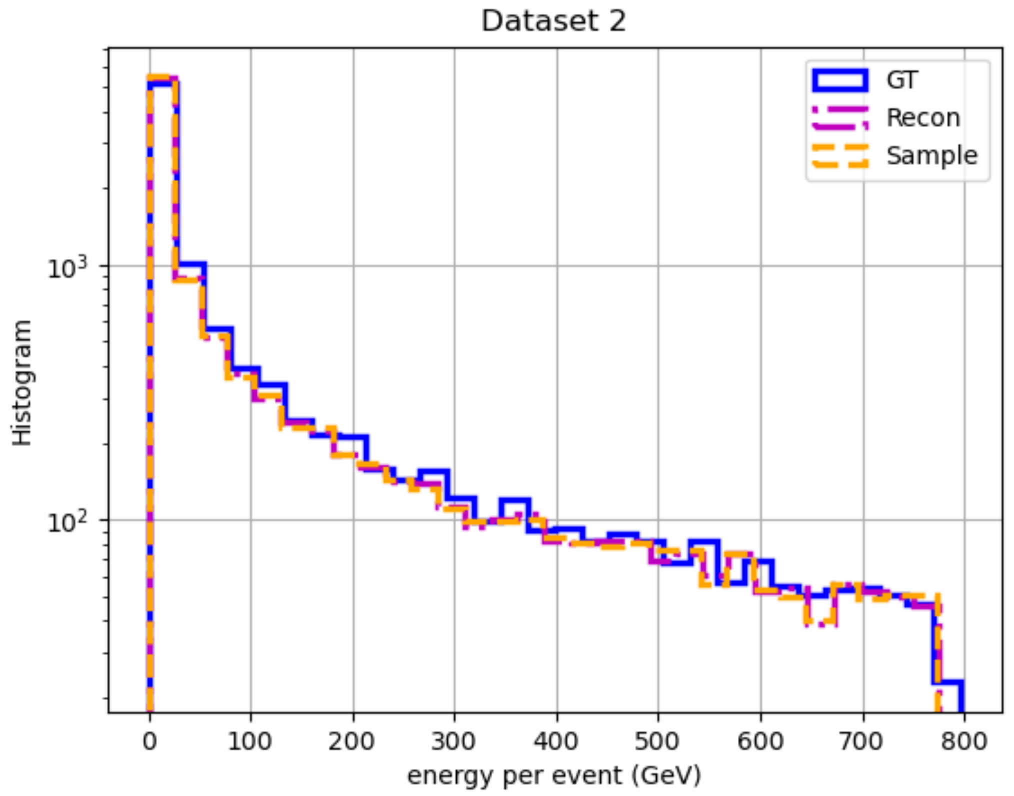


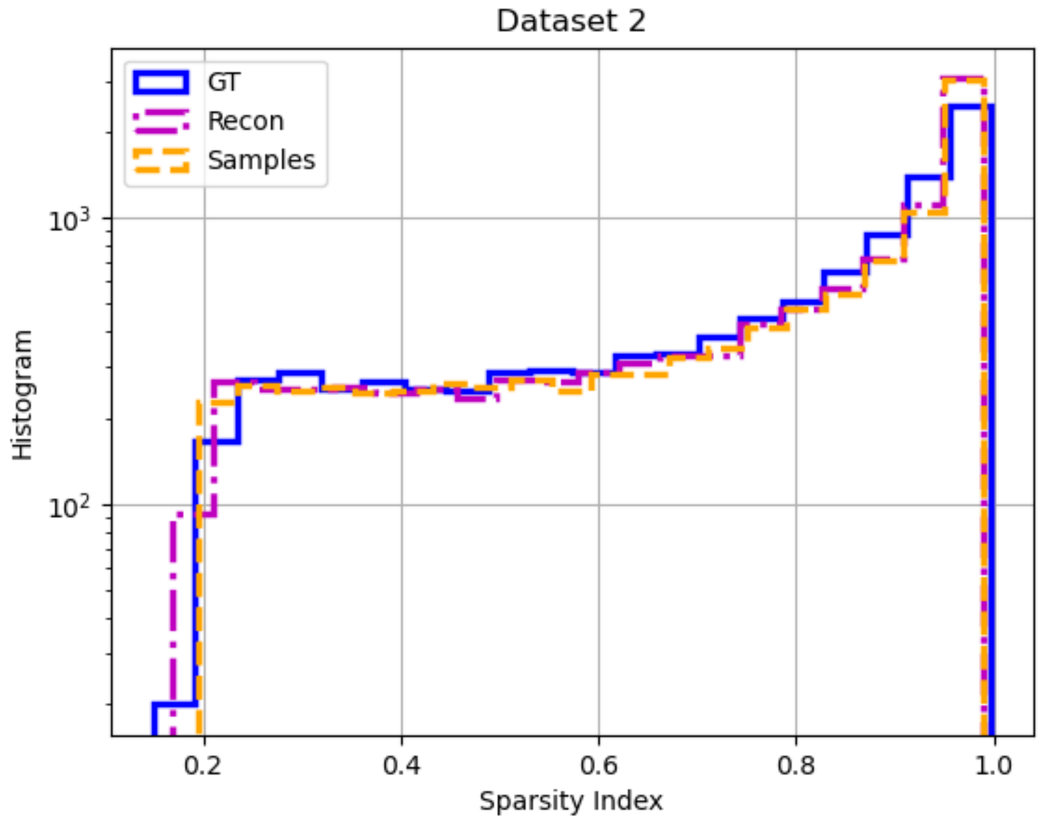
Drawn-cosmos



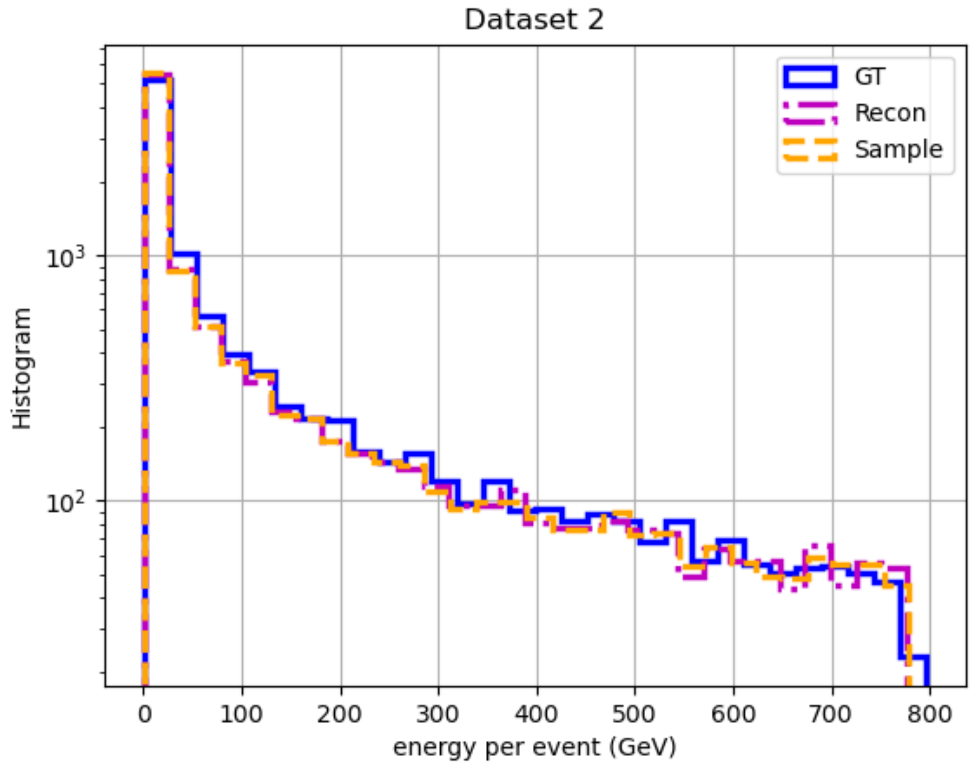


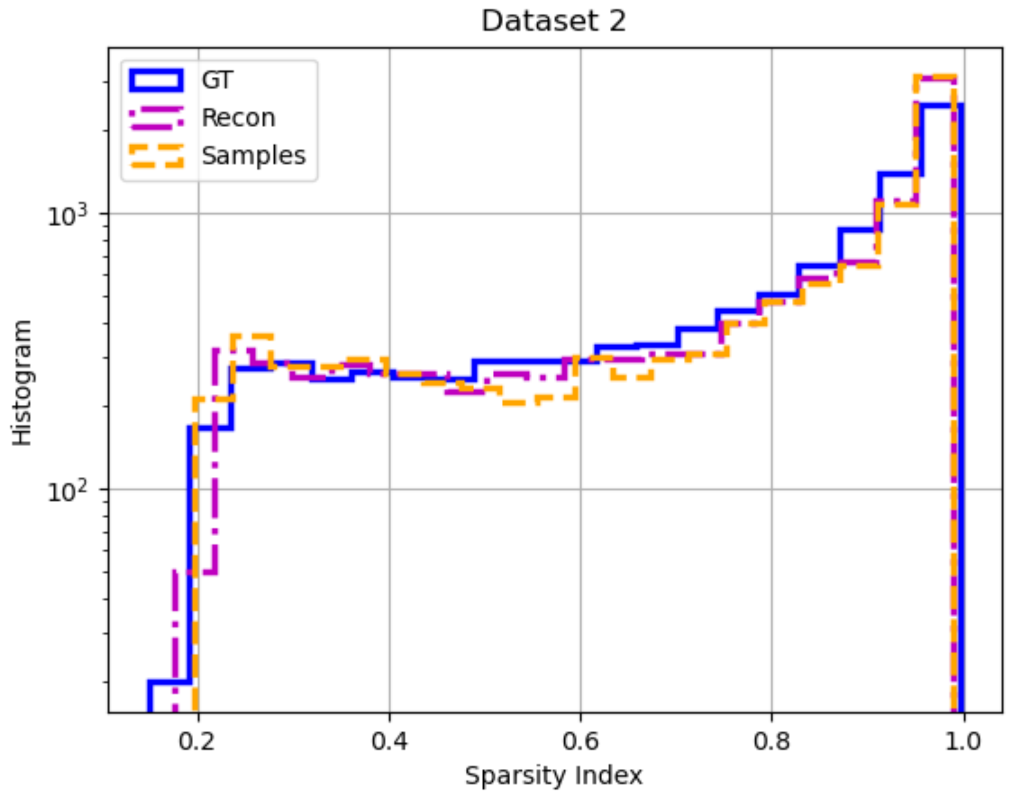
Misty-wind



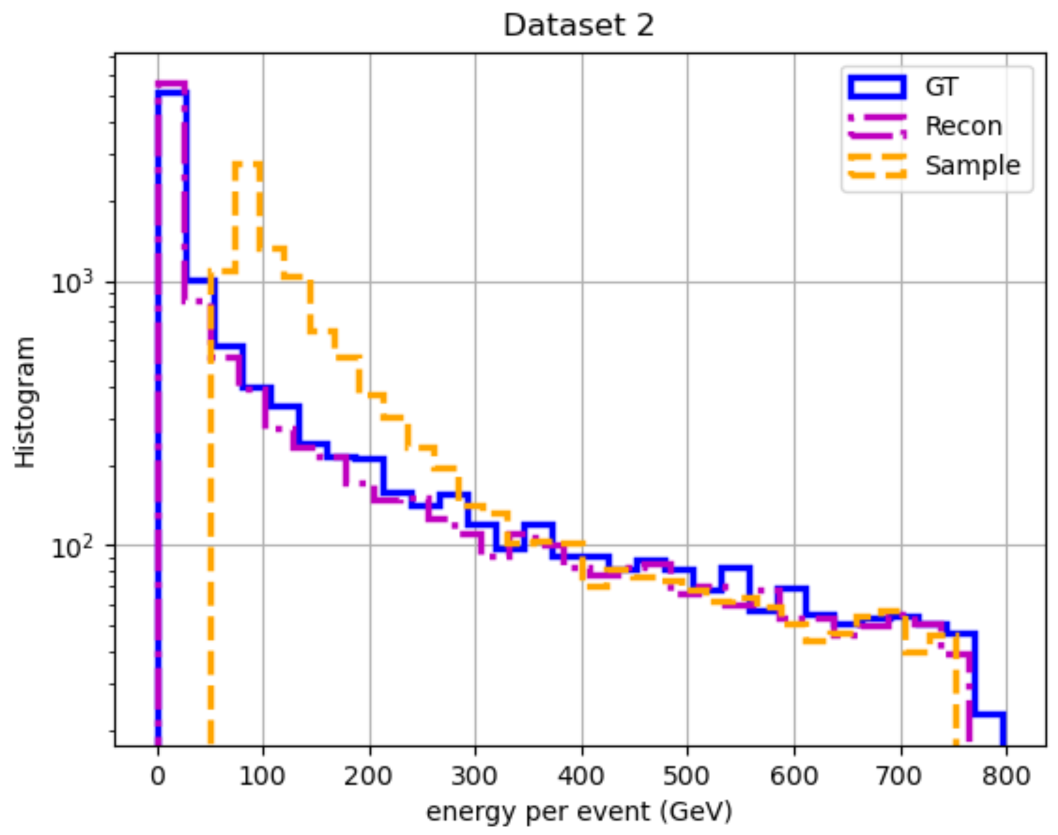


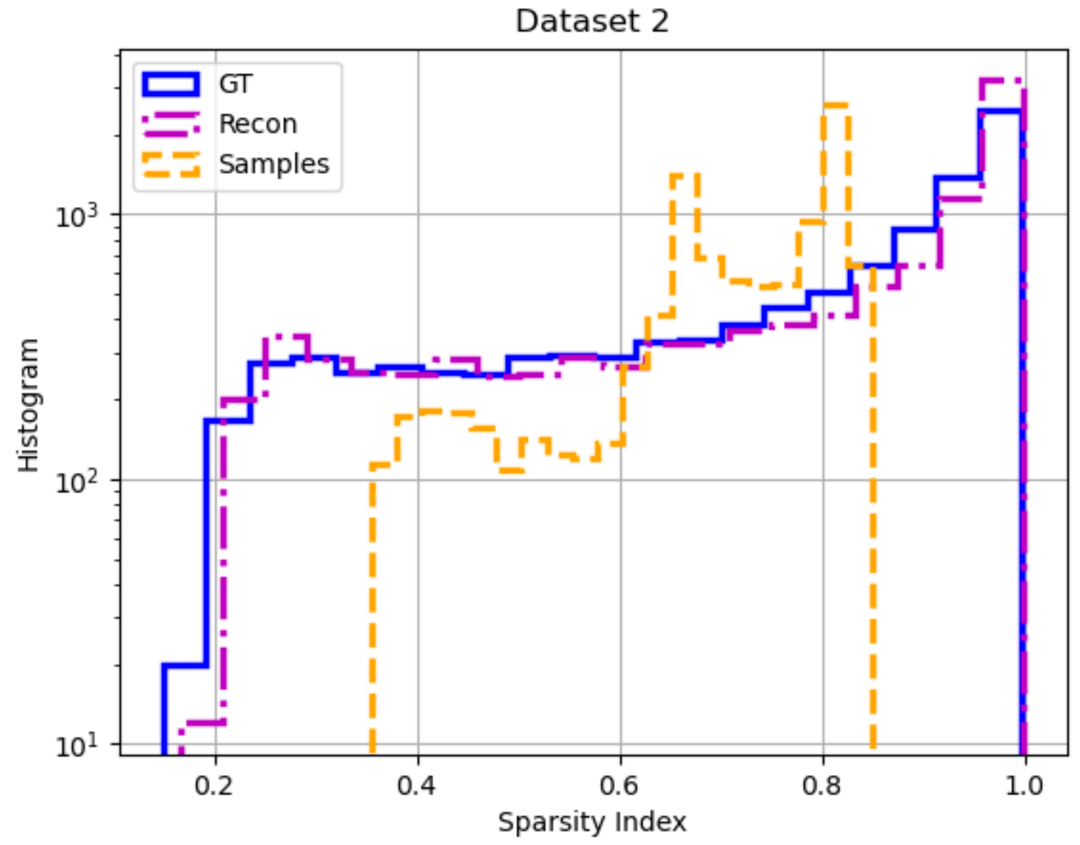
Winter-glade



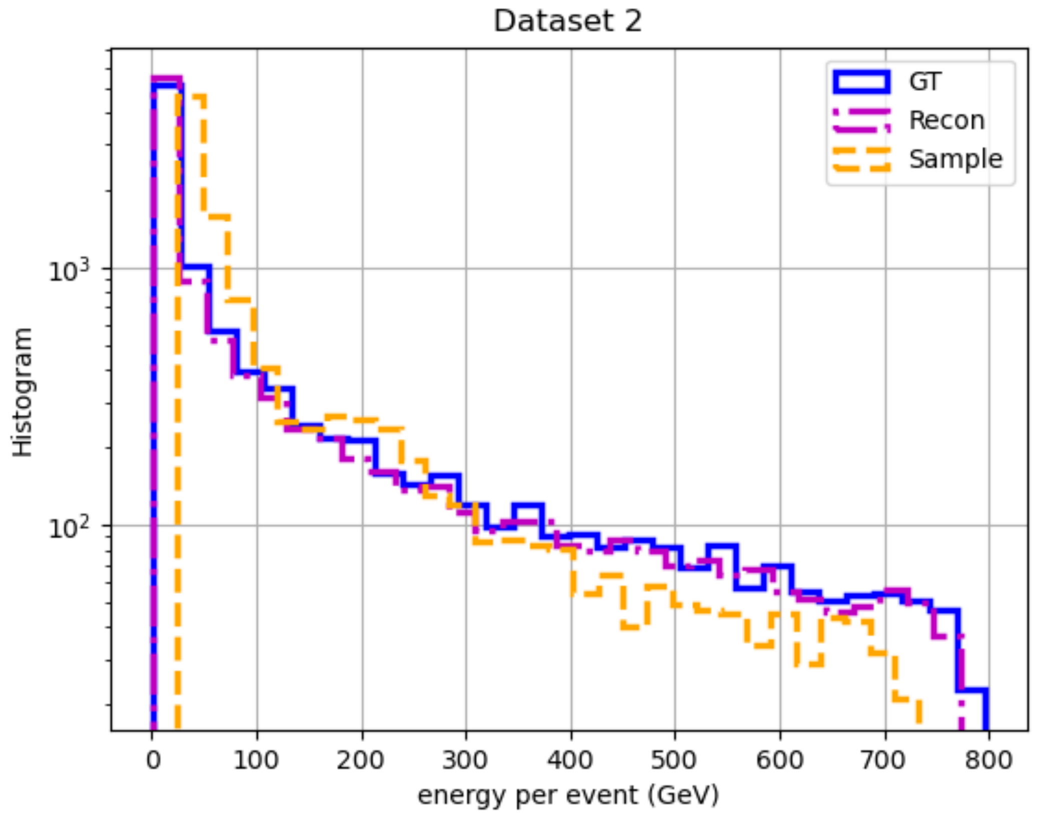


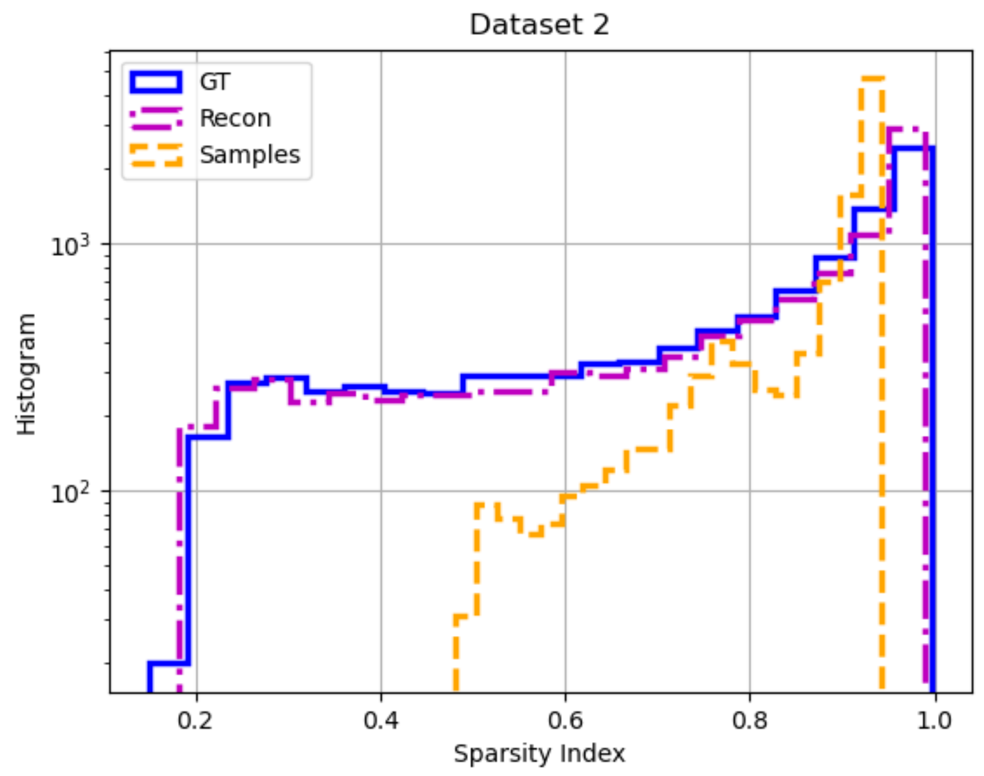
Prime-totem



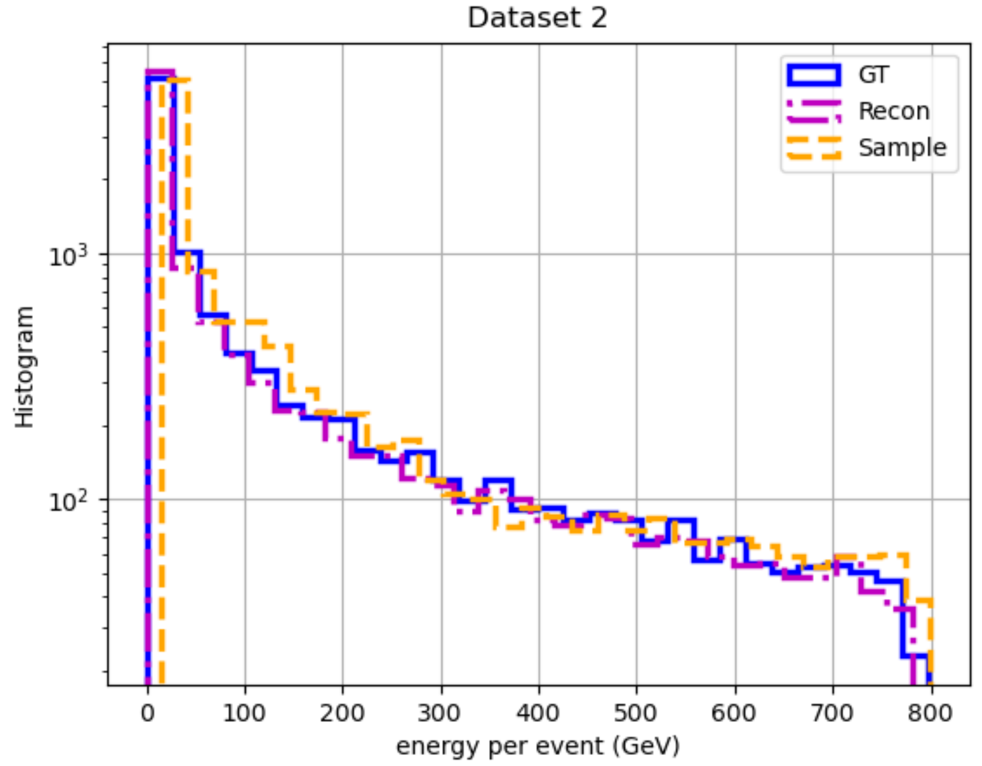


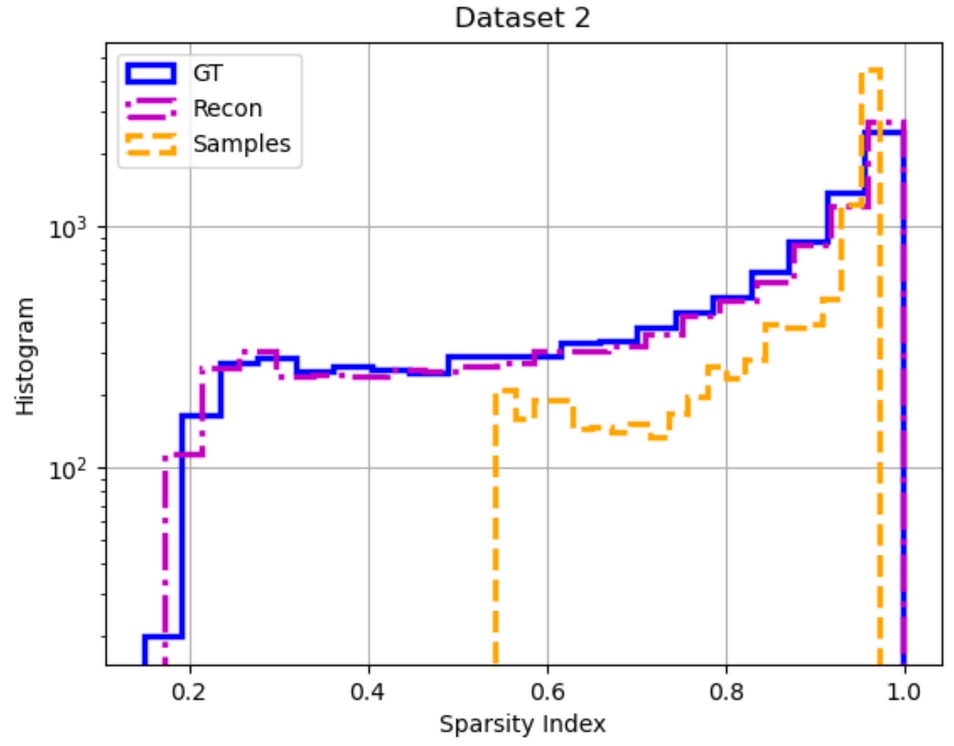
Happy-sun





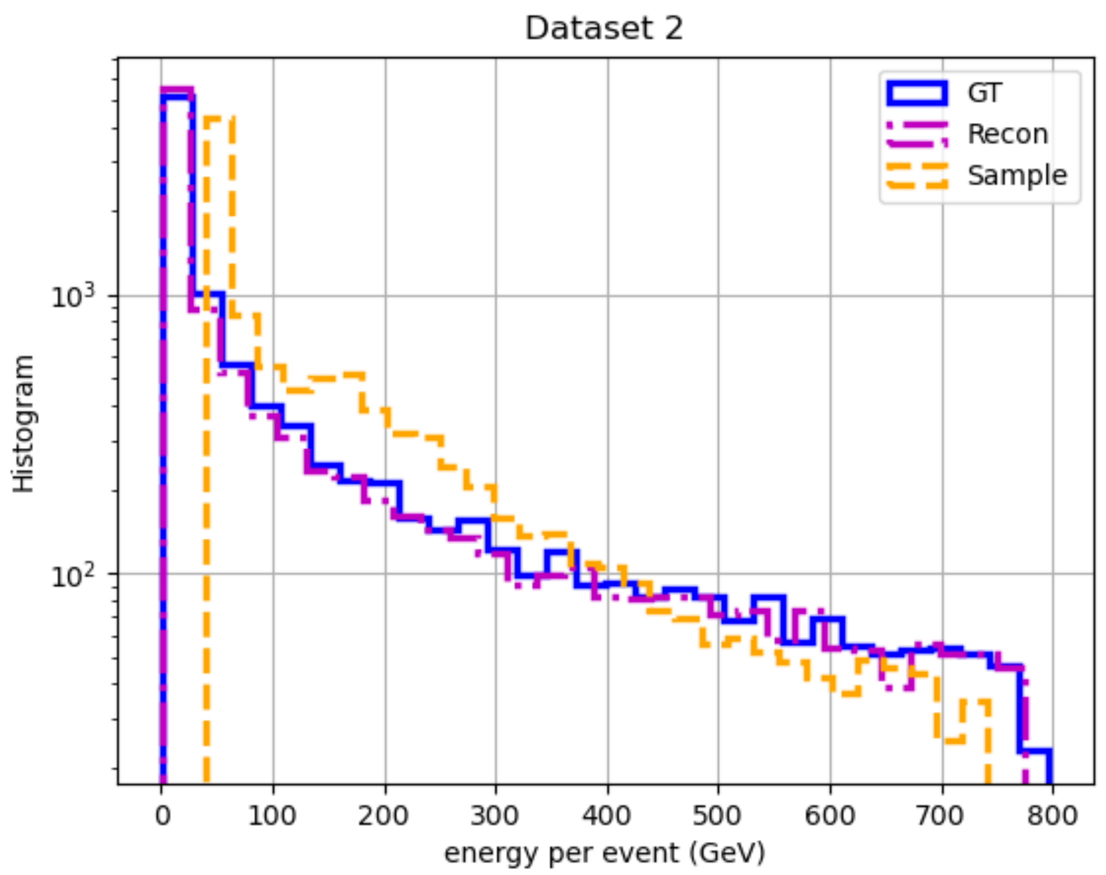
Drawn-cosmos

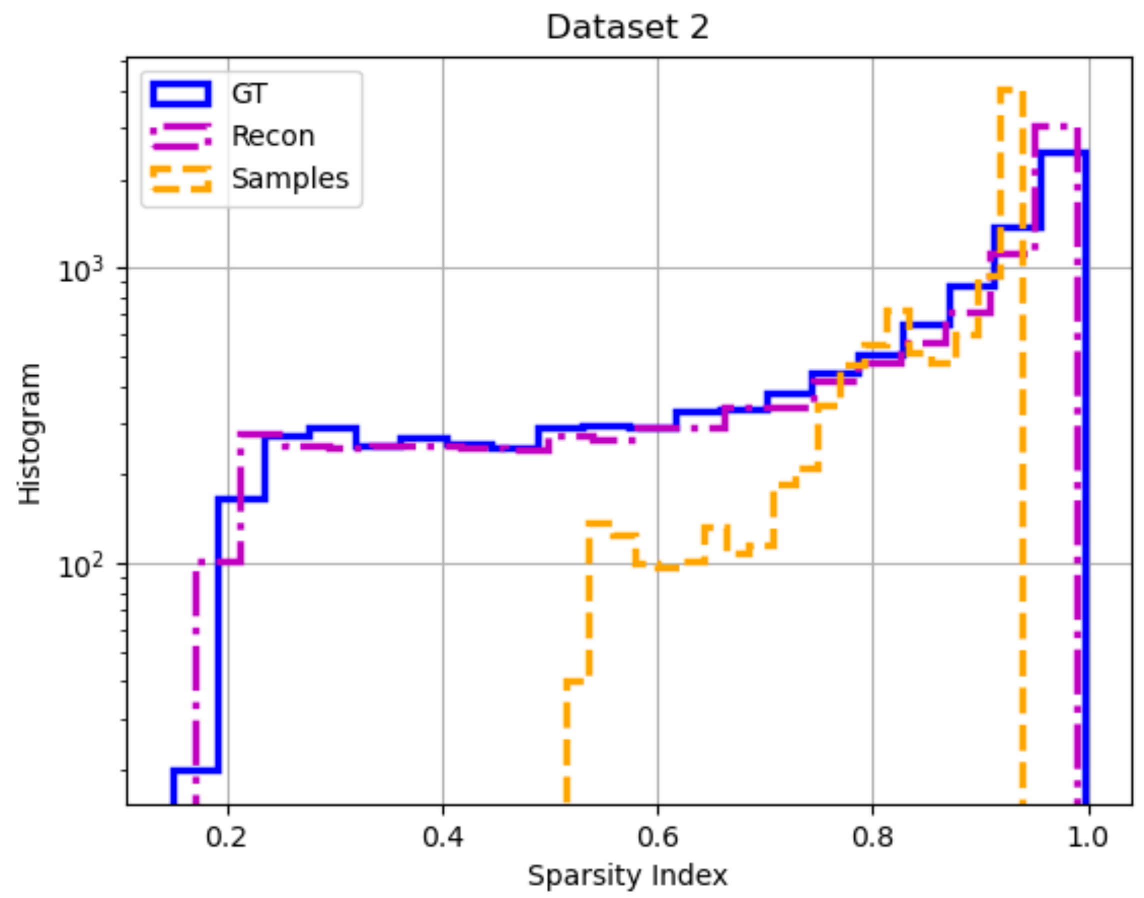




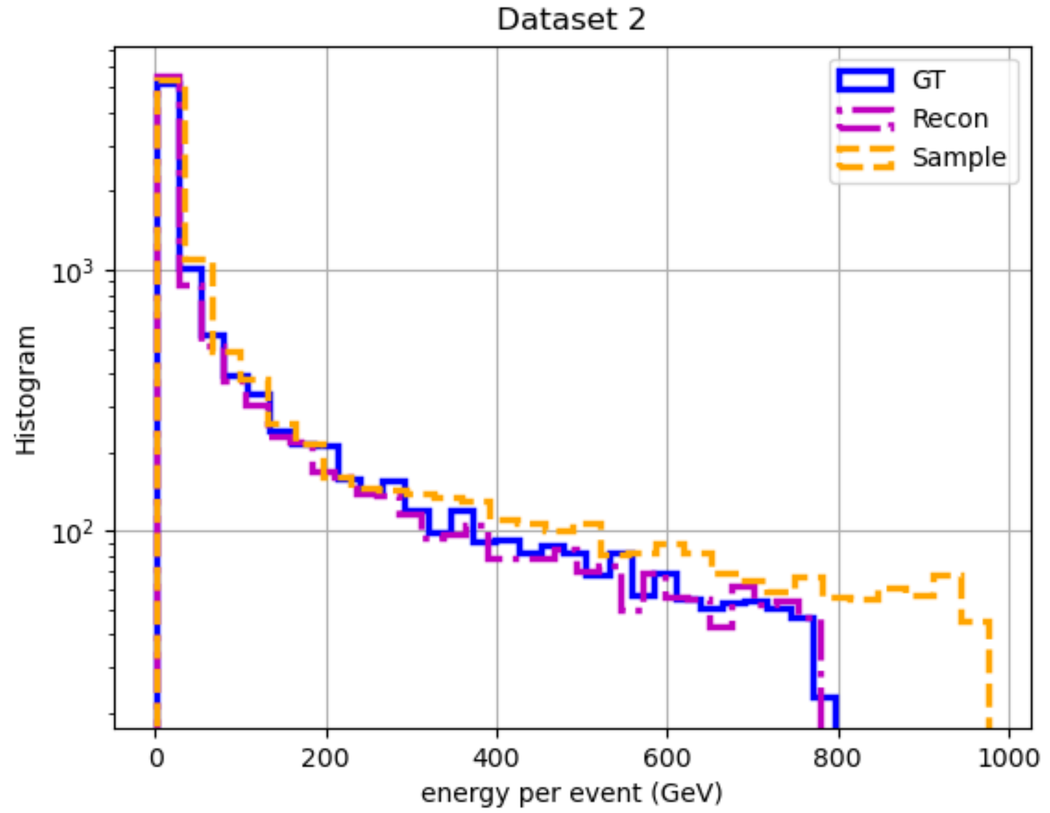
Samples are generated using a non-trained RBM

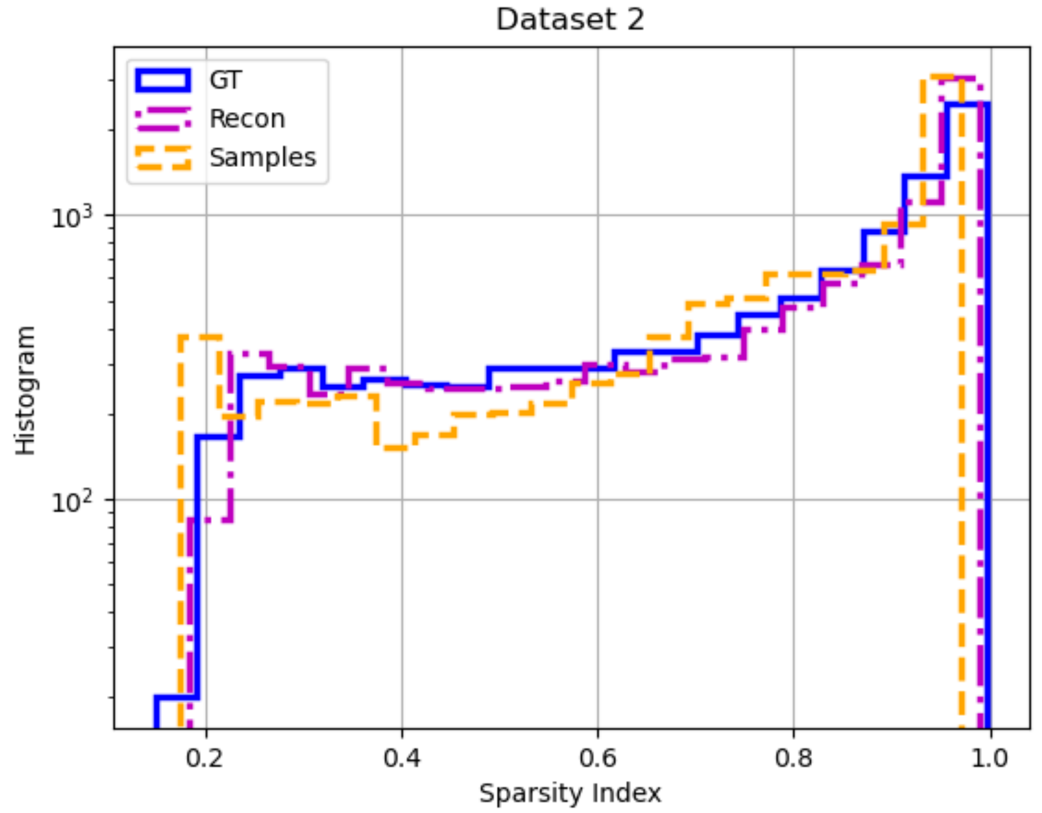
Misty-wind



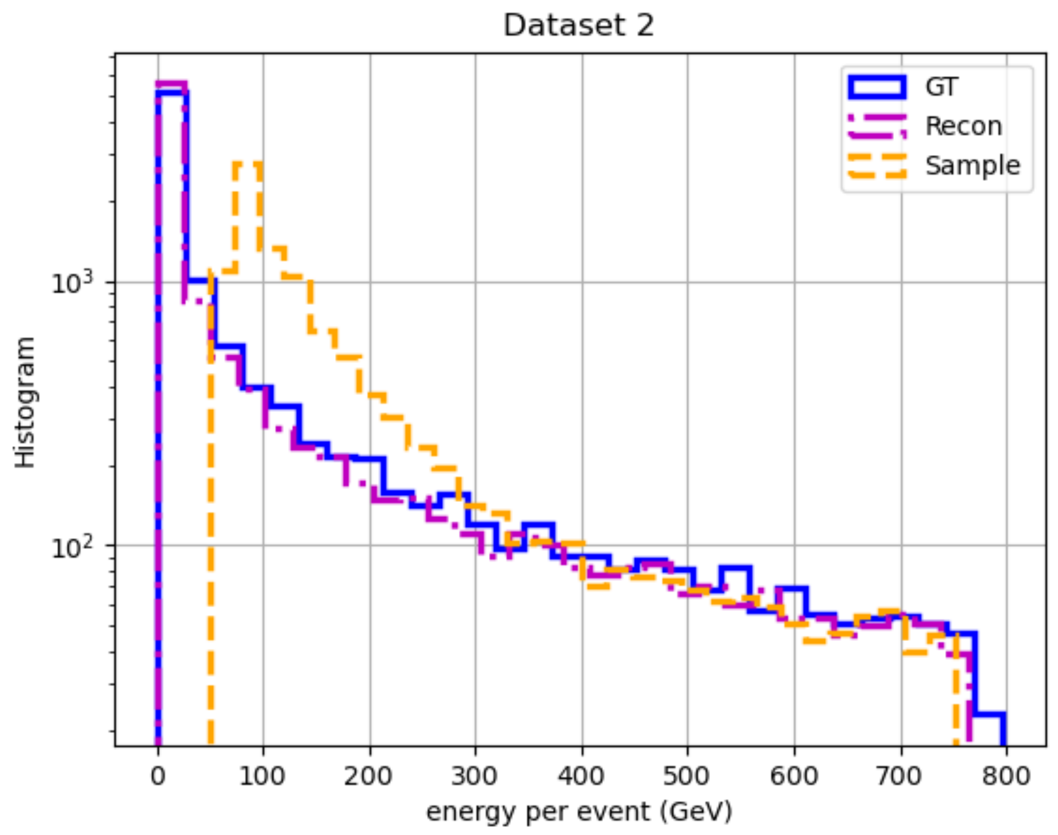


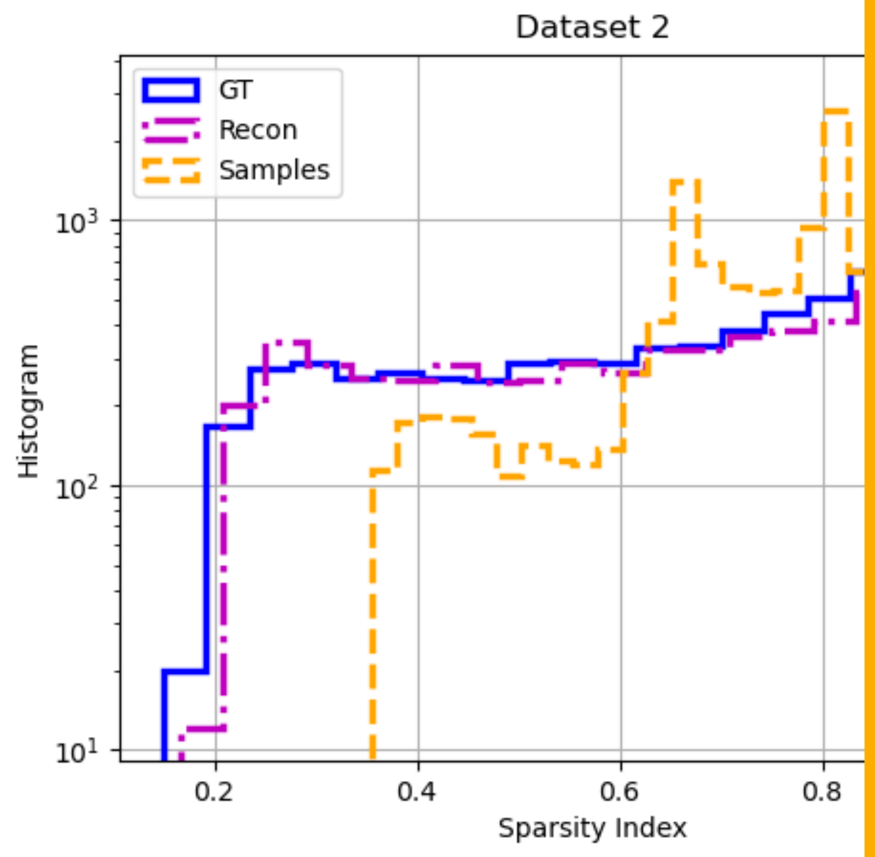
Winter-glade



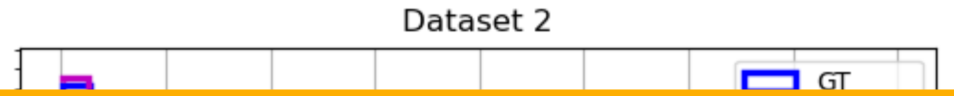


Prime-totem

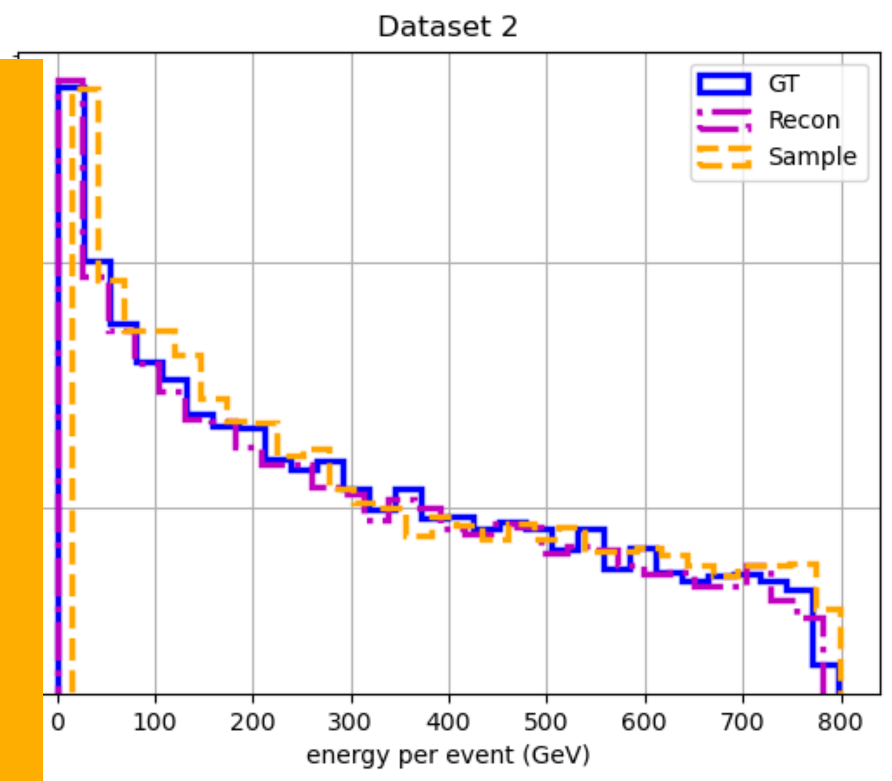


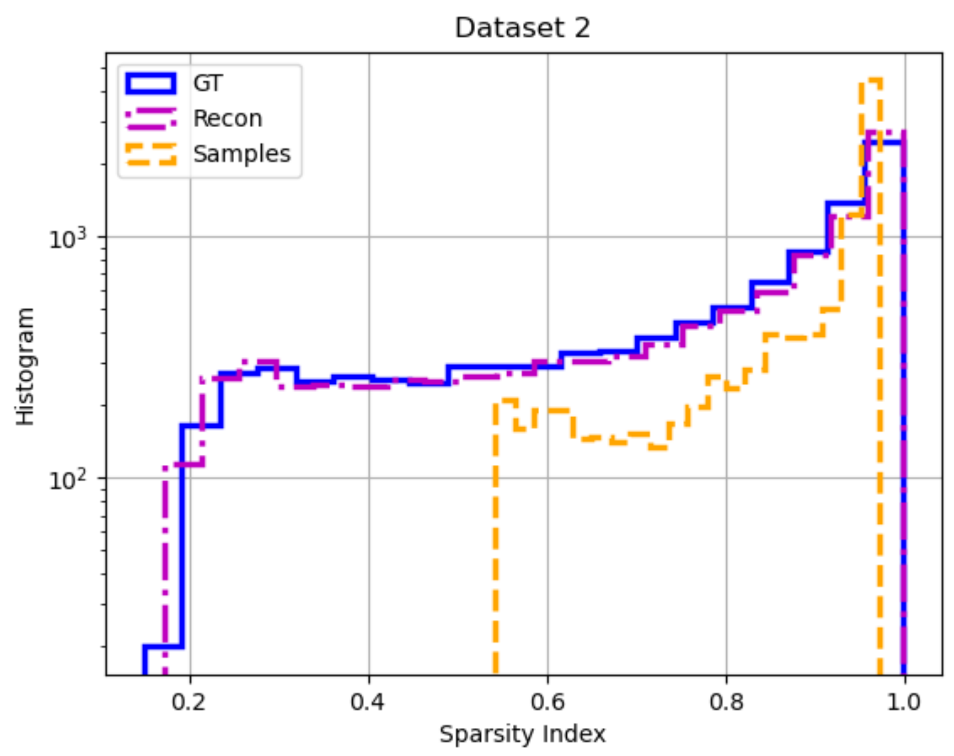


Happy-sun



Drawn-cosmos

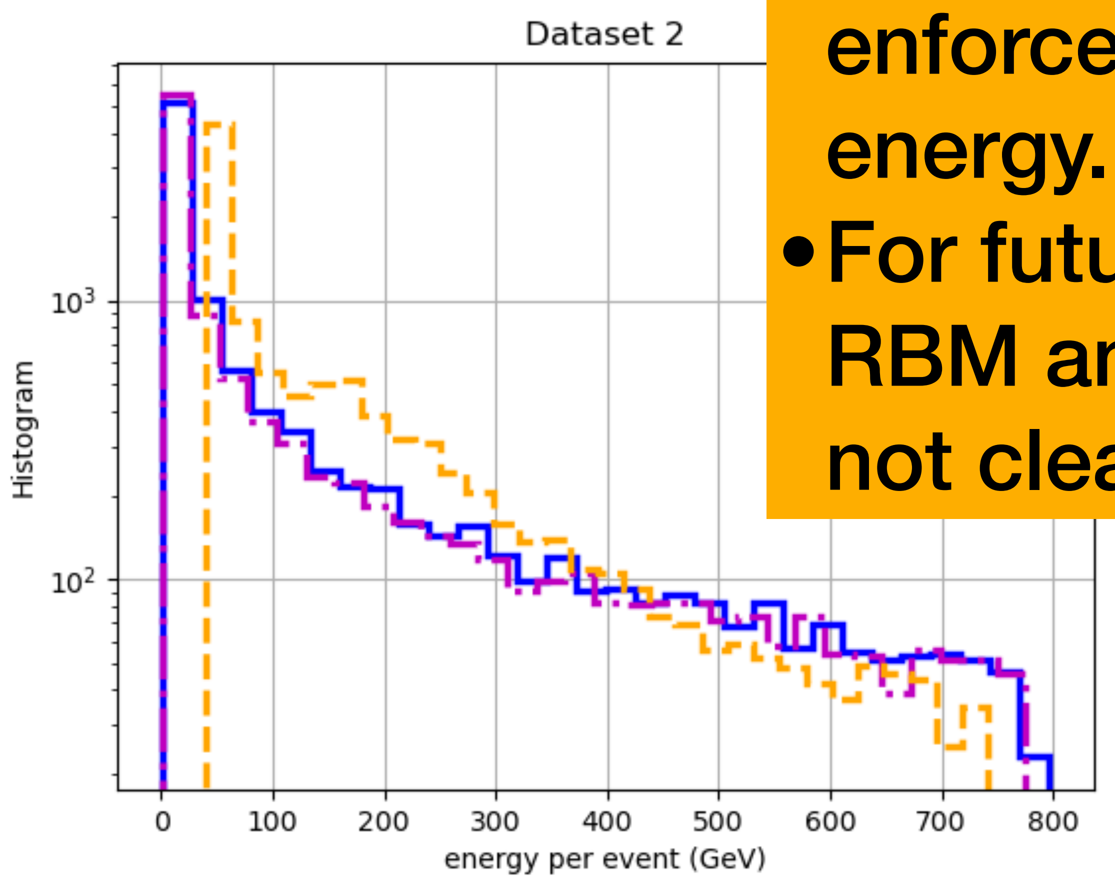


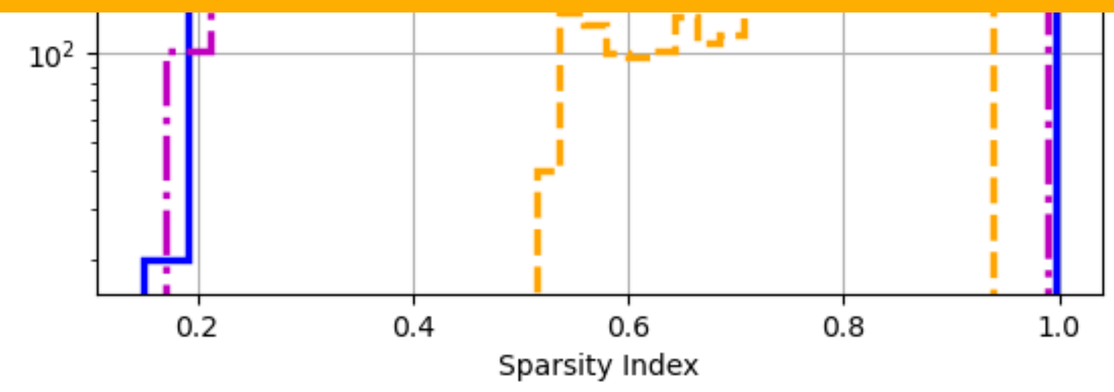


Highlights:

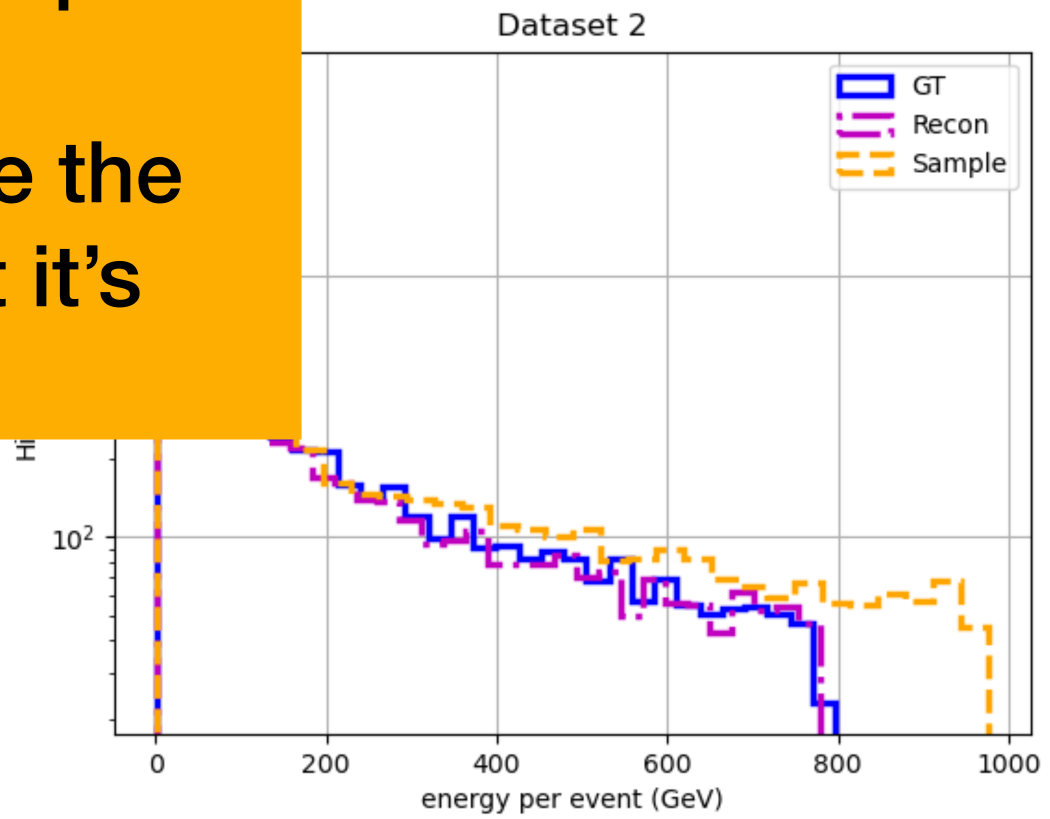
- The decoder is robust enough to cope with the samples of a non-trained RBM.
- Specifically for energy histogram, little does it matter if the RBM is trained.
- This can be understood as follows: In all of these models, the encoder and decoder are conditionalized with the incidence energy. The RBM is never conditionalized. Yet, the energy incidence must be embedded in latent space. Hence, when we sample from the RBM, the sample should correspond to a specific incidence energy (which we don't control). When this generated sample goes through the decoder, we do impose an incidence energy condition. Therefore, the decoder learns to enforce the condition over the *a priori* sampled energy.
- For future work, we need to conditionalize the RBM and, therefore, the QPU as well. But it's not clear whether the latter is feasible.

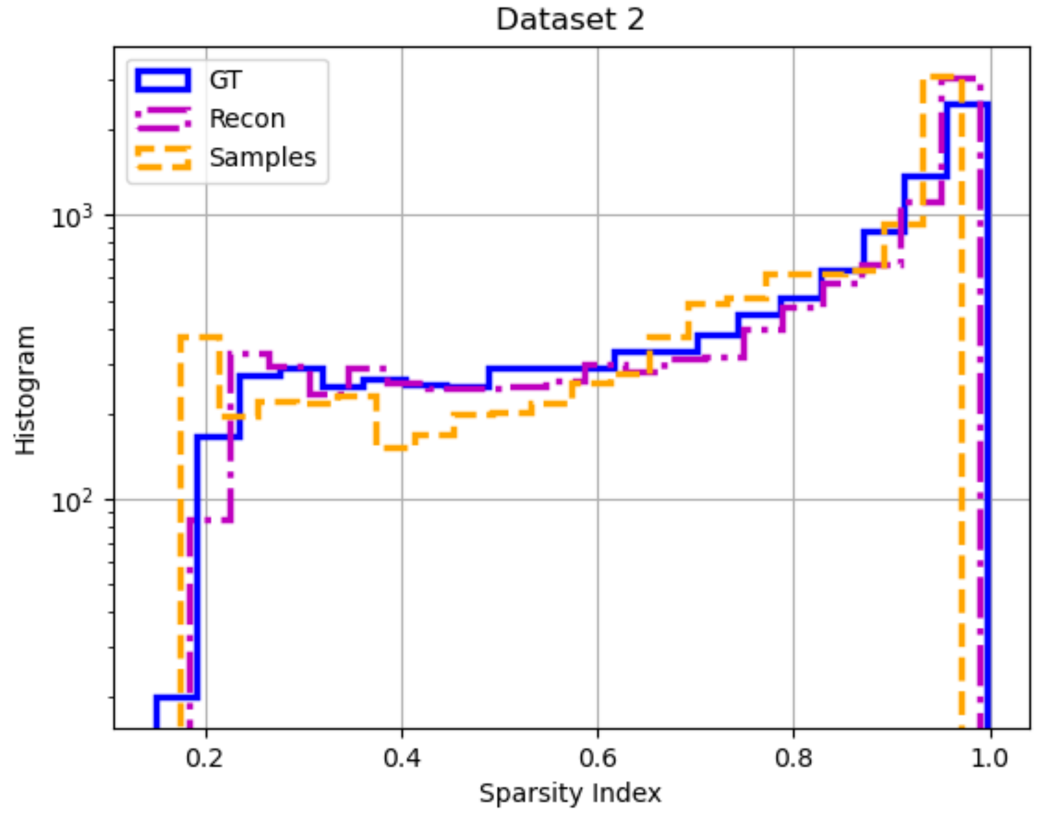
Samples are generated using a non-trained RBM



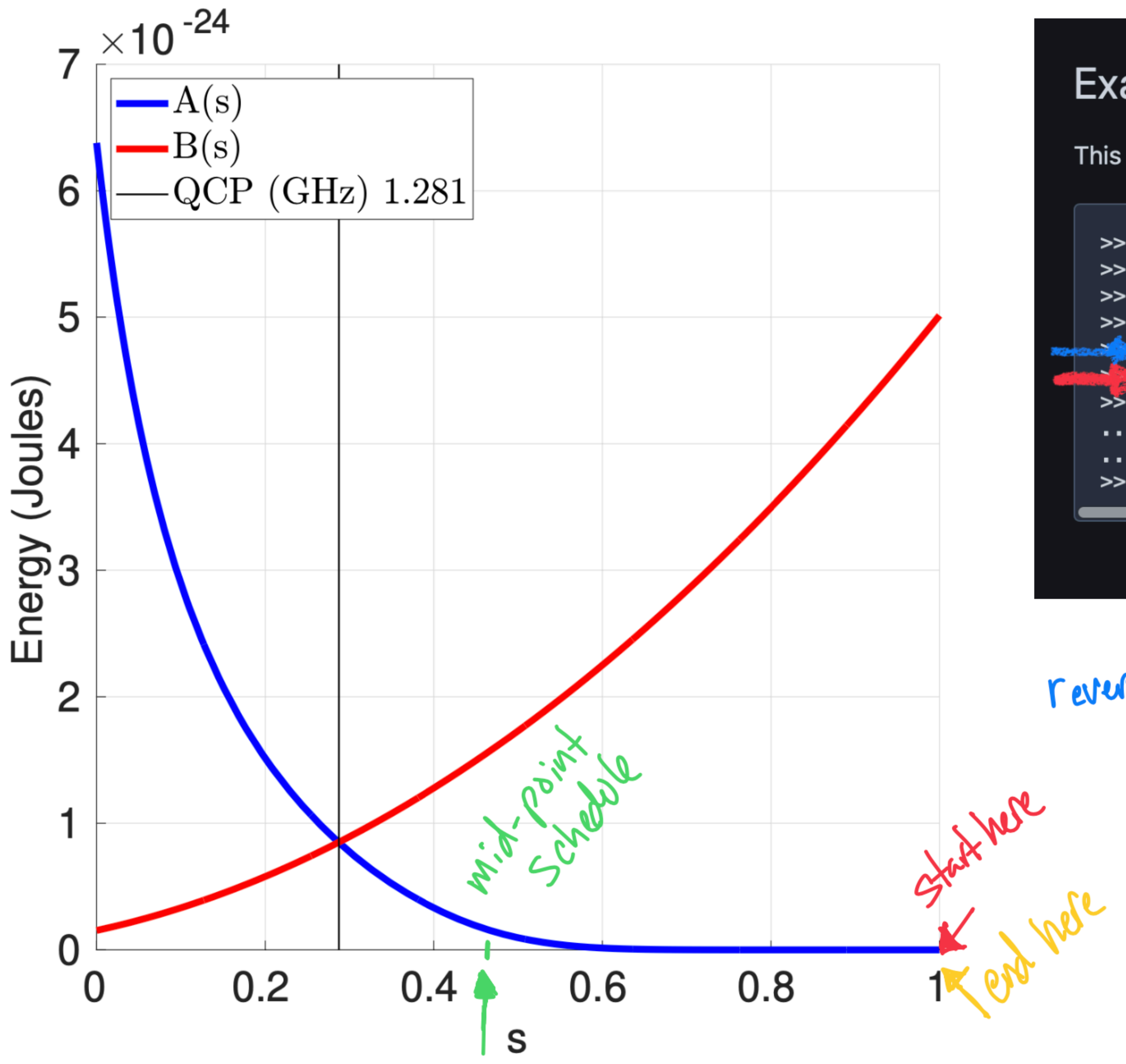


Winter-glade





Encoding energy into RBM.



Example

This illustrative example configures a reverse-anneal schedule on a random native problem.

```
>>> from dwave.system import DWaveSampler
>>> import random
>>> qpu = DWaveSampler()
>>> J = {coupler: random.choice([-1, 1]) for coupler in qpu.edgelist}
>>> initial = {qubit: random.randint(0, 1) for qubit in qpu.nodelist}
>>> reverse_schedule = [[0.0, 1.0], [5, 0.45], [99, 0.45], [100, 1.0]]
>>> reverse_anneal_params = dict(anneal_schedule=reverse_schedule,
...                              initial_state=initial,
...                              reinitialize_state=True)
>>> sampleset = qpu.sample_ising({}, J, num_reads=1000, **reverse_anneal_params)
```

reverse_schedule = [[t_0, s_0], ..., [t_s, s_s], ..., [t_n, s_n]]

$$\mathcal{H}_{ising} = \underbrace{\frac{A(s)}{2}\left(\sum_i \hat{\sigma}_x^{(i)}\right)}_{\text{Initial Hamiltonian}} + \underbrace{\frac{B(s)}{2}\left(\sum_i h_i \hat{\sigma}_z^{(i)} + \sum_{i>j} J_{i,j}\hat{\sigma}_z^{(i)}\hat{\sigma}_z^{(j)}\right)}_{\text{Final Hamiltonian}}$$

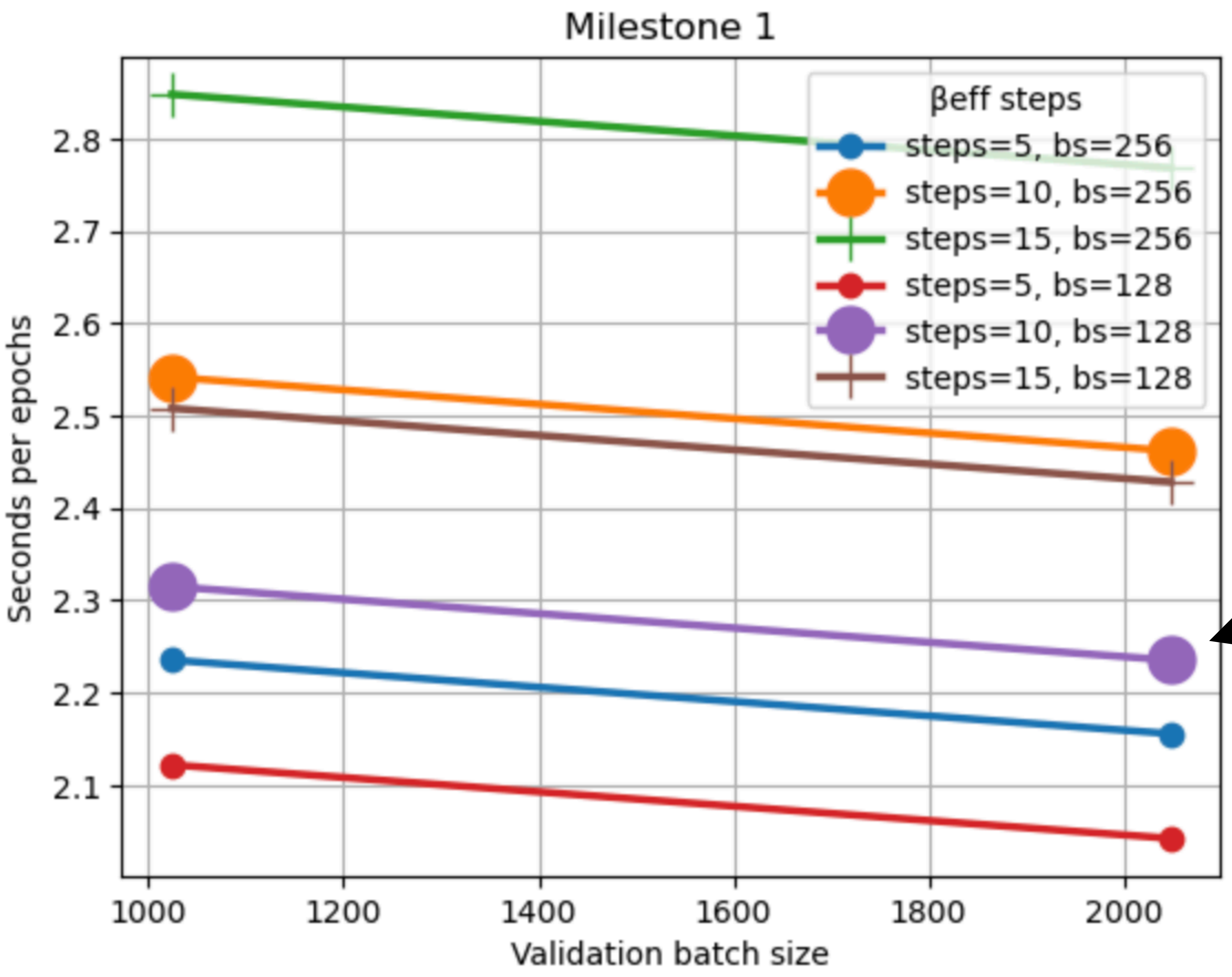
In prior meetings we discussed a possible way to conditionalize the RBM by reverse annealing, i.e.,

- Initialize the QPU with a specific quantum state of \sigma_z and annealing time variable s=1
- Reverse the annealing to s<0.4.
- Anneal back to s=1

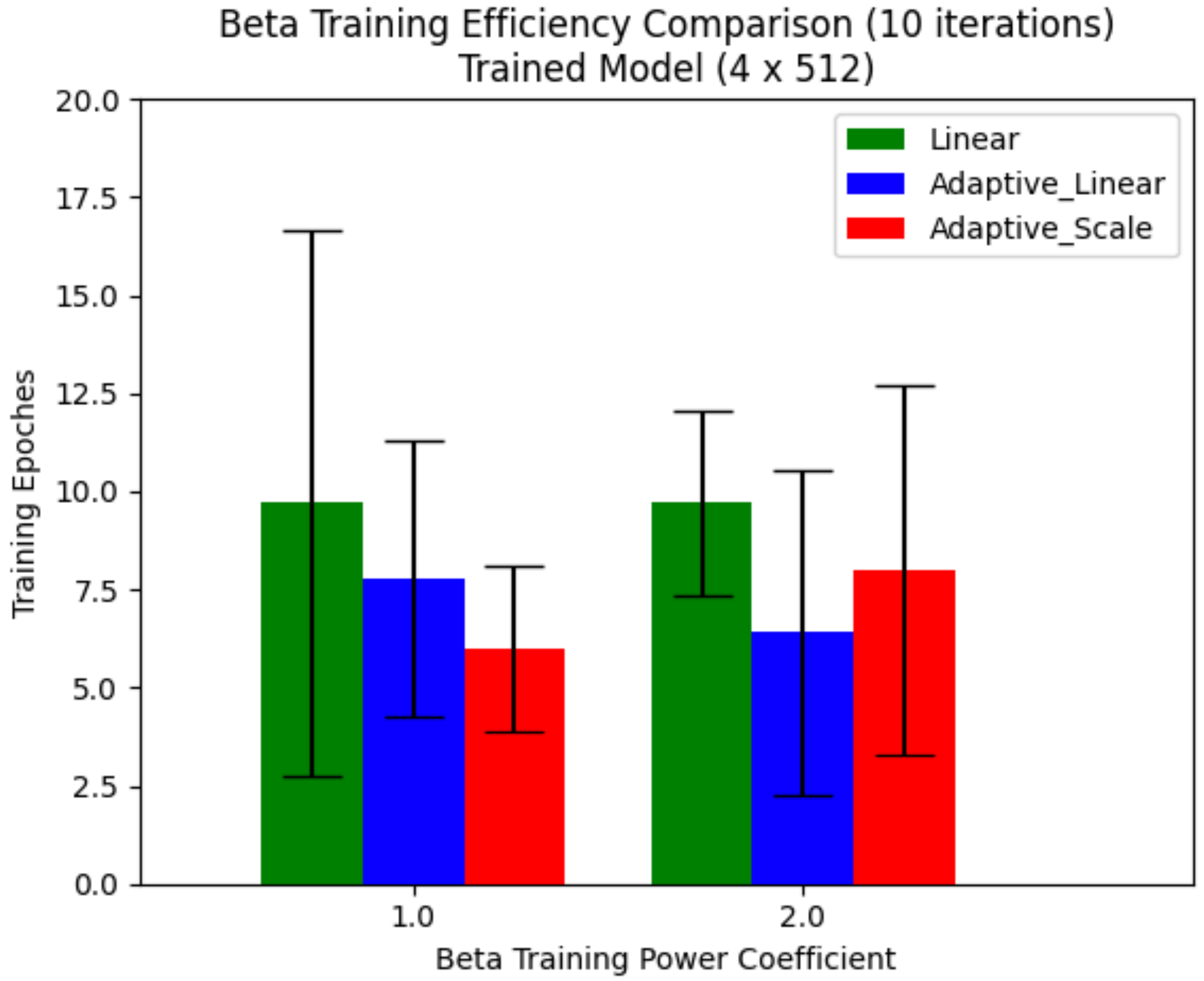
However, in order for this to work, i.e., to not destroy the qubits which encode the incidence energy label, we need to make sure that the term in the dwave Hamiltonian which contain the \sigma_x^j are such that the coefficients corresponding to the qubits \sigma_z^j that encoded the incidence energy are zero.

In this approach, I think, the annealing process will be such that qubit which contain the energy incidence encoding in the \sigma_z basis, will not get projected onto the \sigma_x basis and, hence, ought to not get destroyed.

We need to figure out if and how to specify these coefficients



Train model for 25 epochs using free dwave trial



- PRX paper highlights
  - Architectures
    - CNN
    - FCN
    - 4-partite RBM
  - Energy incidence
    - Condition on encoder and decoder via concat or positional encoding
  - Results/metrics
    - Energy histogram
    - Sparsity histogram
    - Mean energy per r,theta,z
    - Energy distribution for encoded and RBM Gibbs samples
    - Zais and Zrais estimates for partition function => log-likelihood of model
  - Dwave QPU for sampling and validation
  - Method to estimate temperature
    - Sehmi’s method
    - Hao’s method/ adaptive method