Optional Task

Two types of variables are there:

- Variables related to detector hits by a particle
- Overall path variables (Road variables)

The variables related to detector hits give rise to point cloud dataset where every point has information about its x, y and z coordinate, bending angle, and about front or rear hit. Edge convolution, along with pooling and aggregation, can be used to generate representation for this data.

The main thing to consider above is - how the points should be connected.
Some potential ways are –
1) directed edges in the way a particle hits the detectors
2) connect all the points (i.e., fully connected graph) by a particle.
Since the message passed for aggregation is $h_\theta(x_i, x_j)$, a useless interaction in way(2) will not harm the representation, and since there are only a few points in a cloud, it will also not lead to the computational burden. So, way(2) is advisable as it will consider every interaction.

For the second type of variables, we can use dense layers to generate representations. Then, we can concatenate these representations with representations generated above and can feed them to a few other dense layers to generate output.

Proposed Architecture

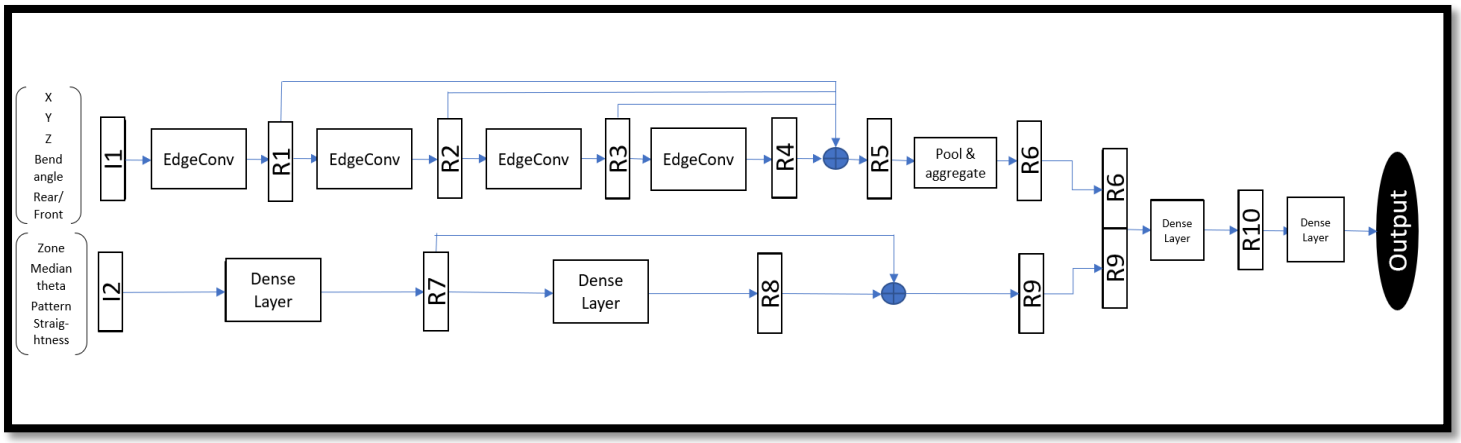

Fig -1 Proposed Model architecture. The upper portion is a Graph convolutional neural network that uses Edge convolution, pooling and aggregation to generate graph representation. While at the bottom, there is a dense neural network, to which other graph variables are fed. The output representations of both the models are concatenated and fed to a few other dense layers, which generate final output. The input to Graph CNN are – x-coordinate, y-coordinate, z-coordinate, bend angle, and front/ rear hit. The input to the dense network are – zone, median theta, and pattern straightness.

Note – If the image is not visible, click here.

Pseudocode

I1 ← Input (graph_nodes)

I2 ← Input (road_variables)

Edges = `[(i,j) for i in range(len(I1)) for j in range(len(I1))]`

R1 = EdgeConv_1 (I1, Edges)

R2 = EdgeConv_2 (R1, Edges)

R3 = EdgeConv_3 (R2, Edges)

R4 = EdgeConv_4 (R3, Edges)

R5 = R1 + R2 + R3 + R4

R6 = Pool_and_Aggregate (R5)

R7 = DenseLayer_1 (I2)

R8 = DenseLayer_2 (R7)

R9 = R7 + R8

R10 = DenseLayer_3 (Concatenate (R6, R9))

Output = DenseLayer_4 (R10)

References

- Fey, Matthias and Lenssen, Jan E. 2019. Fast Graph Representation Learning with PyTorch Geometric. ICLR Workshop on Representation Learning on Graphs and Manifolds.
- Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, Justin M. Solomon. 2019. Dynamic Graph CNN for Learning on Point Clouds. CoRR.