Q1 Supervised Outlier Detection

Code https://github.com/Kainan-Liu/DSAA5002-FinalProject

Problem Statement

In real life, Anomaly Detection is usually in the context of unsupervised learning or semi-supervised learning because the outlier data is extremely difficult to obtain. While in this case, we are asked to do the outlier detection in a supervised way, this may leads to a problem that our machine learning model is extremely easy to **overfit** the anomalies because the number of outliers and inlier is extremely uneven. Therefore, if we simply apply a machine learning model like MLP or other frameworks, it will not generalize well on the test data.

Label	Ratio
0	0.962209
1	0.037791

Tabel 1 Lable Distribution on one of the train set

Experiment

Typically, we can use some resampling methods to change the distribution of the labels, it can change the weight of the outliers in the dataset to some extends and make our machine learning model aware of them. Therefore, I **upsample the anomalies** in the train data and it increases the Recall and Accuracy from (0.01, 0.02) to (0.335, 0.343) compared to the model without any resampling techniques.
Here is my MLP architecture which consists of three linear layers and tanh activation function.

```python
class ANNet(nn.Module):
    def __init__(self, in_features, out_features, *args, **kwargs) -> None:
        super(ANNet, self).__init__(*args, **kwargs)
        self.in_features = in_features
        self.out_features = out_features

        hidden_features = [16, 32, 64]
        self.model = nn.Sequential(
            LinearBlock(in_features=in_features, out_features=hidden_features[0]),
            LinearBlock(in_features=hidden_features[0], out_features=hidden_features[1]),
            LinearBlock(in_features=hidden_features[1], out_features=hidden_features[2]),
            LinearBlock(in_features=hidden_features[2], out_features=out_features, act=False, last=True)
        )

        self.classifier = nn.Sequential(
            nn.Linear(in_features=self.out_features, out_features=1),
            nn.Sigmoid()
        )

    def forward(self, x):
        return self.model(x)
```

Figure 1 Model Architecture sreenshot

Results After resampling:

Other Attempts

Inspired by Deep Weakly-supervised Anomaly Detection which is accpeted to KDD 2023, I try to make a similar framework while in a supervised context. In this paper, they use {anomaly-anomaly}, {anomaly-normal}, {normal-normal} pairs to learn how to detect the anomalies in a contrastive learning approach. And use this approach they can overcome the label imbalance problems

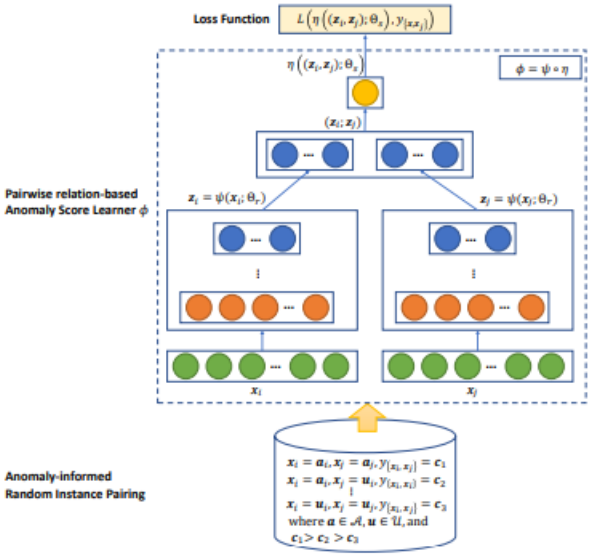
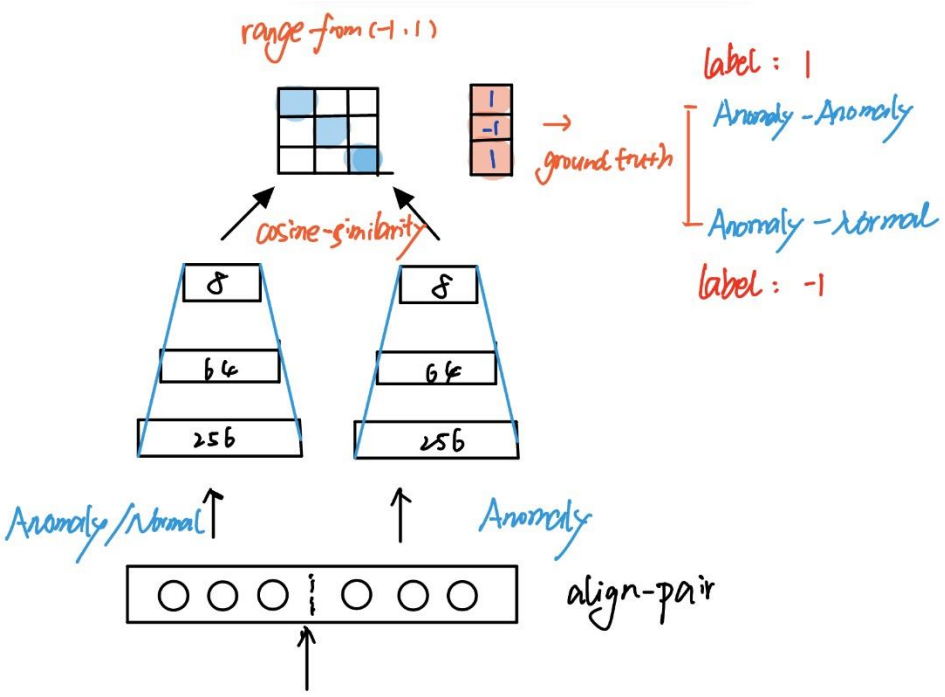


Figure 2 PReNet Framework

The pairs of data is processed by a two stream network which is two shared-weight mlp in their work. And they try to predict a anomaly score label {0,4,8} for three different pairs respectively.

And I think in our case, we can use a similar framework but in a supervised context, which also consists of two stream networks, and we can pretrain on the whole dataset(resample) first and then pass the pair of data to the network. And then we can calculate the similarity between the learned embedding, if the cosine similarity is close to 1, it can be a evidence that these two data is similar to each other. 0 or -1 is dissimilar. When **inference** on the test dataset, we **only send the test data to the left network, and only send the anomaly data on the train dataset to the right**

network, if the cosine similarity(anomaly score) is close to 1, then it is an outlier on the testdataset.



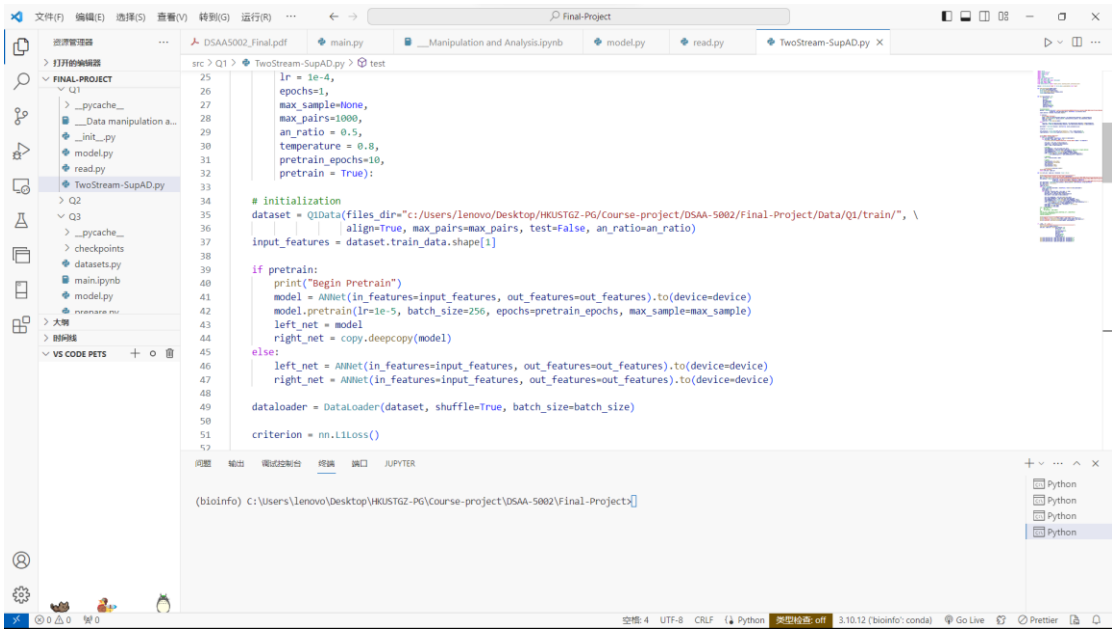
Experiment Results

Method/Metric	Recall	Precision
MLP + Upsampling	0.344	0.349
Two-Stream SupAD	On-going	On-going

Upsampling result:

```
(bioinfo) C:\Users\lenovo\Desktop\HKUSTGZ-PG\Course-project\DSAA-5002\Final-Project>D:/Program/Anaconda3/envs/bioinfo/python.exe c:/Users/lenovo/Desktop/HKUSTGZ-PG/Course-project/DSAA-5002/Final-Project/src/Q1/TwoStream-SupAD.py
==================================================
Begin Pretrain
ratio 2.333366444819708
1    93960
0    40268
Name: Is_Falling, dtype: int64
Test Pretrain model
==Save Results==
Success!
====================Score========================
Recall: 0.34402332361516036
Precision: 0.34911242603550297
Accuracy: 0.9328099048769439
Begin Anomaly Detection
End Training
==================================================
```

Two-Stream SupAD code:


lr = 1e-4,
epochs=1,
max_sample=None,
max_pairs=1000,
an_ratio = 0.5,
temperature = 0.8,
pretrain_epochs=10,
pretrain = True):
# initialization
dataset = Q1Data(files_dir="c:/Users/lenovo/Desktop/HKUSTGZ-PG/Course-project/DSAA-5002/Final-Project/Data/Q1/train/", \
align=True, max_pairs=max_pairs, test=False, an_ratio=an_ratio)
input_features = dataset.train_data.shape[1]
if pretrain:
print("Begin Pretrain")
model = ANNet(in_features=input_features, out_features=out_features).to(device=device)
model.pretrain(lr=1e-5, batch_size=256, epochs=pretrain_epochs, max_sample=max_sample)
left_net = model
right_net = copy.deepcopy(model)
else:
left_net = ANNet(in_features=input_features, out_features=out_features).to(device=device)
right_net = ANNet(in_features=input_features, out_features=out_features).to(device=device)
dataloader = DataLoader(dataset, shuffle=True, batch_size=batch_size)
criterion = nn.L1Loss()
(bioinfo) C:\Users\lenovo\Desktop\HKUSTGZ-PG\Course-project\DSAA-5002\Final-Project>