GNN Settings

We adopt a method similar to GraphSAGE for deep learning-based particle classification using GNNs. In our experiments (Exp A to D), the classification targets are nodes originating from distinct graphs, where the graph structures are constructed using a modified Voronoi method with the parameter $\mathcal{A} = 0.55$, which has been identified as optimal in traditional machine learning approaches. Each node is initially represented by a set of input features, which serve as the input to the GNN for classification.

The feature matrix of the training samples is defined as $X \in \mathbb{R}^{M \times d}$, where M denotes the total number of training samples (i.e., all labeled nodes), and $d = 10$ represents the dimensionality of the node features. For each sample i, the initial input feature vector is denoted as $X_{(i)}$, corresponding to the attributes of the node in its respective graph. Specifically, $X_{(i)}$ comprises ten Type C descriptors, including the clustering coefficient and nine distinct centrality measures. These features serve as the initial input to GNN and provide structural information about the node within its graph.

Each sample is associated with a specific graph, whose structure is represented by the adjacency matrix $A_{(i)} \in \{0,1\}^{N_{(i)} \times N_{(i)}}$, where $N_{(i)}$ represents the total number of nodes in the graph, and $A_{(i)}$ defines the connectivity within it. The adjacency matrices are generated using the modified Voronoi method to capture the topology of node relationships. Notably, GNN computations are confined to the graph of each sample, ensuring that information does not propagate across different graphs.

Within the GNN, all samples share a common set of GNN weight matrices, and node embeddings are computed as follows:

$$H_{(i)}^{(0)} = X_{(i)} \tag{1}$$

$$H_{(i)}^{(l+1)} = \sigma\left(W^{(l)} \cdot \mathcal{D}\left(\frac{1}{|\mathcal{N}_{(i)}(j)|} \sum_{j \in \mathcal{N}_{(i)}} H_{(j)}^{(l)}\right)\right) \tag{2}$$

$$Z_{(i)} = H_{(i)}^{(L)} \tag{3}$$

where $W^{(l)}$ is the shared weight matrix of the l-th GNN layer, and $\mathcal{N}_{(i)}(j)$ denotes the set of neighbors of sample i in its respective graph. Since all graphs share the same GNN architecture and parameters, the model leverages information from multiple graphs to optimize a unified set of weight matrices. Ultimately, sample i undergoes GNN processing to generate its corresponding node embedding $Z_{(i)}$.

Following node embedding extraction, classification is performed using a multilayer perceptron (MLP):

$$\hat{Y}_{(i)} = W_{\text{MLP}}^{(L_{\text{MLP}})} \cdot Z_{(i)} + b_{\text{MLP}}^{(L_{\text{MLP}})} \tag{4}$$

$$P(Y_{(i)}) = \text{Softmax}(\hat{Y}_{(i)}) \tag{5}$$

All node embeddings $Z_{(i)}$ are processed through the same MLP classifier. Although each sample's corresponding message passing is restricted to its respective graph, the MLP training process treats all samples as part of a unified classification task, without distinguishing their originating graphs.

The model is trained by minimizing the cross-entropy loss across all training samples:

$$\mathcal{L} = -\frac{1}{M} \sum_{i=1}^{M} \left(y_{(i)} \log p_{(i)} + (1 - y_{(i)}) \log(1 - p_{(i)})\right) \tag{6}$$

where $y_{(i)}$ represents the true label of sample i, and $p_{(i)}$ denotes the predicted probability of its assigned class. The loss function is computed globally, meaning that all training samples contribute to optimizing the parameters of both GNN and the MLP, ensuring optimal classification performance across all data points.

In this task, the GNN is based on the GraphSAGE architecture and implemented using Pytorch Geometric. The process of model selection is performed using 5-fold cross-validation. To optimize hyperparameters, we employ Bayesian optimization with Optuna, exploring a search space that includes the number of GNN layers, hidden dimensions, dropout rates, learning rates, and the configuration of the MLP classifier. The optimization process runs for 100 iterations, where each iteration refines the search strategy based on prior evaluations and selects new hyperparameter sets for assessment using a GNN trained with Pytorch Geometric. Finally, the model is trained with the optimal hyperparameters on the full training set in a final training run to obtain the optimal model. After obtaining the optimal model, its performance is evaluated on an independent test set. The construction of the training and test sets for Exp A to D is detailed in the paper.