Machine Learning for HGCal

ID and Clustering using graphs and CNN based deep learning

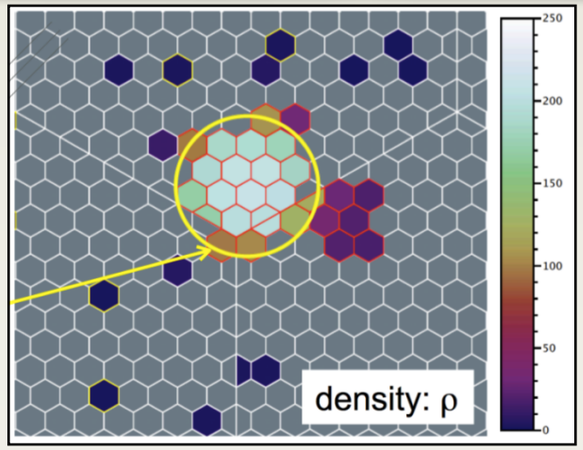
Shah Rukh Qasim | Jan Kieseler
Yutaro Iiyama | Maurizio Pierini

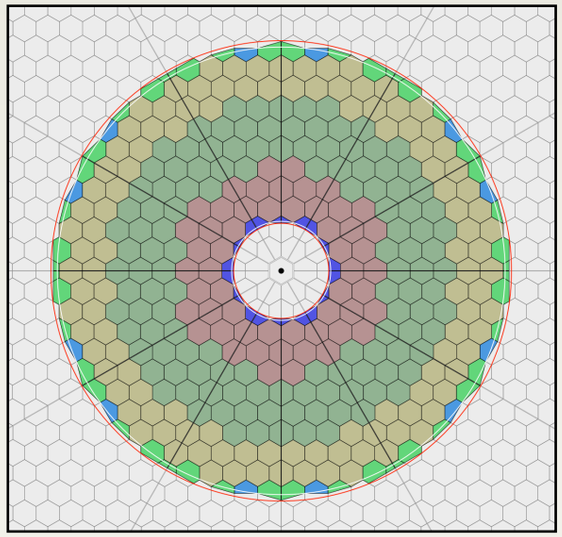
28.11.2018

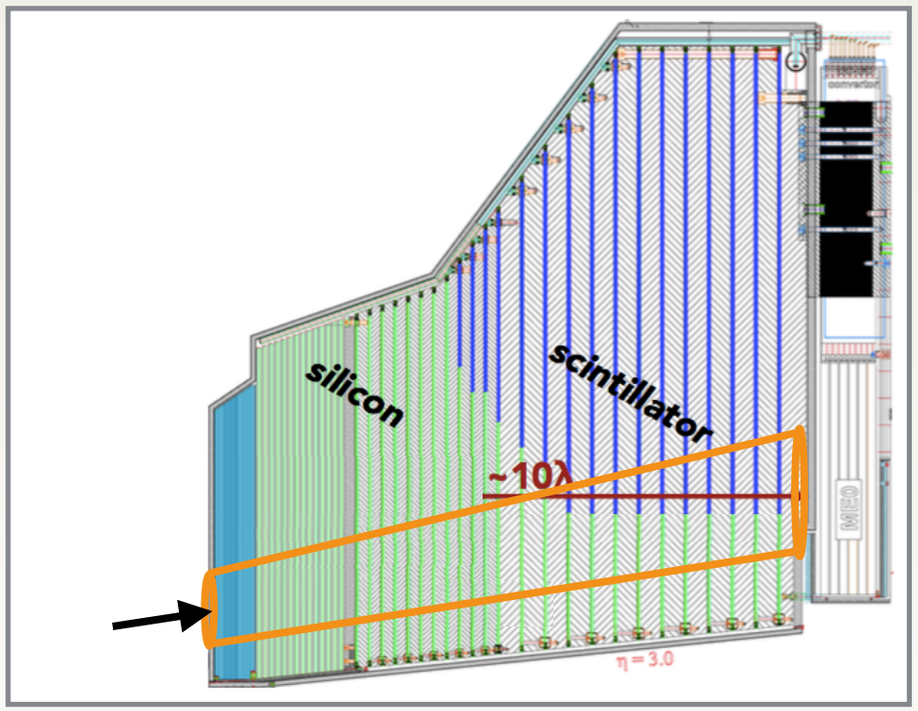
CMG-HGCAL group

M. Beguin, J. Bendavid, A. David, G. Cerminara, A. Gilbert, I. Gorbunov, A. Jafari, J. Kieseler, C. Lange, E. Locci, L. Malgeri, M. Mannelli, A. Martelli, P. Milenovic, F. Moortgat, J. Niedziela, F. Pantaleo, M. Pierini, S.R. Qasim, M. Rovere, M. Selvaggi, P. Silva, Y. Iiyama

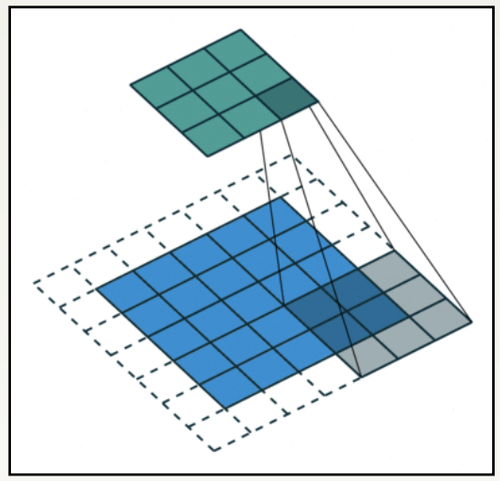
HGCal and Neural Networks (reminder)







- Sensors hexagonal
- Sensor size/area changes with layers
- Sensor size/area changes within one layer



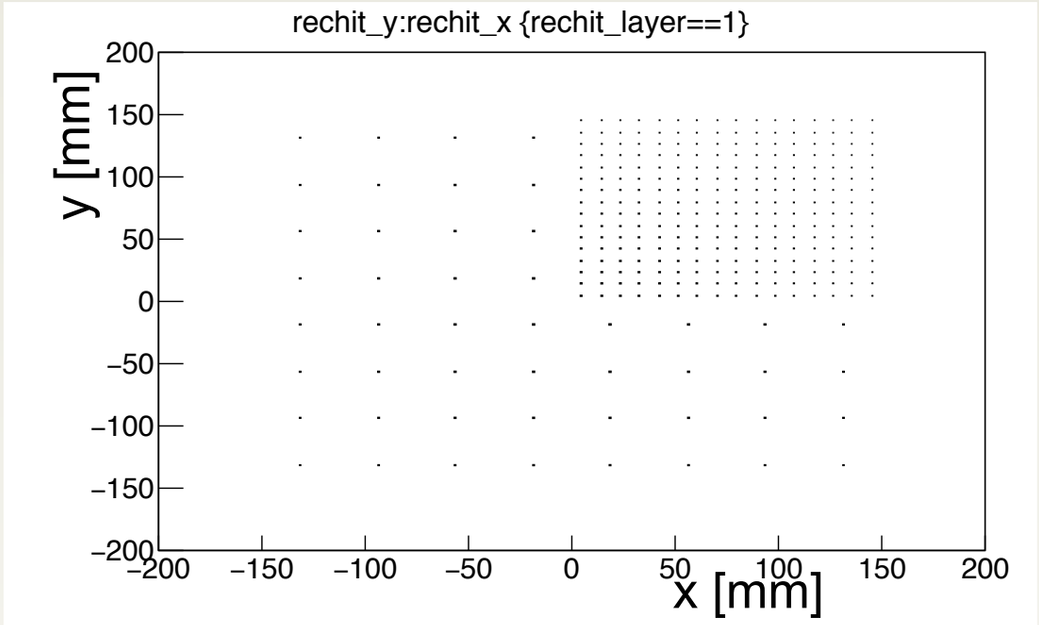
- Convolutional networks (as used in TDR): Uniform pixel size in all dimensions
  - works but not natural to the problem and creates overhead

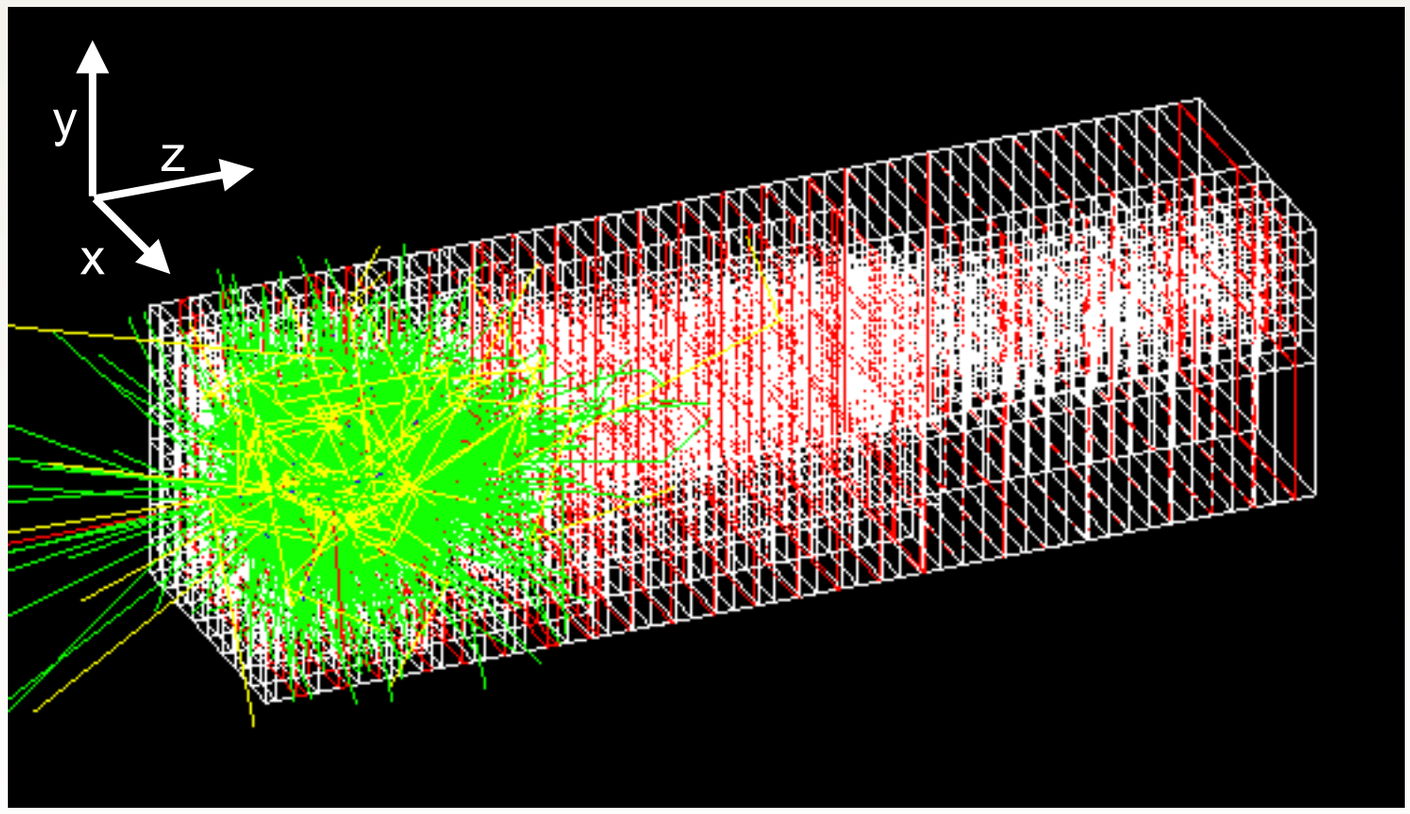
➡Try to use networks not based on a fixed grid

Dataset

- Simple calorimeter simulation
  - Not using HGCal simulation
    - Fast generation of samples and simple to change geometry
    - Direct output in right format (otherwise 3-4 steps)
  - Full Geant4 simulation
  - Tungsten, about 11 λ_0 in z, 30 cm x 30 cm in (x,y)
  - Non homogenous geometry
    - Sensor sizes from about 1 cm^2 to 15x15 cm^2
    - Sizes change within the layer
  - Particles with energies 1-100 GeV
    - 5 cm x 5 cm window around (x,y)=(0,0)
    - Different granular parts are hit
    - Shot from distance of 2 m
    - Total size: 1M events
    - (more details later)

➡Resembles properties of HGCal that pose challenges to ML approaches





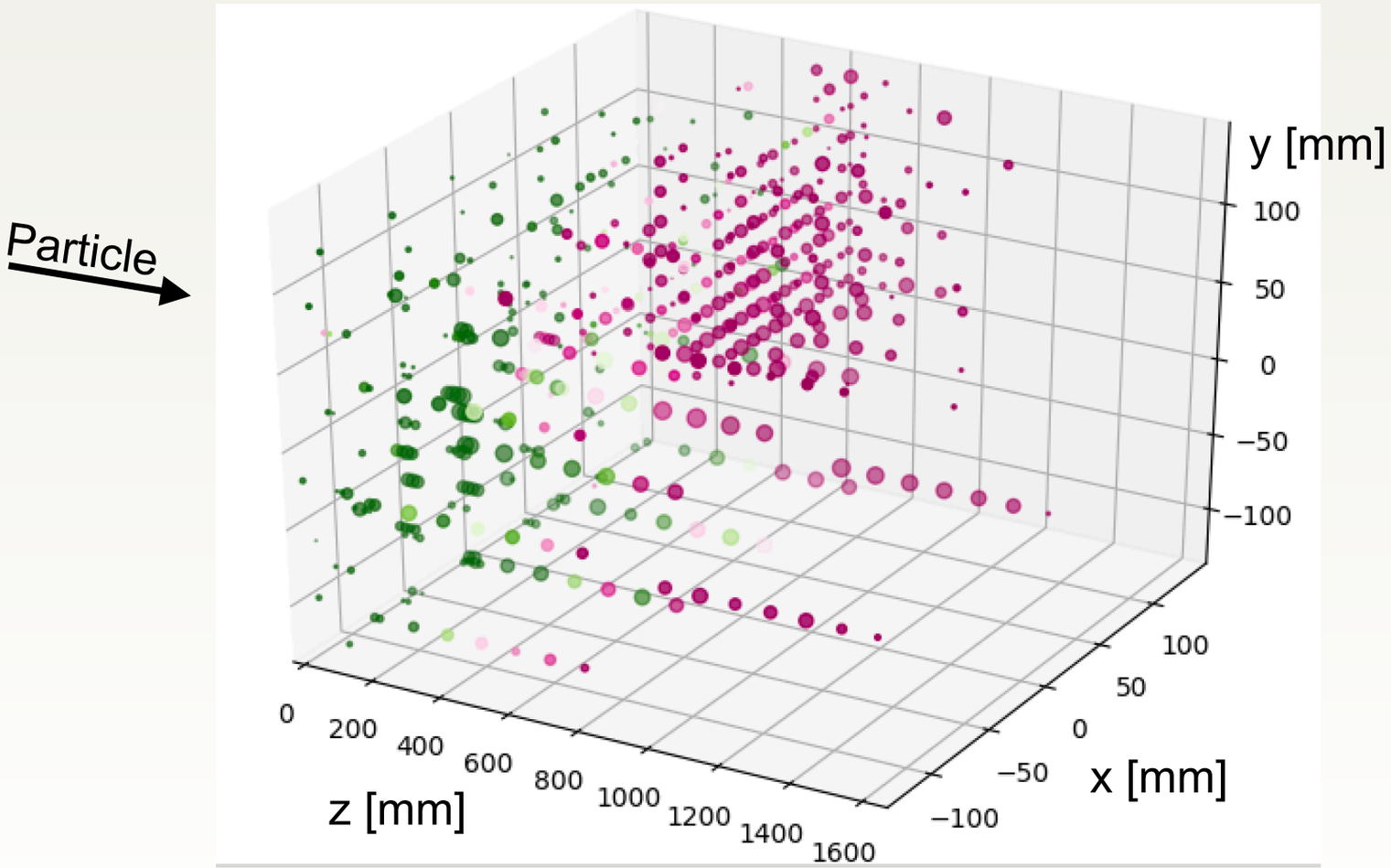
Clustering

- Clustering

  Two showers at in any sample

  Both charged pions

  Dataset: linear combination of showers of both particles



Clustering - Notation

Ground truth shower fractions:

- Matrix of shape [N, 2] donated by **T** (for target)

$$T = \begin{bmatrix} t_1 & t_2 & \dots & t_N \end{bmatrix}$$

- Where t_i is a 2D vector, fractions of of two showers of ith sensor

Predicted shower fractions

- Matrix of shape [N, 2] donated by **P** (for prediction)

$$P = [p_1 \quad p_2 \quad \dots \quad p_N]$$

- Where p_i is a 2D vector, fractions of of two showers of ith sensor

Energy deposited on sensors

- ND vector denoted by **e**

$$e = [e_1 \quad e_2 \quad \dots \quad e_N]$$

- Where e_i is a scalar representing energy deposited on ith sensor

Clustering - Loss function

Loss functions (12)

$$L(\boldsymbol{P}, \boldsymbol{T}, \boldsymbol{e}) = \frac{1}{\hat{E}} \sum_{s=1}^{N} \|(\boldsymbol{p_s} - \boldsymbol{t_s}) * f(e_s)\|_2$$

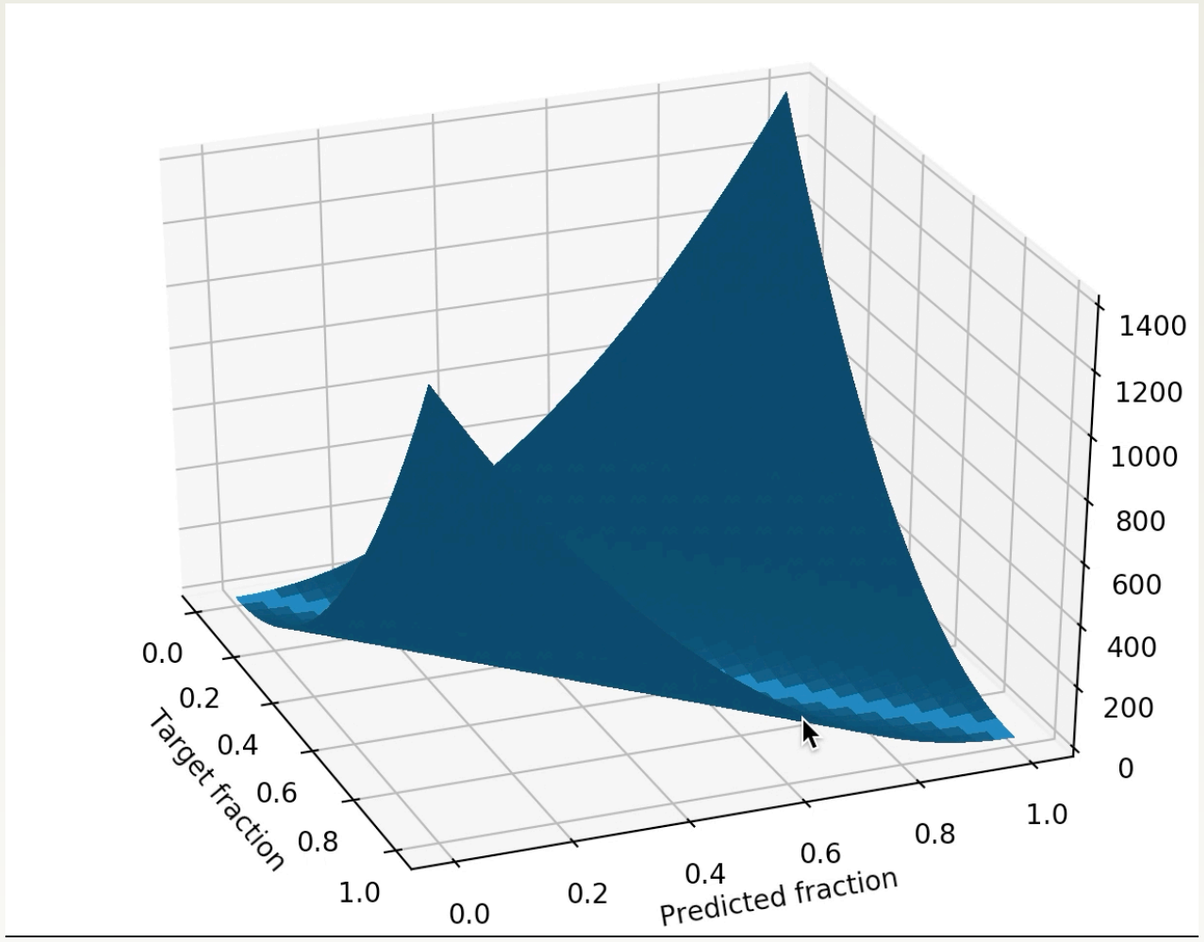
$$\hat{E} = \sum_{s=1}^{N} f(e_s)$$

Tried different

$f(e_s) = \sqrt{e_s}$ (Works best)

$f(e_s) = e_s$

$f(e_s) = ln(e_s + 1)$



$\|(p_s - t_s) * f(e_s)\|_2$

Clustering - Min loss function

Same solution if columns are swapped

$$\boldsymbol{P}' = 1 - \boldsymbol{P}$$

Since:

$$p_{s1} + p_{s2} = 1$$

We could, in theory, use more forgiving loss function:

$$L_{min} = min(L(\boldsymbol{P}, \boldsymbol{T}, \boldsymbol{e}), L(1 - \boldsymbol{P}, \boldsymbol{T}, \boldsymbol{e}))$$

Doesn't help for training

- Showers are well defined (first by x, then by y)

However, can use it for analysis

Swap the predicted or ground truth showers if

$$argmin(L(\boldsymbol{P}, \boldsymbol{T}, \boldsymbol{e}), L(1 - \boldsymbol{P}, \boldsymbol{T}, \boldsymbol{e})) = 2$$

Clustering - Binning

CNN based method

Input is tensor of dimensions [8, 8, 25, 64] (obtained by histogram based binning, scatter op)

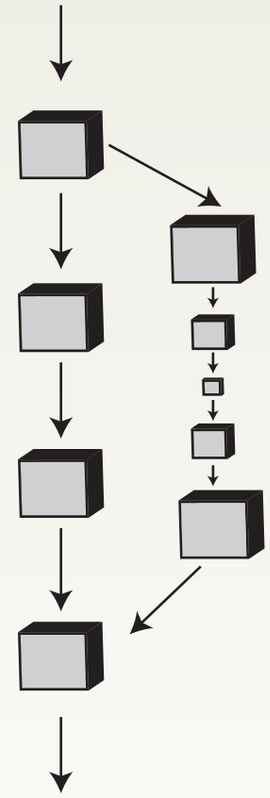
Multiple layers of 3D convolutions

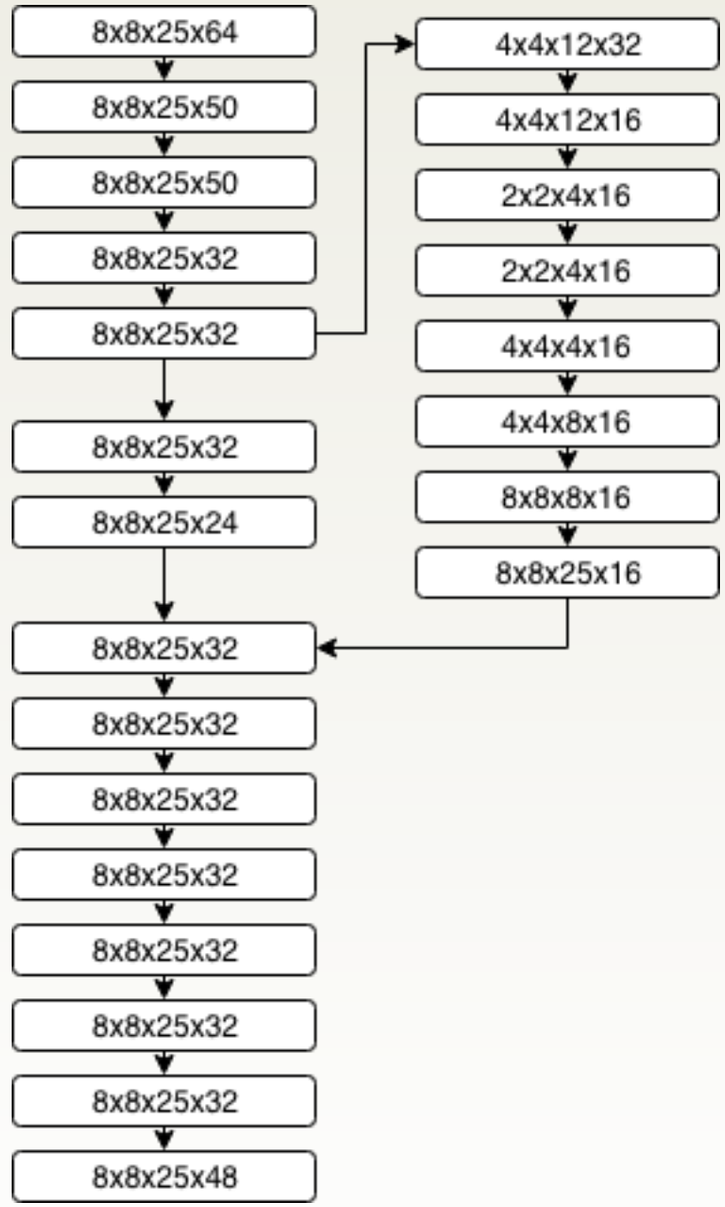
Combined with enc-dec network

- For better receptive field

Unbinned at the end (gather op)

~100k parameters





Graph convolution

Vertex = all sensors

Find k nearest neighbours for each vertex (in euclidean space)

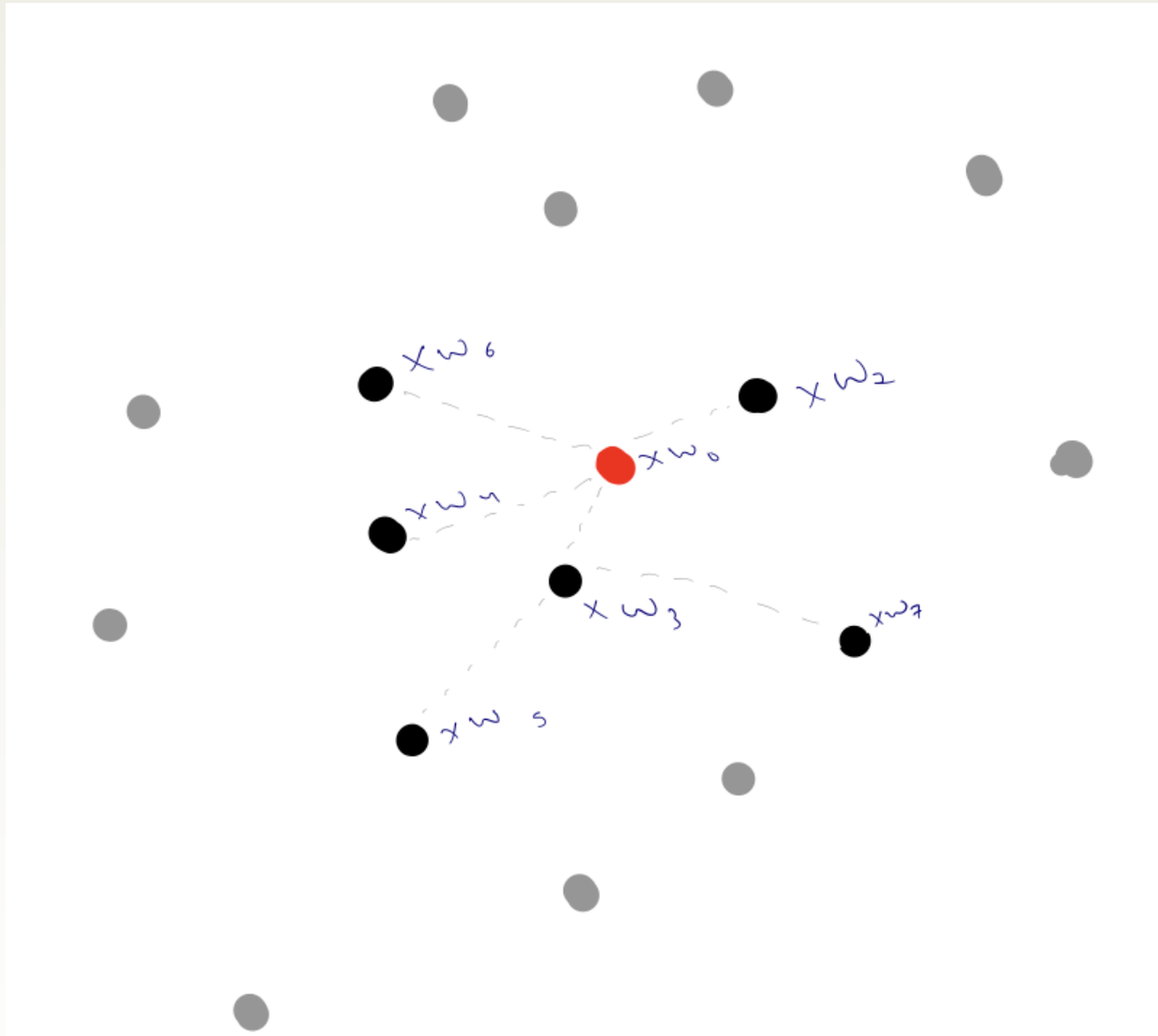
Apply conv filters on selected neighbours for each of the vertex

Propagate the result as in CNNs

5 layers of graph convolution

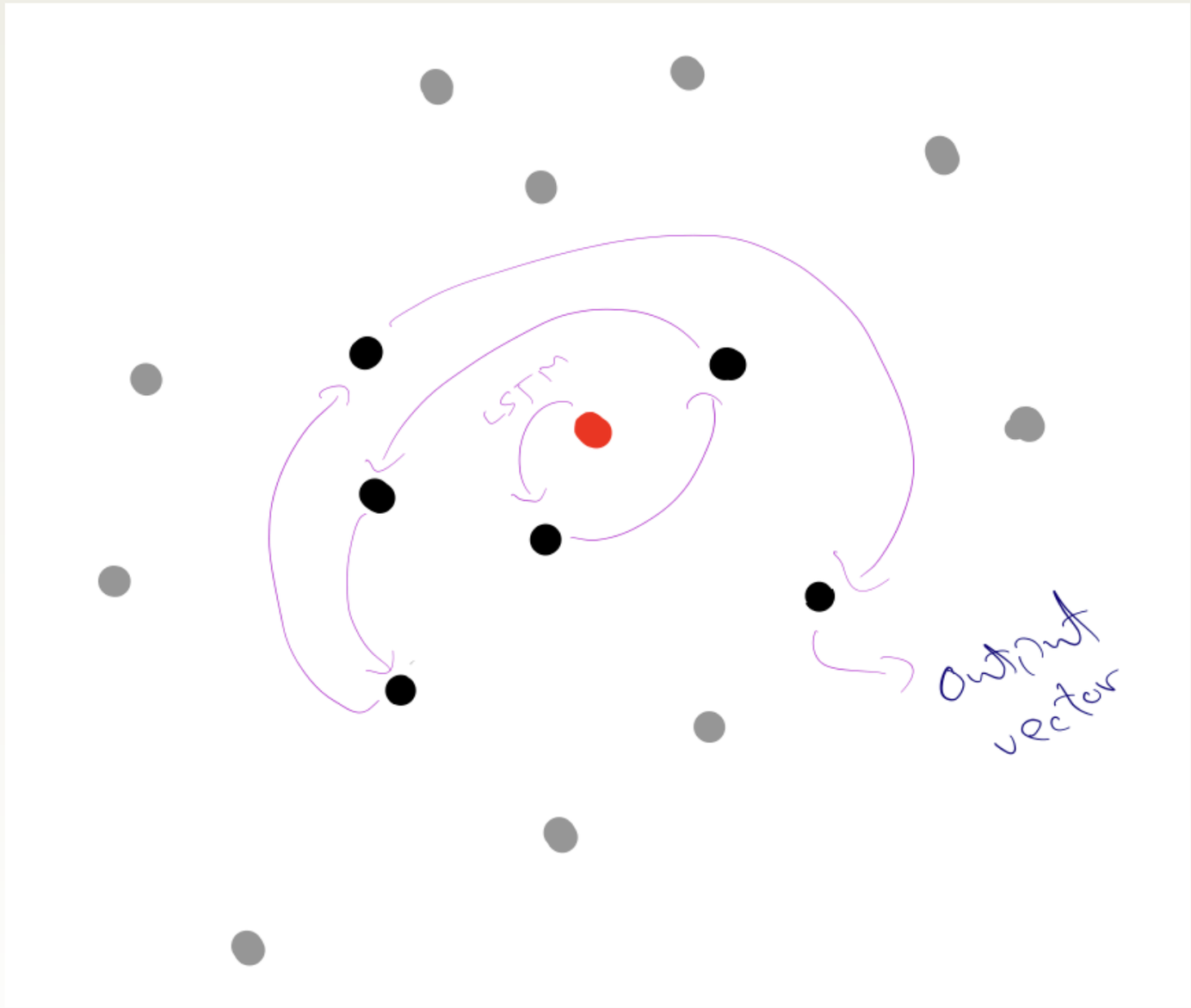
18 neighbours in each of the layers

$$o = \sum_{i=0}^{N} c_i * w_i$$



Graph neural network, LSTM aggregation

- Find k nearest neighbours for each vertex
- Use LSTM to aggregate the result to produce output
- Propagate the result as in graph convolution
- 7 layers of the network
- Each layer works aggregates on 18 neighbours



Graph neural network - Dynamic space

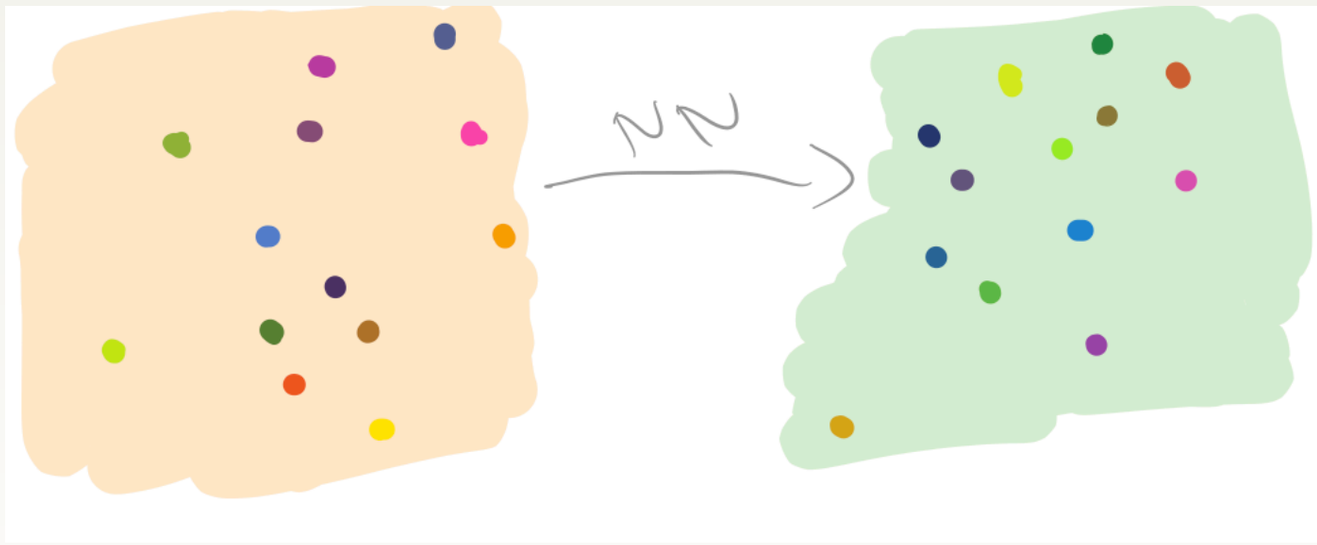
Transform features to another space

$$S' := NN(S)$$

Find nearest neighbours in new space **S'**

Aggregate the nearest neighbours using different methods

- Can be LSTM
- Can be simple filter sum



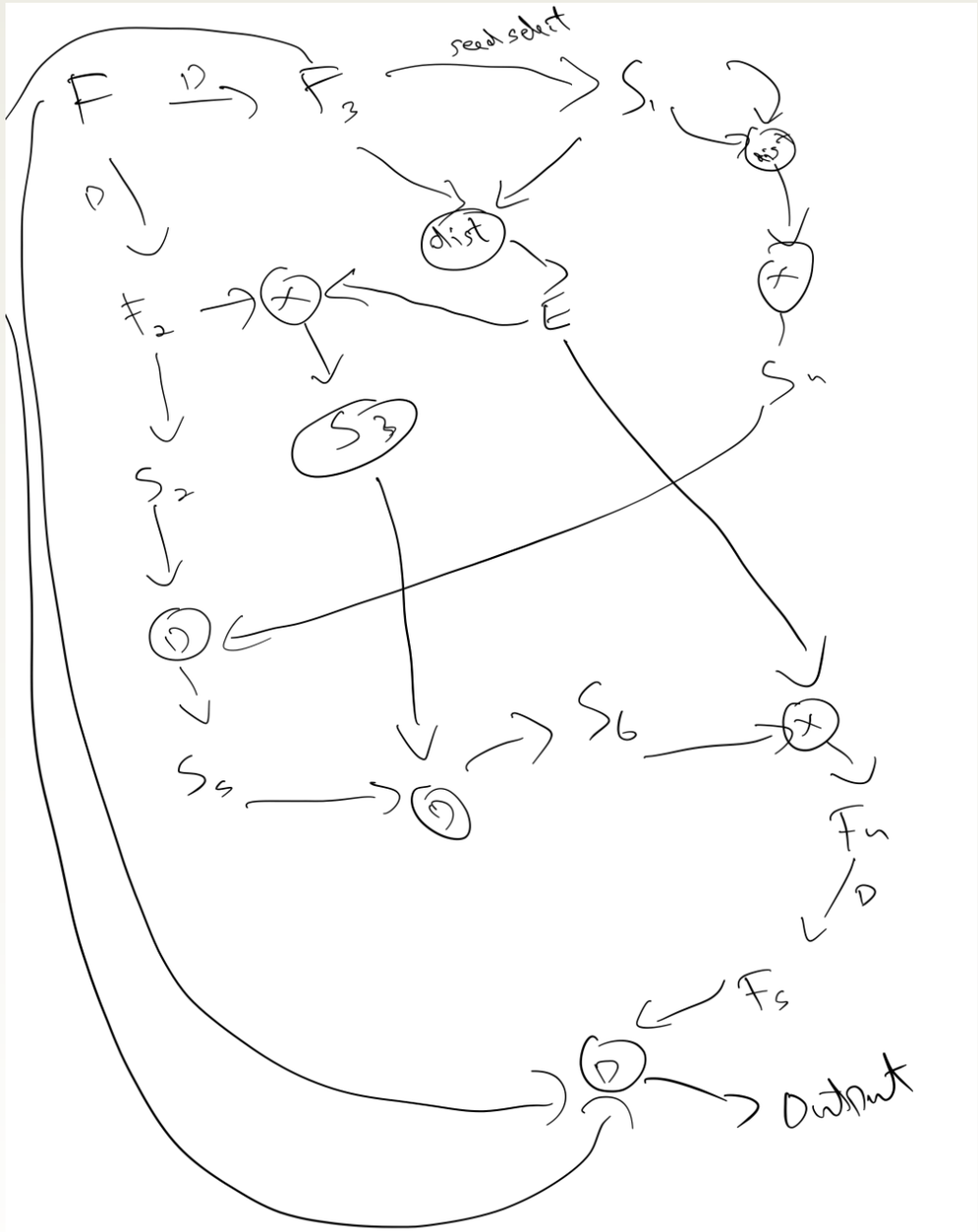
Reference: Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M.: Dynamic Graph CNN for Learning on Point Clouds. ArXiv e-prints (our network is very similar to this)

Graph neural network - Seed based

Each shower has a seed

Define seed as rechit with highest energy (or vertex in terms of graphs)

Complex network



Graph neural network - Seed based

Basic idea: propagate information through seeds

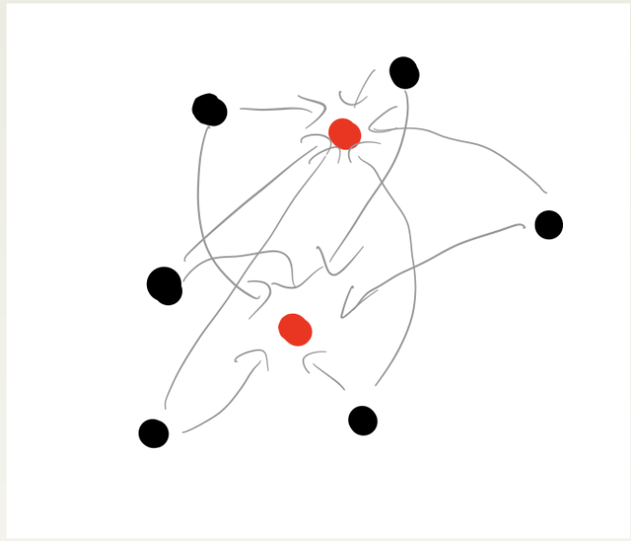
$$v'_s = NN_S(v)$$

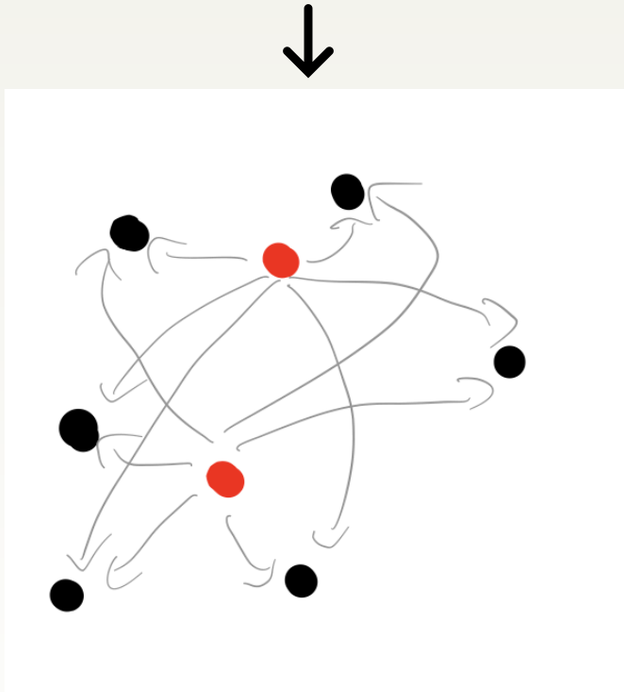
$$v' = NN_F(v)$$

$$e = NN[L_2(v'_s, v_s^{seed}), v'_s - v_s^{seed'}]$$

$$v^{seed''} = NN_c[e \,.\, v', e_s \,.\, v^{seed'}, v^{seed'}]$$

$$v'' = NN_o[v, e \,.\, v^{seed''}]$$





8 layers of previous network

Number of parameters = ~100k

(Another variation: include one random seed in addition)

Results

Test shower:

- Randomly select one of the showers, compute its response

Noise shower

- Shower overlapping with the test shower

$$r_q = \frac{\sum_{s=1}^{N} p_{sq} * e_s}{\sum_{s=1}^{N} t_{sq} * e_s} \qquad q \in \{1,2\}$$

If you test N showers, store N responses in N-d vector **r’**

Results

Mean: (standard statistic)

$$m = \frac{1}{N}\sum_{i}^{N} r'_i$$

Variance: (Standard statistic)

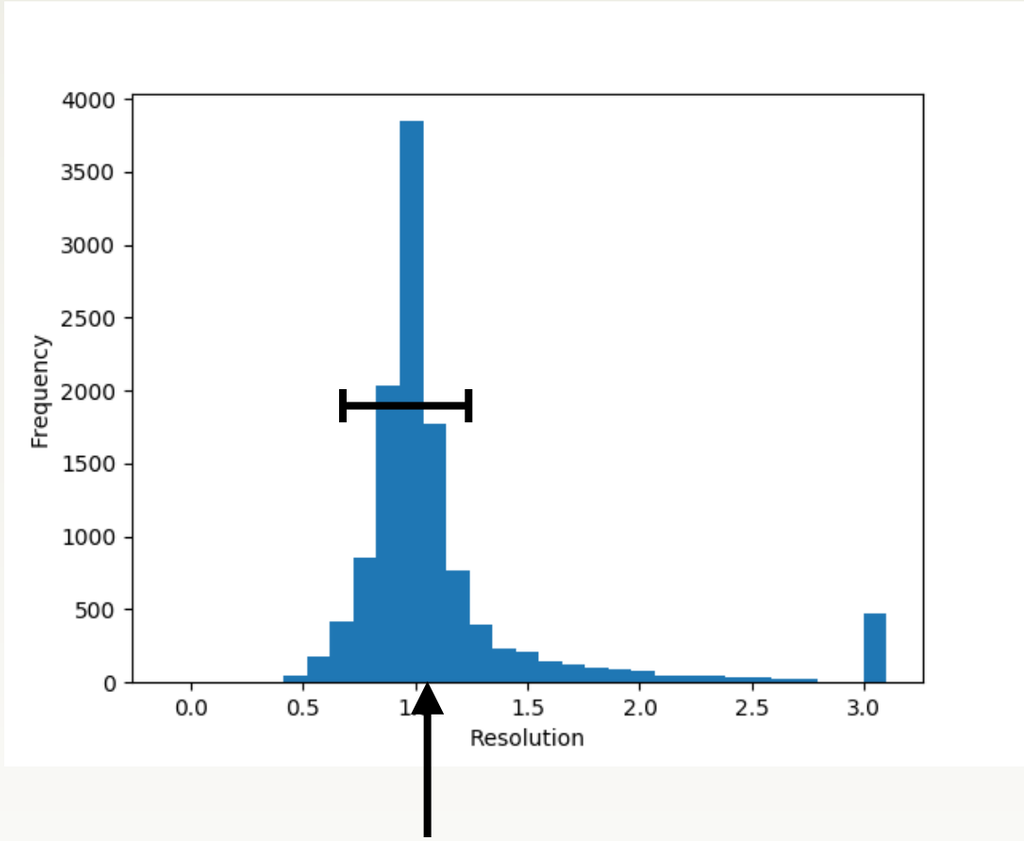
$$v = \frac{1}{N}\sum_{i}^{N} (r'_i - m)^2$$

Variance from 1:

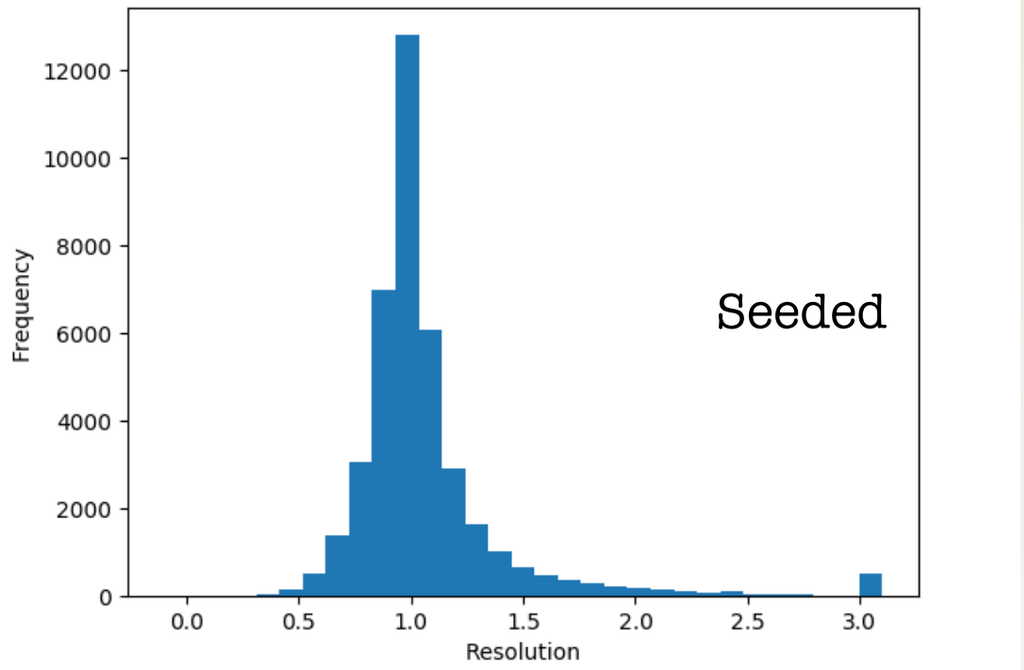
$$v' = \frac{1}{N}\sum_{i}^{N} (r'_i - 1)^2$$

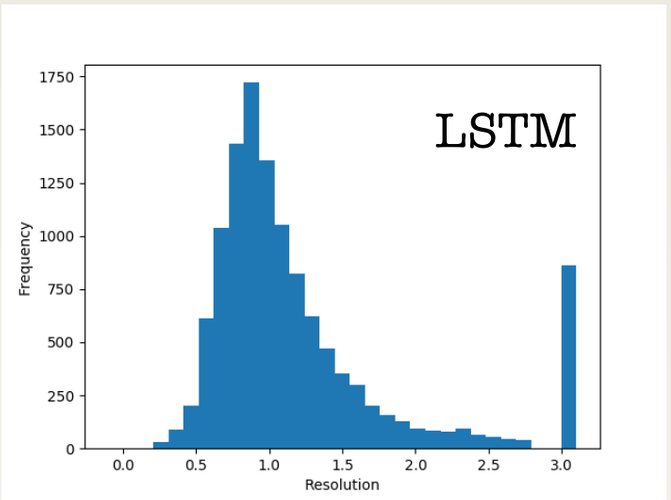
Missed showers fraction:

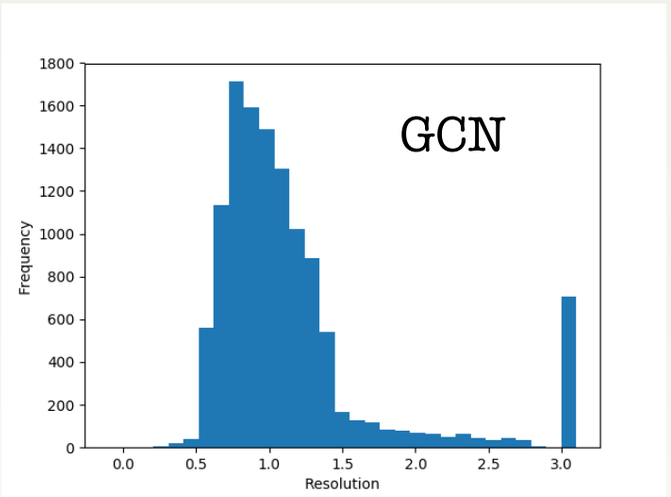
$$q = \frac{|\{r'_i > 1.3\} \cup \{r'_i < 0.7\}|}{|\{\boldsymbol{r}'\}|}$$

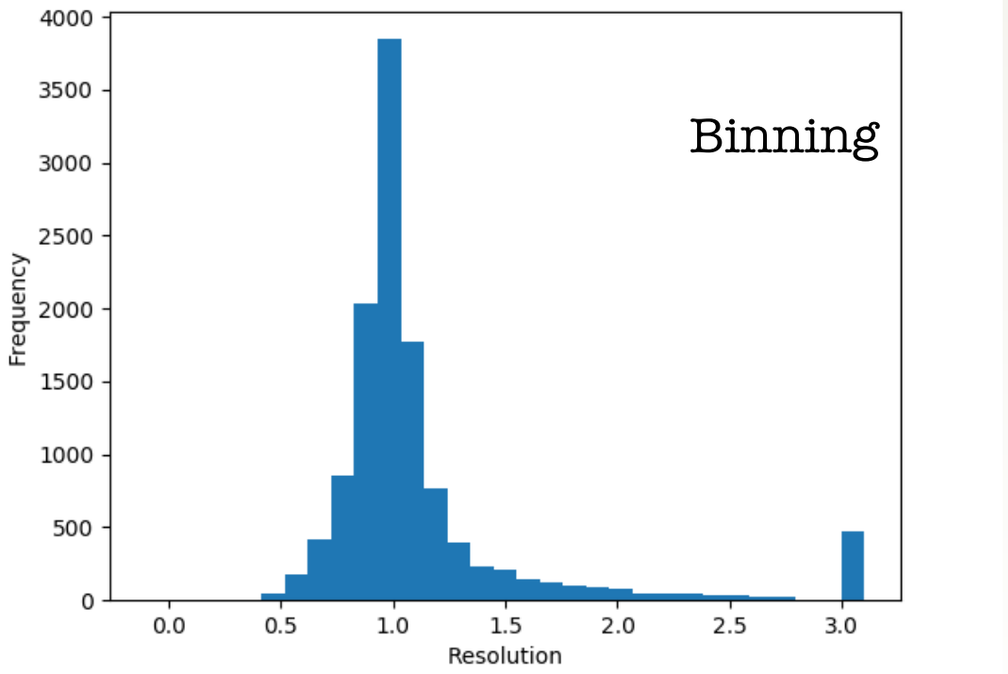


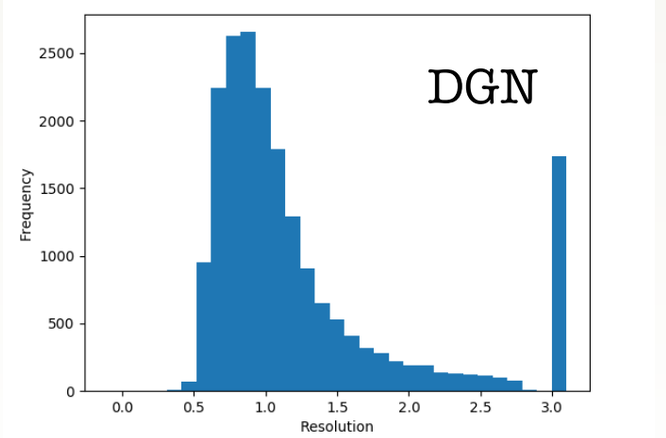
Response histograms



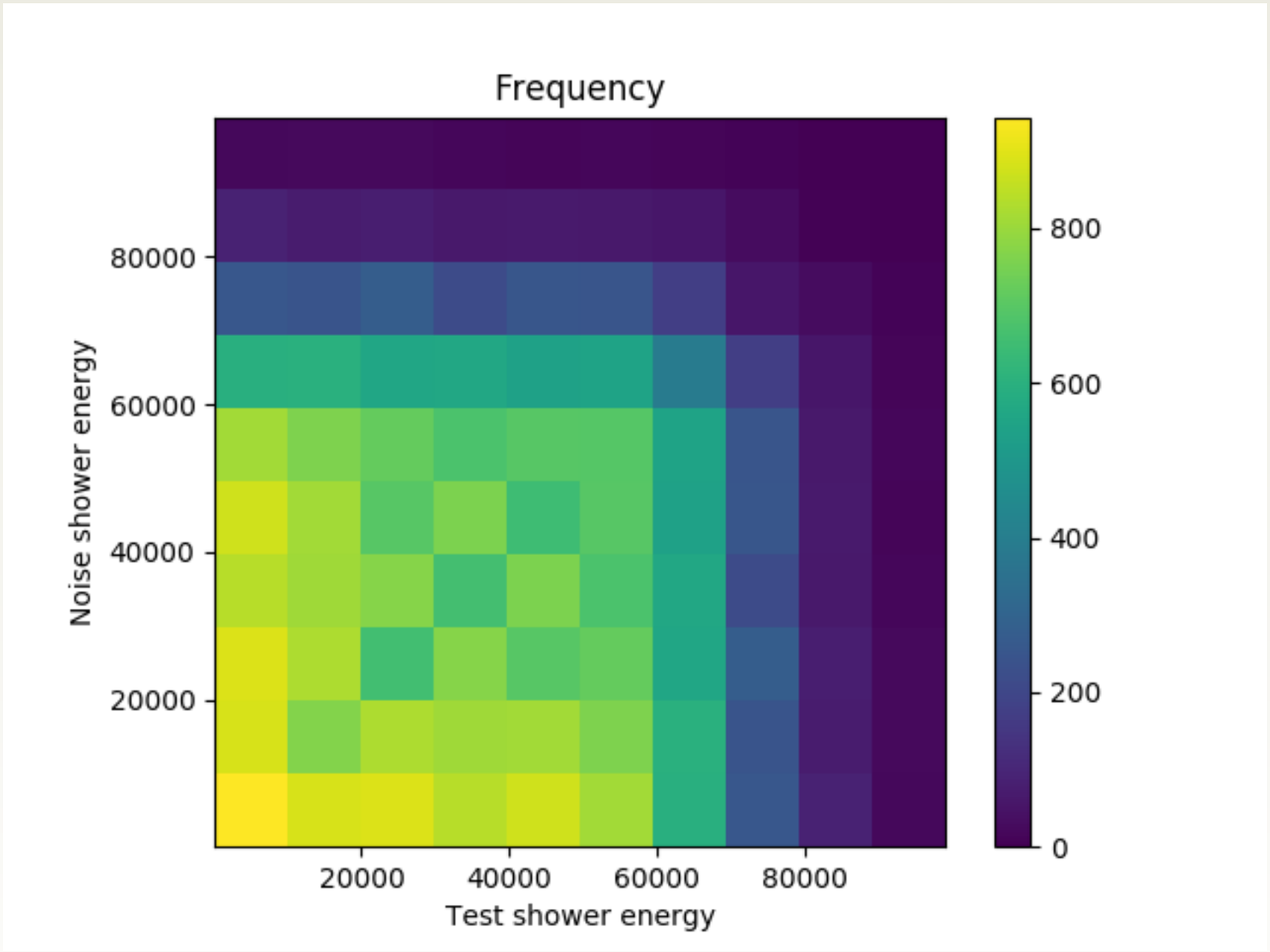




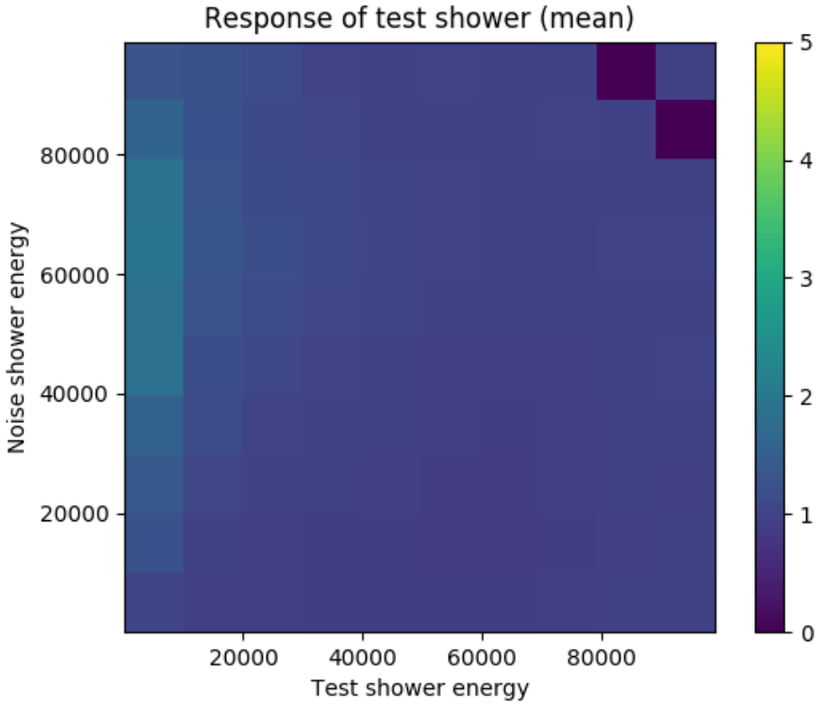




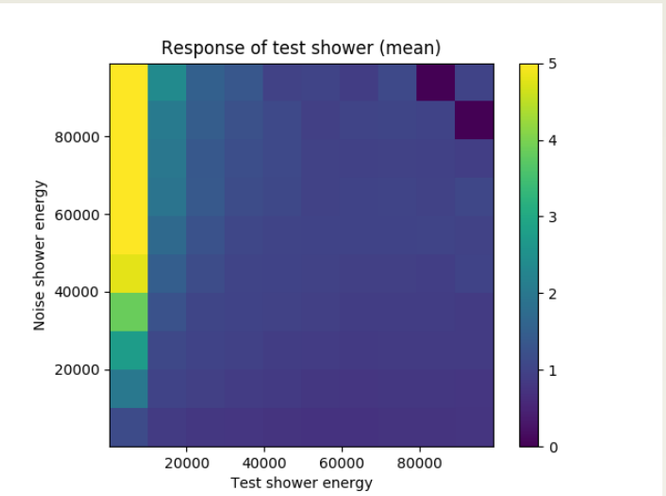
Frequency distribution



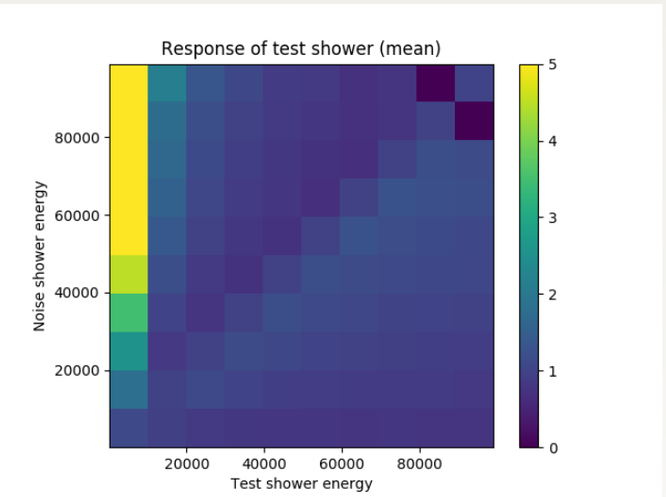
Response (mean)



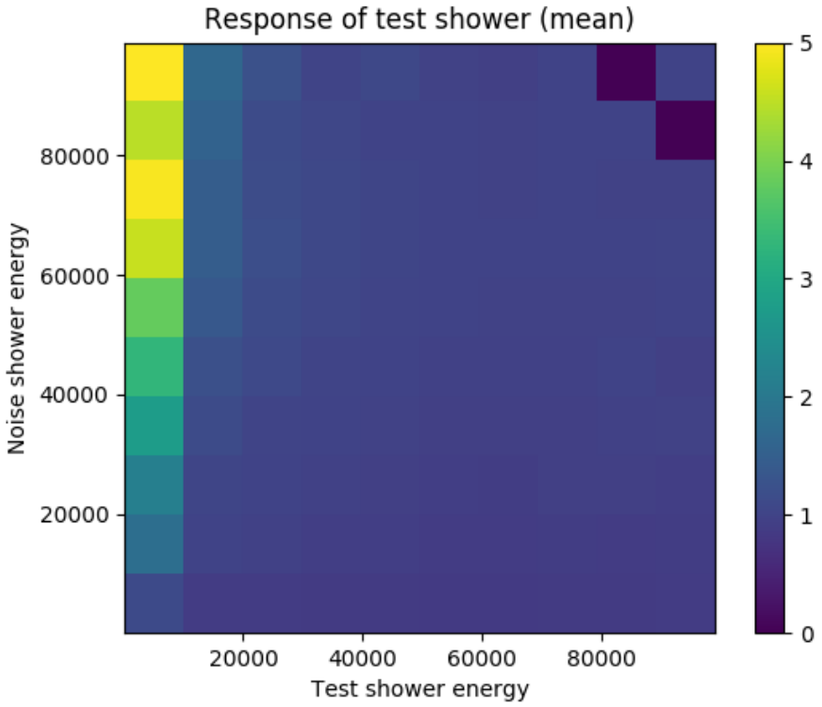
Seeded



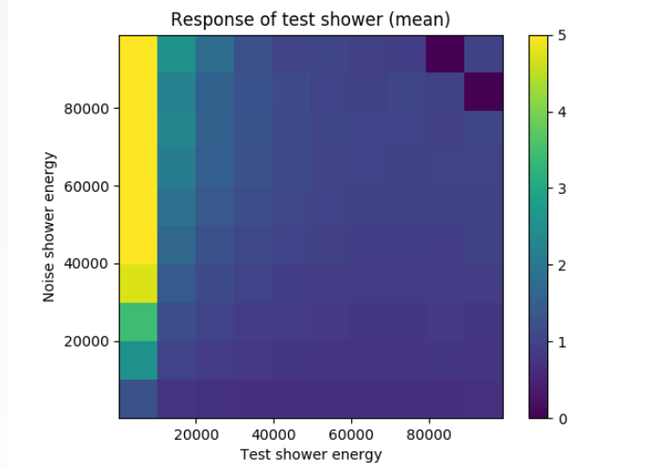
LSTM



GCN

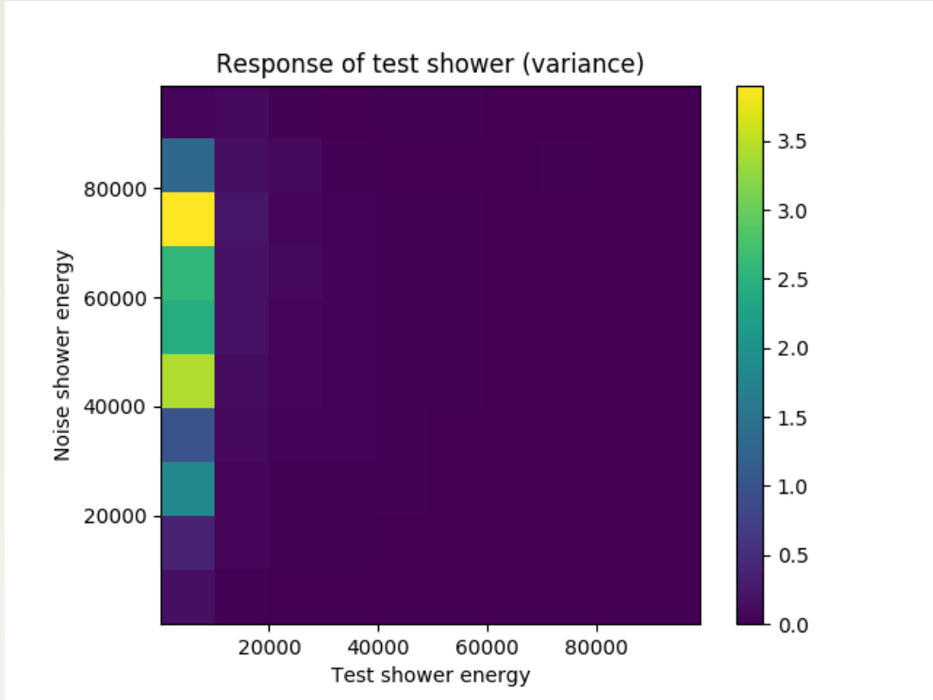


Binning

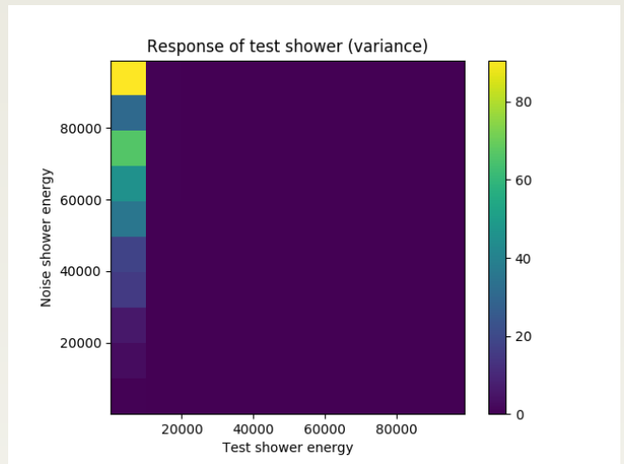


DGN

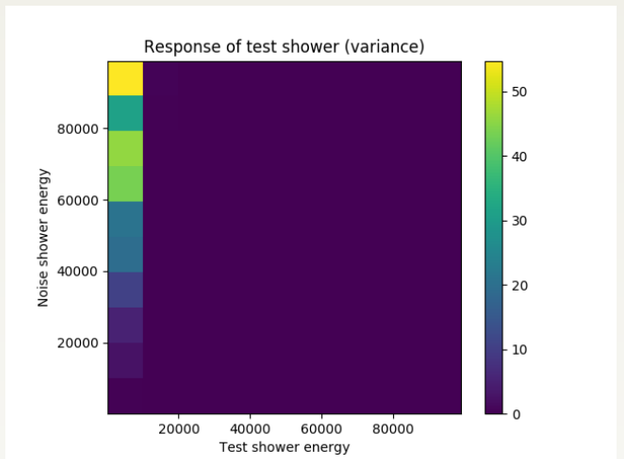
Response (variance)



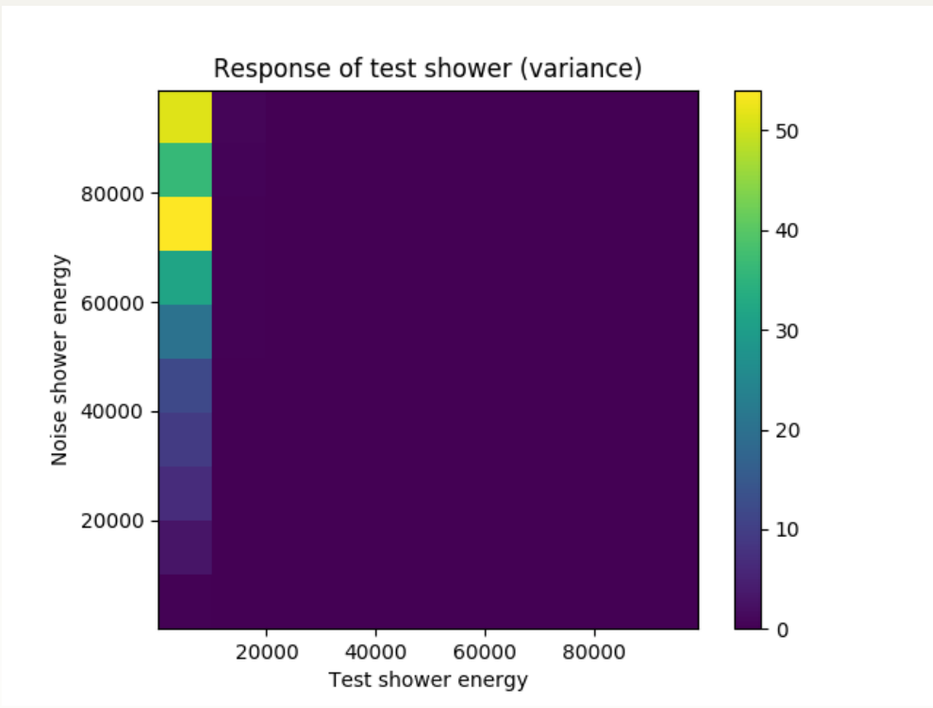
Seeded



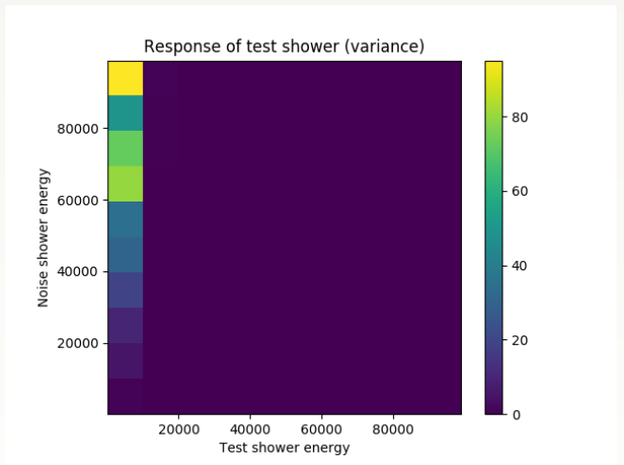
LSTM



GCN

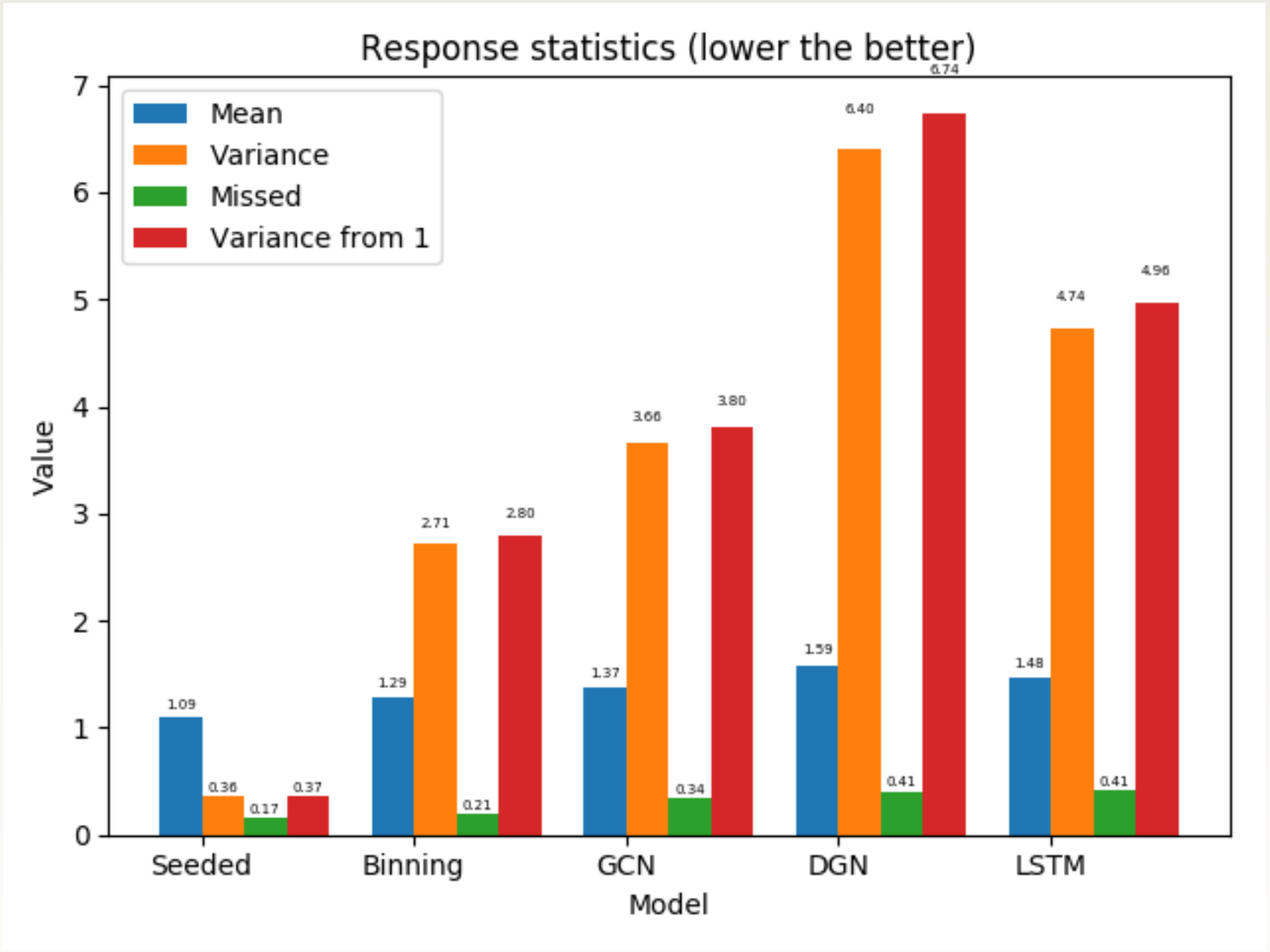


Binning

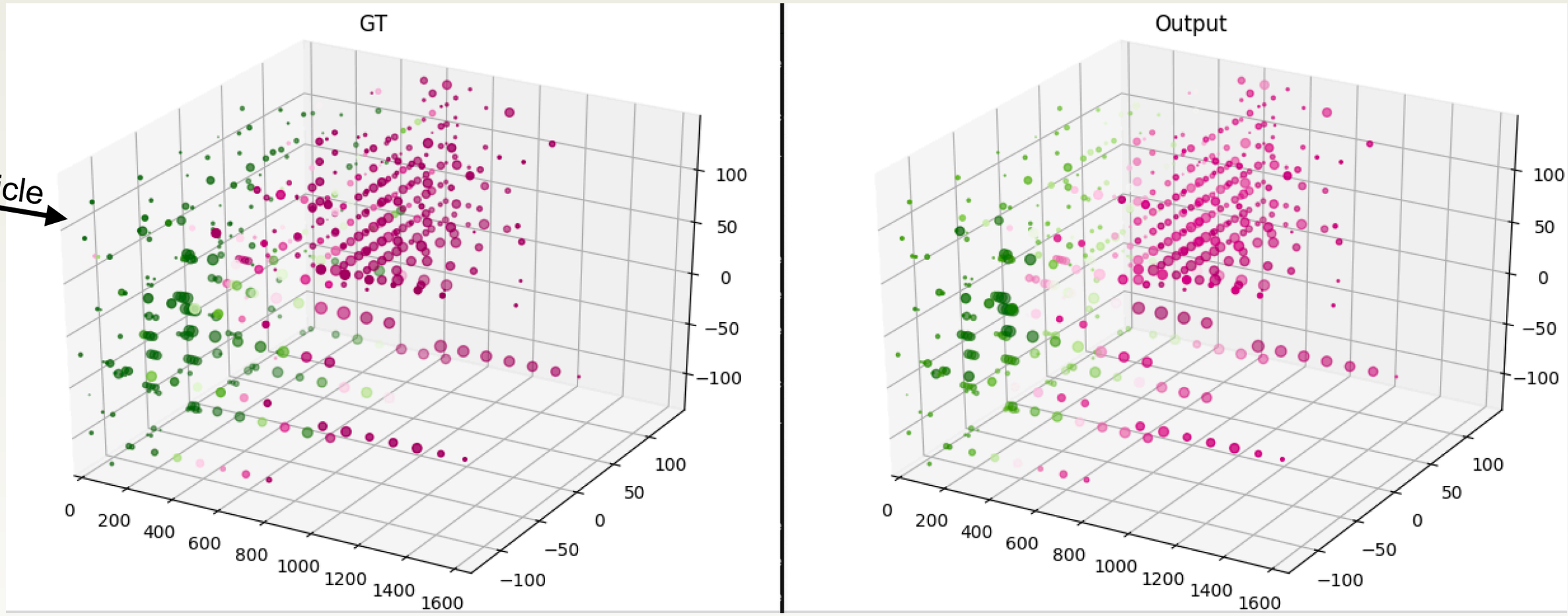


DGN

Results - Summarized



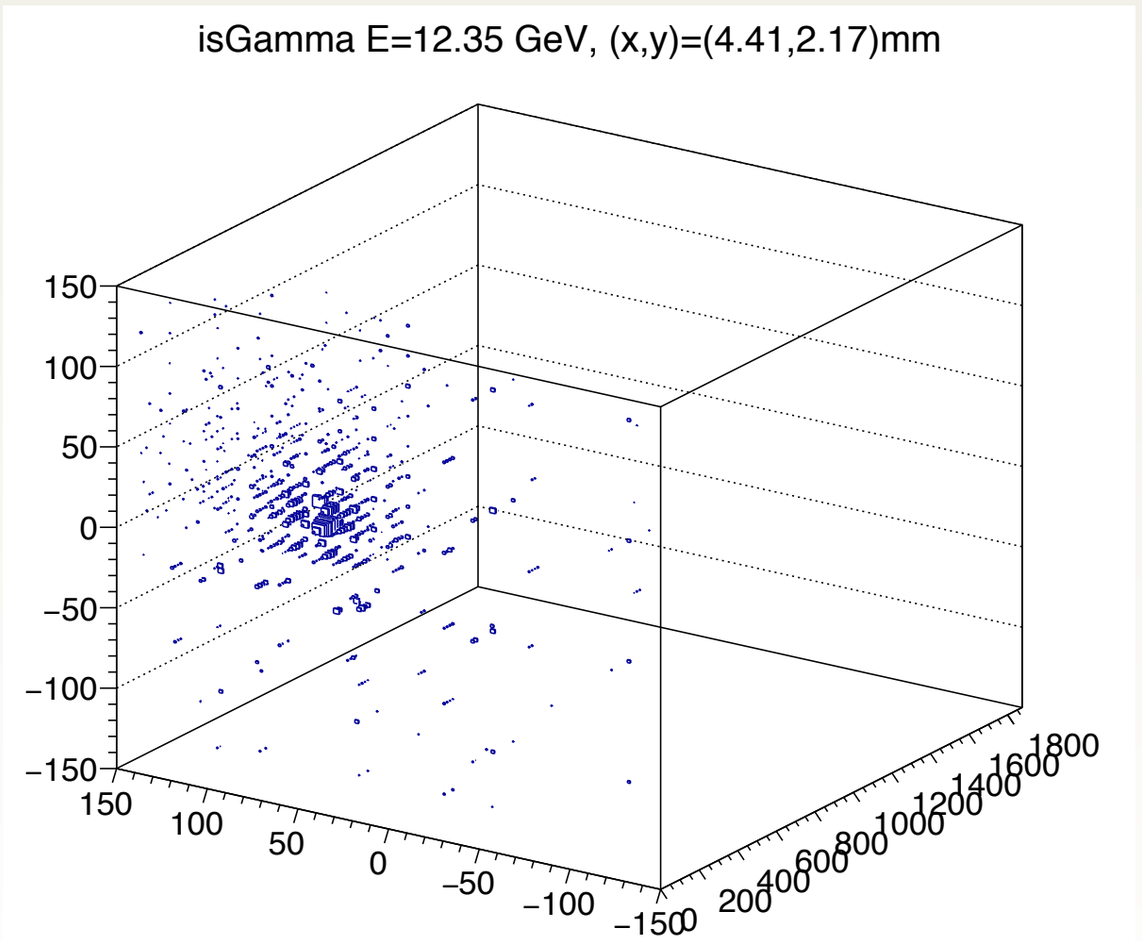
Sample result

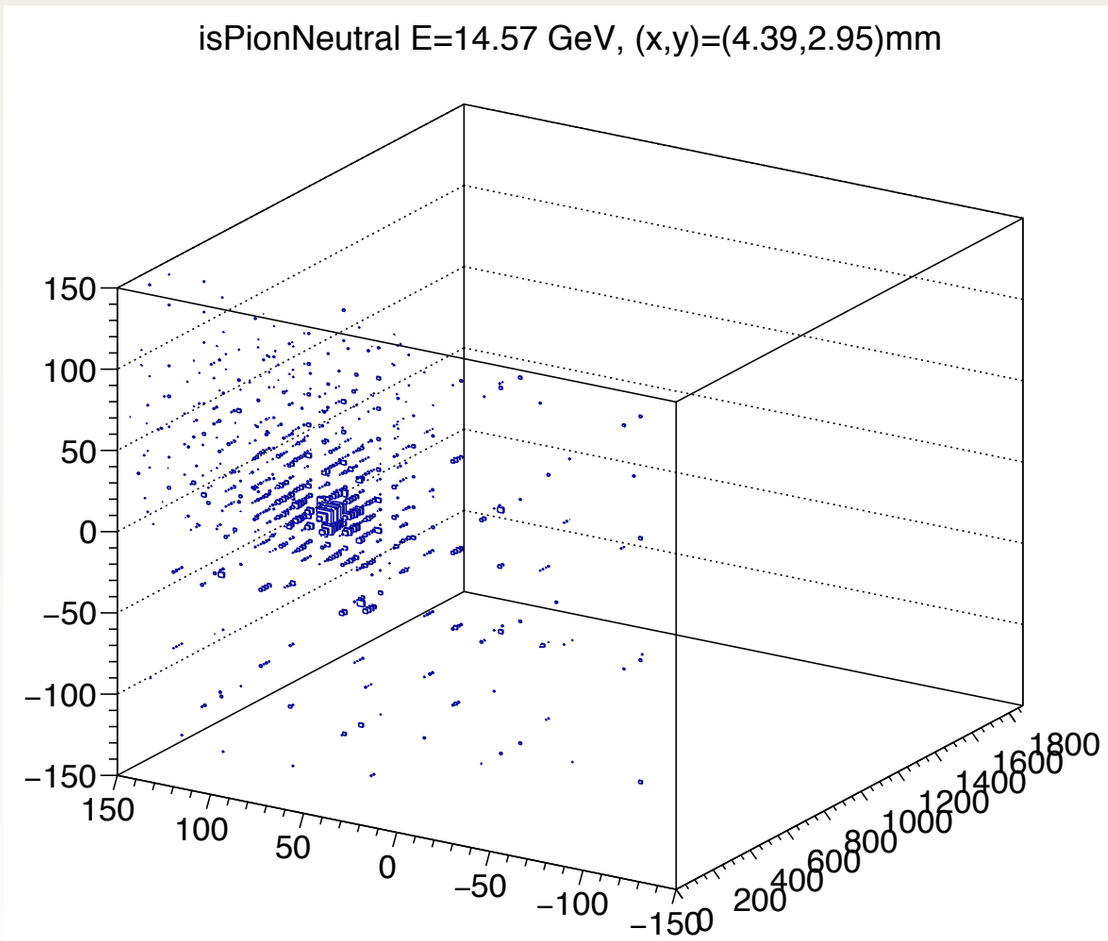


Particle ID

Focus on the most difficult ID problem: γ / π^0

Setup: Single particle, uniform energy distribution 1-100 GeV, uniform position distribution [-5, 5] cm × [-5, 5] cm





Low-energy events in x>0, y>0 quadrature

Starting point - binning method

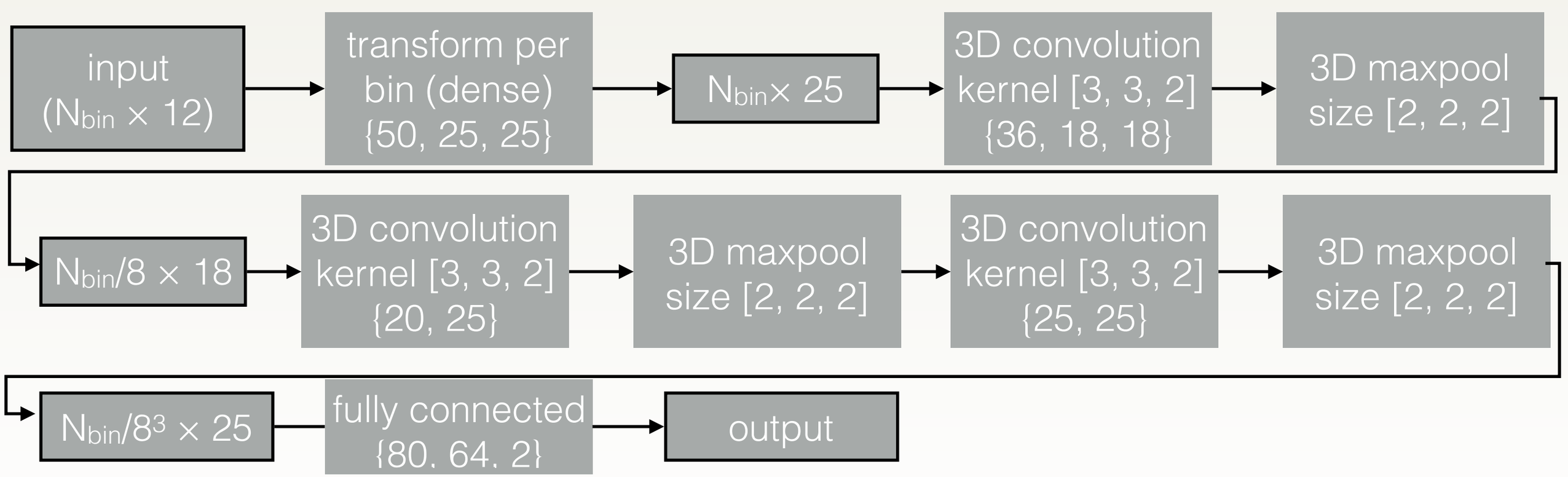
Bin the input space into a grid of 16×16×25

- Bin size 18.75mm in x and y, by layer in z

Each bin contains up to 4 rechits

Bin features: [energy, dx, dy] × 4
dx, dy: relative x and y positions of the hits from the bin center

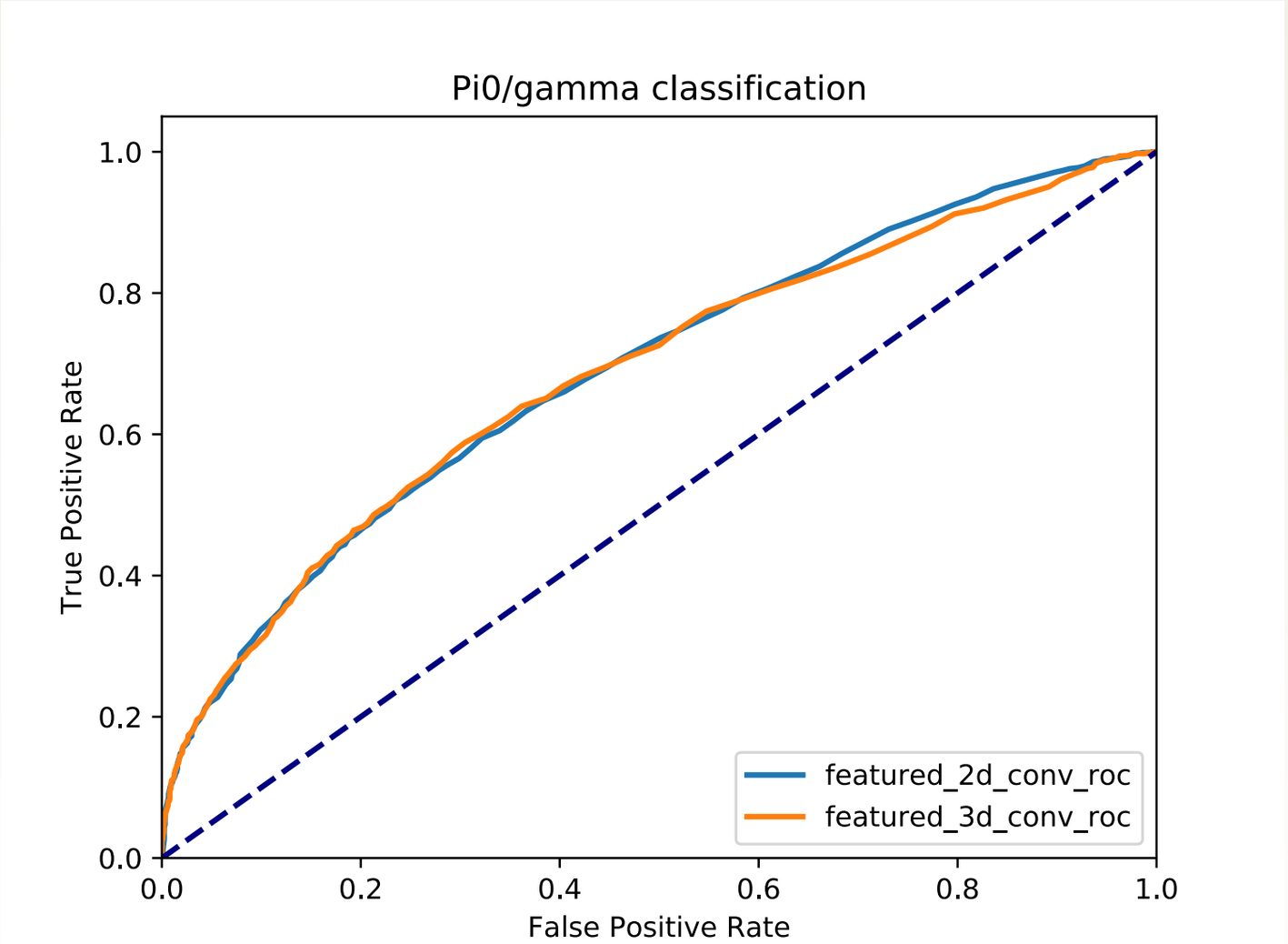
Architecture:



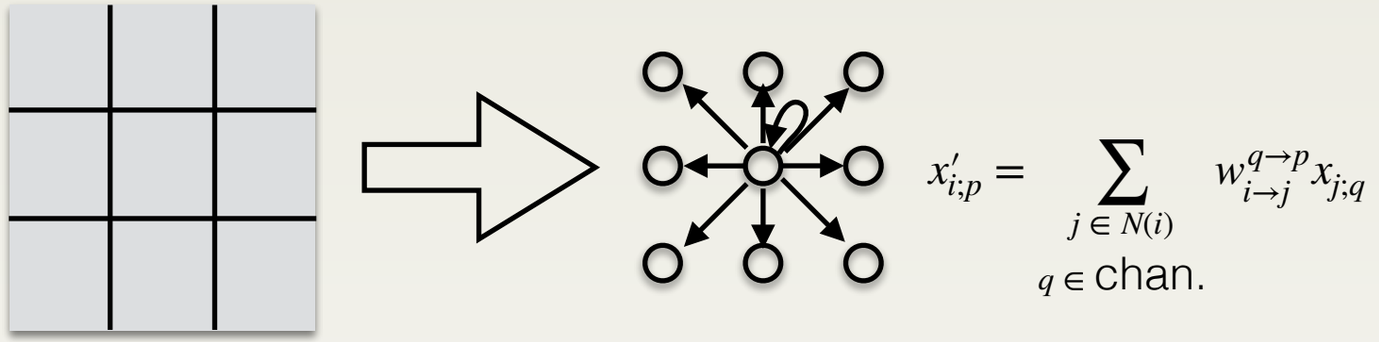
Binning method performance

ROC area-under-curve 0.68

~Identical performance with an alternative architecture based on 2D convolution in xy × dense layer in z

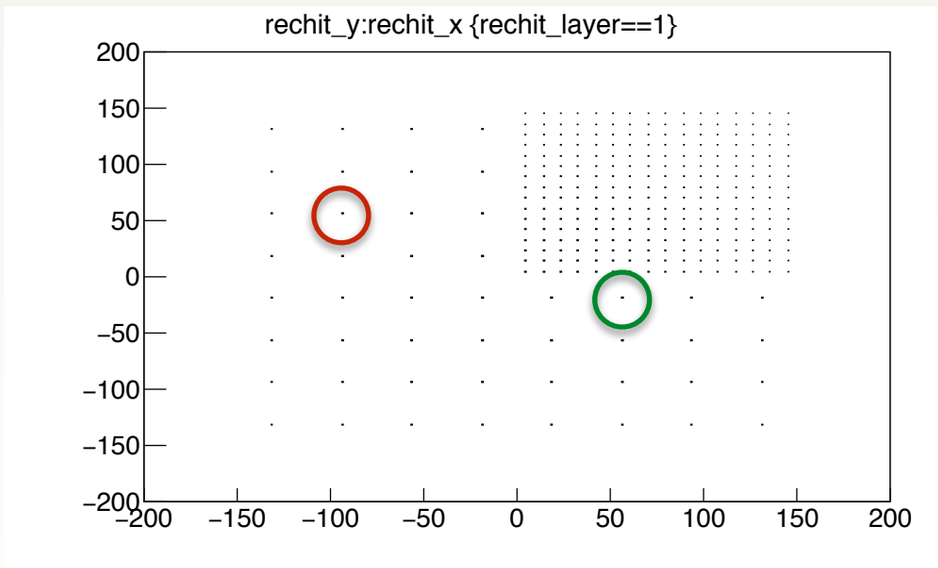


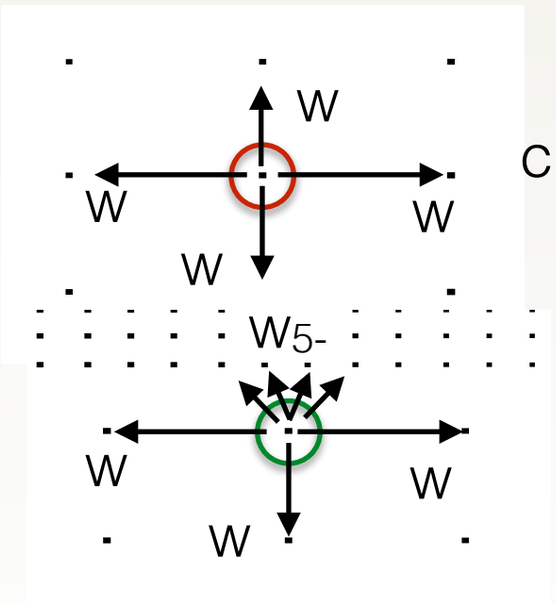
Convolution is a weighted sum over edges of directed graphs



$$x'_{i;p} = \sum_{\substack{j \in N(i) \\ q \in \text{chan.}}} w_{i \to j}^{q \to p} x_{j;q}$$

Generalize: multiple classes of directed graphs from the knowledge of the geometry (weights common within class)



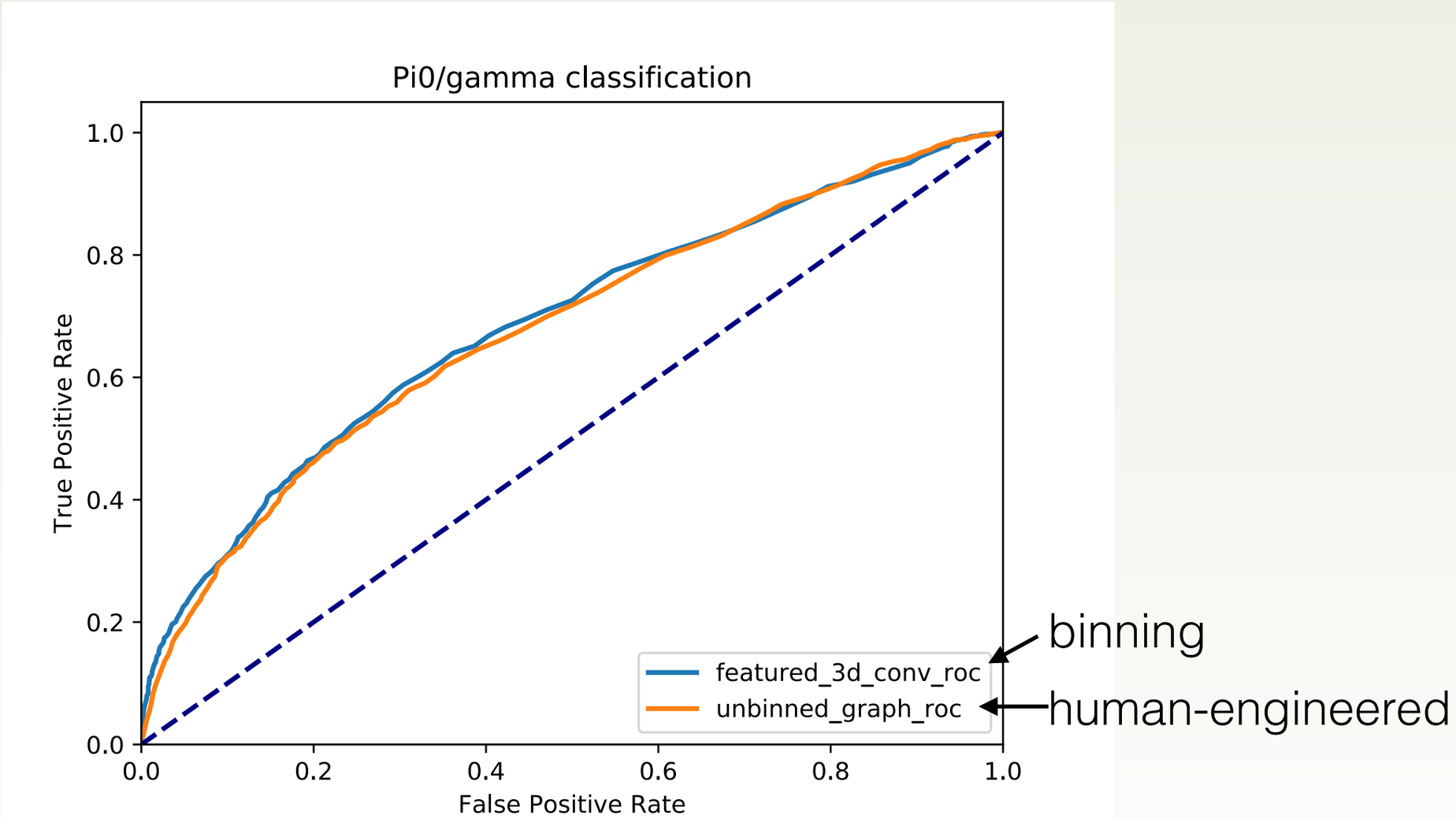


class 1

class 2

Human-engineered adjacency performance

Still ~identical performance (ROCAUC 0.67) to binning method



Summary

DGCNN not training (yet)

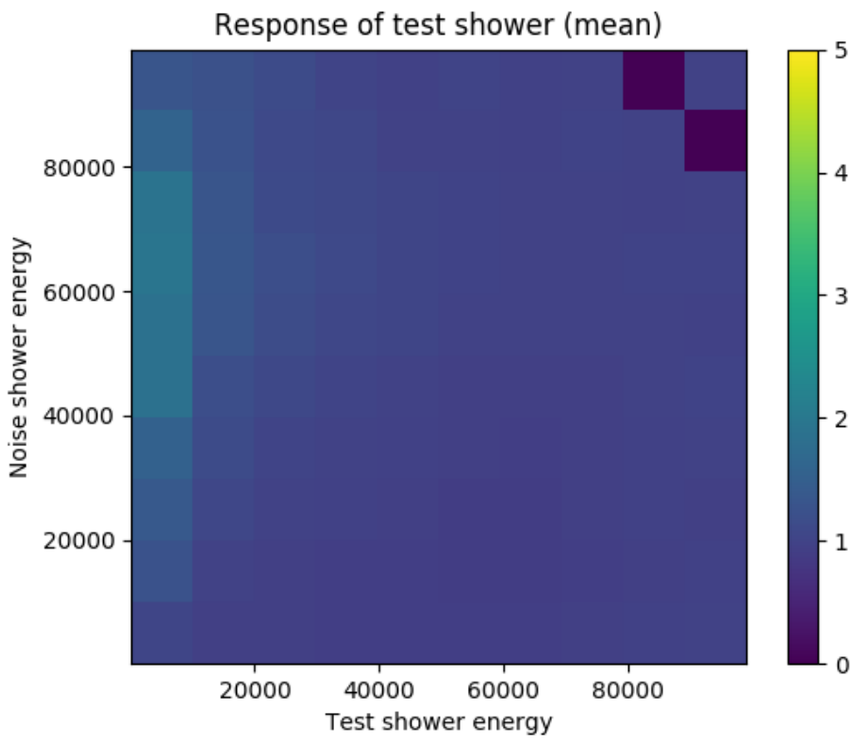
Reason under investigation

Champion: Seeded graph neural network

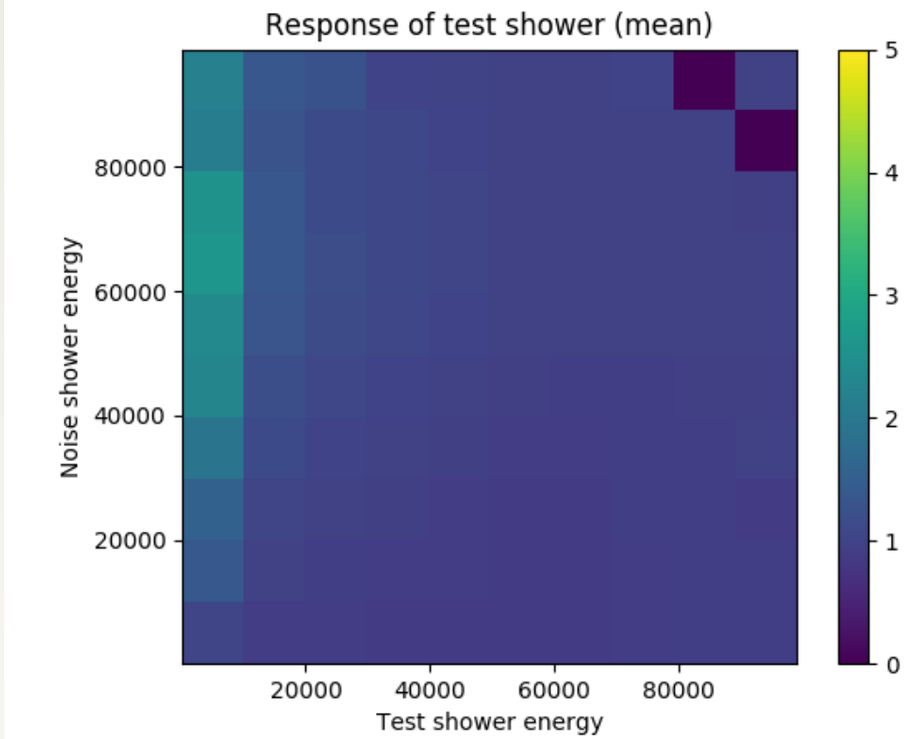
Runner up: Binning based CNN

THE END

Appendex - Comparison with a random seed

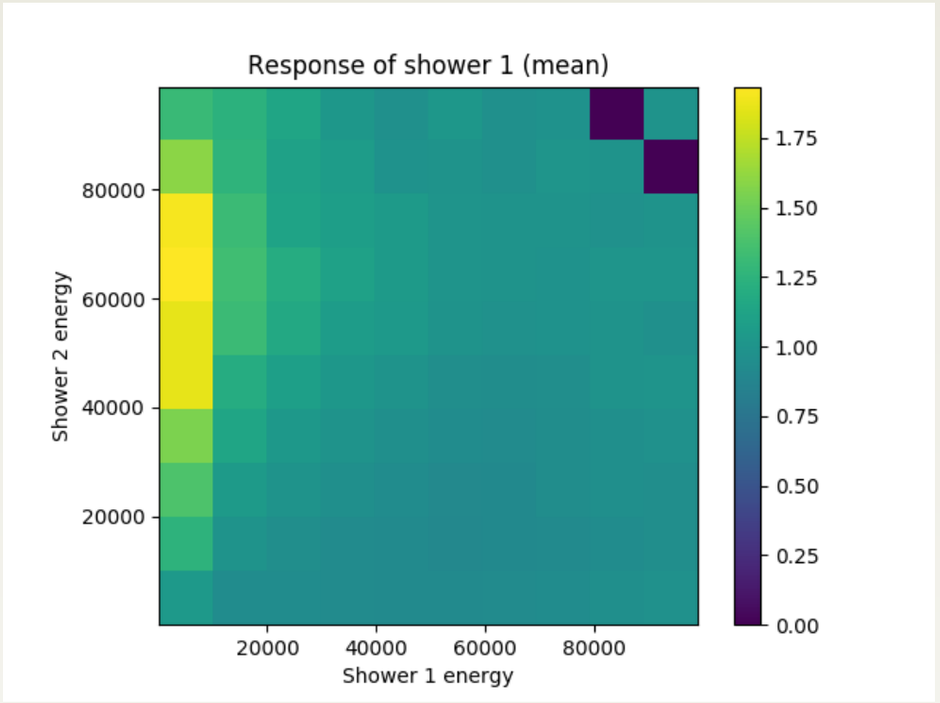


Truth seeds (2 seeds)

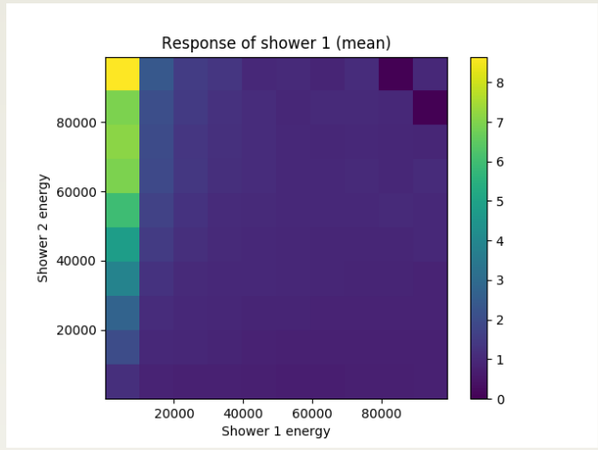


Truth seeds + one random (3 seeds)

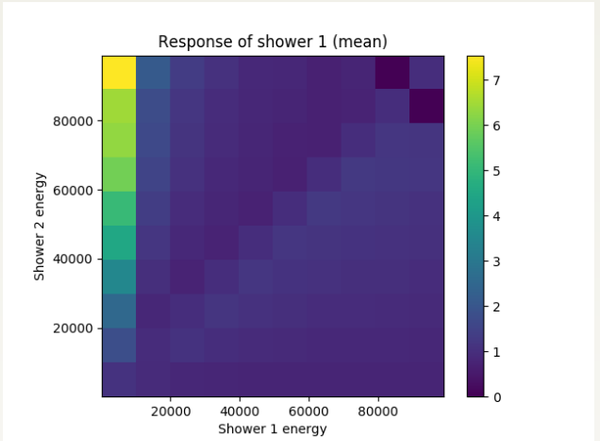
Appendix - Different scales response



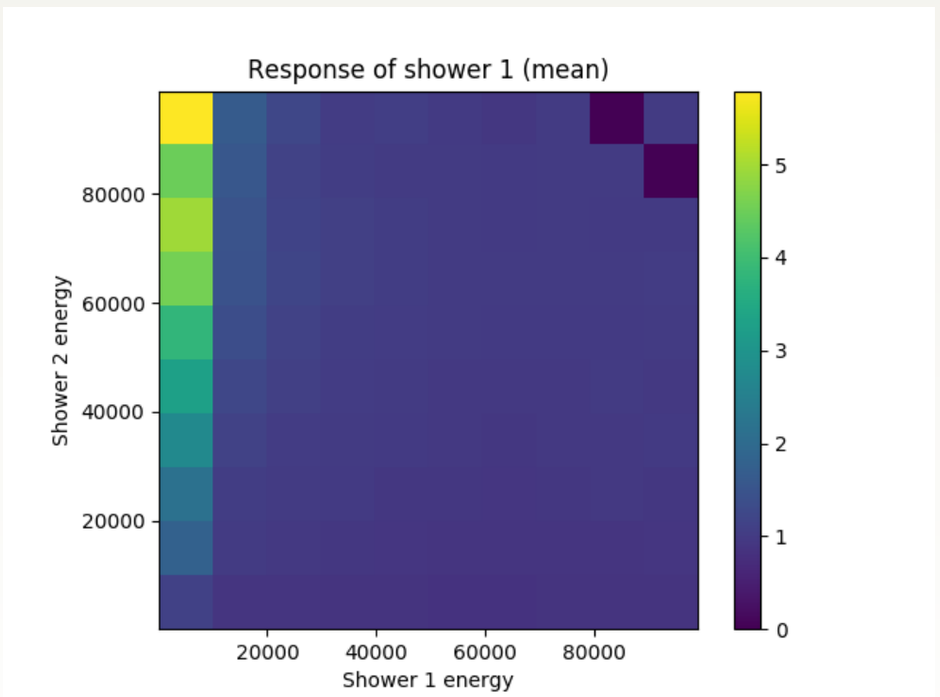
Seeded



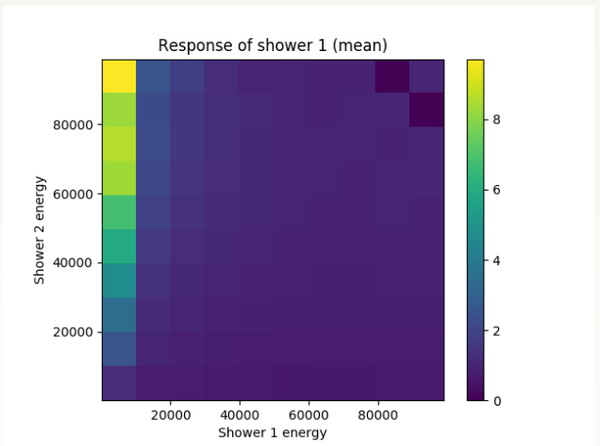
LSTM



GCN



Binning



DGN