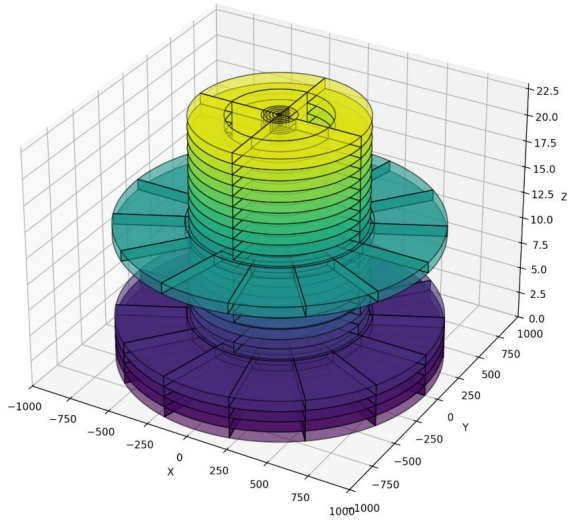
Weekly Update

August 22, 2025

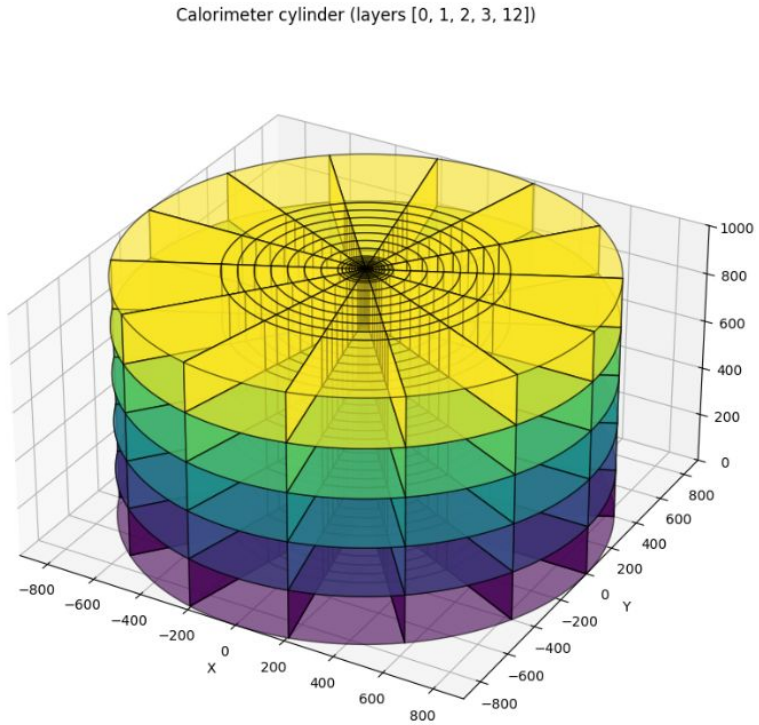
Leo Zhu and Denaisha Kraft

Dataset Binning

Regular binning (all layers):

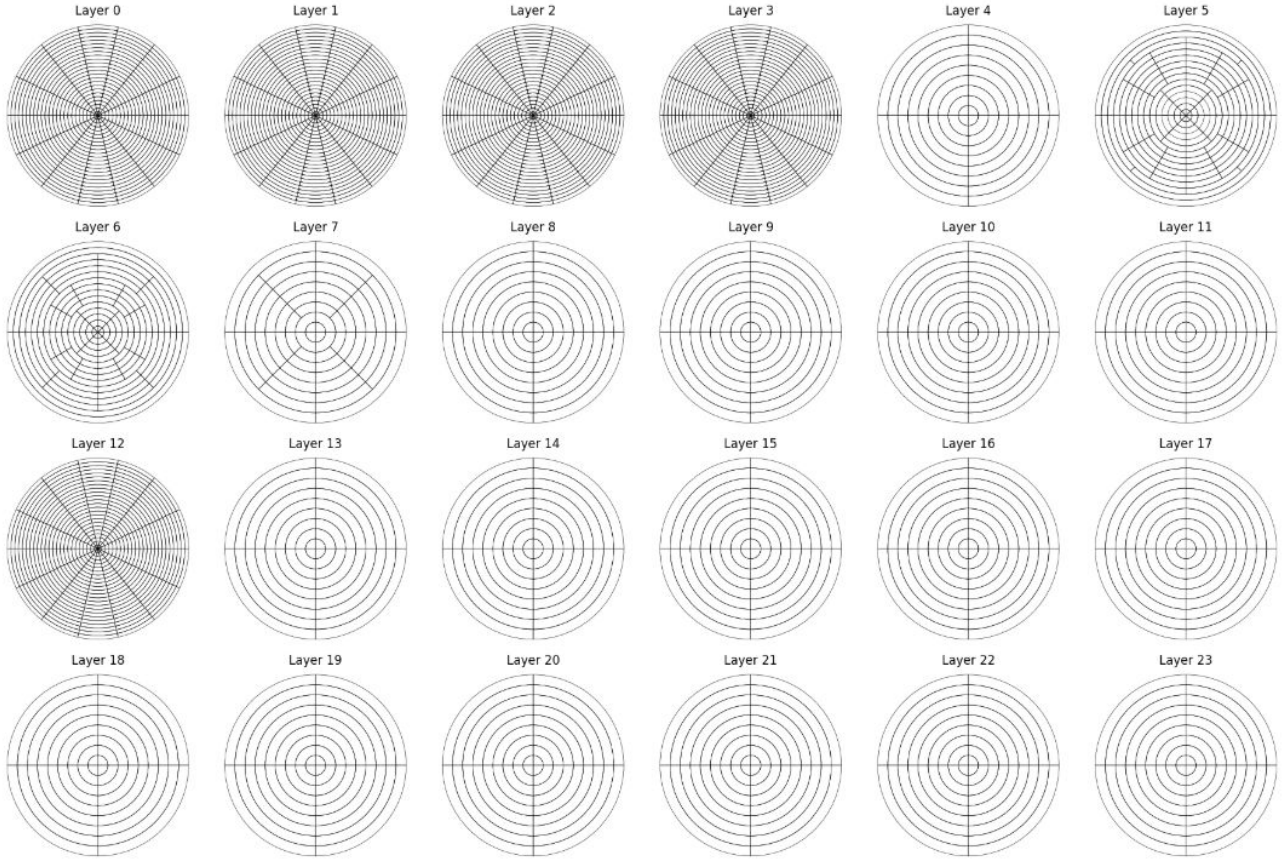


- The 5 active layers in the main dataset (150,000 events) all have the same geometry
- Inactive layers are smaller



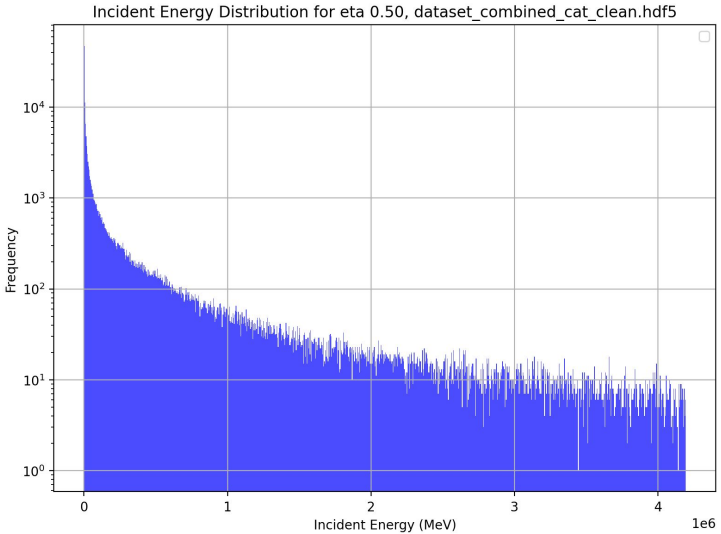
Layer Geometry

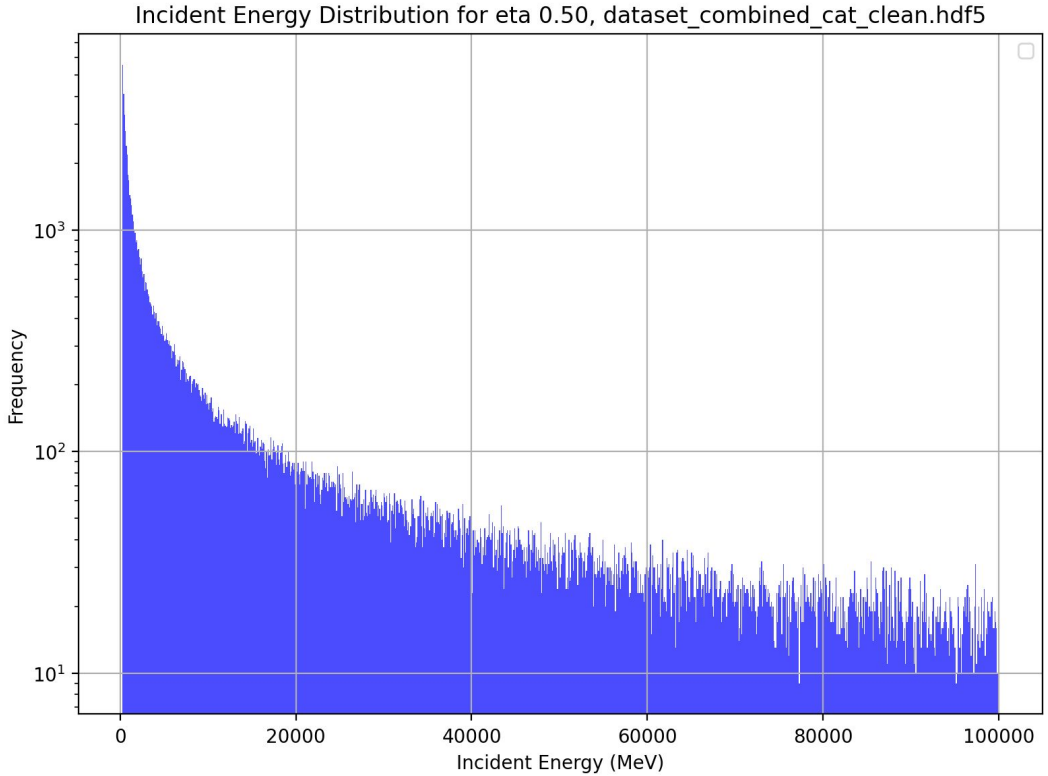
- Layer 0,1,2,3,and 12 are the same
- Layer 5 and 6 are the same
- Layer 7 is different from all others
- Using ‘equal_bin’



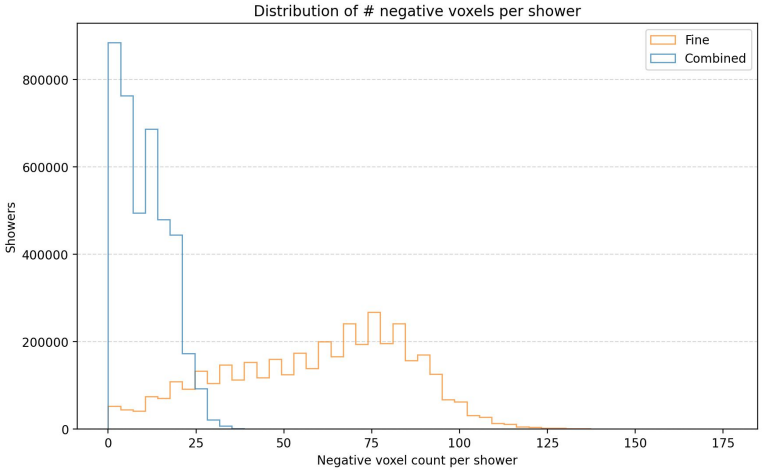
Incidence Energy Distribution (Revisited)

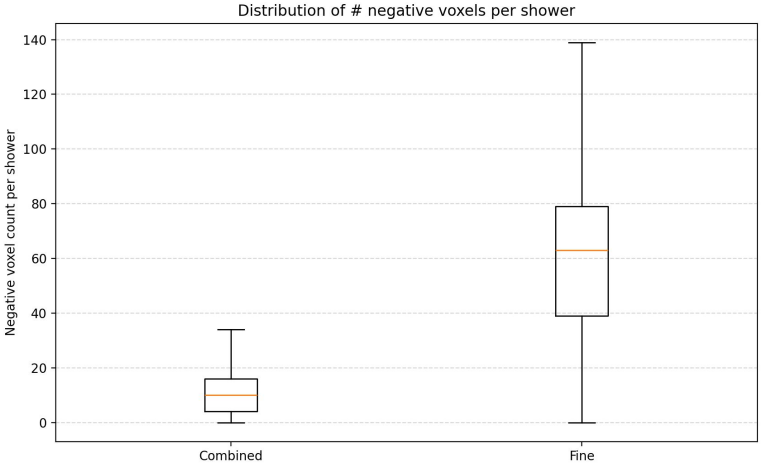
- Sliced off at 100 GeV
- Many events at low energies

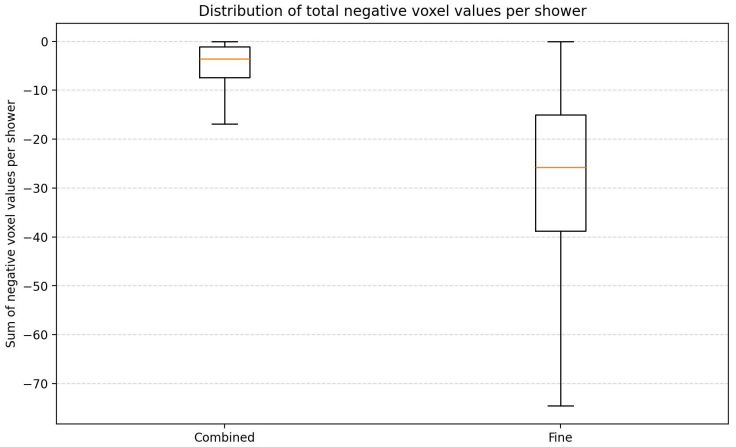




Negative Voxels

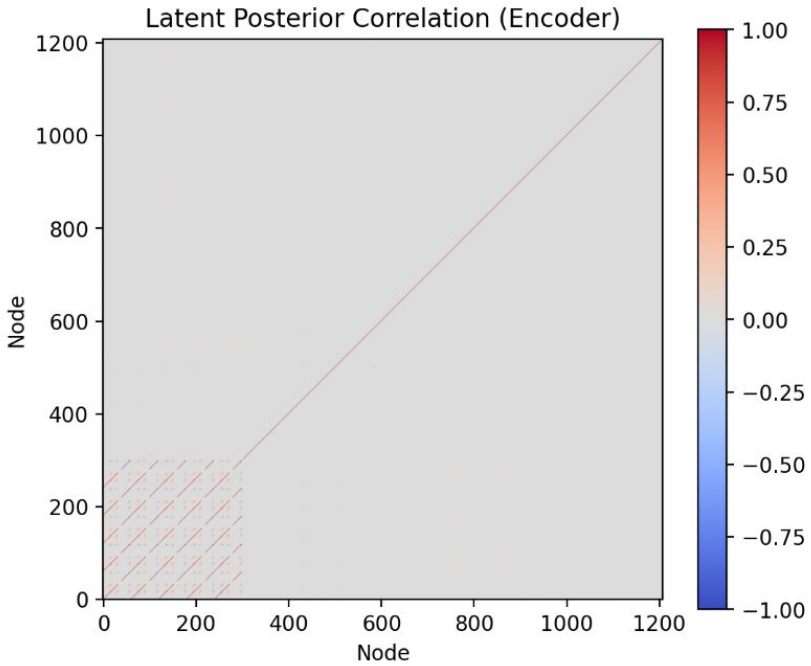


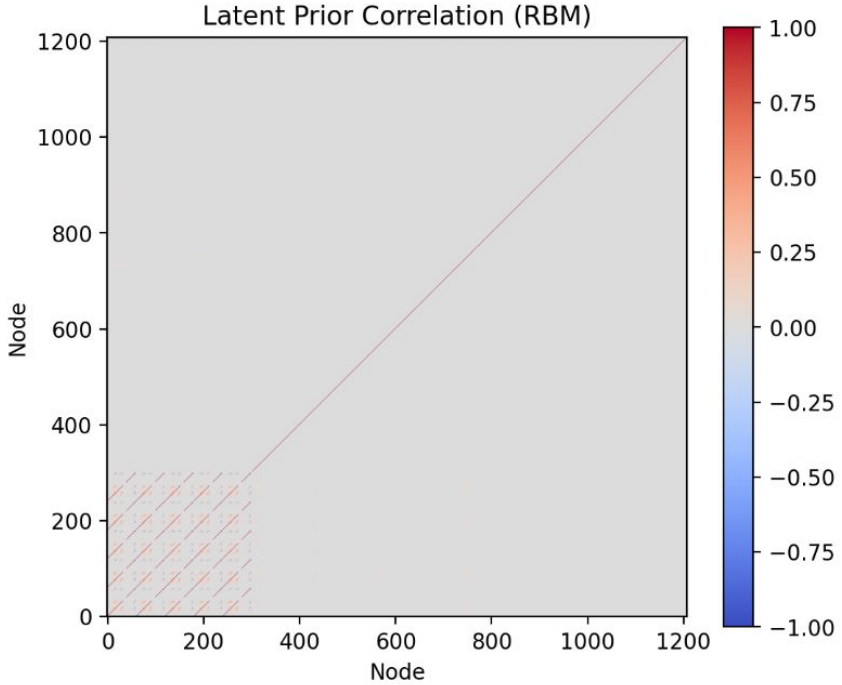




Comparing Nodes Across Partitions

- 4 partitions, 302 nodes per partition
- Looking at correlation between encoder output and RBM
- No cross-partition correlation outside of the first partition
- Compute and save Frobenius metric

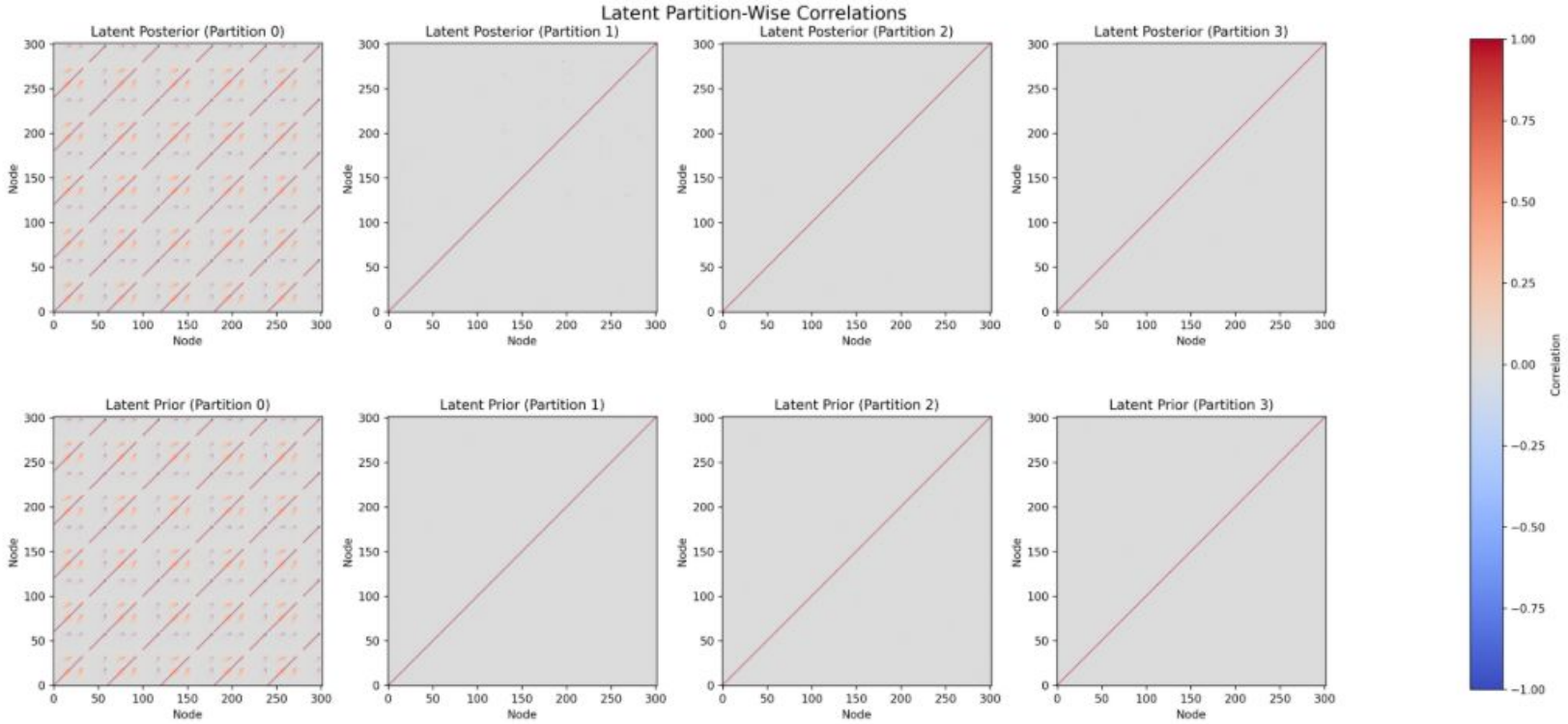




Partition-Wise Comparisons

Similar to previous slide but now isolated per partition

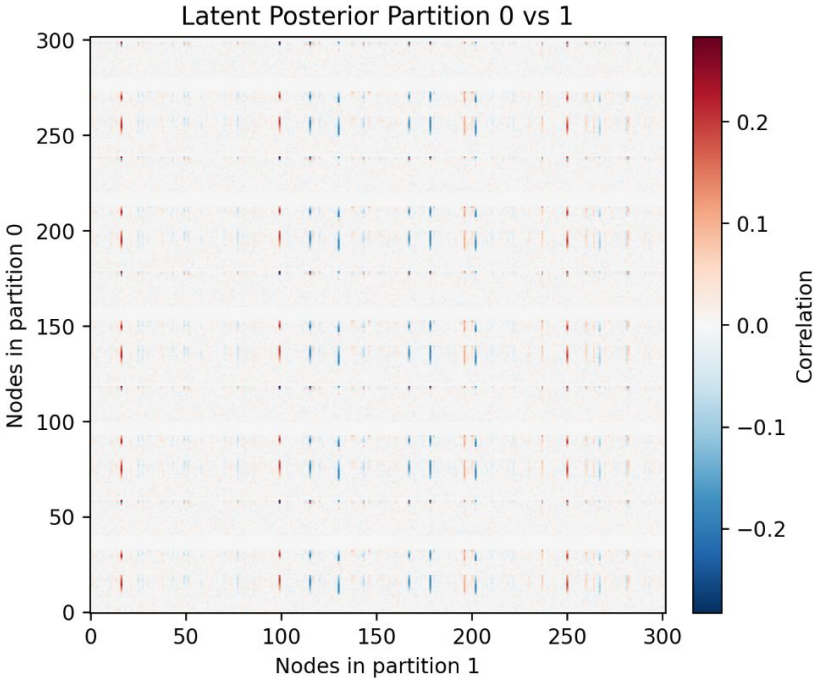
- Compute and save 4 Frobenius metrics (4 partitions)

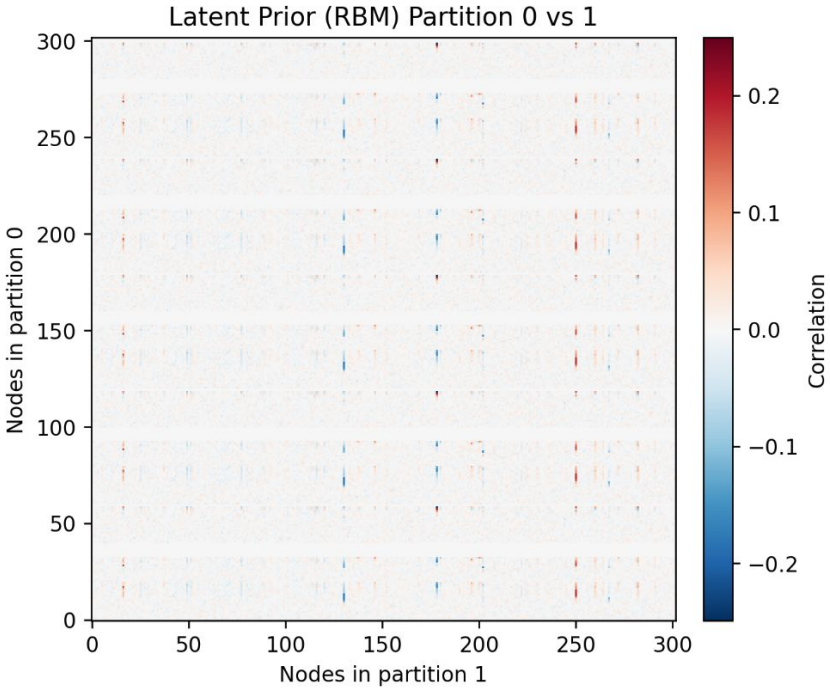


Comparing Partitions

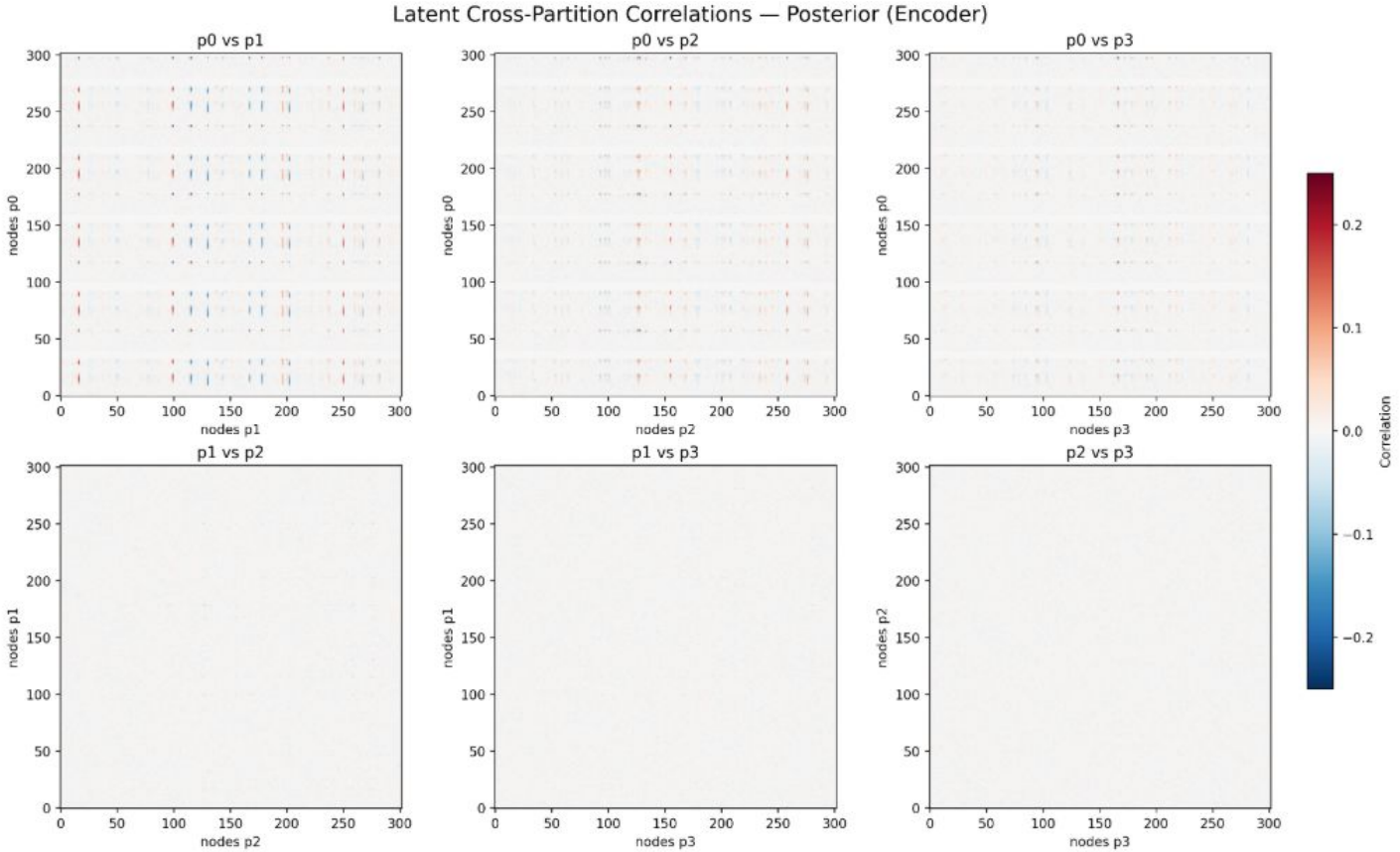
Now looking at the correlation between nodes in different partitions

- Computed for each pair of different partitions (save Frobenius metric)



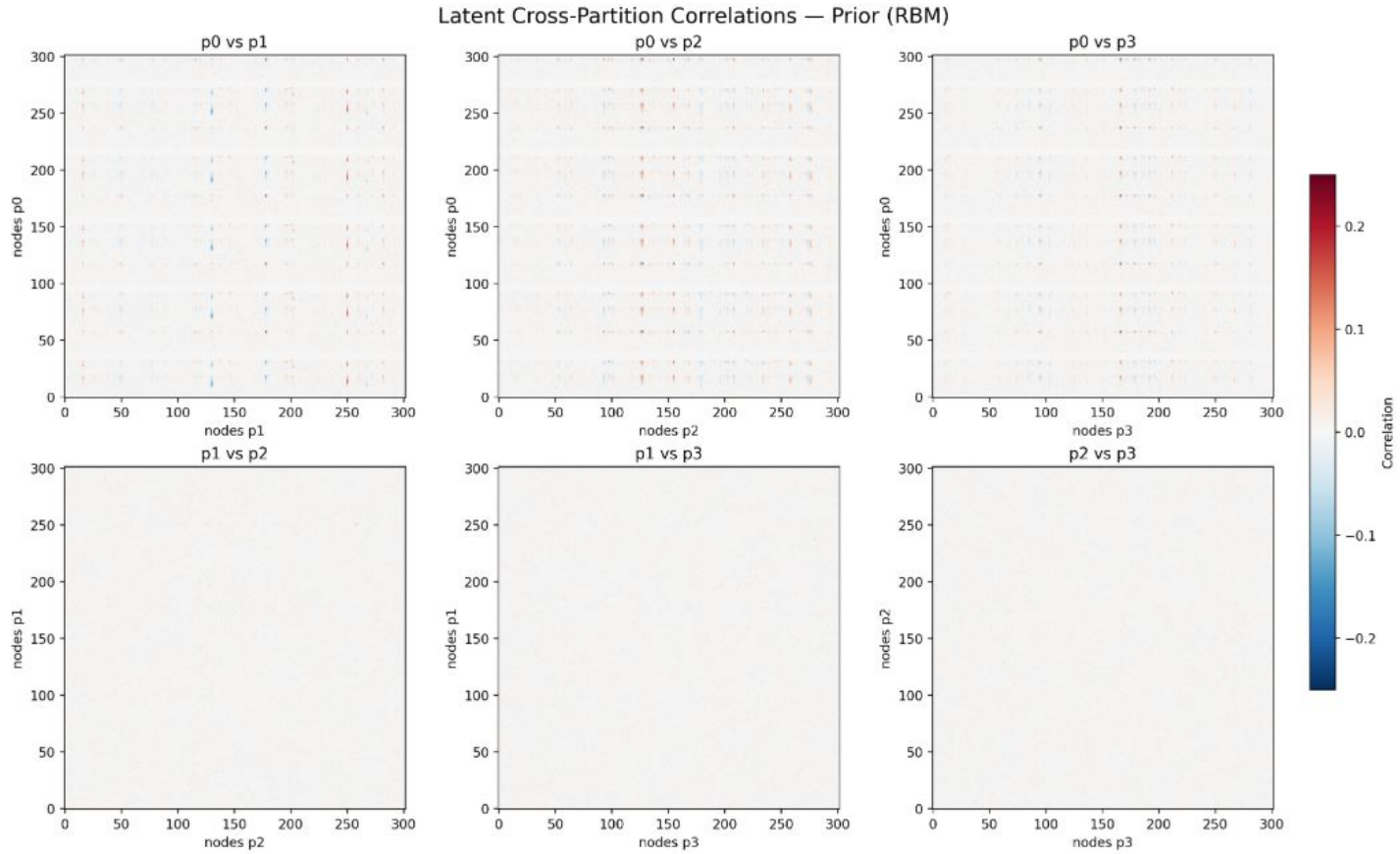


Other Combinations (Encoder)



No correlation pattern when the first partition is not involved

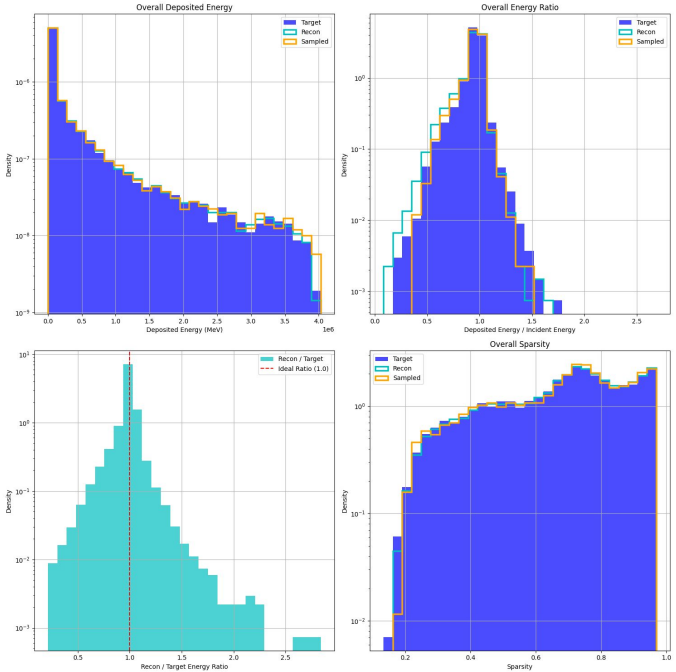
Other Combinations (RBM)

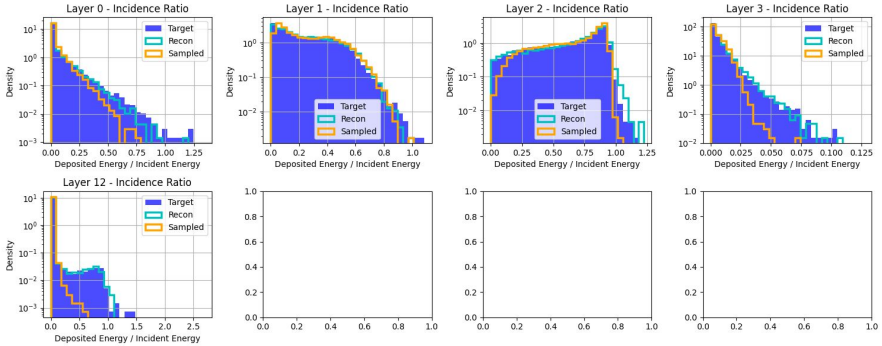


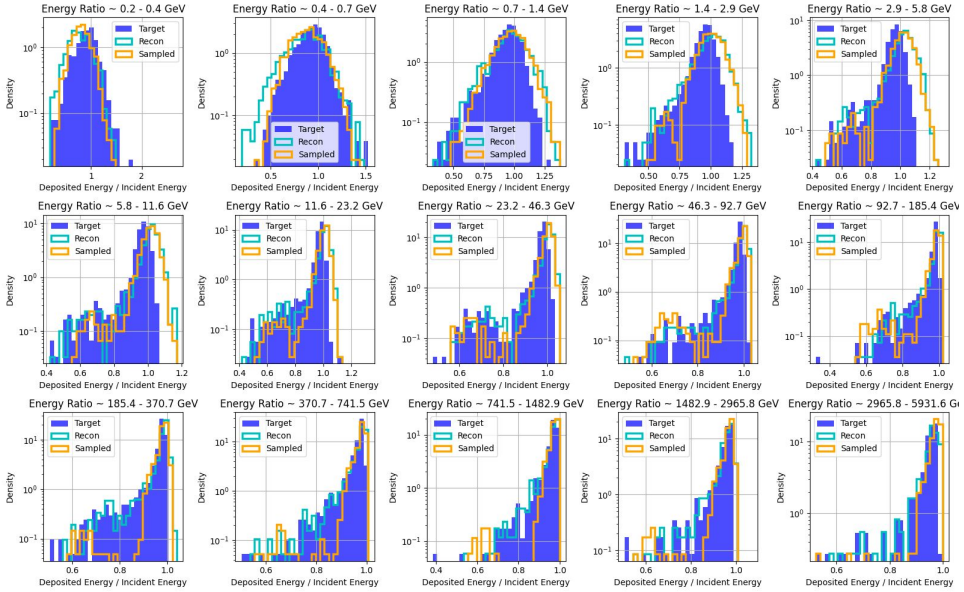
Again no correlation when the first partition is not involved

Model trained on ATLAS New

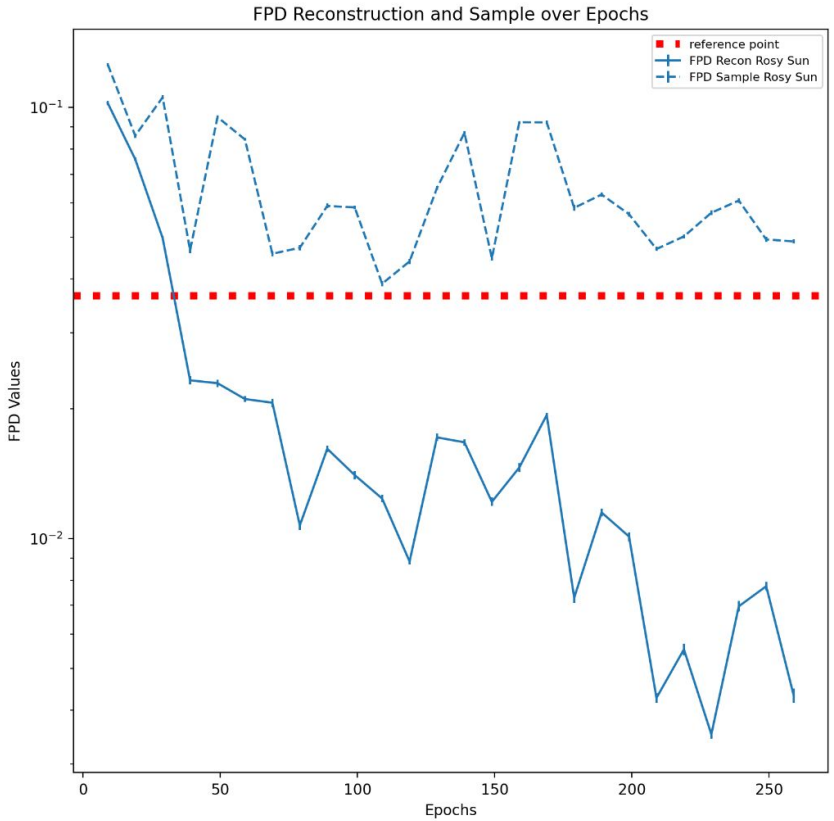
- Epoch 260, training in VAE mode

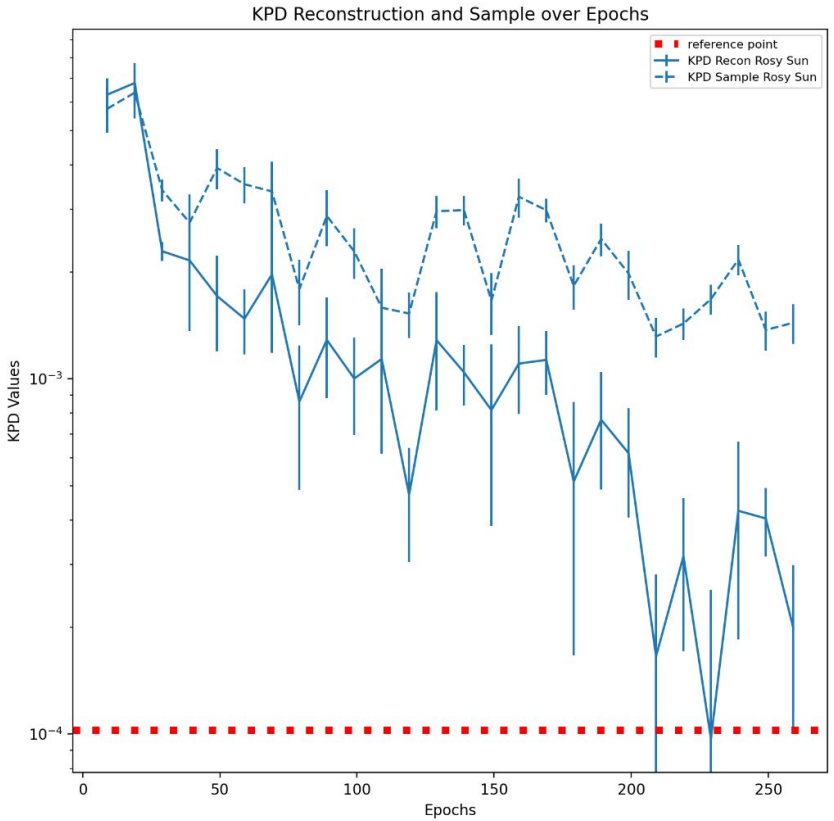






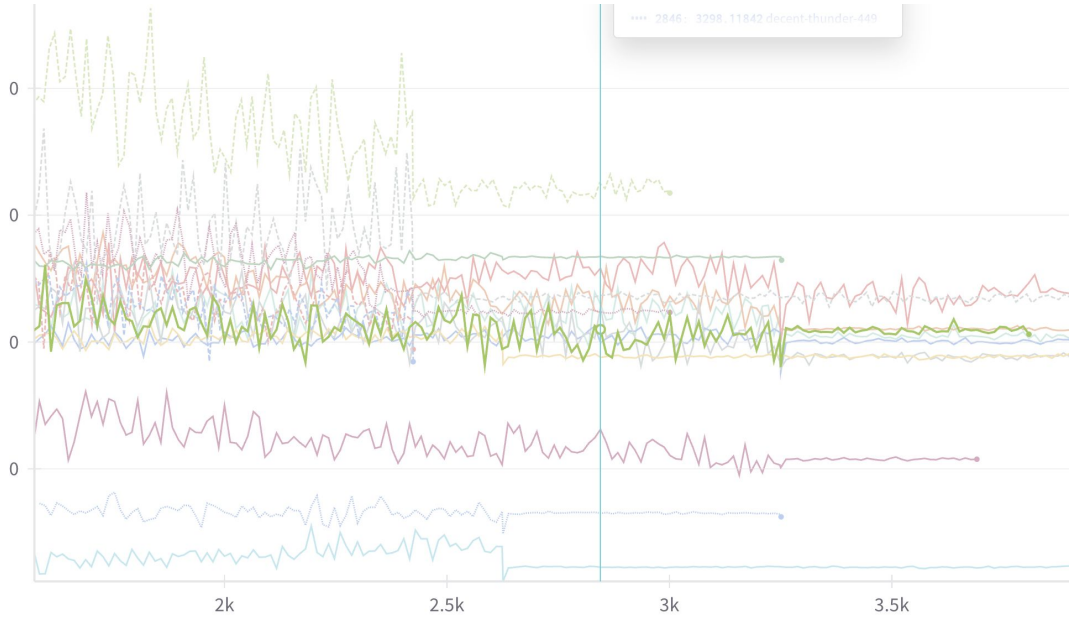
KPD and FPD for ATLAS New





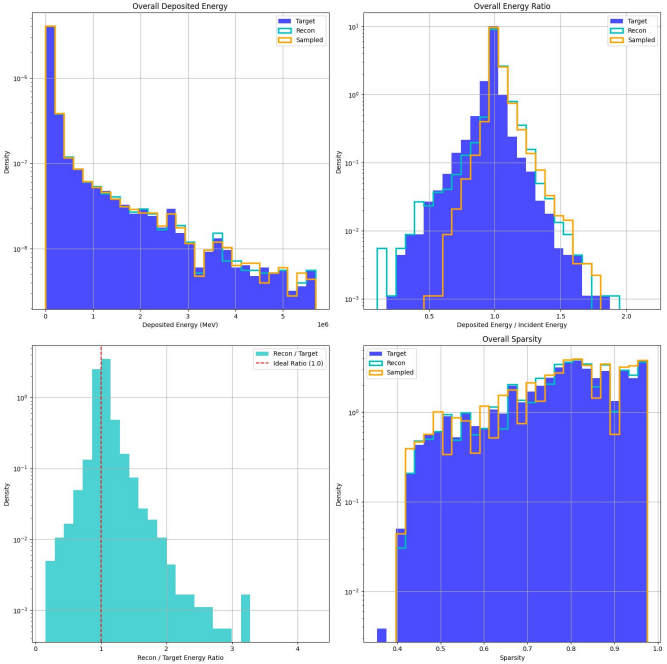
Updates on Decoder: Two Heads are Better than One

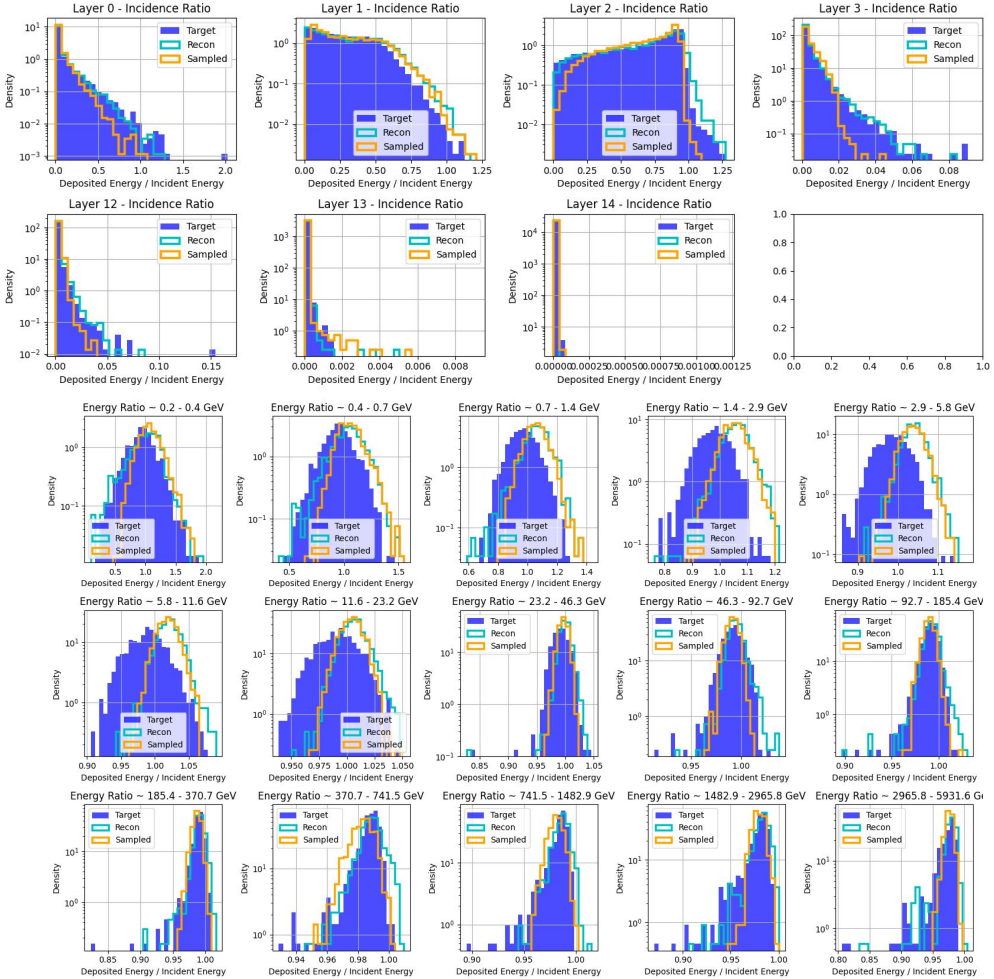
- Removed heads in all subdecoders except for last
- Performance without regularization on par compared to old models with regularization



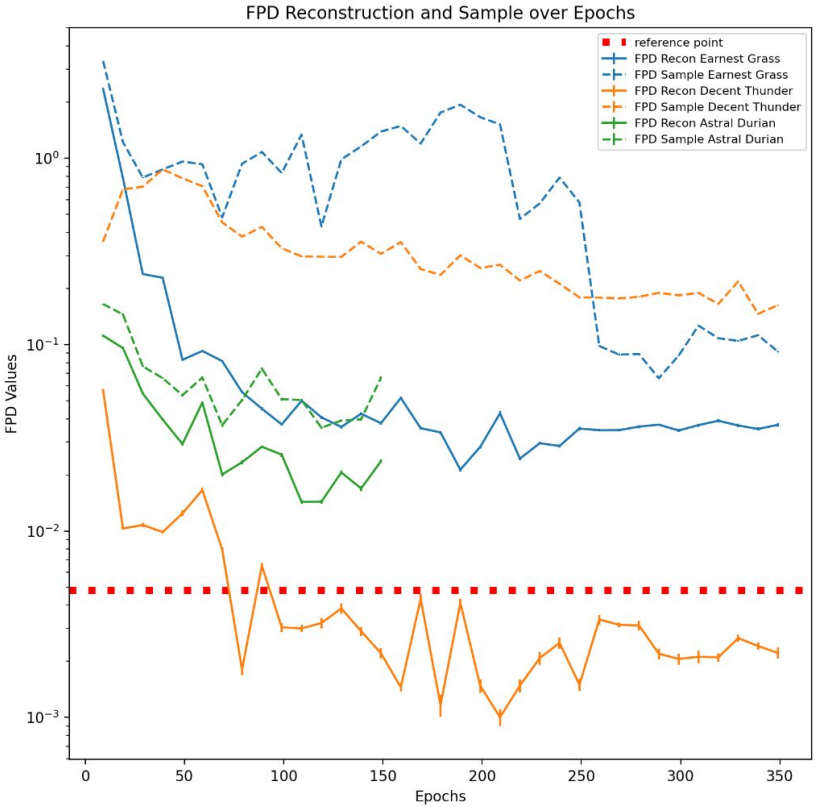
Geodecoder With Two Heads

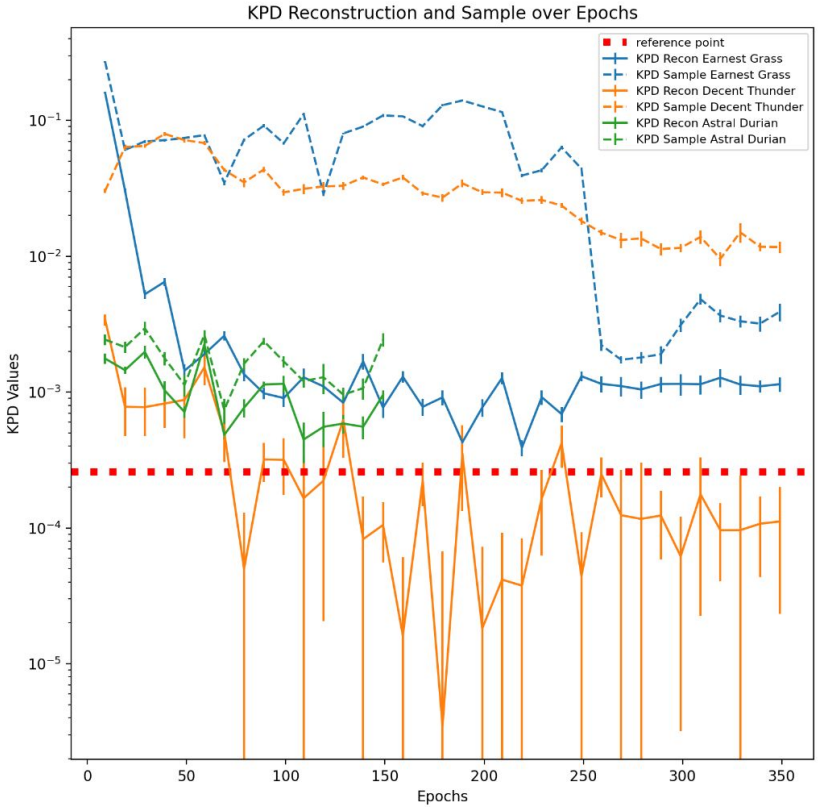
- Epoch 150, Training in VAE Mode





KPD and FPD for Geodecoder





Optimizers Improvements

- Introduced saving and loading optimizers for both VAE and RBM
- Added RBM code to change weights in place to support Pytorch optimizer

```python
@property
def weight_dict(self):
    """Accessor method for a protected variable

    :return: dict with partition combinations as str keys ('01', '02', ...)
             and partition weight matrices as values (w_01, w_02, ...)
    """
    # if self._qpu:
    for key in self._weight_dict.keys():
        self._weight_dict[key] = self._weight_dict[key] \
            * self._weight_mask_dict[key]
    return self._weight_dict
```

```python
import torch
from model.rbm.rbm import RBM
from torch.optim import Adam

class RBMtorch(RBM):
    def __init__(self, cfg=None):
        super(RBMtorch, self).__init__(cfg)

    @property
    def weight_dict(self):
        """Return masked weights as a dict.

        :return: dict with partition combinations as str keys ('01', '02', ...)
                 and masked partition weight matrices as values.
        """
        masked_weight_dict = {}
        for key in self._weight_dict.keys():
            masked_weight_dict[key] = self._weight_dict[key] * self._weight_mask_dict[key]
        return masked_weight_dict


    def initOpt(self):
        params_to_optimize = list(self._weight_dict.values()) + list(self.bias_dict.values())

        self.opt = Adam(
            params_to_optimize,
            lr=self._config.rbm.lr,
        )

    def update_params(self):
        self.opt.zero_grad()

        with torch.no_grad():
            for key, param in self._weight_dict.items():
                if key in self.grad["weight"]:
                    param.grad = -1 * self.grad["weight"][key].detach()
            for key, param in self.bias_dict.items():
                if key in self.grad["bias"]:
                    param.grad = -1 * self.grad["bias"][key].detach()
        self.opt.step()
```