Learning representations of irregular particle-detector geometry with distance-weighted graph networks

Jan Kieseler[1],
Shah Rukh Qasim[2], Yutaro Iiyama[1], Maurizio Pierini[1]

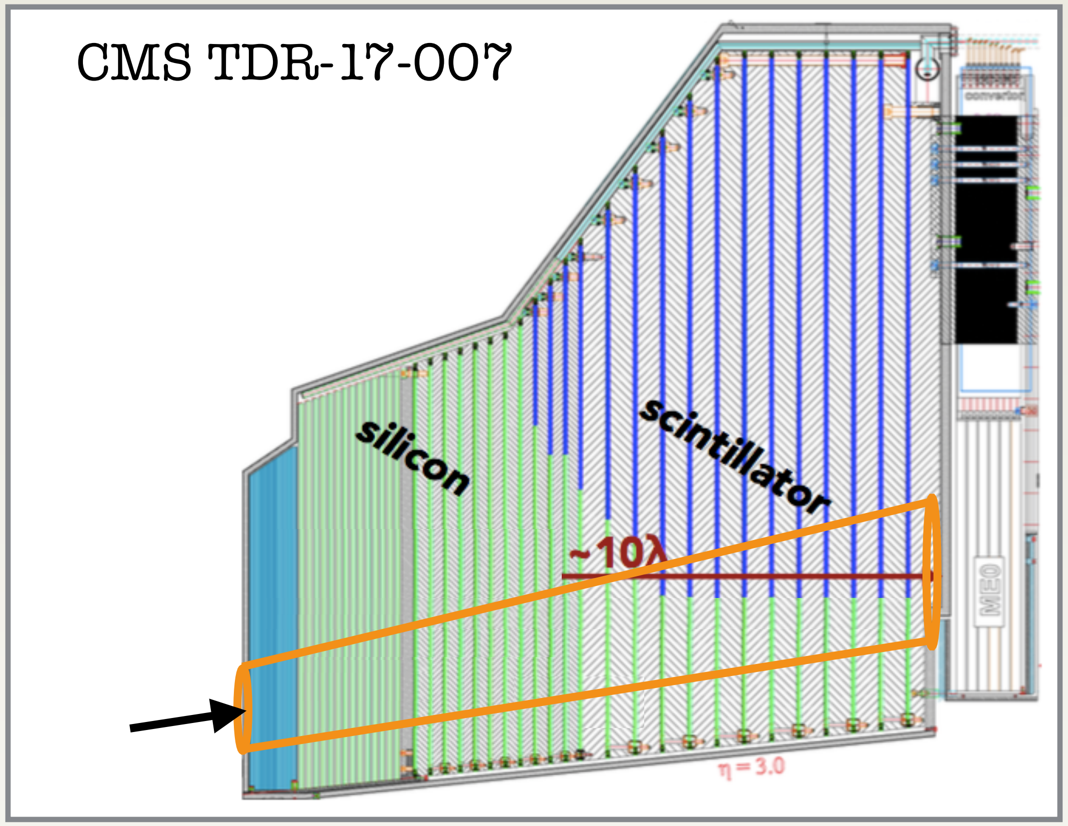
17.4.2019

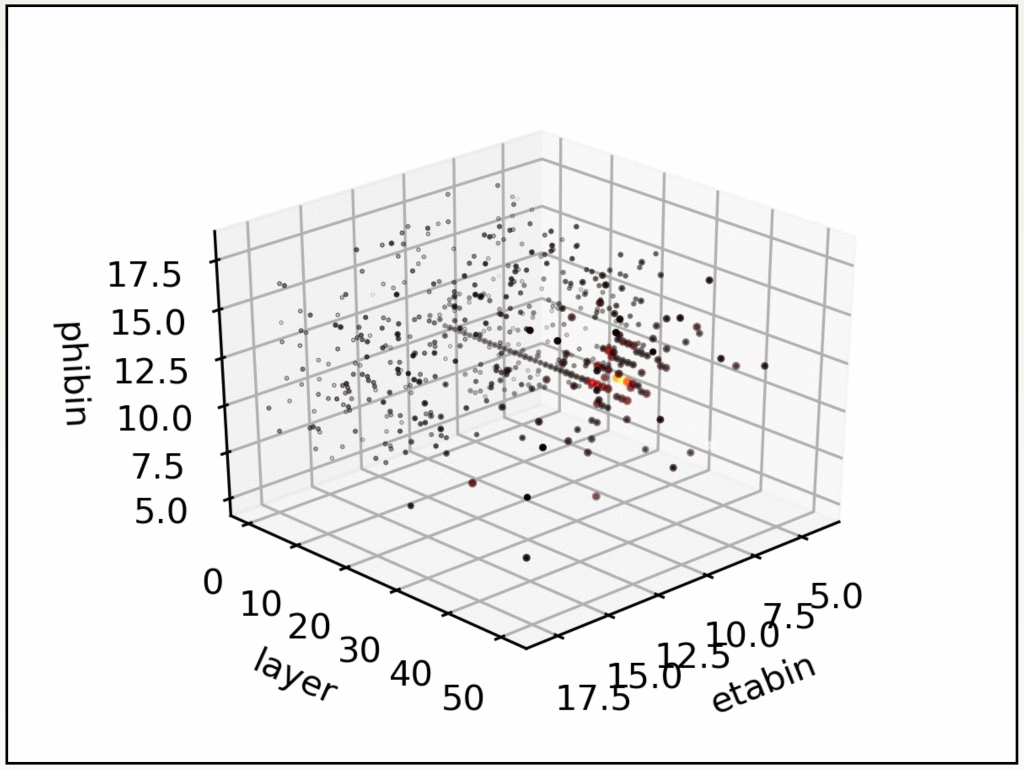
1: CERN
2: National University of Sciences and Technology, Islamabad

High granularity detectors and Computer Vision

- Example: HGCal produces 3D shower images
  - Space
  - Energy (+time) as colour

- Large number of inputs: 6M channels



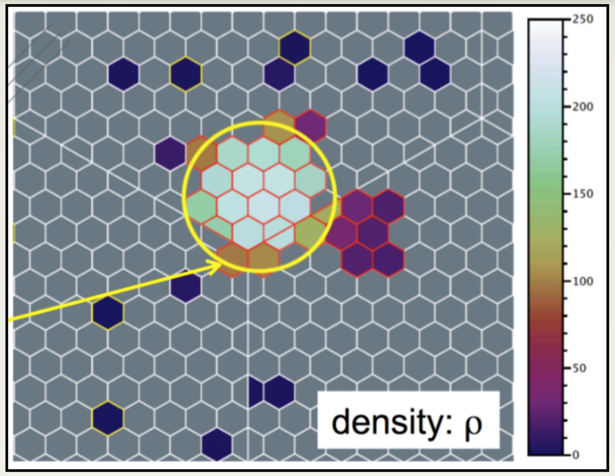


- Tasks:
  - Identify showers in noise
  - Identify particle type from shower shape
  - Measure energy

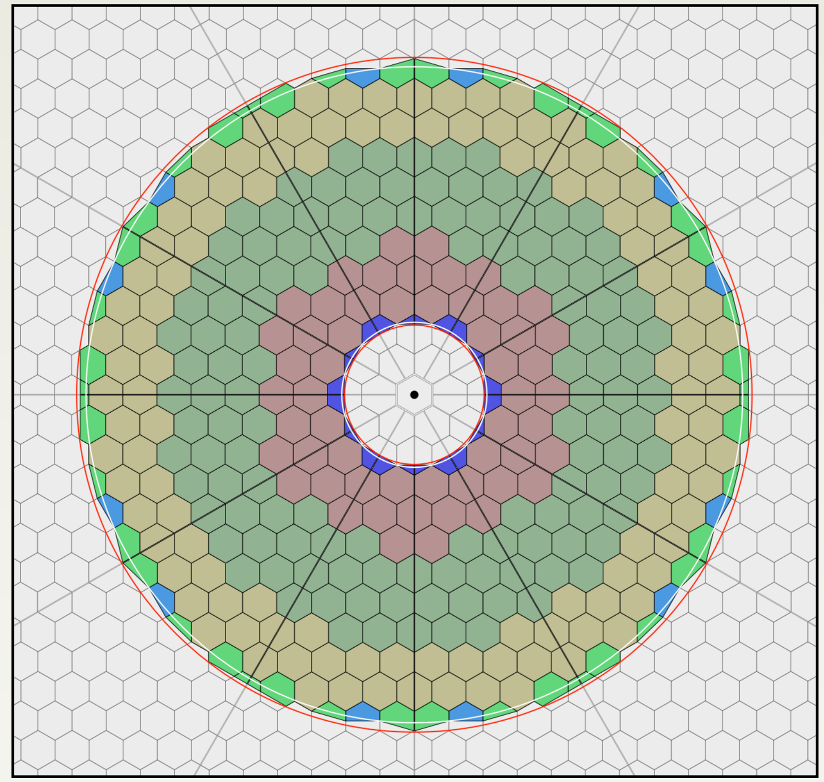
strong similarity to pattern recognition/ computer vision

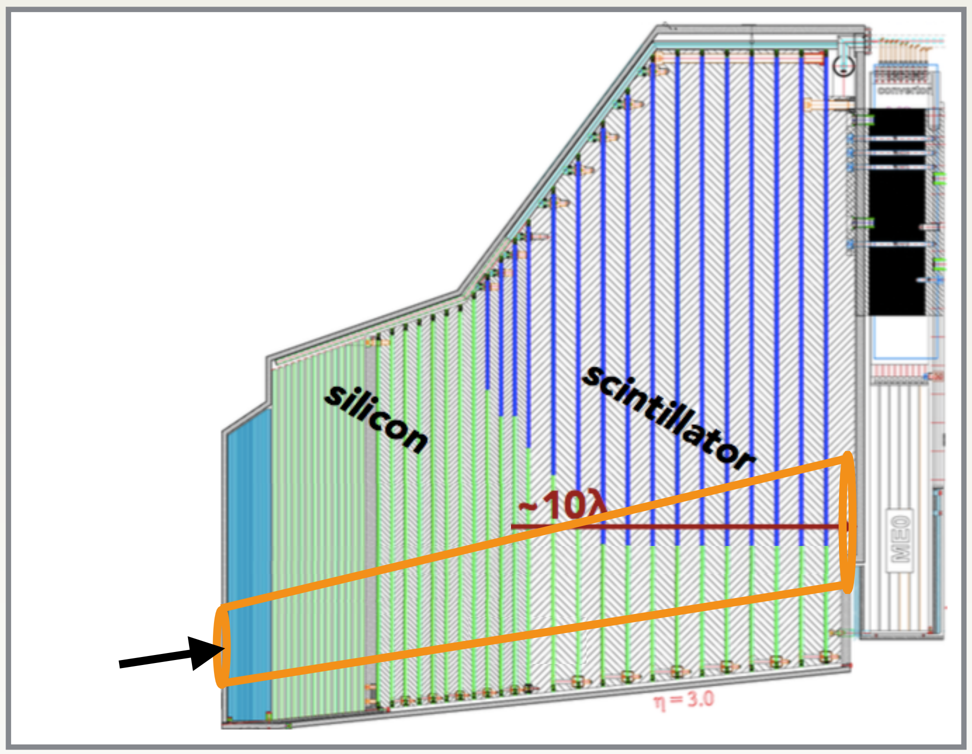
→ Using translation invariance, CNNs seem natural choise

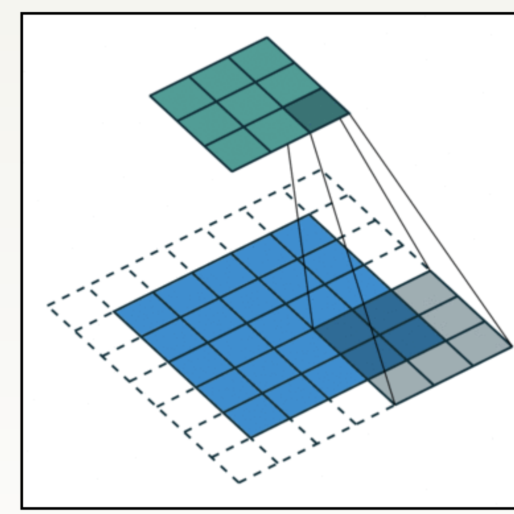
Detectors: General Problem



CMS TDR-17-007





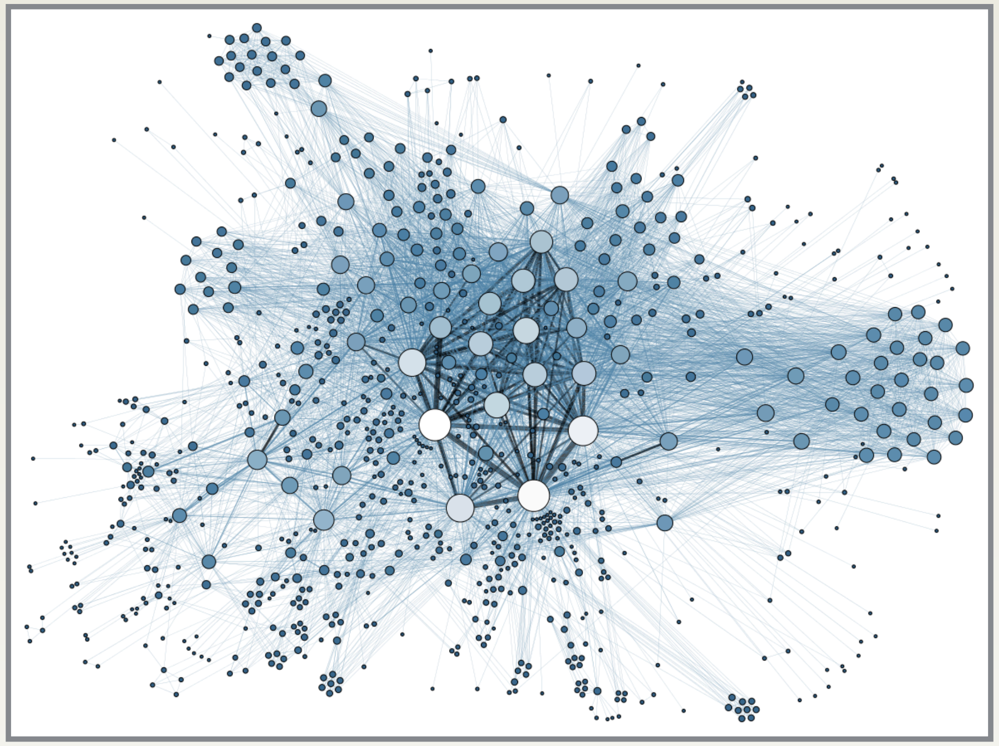


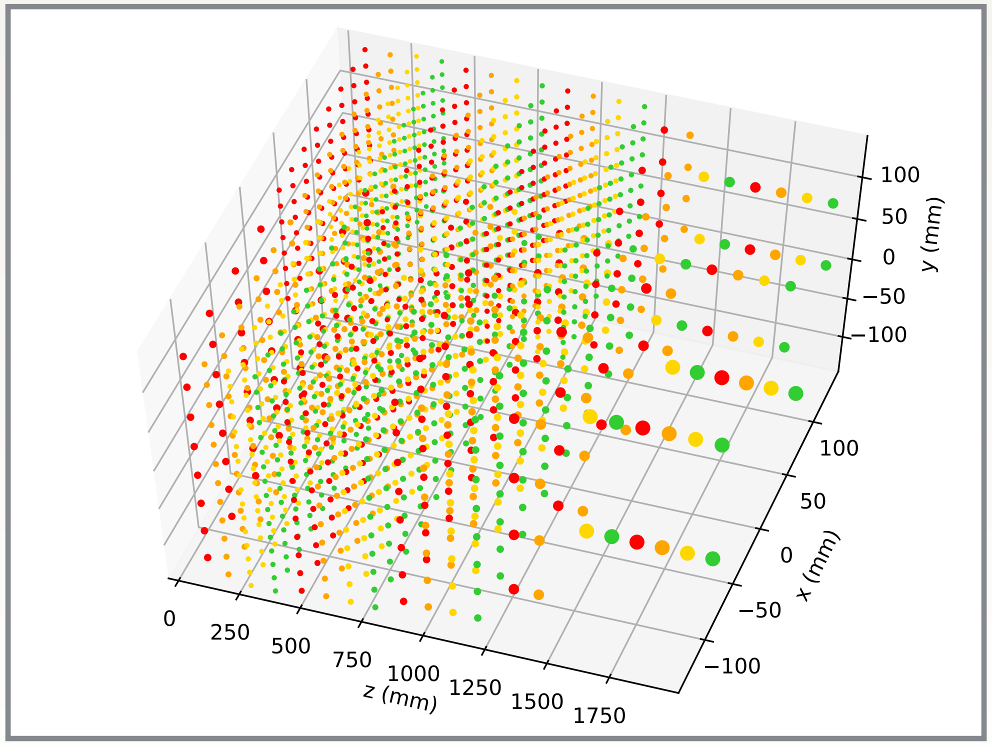
- Irregular geometry: physics driven
- Sparse showers

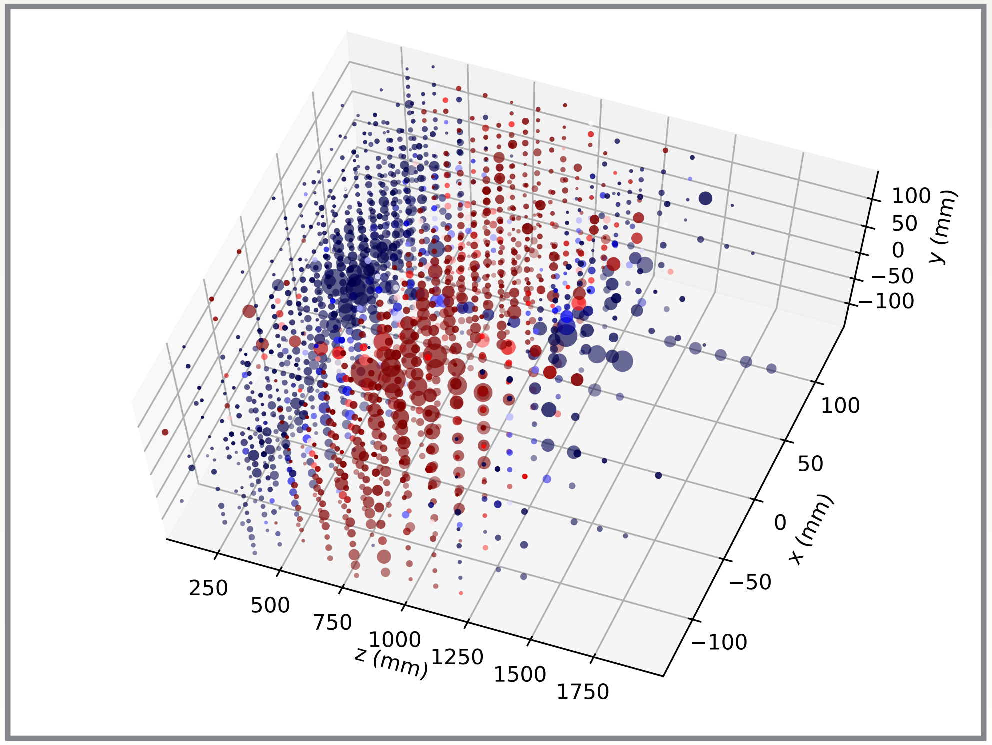
- Uniform, regular pixel size in all dimensions

Going beyond CNNs

- Using graph neural networks for reconstruction
  - Invariant w.r.t. order of inputs
  - Do not depend on a regular geometry
  - In particularly interesting: dynamic graph networks learning space transformations
- Studying approaches for segmentation (clustering)
- Here in a simplified irregular calorimeter
  - Full Tungsten, no absorber, directly consider energy deposits
  - Sensor size and quantity changes with layer and x,y

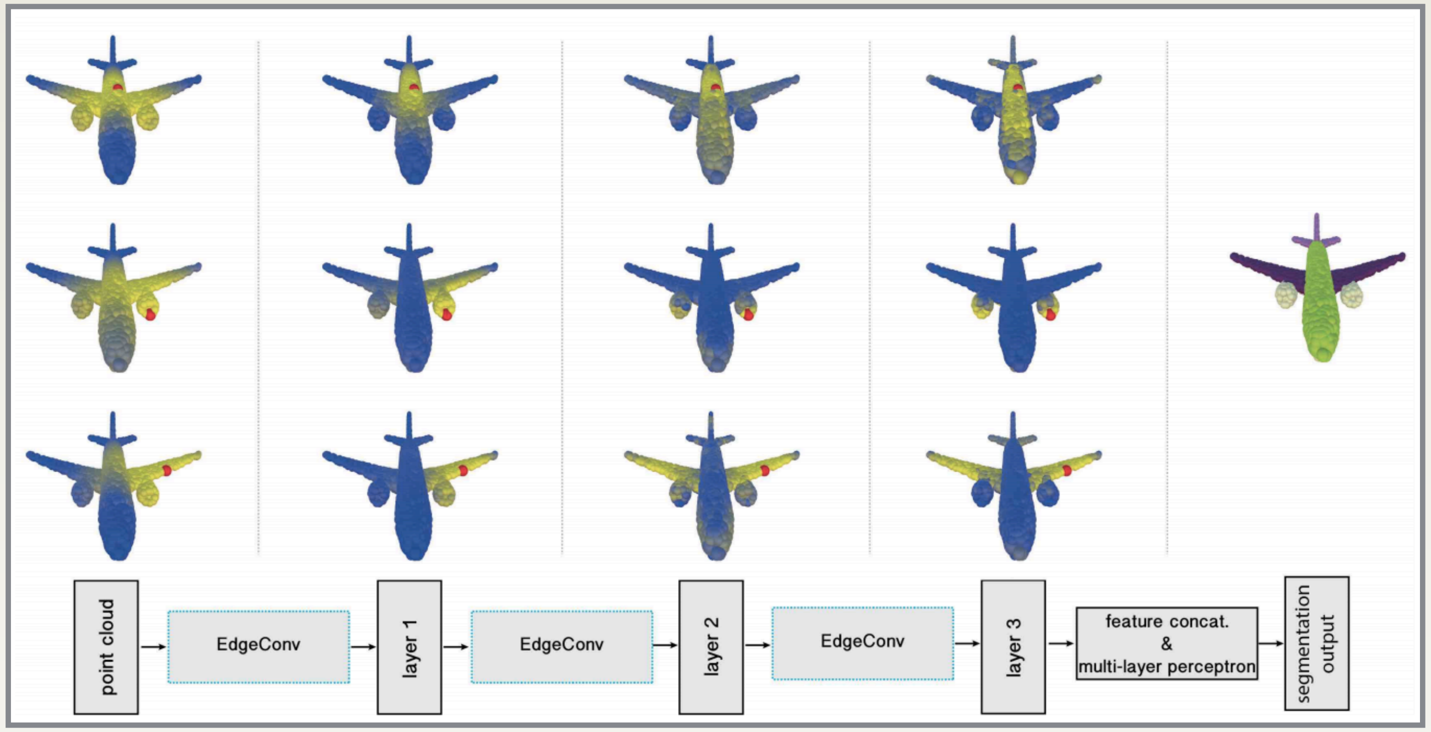




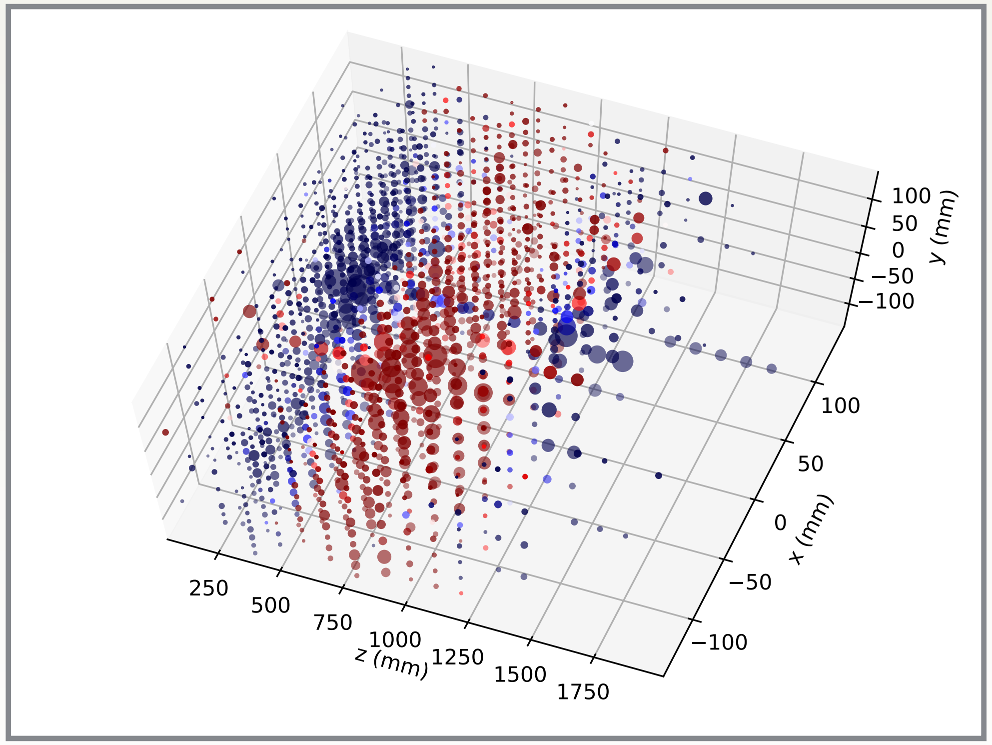


Clustering / Segmentation

- Clustering is more than (just) segmentation: need to identify fractions rather than classify a set of points
- However, proposal for segmentation of point clouds: DGCNN [1] similar to our problem
  - Irregular points / sensors
  - Identifying one shower in presence of others
  - Proven very powerful



- DGCNN (a.k.a EdgeConv layers)
  - Transform features per vertex
  - Calculate L2 distance between vertices with new features
  - Select N neighbours and edges (just difference between features, absolute coord.)
  - Transform edge features
  - Select maximum activation of transformed edge features as new vertex features (max pool in edges)
  - Propagate this information to next layer



[1] arXiv:1801.07829

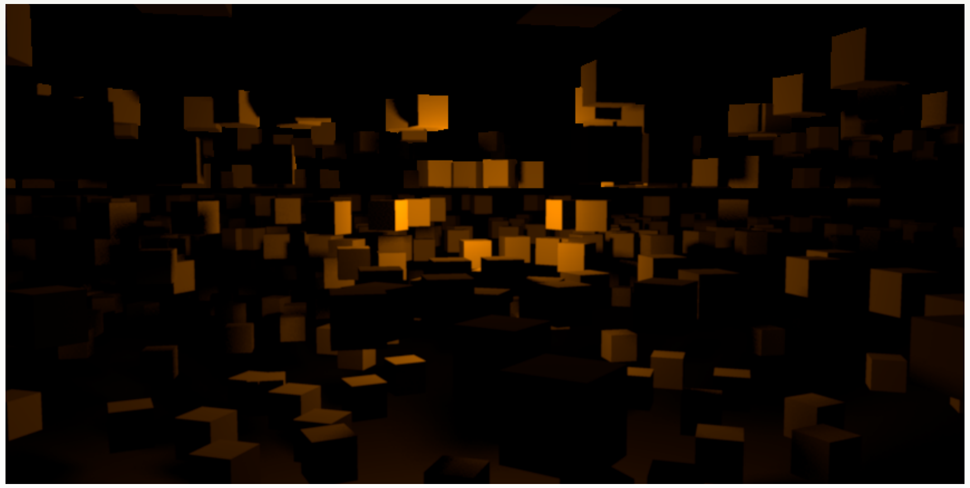
EdgeConv in practice

- DGCNN (a.k.a EdgeConv layers)
  - Transform features per vertex
  - Calculate L2 distance between vertices with new features
  - Select N neighbours and edges (just difference between features, absolute coord.)
  - Transform edge features
  - Select maximum activation of transformed edge features as new vertex features (max pool in edges)

- B x V x F → B x V x F' (F' = 64)
- **B x V x V x F'** → B x V x V x 1
- B x V x N x F'
- (**B x V x N x F' → B x V x N x F''**)^t
- B x V x N x F'' → B x V x F''

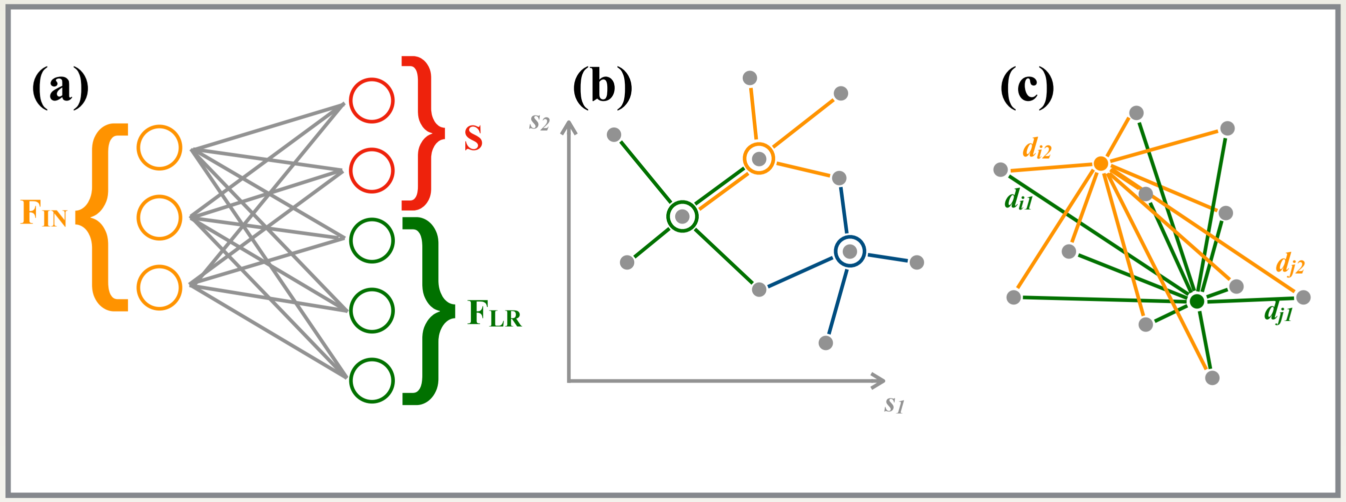
- Many resource intense steps
  - Our training/inference resources are very limited
  - We also need a fast network for triggering applications
  - Our inputs are more complex: coordinate features and also measured features (e.g. energy in a sensor)

- Build new network layers: GravNet and GarNet
  - Allow for learnable space representation
  - Split coordinate space and 'other feature' space
  - Aggregate features from vertices
  - Evaluate distances in coordinate space and apply as weights to features when aggregating
    - Creates gradient w.r.t. distances

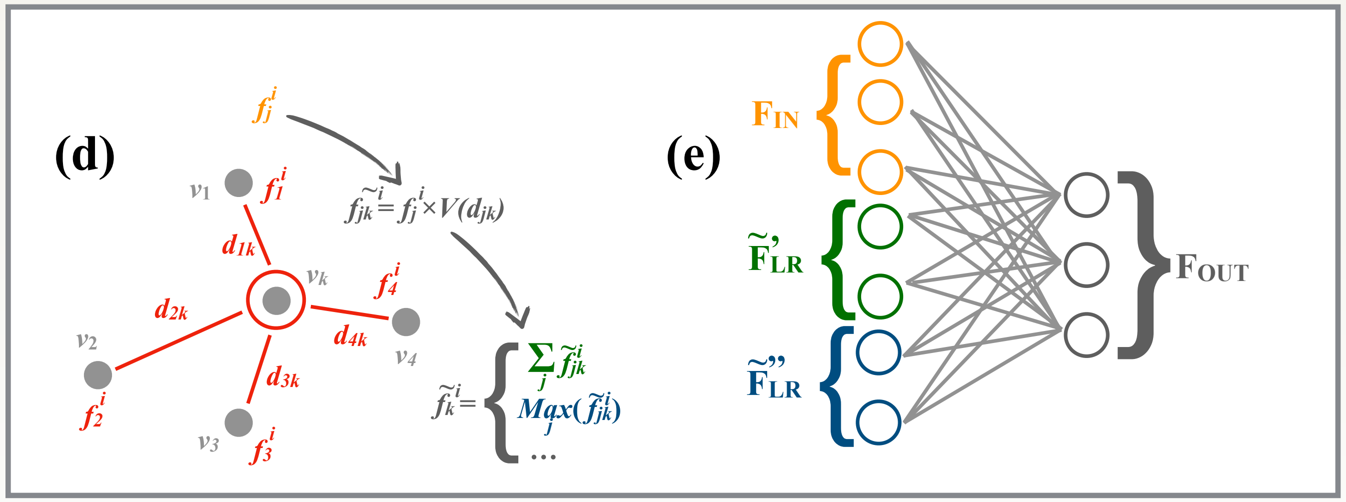


Technical Implementation

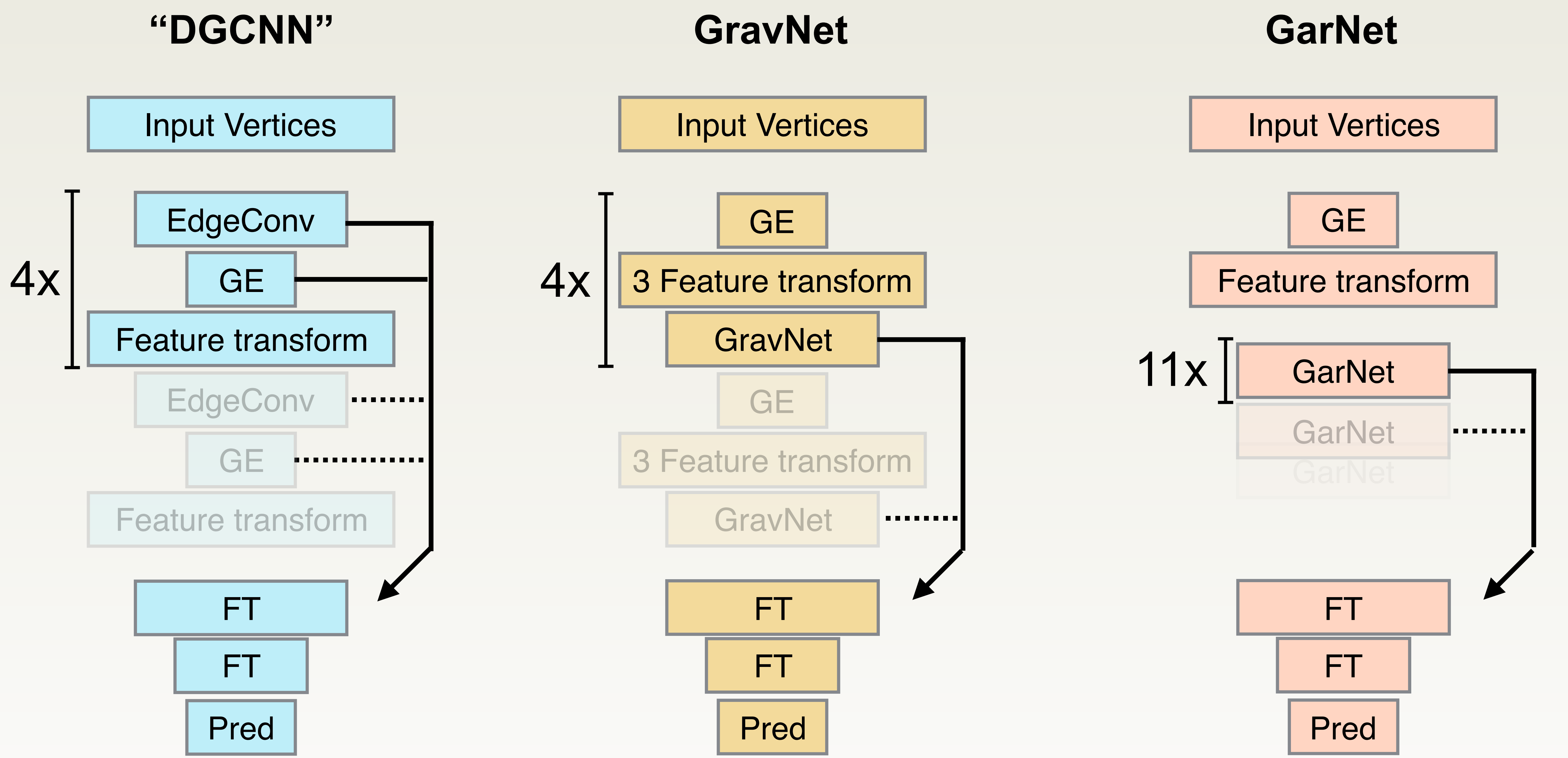
- **GravNet**
  - Project to coordinate and feature space **(a)**
  - Select N neighbours using coordinate space **(b)**
  - Scale neighbour features with distance **(d)**
    - (small distance = large weight)
  - Select maximum and mean of scaled features **(d)**
    - Improves convergence significantly
  - Mix with original vertex information **(e)**



- **GarNet**:
  - Project to coordinate and feature space **(a)**
  - Interpret coordinates as distances to aggregators **(c)**
  - Use distance weights to aggregate mean and max **(d)**
  - Expand back to all vertices using same distance weights **(d)**
  - Mix with original vertex information **(e)**



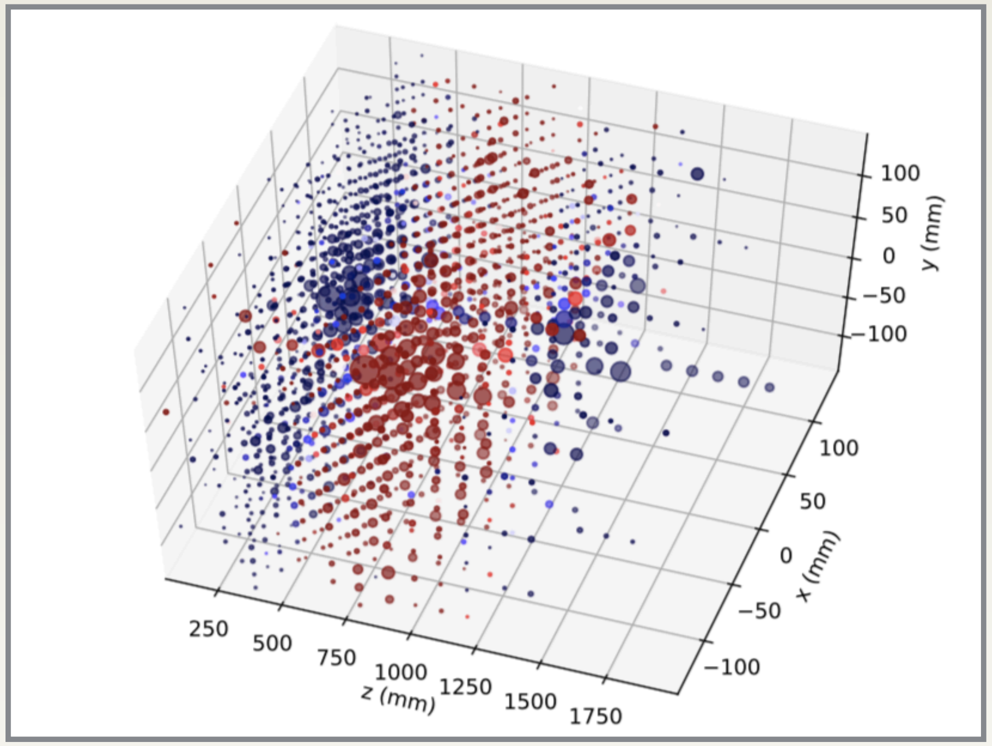
Models



- Similar total depth (counting all trainable transformations)
- All models approx 100k free parameters

Dataset and Training

- Segmentation
  - 16M events
  - Charged pions (E = 10 - 100 GeV)
    - Most complex showers
  - Shot at x,y = [-5,5] cm (random), z = -5cm
  - 2 particles per event



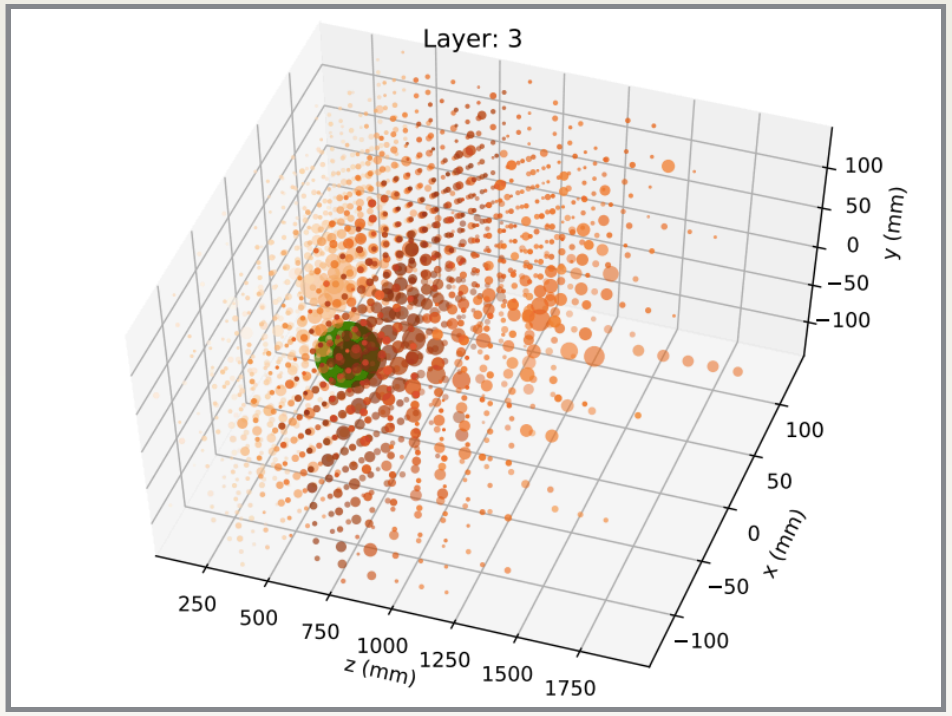
- Calorimeter
  - Tungsten
  - 30 cm x 30 cm x 2m
  - In total 2102 sensors

- Training
  - Using exponentially decaying learning rates starting around 0.0003
  - No dropout
    - With about 100k parameters no overtraining
  - Batch normalisation
  - Minimize:

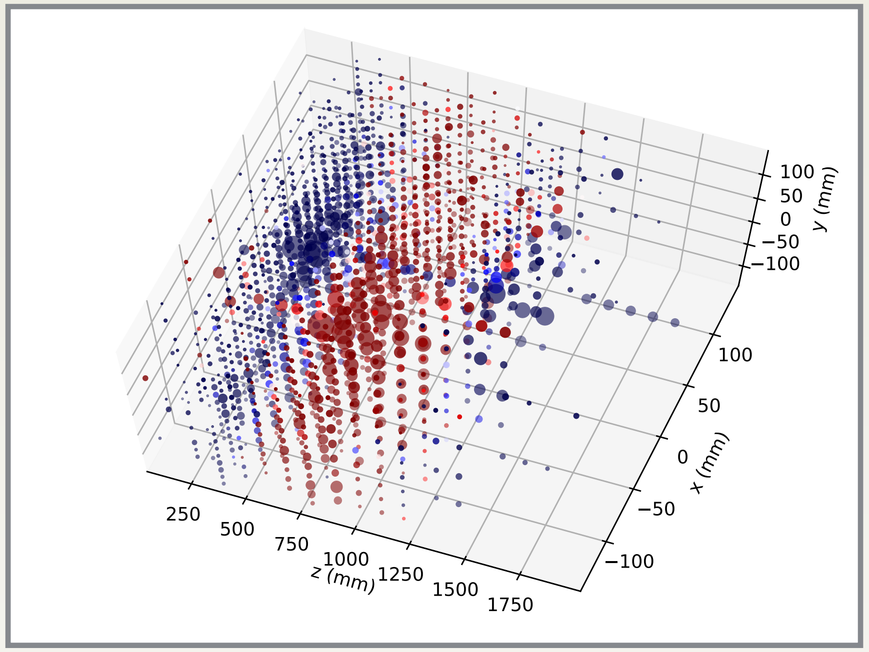
$$L = \sum_k \frac{\sum_i \sqrt{E_i t_{ki}} (p_{ki} - t_{ki})^2}{\sum_i \sqrt{E_i t_{ki}}},$$

Qualitative Performance

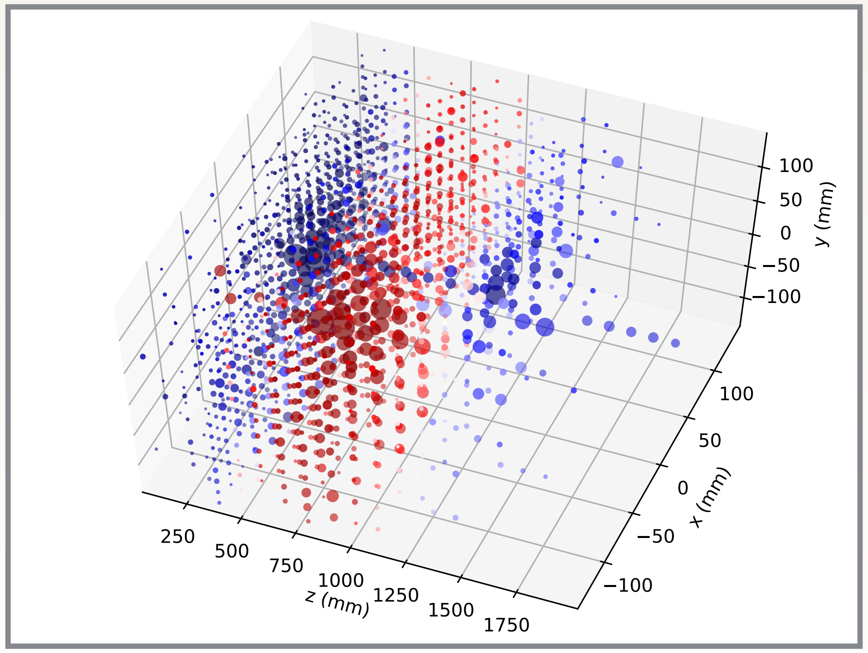
- Use distances to visualise perception of the DNN
  - Here GravNet



Truth - for reference



Prediction



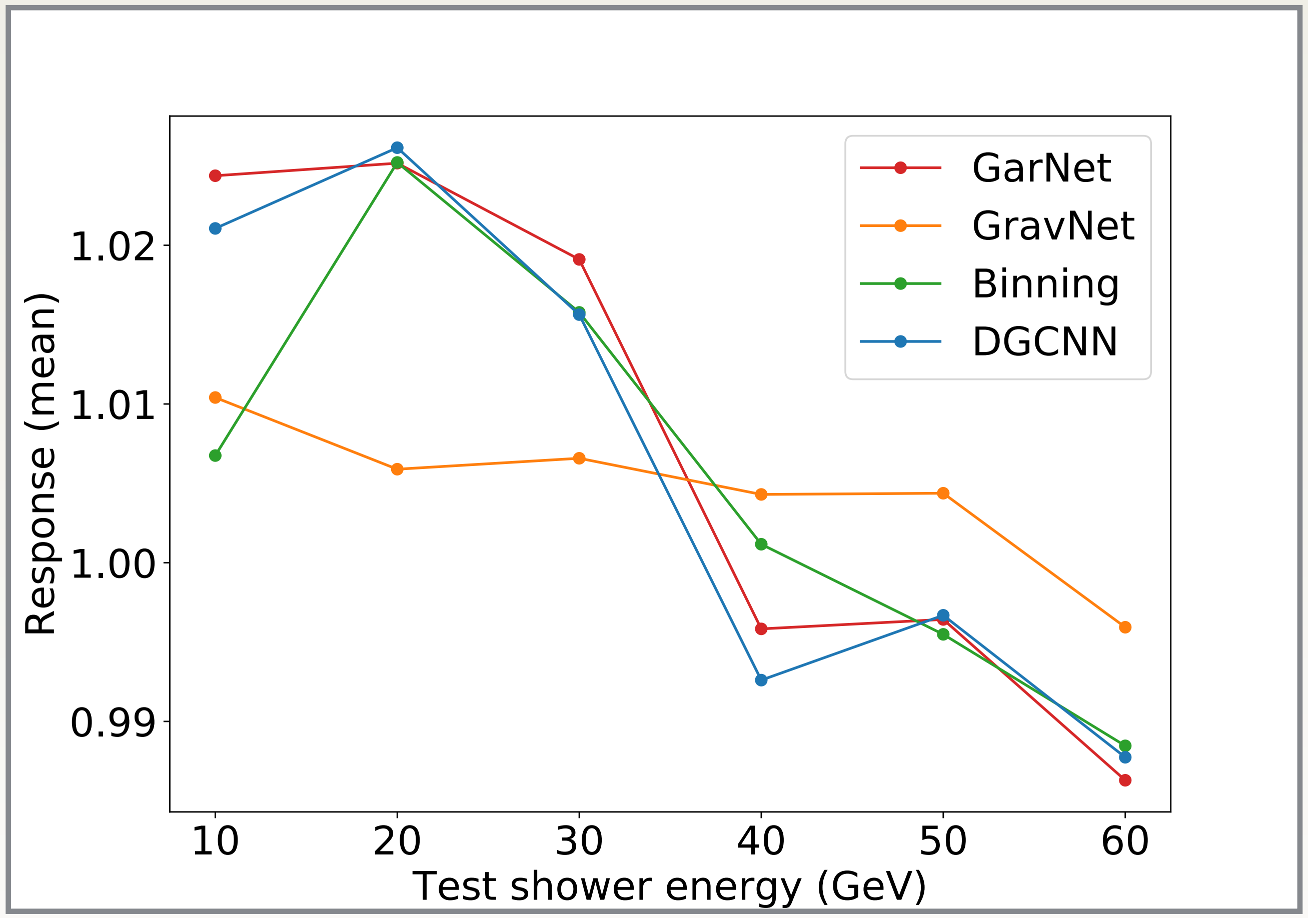
- Showers are successfully reconstructed
  - Connecting **tracks** are identified
  - EM/hadronic components are **linked**
  - Fractions are separated

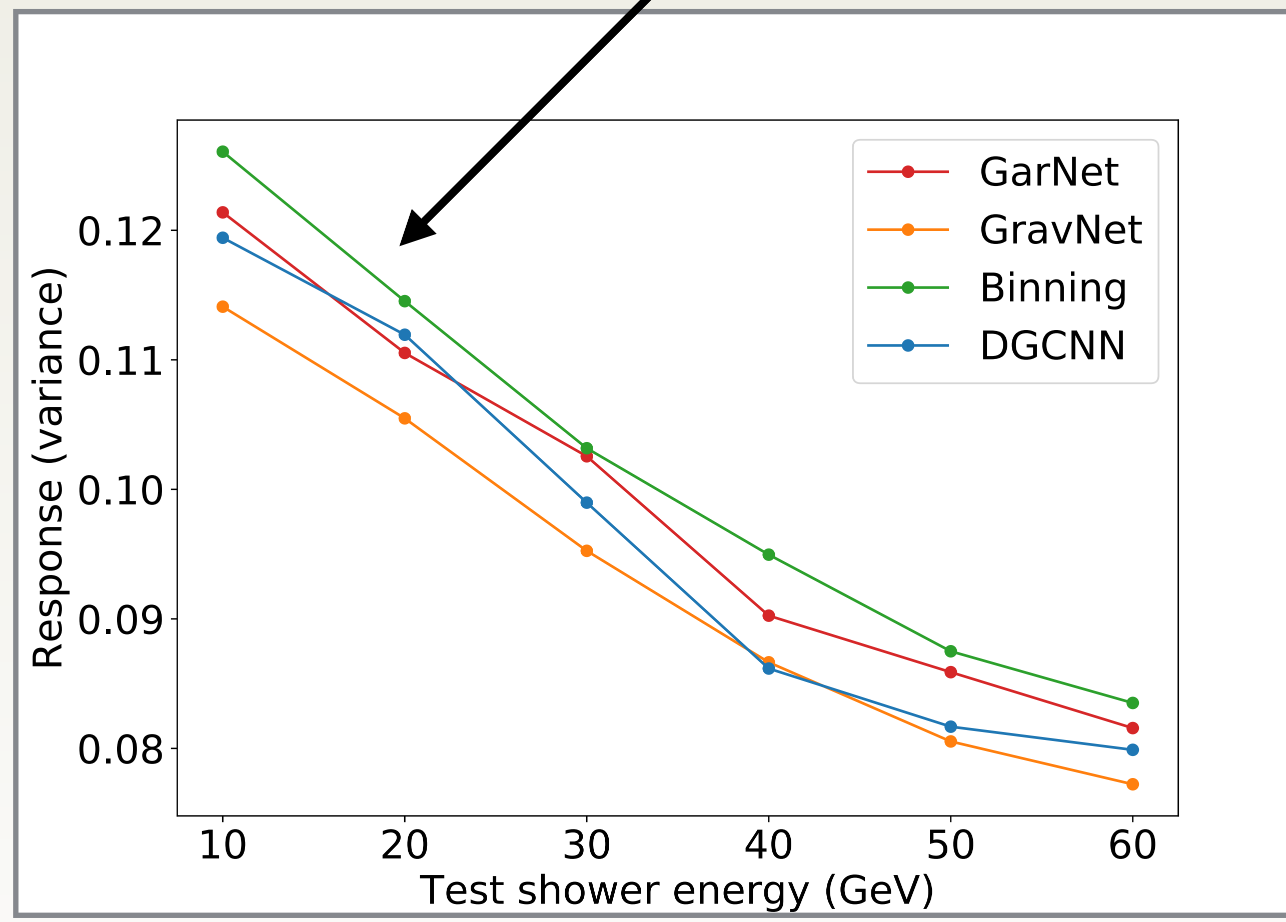
Quantitative Results

- Focus on the overlap region, only (20-80 % overlap)
- Define energy response

$$R_k = \frac{\sum_i E_i p_{ik}}{\sum_i E_i t_{ik}}$$

CNN approach
for comparison

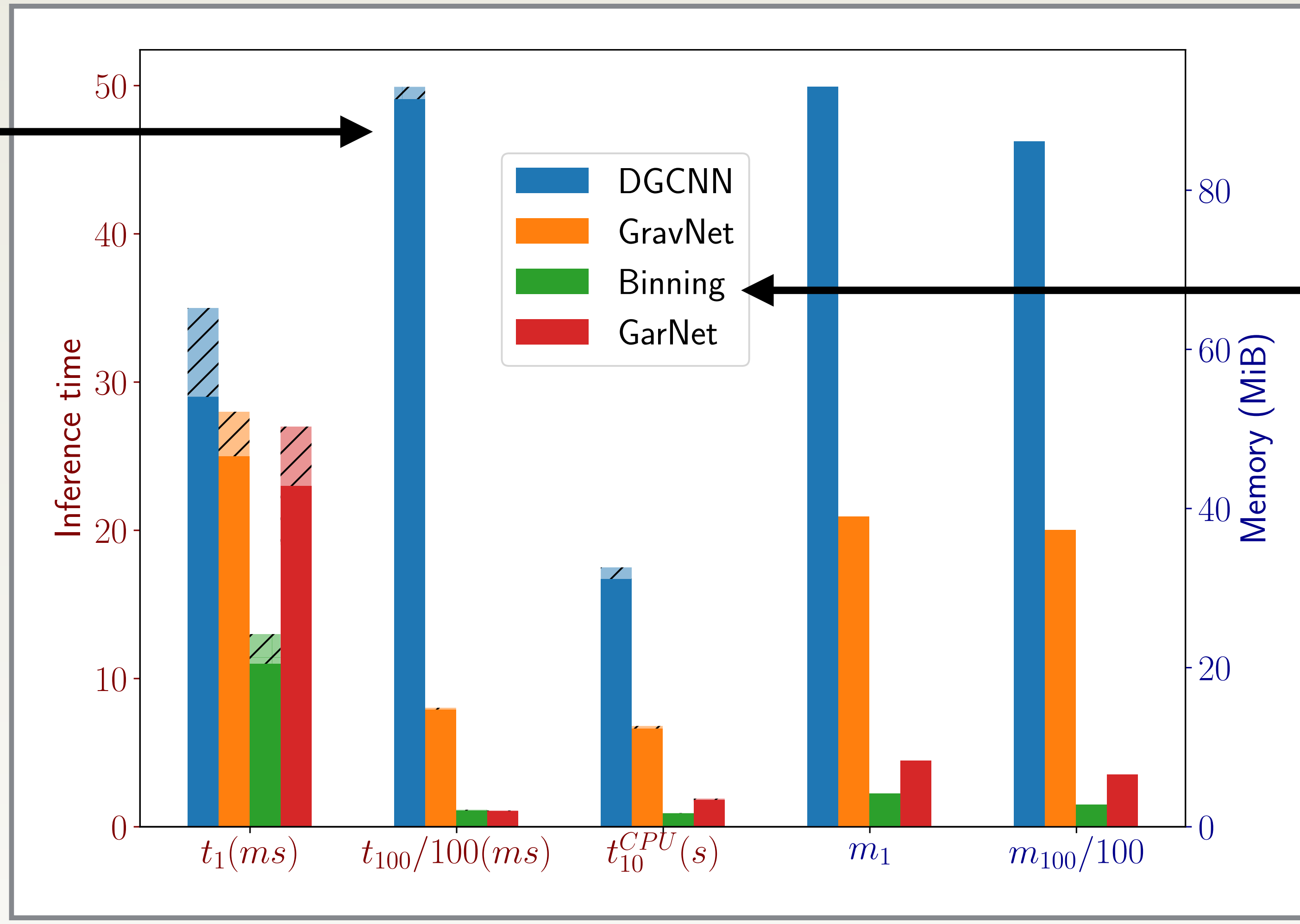




- The graph network based approaches outperform the CNN approach
  - More natural presentation of the detector
- The GravNet model outperforms all approaches

Resource Requirements

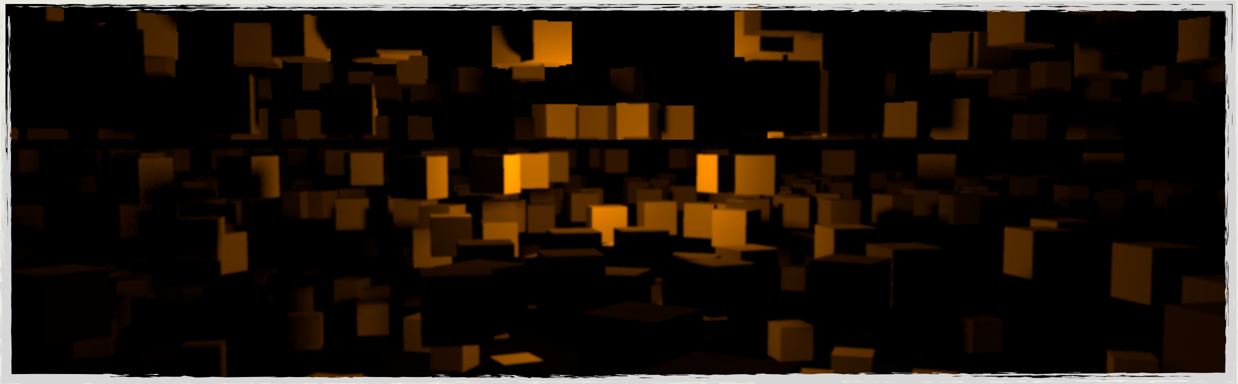
proposal
from ML literature



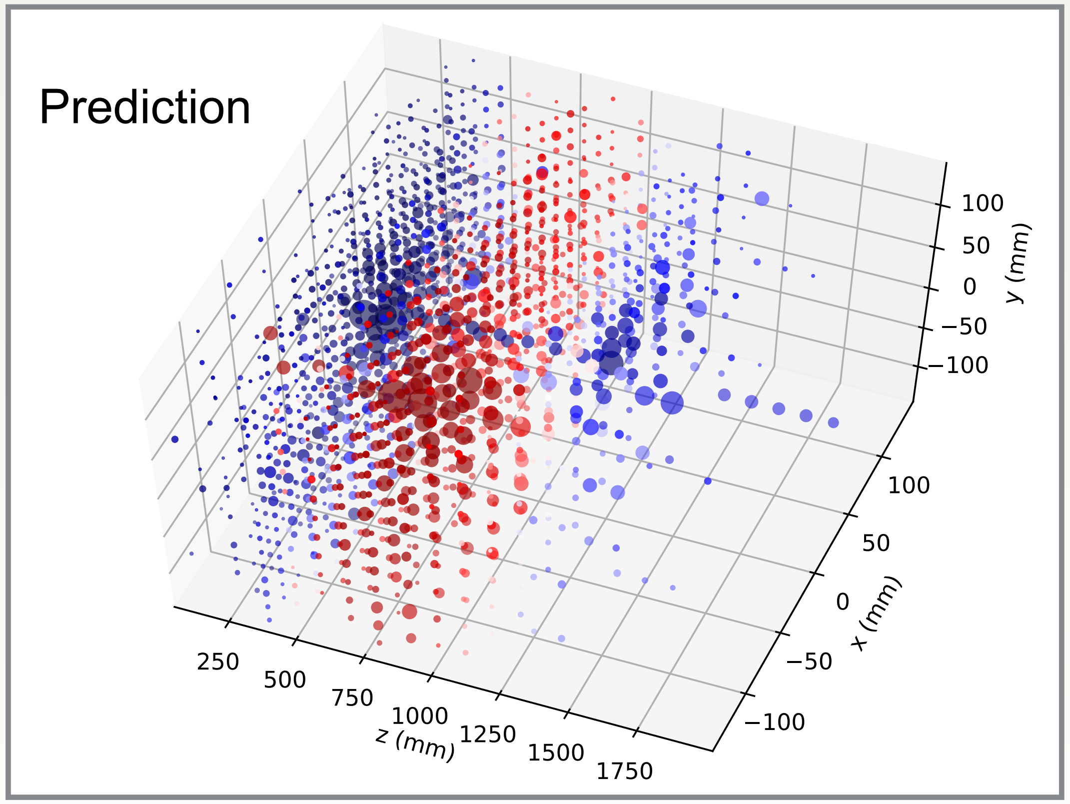
CNN approach
for comparison

- GravNet better performing and lower resource requirements than proposal from literature (DGCNN)
- GarNet very fast, developed with trigger application in mind
  - CNNs profit from highly optimised code and show worse performance and adaptation power to irregular geometries
  - Room for more improvement, e.g. taking more advantage of sparsity

Summary



- Presented graph neural network based approaches for clustering / shower segmentation
- Two new graph neural network layer proposals
  - Based on distance weighting in coordinate space
  - Reduced resource requirements
  - Better performance than suggestions in the literature
- Not limited to clustering
  - Currently studying tracking applications
  - Could be interesting for jet tagging/lepton isolation
- Tensorflow & Keras implementation https://github.com/jkiesele/caloGraphNN
  - Individual ready-to-use layers and full models



S.R. Qasim, JK, Y. Iiyama, M. Pierini
arXiv:1902.07987, submitted to EPJC