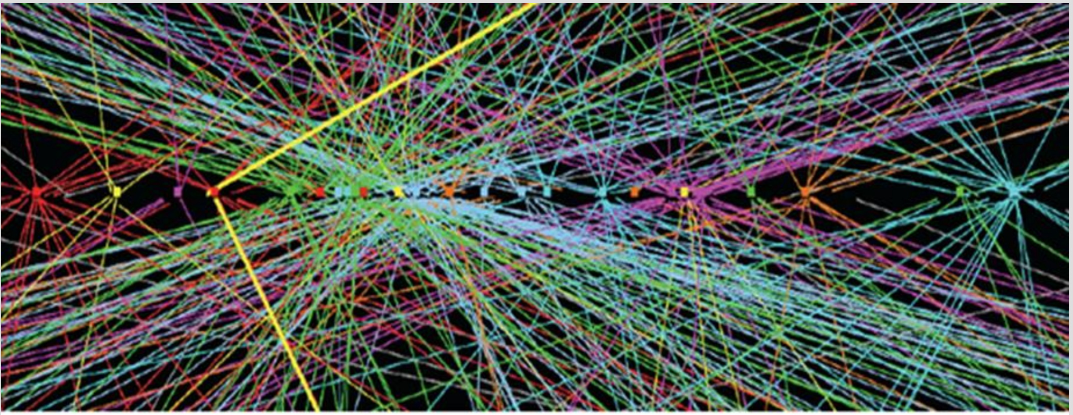
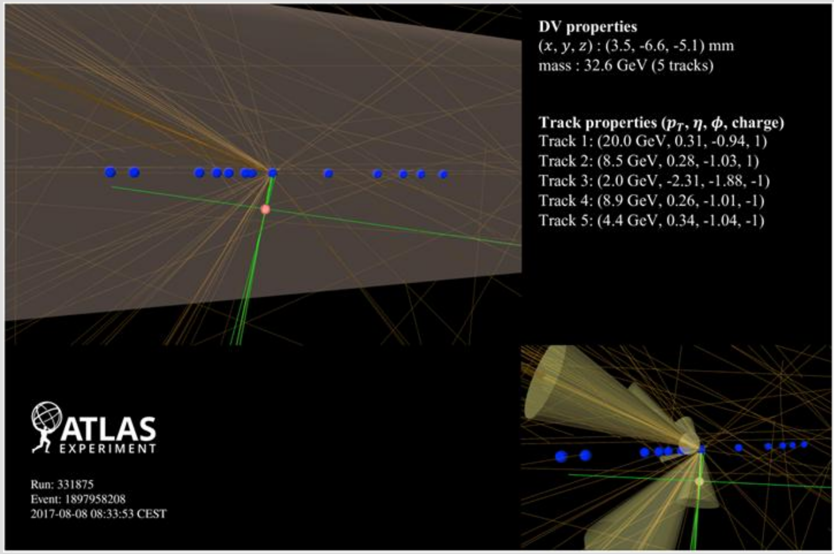
TrackOpt vertex with GNN

V. Kostyukhin
Siegen university

Reminder





DV properties
(x, y, z) : (3.5, -6.6, -5.1) mm
mass : 32.6 GeV (5 tracks)

Track properties (p_T, η, ϕ, charge)
Track 1: (20.0 GeV, 0.31, -0.94, 1)
Track 2: (8.5 GeV, 0.28, -1.03, 1)
Track 3: (2.0 GeV, -2.31, -1.88, -1)
Track 4: (8.9 GeV, 0.26, -1.01, -1)
Track 5: (4.4 GeV, 0.34, -1.04, -1)

Run: 331875
Event: 1897958208
2017-08-08 08:33:53 CEST

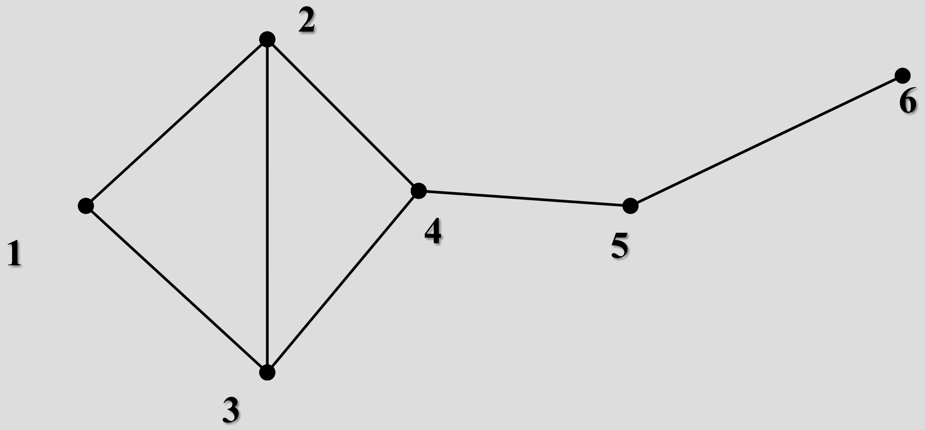
Problem:

Having a set of tracks find all vertices (track production points)

Solution idea:

1. Create a track-track compatibility graph using a priory physics knowledge, i.e. explicitly calculate a point of closest approach of the 2 tracks in 3D space and the track/vertex parameters at this point.
2. Based on this information, estimate a probability (GNN) that a given 2-track vertex is real
3. Use LMC algorithm to partition this graph with weighted edges

Graph



Example:
6 tracks (nodes)
7 possible 2-track vertices (edges)

Node vector of features(track parameters): td0, tZ, tPhi, tEta, tQoP, tTheta, tChi2, tNDoF, tCovD0, tCovZ,tCovD0Z, tSignif

Edge vector of features(2-track vertex): vChi2, vX, vY, vZ, vcovXX, vcovXY, vcovYY, vcovXZ, vcovYZ, vcovZZ, vsumPt, vEta, vPhi, vDZ, massPiPi, vtrue, vCharge, isGamma, isKs, isLambda + truth_label

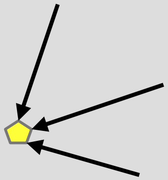
Prepared graphs saved in ROOT format (~1.5mb/ev, ~1400nodes/ev, ~26k edges/ev)

For the moment, to save CPU, for tracks closer that 3σ to the beamline no real vertex fit is done.
Compatibility is calculated based on Z track position on the beamline (1D fit, like in the PV finding paper).

Edge weights with DGL

Edge weights estimation is implemented in DGL

1) Node_hidden_state = NN{Concat(Node_features, Mean(Edge_features)}



2) Edge_weight = NN{ Concat(Node_hidden_state_i, Edge_features, Node_hidden_state_j)}

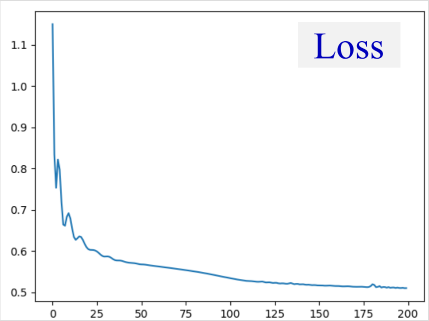
3) Node_hidden_state = NN{Concat(Node_features, Weighted_Mean(Edge_features)}
4) Upd_Edge_weight = NN{ Concat(Node_hidden_state_i, Edge_features, Node_hidden_state_j)}

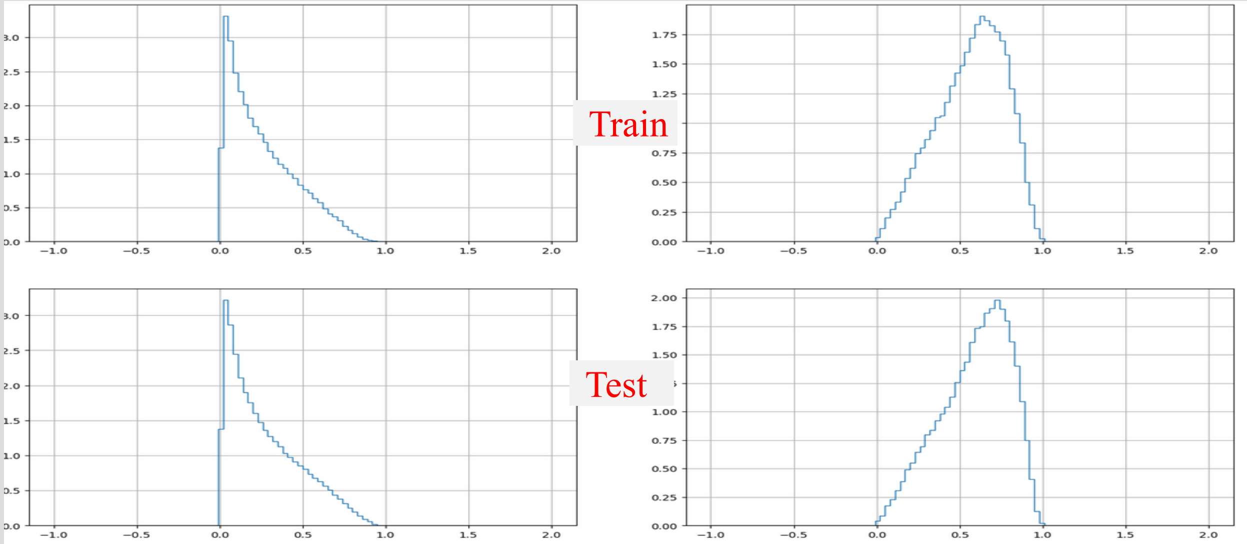
Steps (3),(4) happened to be not needed, final weights are practically the same.

Loss – binary cross-entropy (classification)
Activations – Mish + Sigmoid to get probability

DGL results (preliminary)

Training: 200 epoch, 100 graphs(events) batch train/test
loss= 0.5098(train), 0.5080(test) – no overtraining

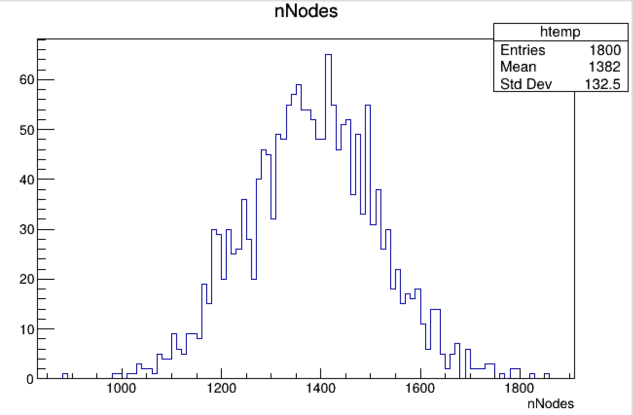




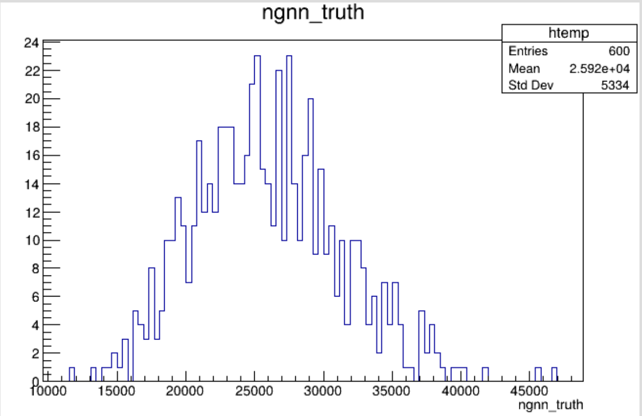
AUC(train)= 0.8265. AUC(test)= 0.8294 for both primary and secondary edges(aka. 2-track vertices)

For comparison - XGBoost classification of edges based on the same edge features: AUC ~ 0.75

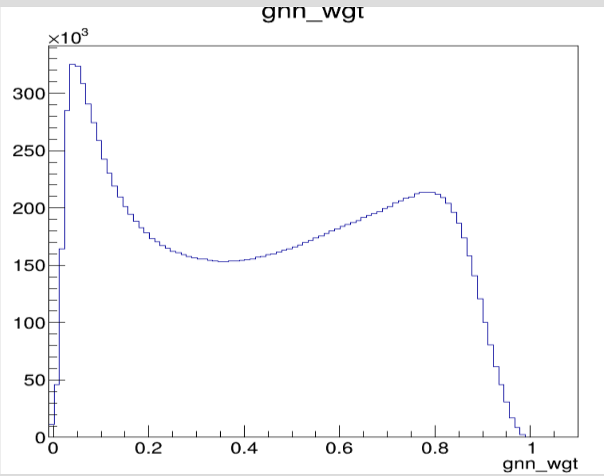
Number of nodes



Number of edges

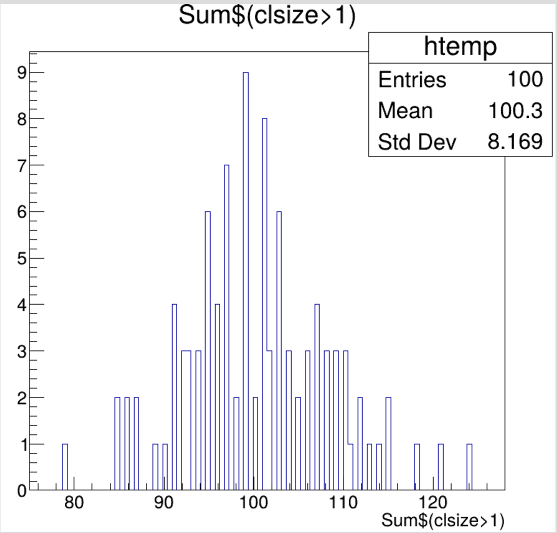


LMC after DGL

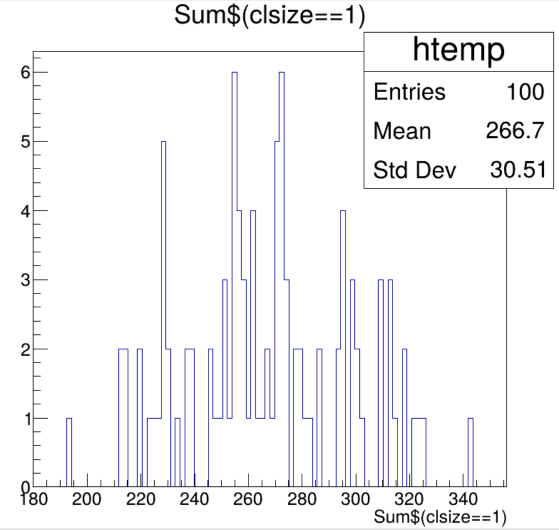


For LMC: weight=WGT$_{GNN}$-0.5

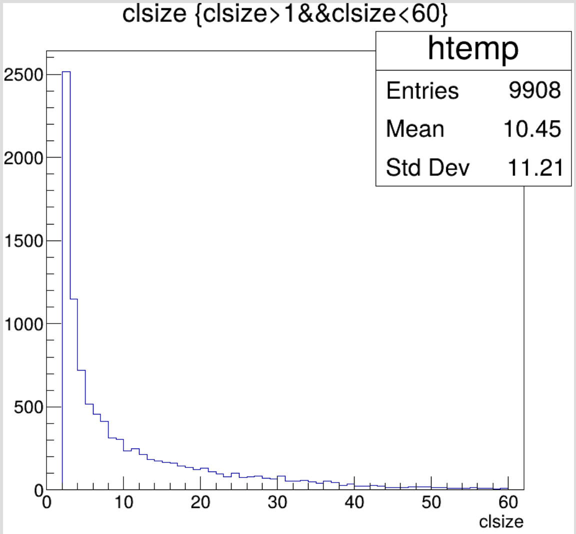
$N_{cluster}$ (size>1)



N_{single_track}

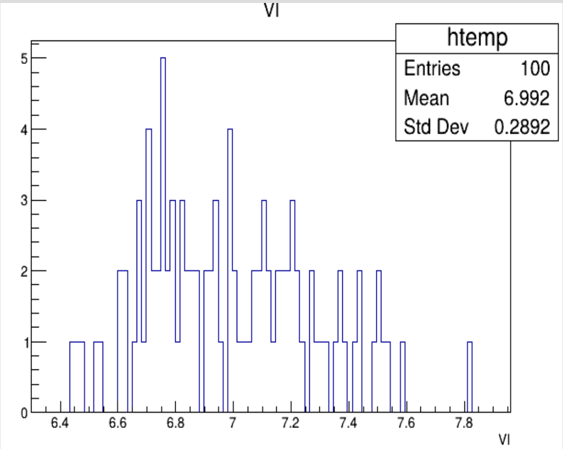


Cluster size

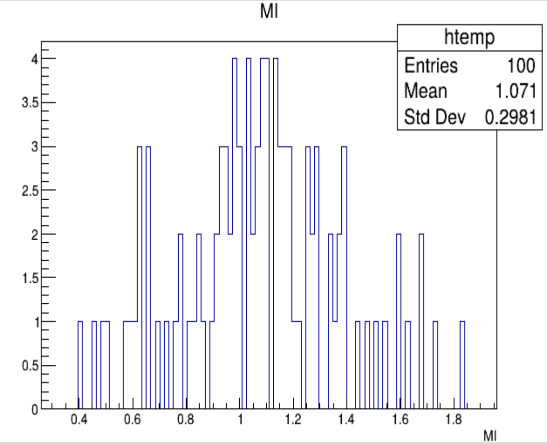


LMC metrics

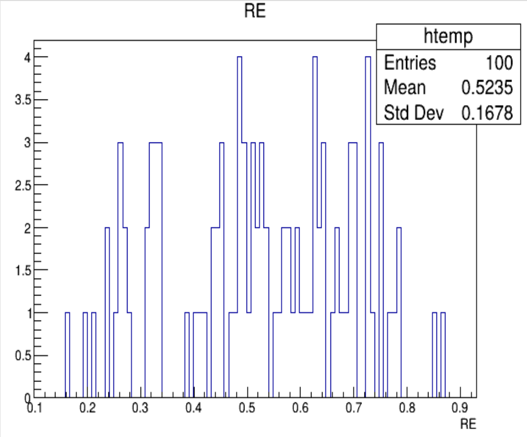
Variation of information



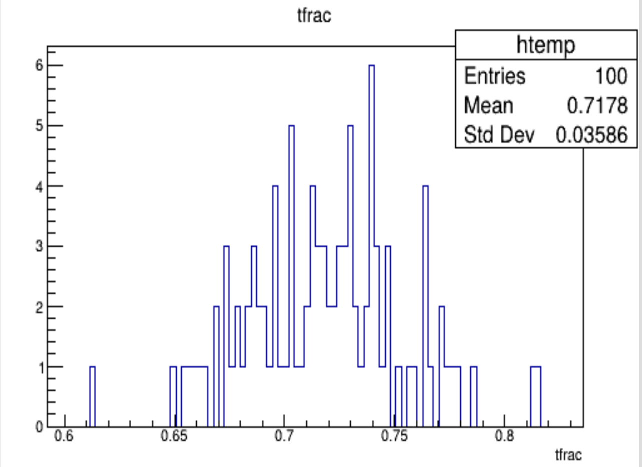
Mutual information



Rand index



Fraction of correctly resolved true edges



Conclusions and next steps

1) Basic machinery for data processing is created and works, although requires polishing/optimisation
2) R&D on GNN layer versions, LMC parameters, constraints, informative problem-dependent metrics, etc. can be started.
3) Current technical problems(work in progress):
    a. DGL doesn't export GNN models directly (ONNX?). PyTorch backend/export?
    b. UPROOT buffer sizes – limit number of graphs for saving
    c. …