# Task 3

**IDEA**:-

For the binary classification task of differentiating between a gluon and quark particle from point cloud, a dynamic edge convolutional neural network is used. In this type of model, an edge convolution operation is performed between features of nodes connected by an edge. A graph(edge indices) is generated dynamical on each batch of training data using k-nearest neighbor approach. Further details can be found in this paper – https://arxiv.org/pdf/1801.07829.pdf.
I think edge convolution is well suited for this task because it tries to extracts the maximal features present in the proximity of node. The value of convolution is higher for nodes with similar features and we assume that any 2 gluon particle or any 2 quark particle have similar features(unless they don't!). So, edge convolution would predict the correct label even if the particle is surrounded by majority of particles from the other label.
Also, bagging approach is used in which final prediction is the average of weighted prediction made by 3 different models. This reduces the variance in our prediction.

**DATA**:-

For the task, the first 8 files from QG_Jets_(0-7) were used(8e5 samples). First, the data was flattened using np.reshape(). Then the whole dataset was divided into 3 subsets of size 2e5 which were fed into 3 different models for training and 2 subsets of size 1e5 of which one was used for validation and the other for testing.

**MODEL**:-

Three different models were trained on different subsets of data. The architecture of all the 3 models was same with differences in learning rate, dropout ratio and value of k for constructing dynamic k-nn graph.

| Model no. | Dropout ratio | Value of k | Learning rate |
|---|---|---|---|
| 1 | 0.3 | 30 | 0.0004 |
| 2 | 0.3 | 25 | 0.0002 |
| 3 | 0.2 | 20 | 0.0003 |

Each model consisted of 5 dynamic edge convolutional layers of size 128, 256, 128, 32, 2. Each layer is followed by dropout and uses ReLU activation except for the last layer which uses softmax activation. Adam optimizer with default parameters along with above learning rate is used for every model. CrossEntropy loss is used in all the 3 models.

**TRAINING AND PREDICTION**:-

The 3 models were trained on Google Colab using Pytorch v1.4, pytorch-geometric and CUDA 10.1. Training data was normalized before training. Batch size = 100 and no.of epochs = 75 was used during training of each model. 3 models were trained on 3 different notebooks on google

colab which took about 2.5-3 hours for each model. After training, a weight parameter was generated for each model based on their accuracy on the validation set. The final prediction on the test set was made by taking a weighted average of the prediction of each model.

Accuracy achieved = 77.568%

**CHOICE OF HYPERPARAMETERS** :-

The size of training subset = 2e5, batch size = 100 and no.of epochs = 75 were chosen due to time and resource constraints. As google provides resources for about 10-11 hours in a day, training on larger samples would have been difficult as it would have involved saving the model quite often. So to ease the process of training, the above hyperparameters were chosen. The choice for k in constructing a k-nn graph was taken from the paper. Parameters of the model(architecture, learning rate and dropout ratio) were chosen by trail and error method by checking which values produced better results.

# IDEA FOR OPTIONAL TASK

Using the idea of edge convolution, task 1 and 2 could also be solved similarly. For predicting the momentum of a particle, if we can make the assumption that particles with higher momentum and lower momentum particles have different set of features but have similar features among themselves then edge convolution could be implemented as before. Only in this case instead of selecting max features, we would have to consider average features of all the neighbors for a node.