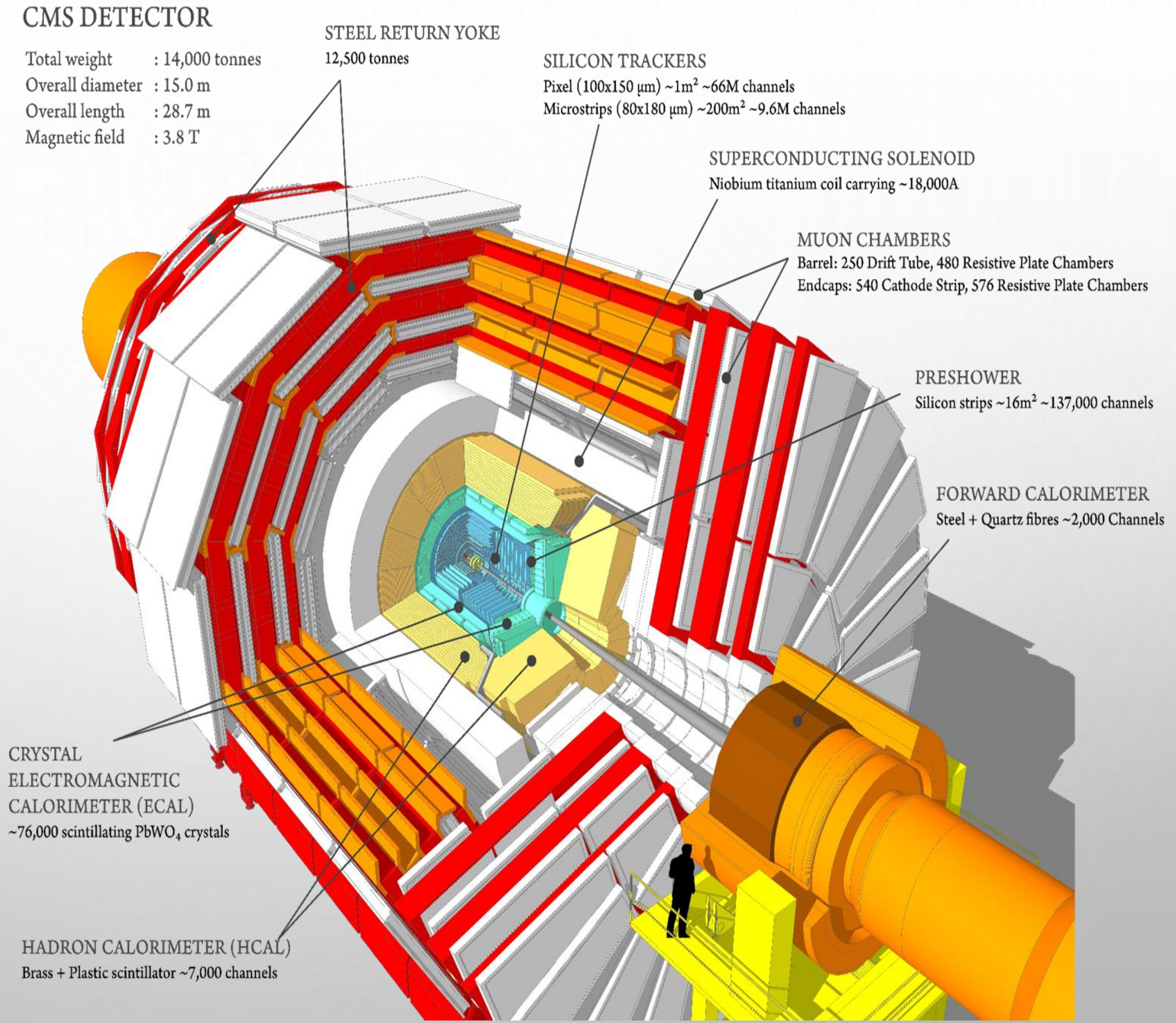
# GNNS FOR PARTICLE TRACKING

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

# LHC AND CMS

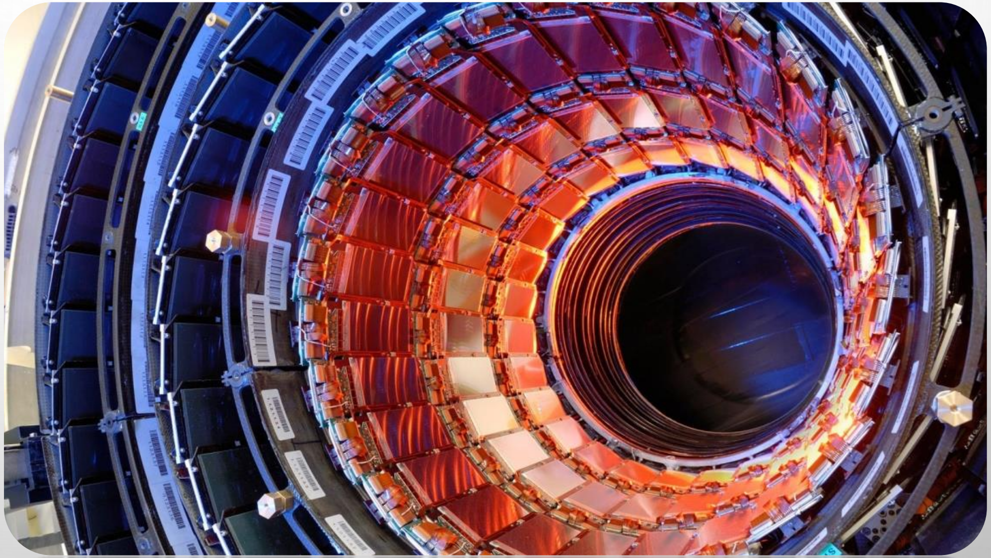
- LARGE HADRON COLLIDER ACCELERATES PROTONS AND COLLIDES THEM
- THE COMPACT MUON SOLENOID (CMS) DETECTOR IS AT ONE OF THE COLLISION POINTS
- CMS IS A GENERAL PURPOSE DETECTOR



# 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, THEN BUILDING THE TRACK.

- INITIAL GOAL - OPTIMIZING GNN
- MY ROLE - WORKING ON GRAPH CONSTRUCTION

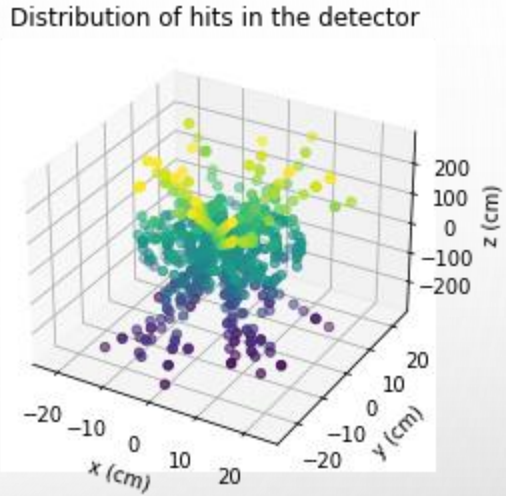
# BEGINNING

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

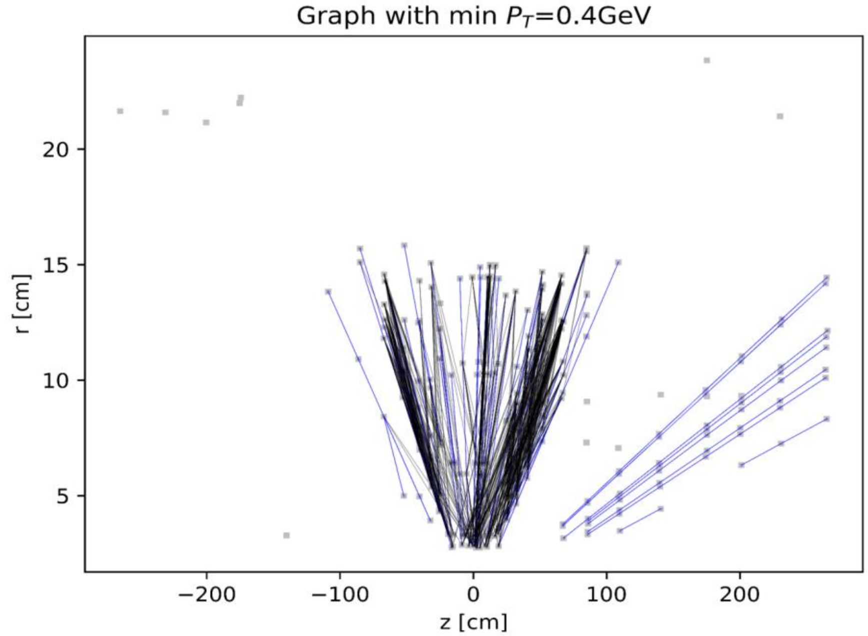
- CMS EVENT FILES -> HITGRAPH
- ONE GRAPH = ONE EVENT
- NODE INFORMATION: R,
- EDGE INFORMATION: DR, DR, DZ, DPHI
- PARTICLE: PARTICLE ID, MOMENTUM AND ETA
- Y LABEL: 1 IF A TRUE EDGE, 0 OTHERWISE

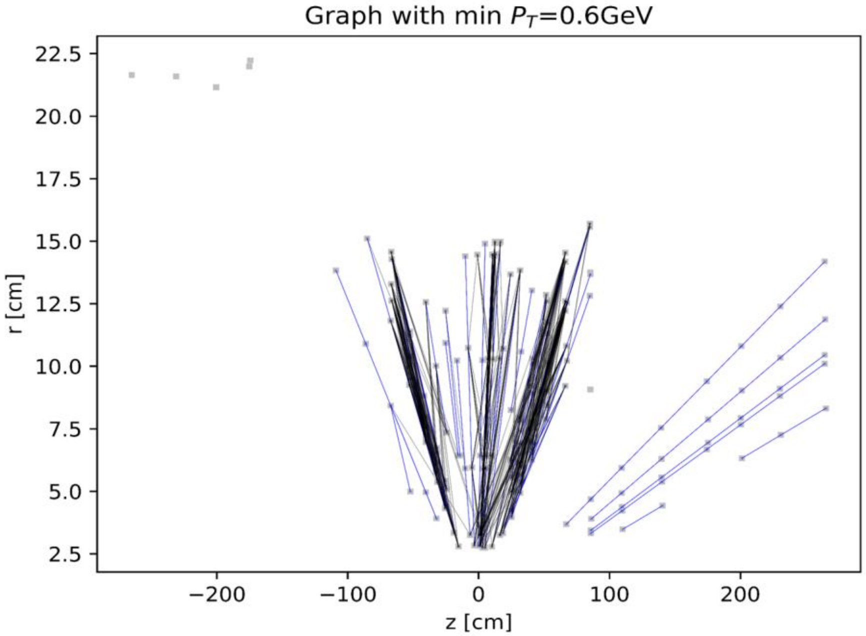
```
# Start with all possible pairs of hits
hit_pairs = hits1.reset_index().merge(hits2.reset_index(), on='evt', suffixes=('_1', '_2'))

#print(hit_pairs)
# 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
eta_1 = calc_eta(hit_pairs.r_1, hit_pairs.z_1)
eta_2 = calc_eta(hit_pairs.r_2, hit_pairs.z_2)
deta = eta_2 - eta_1
dR = np.sqrt(deta**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



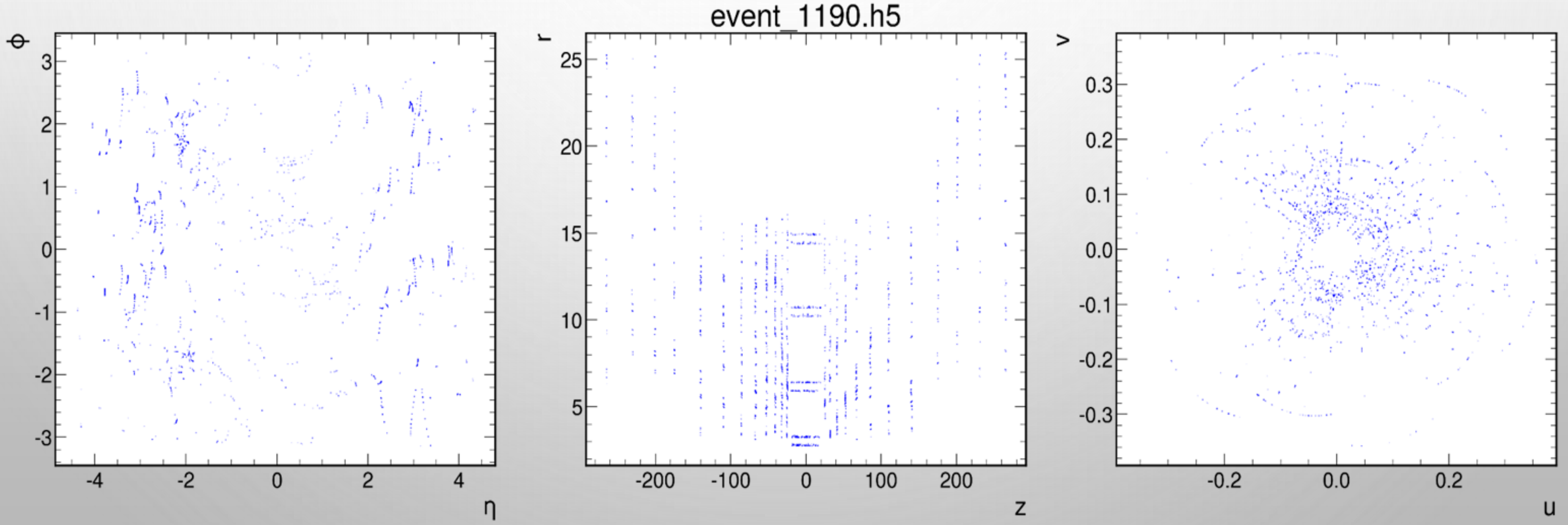



# MORE GRAPH BUILDING

- 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