结果

```
classification task
HierarchicalGraphMatchNetwork(
  (gc1): DenseGCNConv(11, 100)
  (gc2): DenseGCNConv(100, 100)
  (gc3): DenseGCNConv(100, 100)
  (gc_cross1): DenseGCNConv(11, 100)
  (gc_cross2): DenseGCNConv(100, 100)
  (gc_cross3): DenseGCNConv(100, 100)
  (fc1): Linear(in_features=100, out_features=100, bias=True)
  (fc2): Linear(in_features=100, out_features=100, bias=True)
  (global_fc_agg): Linear(in_features=100, out_features=100, bias=True)
  (agg_bilstm): LSTM(100, 100, batch_first=True, bidirectional=True)
  (agg_lstm): LSTM(100, 100, batch_first=True)
  (nn): Linear(in_features=200, out_features=100, bias=True)
)
/mnt/share/CGMN/CGMN/data/CFG/OpenSSL_11ACFG_min3_max13/acfgSSL_6/split_by_own_remove_space.json
/mnt/share/CGMN/CGMN/data/CFG/OpenSSL_11ACFG_min3_max13/acfgSSL_6 Train: 35 graphs, 35 functions
/mnt/share/CGMN/CGMN/data/CFG/OpenSSL_11ACFG_min3_max13/acfgSSL_6 Dev  : 27 graphs, 27 functions
/mnt/share/CGMN/CGMN/data/CFG/OpenSSL_11ACFG_min3_max13/acfgSSL_6 Test : 27 graphs, 27 functions

  /opt/conda/lib/python3.7/site-packages/sklearn/utils/validation.py:72: Dat
    return f(**kwargs)
  Before Training: the test AUC score is: 0.8991769547325102
  Before Training: the test lrAUC score is: 0.8607954545454546
  Epoch {} : 1
          Training Contrastive: 0/7: index = 2 loss = 2.9879837036132812
          Training Contrastive: 1/7: index = 6 loss = 3.110060453414917
          Training Contrastive: 2/7: index = 4 loss = 3.006560802459717
          Training Contrastive: 3/7: index = 5 loss = 2.8688554763793945
          Training Contrastive: 4/7: index = 0 loss = 2.789565086364746
          Training Contrastive: 5/7: index = 3 loss = 2.8951687812805176
          Training Contrastive: 6/7: index = 1 loss = 2.9641880989074707
              /opt/conda/lib/python3.7/site-packages/sklearn
                return f(**kwargs)
              Final Testing: the cosine AUC score is: 1.0
              Final testing: the lr AUC score is: 1.0
              Best_valid_auc =  0.9814814814814815
              Best_test_auc =  1.0
              END
```

```
        Training Contrastive: 90/99: index = 5 loss = 5.416337966918945
/opt/conda/lib/python3.7/site-packages/sklearn/utils/validation.py:72: DataConversi
onWarning: A column-vector y was passed when a 1d array was expected. Please change
 the shape of y to (n_samples, ), for example using ravel().
  return f(**kwargs)
Now, the evaluation score is: 0.6283779919357144
Now, the evaluation lr score is: 0.5617487782327302
/opt/conda/lib/python3.7/site-packages/sklearn/utils/validation.py:72: DataConversi
onWarning: A column-vector y was passed when a 1d array was expected. Please change
 the shape of y to (n_samples, ), for example using ravel().
  return f(**kwargs)
Now, the test score is: 0.6577618814223904
Now, the test lr score is: 0.578509259111462
EPOCH 50/50:     Contrastive loss = 4.942800045013428 @ 2024-09-23 02:06:49.329964
best_val_auc =  0.6294646687065687
best_lr_auc =  0.6330426356589147
/opt/conda/lib/python3.7/site-packages/sklearn/utils/validation.py:72: DataConversi
onWarning: A column-vector y was passed when a 1d array was expected. Please change
 the shape of y to (n_samples, ), for example using ravel().
  return f(**kwargs)
Final valid: the cosine AUC score is: 0.6294646687065687
Final valid: the lr AUC score is: 0.593448397323001
/opt/conda/lib/python3.7/site-packages/sklearn/utils/validation.py:72: DataConversi
onWarning: A column-vector y was passed when a 1d array was expected. Please change
 the shape of y to (n_samples, ), for example using ravel().
  return f(**kwargs)
Final Testing: the cosine AUC score is: 0.6571640498242294
Final testing: the lr AUC score is: 0.5792340952944326
Best_valid_auc =  0.6294646687065687
Best_test_auc =  0.6571640498242294
END
```

使用样本数据得到的结果如上

```
root@csip-090:/mnt/share/CGMN/CGMN/src# python Calculate_similarity.py
/mnt/share/CGMN/CGMN/data/CFG/OpenSSL_11ACFG_min3_max13/valid/QFN28LK(Cu)-90-450Rev1_2.txt
/mnt/share/CGMN/CGMN/data/CFG/OpenSSL_11ACFG_min3_max13/valid/QFN9LM(NiPdAu)-439Rev1_1.txt
classification task
图对的相似度为: 0.9596171379089355
root@csip-090:/mnt/share/CGMN/CGMN/src# python Calculate_similarity.py
/mnt/share/CGMN/CGMN/data/CFG/OpenSSL_11ACFG_min3_max13/valid/QFN28LK(Cu)-90-450Rev1_2.txt
/mnt/share/CGMN/CGMN/data/CFG/OpenSSL_11ACFG_min3_max13/valid/QFN9LM(NiPdAu)-439Rev1_2.txt
classification task
图对的相似度为: 0.9633334875106812
root@csip-090:/mnt/share/CGMN/CGMN/src# python Calculate_similarity.py
/mnt/share/CGMN/CGMN/data/CFG/OpenSSL_11ACFG_min3_max13/valid/QFN28LK(Cu)-90-450Rev1_2.txt
/mnt/share/CGMN/CGMN/data/CFG/OpenSSL_11ACFG_min3_max13/valid/QFN9LM(NiPdAu)-439Rev1_4.txt
classification task
图对的相似度为: 0.8332219123840332
root@csip-090:/mnt/share/CGMN/CGMN/src# python Calculate_similarity.py
/mnt/share/CGMN/CGMN/data/CFG/OpenSSL_11ACFG_min3_max13/valid/QFN28LK(Cu)-90-450Rev1_2.txt
/mnt/share/CGMN/CGMN/data/CFG/OpenSSL_11ACFG_min3_max13/QFN9LM(NiPdAu)-439Rev1_5.txt
Traceback (most recent call last):
```

```
root@csip-090:/mnt/share/CGMN/CGMN/src# python Calculate_similarity.py
/mnt/share/CGMN/CGMN/data/CFG/OpenSSL_11ACFG_min3_max13/valid/QFN28LK(Cu)-90-450Rev1_2.txt
/mnt/share/CGMN/CGMN/data/CFG/OpenSSL_11ACFG_min3_max13/valid/QFN9LM(NiPdAu)-439Rev1_5.txt
classification task
图对的相似度为: 0.9405058026313782
root@csip-090:/mnt/share/CGMN/CGMN/src# python Calculate_similarity.py
/mnt/share/CGMN/CGMN/data/CFG/OpenSSL_11ACFG_min3_max13/valid/QFN28LK(Cu)-90-450Rev1_2.txt
/mnt/share/CGMN/CGMN/data/CFG/OpenSSL_11ACFG_min3_max13/valid/QFN28LK(Cu)-90-450Rev1_2.txt
classification task
图对的相似度为: 1.0
```

仅仅使用余弦相似度，不经过深度网络

```
root@csip-090:/mnt/share/CGMN/CGMN/src# python only_cos.py
/mnt/share/CGMN/CGMN/data/CFG/OpenSSL_11ACFG_min3_max13/valid/QFN28LK(Cu)-90-450Rev1_2.txt
/mnt/share/CGMN/CGMN/data/CFG/OpenSSL_11ACFG_min3_max13/valid/QFN9LM(NiPdAu)-439Rev1_1.txt
图对的相似度为: 0.8760697841644287
```

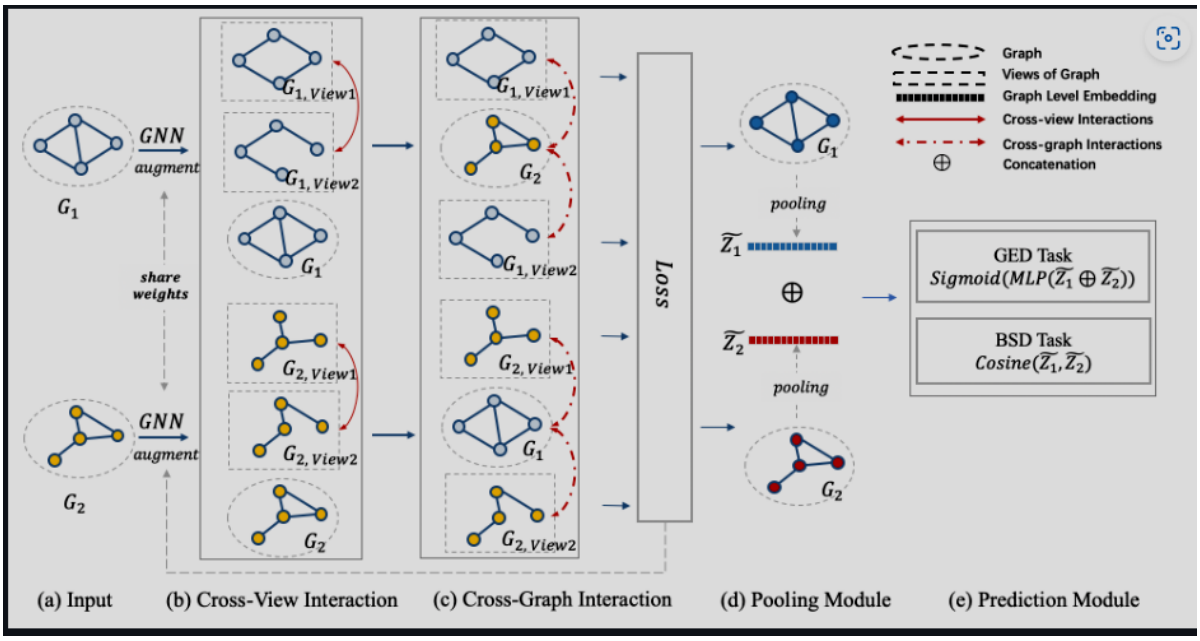
```
root@csip-090:/mnt/share/CGMN/CGMN/src# python only_cos.py
/mnt/share/CGMN/CGMN/data/CFG/OpenSSL_11ACFG_min3_max13/valid/QFN28LK(Cu)-90-450Rev1_2.txt
/mnt/share/CGMN/CGMN/data/CFG/OpenSSL_11ACFG_min3_max13/valid/QFN9LM(NiPdAu)-439Rev1_2.txt
图对的相似度为: 0.8972756266593933
root@csip-090:/mnt/share/CGMN/CGMN/src# python only_cos.py
/mnt/share/CGMN/CGMN/data/CFG/OpenSSL_11ACFG_min3_max13/valid/QFN28LK(Cu)-90-450Rev1_2.txt
/mnt/share/CGMN/CGMN/data/CFG/OpenSSL_11ACFG_min3_max13/valid/QFN9LM(NiPdAu)-439Rev1_4.txt
图对的相似度为: 0.431698739528656
root@csip-090:/mnt/share/CGMN/CGMN/src# python only_cos.py
/mnt/share/CGMN/CGMN/data/CFG/OpenSSL_11ACFG_min3_max13/valid/QFN28LK(Cu)-90-450Rev1_2.txt
/mnt/share/CGMN/CGMN/data/CFG/OpenSSL_11ACFG_min3_max13/valid/QFN9LM(NiPdAu)-439Rev1_5.txt
图对的相似度为: 0.8378084301948547
root@csip-090:/mnt/share/CGMN/CGMN/src# python only_cos.py
/mnt/share/CGMN/CGMN/data/CFG/OpenSSL_11ACFG_min3_max13/valid/QFN28LK(Cu)-90-450Rev1_2.txt
/mnt/share/CGMN/CGMN/data/CFG/OpenSSL_11ACFG_min3_max13/valid/QFN28LK(Cu)-90-450Rev1_2.txt
图对的相似度为: 1.0
```

论文介绍

Cgmn: A contrastive graph matching network for self-supervised graph similarity learning CCF A

D Jin, L Wang, Y Zheng, X Li, F Jiang, W Lin, S Pan

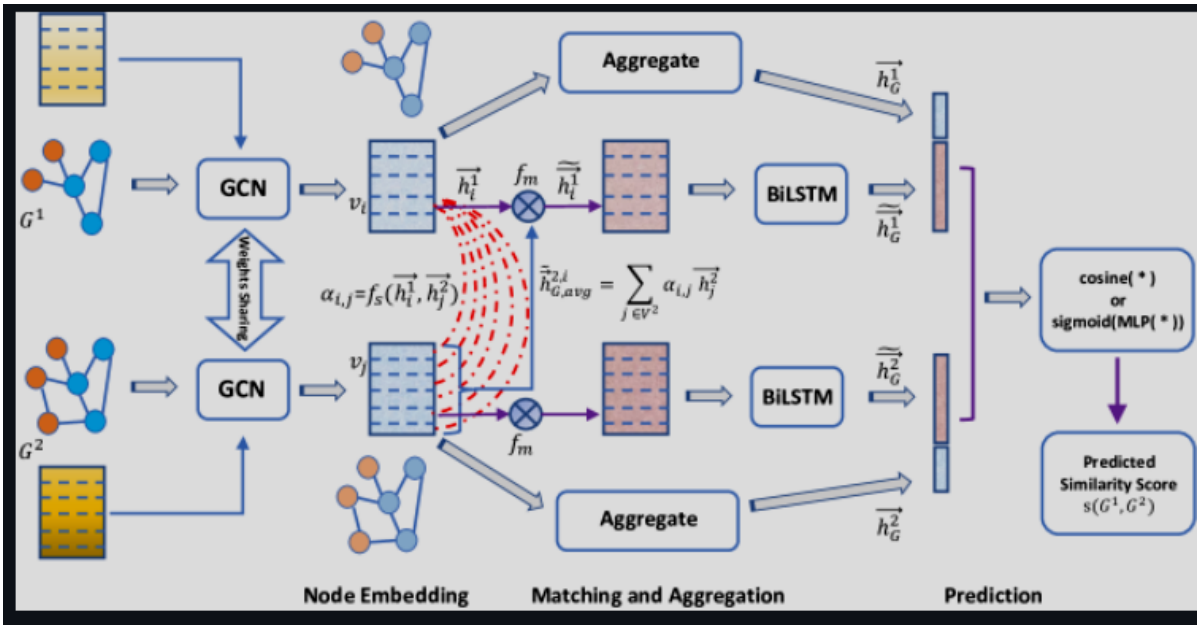
arXiv preprint arXiv:2205.15083, 2022 · arxiv.org



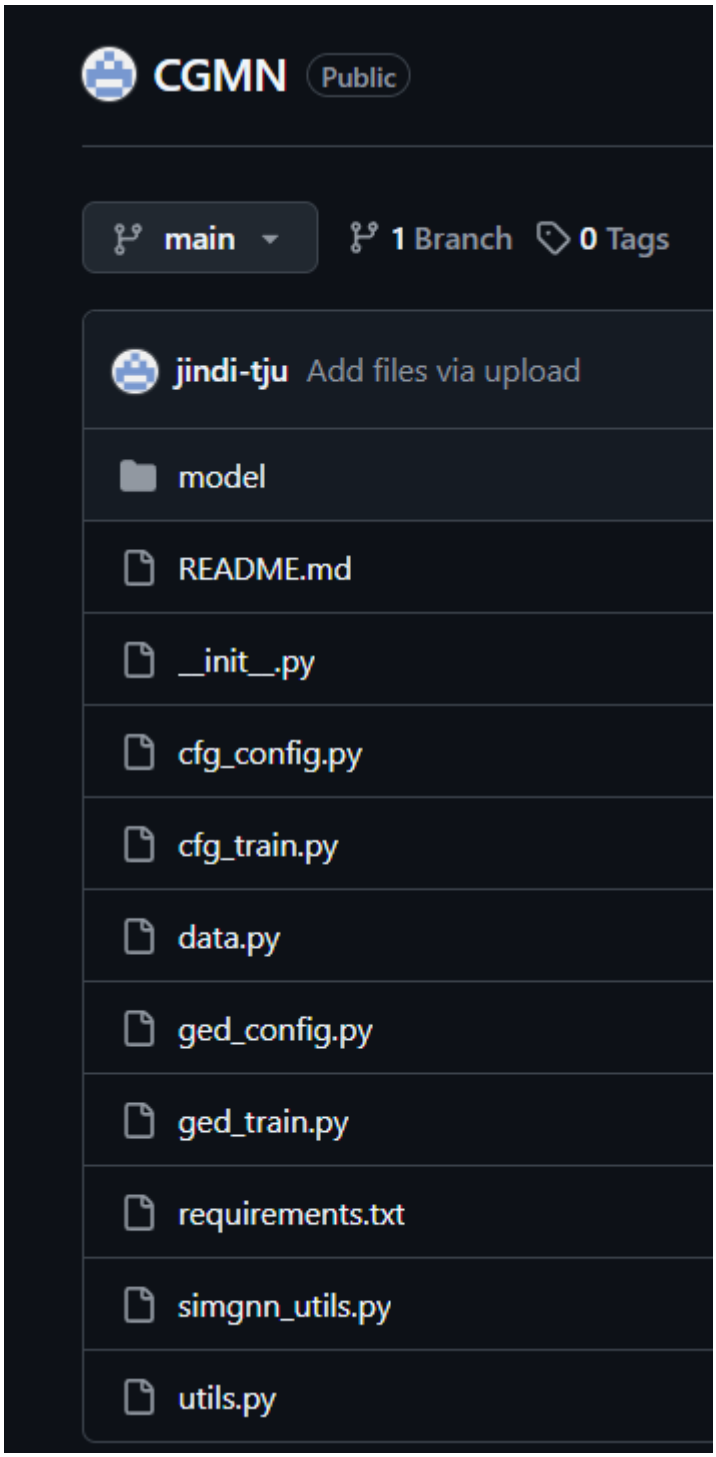
Multilevel graph matching networks for deep graph similarity learning CCF B

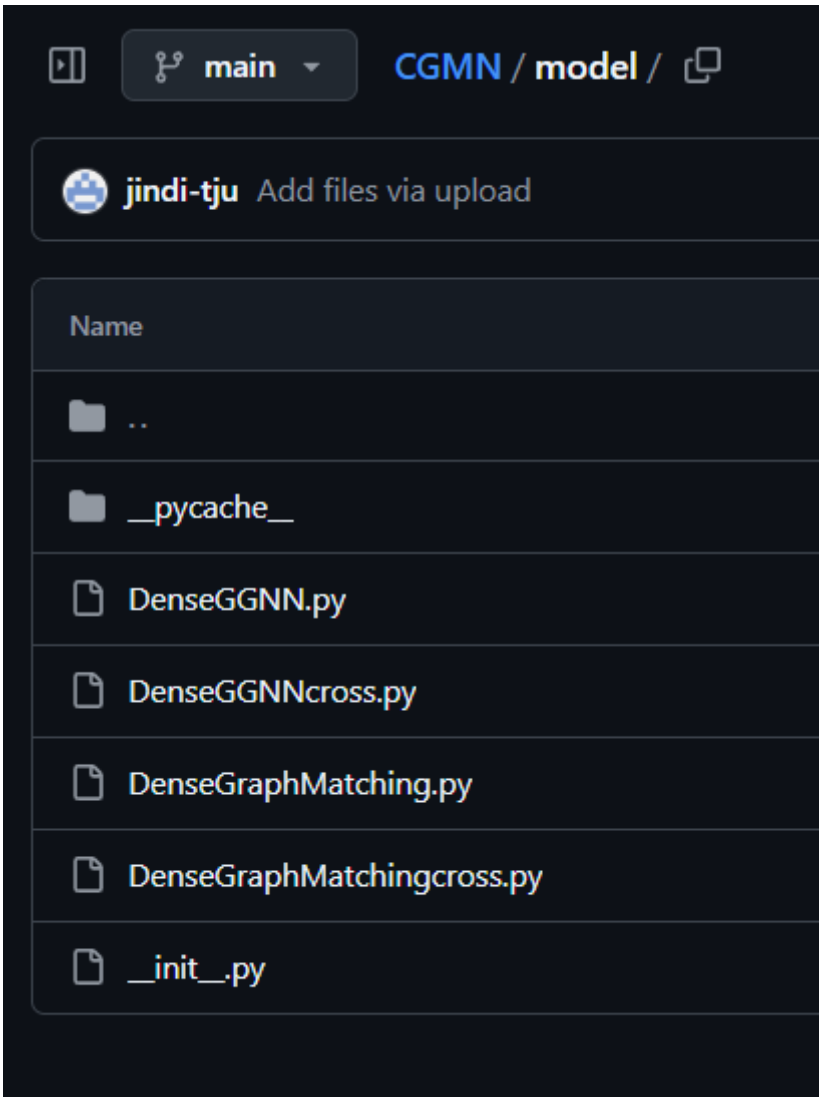
X Ling, L Wu, S Wang, T Ma, F Xu, AX Liu, C Wu, S Ji

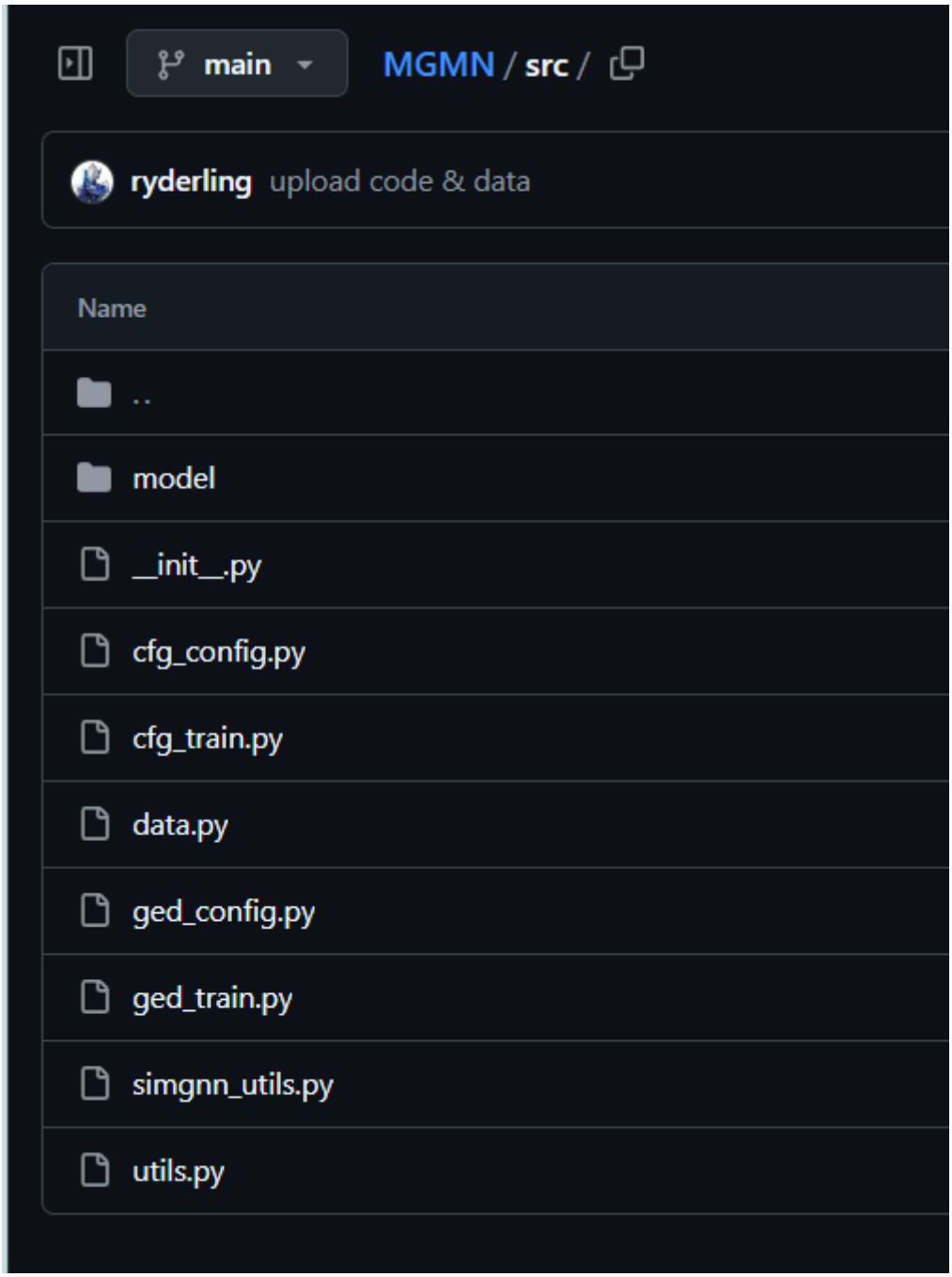
IEEE Transactions on Neural Networks and Learning Systems, 2021 • ieeexplore.ieee.org

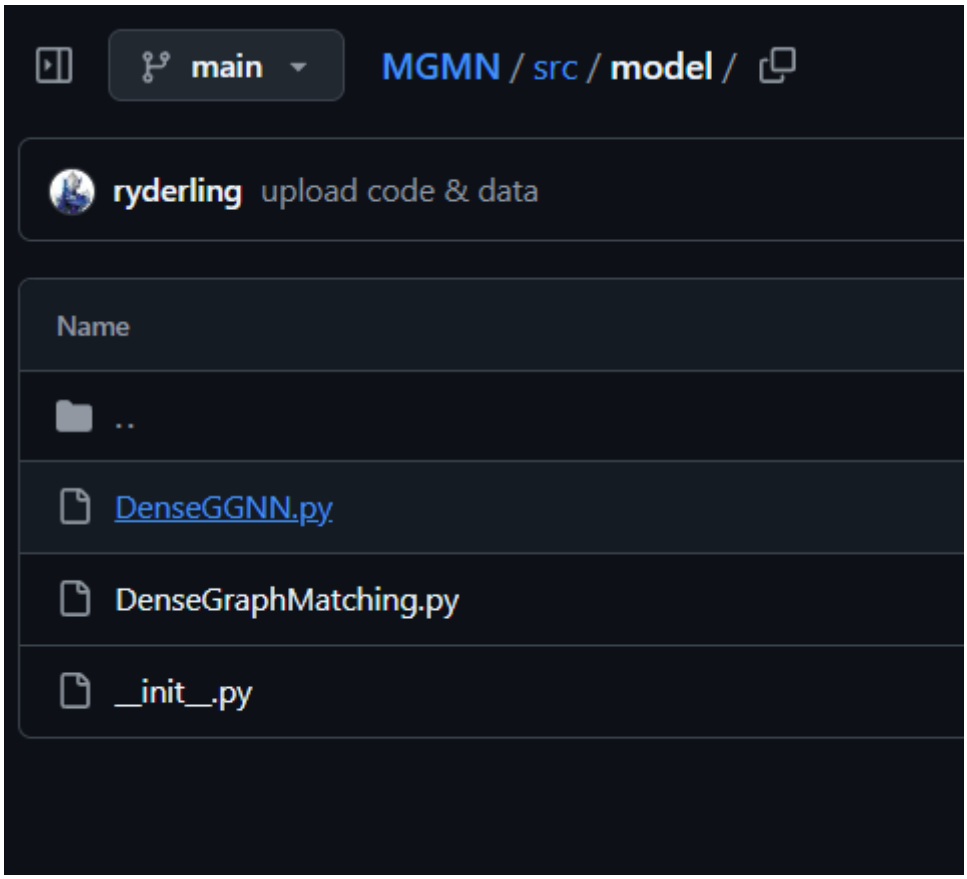


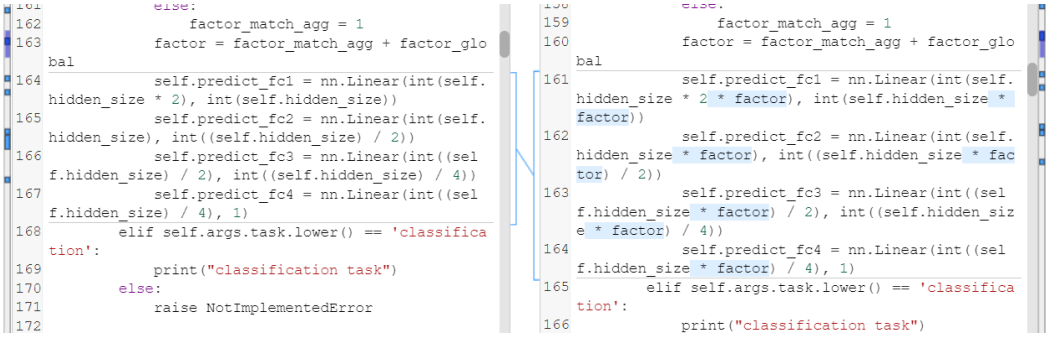
CGMN来自MGMN，CGMN对MGMN的代码，model文件没有改动多少，像模型部分文件名称的改动...，两者使用相同的数据集，并且数据集来自SimGNN这篇论文，而且都使用的SimGNN提供的一些工具函数


CGMN
Public
main
1 Branch
0 Tags
jindi-tju Add files via upload
model
README.md
__init__.py
cfg_config.py
cfg_train.py
data.py
ged_config.py
ged_train.py
requirements.txt
simgnn_utils.py
utils.py


main
CGMN / model /
jindi-tju Add files via upload
Name
..
__pycache__
DenseGGNN.py
DenseGGNNcross.py
DenseGraphMatching.py
DenseGraphMatchingcross.py
__init__.py

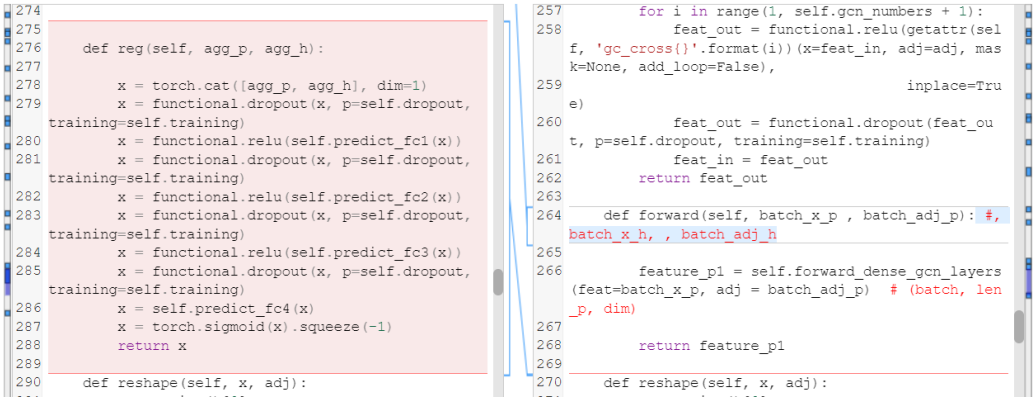

main
MGMN / src /
ryderling upload code & data
Name
..
model
__init__.py
cfg_config.py
cfg_train.py
data.py
ged_config.py
ged_train.py
simgnn_utils.py
utils.py



- 

```
            else:
                factor_match_agg = 1
            factor = factor_match_agg + factor_global
            self.predict_fc1 = nn.Linear(int(self.hidden_size * 2), int(self.hidden_size))
            self.predict_fc2 = nn.Linear(int(self.hidden_size), int((self.hidden_size) / 2))
            self.predict_fc3 = nn.Linear(int((self.hidden_size) / 2), int((self.hidden_size) / 4))
            self.predict_fc4 = nn.Linear(int((self.hidden_size) / 4), 1)
        elif self.args.task.lower() == 'classification':
            print("classification task")
        else:
            raise NotImplementedError
```

```
            else:
                factor_match_agg = 1
            factor = factor_match_agg + factor_global
            self.predict_fc1 = nn.Linear(int(self.hidden_size * 2 * factor), int(self.hidden_size * factor))
            self.predict_fc2 = nn.Linear(int(self.hidden_size * factor), int((self.hidden_size * factor) / 2))
            self.predict_fc3 = nn.Linear(int((self.hidden_size * factor) / 2), int((self.hidden_size * factor) / 4))
            self.predict_fc4 = nn.Linear(int((self.hidden_size * factor) / 4), 1)
        elif self.args.task.lower() == 'classification':
            print("classification task")
```

- 

```
    def reg(self, agg_p, agg_h):

        x = torch.cat([agg_p, agg_h], dim=1)
        x = functional.dropout(x, p=self.dropout, training=self.training)
        x = functional.relu(self.predict_fc1(x))
        x = functional.dropout(x, p=self.dropout, training=self.training)
        x = functional.relu(self.predict_fc2(x))
        x = functional.dropout(x, p=self.dropout, training=self.training)
        x = functional.relu(self.predict_fc3(x))
        x = functional.dropout(x, p=self.dropout, training=self.training)
        x = self.predict_fc4(x)
        x = torch.sigmoid(x).squeeze(-1)
        return x

    def reshape(self, x, adj):
```

```
        for i in range(1, self.gcn_numbers + 1):
            feat_out = functional.relu(getattr(self, 'gc_cross{}'.format(i))(x=feat_in, adj=adj, mask=None, add_loop=False),
                                       inplace=True)
            feat_out = functional.dropout(feat_out, p=self.dropout, training=self.training)
            feat_in = feat_out
        return feat_out

    def forward(self, batch_x_p , batch_adj_p): #, batch_x_h, , batch_adj_h

        feature_p1 = self.forward_dense_gcn_layers(feat=batch_x_p, adj = batch_adj_p)  # (batch, len_p, dim)

        return feature_p1

    def reshape(self, x, adj):
```

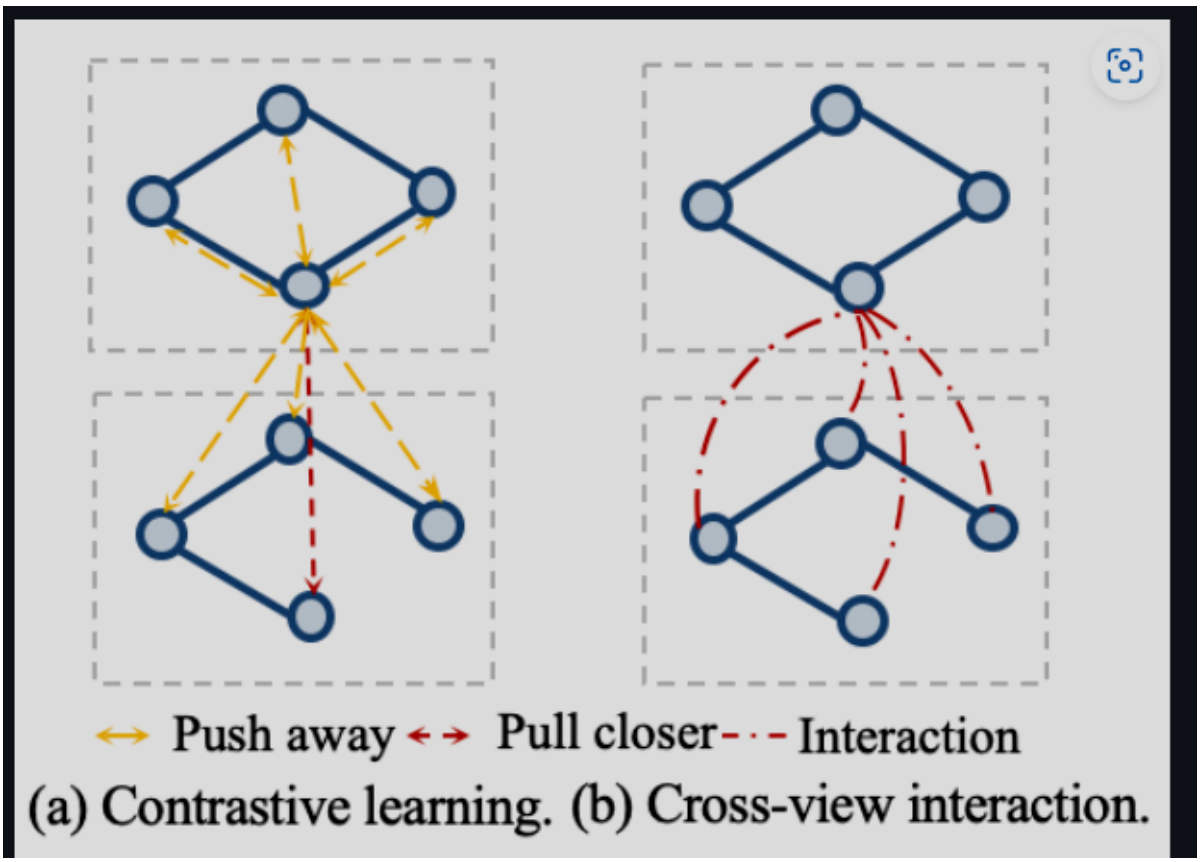
MGMN原理介绍

通过普通编码，得到节点的特征矩阵和邻接矩阵，建立其图的结构，这样才能输入到神经网络去训练，训练的结果其实就是优化编码，最后根据编码计算相似度

对比学习：拉近正样本和原图的表示，拉远负样本和原图的表示，正样本：对原图删除边，屏蔽节点特征，负样本：另一个图

孪生网络：由两个相同子网络组成，这两个网络共享权重，分别处理两个输入，通过比较两个输入的输出结果，可以判断他们的相似性，更好地生成节点的嵌入表示

在图神经网络中，孪生网络通过对比学习来调整网络参数，使得来自相同的DXF文件的增强视图生成的嵌入更加接近，而不同文件的视图嵌入更加不同



匹配的原理：将节点与其它所有节点计算相似性得分，作为权重×到自身节点上，最后再融合进自身节点，融合的方式有很多，虽然论文中展示的是直接串联，但它代码提供了很多种融合方式

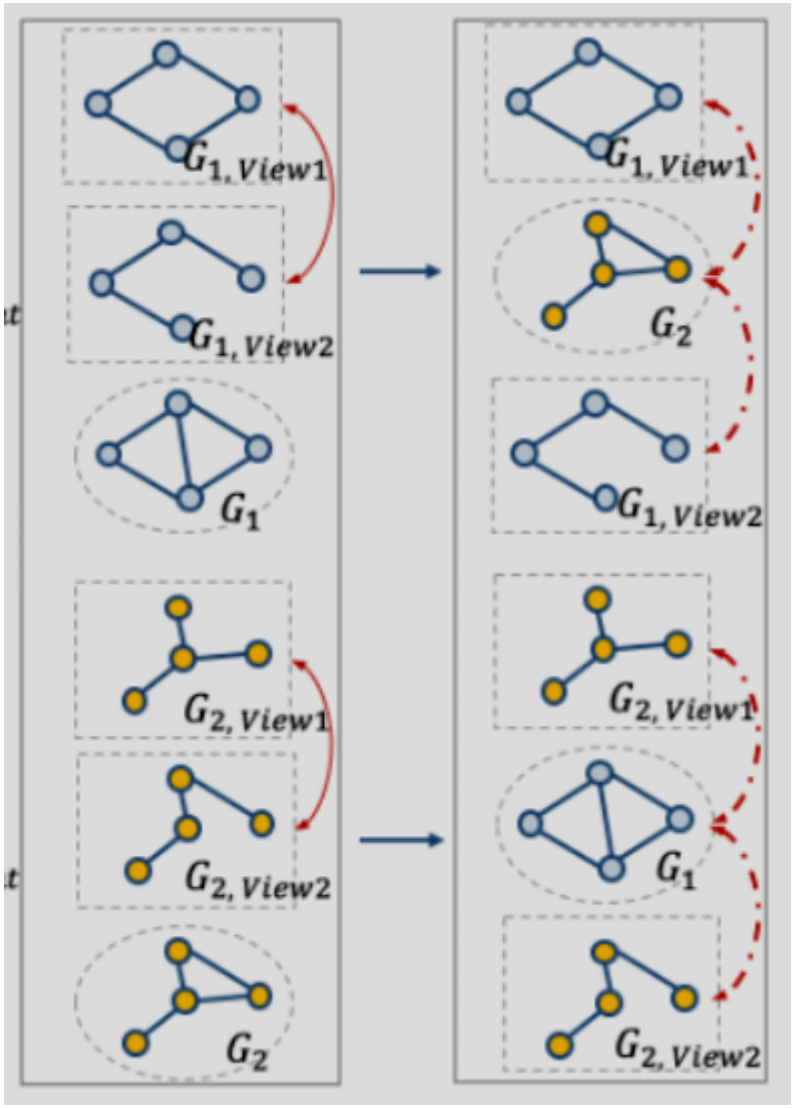
论文创新点：提供了交叉视图匹配和跨图匹配

交叉视图匹配：同一幅图，通过图增强效果（增加或减少边，掩盖节点特征），达到区分同一幅图中节点的差异

跨图匹配：学习不同图对应节点的一致性，区分不同节点的差异

$$\hat{h}_u = h_u \oplus \sum_{v \in G_{1,View2}} \cos(h_u, h_v) h_v, \tag{3}$$

$$h_u^* = h_u \oplus \sum_{i \in G_{2,View1}} \cos(h_u, \hat{h}_i)\hat{h}_i \oplus \sum_{j \in G_{2,View2}} \cos(h_u, \hat{h}_j)\hat{h}_j. \tag{4}$$




```
        elif self.args.match.lower() == 'concat':
            multi_p = self.nn(torch.cat((feature_p, att_mean_h), dim=-1))
            multi_h = self.nn(torch.cat((feature_h, att_mean_p), dim=-1))
        elif self.args.match.lower() == 'euccos':
            batch_size = feature_p.shape[0]
            feature_p_reshape = feature_p.reshape(-1, self.gcn_last_filter)
            feature_h_reshape = feature_h.reshape(-1, self.gcn_last_filter)
            att_mean_p_reshape = att_mean_p.reshape(-1, self.gcn_last_filter)
            att_mean_h_reshape = att_mean_h.reshape(-1, self.gcn_last_filter)
            l2_dist_p = functional.pairwise_distance(feature_p_reshape, att_mean_h_reshape).reshape(
                batch_size, -1, 1)
            cosine_dist_p = functional.cosine_similarity(feature_p_reshape, att_mean_h_reshape).reshape(
                batch_size, -1, 1)
            l2_dist_h = functional.pairwise_distance(feature_h_reshape, att_mean_p_reshape).reshape(
                batch_size, -1, 1)
            cosine_dist_h = functional.cosine_similarity(feature_h_reshape, att_mean_p_reshape).reshape(
                batch_size, -1, 1)
            multi_p = torch.cat((l2_dist_p, cosine_dist_p), dim=-1)
            multi_h = torch.cat((l2_dist_h, cosine_dist_h), dim=-1)
        elif self.args.match.lower() == 'sub':
            multi_p = feature_p - att_mean_h
            multi_h = feature_h - att_mean_p
        elif self.args.match.lower() == 'mul':
            multi_p = feature_p * att_mean_h
            multi_h = feature_h * att_mean_p
        elif self.args.match.lower() == 'submul':
            multi_p = functional.relu(self.nn(torch.cat((feature_p - att_mean_h, feature_p * att_mean_h), dim=-1)))
            multi_h = functional.relu(self.nn(torch.cat((feature_h - att_mean_p, feature_h * att_mean_p), dim=-1)))
```

损失函数：

Cross-graph interaction and cross-view interaction are different. The main difference lies in not only the object of action, but also the purpose. The purpose of cross-view interaction is to maximize node agreement in different views, whereas cross-graph interaction aims to learn the node matching between different graphs. After cross-view interaction and cross-graph interaction learning, we construct the self-supervised loss by maximizing the agreement of positive samples and minimizing the agreement of negative samples:

跨图交互和跨视图交互是不同的。主要区别不仅在于行动对象，还在于目的。跨视图交互的目的是最大化不同视图中的节点一致性，而跨图交互旨在了解不同图之间的节点匹配。经过跨视图交互和跨图交互学习，我们通过最大化正样本的一致性和最小化负样本的一致性来构建自监督损失：

$$loss(h_u^*, h_v^*) = -\log \frac{\mathrm{sim}(h_u^*, h_v^*)}{\mathrm{sim}(h_u^*, h_v^*) + \sum_{k=1}^{N} \mathrm{sim}(h_u^*, h_k^*)}, \tag{5}$$

where u and v are a pair of positive samples and N is the number of negative samples. Both node u and node v need to be calculated, then the total loss can be defined as their average result:

其中 u 和 v 是一对正样本，是 N 负样本的数量。节点 u 和节点 v 都需要计算，那么总损失可以定义为它们的平均结果：

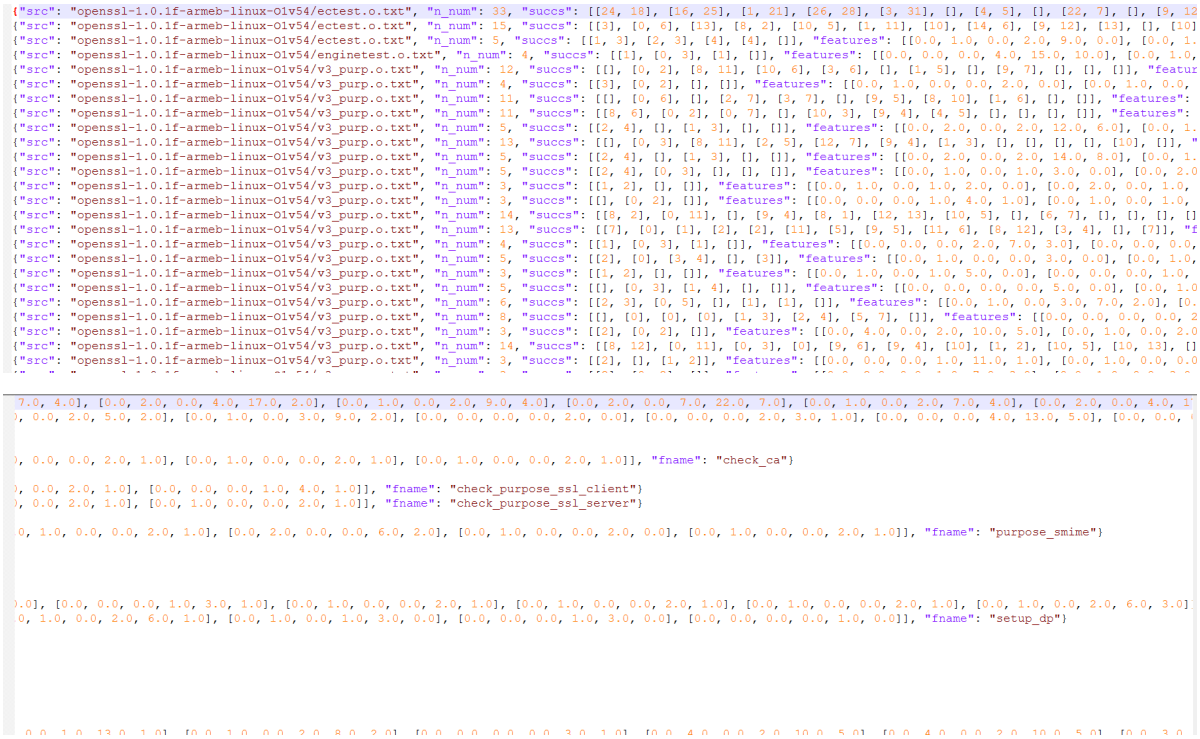
$$\mathcal{L} = \frac{1}{2}\left[loss(h_u^*, h_v^*) + loss(h_v^*, h_u^*)\right]. \tag{6}$$

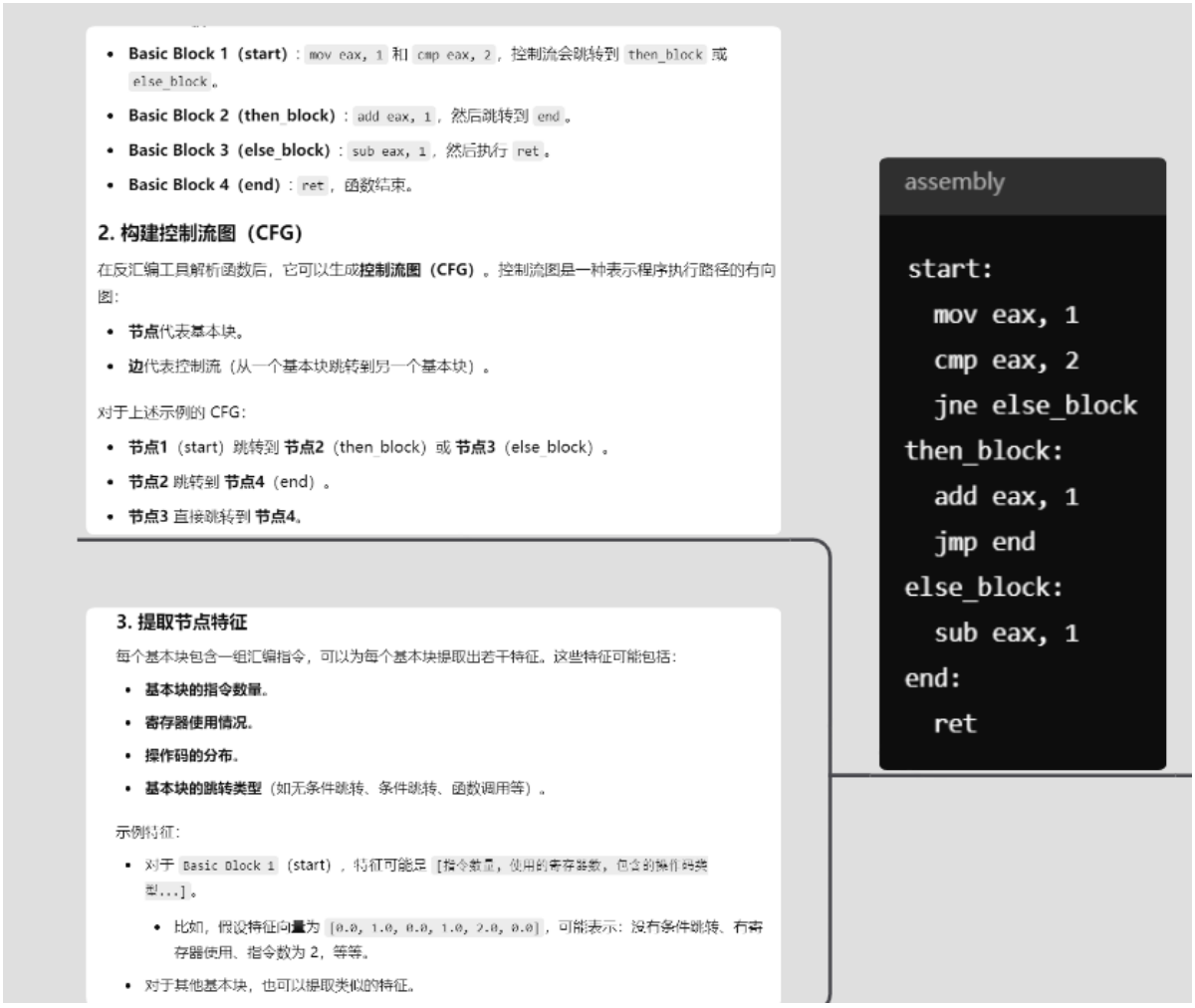
损失函数的构造：分子为正样本之间的相似性得分，分母为（正负样本之间的相似性得分之和）

论文中的下游任务

CFG

分类任务





一个json中有多个图，一行数据表示一个图(一个函数，一个src，一个代码文件,一个fname，fname指函数名称），一个代码文件多个函数，不同的数据可能fname相同，也就是同一个函数经过不同的编译器得到的文件是不同的，造成特征矩阵有差，但是是相似的

回归任务是通过模型得到相似性分数后，通过逻辑回归与类别预测，把相似性分数映射到类别空间，最后通过预测的类别与实际类别去评估好坏，回归任务里正样本从相同的fname中产生，负样本从不同fname中产生，这些正负样本通过孪生网络去学习

GED

回归任务

GED数据集缺少

- An example directory structure is:

```
data
├── GED
│   ├── data
│   │   ├── AIDS700nef/
│   │   ├── LINUX/
│   ├── result
│   │   ├── aids700nef/
│   │   ├── linux/
│   ├── save
│   │   ├── dist_mat/
│   │   ├── aids700nef_ged_astar_gidpair_dist_map.pickle
│   │   ├── linux_ged_astar_gidpair_dist_map.pickle
```

DWG建图

按图层分层

获得该图层所有实体的位置属性，最后上下左右分别取最大，作为该图层的面积范围，若有重叠，则这两个图层连一条边，若一个图层里面包含一个图层，则认为这两个图层之间没有边，这样就把位置信息考虑进编码里

状	名称	开	冻	锁	打	颜色	线型	线宽	透明度	新	说明
	0					白	Continuous	默认	0		
	1					白	HIDDEN	默认	0		
	1D					青	Continuous	默认	0		
	BTM					白	HIDDEN	默认	0		
	Center					黄	CENTER	默认	0		
	Defpoints					白	Continuous	默认	0		
✓	DIMENSI...					绿	Continuous	默认	0		
	HEB					青	Continuous	默认	0		
	PKG					红	PHANTOM2	默认	0		
	TB					白	Continuous	默认	0		

实体数量编码

后面两个都没有，先去掉

```
self.entity_types = [
    'INSERT', 'LINE', 'TEXT', 'MTEXT', 'HATCH', 'LWPOLYLINE',
    'LEADER', 'CIRCLE', 'DIMENSION', 'ARC', 'SPLINE', 'ELLIPSE', 'POLYLINE'
]
```

特征矩阵:

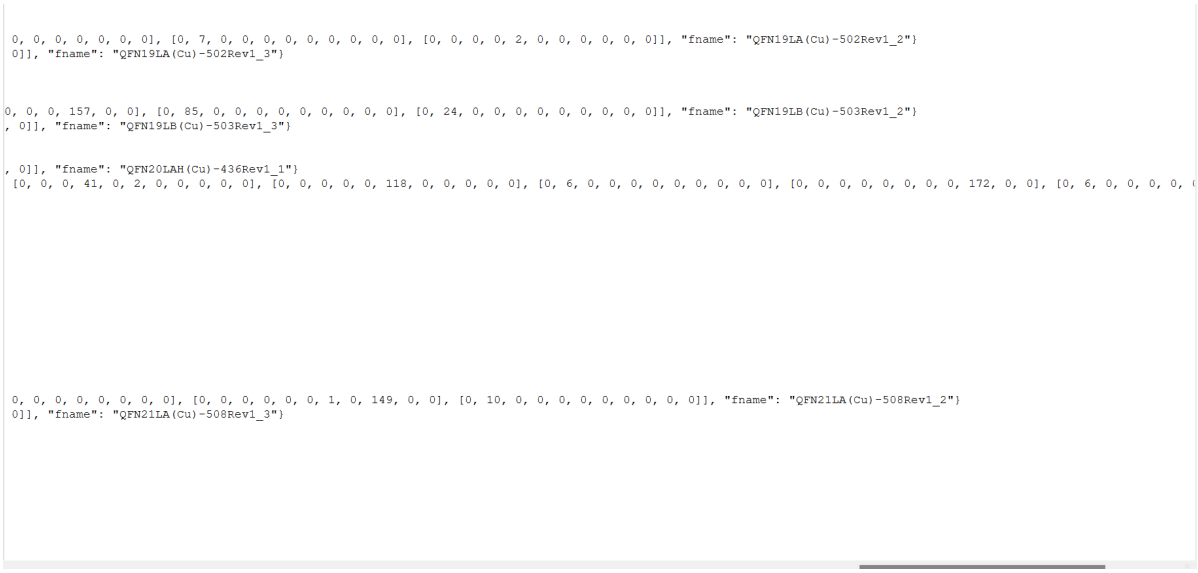
```
[[ 0  0  2  0  0  0  0  0  0  0  0]
 [ 0  8  0  0  0  0  0  0  0  0  0]
 [ 1 44  2  4  0  0  1  0  0  0  0]
 [ 1 24  0  0  0  2  0  8  1  2  0]
 [ 4  1  0  0  0  0  0  0  0  0  0]
 [ 2  1  0  0  0  0  0 10 16  0  0]
 [ 0  0  7  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  4  0  0  0  0  0]]
```

邻接矩阵:

	1D	TB	1	HEB	PKG	BTM	0	Center	DIMENSION
1D	0	1	1	1	1	1	1	1	1
TB	1	0	0	0	0	0	1	0	1
1	1	0	0	1	0	0	0	0	0
HEB	1	0	1	0	1	1	1	1	1
PKG	1	0	0	1	0	1	1	1	1
BTM	1	0	0	1	1	0	1	1	1
0	1	1	0	1	1	1	0	1	1
Center	1	0	0	1	1	1	1	0	1
DIMENSION	1	1	0	1	1	1	1	1	0

- 每个分割后的dxf文件都是一张图


```
{"src": "QFN16Z(Cu)-375Rev3_5.dxf", "n_num": 4, "succs": [[1, 2, 3], [0, 2, 3], [0, 1], [0, 1]], "features": [[0, 172, 0, 1, 4, 24, 4, 15, 95, 4, 1], [0, 8, 5, 1, 0,
{"src": "QFN19LA(Cu)-502Rev1_1.dxf", "n_num": 1, "succs": [[]], "features": [[0, 41, 169, 0, 0, 5, 0, 0, 0, 0, 0]], "fname": "QFN19LA(Cu)-502Rev1_1"}
{"src": "QFN19LA(Cu)-502Rev1_2.dxf", "n_num": 10, "succs": [[1, 2, 3, 4, 5, 6, 7, 9], [0, 2, 3, 4, 5, 6, 7, 9], [0, 1, 3, 4, 5, 6, 7, 8, 9], [0, 1, 2, 4, 5, 6, 7, 8,
{"src": "QFN19LA(Cu)-502Rev1_3.dxf", "n_num": 9, "succs": [[2, 3, 4, 5, 6, 7, 8], [2, 3, 4], [0, 1, 3, 4, 5, 6, 7, 8], [0, 1, 2, 4, 5, 6, 7, 8], [0, 1, 2, 3, 5, 6, 7
{"src": "QFN19LA(Cu)-502Rev1_4.dxf", "n_num": 8, "succs": [[1, 2, 3, 5, 6, 7], [0, 2, 3, 4, 5, 6, 7], [0, 1, 3, 4, 5, 6, 7], [0, 1, 2, 6], [1, 2], [0, 1, 2], [0, 1,
{"src": "QFN19LA(Cu)-502Rev1_5.dxf", "n_num": 7, "succs": [[], [2, 3, 4, 5, 6], [1, 3, 4, 5, 6], [1, 2, 4, 5, 6], [1, 2, 3, 5, 6], [1, 2, 3, 4, 6], [1, 2, 3, 4, 5]],
{"src": "QFN19LB(Cu)-503Rev1_1.dxf", "n_num": 1, "succs": [[]], "features": [[0, 41, 169, 0, 0, 5, 0, 0, 0, 0, 0]], "fname": "QFN19LB(Cu)-503Rev1_1"}
{"src": "QFN19LB(Cu)-503Rev1_2.dxf", "n_num": 10, "succs": [[2, 3, 4, 5, 6, 7, 8, 9], [2, 3, 4], [0, 1, 3, 4, 5, 6, 7, 8, 9], [0, 1, 2, 4, 5, 6, 7, 8, 9], [0, 1, 2,
{"src": "QFN19LB(Cu)-503Rev1_3.dxf", "n_num": 9, "succs": [[2, 3, 4, 5, 6, 7, 8], [2, 3, 4], [0, 1, 3, 4, 5, 6, 7, 8], [0, 1, 2, 4, 5, 6, 7, 8], [0, 1, 2, 3, 5, 6, 7
{"src": "QFN19LB(Cu)-503Rev1_4.dxf", "n_num": 8, "succs": [[1, 2, 3, 5, 6, 7], [0, 2, 3, 4, 5, 6, 7], [0, 1, 3, 4, 5, 6, 7], [0, 1, 2, 6], [1, 2], [0, 1, 2], [0, 1,
{"src": "QFN19LB(Cu)-503Rev1_5.dxf", "n_num": 7, "succs": [[], [2, 3, 4, 5, 6], [1, 3, 4, 5, 6], [1, 2, 4, 5, 6], [1, 2, 3, 5, 6], [1, 2, 3, 4, 6], [1, 2, 3, 4, 5]],
{"src": "QFN20LAH(Cu)-436Rev1_1.dxf", "n_num": 9, "succs": [[2, 3, 4, 5, 6, 7, 8], [2, 3, 8], [0, 1, 3, 4, 5, 6, 7, 8], [0, 1, 2, 4, 5, 6, 7, 8], [0, 2, 3, 5, 6, 7,
{"src": "QFN20LAH(Cu)-436Rev1_2.dxf", "n_num": 11, "succs": [[2, 3, 4, 5, 6, 7, 9, 10], [2, 3], [0, 1, 3, 4, 5, 6, 7, 8, 9, 10], [0, 1, 2, 4, 5, 6, 7, 8, 9, 10], [0,
{"src": "QFN20LAH(Cu)-436Rev1_3.dxf", "n_num": 8, "succs": [[1, 2, 4], [0, 2, 3, 4, 5, 6, 7], [0, 1, 3, 4, 5, 6, 7], [1, 2, 4], [0, 1, 2, 3, 5, 6, 7], [1, 2, 4, 6, 7
{"src": "QFN20LAH(Cu)-436Rev1_4.dxf", "n_num": 6, "succs": [[2, 3, 4, 5], [], [0, 3, 4, 5], [0, 2, 4, 5], [0, 2, 3, 5], [0, 2, 3, 4]], "features": [[0, 8, 0, 0, 0, 0
{"src": "QFN20LAH(Cu)-436Rev1_5.dxf", "n_num": 1, "succs": [[]], "features": [[0, 38, 162, 0, 0, 10, 0, 0, 0, 0, 0]], "fname": "QFN20LAH(Cu)-436Rev1_5"}
{"src": "QFN20LAK(Cu)-442Rev2_1.dxf", "n_num": 3, "succs": [[1, 2], [0, 2], [0, 1]], "features": [[0, 0, 2, 4, 2, 0, 2, 0, 0, 0, 0], [0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0
{"src": "QFN20LAK(Cu)-442Rev2_2.dxf", "n_num": 4, "succs": [[1, 2], [0], [0, 3], [2]], "features": [[0, 0, 3, 2, 2, 0, 0, 0, 0, 0, 0], [0, 8, 0, 0, 0, 0, 0, 0, 0, 0,
{"src": "QFN20LAK(Cu)-442Rev2_3.dxf", "n_num": 1, "succs": [[]], "features": [[0, 38, 162, 0, 0, 10, 0, 0, 0, 0, 0]], "fname": "QFN20LAK(Cu)-442Rev2_3"}
{"src": "QFN20LAK(Cu)-442Rev2_4.dxf", "n_num": 4, "succs": [[1, 2, 3], [0, 2, 3], [0, 1, 3], [0, 1, 2]], "features": [[0, 19, 0, 0, 0, 0, 0, 0, 0, 26, 0], [0, 849, 1
{"src": "QFN20LAK(Cu)-442Rev2_5.dxf", "n_num": 4, "succs": [[1, 2, 3], [0, 2, 3], [0, 1, 3], [0, 1, 2]], "features": [[0, 4, 4, 1, 2, 0, 0, 0, 0, 0, 0], [61, 558, 20
{"src": "QFN20LAK(Cu)-442Rev2_6.dxf", "n_num": 6, "succs": [[2, 3, 4, 5], [], [0, 3, 4, 5], [0, 2, 4, 5], [0, 2, 3, 5], [0, 2, 3, 4]], "features": [[0, 8, 0, 0, 0, 0
{"src": "QFN20TLAN(NiPdAu)-455Rev1_1.dxf", "n_num": 8, "succs": [[1, 2, 3, 4, 5, 7], [0, 2, 3, 4, 5, 6, 7], [0, 1, 3, 4, 5, 6, 7], [0, 1, 2, 4, 5, 7], [0, 1, 2, 3, 5
{"src": "QFN20TLAN(NiPdAu)-455Rev1_2.dxf", "n_num": 7, "succs": [[1, 2, 3, 4, 5, 6], [0, 2, 3, 4, 5, 6], [0, 1, 3, 4, 5, 6], [0, 1, 2, 4, 5, 6], [0, 1, 2, 3, 5, 6],
{"src": "QFN20TLAN(NiPdAu)-455Rev1_3.dxf", "n_num": 3, "succs": [[1, 2], [0], [0]], "features": [[0, 44, 166, 0, 0, 16, 0, 0, 0, 0, 0], [0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0
{"src": "QFN20TLAN(NiPdAu)-455Rev1_4.dxf", "n_num": 6, "succs": [[1, 2, 3, 4, 5], [0, 2, 3, 4, 5], [0, 1, 3, 4, 5], [0, 1, 2, 4, 5], [0, 1, 2, 3, 5], [0, 1, 2, 3, 4]
{"src": "QFN21LA(Cu)-508Rev1_1.dxf", "n_num": 1, "succs": [[]], "features": [[0, 41, 169, 0, 0, 5, 0, 0, 0, 0, 0]], "fname": "QFN21LA(Cu)-508Rev1_1"}
{"src": "QFN21LA(Cu)-508Rev1_2.dxf", "n_num": 10, "succs": [[2, 3, 4, 5, 6, 7, 8, 9], [2, 3, 4, 8], [0, 1, 3, 4, 5, 6, 7, 8, 9], [0, 1, 2, 4, 5, 6, 7, 8, 9], [0, 1,
{"src": "QFN21LA(Cu)-508Rev1_3.dxf", "n_num": 9, "succs": [[2, 3, 4, 5, 6, 7, 8], [2, 3, 4], [0, 1, 3, 4, 5, 6, 7, 8], [0, 1, 2, 4, 5, 6, 7, 8], [0, 1, 2, 3, 5, 6, 7
{"src": "QFN21LA(Cu)-508Rev1_4.dxf", "n_num": 8, "succs": [[1, 2, 3, 5, 6, 7], [0, 2, 3, 4, 5, 6, 7], [0, 1, 3, 4, 5, 6, 7], [0, 1, 2, 6], [1, 2], [0, 1, 2], [0, 1,
{"src": "QFN21LA(Cu)-508Rev1_5.dxf", "n_num": 7, "succs": [[], [2, 3, 4, 5, 6], [1, 3, 4, 5, 6], [1, 2, 4, 5, 6], [1, 2, 3, 5, 6], [1, 2, 3, 4, 6], [1, 2, 3, 4, 5]],
{"src": "QFN22LD(Cu)-532Rev1_1.dxf", "n_num": 7, "succs": [[1, 2, 3, 4, 5, 6], [0, 2, 3, 4, 5, 6], [0, 1, 3, 4, 5, 6], [0, 1, 2, 4, 5, 6], [0, 1, 2, 3, 5, 6], [0, 1,
{"src": "QFN22LD(Cu)-532Rev1_2.dxf", "n_num": 9, "succs": [[1, 2, 3, 4, 5, 6, 7], [0, 2, 3, 4, 5, 6, 7, 8], [0, 1, 3, 4, 5, 6], [0, 1, 2, 4, 5, 6], [0, 1, 2, 3, 5, 6
{"src": "QFN22LD(Cu)-532Rev1_3.dxf", "n_num": 7, "succs": [[1, 2, 4], [0, 2, 3, 4, 5, 6], [0, 1, 3, 4, 5, 6], [1, 2, 4, 5, 6], [0, 1, 2, 3, 5, 6], [1, 2, 3, 4, 6], [
{"src": "QFN22LD(Cu)-532Rev1_4.dxf", "n_num": 6, "succs": [[2, 3, 4, 5], [], [0, 3, 4, 5], [0, 2, 4, 5], [0, 2, 3, 5], [0, 2, 3, 4]], "features": [[0, 8, 0, 0, 0, 0,
{"src": "QFN22LD(Cu)-532Rev1_5.dxf", "n_num": 1, "succs": [[]], "features": [[0, 38, 164, 0, 0, 10, 0, 0, 0, 0, 0]], "fname": "QFN22LD(Cu)-532Rev1_5"}
```

```
 0, 0, 0, 0, 0, 0, 0], [0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0]], "fname": "QFN19LA(Cu)-502Rev1_2"}
 0]], "fname": "QFN19LA(Cu)-502Rev1_3"}


0, 0, 0, 157, 0, 0], [0, 85, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 24, 0, 0, 0, 0, 0, 0, 0, 0, 0]], "fname": "QFN19LB(Cu)-503Rev1_2"}
, 0]], "fname": "QFN19LB(Cu)-503Rev1_3"}


, 0]], "fname": "QFN20LAH(Cu)-436Rev1_1"}
 [0, 0, 0, 41, 0, 2, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 118, 0, 0, 0, 0, 0], [0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 172, 0, 0], [0, 6, 0, 0, 0, 0, 0,
```



```
 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 1, 0, 149, 0, 0], [0, 10, 0, 0, 0, 0, 0, 0, 0, 0, 0]], "fname": "QFN21LA(Cu)-508Rev1_2"}
 0]], "fname": "QFN21LA(Cu)-508Rev1_3"}
```

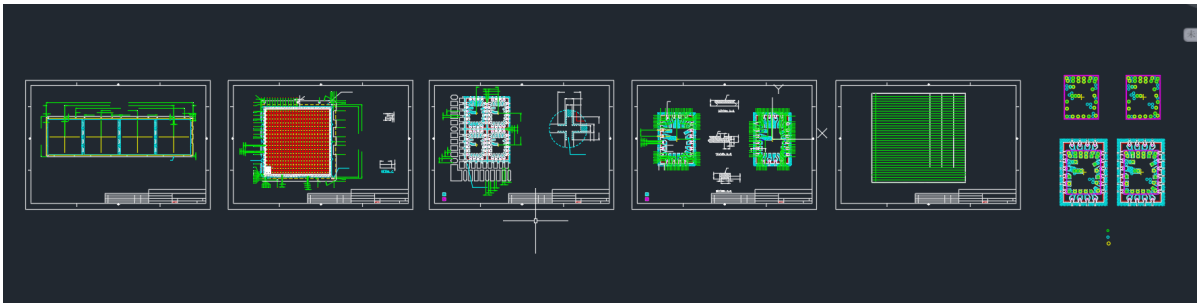

需要讨论的地方

线形，颜色，怎么考虑进去

颜色类型可直接这样按数量？算距离是有问题的

相同一个dxf中,fname标志为相同

看看效果，这样就分成6类，正样本从每一类中选，不会像之前一样，正样本就是自身和自身，但是第六类有的规格不是这样，第4类有的类型也不太一样



接下来要做的事情

GED任务看看能不能跑通，因为数据集缺失，如果能跑通看看效果

删除一些实体，测测相似度

整合一些其它论文的方法，看看能不能融入这个模型中

对于一个图，有若干图层，我们对其实体进行聚类，再加权到实体编码

会议结果

图层问题

设计者不规范，同样的实体，设计者可能放在A图层，也可能放在B图层，如果往图层方向考虑，我们的代码结果是不同的，图层不能利用了，我们需要找其它方向去建立边，从具体的位置去考虑