**Supplementary Table S1.** **The source code and hyperparameters of the neural network models**

1. Python source code of the neural networkmodels in this study

import dgl

import torch.nn.functional as F

import torch

import torch as t

import dgl.nn.pytorch as dglnn

import torch.nn as nn

class GCN\_attention(nn.Module):

def \_\_init\_\_(self, config, hidden\_dim):

super(GCN\_attention, self).\_\_init\_\_()

self.conv1 = dglnn.GraphConv(config.d, hidden\_dim, norm='none', allow\_zero\_in\_degree=True)

self.conv2 = dglnn.GraphConv(hidden\_dim, hidden\_dim, norm='none', allow\_zero\_in\_degree=True)

self.multihead\_attn = nn.MultiheadAttention(hidden\_dim, num\_heads=2)

self.classify = MLP(inSize=hidden\_dim, outSize=config.n\_classes)

self.moduleList = nn.ModuleList([self.conv1, self.conv2, self.multihead\_attn, self.classify])

def forward(self, g, h):

# Apply graph convolution networks and activation functions

h = F.relu(self.conv1(g, h, edge\_weight=g.edata['weight']))

h = F.relu(self.conv2(g, h, edge\_weight=g.edata['weight']))

with g.local\_scope():

g.ndata['h'] = h

# Use the average readout to get the graph representation

hg = dgl.mean\_nodes(g, 'h')

# Applying two-head attention mechanism

attention\_output, \_ = self.multihead\_attn(hg.unsqueeze(0), hg.unsqueeze(0), hg.unsqueeze(0), key\_padding\_mask=None, attn\_mask=None)

hg = attention\_output.squeeze(0)

return self.classify(hg.float())

class MLP(nn.Module):

def \_\_init\_\_(self, inSize, outSize, hiddenList=[], dropout=0.5, actFunc=nn.ReLU):

super(MLP, self).\_\_init\_\_()

layers = nn.Sequential()

for i,os in enumerate(hiddenList):

layers.add\_module(str(i\*2), nn.Linear(inSize, os))

layers.add\_module(str(i\*2+1), actFunc())

inSize = os

self.hiddenLayers = layers

self.dropout = nn.Dropout(p=dropout)

self.out = nn.Linear(inSize, outSize)

def forward(self, x):

x = self.hiddenLayers(x)

return self.out(self.dropout(x))

1. Hyperparameters used for training models

|  |  |
| --- | --- |
| Hyperparameter | Value |
| K value of k-mer | 5 |
| Node feature dimension | 128 |
| Hidden layer of the graph convolutional networks | 64 |
| Batch size | 8 |
| Epochs | 1000 |
| Learning rate | 0.003 |
| K of Fold | 10 |
| Early Stop | 100 |