GNNS FOR PARTICLE TRACKING

SOPHIA KORTE
FLORIDA STATE UNIVERSITY
MENTOR: ISOBEL OJALVO (PRINCETON)
LHC AND CMS

- LARGE HADRON COLLIDER ACCELERATES PROTONS AND COLLIDES THEM
- THE COMPACT MUON SOLENOID (CMS) DETECTOR IS AT ONE OF THE COLLISION POINTS
- CMS IS A GENERAL PURPOSE DETECTOR
BACKGROUND INFORMATION

HOW CMS WORKS

- BENDING PARTICLES
- TRACKING
- MEASURING ENERGY
- TRIGGER
HL-LHC AND SOME PROBLEMS

- HL-LHC
- LUMINOSITY AND PILEUP
- TRIGGER
- KALMAN FILTER
- TIME SPENT IN TRACKING INCREASES RAPIDLY WITH PILEUP
- NEED FOR OTHER ALGORITHMS
PROJECT SUMMARY

- Use Graph Neural Networks (GNNS) to reconstruct the trajectories of particles.
- Detector data can be represented as a graph, with the hits of particles as nodes and the possible trajectories as edges in the graph.
- The GNN is an interaction network (IN) that has three steps: graph building, finding the edge weights through edge classification, and then building the track.

- Initial goal - optimizing GNN
- My role - working on graph construction
BEGINNING

FIRST STEPS

- UNDERSTANDING CMS DATA - CMS_TTBAR_NOPILEUP SAMPLE
- STUDYING GRAPHS - HITS AS NODES AND TRACKS AS EDGES
- GRAPH CAN BE A PYTORCH DATA OBJECT
- UNDERSTANDING GNNS AND RUNNING IN

THREE STAGES OF GNN

- CONVERTING TRACKER DATA TO A HITGRAPH
- EDGE CLASSIFICATION, PREDICT EDGE WEIGHTS (PROBABILITIES THAT EDGES ARE REAL TRACK SEGMENTS)
- TRACK BUILDING, CUT EDGE WEIGHTS BELOW SOME THRESHOLD, APPLY CLUSTERING ALGORITHM
GRAPH BUILDING

- CMS EVENT FILES -> HITGRAPH
- ONE GRAPH = ONE EVENT
- NODE INFORMATION: R,
- EDGE INFORMATION: DR, DR, DZ, DPHI
- PARTICLE: PARTICLE ID, MOMENTUM AND ETA
- Y LABEL: 1 IF A TRUE EDGE, 0 OTHERWISE

```python
# Start with all possible pairs of hits
hit_pairs = hits1.reset_index().merge(hits2.reset_index(), on='evt', suffixes=('_1', '_2'))

# Compute line through the points

dphi = calc_dphi(hit_pairs.phi_1, hit_pairs.phi_2)
dz = hit_pairs.z_2 - hit_pairs.z_1
dr = hit_pairs.r_2 - hit_pairs.r_1
dr = hit_pairs.r_2 - hit_pairs.r_1
eta_1 = calc_eta(hit_pairs.r_1, hit_pairs.z_1)
eta_2 = calc_eta(hit_pairs.r_2, hit_pairs.z_2)
eta = eta_2 - eta_1
dr = np.sqrt(eta**2 + dphi**2)
phi_slope = dphi / dr
z0 = hit_pairs.z_1 - hit_pairs.r_1 * dz / dr

# Filter segments according to phi slope and z0 criteria

good_seg_mask = ((phi_slope.abs() < phi_slope_max) &
                 (z0.abs() < z0_max))

dr = dr[good_seg_mask]
dphi = dphi[good_seg_mask]
dz = dz[good_seg_mask]
dR = dR[good_seg_mask]

return hit_pairs[good_seg_mask], dr, dphi, dz, dR
```
RESULTS FROM GRAPH BUILDING

- RZ PLOTS: BLACK FOR FALSE EDGES, BLUE FOR TRACK SEGMENTS
MORE GRAPH BUILDING

- GROUP CREATED A PYTHON PACKAGE THAT CAN BE PIP INSTALLED
  - HTTPS://GITHUB.COM/GAGEDEZOORT/GNN_TRACKING
- GRAPH CONSTRUCTION SHOULD BE CONFIGURABLE AS PYTHON OBJECT
- MAKES OPTIMIZATION EASIER
- TRACKML GEOMETRY HARDCODED
- EVENTPLOTTER CLASS - ETA-PHI, RZ AND UV PLOTS
- POINTCLOUDBUILDER CLASS - POINT CLOUD CONSTRUCTION, BREAK UP HITS INTO SECTORS
- GRAPH CONSTRUCTION - EXTEND EDGES BETWEEN HITS IN THE POINT CLOUDS
RESULTS FROM EVENTPLOTTER

- Plots can be obtained from CMS event files, no need for building the graph first.
THINGS THAT I DID AND LEARNED

- Used CMS data to build graphs and run in
- Started with graphs built by considering all pairs of hits and drawing an edge between them if the edge has certain geometric properties, ended with point clouds for graph construction

- Plots from EventPlotter
- Worked on adapting the PointCloudBuilder for the CMS data until a certain point - time constraint and need for more coding experience to continue working on the point clouds and the GraphPlotter class

- CMS data more organized than TrackML data
- Improved Python skills
THANK YOU!

- MY MENTOR, ISOBEL OJALVO
- ALL OF THE MEMBERS OF OUR GROUP, ESPECIALLY GAGE DEZOORT
- IRIS HEP