GNN report

Rigers Aliaj

September 2025

1 Data and features

We use the denominator graph edges $7, 8, 9, 10, 11$ loops as well as their corresponding coefficients to train a GNN on binary classification. The coefficient is either 1 or 0, depending on whether there is a numerator completion of the graph that contributes to the correlator. We input various features (as node features) into the model and see which lead to the best performance. The whole list of features considered follows

1. Columns of the adjacency matrix
2. Columns of identity matrix
3. Laplacian eigenvectors with the 3 biggest eigenvalues
4. degree
5. betweenness
6. clustering
7. pagerank
8. closeness
9. face count

Eventually features $3 - 9$ were the most expressive for the GNN. We ran the model with just the identity matrix columns as features to understand the importance of the other features. The experiments on 7,8 loops led to the model not even managing to learn the training set and no further experiments were made. Moreover, adding the identity columns to the set of features made the models perform worse.

2 Runs and results

2.1 GIN

Intro In terms of the architecture, we almost exclusively work with a simple GINN (Graph Isomorphism Neural Network). This ensures that the model will treat equally graphs that are isomoprhic. However it might still fail to distinguish between non-isomorphic graphs that cannot be distinguished by the 1-WL graph isomorphism test. This makes it as expressible as the 1-WL test. The architecture is expected to be ideal for graph level tasks with few nodes.

Based on features 3-9 we trained and tested the model for graphs of various loop orders. Since we deal with binary classification, the choice of a threshold affects heavily whether a graph will be identified with 0 or 1. Therefore, we present the ROC AUC and PR AUC metrics which are threshold independent. ROC AUC is the area under the curve of true positive rates (TPR) and false positive (FPR) rates, for different thresholds. These are defined as:

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}, \qquad \mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}} \tag{1}$$

The value ROC AUC can be interpreted as follows: It gives the probability of a classifier to rank a randomly chosen denominator graph that contributes (coefficient=1) higher than one that does not contribute (coefficient=0). With the increase in the number of loops however, most of the graphs at hand do not contribute. For instance, at 11-loop just 10 % of the graphs have coefficient 1. In this case FPR becomes very small, since there are many TN resulting in an artificially large FPR. To overcome this problem we use PR AUC. This computes the area under the precision-recall curve for different thresholds. These are defined as[1]

$$\mathrm{Precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}, \qquad \mathrm{Recall} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{2}$$

[1]In our application we are interested in reducing the ansatz for the correlator, therefore including extra graphs in the ansatz that eventually do not contribute (recall on 0) is not as bad removing graphs that eventually contribute to the ansatz (recall on 1). Thus, instead of using ROC AUC and PR AUC we can optimize the threshold to maximize the recall on 1's.

Note that since we are mostly interested in minimizing FN while FP are not a problem, the performance of the model on an unbalanced set (negatives dominate) is better described by PR AUC. The reason is that these metrics focus on the minority class (TP,FN,FP) only.

Machines & Architecture The single loop runs are performed with 3 layers of 64 hidden channels. Then for testing on 10 loops we used 256 hidden channels and finally for testing on 11 loops we chose 512 hidden channels. Finally, the following choices have been made: learning rate: 0.0003, weight decay: 0.00003, dropout: 0.3, epochs=100. The machine used for single-loop training has the following specs:

CPU: Intel(R) Core(TM) i5-8365U CPU @ 1.60GHz (8 cores)

GPU: Intel Corporation WhiskeyLake-U GT2 [UHD Graphics 620] (rev 02)

RAM: 15Gi.

The multi-loop ones were performed with the Maxwell Cluster with the following machines used:

CPU: 96 cores, 754Gi RAM

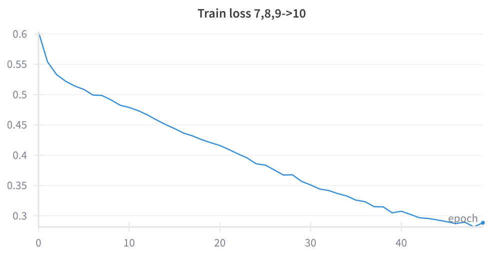
GPU: 4xNVIDIA L40S, 46068 MiB

Results Next follows a table with our findings. We use a 80-20 split for training and testing on single-loop experiments. For multi-loop experiments we take a random 80% of the training set for training and we test on all test graphs. The randomness of the chosen subset of graphs helps to avoid the shifting of the model towards special examples.

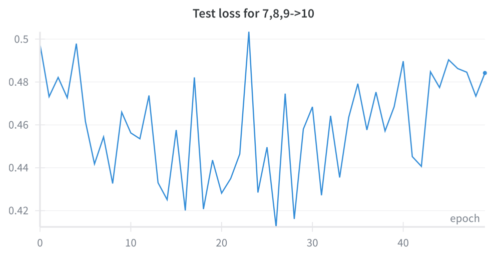
Table 1: Experimental Results GIN

ROC/PR AUC (%)	$8 \to 8$	$9 \to 9$	$7,8,9 \to 10$	$7,8,9,10 \to 11$
train	96.0/96.2	98.4/97.7	99.0/98.9	99.9/ 99.7
test	85.8/86.2	95.8/93.9	92.6/86.4	97.2/ 93.2
time (s)	28	171	720	5185

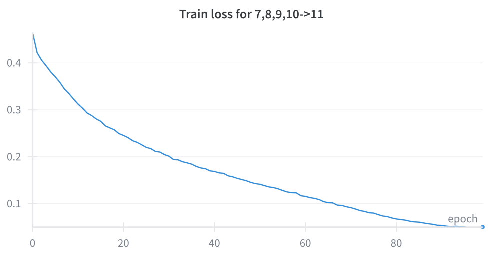
We observe that for the multiloop experiments (especially the one predicting 10-loop graph coefficients) the test metrics are not as good as the training ones. This is a typical sign of overfitting. In our binary classification we use `nn.BCEWithLogitsLoss` without any positive weight and with precision=0.5. The relative loss curves have the following forms:



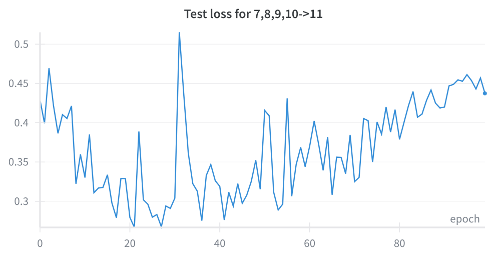
(a) Train $7, 8, 9 \to 10$



(b) Test $7, 8, 9 \to 10$



(c) Train $7, 8, 9, 10 \to 11$

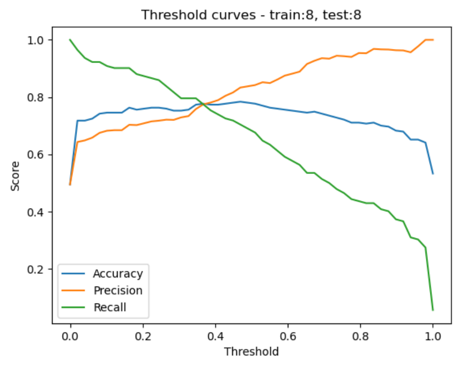


(d) Test $7, 8, 9, 10 \to 11$

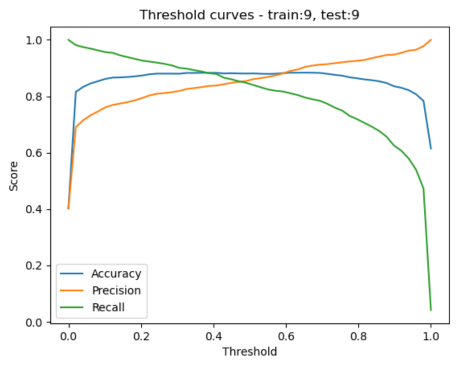
Figure 1: Train and Test loss for each epoch.

Indeed, the test loss is not reduced through training consistently, it rather fluctuates. However, all the relevant metrics (precision, accuracy, PR-AUC etc.) do increase on the test set through training. I am not sure what to make of this behaviour. One would imagine that an overfitted model will get increasingly better on the training set and increasingly worse on the test one, not just fluctuating.

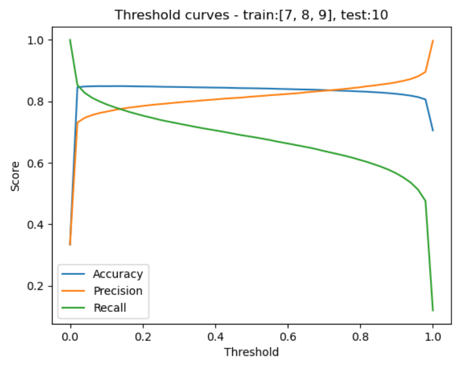
Additionally we present the behaviour of accuracy, precision and recall for different thresholds and for different experiments below.



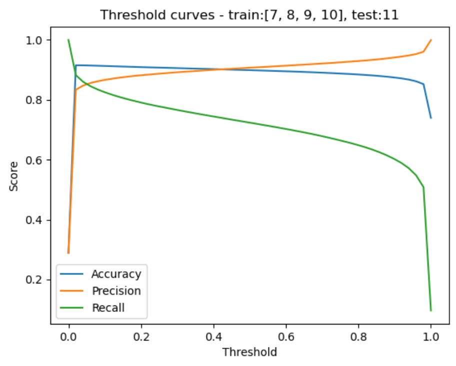
(a) $8 \to 8$



(b) $9 \to 9$



(c) $7, 8, 9 \to 10$



(d) $7, 8, 9, 10 \to 11$

Figure 2: Accuracy, Precision and Recall as functions of the threshold for the GIN experiments.

where, the accuracy is defined by

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{P} + \text{N}}. \tag{3}$$

As expected, Accuracy is not affected at all while Recall and Precision swap roles.

2.2 PlanE

Additionally we studied the performance of the model described in `https://arxiv.org/abs/2307.01180` on our dataset. This is representation learning model on planar graphs. It takes as input the graph structure and possible node or edge features. Based on these, it determines a representation through certain decompositions on the planar graph (BLOCKCUT, SPQR). The authros claim that any pair of graphs with V_1, V_2 number of vertices can be distinguished by a variant of their model with at most $\log_2(\max\{|V_1|, |V_2|\}) + 1$ layers.

Training for 50 epochs on a 80/10/10 split for training test and validation using 4 layers of 64 dimensions produced [2]:

Table 2: Experimental Results PlanE

ROC AUC (%)	$7 \to 7$	$8 \to 8$	$9 \to 9$
train	96.0	94.0	99.0
test	87.3	78.8	85.5
valid	87.3	79.4	85.3

It turns out that for our dataset PlanE does not perform significantly better at least for single loop cases. Given the much higher training time required for this model compared to our GIN we did not test on further cases.

[2] These are lower loop cases where PR AUC and ROC AUC are not expected to differ much.