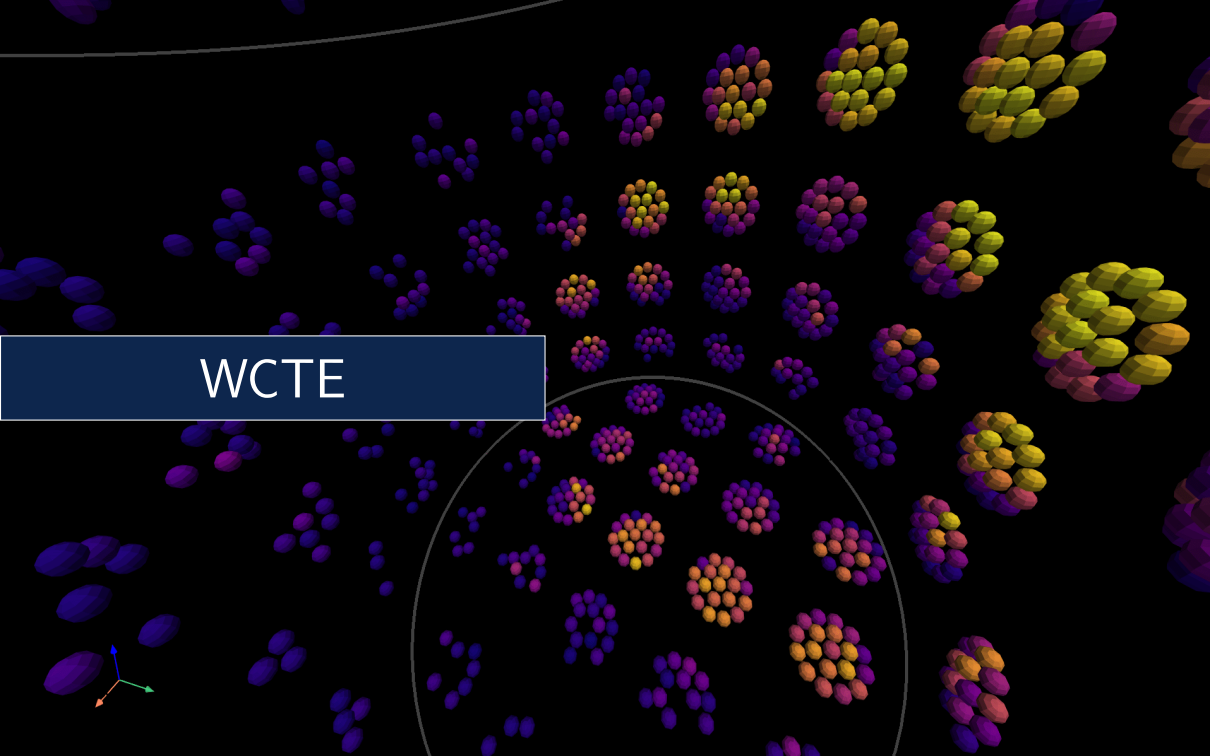
Event Reconstruction in WCTE

Graph Neural Networks

Mathieu Ferey

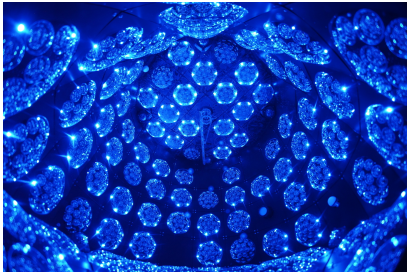
NPML 2025 | 27/10/2025

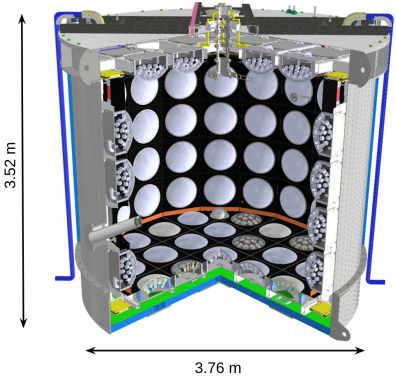

WCTE

The Water Cherenkov Test Experiment (WCTE)

HK physics goals require reducing systematic uncertainties

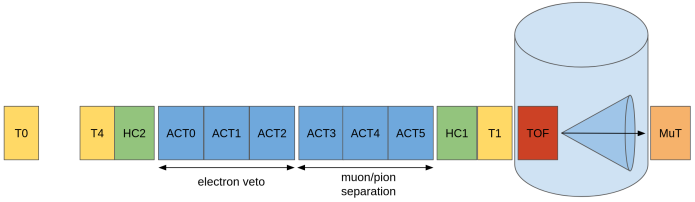
- PMT response
- particle interaction in water
- reconstruction algorithms
- propagation of Cherenkov light in water





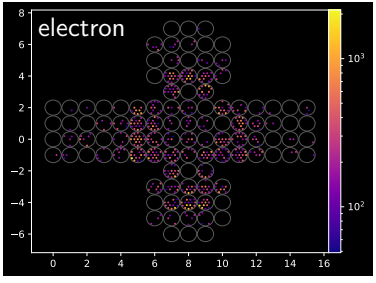
→ WCTE is a demonstrator for water Cherenkov detectors technologies, event analysis and Cherenkov physics, paving the way to IWCD and HK.

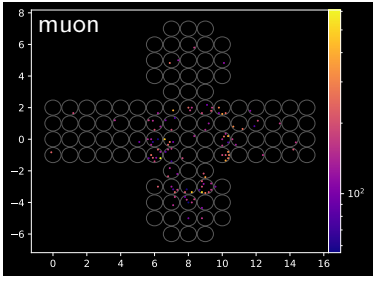
WCTE beamline



WCTE is placed in a beam of e, μ, π, protons, with a possibility of gamma production via bremsstrahlung.

Beamline monitors (TOF, Aerogel Cherenkov detectors...) provide precise particle ID and momentum measurement.



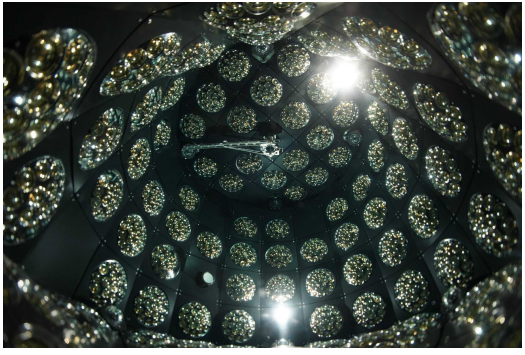


Machine learning reconstruction

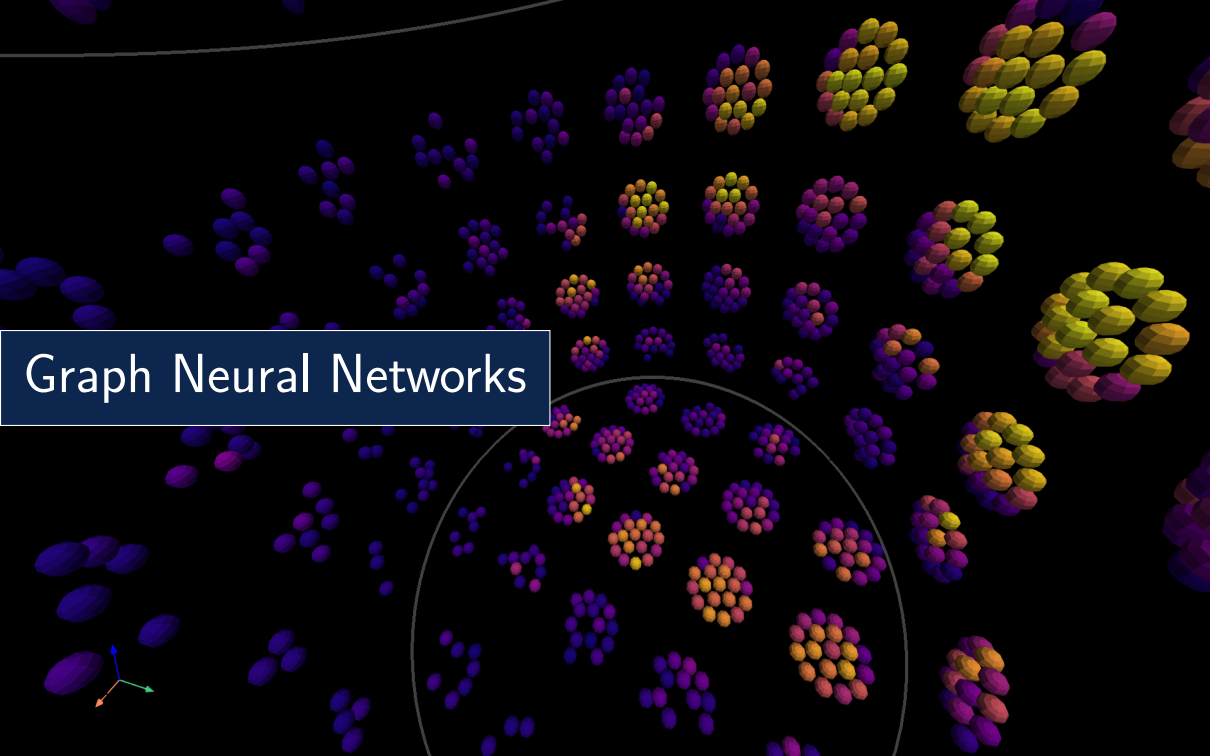
- traditional reconstruction algorithms are slow and struggle on specific tasks (e^-/γ, small towall events...)
- Neural Networks, once trained, are fast
- Neural Networks train on simulation $\rightarrow$ accurate simulation needed, as ML could learn the slightest bias

WCTE is a unique opportunity to develop and test ML reconstruction algorithms

- small detector: simulations are lightweight and training is fast
- availability of well understood beam data will allow to test the robustness of our algorithms

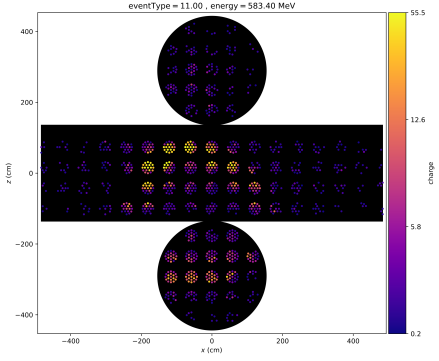


Graph Neural Networks



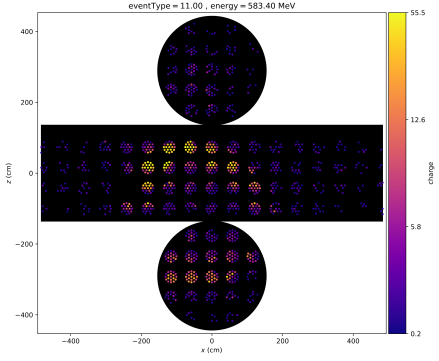
Convolutional Neural Networks

- based on unfolded tank images
- regular grid structure
- has proven its efficiency (see Nick's talk and Kieren's poster)



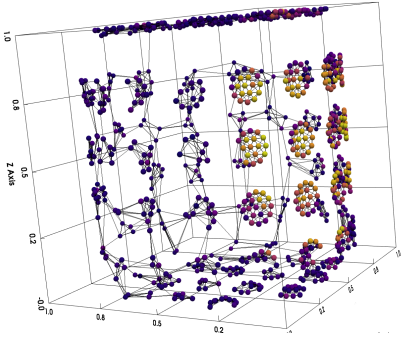
Convolutional Neural Networks

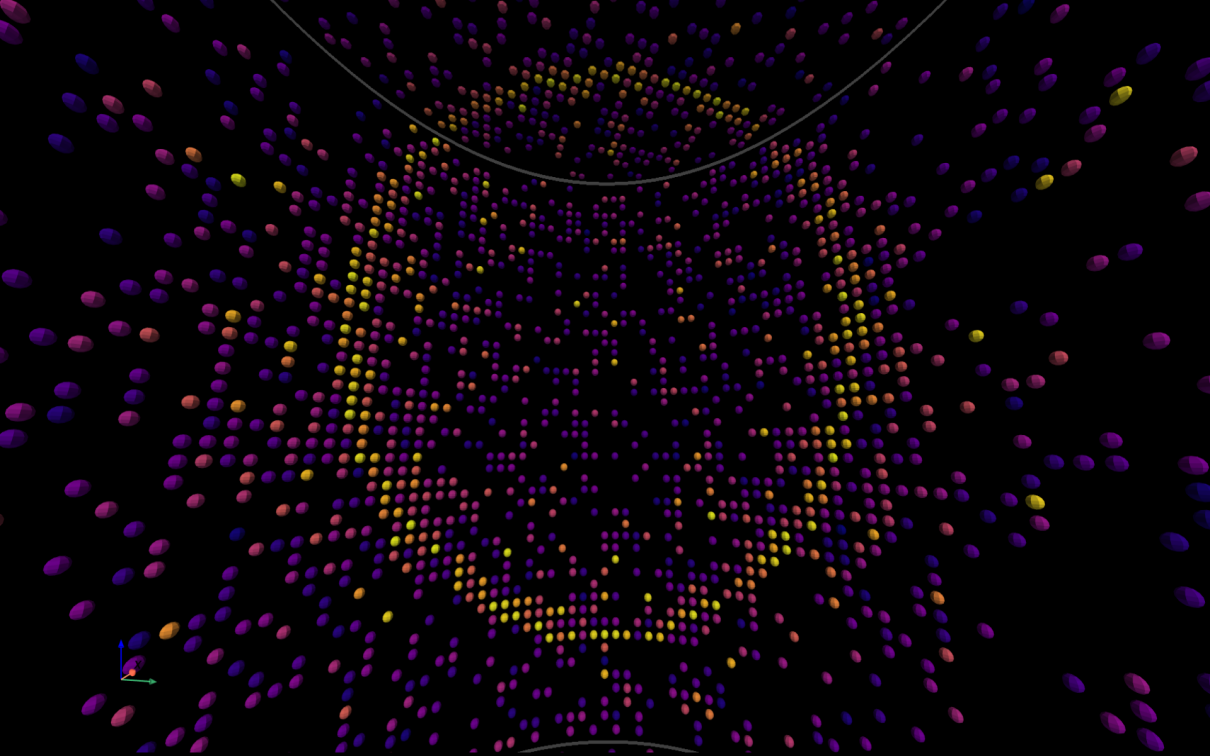
- based on unfolded tank images
- regular grid structure
- has proven its efficiency (see Nick's talk and Kieren's poster)

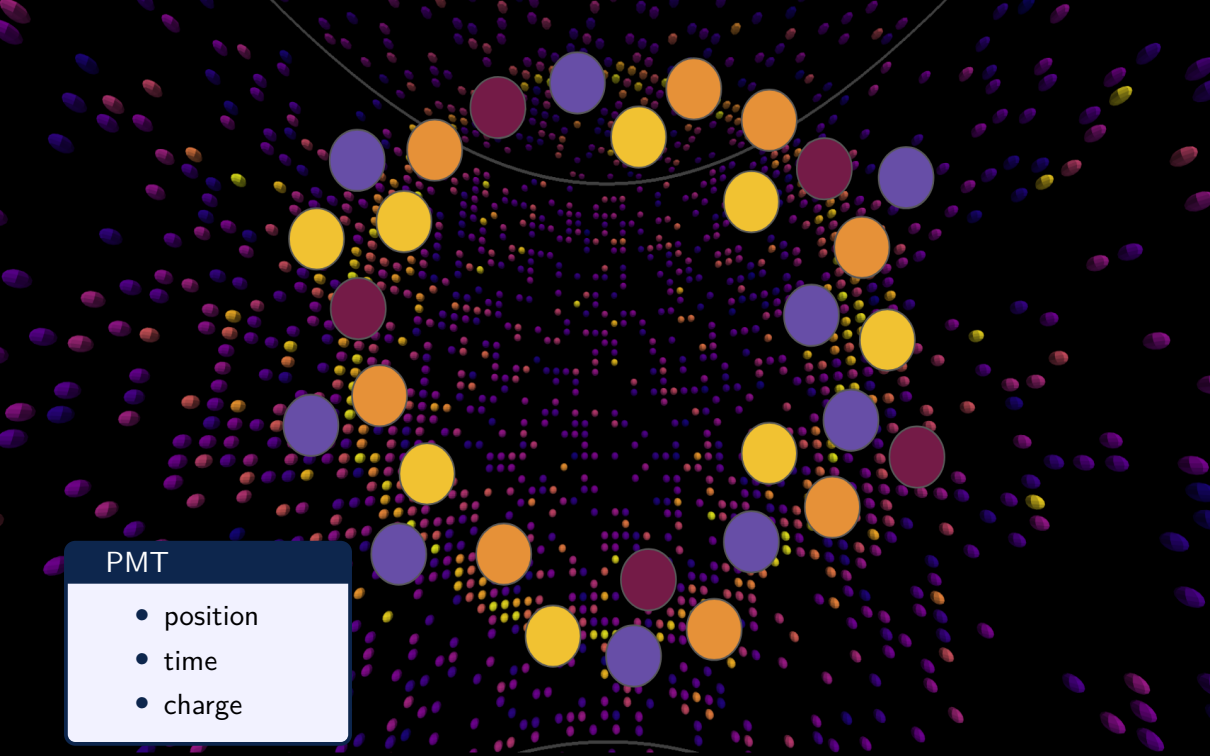


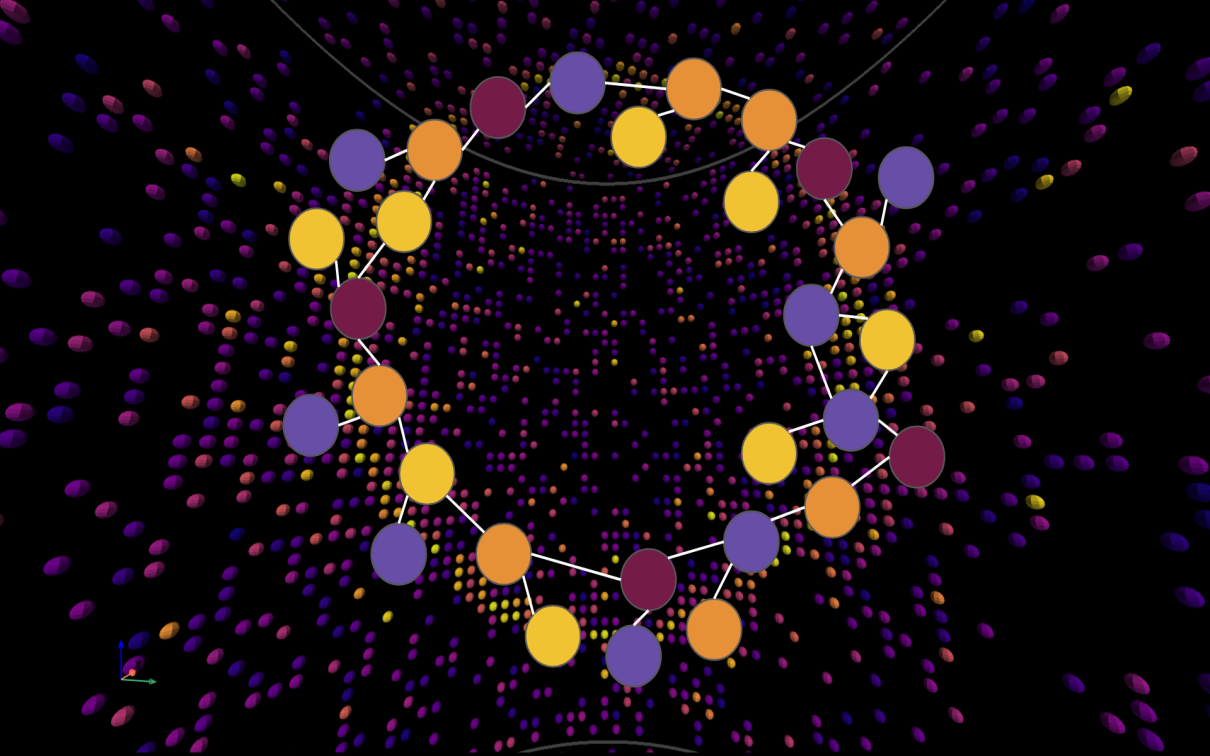
Graph Neural Networks

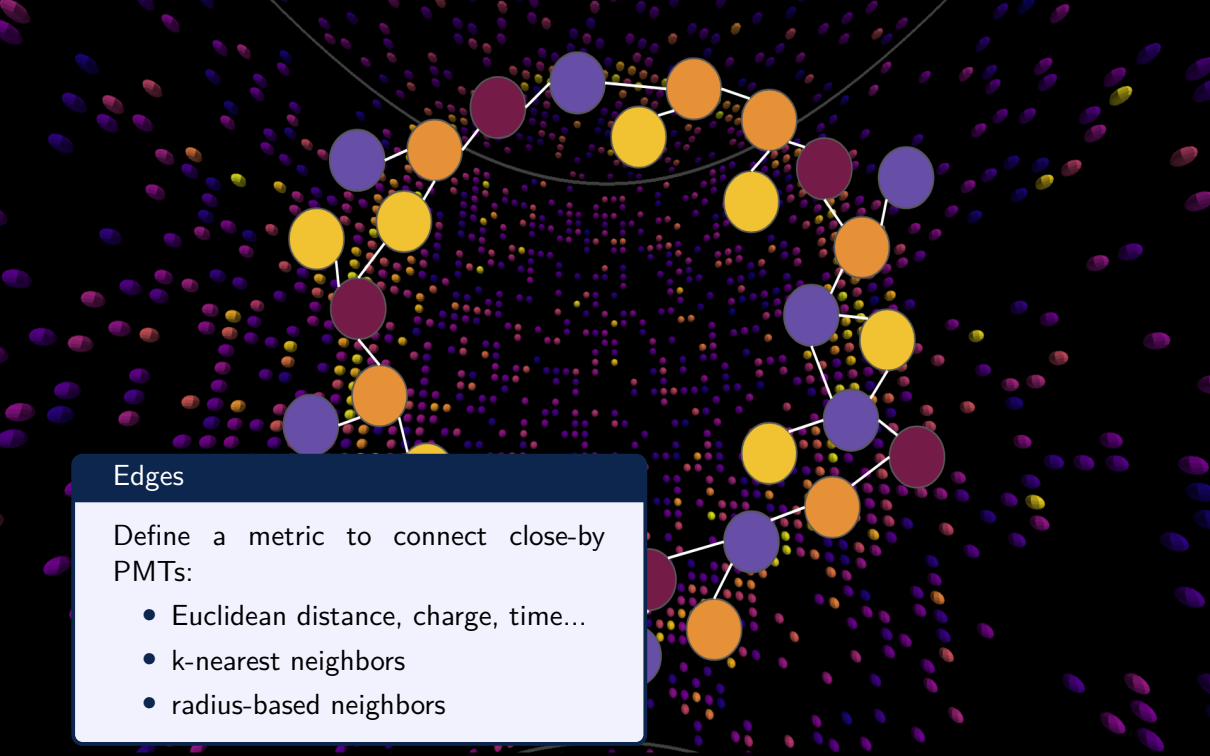
- PMTs as point cloud
- irregular structure, adapted to Cherenkov events
- relatively new, lots of things to explore!






PMT
position
time
charge



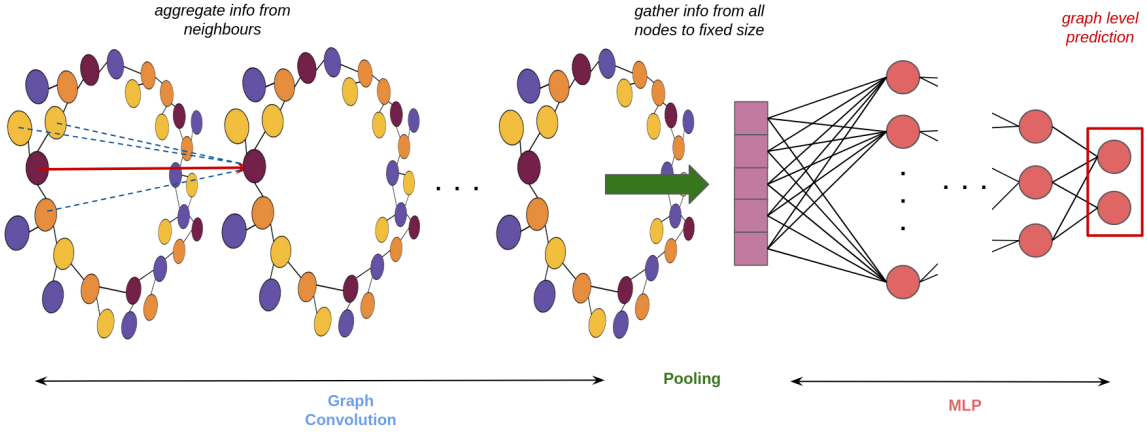


Edges

Define a metric to connect close-by PMTs:

- Euclidean distance, charge, time...
- k-nearest neighbors
- radius-based neighbors

GNN: how does it work?



Graph Attention Networks (GAT)

https://arxiv.org/pdf/1710.10903

- computes "attention coefficients" between neighbours
- attention coefficients are computed as a single-layer feedforward neural-network, normalized with a softmax
- weighs the importance of each neighbour to a given node
- possibility of multi-head attention for more stability

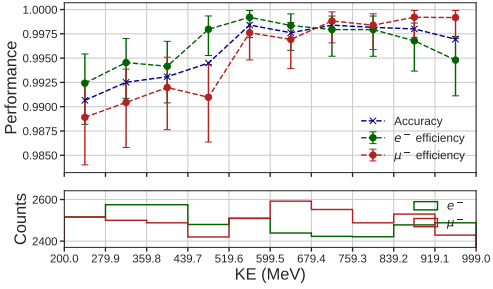
ResGated Graph Networks

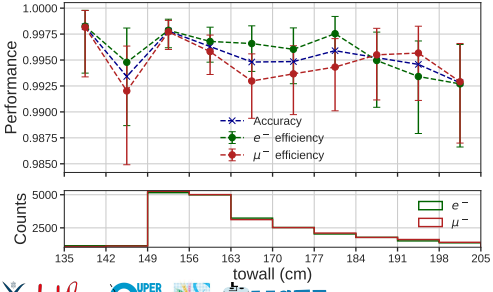
https://arxiv.org/pdf/1711.07553

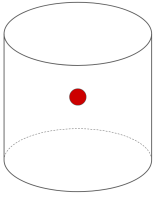
- uses a gate (sigmoid) on neighbouring nodes
- residual connection to help with training deeper networks
- only keep the most relevant neighbours of a given node

Results overview

e^-/μ^- separation: center of the tank



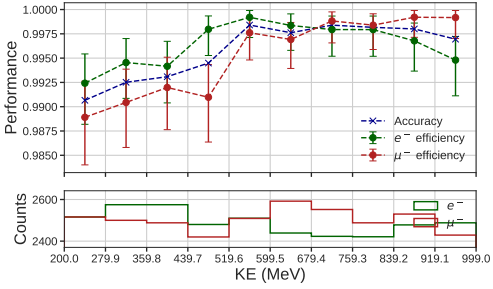


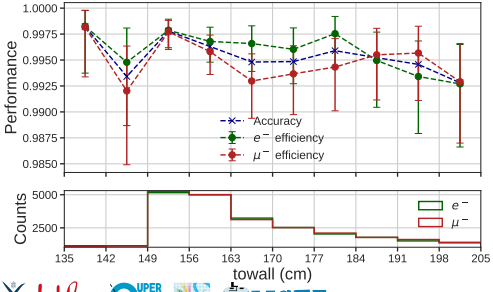


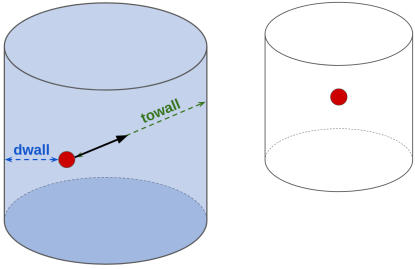
- slight decrease of performance at low KE

signal efficiency at 5% background acceptance

	Center of tank	**Uniform in tank**
e/μ	99.96%	—
μ/π	—	—





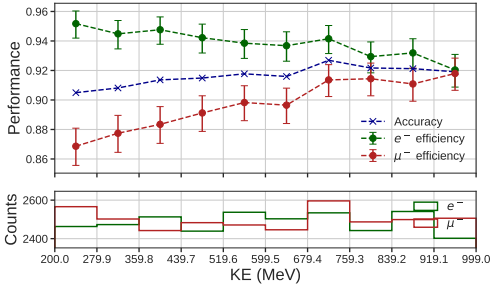


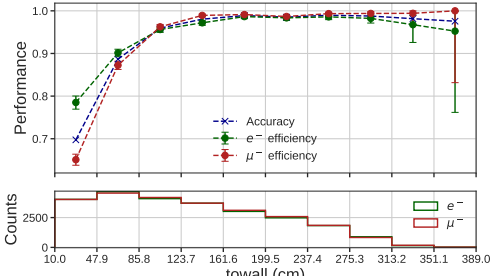
- slight decrease of performance at low KE

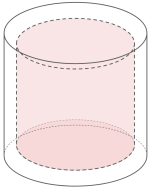
signal efficiency at 5% background acceptance

	Center of tank	**Uniform in tank**
e/μ	99.96%	—
μ/π	—	—

e^-/μ^- separation: uniform in the tank





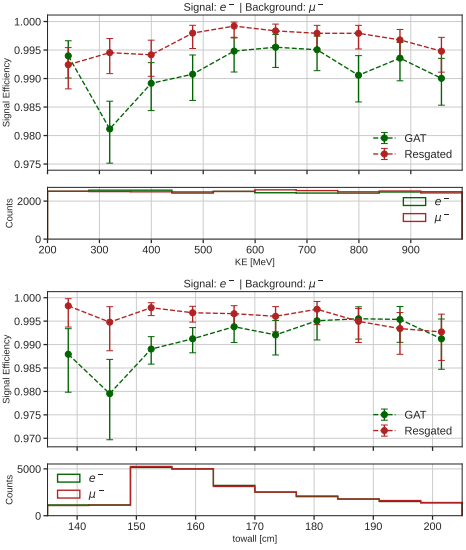


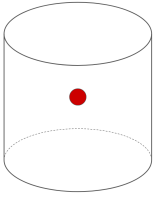
- muon efficiency decreases at low KE
- electron efficiency decreases at high KE
- overall loss of performance at low towall
- the model learns that muons are more likely to go out of the tank

signal efficiency at 5% background acceptance

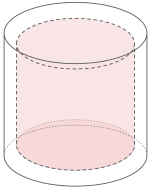
	Center of tank	**Uniform in tank**
e/μ	99.96%	88.30%
μ/π	—	—

Vertex at the center of the tank

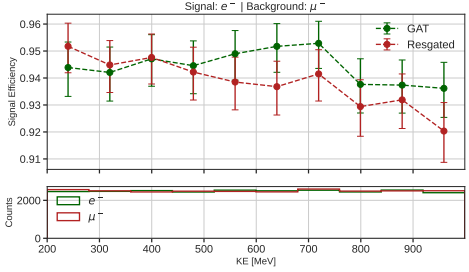


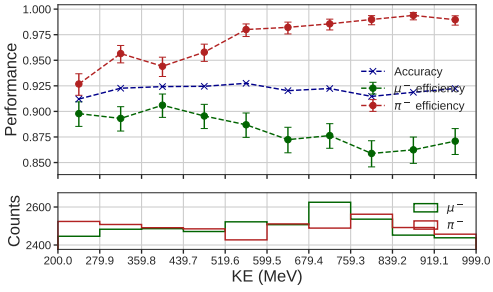


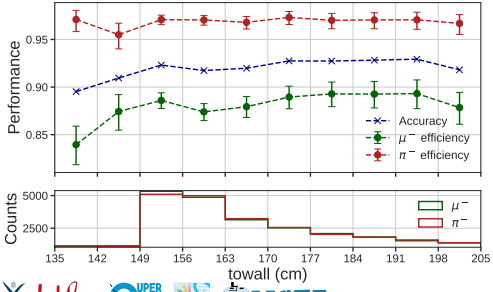
- Similar perfomances for both architectures
- ResGated seems to perform slightly better, but GAT training can still be pushed a bit further

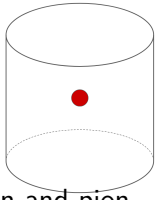


Vertex uniform in the tank







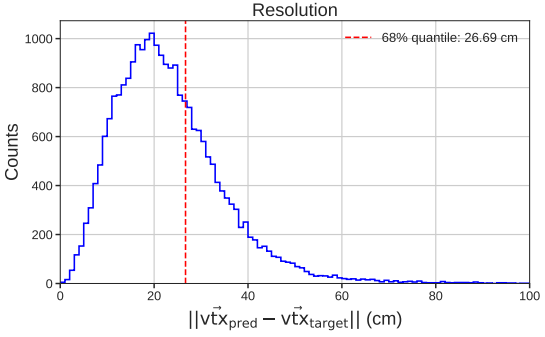


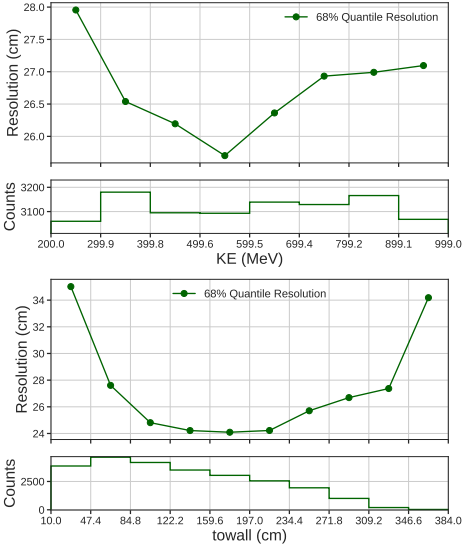
- big asymmetry between muon and pion efficiency
- pion efficiency decreases at low KE
- muon efficiency decreases at high KE

signal efficiency at 5% background acceptance

	Center of tank	**Uniform in tank**
e/μ	99.96%	88.30%
μ/π	62.22%-88.49%	—

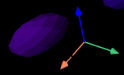
e^- vertex regression





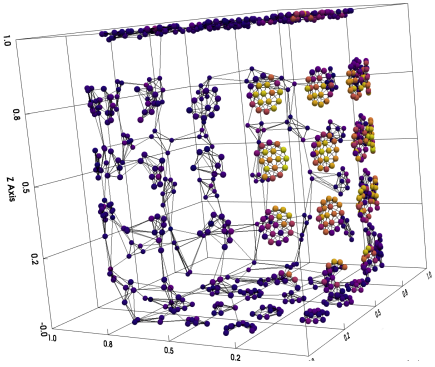
- 26.69cm resolution on vertex position
- decrease of resolution at low KE
- decrease of resolution at low towall

Final words



Room for improvement:

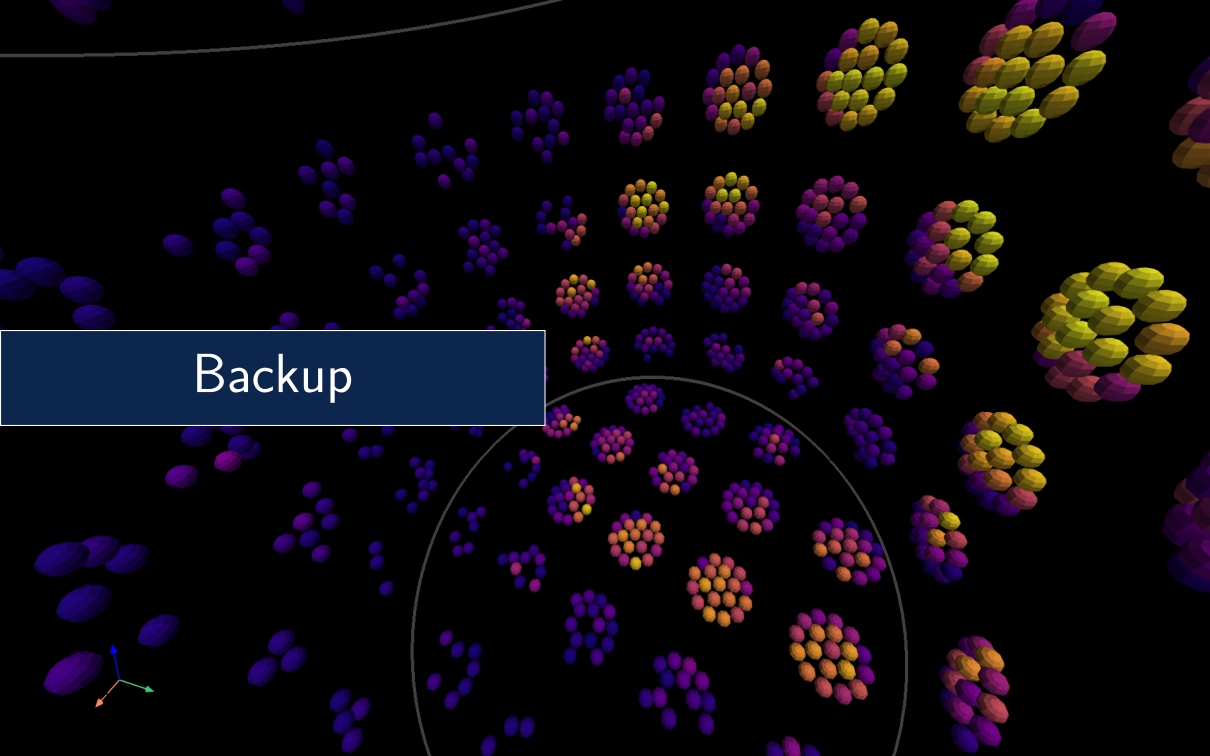
- small datasets and small networks so far
- pooling at the end of the graph convolution might lose too much information
- need to rethink graphs with mPMTs so as to better capture the ring in the graph structure
- use PMTs orientation in mPMTs as well



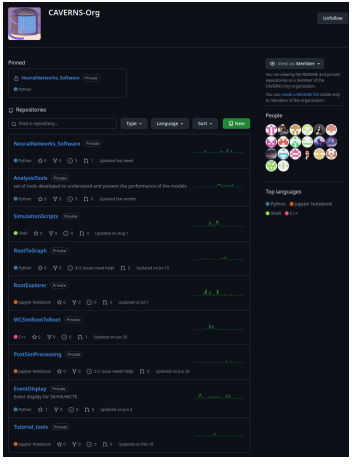
Prospects

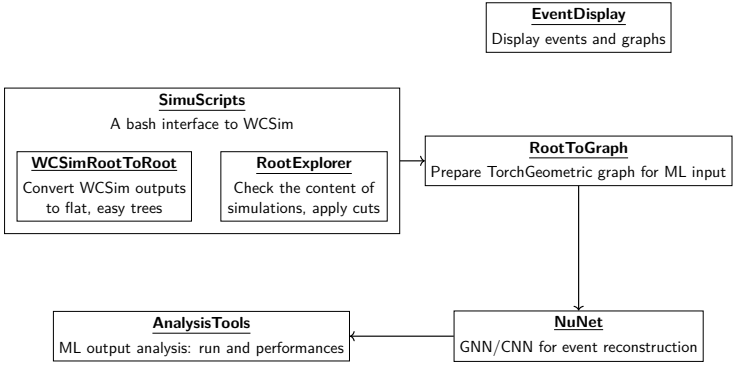
- Compare with other algorithms, both traditional (FiTQun) and ML (watchmal)
- Apply on data once they're understood and simulations are tuned accordingly

Backup



A small but growing team!





Graph Attention Networks (GAT)

https://arxiv.org/pdf/1710.10903

- start with a set of node features $\mathbf{h} = \{\vec{h}_1, \vec{h}_2, ..., \vec{h}_N\}$, $\vec{h}_i \in \mathbb{R}^F$
- apply a shared linear transformation $\mathbf{W} \in \mathbb{R}^{F' \times F}$ to every node
- compute attention coefficients based on a shared attentional mechanism $a : \mathbb{R}^{F'} \times \mathbb{R}^{F'} \to \mathbb{R}$

$$e_{ij} = a(\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_j),\ j \in \mathcal{N}_i$$

- the attention mechanism a is a single-layer MLP with a ReLU activation
- the output attention coefficients are then computed using a softmax function:

$$\alpha_{ij} = \frac{e_{ij}}{\sum_{k \in \mathcal{N}_i} e_{ik}}$$

- adding non-linearity and keeping residual connections, the final output of the layer for node i is:

$$\vec{h}'_i = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} \mathbf{W}\vec{h}_j\right) + \vec{h}_i$$

https://arxiv.org/pdf/1711.07553

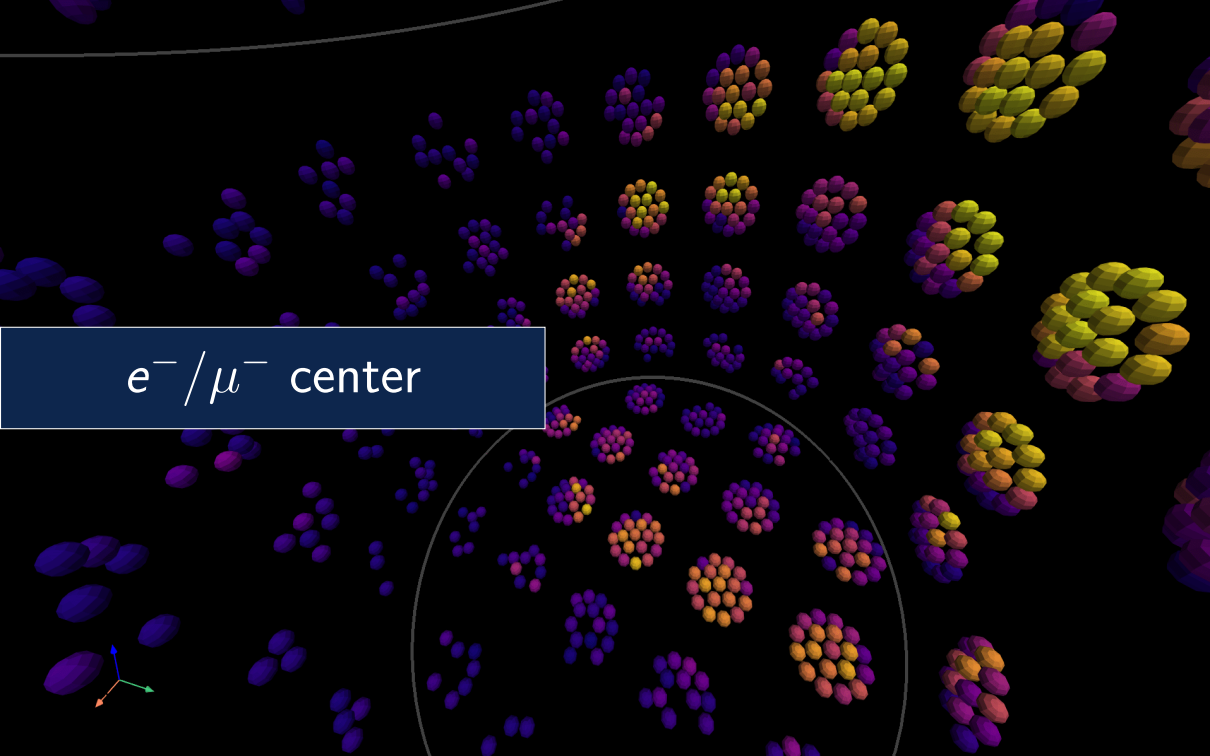
Very similar to GAT, but uses a gated mechanism instead of an attention mechanism

$$\vec{h}'_i = \mathrm{ReLU}\left(\sum_{j\in\mathcal{N}_i} \eta_{ij}\mathbf{W}_1\vec{h}_j + \mathbf{W}_2\vec{h}_i\right)$$

where η_{ij} is a gate computed as:

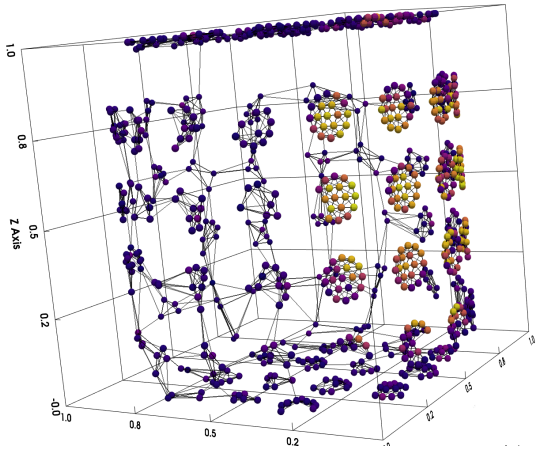
$$\eta_{ij} = \sigma\left(\mathbf{W_3}\vec{h}_i + \mathbf{W_4}\vec{h}_j\right)$$

with σ the sigmoid function.


e^-/μ^- center

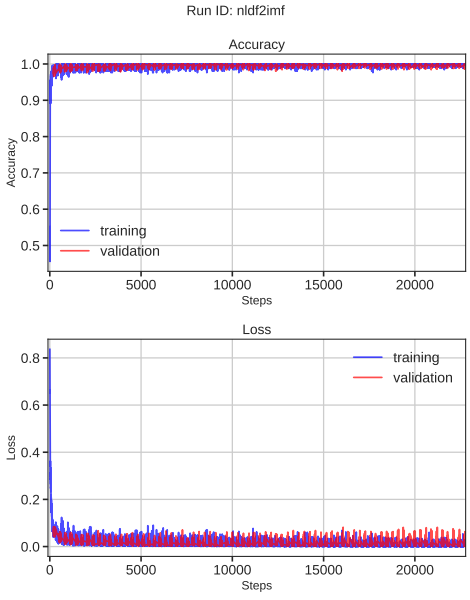
The dataset

- **Simulation**
  - WCSim 1.12.20
  - 100k e^- and 100k μ^-
  - from the center of the tank, isotropic
  - 200-1000 MeV
- **Graph**
  - Train/Val/Test split: 100k/50k/50k
  - knn graph: k=5
  - nodes: charge, time
  - edges: hits

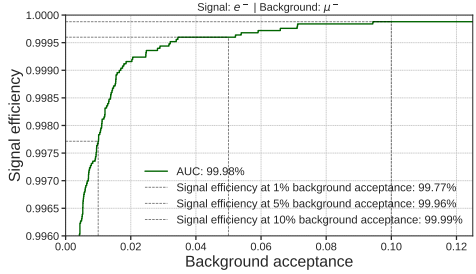


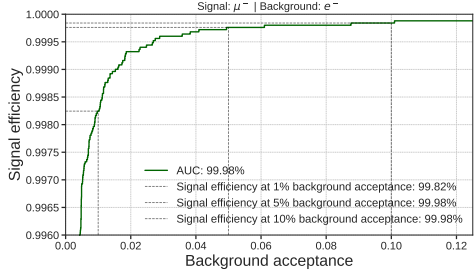
Hyperparameter	Value
Model	ResGatedConv_v2
Conv In Channels	[16, 32, 64]
Linear Out Features	[64, 32, 16, 8, 2]
Train Batch Size	500
Epochs	75
Loss Function	CrossEntropyLoss
Optimizer	Adam
Learning Rate	0.001

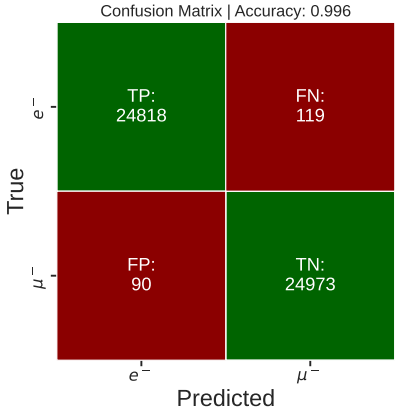
Number of parameters: 18314
Run time: 38.10 mins on two gpus



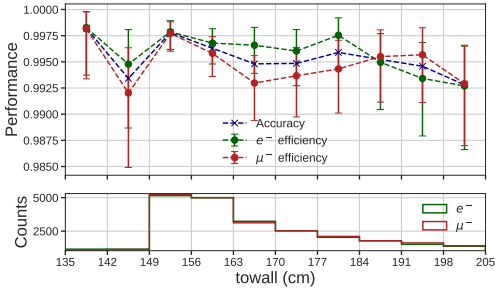
Results: performances

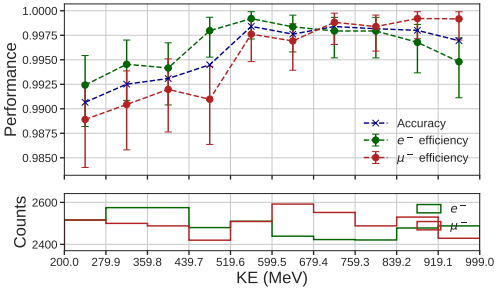




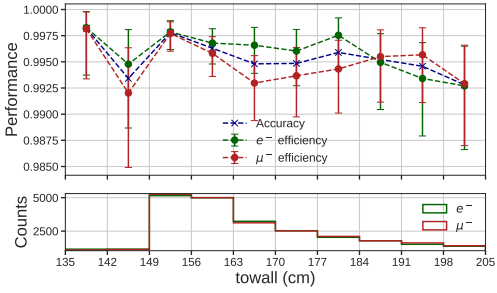


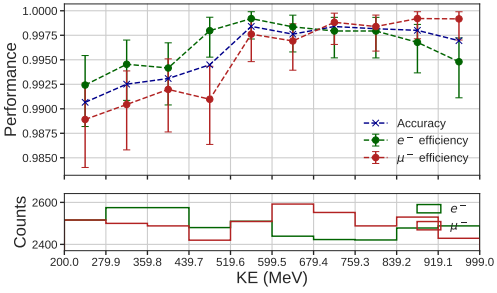
Results: efficiencies

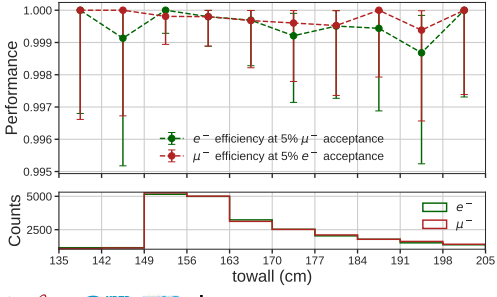


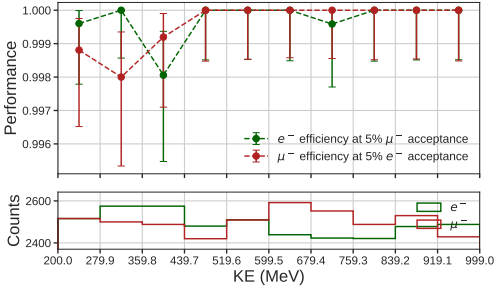


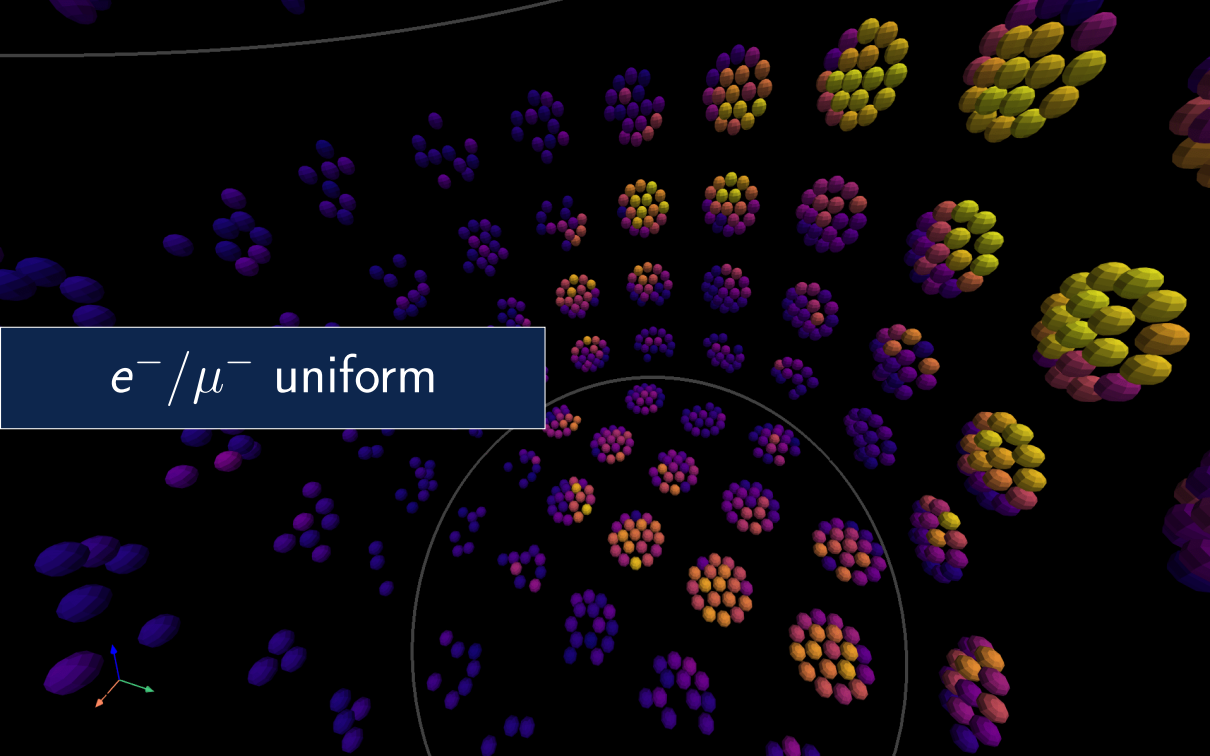
Results: efficiencies





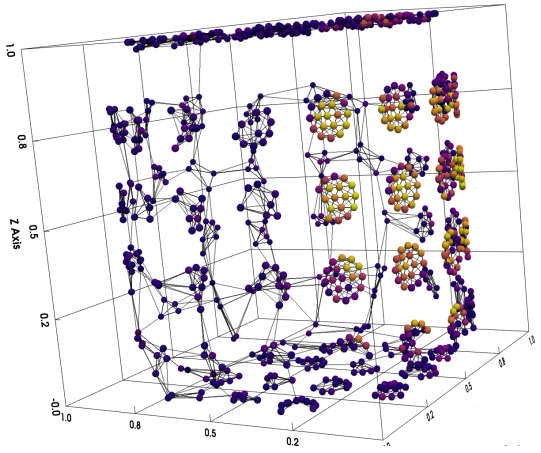





e^-/μ^- uniform

The dataset

- **Simulation**
  - WCSim 1.12.20
  - 100k e^- and 100k μ^-
  - uniform in the tank, isotropic
  - 200-1000 MeV
- **Graph**
  - Train/Val/Test split: 100k/50k/50k
  - knn graph: k=5
  - nodes: charge, time
  - edges: hits

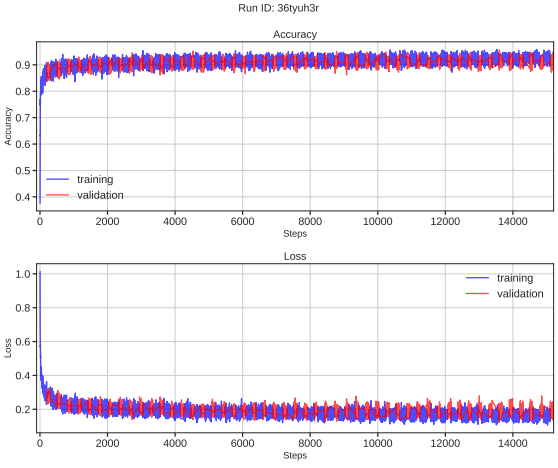


The GNN

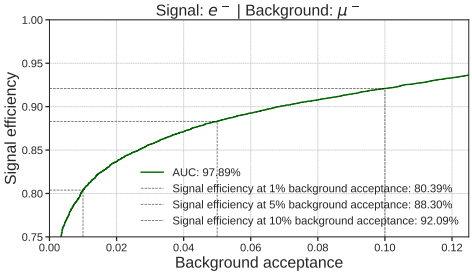
Hyperparameter	Value
Model	ResGatedConv_v2
Conv In Channels	[16, 32, 64]
Linear Out Features	[64, 32, 16, 8, 2]
Train Batch Size	500
Epochs	75
Loss Function	CrossEntropyLoss
Optimizer	Adam
Learning Rate	0.001

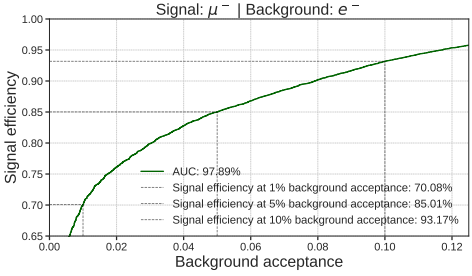
Number of parameters: 18314

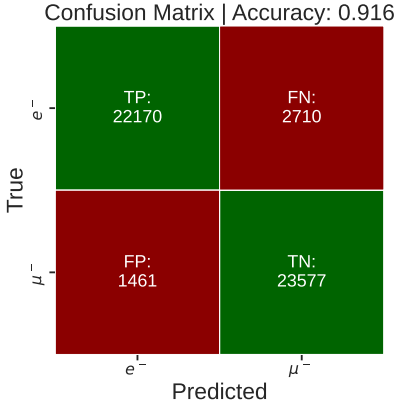
Run time: 40.20 mins on one gpus



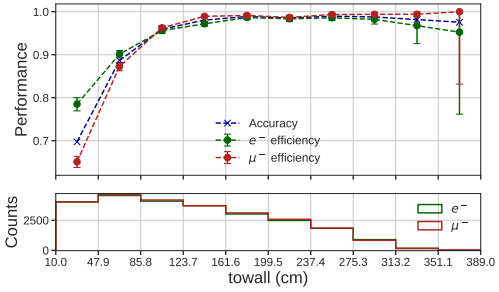
Results: performances

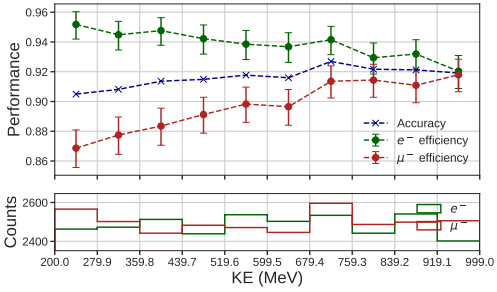




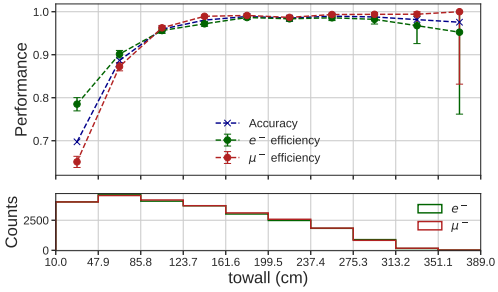


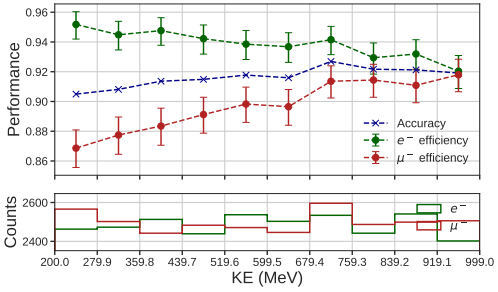
Results: efficiencies

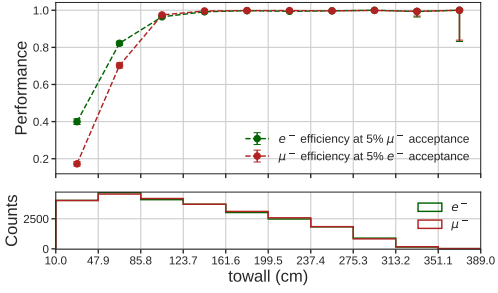


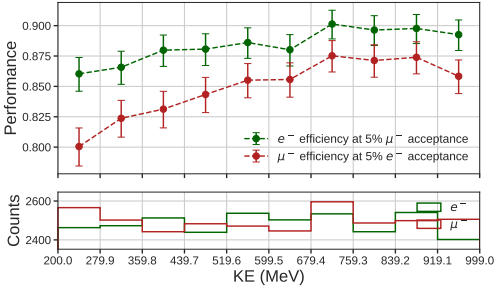


Results: efficiencies

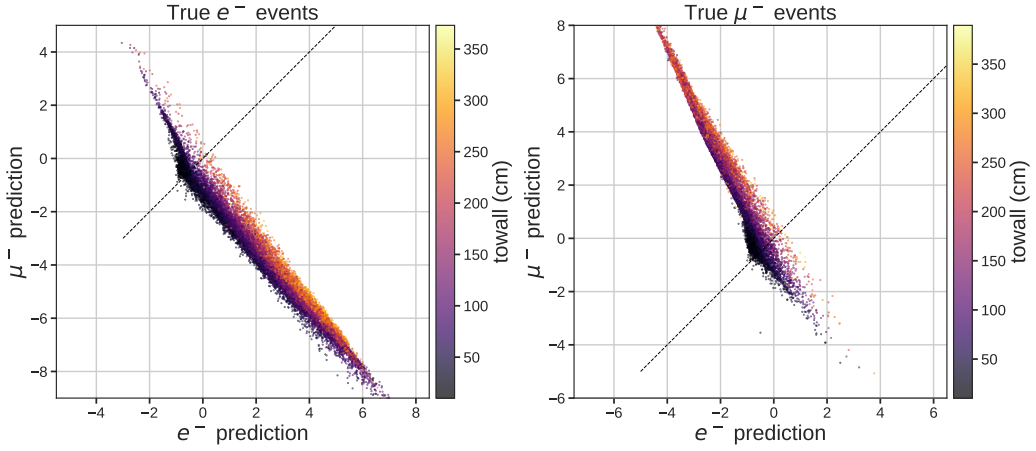








Prediction distributions

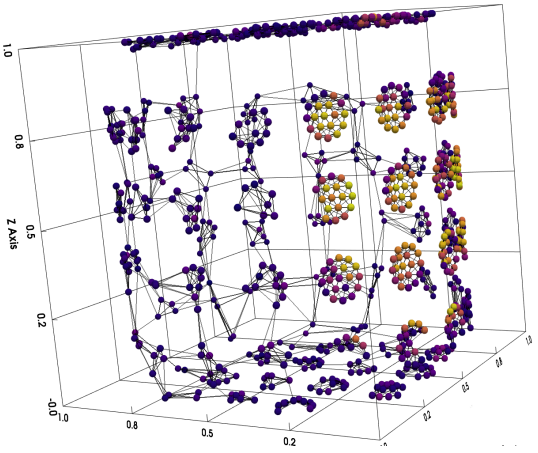


several trainings on different towall ranges


μ^-/π^- center

The dataset

- **Simulation**
  - WCSim 1.12.20
  - 100k μ^- and 100k π^-
  - from the center of the tank, isotropic
  - 200-1000 MeV
- **Graph**
  - Train/Val/Test split: 100k/50k/50k
  - knn graph: k=5
  - nodes: charge, time
  - edges: hits

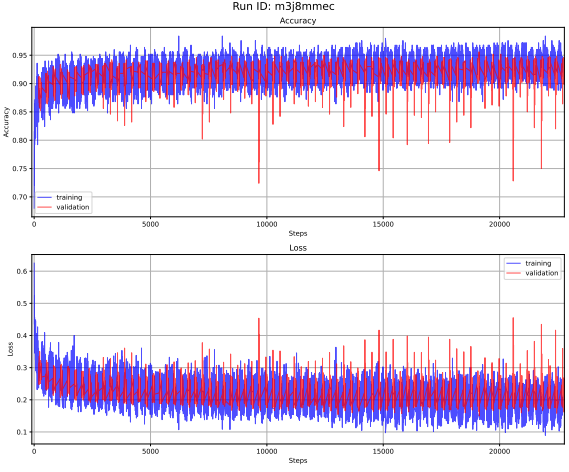


The GNN

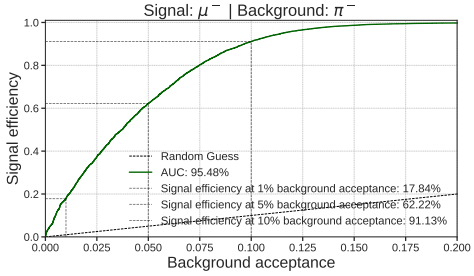
Hyperparameter	Value
Model	ResGatedConv_v2
Conv In Channels	[16, 32, 64]
Linear Out Features	[64, 32, 16, 8, 2]
Train Batch Size	500
Epochs	75
Loss Function	CrossEntropyLoss
Optimizer	Adam
Learning Rate	0.001

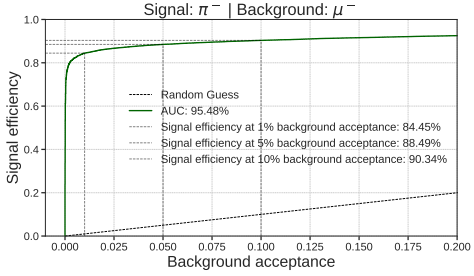
Number of parameters: 18314

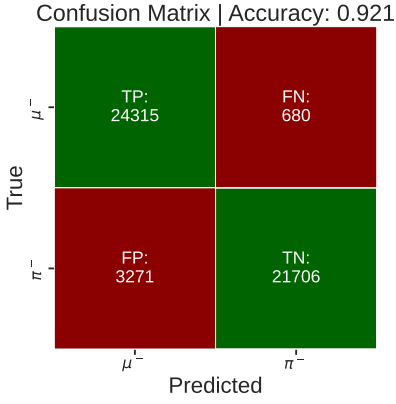
Run time: 31.85 mins on two gpus



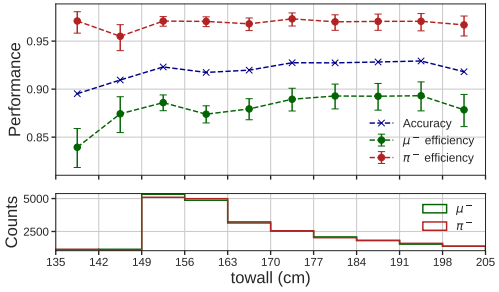
Results: performances

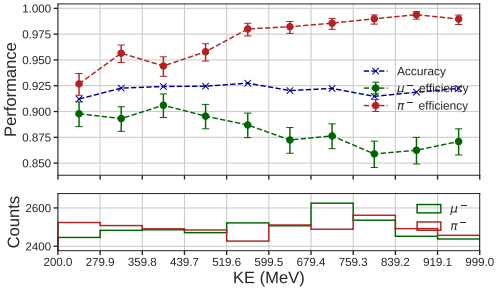




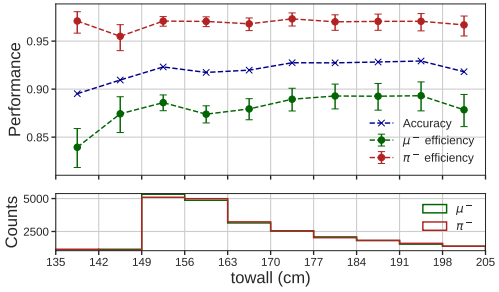


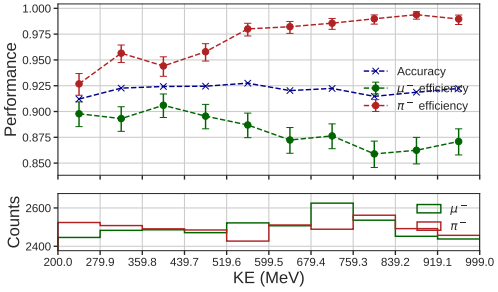
Results: efficiencies

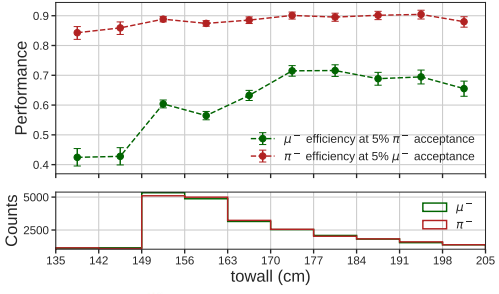


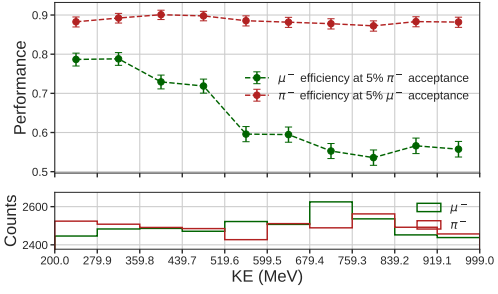


Results: efficiencies



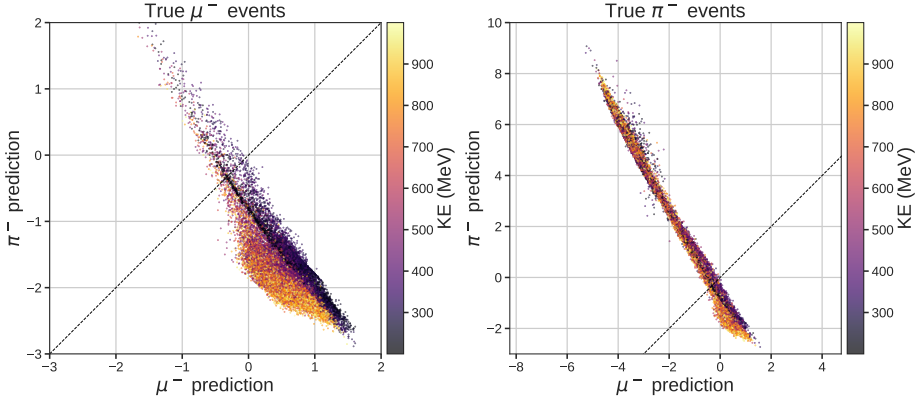


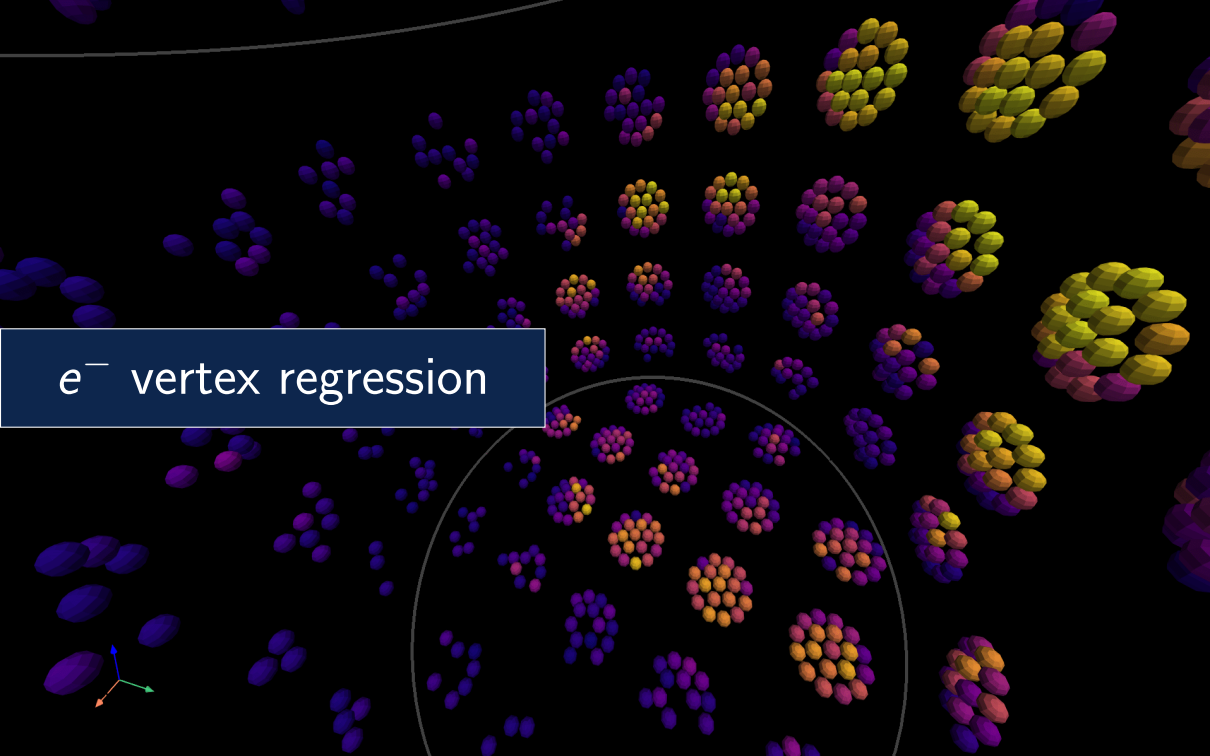




Prediction distributions

High energy pions look like muons.




e^- vertex regression

The dataset

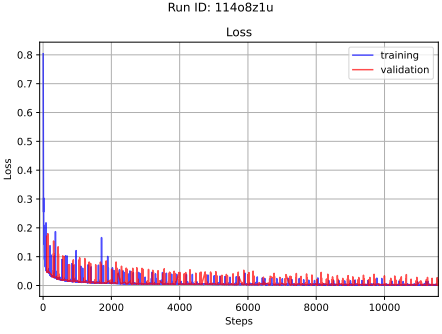
- Dataset: 100k e^-
- Uniform and isotropic in the tank
- 200-1000 MeV
- Train/Val/Test split: 50k/25k/25k
- knn graph: k=5
- nodes: hits, time
- edges: time

The GNN

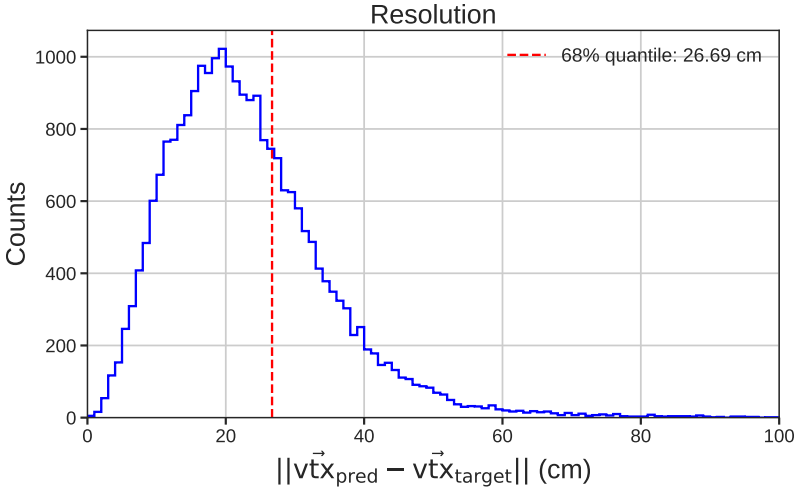
Hyperparameter	Value
Model	ResGatedConv_v2
Conv In Channels	[8, 32, 64]
Linear Out Features	[32, 16, 3]
Train Batch Size	500
Epochs	75
Loss Function	MSELoss
Optimizer	Adam
Learning Rate	0.0005

Number of parameters: 12827

Run time: 26.29 mins on two gpus



Results: vertex resolution



What about a prefit?