Supplementary Materials for the Submission Titled

Iterative Graph Neural Network Enhancement using Explanations

In Appendix A we collect the necessary background about graph neural networks, the GNNEXPLAINER system, and frequent connected subgraph mining. Appendix B gives a high-level pictorial representation of the EEGL framework. In Appendix C we describe the *pattern-extraction module* which is a pre-requisite step for the *top-k* pattern filtering and the annotation phases. In Appendix D we report the detailed experimental results obtained for M_2' (Fig. 1e in the submission) for all 10 folds. We have currently provided source code as a de-identified zip file with the supplementary material. If the paper is accepted then the source code will be publicly shared on Github.

A BACKGROUND

For necessary background on GNN, GNNEXPLAINER, the Weisfeiler-Leman algorithm, and frequent subgraph mining, the reader is referred to (15; 23; 27).

Graph Neural Networks (GNN) We give a simplified description of GNN, corresponding to the type of networks used in GNNEXPLAINER (27). In this paper we consider node classification. A GNN model Φ has as its inputs a graph $G = (V, E)$ and a feature matrix $X \in \mathbb{R}^{d \times |V|}$, where $X_v \in \mathbb{R}^d$ is a d-dimensional feature vector associated for every node $v \in V$. It computes the representation z_v, the *embedding* of each node $v \in V$, via L layers of *neural message passing* as follows. Starting with initial node features $h_i^0 = X_{v_i}$ for every $v_i \in V$, repeat the following three steps for each layer $l \in [L]$:

(i) Compute neural messages $m_{ij}^l = \text{MSG}(h_i^{l-1}, h_j^{l-1}, r_{ij})$ for every pair of nodes $(v_i, v_j) \in V$, where MSG is the function which computes the message passed from v_j to v_i, and r_{ij} is a relation between v_i and v_j, e.g., the edge relation. Usually the message is h_j^{l-1} if $(v_i, v_j) \in E$, else 0.

(ii) For each node v_i aggregate the messages from its neighborhood $\mathcal{N}_{v_i}$ as $M_i^l = \text{AGG}(m_{ij}^l | v_j \in \mathcal{N}_{v_i})$, where AGG is the function used to aggregate all the messages passed from the neighborhood. AGG should be invariant or equivariant to the permutations of its inputs.

(iii) For every node v_i update its features for using the aggregated messages from the neighborhood and the previous features as $h_i^l = \text{UPDATE}(M_i^l, h_i^{l-1})$, where UPDATE is the function used to combine the two inputs.

After the computation through the L layers, the embedding for every node v_i is the features computed for the L'th layer i.e. $z_{v_i} = h_i^L$. For a fully supervised node classification task, the training loss is $\mathcal{L} = \sum_{v \in \mathcal{V}_{train}} -\log(\text{softmax}(z_v, y_v))$, where $y_v \in \{0, 1\}^c$ is *one-hot* vector indicating the ground truth label for node v, c is number of labels, and $\mathcal{V}_{train}$ is the set of nodes in the training set.

GNNEXPLAINER (28) was the first general model-agnostic approach specifically designed for post-hoc explanations on graph learning tasks. Given a graph G, and a query in the form of a node in the graph, it identifies a compact subgraph structure and a small subset of node features that show significant evidence in playing a role in the prediction of the query node. The GNNExplainer procedure accomplishes this by formulating the explanation generation problem as an optimization task that maximizes the mutual information between a GNN's prediction and subgraph structures.

Frequent Subgraph Mining Frequent subgraph mining, a well-established subfield of data mining, is concerned with the following problem: *Given* some finite (multi)set $\mathcal{D}$ of graphs and a frequency threshold $\tau \in (0, 1]$, *generate* all *connected* graphs that are subgraphs of at least $\lceil \tau |\mathcal{D}| \rceil$ graphs in $\mathcal{D}$. In the EEGL system we use GASTON (23), one of the most popular algorithms for the generation of frequent connected subgraphs. Similarly to most other heuristics designed for frequent subgraph mining, GASTON is based on the generate-and-test paradigm. One if its distinguishing features

compared to other frequent subgraph mining algorithms is that it enumerates frequent patterns in accordance with a structural hierarchy (path, trees, and cyclic graphs), where cyclic patterns are generated in a DFS manner, "refining" frequent patterns by extending them with some new edge. Regarding its pattern matching component, i.e., which decides whether a candidate pattern is frequent or not, GASTON utilizes that frequent patterns are generated in increasing size. Instead of testing subgraph isomorphism from scratch for the graphs in $\mathcal{D}$, it stores the embeddings of the frequent patterns already found and decides subgraph isomorphism for a candidate pattern by utilizing the embeddings calculated for its antecedents.

B EXPLANATION ENHANCED GRAPH LEARNING

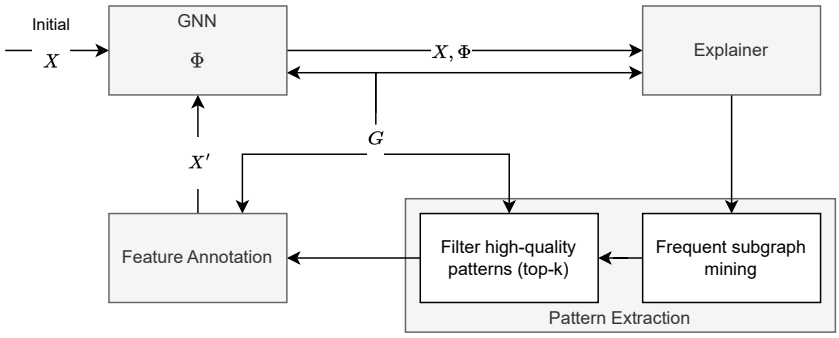


Figure 4: High-level depiction of the EEGL Process

Figure 4 shows a high level abstraction of the four major phases of the EEGL framework. We also note that the *pattern extraction* module consists of two sub-modules 1. Frequent-subgraph mining 2. Top-k pattern filtering

C PATTERN EXTRACTION MODULE

For a class label $c \in C$, function MAXIMAL_FREQUENT_PATTERN_MINING (line 9 of Alg. 1) computes a set of maximal frequent rooted patterns in $\mathcal{E}_c$ in two steps: (i) It generates a set of frequent rooted subgraphs of $\mathcal{E}_c$ and (ii) selects the maximal rooted patterns from this set.

Regarding (i), the following frequent pattern mining problem is solved: *Given* $\mathcal{E}_c$ containing m explanation graphs for some $m \geq 0$ integer and a frequency threshold $\tau \in (0, 1]$, *enumerate* the set of rooted patterns (P, r) such that (P, r) is frequent, i.e., there is a set $\mathcal{E}'_c \subseteq \mathcal{E}_c$ of rooted explanation graphs such that $|\mathcal{E}'_c| = \lceil \tau m \rceil$ and for all $(P', r') \in \mathcal{E}'_c$ there exists a rooted subgraph isomorphism from (P, r) to (P', r'). Note that $\mathcal{E}_c$ may contain explanation graphs that are not associated with a root. While such explanation graphs do not support any of the frequent rooted patterns, they have an impact on the rooted patterns' (relative) frequencies. The above problem is solved by GASTON (22) as follows: For all rooted explanation graphs $(P, v) \in \mathcal{E}_c$, v is associated with a distinguished node attribute label. Running GASTON on these modified explanation graphs with frequency threshold τ, it returns a set of frequent subgraphs. From this set we keep only the connected components of the frequent subgraphs that contain a node with the distinguished attribute label. This node will be regarded as the root of the pattern.

Regarding (ii), we remove all rooted patterns returned in the previous step that have a rooted subgraph isomorphism to some other rooted pattern and return the remaining rooted pattern set.

D RESULTS

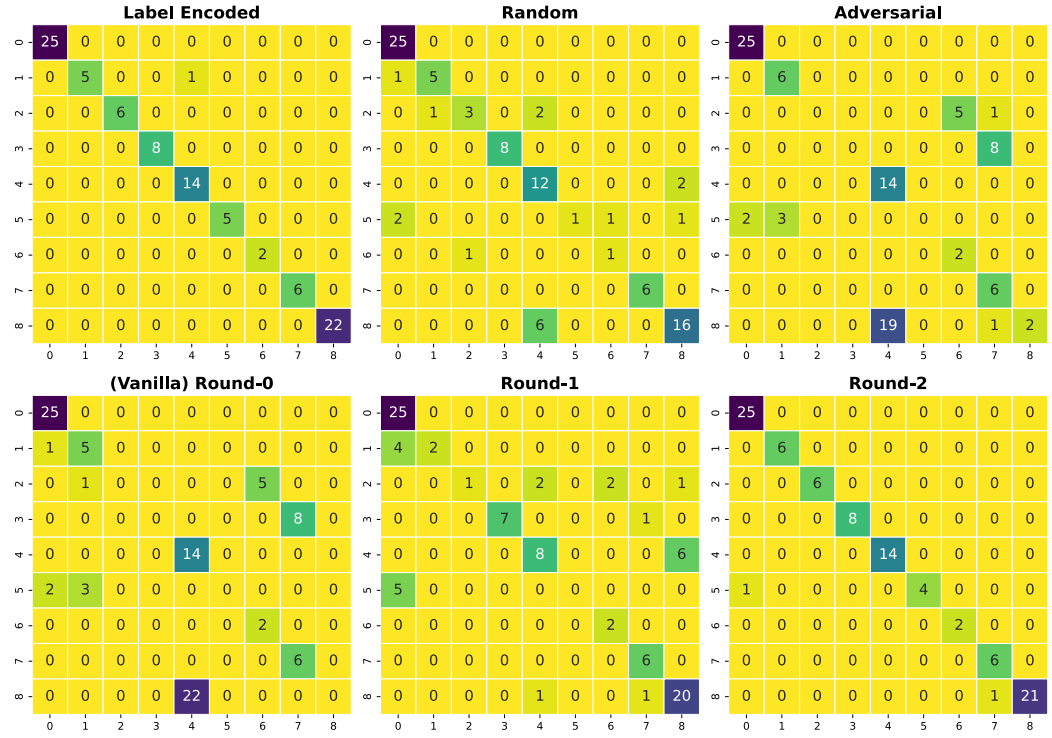
In this section we present the detailed results obtained for M'_2 (Fig. 1e in the submission). For each of the 10 folds, we present the

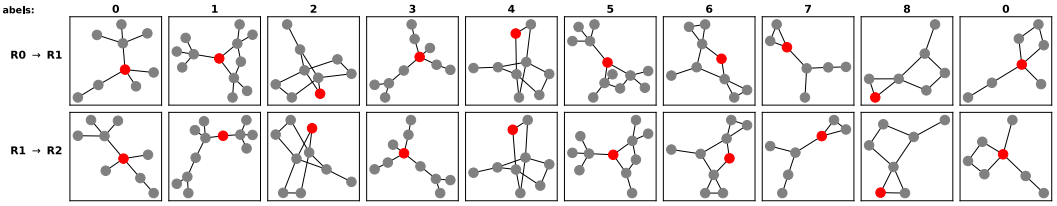
- the weighted F1-score results in percentage (top left table) obtained with the label encoding, random, and adversarial settings, as well as after the three iterations of EEGL (Round-0, Round-1, Round-2),
- the frequencies of the node labels in the test data (top right table),
- the confusion matrices obtained with the label encoding, random, and adversarial settings, as well as after the three iterations of EEGL (Round-0, Round-1, Round-2),
- the $d = 10$ maximal frequent subgraphs extracted by EEGL in the first (R0 $\rightarrow$ R1) and the second (R1 $\rightarrow$ R2) iteration. (Class labels are indicated on the top. Note that we can have more than one pattern for a node label.)

FOLD 01

	F1-Score
Label Encoded	98.91
Random	80.75
Adversarial	47.88
(Vanilla) Round-0	42.46
Round-1	71.02
Round-2	97.85

	0	1	2	3	4	5	6	7	8
Frequency	25	6	6	8	14	5	2	6	22

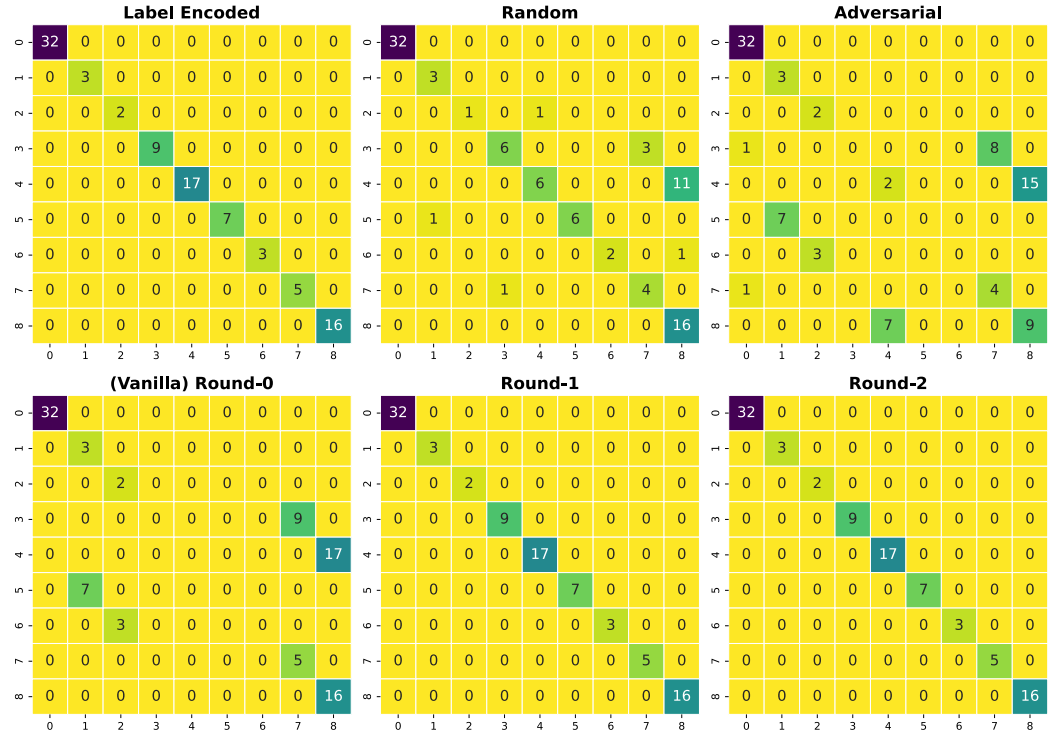


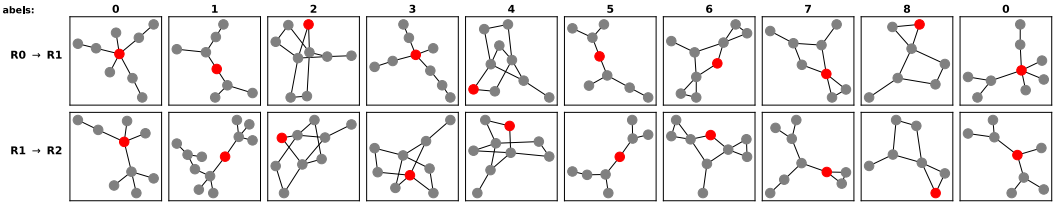


FOLD 02

	F1-Score
Label Encoded	100.00
Random	79.77
Adversarial	48.64
(Vanilla) Round-0	50.65
Round-1	100.00
Round-2	100.00

	0	1	2	3	4	5	6	7	8
Frequency	32	3	2	9	17	7	3	5	16

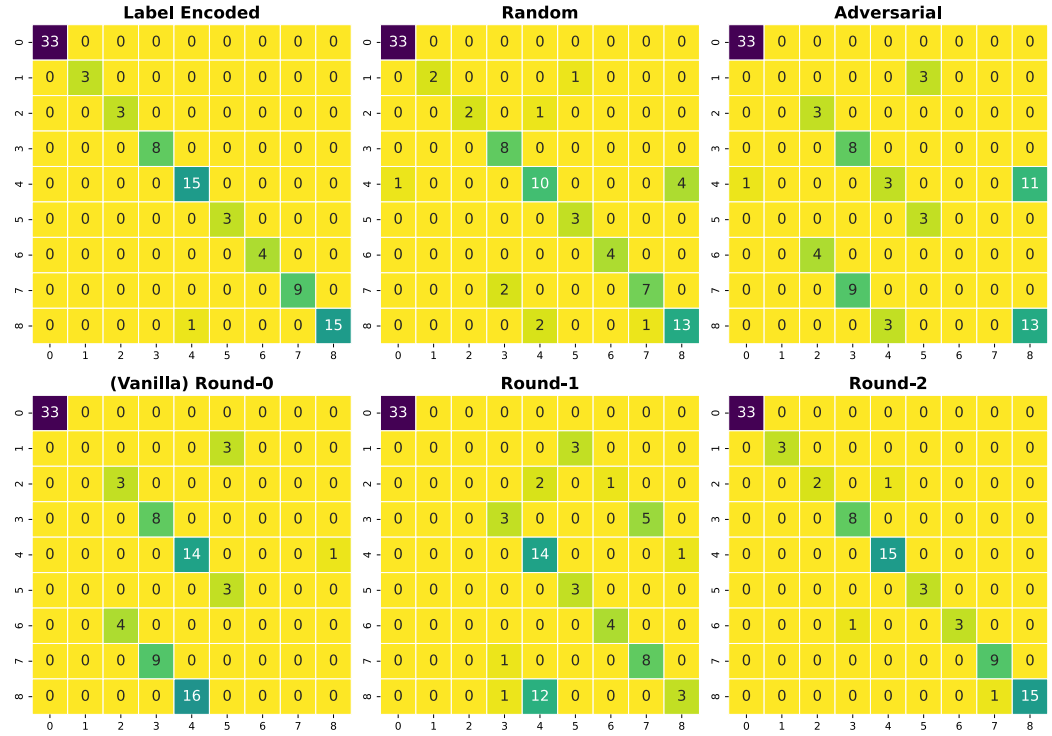


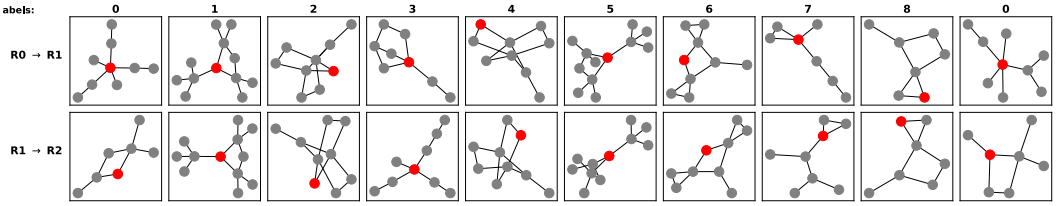


FOLD 03

	F1-Score
Label Encoded	98.94
Random	86.94
Adversarial	59.69
(Vanilla) Round-0	54.52
Round-1	67.41
Round-2	96.69

	0	1	2	3	4	5	6	7	8
Frequency	33	3	3	8	15	3	4	9	16

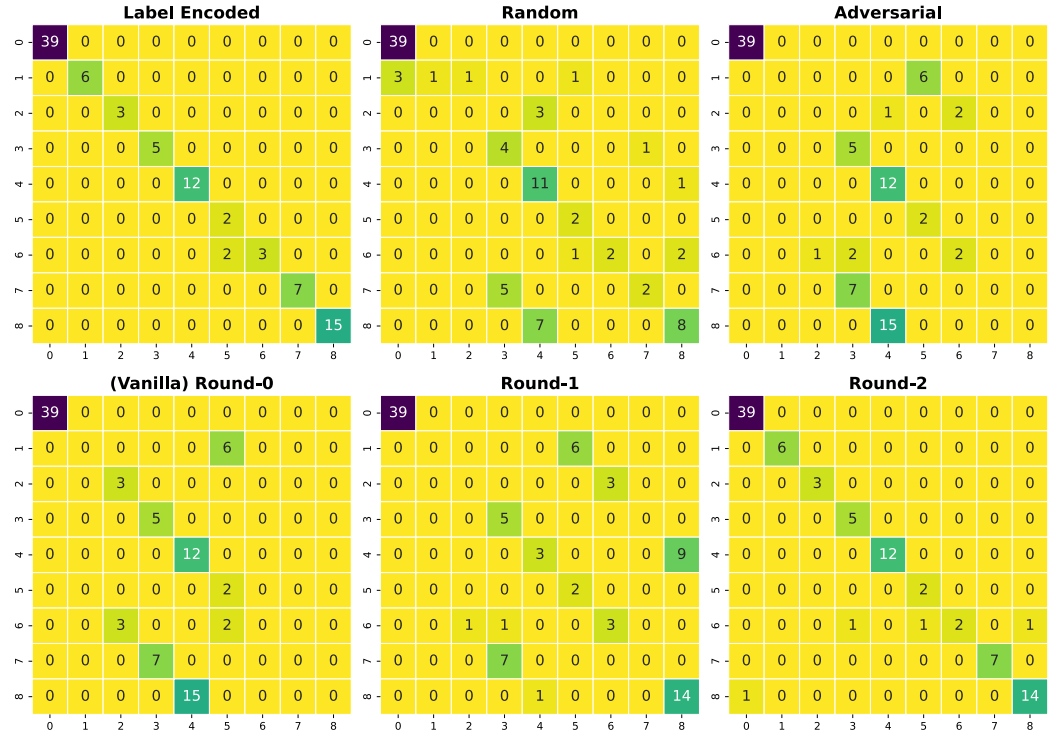


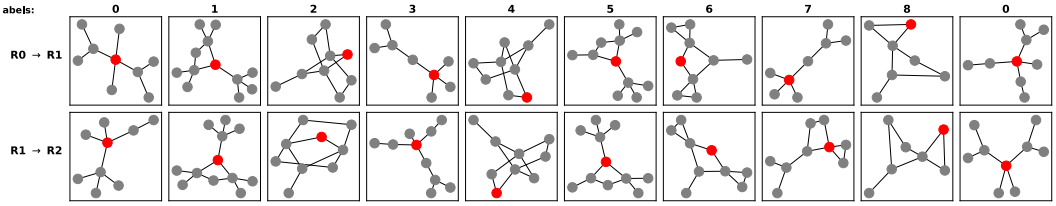


FOLD 04

	F1-Score
Label Encoded	97.96
Random	70.58
Adversarial	55.16
(Vanilla) Round-0	55.31
Round-1	64.74
Round-2	95.22

	0	1	2	3	4	5	6	7	8
Frequency	39	6	3	5	12	2	5	7	15

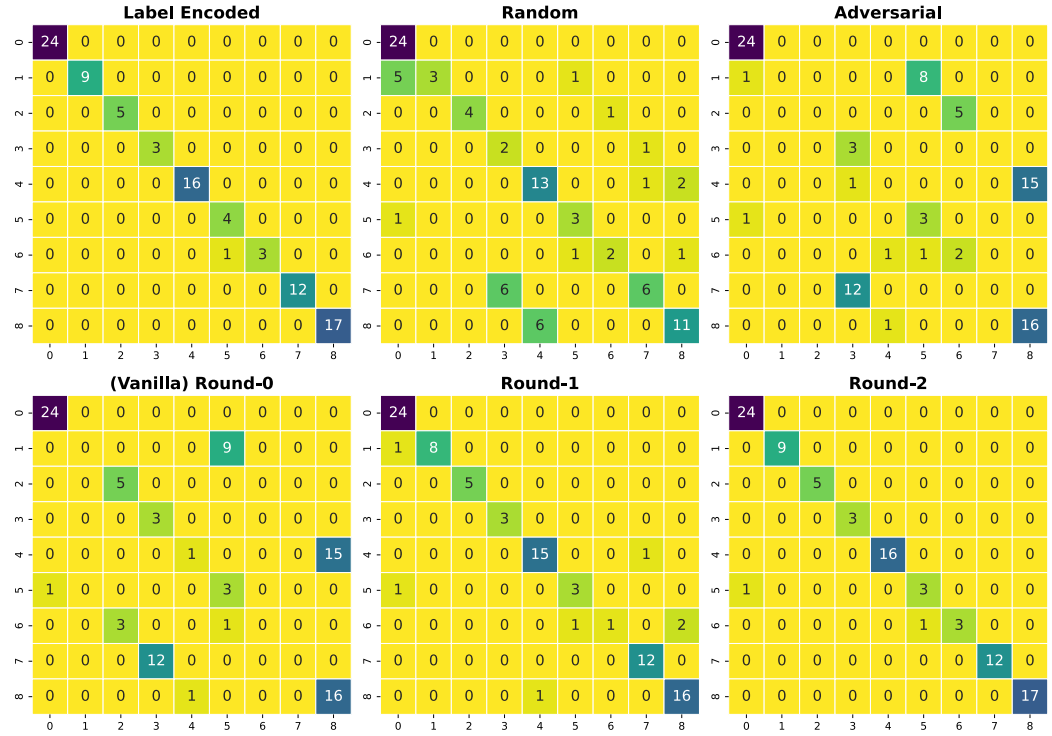


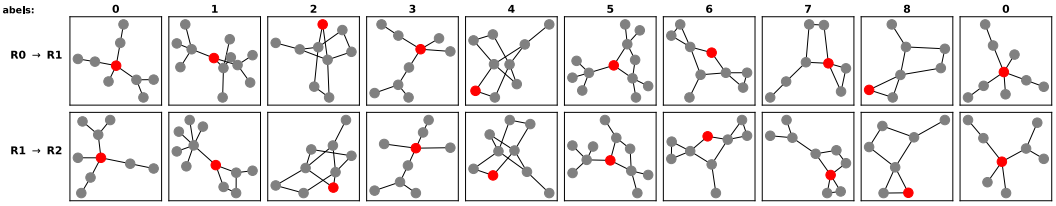


FOLD 05

	F1-Score
Label Encoded	98.92
Random	71.78
Adversarial	40.72
(Vanilla) Round-0	45.62
Round-1	91.67
Round-2	97.81

	0	1	2	3	4	5	6	7	8
Frequency	24	9	5	3	16	4	4	12	17

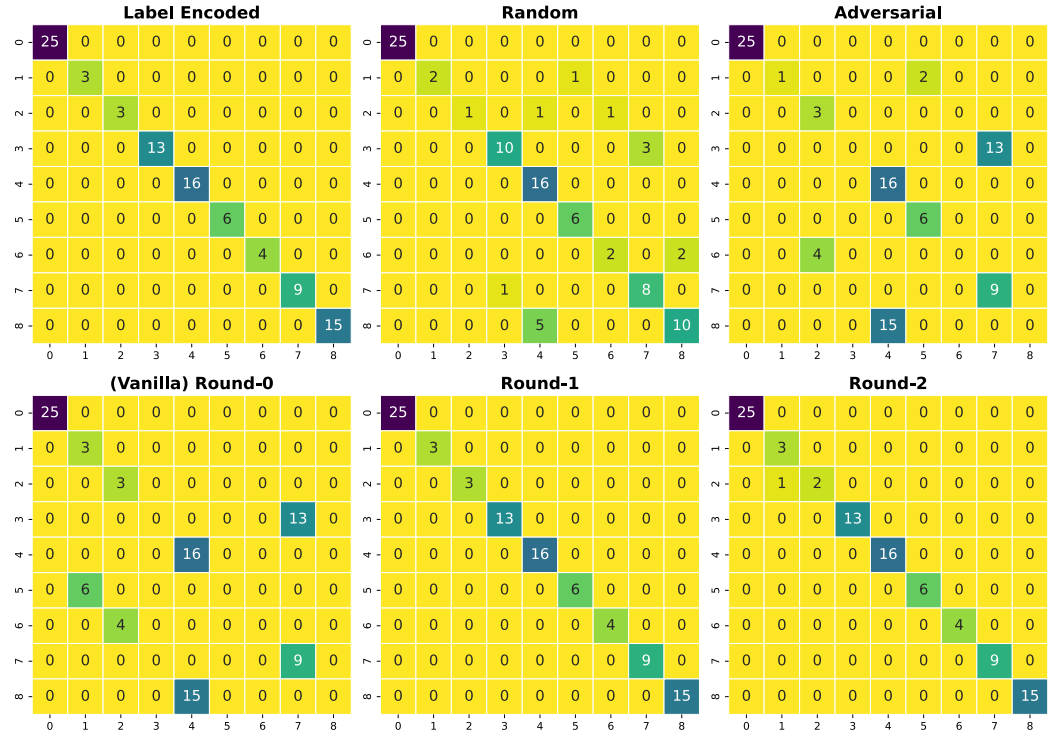


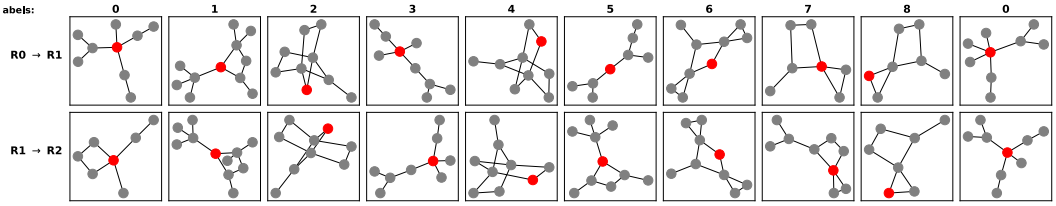


FOLD 06

	F1-Score
Label Encoded	100.00
Random	84.41
Adversarial	52.73
(Vanilla) Round-0	47.25
Round-1	100.00
Round-2	98.91

	0	1	2	3	4	5	6	7	8
Frequency	25	3	3	13	16	6	4	9	15

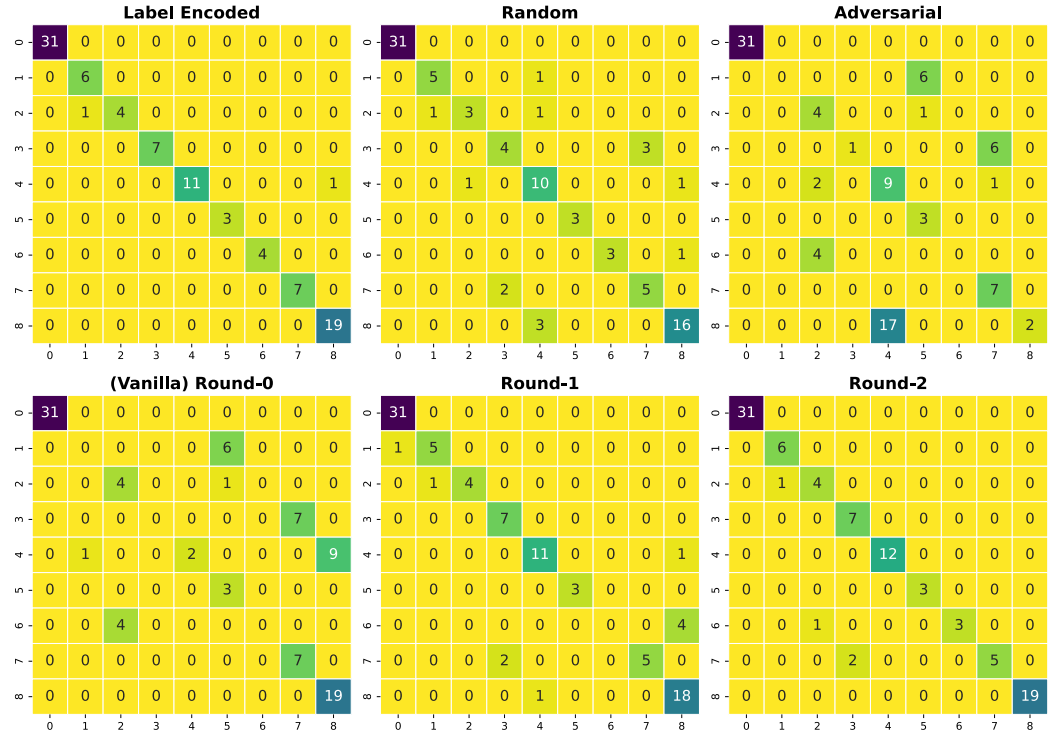


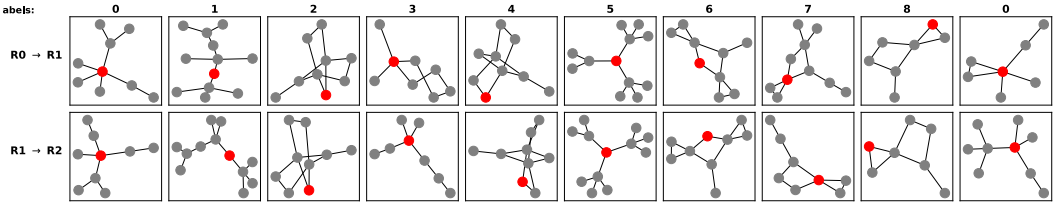


FOLD 07

	F1-Score
Label Encoded	97.84
Random	85.17
Adversarial	54.01
(Vanilla) Round-0	62.68
Round-1	87.44
Round-2	95.67

	0	1	2	3	4	5	6	7	8
Frequency	31	6	5	7	12	3	4	7	19

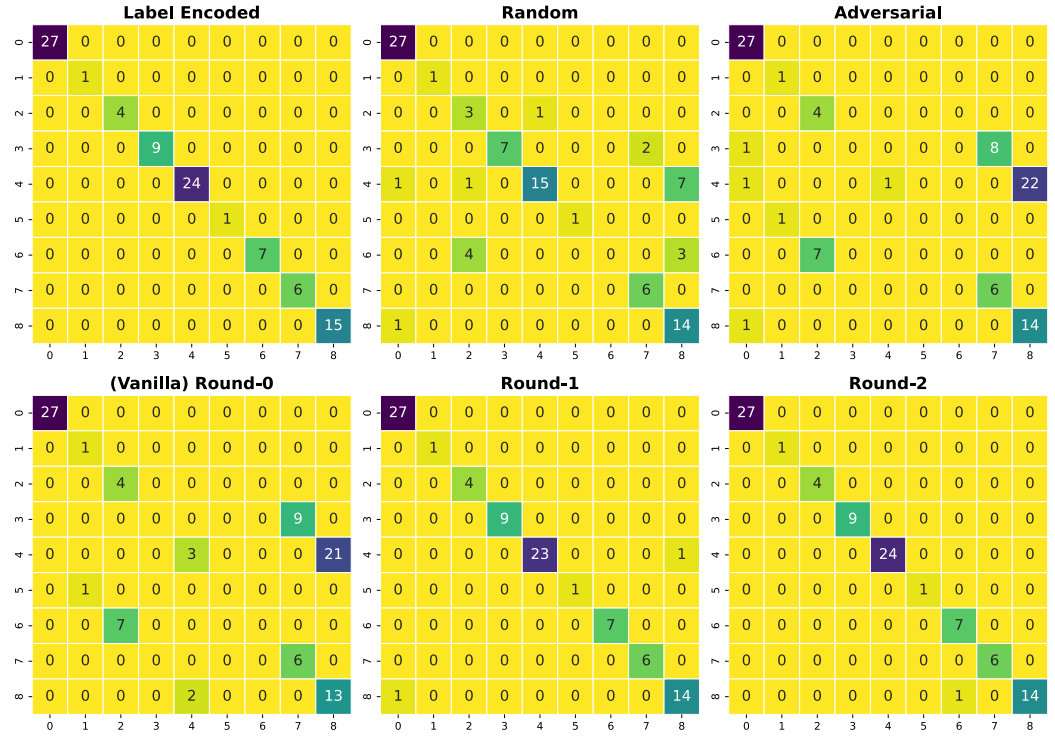


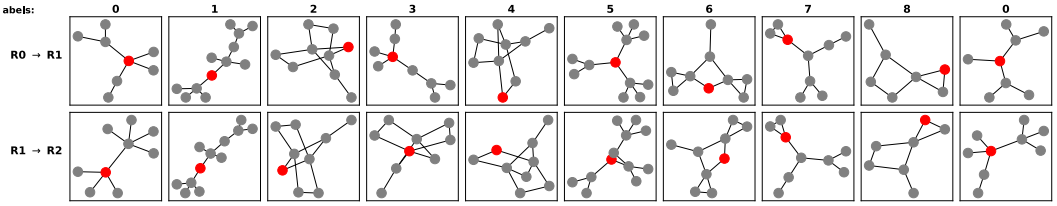


FOLD 08

	F1-Score
Label Encoded	100.00
Random	76.41
Adversarial	44.82
(Vanilla) Round-0	49.10
Round-1	97.87
Round-2	98.95

	0	1	2	3	4	5	6	7	8
Frequency	27	1	4	9	24	1	7	6	15

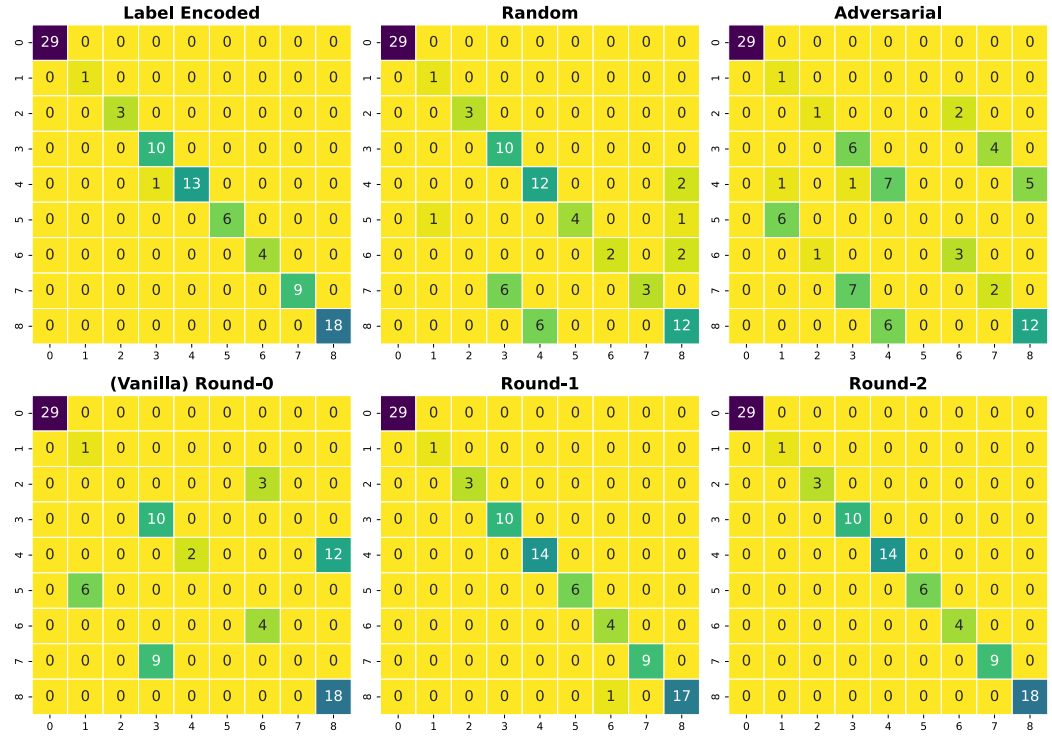


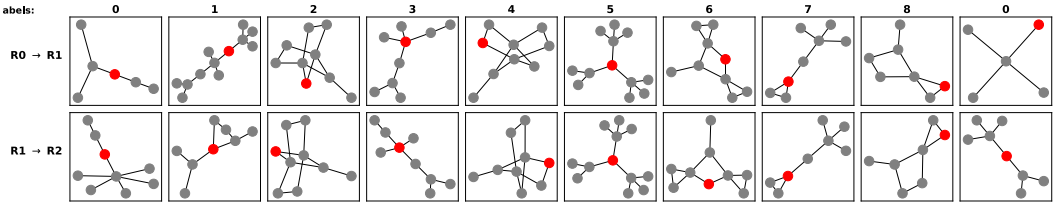


FOLD 09

	F1-Score
Label Encoded	98.94
Random	79.97
Adversarial	63.93
(Vanilla) Round-0	59.63
Round-1	98.98
Round-2	100.00

	0	1	2	3	4	5	6	7	8
Frequency	29	1	3	10	14	6	4	9	18





FOLD 10

	F1-Score
Label Encoded	100.00
Random	78.57
Adversarial	57.24
(Vanilla) Round-0	55.21
Round-1	100.00
Round-2	100.00

	0	1	2	3	4	5	6	7	8
Frequency	35	2	6	8	20	3	3	10	7