Energy Reconstruction with CALIFA

Tobias Jenegger

CALIFA Calorimeter

Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC-2094 – 390783311,
BMBF 05P19WOFN1, 05P21WOFN1
and the FAIR Phase-0 program

TUM Members:
Roman Gernhäuser,Lukas Ponnath,Philipp Klenze,Tobias Jenegger

CALIFA Detector @ R^{3}B

CALorimeter for the **I**n **F**light detection of γ-rays and light charged p**A**rticles

Endcap:

iPhos:

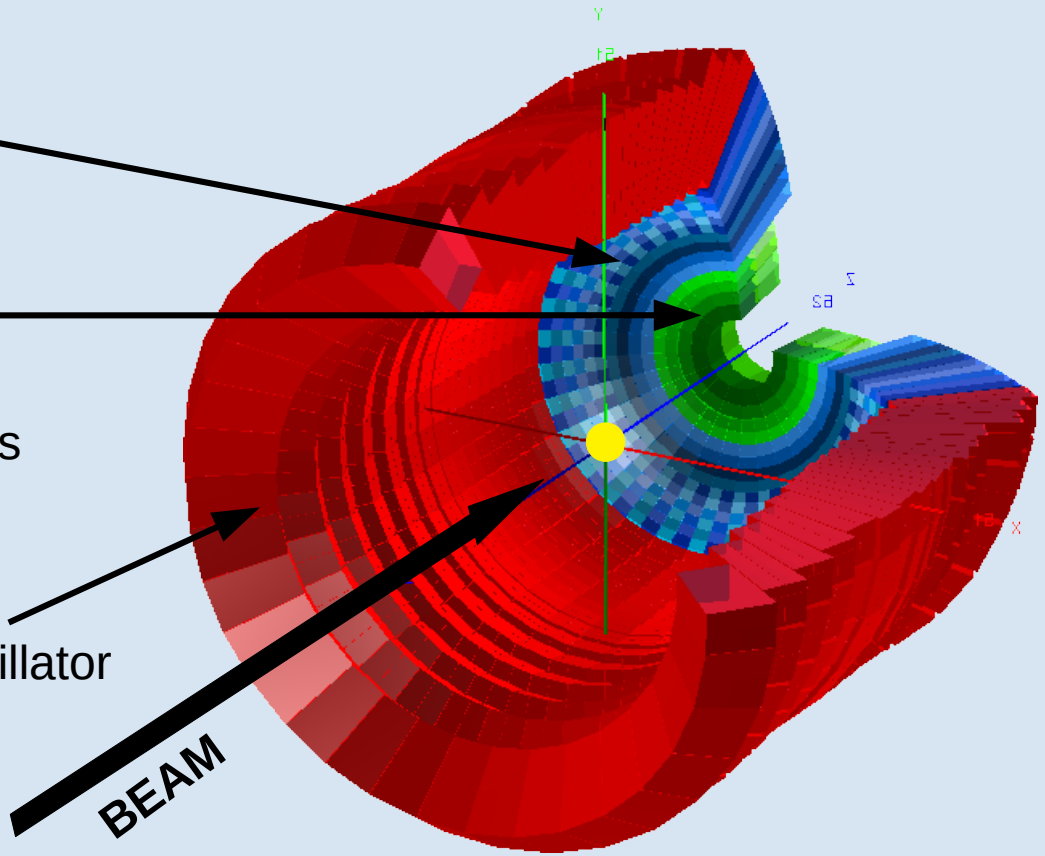
512 CsI(Tl) crystals

CEPA:

96 LaBr3 & LaCl3 crystals

Barrel:

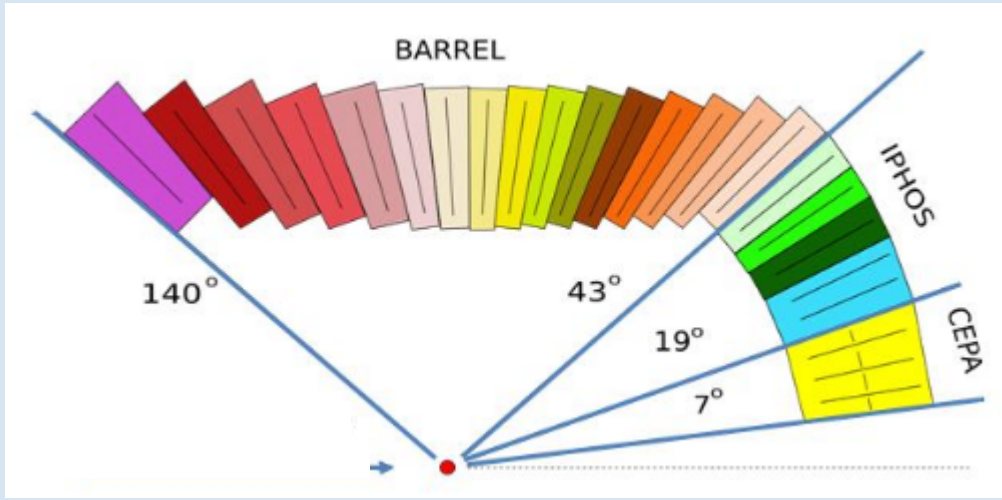
1952 CsI(Tl) scintillator crystals



Over 2500 crystal channels!

Requirements:

-high dynamic range:
  100 keV γ-rays – 700 AMeV charged particles
-high efficiency
-high granularity → Doppler correction
-particle identification



Observables

Energy

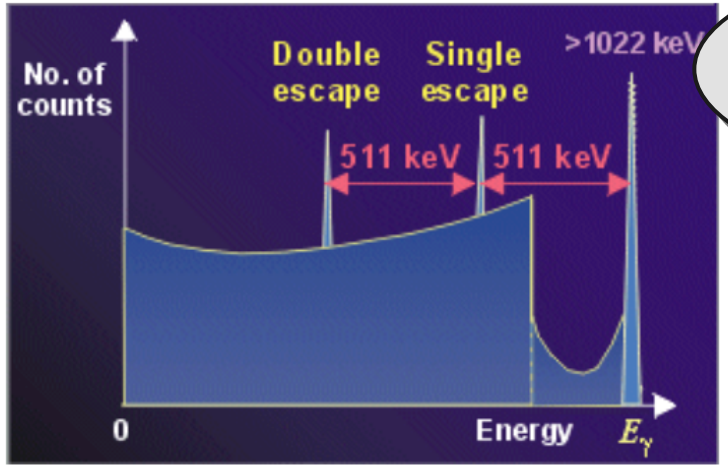
Hit Time

Spatial Position

Correlation with neighboring crystals

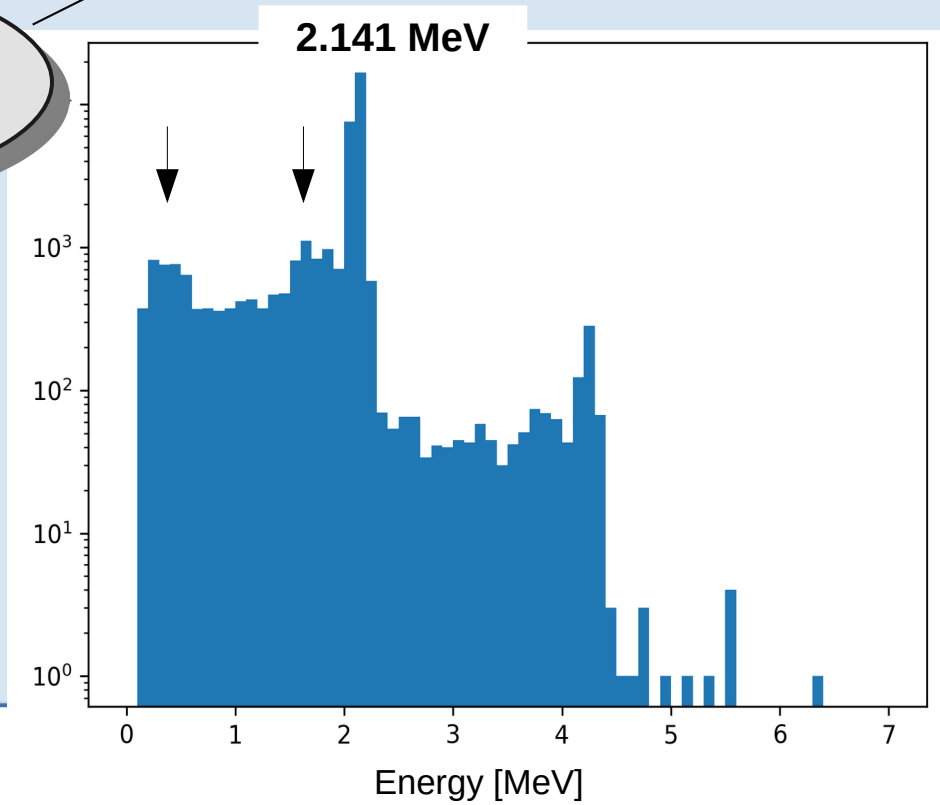
Customized Cluster

In a real detector, if the incident gamma energy is above 1022 keV, pair production events result in the production of two 511 keV annihilation gamma-rays.



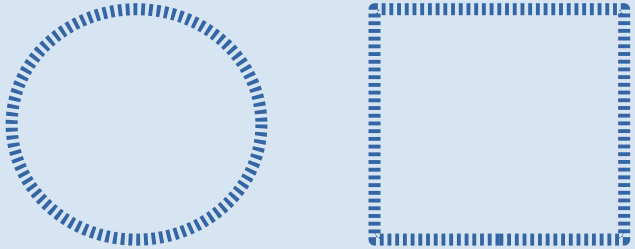
If only one of these gamma-rays escapes while the other is completely absorbed in the detector, 511 keV will be lost from the detector. This results in a separate peak in the spectrum representing Eg-511 keV, called the **single** escape peak.

If both annihilation gamma-rays escape this gives rise to the **double** escape peak at Eg-1022 keV.



Standard Cluster Algorithm

User defines shape and size of cluster:

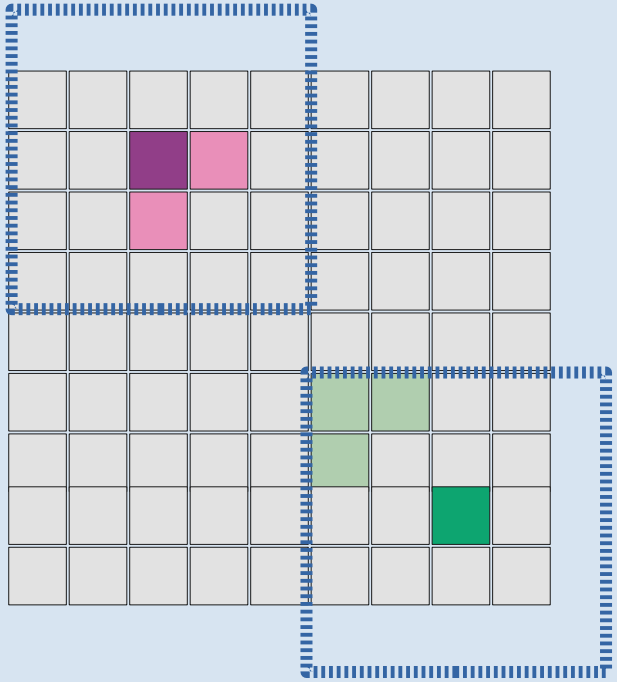


(and set energy threshold for single crystals)

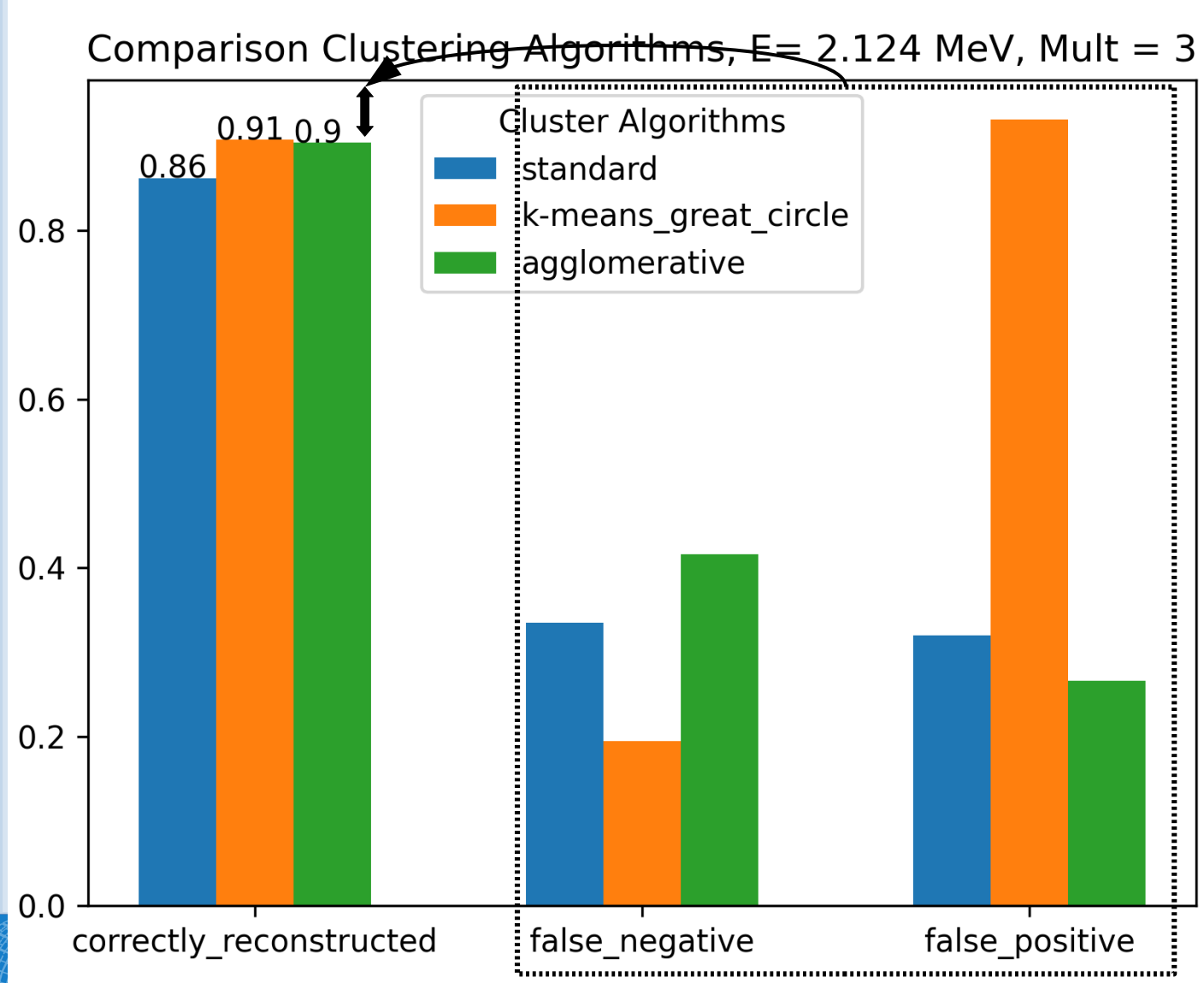
Sort the hit list according to their energy

30. MeV
22. MeV
10. MeV
5. MeV
3. MeV
2.5 MeV
0.7 MeV

1. create cluster centered around first hit
2. loop over all hits in list
   → if hit inside cluster add it and remove it from the list
3. Do this procedure until list is empty



Summary Clustering Methods



false_negative:

true cluster

$\begin{pmatrix} 1.2 \\ 0.8 \\ \mathbf{0.124} \end{pmatrix}$

reconstructed cluster

$\begin{pmatrix} 1.2 \\ 0.8 \\ 0.511 \end{pmatrix}$

→ false negative = $\frac{0.124}{2.124}$

false_positive:

true cluster

$\begin{pmatrix} 1.2 \\ 0.8 \\ 0.124 \end{pmatrix}$

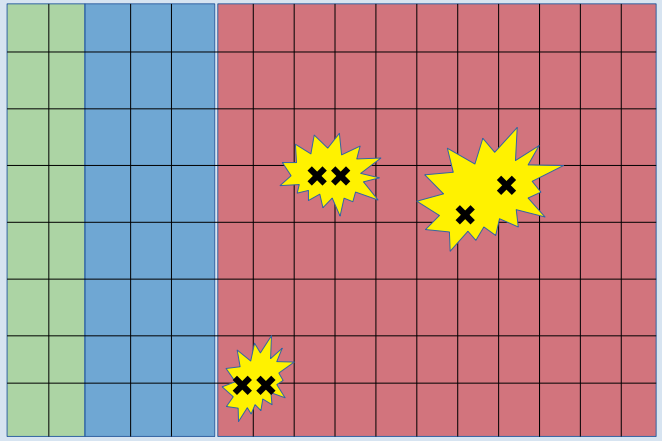
reconstructed cluster

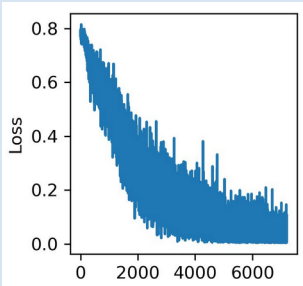
$\begin{pmatrix} 1.2 \\ 0.8 \\ \mathbf{0.511} \end{pmatrix}$

→ false positive = $\frac{0.511}{2.124}$

Invariant Slot Attention Model

Starting with energy and position information (no time):

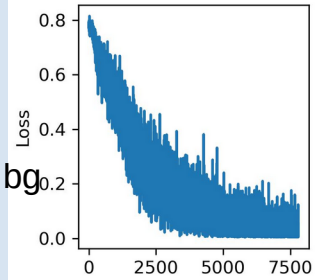


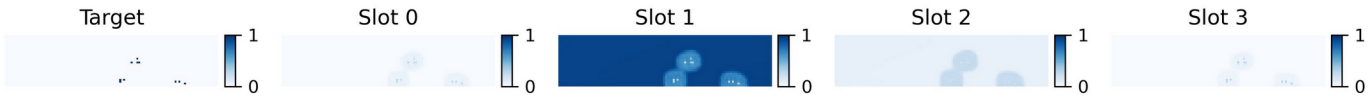


iter 7250 , loss 0.009908747

```
class InvariantSlotAttention(torch.nn.Module):
    def __init__(self,
                 resolution=(27,112),
                 xlow=-0.5,
                 xhigh=0.5,
                 k_slots=4,
                 num_conv_layers=3,
                 hidden_dim=16,
                 final_cnn_relu=False,
                 query_dim=32,
                 n_iter=2,
                 pixel_mult=1,
                 device='cpu'
                 ):
        '''
```

3 clusters + bg





iter 7800 , loss 0.020807367

Parameters I tuned:
Learning rate: 1e-4 to 1e-5
hidden dim: 16 – 32
query_dim: 10 – 16 – 32 - 64

Invariant Slot Attention Model – also with time info

→ Dimension of mask: → same as before! 10 x 3 x 27 x 112

→ Dimension of evt_histogram_array : 10 x **2** x 27 x 112

Lr = 5e-5

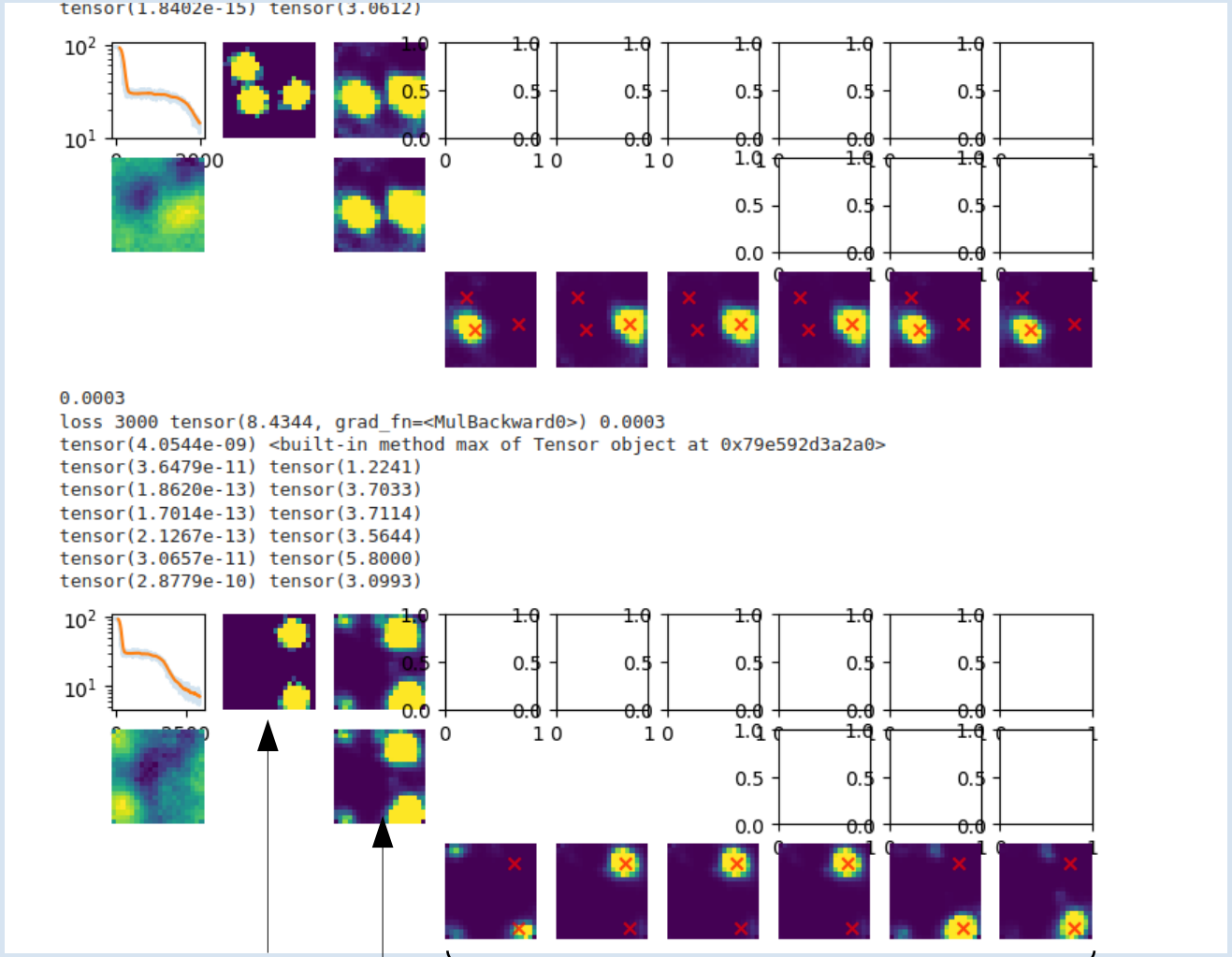
#observables:
time & energy

```
self.gru = torch.nn.GRUCell(self.query_dim, self.query_dim)

kwargs = {'out_channels': hidden_dim,'kernel_size': 5, 'padding':2 }
#cnn_layers = [torch.nn.Conv2d(1,**kwargs)] old cnn, with one input channel, energy
cnn_layers = [torch.nn.Conv2d(2,**kwargs)] #now also with time info

for i in range(num_conv_layers-1):
```

Loss function does not converge!

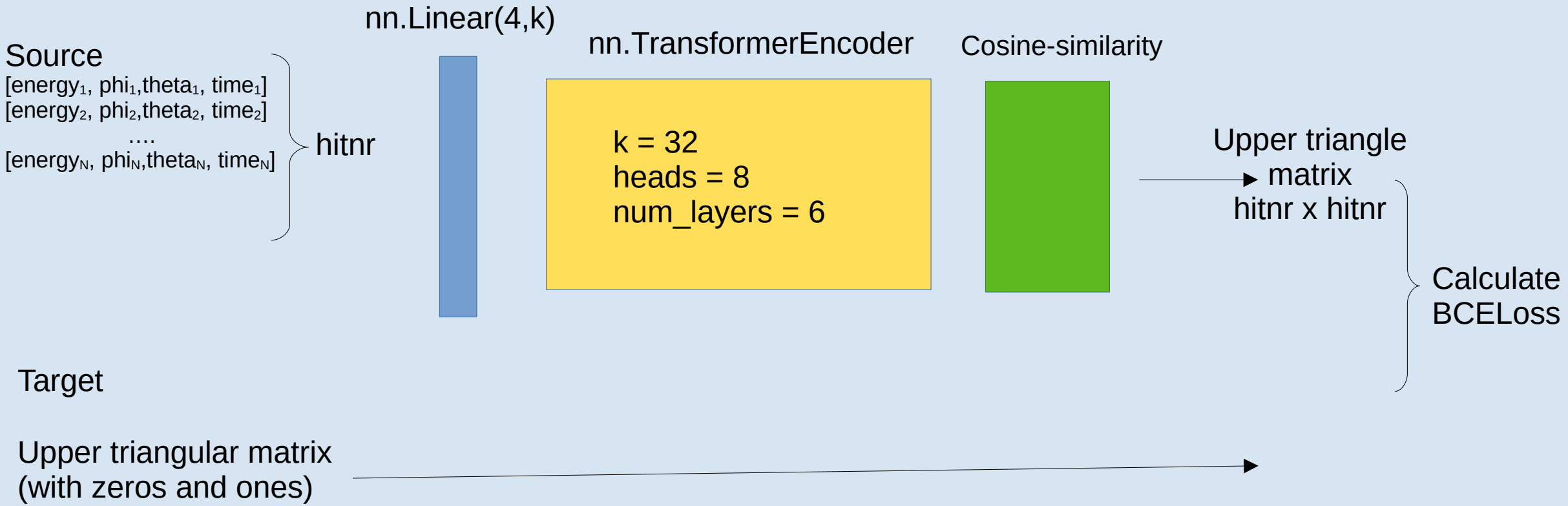
slot_and_tspn_onenotebook from Lukas Heinrich

https://gist.github.com/lukasheinrich/31d06bc4918e52d7ae3663a197b90d71



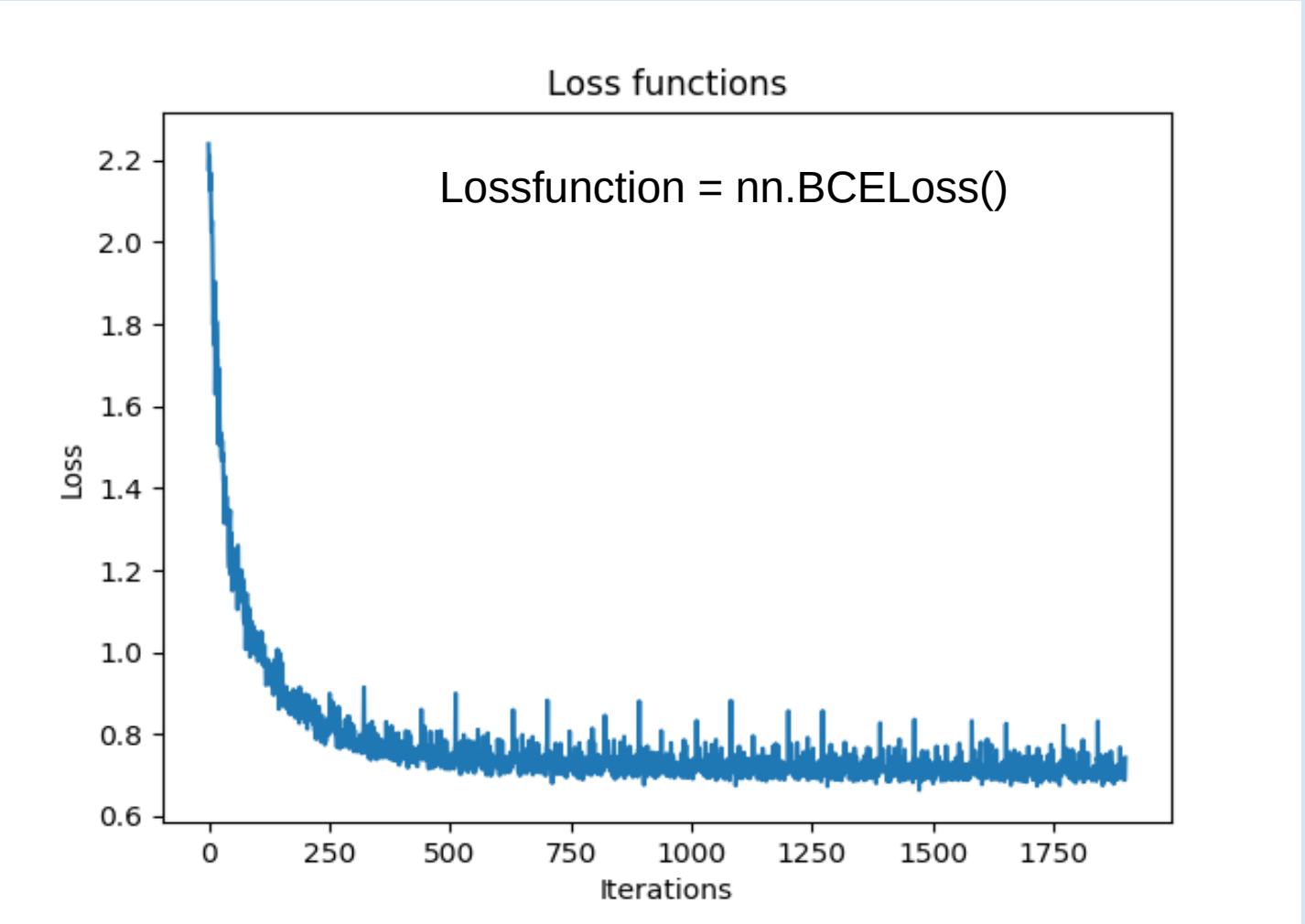

```python
class AttModel(torch.nn.Module):
    def __init__(self):
        super().__init__()
        self.latent_dim = 32

        # self.encoder = TSPNEncoder(n_slots = 6)
        # self.encoder = SlotAttentionEncoder(n_slots = 6)
        self.encoder = AddNoiseEncoder(n_slots = 6)
        self.decoder = torch.nn.Sequential(
            torch.nn.Linear(self.latent_dim,128),
            torch.nn.ReLU(),
            torch.nn.Linear(128,256),
            torch.nn.ReLU(),
            torch.nn.Linear(256,NBINS*NBINS),
            torch.nn.Unflatten(-1,(NBINS,NBINS))
        )


    def forward(self, data):
        Nbatch, *_ = data.shape
        positions,queries = self.encoder(data)
        decoded = self.decoder(positions).exp()/2.
        reco = decoded.sum(dim = 1)
        return reco,queries,decoded
```

Transformer model



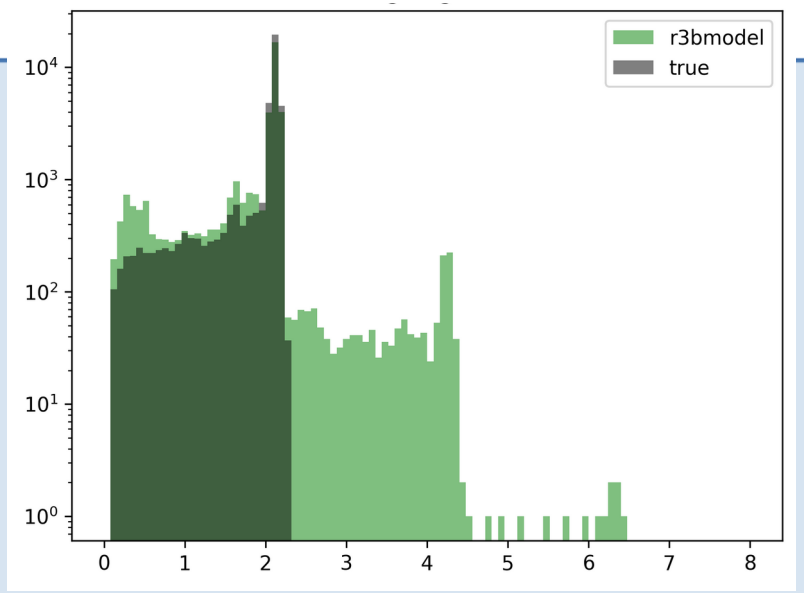
Transformer model – further parameters

Batchsize = 64
Feature number = 32
n_epochs = 10
Loss_rate = 2e-4
Loss function = nn.BCELoss()



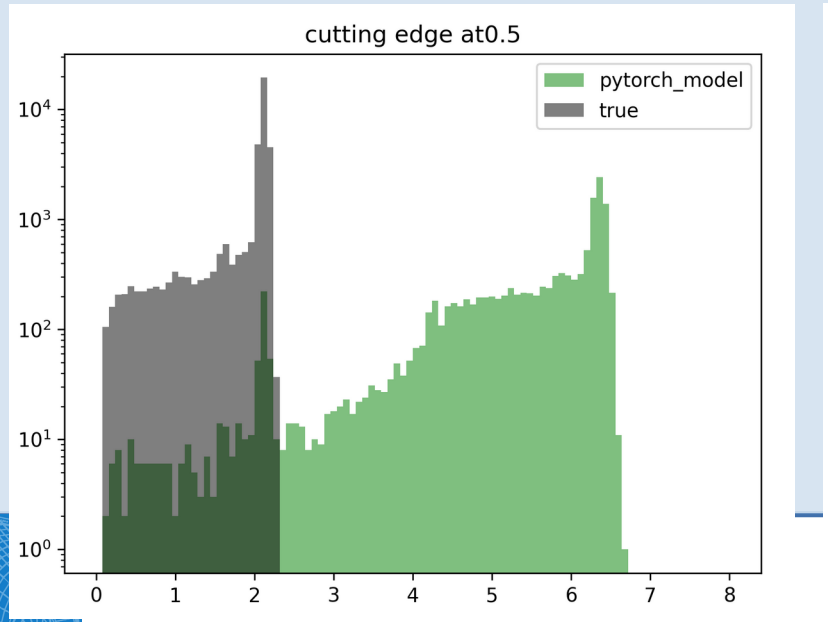
How do the energy spectra look like?

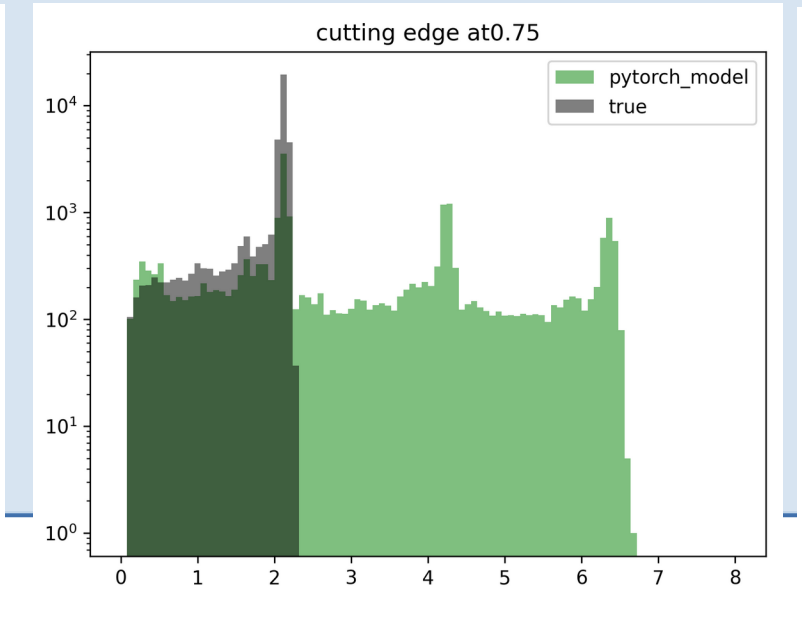
Standard Cluster vs True Clusters

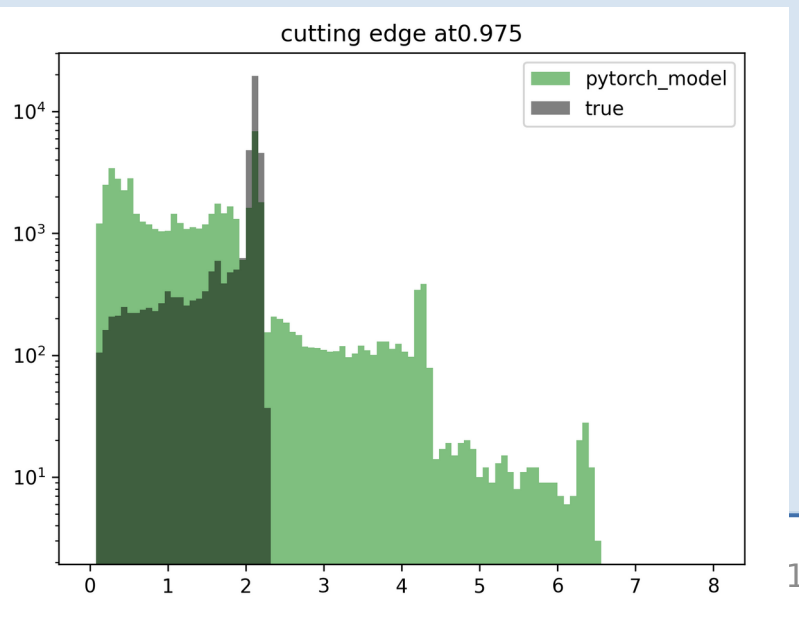


How to clusterize hits from output of transformer model:

1) Take the upper triangular matrix tri[hitnr x hitnr]
2) set “merge cut”. If tri[i,j] > “merge_cut” → hits belong to same cluster
3) do this for all combinations and merge them appropriately







Why energy spectra so bad while loss function seems to decrease ?

Most entries in model output tensor ~0.5. This diminishes the loss BCELoss function!

How to improve?

- Include some cut condition in the forward part of the transformer model

```
#out_ret_val = torch.where(ret_val > 0.7, torch.FloatTensor(1,requires_grad=True), torch.FloatTensor(0,requires_grad=True))
```

→ discontinuity of loss function → no learning!

- Use linear net instead of cosine similarity

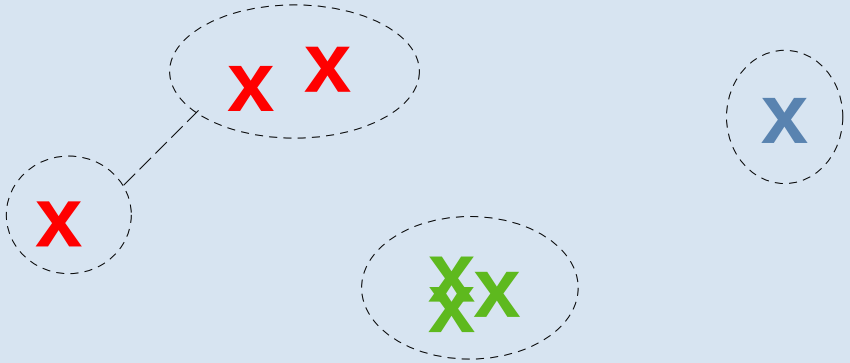
```
output_tensor = torch.cat([tensor_i.expand(-1, -1, expansion_factor, -1), tensor_j.expand(-1, expansion_factor, -1, -1)], dim=-1)
#small net:
net = nn.Sequential(
        nn.Linear(64,8),
        nn.ReLU(),
        nn.Linear(8,1),
        nn.Sigmoid()
        )
res = net(output_tensor)
temp_res = torch.squeeze(res)
upper_tri_mask = torch.triu(torch.ones((temp_res.shape[1],temp_res.shape[1])),diagonal=1).bool() #out[1] is max hit number in batch
result = temp_res[:,upper_tri_mask]
```

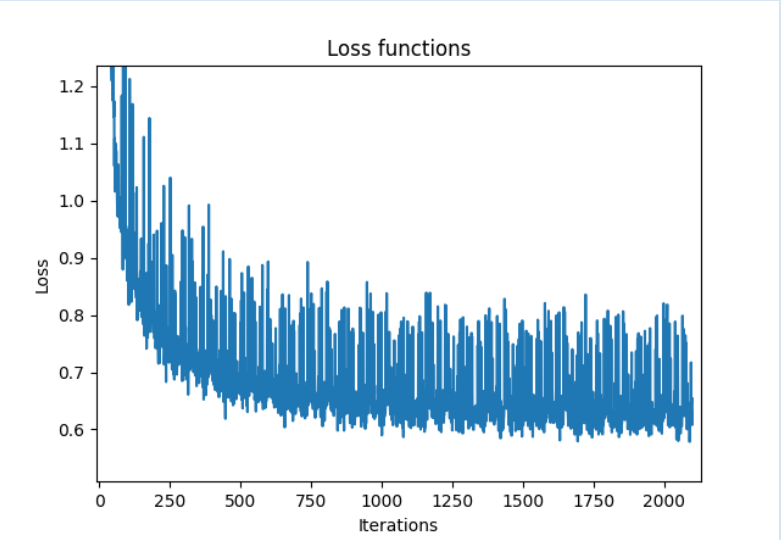
No improvements!

Agglomerative Model + Transformer Model

Idea:

1. Use first agglomerative method to cluster
2. Select events where we have too many clusters (false negative)

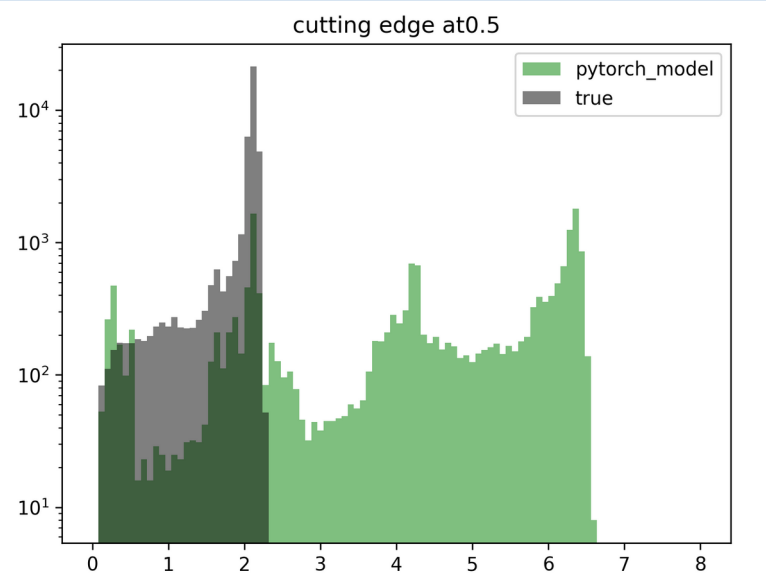


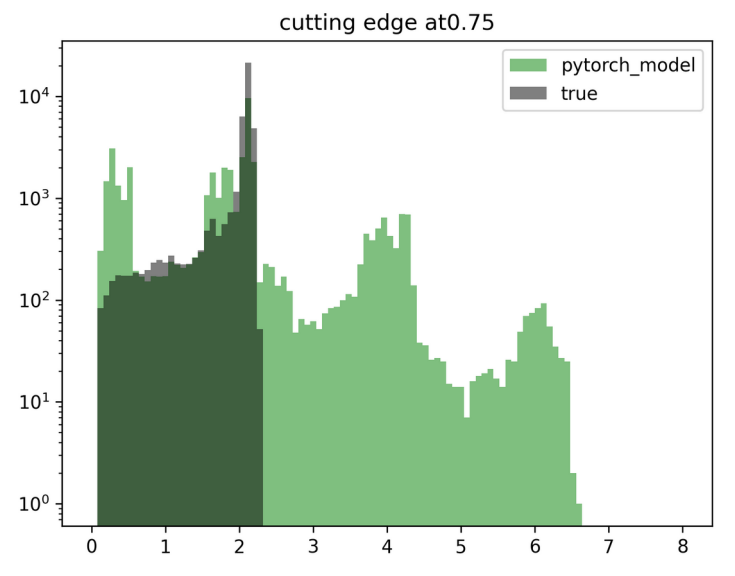


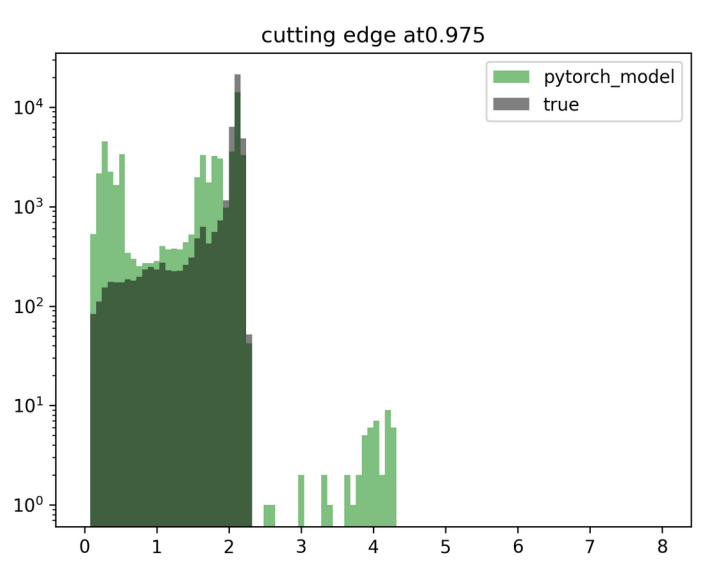
3. Feed the clusters to the transformer model (calculating cm of clusters)

Transformer Model

Agglomerative Model + Transformer Model - Spectra







Thank you!

CALIFA @ Technical University of Munich (TUM)

Roman Gernhäuser, Lukas Ponnath, Philipp Klenze, Tobias Jenegger

GEFÖRDERT VOM