Progess of the Project

Tsung-Min Pai

2023/11/17

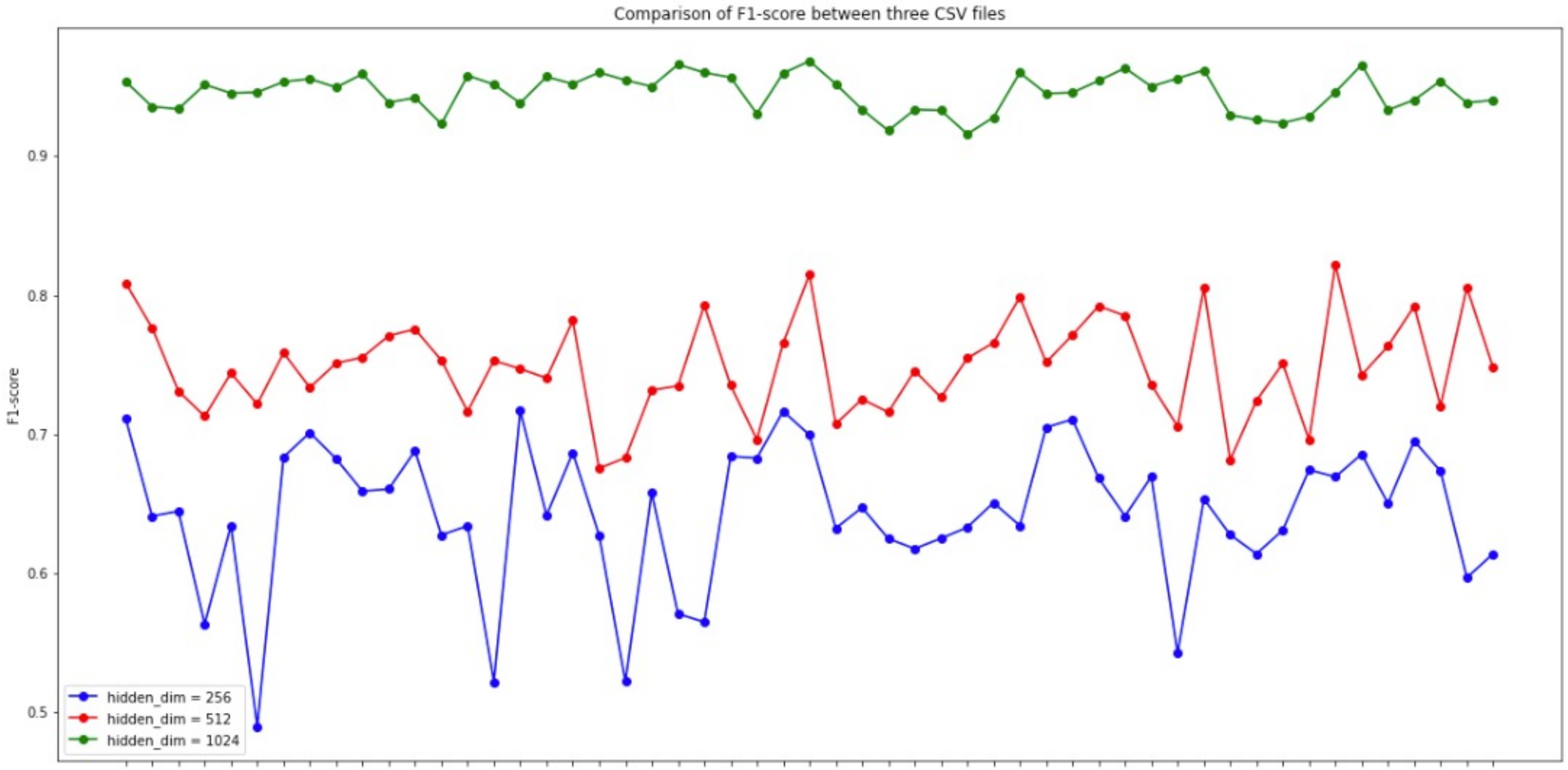
Outline

- **GNN**
  - **Experiment**
    - **Recap**
    - **Observe the distribution of the prediction**
    - **Remove the predictable TTP from the dataset**
  - **Paper**
    - **GraphSMOTE**
- **Future Work**

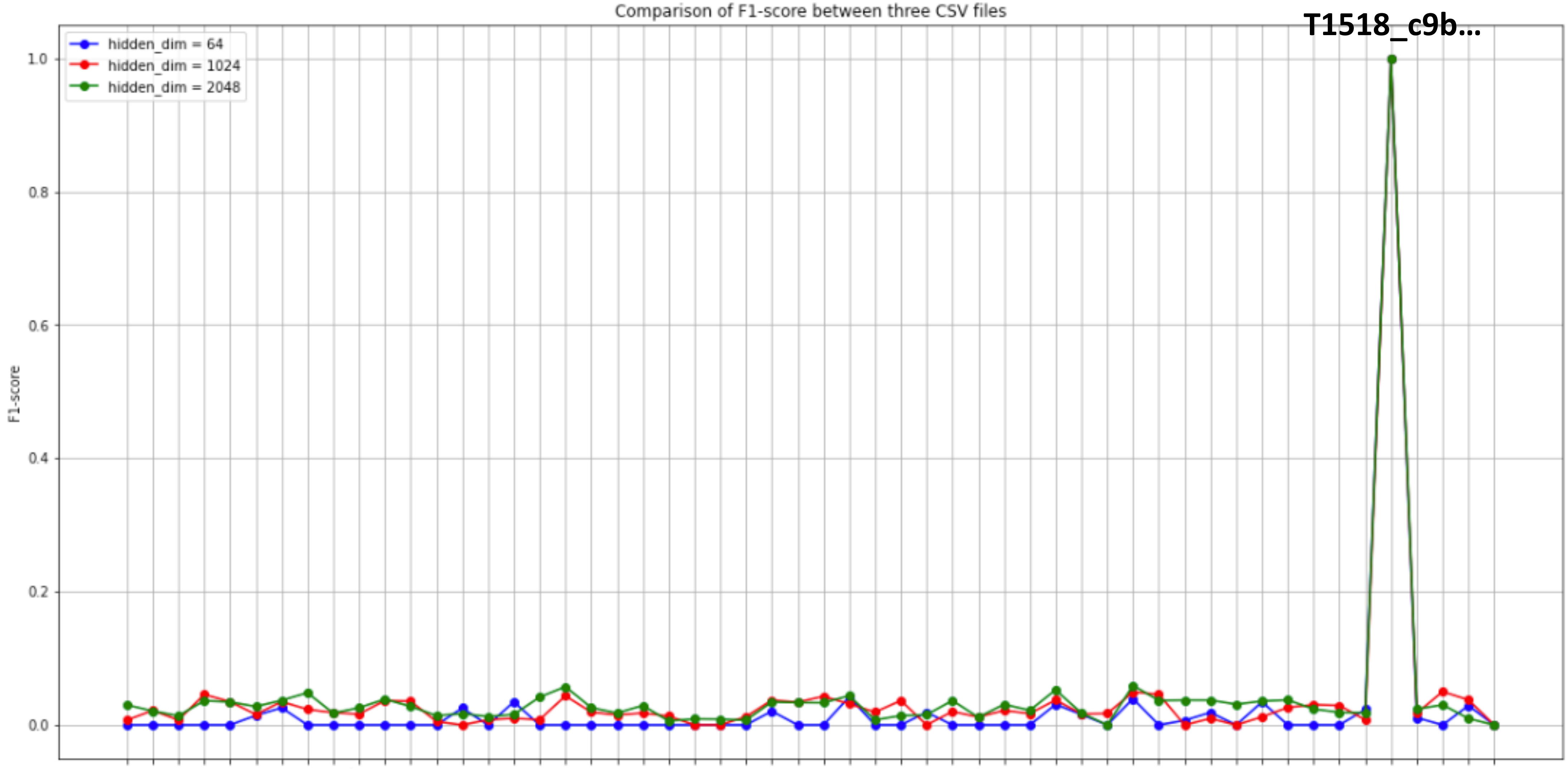
Experiment - Recap

Oversampling

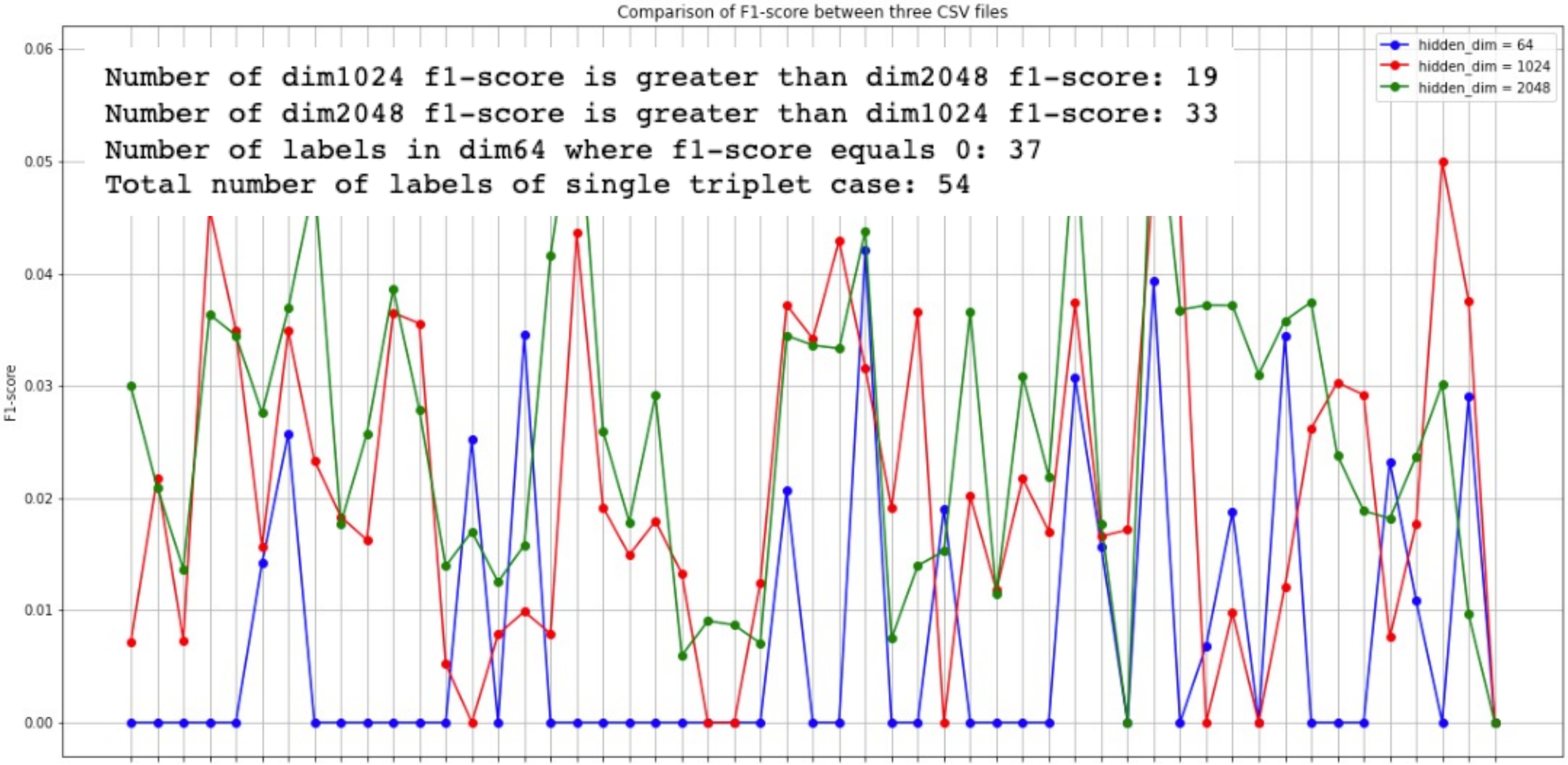
- Previous Trial: hope to see the result on **training** dataset
  - Use data with **320** times single triplet → # of training data = 13657600
  - Larger hidden **dimension** → more neurons to remember the data
  - Let the model **overfit** first → Succeed



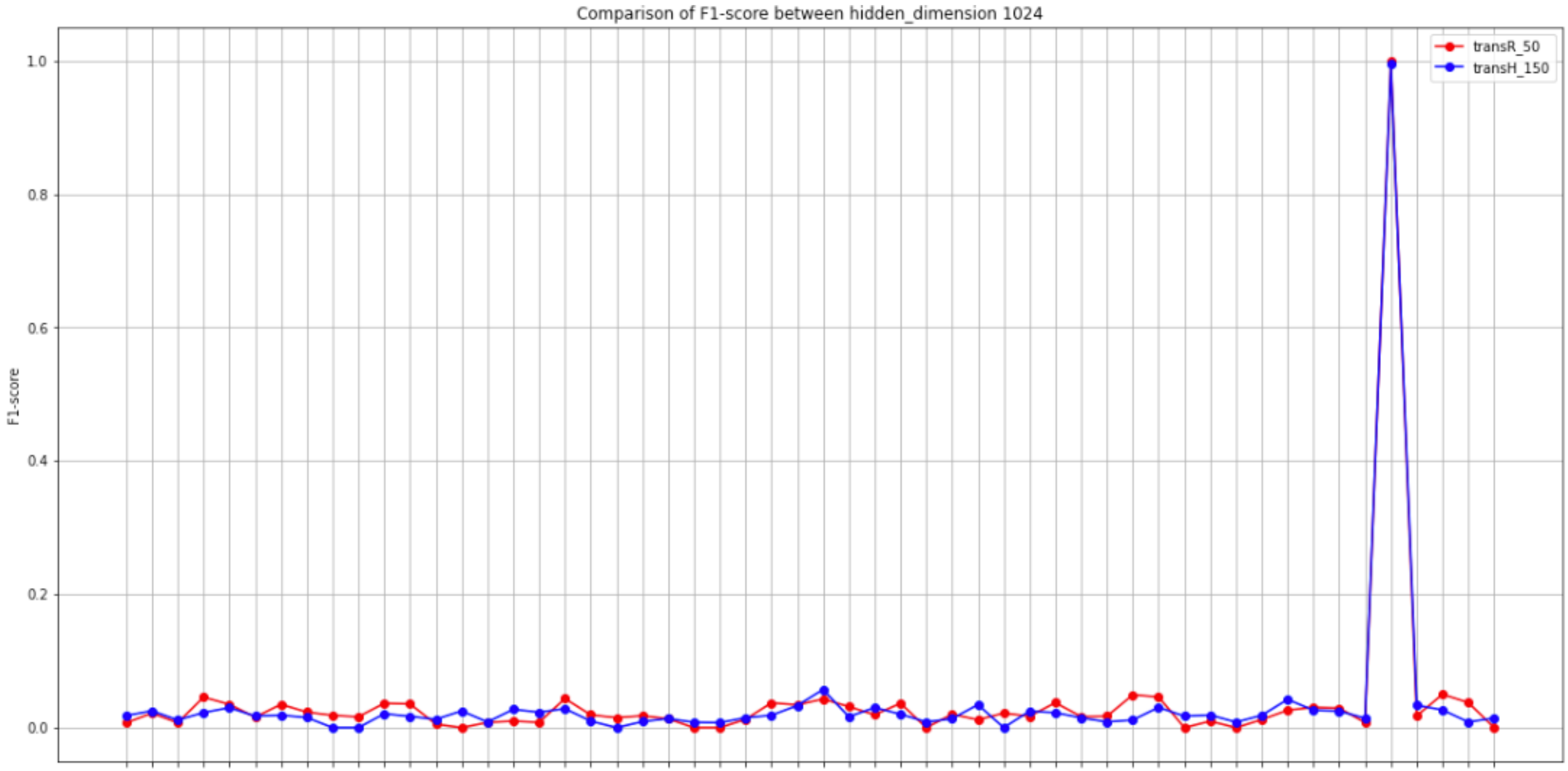
Observation on Different Dimension



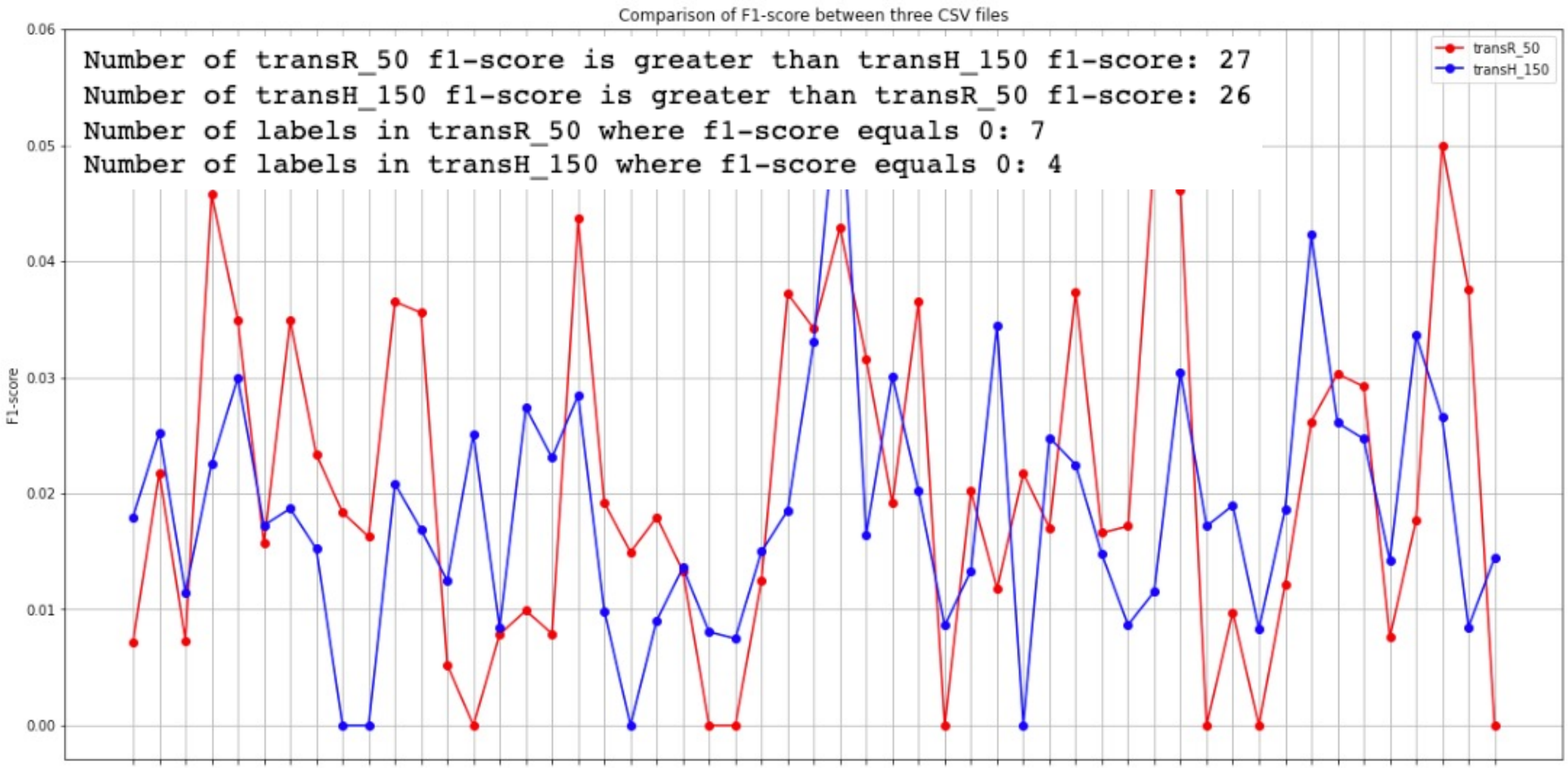
Observation on Different Dimension



Observation on Different Embedding



Observation on Different Embedding



Conclusion

- Noticed that **T1518_c9b** always got predicted in all the experiments
- **Hidden Dimension** do have an effect on the result → but it’s all about from 0 to 0.05
- **Embedding** (transR_50 and tansH_150) seems to have the similar result → try more

Experiment - Observe

Thoughts

- Try the ensemble
  - Different GNN
    - Different **hidden dimension**
    - Different **embedding**
    - Different **model**

  - Different type of model → **MLP, RNN, GNN…**
    - Maybe the different can identify the different single triplet class

Ensemble - Voting

- Noticed that **T1518_c9b** always got predicted in all the experiments
  - Euni's MLP and RNN also predict **T1518_c9b** perfectly → **Reason ?!**
- Experiment result on the single triplet case:

accuracy			0.02	5300
macro avg	0.02	0.02	0.02	5300
weighted avg	0.03	0.02	0.02	5300

- Voting these three models (MLP, RNN, GNN) may not be useful

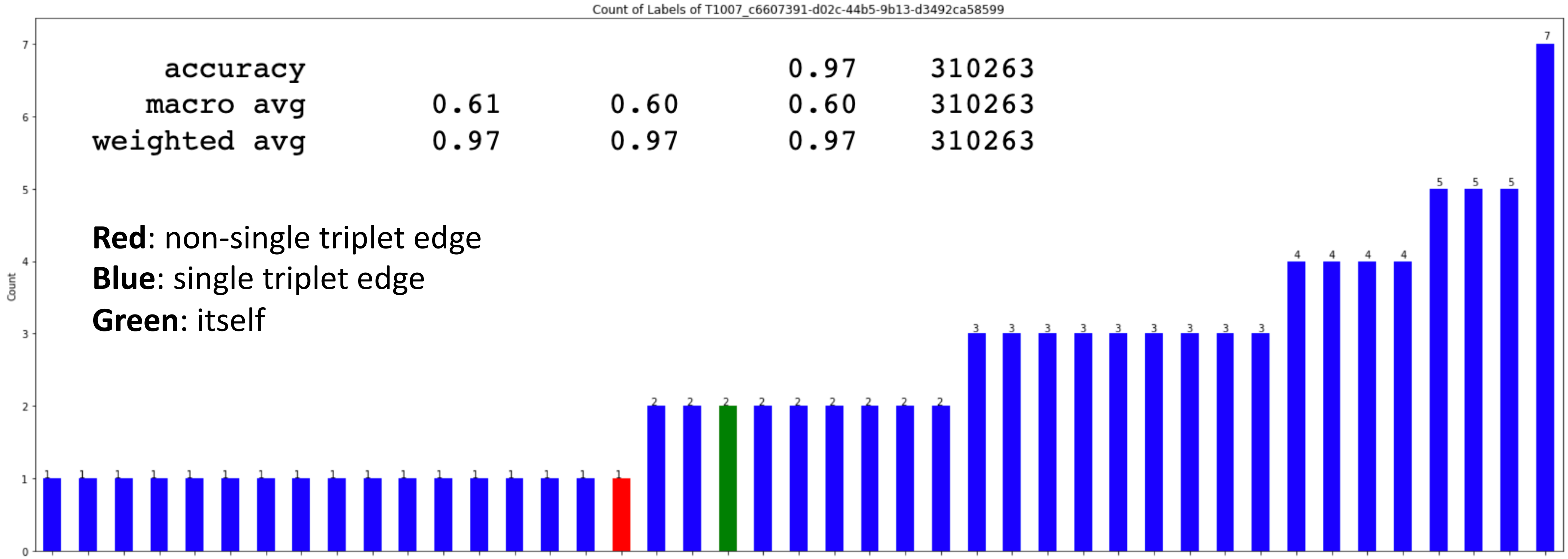
true_label	predicted_label
T1003.003_9f73269695e54311dd61dc68940fb3e1	T1490_9e5e4c0655fd1b5be88bd40b8251175f
T1003.003_9f73269695e54311dd61dc68940fb3e1	T1490_9e5e4c0655fd1b5be88bd40b8251175f
T1003.003_9f73269695e54311dd61dc68940fb3e1	T1490_9e5e4c0655fd1b5be88bd40b8251175f
T1003.003_9f73269695e54311dd61dc68940fb3e1	T1564.003_9a2edad4053a2b59fb9167a9bc29e7dc
T1003.003_9f73269695e54311dd61dc68940fb3e1	T1499_2fe2d5e6-7b06-4fc0-bf71-6966a1226731
T1003.003_9f73269695e54311dd61dc68940fb3e1	T1490_9e5e4c0655fd1b5be88bd40b8251175f

Since the order of the prediction would be the same

Observation of the Prediction

- Use the original training set

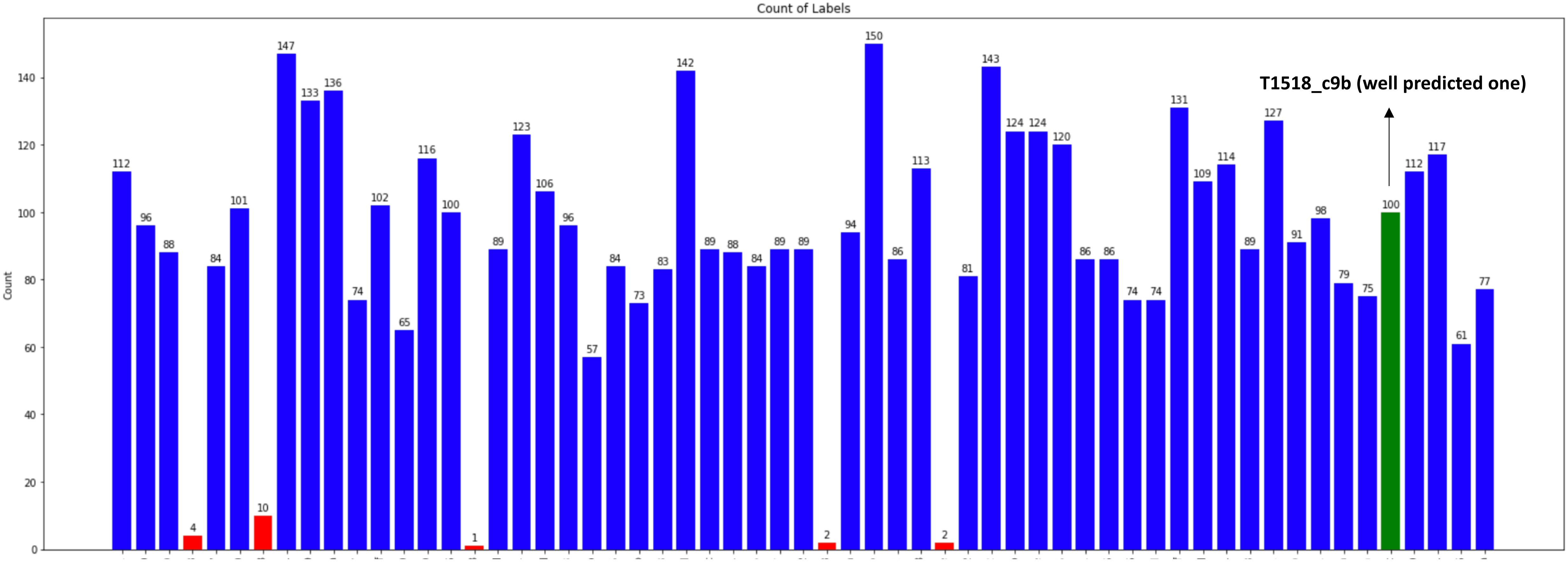
- The distribution of the prediction is so sparse
    - Most of the predictions are like this:



Observation of the Prediction

- The # of the predicted labels

```
Top 5 Labels and Counts:
T1082_29451844-9b76-4e16-a9ee-d6feab4b24db: 150
T1016_921055f4-5970-4707-909e-62f594234d91: 147
T1124_fa6e8607-e0b1-425d-8924-9b894da5a002: 143
T1053.005_ee454be9197890de62705ce6255933fd: 142
T1016_e8017c46-acb8-400c-a4b5-b3362b5b5baa: 136
```

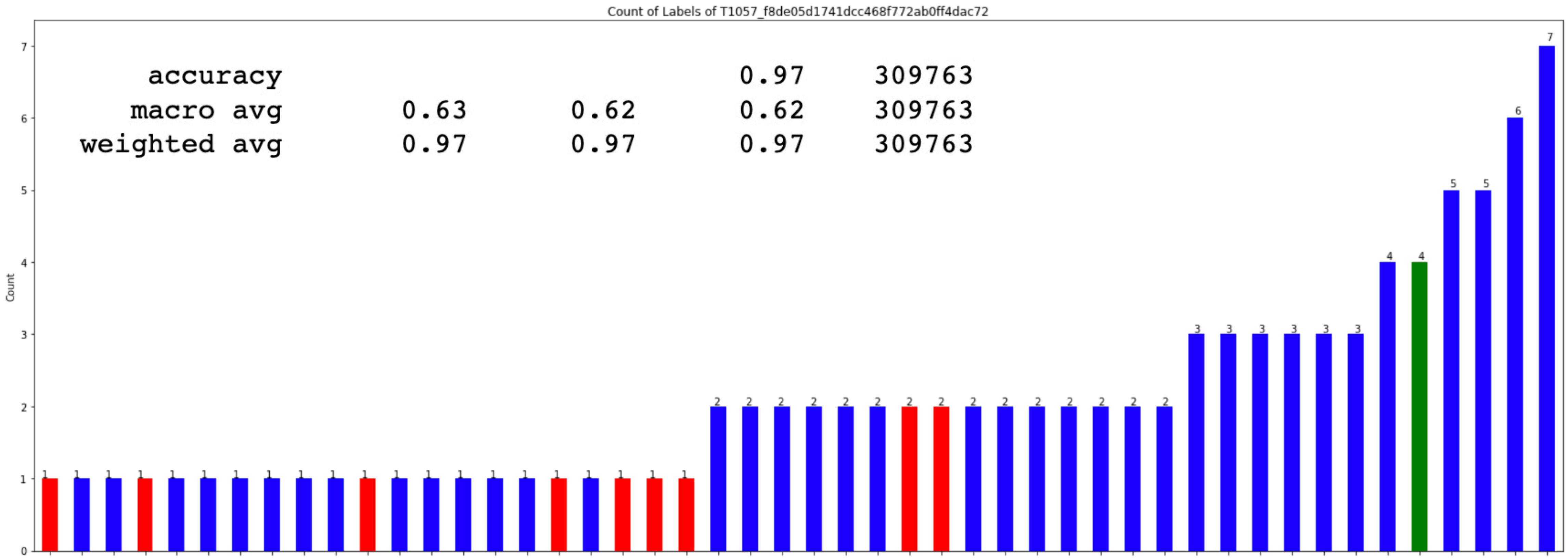


Experiment - Remove

Remove the Popular TTPs

```
Top 5 Labels and Counts:
T1082_29451844-9b76-4e16-a9ee-d6feab4b24db: 150
T1016_921055f4-5970-4707-909e-62f594234d91: 147
T1124_fa6e8607-e0b1-425d-8924-9b894da5a002: 143
T1053.005_ee454be9197890de62705ce6255933fd: 142
T1016_e8017c46-acb8-400c-a4b5-b3362b5b5baa: 136
```

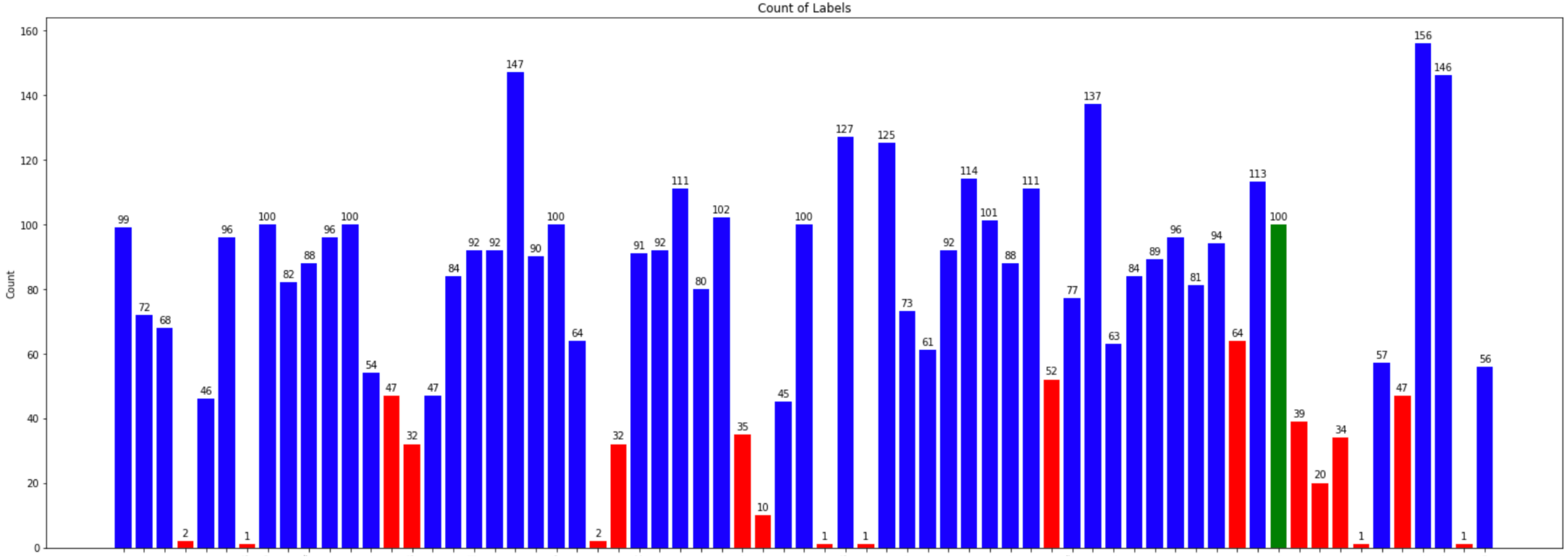
- Remove these 5 TTPs from the dataset and then train it again:
  - More prediction on the non-single triplet case is like this:



accuracy			0.97	309763
macro avg	0.63	0.62	0.62	309763
weighted avg	0.97	0.97	0.97	309763

Remove the Popular TTPs

- Remove these 5 TTPs from the dataset and then train it again:
- The # of the predicted labels



GNN - Paper

GraphSMOTE: Imbalanced Node Classification on Graphs with Graph Neural Networks

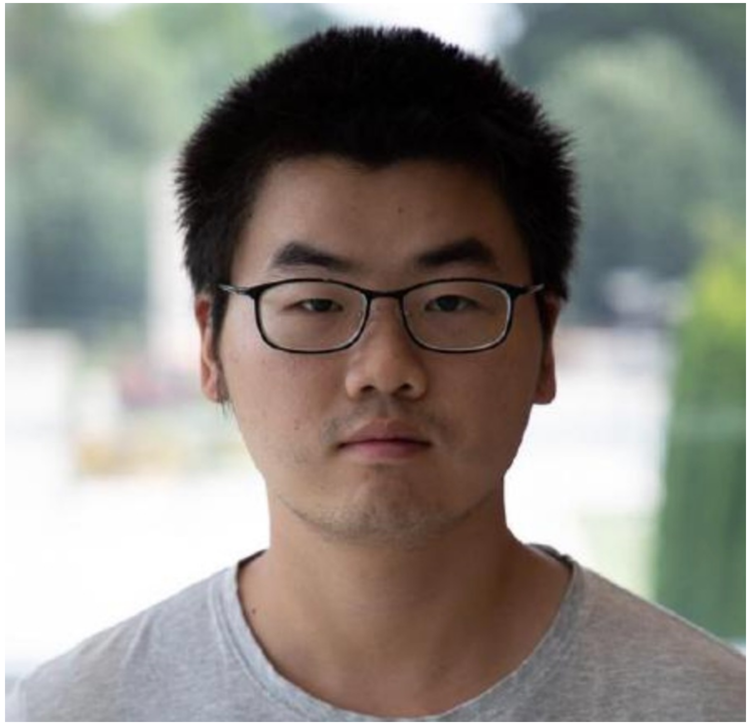
WSDM ’21, March 8–12, 2021, Virtual Event, Israel

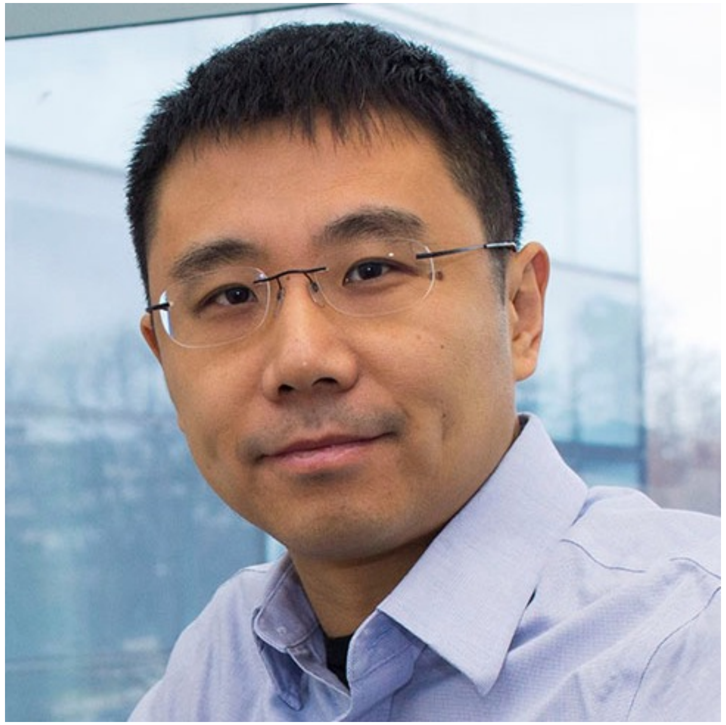
Tianxiang Zhao, Xiang Zhang, Suhang Wang

{tkz5084,xzz89,szw494}@psu.edu

College of Information Science and Technology, Penn State University State College, The USA

https://github.com/TianxiangZhao/GraphSmote





Tianxiang Zhao
PhD student in PSU

co-advised by Prof.Xiang Zhang and Prof.Suhang Wang.

https://scholar.google.com/citations?user=pXkPq3YAAAAJ&hl=en

Xiang Zhang
Associate Professor in PSU

Data Mining, Machine Learning and Big Data Analytics

https://ist.psu.edu/directory/xzz89

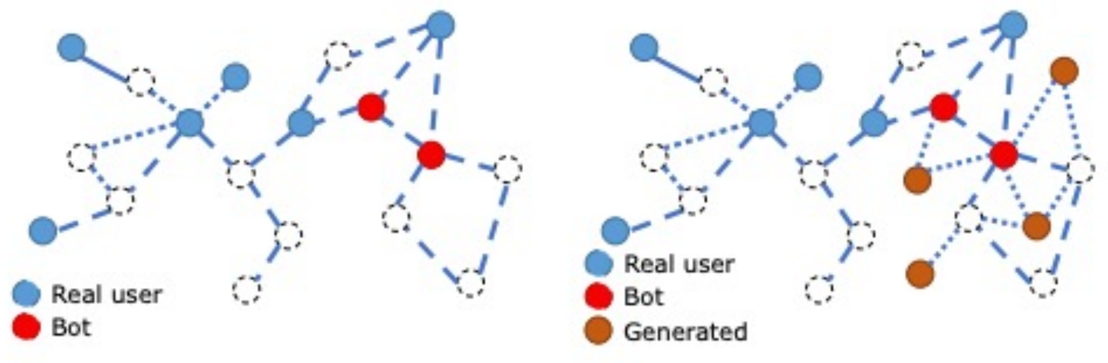
Suhang Wang
Associate Professor in PSU

Data Mining, Machine Learning and Social Media Mining

https://suhangwang.ist.psu.edu/

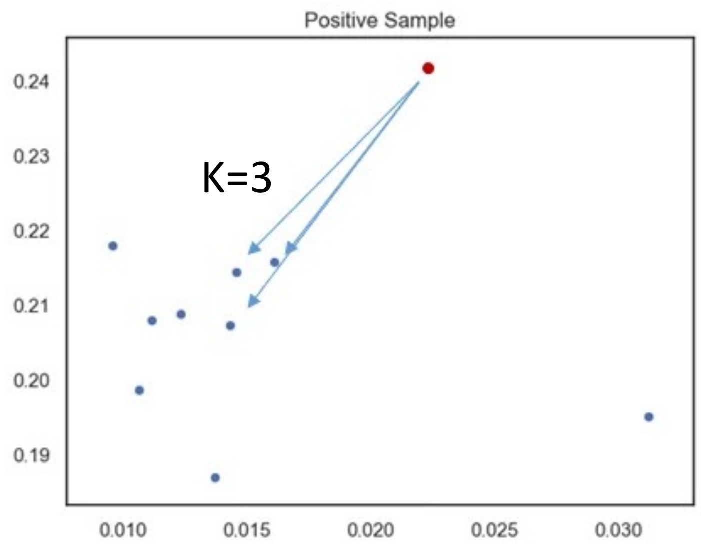
GraphSMOTE

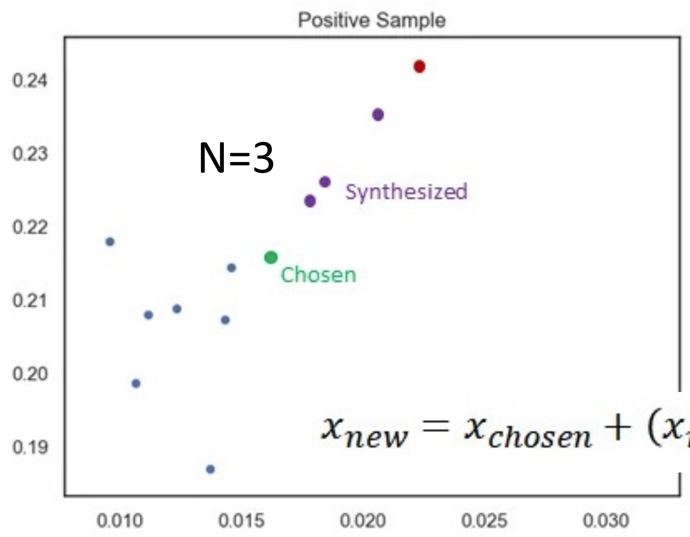
- **Task:**



(a) Bot detection task (b) After over-sampling

- **Synthesized Minority Oversampling Technique (SMOTE)**





- Used on i.i.d. data
- Doesn’t consider the structure of the graph
- No edge connection between the nodes

$$x_{new} = x_{chosen} + (x_{nearest} - x_{chosen}) * \delta \; ; \; \delta \in [0,1]$$

GraphSMOTE

- **Framework**
  - a GNN-based feature extractor
    → Use **GraphSAGE**
  - Synthetic Node Generation
    → Use **SMOTE** algorithm
  - Edge Generator
    → weighted inner product **decoder**
  - GNN Classifier (downstream task)

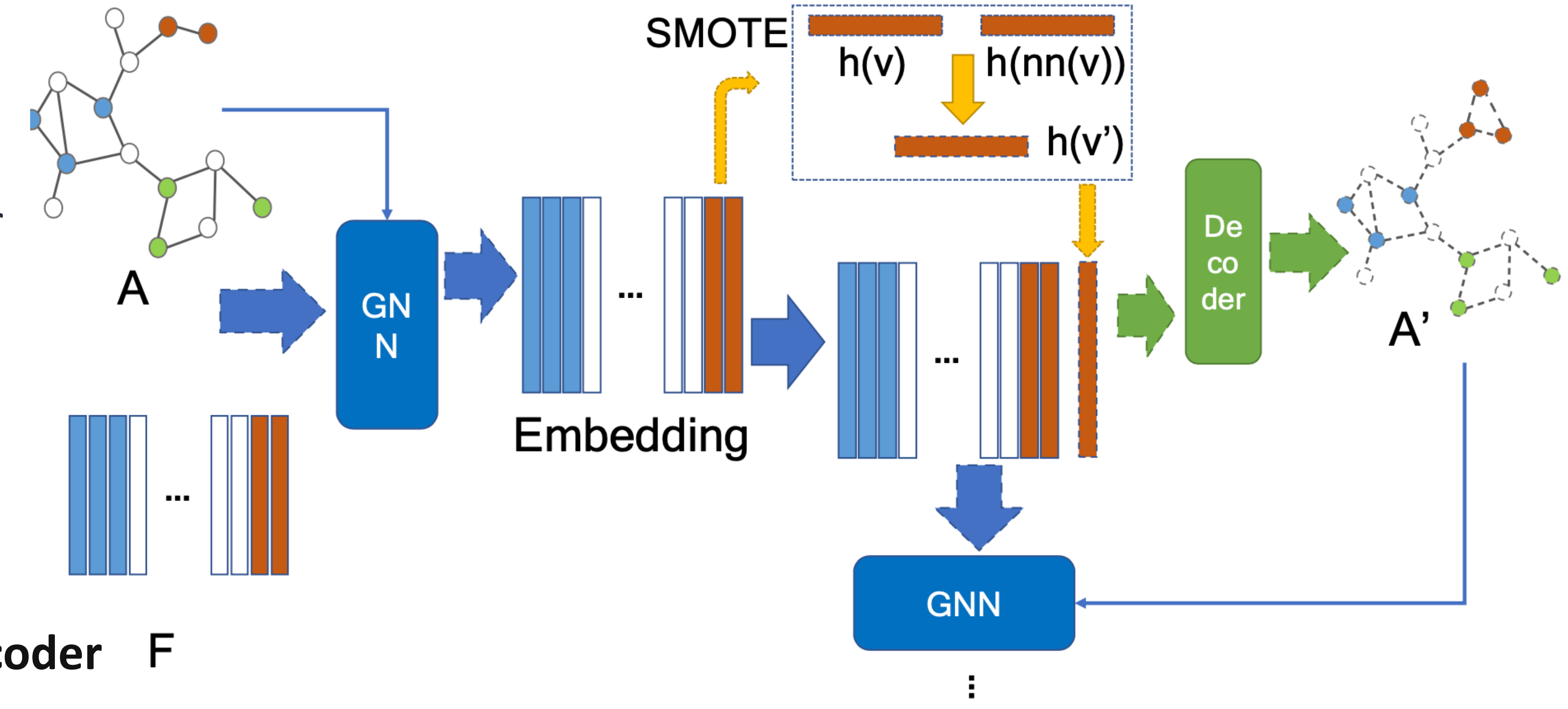


Figure 2: Overview of the framework

Future Work

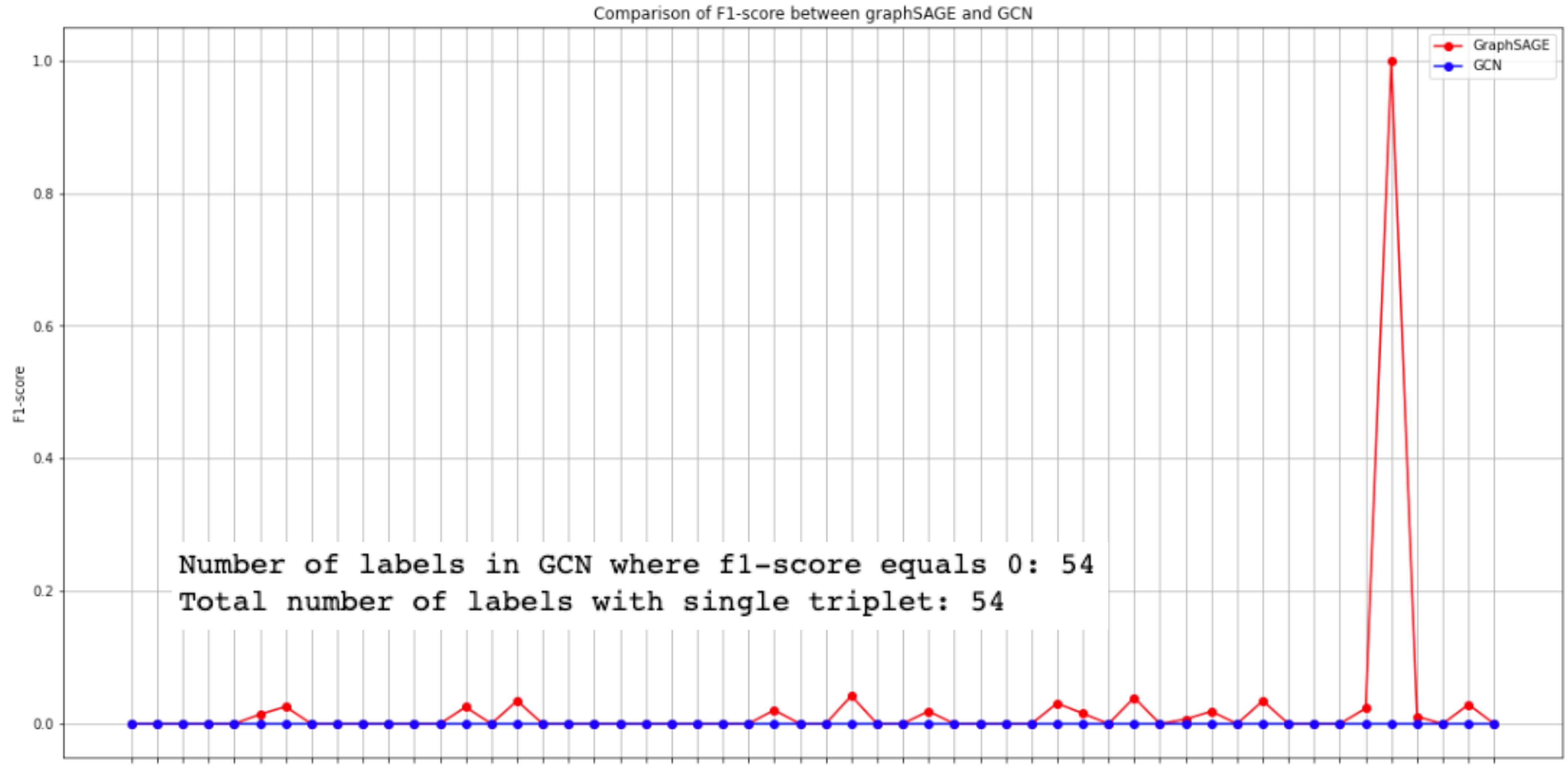
Future Work

- **GNN**
  - Try some other methods to improve the performance of single triplet issue
    - Figure out why the model can detect the **T1518_c9b** (if available?)
    - Read the **GraphSMOTE** paper

Thanks!!

Appendix

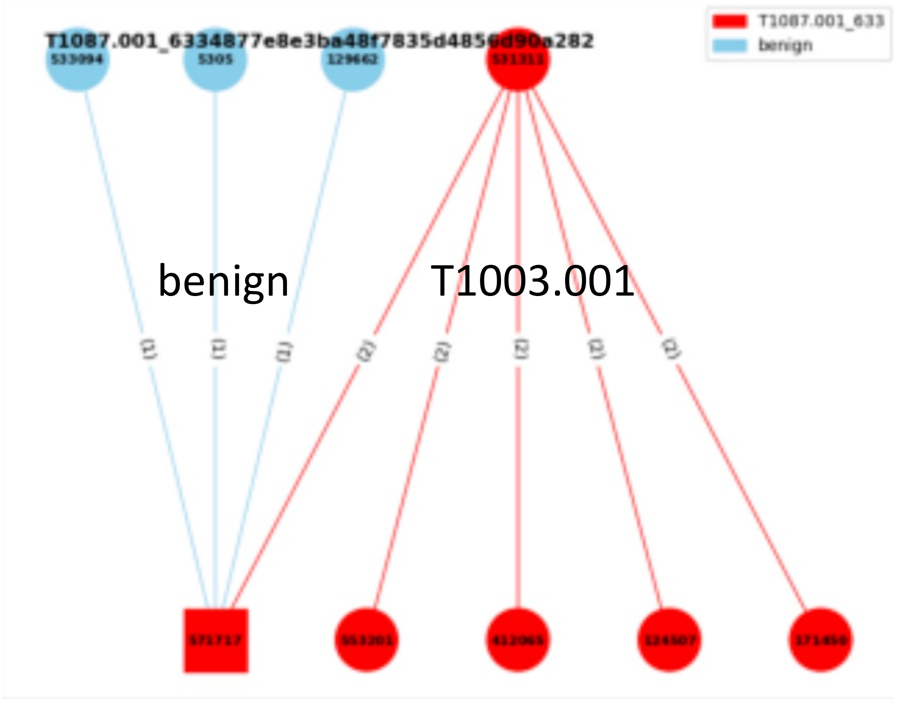
Observation on Different Model



Experiment 3

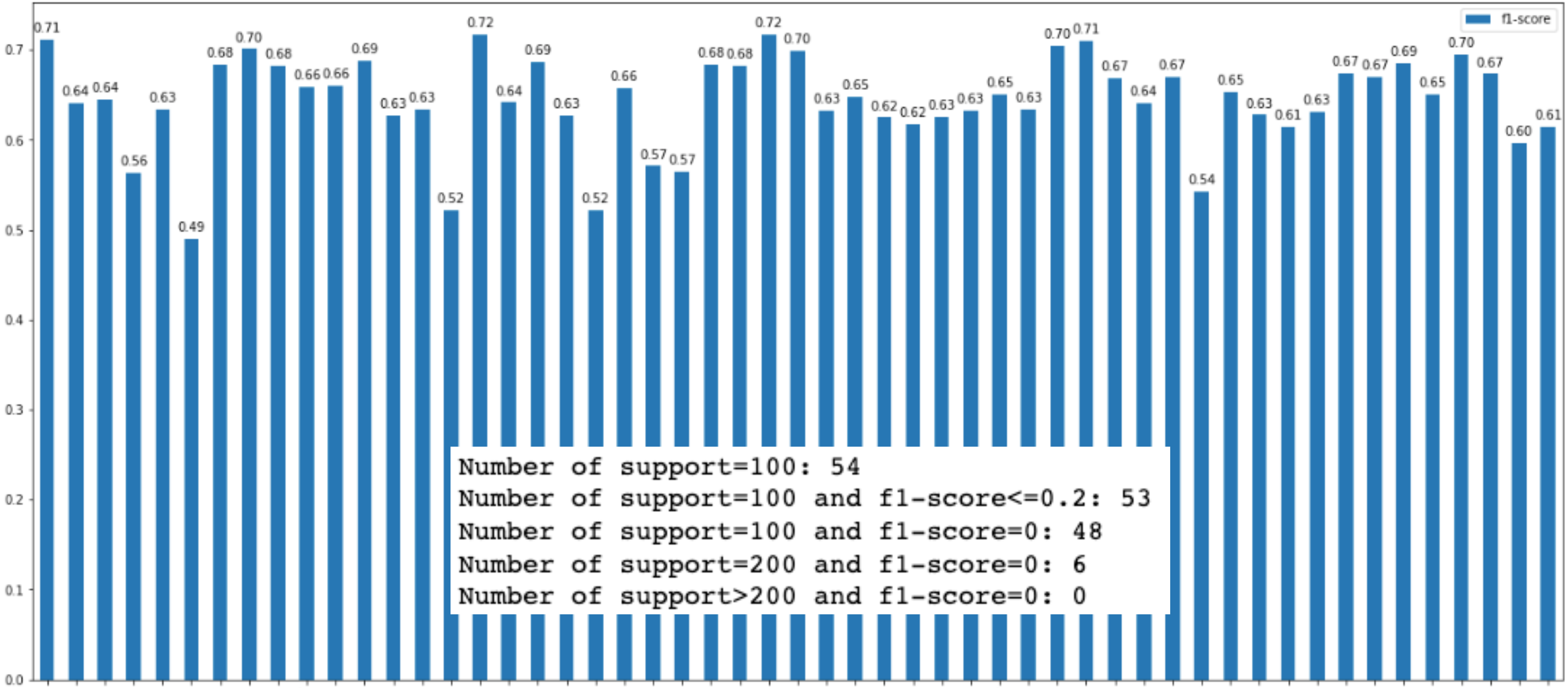
- **Experiment 3:**
  - Consider the **neighbor** benign nodes
  - Edge classification

  - Given a graph → label the triplets with the benign or the specific AP



Experiment 3 - Oversampling

Bar Chart of hidden dimension = 256:



Experiment 3 - Oversampling

- Current Trial: Duplicate the data with single triplets → 20, 40, 80, 320 times

20 times

```
Number of support=100: 54
Number of support=100 and f1-score<=0.2: 53
Number of support=100 and f1-score=0: 21
Number of support=200 and f1-score=0: 10
Number of support>200 and f1-score=0: 1
```

accuracy	0.971560	0.971560	0.971560	0.97156
macro avg	0.597445	0.600577	0.594684	310263.00000
weighted avg	0.970613	0.971560	0.970482	310263.00000

40 times

```
Number of support=100: 54
Number of support=100 and f1-score<=0.2: 53
Number of support=100 and f1-score=0: 14
Number of support=200 and f1-score=0: 10
Number of support>200 and f1-score=0: 2
```

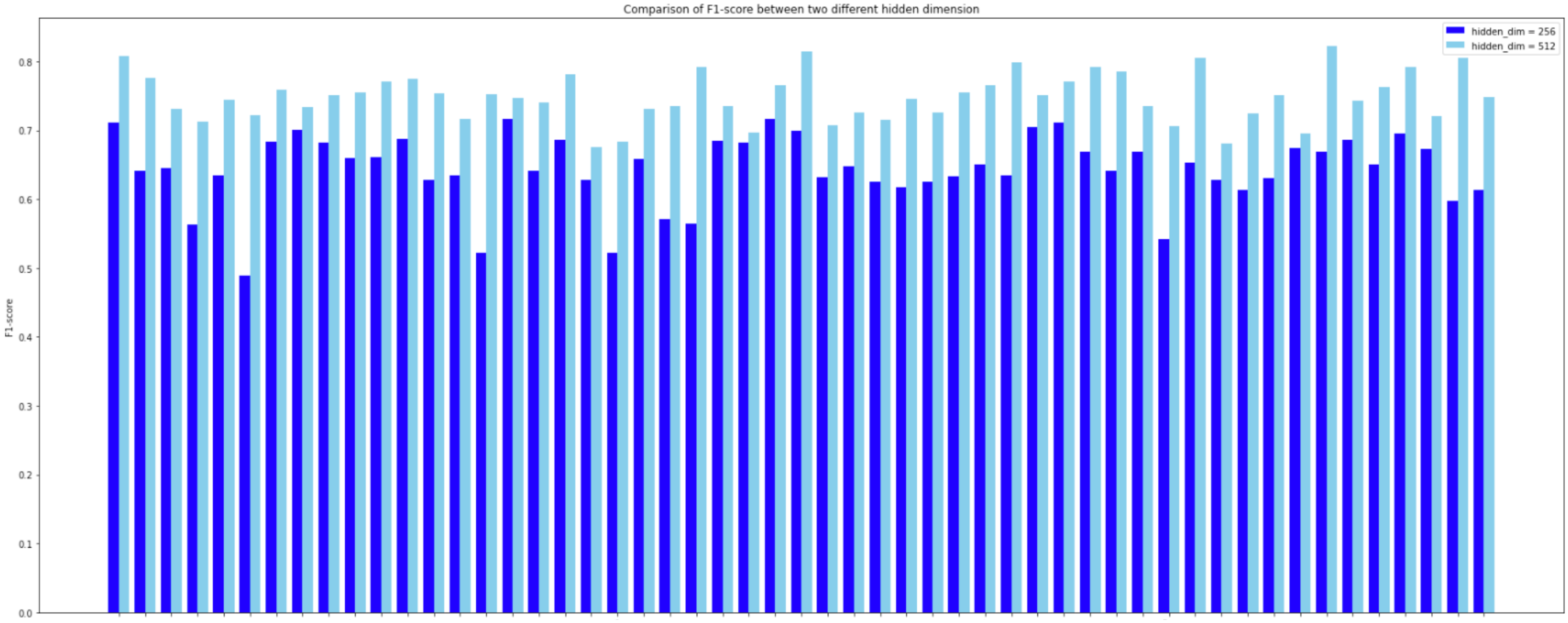
accuracy	0.971318	0.971318	0.971318	0.971318
macro avg	0.602542	0.597866	0.594387	310263.000000
weighted avg	0.971349	0.971318	0.970477	310263.000000

80 times

```
Number of support=100: 54
Number of support=100 and f1-score<=0.2: 53
Number of support=100 and f1-score=0: 18
Number of support=200 and f1-score=0: 10
Number of support>200 and f1-score=0: 3
```

accuracy	0.971463	0.971463	0.971463	0.971463
macro avg	0.596490	0.598077	0.594237	310263.000000
weighted avg	0.970626	0.971463	0.970567	310263.000000

Experiment 3 - Oversampling



Experiment 3 - Model

- Concept from the DGL official website:
    1. Let the dgl graph's edge data have the attribute: **edata["label"]**
    2. Use **GraphSAGE** model to get the new **node embedding**
    3. Use **MLP** model to get the **score** of the edge
    4. Concatenate these two models
    5. Train the final model

```
g.ndata['feat'] = th.tensor(data["node_feat"])
g.edata['feat'] = th.tensor(data["edge_attr"])
g.edata['label'] = th.tensor(data["labels"])
```

```
def model_fn(batched_g, model, criterion, device, count=1, which_type='train'):
    """Forward a batch through the model."""
    batched_g = batched_g.to(device)
    labels = batched_g.edata['label'].to(device)

    logits = model(batched_g, batched_g.ndata['feat'].float())
    loss = criterion(logits, labels)

    output = torch.softmax(logits, dim=1)
    preds = output.argmax(1)

    accuracy = torch.mean((preds == labels).float())
```

Experiment 3 - Model

```python
class GraphSAGE(nn.Module):
    def __init__(self, in_dim, hidden_dim, out_dim):
        super(GraphSAGE, self).__init__()
        self.layer1 = dglnn.SAGEConv(in_dim, hidden_dim, 'pool')
        self.layer2 = dglnn.SAGEConv(hidden_dim, out_dim, 'pool')

    def forward(self, g, inputs):
        h = self.layer1(g, inputs)
        h = torch.relu(h)
        h = self.layer2(g, h)
        return h
```

```python
class MLPPredictor(nn.Module):
    def __init__(self, out_feats, out_classes):
        super().__init__()
        self.W = nn.Linear(out_feats*2, out_classes)

    def apply_edges(self, edges):
        h_u = edges.src['h']
        h_v = edges.dst['h']
        score = self.W(torch.cat([h_u, h_v], 1))
        return {'score': score}

    def forward(self, graph, h):
        with graph.local_scope():
            graph.ndata['h'] = h
            graph.apply_edges(self.apply_edges)
            return graph.edata['score']
```

```python
class Model(nn.Module):
    def __init__(self, in_features, hidden_features, out_features, num_classes):
        super().__init__()
        self.sage = GraphSAGE(in_features, hidden_features, out_features)
        self.pred = MLPPredictor(out_features, num_classes)

    def forward(self, g, node_feat, return_logits=False):
        h = self.sage(g, node_feat)
        logits = self.pred(g, h)

        return logits
```

Experiment 3 - Result

- **Format of the edge labels:**
  - Label 65 is benign

```
labels of Test: tensor([155,  65, 155, 155, 155], device='cuda:0') torch.Size([5])
predicted of Test: tensor([155,  65, 155, 155, 155], device='cuda:0') torch.Size([5])
labels of Test: tensor([61, 61, 61], device='cuda:0') torch.Size([3])
predicted of Test: tensor([61, 61, 61], device='cuda:0') torch.Size([3])
```

- **Classification report**:
  - transR_50:

4a0dc2e1f5d1a	0.00	0.00	0.00	100
167175e8a019a	1.00	1.00	1.00	800
c3579e9e3737b	1.00	1.00	1.00	6200
43d838e0791ca	1.00	1.00	1.00	600
benign	1.00	1.00	1.00	134563
accuracy			0.97	310263
macro avg	0.60	0.61	0.60	310263
weighted avg	0.97	0.97	0.97	310263

  - secureBERT_50:

714a0dc2e1f5d1a	0.00	0.00	0.00	100
fb167175e8a019a	0.98	1.00	0.99	800
2ac3579e9e3737b	0.97	0.98	0.98	6200
0243d838e0791ca	0.91	0.83	0.87	600
benign	0.99	1.00	0.99	134563
accuracy			0.92	310263
macro avg	0.52	0.48	0.49	310263
weighted avg	0.90	0.92	0.91	310263

- **Macro average** is similar to previous experiments → won't be affected by benign
- **Weighted average** is very high since the # of the benign is high(unbalanced) and predictable
- TransX family performs better than secureBERT

Experiment 3 – Noise

- Current Trial 1:
  - Add the **noise** to the node feature

```python
def collate(samples):
    data_list = samples
    batched_graphs = []
    for data in data_list:
        g = dgl.graph((th.tensor(data["edge_index"][0]), th.tensor(data["edge_index"][1])), num_nodes=data["num_nodes"]

        node_feat = th.tensor(data["node_feat"])

        noise = th.normal(mean=0, std=0.01, size=node_feat.shape, device=node_feat.device)
        node_feat += noise

        g.ndata['feat'] = node_feat
        g.edata['feat'] = th.tensor(data["edge_attr"])
        g.edata['label'] = th.tensor(data["labels"])  # Add edge labels to graph

        batched_graphs.append(g)

    return dgl.batch(batched_graphs)
```

```
Number of support=100: 54
Number of support=100 and f1-score<=0.2: 53
Number of support=100 and f1-score=0: 46
Number of support=200 and f1-score=0: 4
Number of support>200 and f1-score=0: 0
```

Experiment 3 – K-fold validation

k-fold cross-validation（k 折交叉驗證）是一種在機器學習中常用的模型評估方法，尤其在有限的數據集上評估模型性能時非常有效。其主要優點包括：

1. **更可靠的性能估計**：通過將數據集分成 k 個子集，每次使用其中一個子集作為測試集，其餘 k-1 個子集作為訓練集，然後重複此過程 k 次，每次選擇不同的子集作為測試集，可以使得模型評估的結果更加穩定和可靠。
2. **充分利用數據**：每個數據點都被用作 k-1 次的訓練和 1 次的測試，這意味著每個數據點都被充分利用，這在數據量不大時尤其重要。
3. **減少偏差**：由於模型需要在 k 個不同的訓練集上訓練，然後在 k 個不同的測試集上測試，這有助於降低模型在特定數據集上的性能估計偏差。
4. **更好的泛化性能評估**：通過對不同的訓練集和測試集進行訓練和測試，可以幫助評估模型對未見數據的泛化能力。
5. **減少過擬合風險**：k-fold cross-validation 有助於識別模型是否對特定的訓練數據集過度擬合，因為它必須在多個不同的訓練集上表現良好。

然而，k-fold cross-validation 也有其局限性，例如計算成本較高（尤其是 k 較大和模型訓練時間較長時），並且結果可能依賴於 k 的選擇以及數據分割的方式。這些都是在使用此方法時需要考慮的因素。