# Graph Learning on Financial Tabular Data

## Cascade and Interleaved architectures using GNNs and Transformers

**Enachioiu Sorin-Catalin**[1]

**Supervisor(s): Dr. Kubilay Atasu**[1]**, Halil Cagri Bilgi**[1]

[1]**EEMCS, Delft University of Technology, The Netherlands**

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 22, 2025

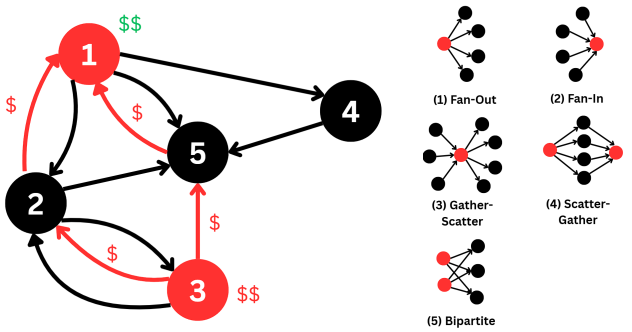
Name of the student: Enachioiu Sorin-Catalin
Final project course: CSE3000 Research Project
Thesis committee: Dr. Kubilay Atasu, Halil Cagri Bilgi, Dr. Thomas Hollt

## Abstract

Detecting money-laundering activity in financial transactions is challenging due to the multigraph nature of the problem as well as the intricate fraud patterns that exist. In this work we introduce two architectures, Cascade and Interleaved. These architectures combine the expressive power of local message passing (MP) from Graph Neural Networks (GNNs) with the one of global message passing from Transformers. Both models leverage the Principal Neighborhood Aggregation (PNA) GNN for capturing rich local structure. We also incorporate the MEGA two-stage aggregation scheme to distinguish transactions that have the same source and destination accounts from other transactions. We further enhance our architectures with PEARL, a learnable positional encoding framework that has a reduced overhead compared to other techniques. We evaluate our models on the IBM transactions for Anti-Money Laundering (AML) synthetic datasets. We achieve significant improvements compared to the PNA baseline, and come close to tie SOTA results, while requiring less feature engineering on the input graphs and also show that the application of learnable positional encodings in financial fraud detection tasks is promising.


| Trans ID | Timestamp | Src Bank | Src Acct | Dst Bank | Dst Acct | Amount | Currency | Payment Type |
|---|---|---|---|---|---|---|---|---|
| 0 | 9 AUG 2023 12:45 | A | 1 | A | 2 | 1 000 | EUR | Cheque |
| 1 | 14 AUG 2023 15:30 | A | 2 | B | 3 | 1 710 | USD | ACH |
| 2 | 21 AUG 2023 06:30 | B | 4 | B | 5 | 5 000 | RON | Credit Card |
| 3 | 1 SEP 2023 11:32 | C | 6 | A | 2 | 3 200 | USD | Wire |
| 4 | 10 SEP 2023 17:45 | B | 3 | B | 4 | 1 250 | EUR | Credit Card |
| 5 | 25 SEP 2023 19:45 | A | 2 | A | 1 | 2 000 | EUR | Cheque |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |

(a) Transactions in Tabular Format




(b) Transactions in Graph Format (c) Fraud Patterns

Figure 1: (a) Tabular representation of financial transactions, the ones highlighted in pink are illicit. (b) Graph representation of financial transactions in which nodes represent accounts, and edges represent transactions. There can be multiple transactions between two accounts, making the transaction graph a multigraph. Red nodes represent fraudsters, black nodes represent innocent entities. Red edges are illicit transactions, one account can make both illicit and non-illicit transactions depicted with black. (c) Shows some examples of possible fraud patterns.

## 1 Introduction

Recent advances in machine learning such as using Transformers on Graphs [1] bring many new possibilities to explore. One of them being how they can be combined with GNNs[2]. This paper tackles the problem of detecting money laundering based on financial transactions. Although these records are stored in a relational database, as shown in Figure 1a, they can be represented as a graph [3], as shown in Figure 1b, making it suitable for a large range of algorithms. Furthermore, the ever-evolving nature of fraud schemes makes it hard for traditional algorithms to detect and adapt to them, thus, recent work focuses more on Graph Learning approaches.

One important part of previous work was to establish a viable way of generating synthetic data to train models for fraud detection. One synthetic data generator is described by Altman, E. et al. [4], and it is capable to incorporate intricate money laundering patterns (see Figure 1c), that are also present in real world financial data. This technique was used to create the IBM Transactions for Anti Money Laundering (AML) datasets [5] used in our research. Furthermore, models using GNNs [6] [7], more specifically Message Passing Neural Networks (MPNNs) [8] with specific enhancements, and also models using Transformers [9] have been implemented to solve this task and have given promising results.

Even though some models that detect money laundering use standalone GNNs and Transformers, there are still a lot of unexplored architectures that combine them. This is exactly the contribution that this paper brings. More specifically, we will look into how a Transformer able to perform Global Message Passing (MP) on Nodes can be combined with a MPNN that performs local message passing and compare their performance, F1 score, to a baseline obtained from running a PNA [10], which is a type of GNN, on the same task.

The main **research question** this paper addresses is: *How well do Cascade and Interleaved models, that use Transformers for Global message passing on Nodes, perform on the IBM Anti Money Laundering (AML) datasets compared to a PNA baseline ?*

In summary, the main contributions of this paper are:

- **Models.** We introduce two models that integrate Global Message passing on nodes, Cascade and Interleaved. These models show that the performance of PNA on financial fraud detection can be enhanced by combining it with a Transformer.
- **Application of SOTA Multigraph Techniques.** We integrate the MEGA [6] framework in our models, enhancing the way information is propagated in message passing for multigraphs. We show that the integration of MEGA in our models can enhance their performance.
- **Applying Learnable Positional Encodings for Fraud Detection** Rather than just computing positional encodings with a meaningful overhead, we study the usage of PEARL [11], a novel and efficient way to make learnable positional encodings for graphs. We show that the integration of PEARL can enhance model performance.

## 2 Related Work

In this section, we present the most important previous work related to the AML datasets. We then introduce PEARL, the type of positional encodings that we use within this research.

### 2.1 GNN improvements

Since a transaction graph is a multigraph, as we can have multiple edges between two nodes, there was a need to augment normal methods using Graph Neural Networks to be better suited for this kind of task. Thus, in the paper by Egressy et al. Multi-GNN [7], transaction graphs are equipped with egoIDs, port numbering, and reverse message passing. Out of these three additions, port numbering is the most relevant to our work, as it enables distinguishing messages coming from the same neighbor from the rest in message passing rounds.

In Mega GNN[6], the idea of port numbering was refined, and, instead of trying to add extra features to the edges of the graph, to distinguish messages coming from the same neighbor, the aggregation step of MPNNs was modified in order to take two stages, first aggregate messages from the same neighbor together, and then in the second stage combine these aggregations coming from each neighbor. In this work we only use the variant of MEGA with unidirectional MP which is simpler to implement but not as powerful as the one that integrates both reverse MP and egoIDs.

### 2.2 Transformer-based models

To address the limitations that GNNs face when learning intricate patterns, a new model called FraudGT [9] was introduced, using a graph transformer on nodes, with extra additions such as edge-based message passing, and edge-based attention bias. This model also uses the three enhancements introduced in Multi-GNN and has shown that it is worth considering Transformers for fraud detection tasks.

### 2.3 Positional Encodings - PEARL

Positional encodings are usually computed statically based on various properties of a graph. Kanatsoulis et al. [11] introduce a novel technique in which positional encodings are approximated and also learnable, thanks to the use of a GNN, the use of pooling functions, and a filter that uses the Laplacian of the graph. This technique also comes with a small overhead compared to others. The PEARL framework can be integrated into existing GNN or Transformer architectures as a standalone module, producing positional encodings that are directly combined with the initial input or passed on to downstream layers of the model.

## 3 Background

In this section, we introduce the most important building blocks necessary to understand our research. For a better understanding of the notation used within this study, we have included the most important ones in Table 1.

### 3.1 Multigraphs and Financial Transactions

Let $G$ be a directed transaction multigraph, $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{H}, \mathbf{E})$. Then $\mathcal{V}$ is the node set representing accounts, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the edge set, which can consist of multiple parallel edges, corresponding to transactions between accounts. Node attributes are represented by $\mathbf{H} \in \mathbb{R}^{n \times d_n}$, whilst edge attributes are represented by $\mathbf{E} \in \mathbb{R}^{n \times n \times d_e}$.

Table 1: Notation utilised in this study.

| Notation | Description |
|---|---|
| $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ | A graph $\mathcal{G}$ contains node set $\mathcal{V}$ and edge set $\mathcal{E}$. |
| $n \in \mathbb{R}$ | Number of nodes in the graph. |
| $d_n, d_e$ | Dimensions of node and edge attributes |
| $\mathbf{H} \in \mathbb{R}^{n \times d_n}$ | Node features in the graph. |
| $\widehat{\mathbf{H}} \in \mathbb{R}^{n \times d_n}$ | Updated node representations of the graph. |
| $\mathbf{E} \in \mathbb{R}^{n \times n \times d_e}$ | Edge features in the graph. |
| $m_{uv}$ | Message from node $u$ to node $v$. |
| $e_{uv}$ | Edge feature between nodes $u$ and $v$. |
| $h_u$ | Node feature of node $u$. |
| $\mathcal{N}(v)$ | Neighborhood of node $v$. |
| $\mathbf{W}$ | Learnable weight matrix . |
| $\mathbf{Q},\ \mathbf{K},\ \mathbf{V}$ | Query, key, and value matrices in the attention mechanism. |
| $\|\|$ | Concatenation |

### 3.2 Message Passing Neural Networks (MPNNs)

Since graphs are a natural way of representing many problems, Machine Learning algorithms able to work on them were a must. This is what lead to the development of MPNNs [8], which are one of the most common family of GNNs. They are based on a message-passing mechanism, which combines the information of a node with that of its neighbors. More precisely, in a MPNN, one layer in which node features are updated consists of the following three steps, also shown in Figure 2: (1) each node computes a message $m_{uv}$ with its current state and sends it to all its neighbors, (2) each node aggregates all the messages it received in an embedding $a_u$, and (3) each node updates its state based on $h_u$ and $a_u$

$$\text{Message:} \quad m_{uv}^{(l)} = f^{(l-1)}\big(h_u^{(l-1)}, e_{uv}^{(l-1)}\big), \tag{1}$$

$$\text{Aggregate:} \quad a_u^{(l)} = AGG\big\{\{\, m_{uv}^{(l)} : v \in \mathcal{N}(u)\}\big\}, \tag{2}$$

$$\text{Update:} \quad h_u^{(l)} = g_u^{(l-1)}\big(h_u^{(l-1)}, a_u^{(l)}\big). \tag{3}$$

Where $f^{(l-1)}$ is a function that computes the message for a node, AGG is usually a permutation invariant operation, and $g_u^{(l-1)}$ is an update function for the embedding of a vertex.

Furthermore, edge features can also be updated, by making use of the updated embeddings of the source and destination node features as follows:

$$e_{uv}^{(l)} = e_{uv}^{(l-1)} + g_e^{(l-1)}\Big(h_u^{(l)}, h_v^{(l)}, e_{uv}^{(l-1)}\Big) \tag{4}$$

Where $g_e^{(l-1)}$ is an update function for the embedding of an edge.

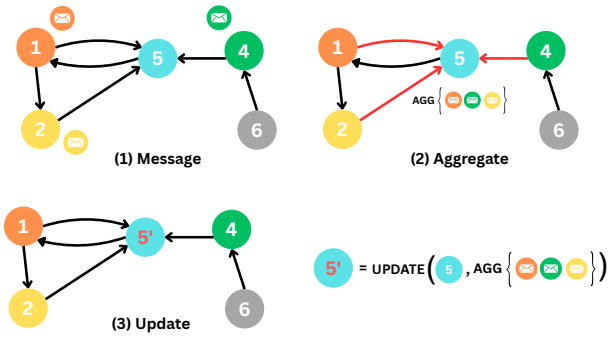



Figure 2: Illustration of the MPNN three step update for a node embedding.

Even though MPNNs are a powerful set of tools that can be used when working with graphs, they still have some limitations, such as over-smoothing and over-squashing [12] [13].

### 3.3 Principal Neighborhood Aggregation (PNA)

PNA [10] is a specific type of MPNN that encompasses multiple aggregation functions at the same time in the Aggregate step presented above, thus being able to capture richer local contexts. It also introduces scalers, which are functions based on the number of aggregated messages within a node. The scalers are multiplied with the aggregated value and can be: an attenuation, an amplification or an identity.

A PNA equipped with an edge update (EU) function is called PNA + EU, since edge updates are essential for our task, we'll simply refer to PNA + EU as PNA.

### 3.4 Transformer Encoder

The Transformer model was introduced in the Attention is All You Need paper [14]. It comprises two main parts, a Transformer Encoder and a Transformer Decoder. For our graph tasks, we are only interested in the Transformer Encoder, which we use to update nodes based on similarities between them. This can also be thought about as a round of global message passing, as each node attends and updates its information based on the values of all the nodes in the graph, including itself, treating the graph as fully connected and capturing long-range dependencies.

The Transformer Encoder consists of a stack of L encoder layers. Each encoder layer consists of two parts: a multi-head self-attention (MHA) module, and fully connected feed-forward networks (FFN). Both parts also employ skip connections and layer normalization.

The Transformer Encoder works as follows: (1) projects the input into the Query, Key and Value spaces, (2) computes dot-product attention, (3) concatenates multiple attention heads and project back to the input dimension, (4) passes the previous result to a Position-wise feed forward network (FFN).

Given the input node representations $\mathbf{H} \in \mathbb{R}^{n \times d_n}$, we compute

$$\begin{aligned} Q^{(l)} &= \mathbf{H}^{(l-1)} W^Q, \\ K^{(l)} &= \mathbf{H}^{(l-1)} W^K, \\ V^{(l)} &= \mathbf{H}^{(l-1)} W^V. \end{aligned} \tag{5}$$

where $W^Q, W^K, W^V \in \mathbb{R}^{d_n \times d_h}$ are learnable weight matrices, and $d_h$ is their feature dimension. The scaled dot-product attention is then

$$\mathrm{Attn}(Q, K, V) = softmax\Big(\frac{QK^\top}{\sqrt{d_h}}\Big) V \tag{6}$$

To allow the model to capture different types of relationships in parallel and have better stability, $H$ attention heads are used. We denote with $h$ the index of different attention heads. The $H$ head outputs are concatenated and projected back to the input dimension:

$$\mathrm{MHA}\big(H^{(l-1)}\big) = \Big\|_{h=1}^{H} \mathrm{Attn}\big(Q^{(h,l)}, K^{(h,l)}, V^{(h,l)}\big) W_O^{(l)} \tag{7}$$

where $W_O^{(l)} \in \mathbb{R}^{d_n \times d_n}$ is a learnable weight matrix.

Thus, by adding skip connections and normalization, one Transformer Encoder layer updates initial features as follows:

$$\widehat{H}^{(l)} = H^{(l-1)} + \mathrm{MHA}(H^{(l-1)}), \tag{8}$$

$$H^{(l)} = \widehat{\mathrm{H}}^{(\mathrm{l})} + \mathrm{FFN}(\widehat{\mathrm{H}}^{(\mathrm{l})}). \tag{9}$$

### 3.5 Positional Encodings (PEs)

PEs are an important component for a transformer, as they bind a notion of position to inputs, which is helpful when looking for similarities between them, as the position might also affect the impact one input has on others. For graphs, there are multiple types of possible positional encodings, a comparison between the power of various positional encodings can be found in [15]. Some of the most common PEs used for graphs are Laplacian positional encodings [16] and Random walk positional encodings [17].

## 4 Proposed Method

In this section, we begin with a short motivation for our method and then introduce our models, Cascade and Interleaved, which address the limitations that usual MPNNs face when learning intricate fraud patterns. We then explain the two enhancements we brought to them, MEGA and PEARL.

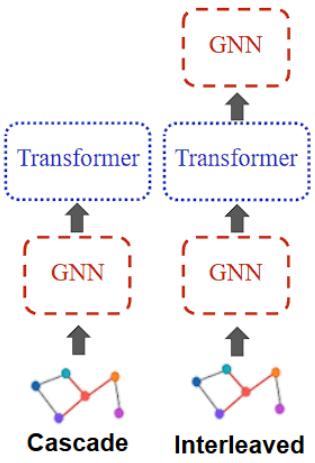



Figure 3: Model Architectures

### 4.1 Motivation

Effective fraud detection can benefit from considering both local interactions and global ones because of the complexity of fraud patterns. In our case we are investigating the use of global node similarities by the means of a Transformer, because it can capture long-range dependencies that a purely local GNN aggregation would miss due to its limitations. However, the AML graphs do not have any meaningful node features, only edge ones. Thus we need a way to encode local context, based on just transaction/edge information. To this end, we apply a GNN to learn meaningful node embeddings that encapsulate local neighborhood information and only use the transaction information. Then use these node embeddings within a Transformer Encoder on Nodes to capture long-range dependencies and global similarities.

Therefore, by considering these two architectures, Cascade and Interleaved, shown in Figure 3, we aim to assess how the performance of a standalone MPNN, in our case PNA, can be improved by combining it with a Transformer.

### 4.2 Cascade model

Our first model architecture, Cascade, shown in Figure 3, does exactly the two step pipeline, combining local and global information, described above. The Cascade model was first introduced for capturing long-range dependencies in graph representations by Zhanghao Wu et al. [18]. To the best of our knowledge, prior work on the AML datasets do not include any similar models to Cascade.

The Cascade model works by taking the initial graph as input, which does not have any node embeddings. Then, it first passes the graph through a PNA to embed local structure and also use the information existent in edges for creating meaningful node embeddings. Next, we pass these embeddings through a Transformer Encoder that is able to look for global similarities and update the embeddings accordingly. In the end these final embeddings are used to predict which transactions are illicit. Thus, with the Cascade model we are able to combine the benefits of both local and global information.

### 4.3 Interleaved model

Because in the Cascade model node embeddings gain a lot of global information after passing them through the Transformer Encoder, we consider that it would be meaningful to propagate locally this new information with one additional MPNN at the end of the Cascade model. This inspired the design of the Interleaved architecture, shown in Figure 3.

The Interleaved model as we introduce it does not seem to have been presented before in literature, yet a similar notion to it, in which local operators represented by Graph Convolutional Layers and global operators represented by Transformer layers are interleaved one after another was introduced by Shuo Yin et al. in [19].

### 4.4 Enhancements

#### 4.4.1 Integration of SOTA multigraph Techniques

In MEGA-GNN the results of PNA were considerably improved by the addition of the two step aggregation. Thus, we also integrate it within our PNA, and then use this MEGA-PNA as a backbone to both Cascade and Interleaved models to see if they can improve over MEGA-PNA as well.

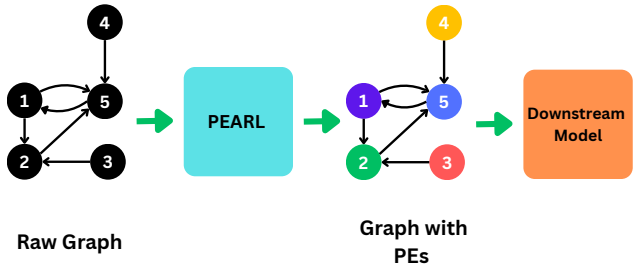



Figure 4: Diagram showing how we integrated PEARL into our work. On the left there is the Raw Graph, without any actual node features, depicted as black nodes in the entire graph. Then, we use PEARL on this graph to generate PEs for the nodes, which we then concatenate to the node features, denoted by colors in the graph on the right. This new Graph with PEs can then be passed to any Downstream model.

#### 4.4.2 Applying Learnable Positional Encodings for Fraud Detection

Because we are working with Transformers it is natural to think about implementing positional encodings. There are two main reasons why we considered using PEARL over other methods: *scalability*, due to reduced overhead in both memory and space, which is really helpful when working with the AML datasets, as they encompass large graphs, with a high amount of nodes, and *expressive power*. We use the PEARL model as a pre-processing step that works separately from our models, it takes the input graph and then concatenates to the node features the PEs. Then this new graph with PEs is passed to any downstream model, as shown in Figure 4. Thus, making the use of PEARL easy with any model. We only use a really simple PEARL model, that adds only

| Models | Small-HI | Small-LI | #params |
|---|---|---|---|
| PNA | 63.48 ± 3.56 | 21.67 ± 1.96 | 32,197 |
| + PEARL | 65.57 ± 3.97 | 25.35 ± 2.88 | 33,547 |
| + MEGA | 72.58 ± 1.35 | 43.71 ± 1.14 | 41,837 |
| + MEGA + PEARL | 74.45 ± 0.89 | 45.00 ± 1.02 | 43,187 |
| Cascade | 64.99 ± 2.31 | 25.42 ± 0.37 | 37,257 |
| + PEARL | 61.84 ± 5.55 | 27.81 ± 3.42 | 38,607 |
| + MEGA | 73.25 ± 0.66 | 45.50 ± 0.98 | 46,897 |
| + MEGA + PEARL | 73.64 ± 1.82 | 46.07 ± 0.93 | 48,247 |
| Interleaved | 68.40 ± 2.86 | 31.30 ± 3.17 | 64,937 |
| + PEARL | 67.63 ± 1.15 | 34.86 ± 2.17 | 66,287 |
| + MEGA | 75.31 ± 0.45 | 46.05 ± 0.88 | 84,217 |
| + MEGA + PEARL | 73.28 ± 2.18 | 44.15 ± 0.89 | 85,567 |

Table 2: Benchmark results of F1 scores (%) on the AML edge classification task, as well as the number of parameters for each model. We highlight the first and second best results

1350 parameters for any of the models, thus, it is a really lightweight and powerful enhancement.

## 5 Experimental Setup and Results

In this section, we describe the datasets used, implementation, baselines, and evaluation metrics, and then present our experimental results.

### 5.1 Experimental Setup

*Datasets.* We use the synthetic, publicly-available IBM Transactions for Anti Money Laundering (AML) [5] datasets. Real transaction data is hard to obtain because it is subject to privacy regulations and fragmented across multiple financial institutions, making it hard for researchers to obtain a good view of real laundering networks.

Table 3: Detailed view of the statistics for the Small AML datasets.

| Dataset | # nodes | # edges | Illicit Ratio |
|---|---|---|---|
| AML Small HI | 0.5 M | 5 M | 0.07 % |
| AML Small LI | 0.7 M | 7 M | 0.05 % |

The AML datasets overcome these issues by simulating interbank transaction graphs with ground-truth labels, and are offered in three size variants: Small, Medium, and Large. Each of the three variants has two illicit-transaction settings, based on the rate of laundering transactions: **Low Illicit (LI)**, consisting of a highly imbalanced scenario, with rare money-laundering events and **High Illicit (HI)**, which considers a scenario in which laundering transactions make up a larger fraction of the total. Table 3 provides detailed statistics for the Small AML datasets. We only use the High Illicit and Low Illicit AML datasets of the Small size within this study.

*Implementation.* The models presented in this work are implemented using PyTorch Geometric [20]. We use the same 60-20-20 temporal train-validation-test split as in Multi-GNN for the datasets. To ensure statistical significance, we present the mean ± standard deviation for each experiment, computed over five runs, and initialized with random seeds. Models are trained for 80 epochs using the AdamW optimizer with a cosine scheduler, and a weighted cross-entropy loss function to counter class imbalance.

*Baselines.* To evaluate whether the Cascade and Interleaved models enhance the expressive power of a PNA, as well as understand what benefits each PEARL and MEGA can bring, we use four different configurations for the PNA as baselines: PNA, PNA + PEARL, PNA + MEGA, PNA + PEARL + MEGA. We use these PNA variants as backbones for the Cascade and Interleaved models. Thus, we will first compare what happens to the PNA with each addition, and then compare what improvements we get to one PNA variant when we integrate it within our models.

*Evaluation.* Because the AML datasets are highly imbalanced, we use the F1 score to assess performance, consistent with previous works [7], [6] [9]. Given true positives (TP), false positives (FP) and false negatives (FN) then, the F1 score can be computed as:

$$\begin{aligned} \text{Precision} &= \frac{\text{TP}}{\text{TP} + \text{FP}} \\ \text{Recall} &= \frac{\text{TP}}{\text{TP} + \text{FN}} \\ F_1 \text{ score} &= 2 \cdot \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \end{aligned} \tag{10}$$

In our experiments, we report the F1 test score corresponding to the highest F1 score in validation.

*Hardware.* We used NVIDIA A40 GPUs with 64 GB of memory to perform training and inference.

### 5.2 Results

We now describe the experimental results that we obtained, which are displayed in Table 2. We will begin by present-

ing the results achieved by applying MEGA and PEARL enhancements on PNA. Then, continue by presenting the results achieved by Cascade and Interleaved for each PNA variant.

#### 5.2.1 PNA Variants

Table 2 displays at the top the four PNA variants on both the Small-HI and Small-LI datasets. The addition of PEARL to the vanilla PNA yields a modest increase, about 2% on HI and 4% on LI, showing that positional encodings can boost the performance of just a MPNN. The PNA + MEGA Variant sees a significant performance improvement across both datasets. Lastly, the addition of both MEGA and PEARL achieves the best performance, 74.45% on HI and 45.00% on LI, showing that both enhancements bring benefits that are complementary.

#### 5.2.2 Cascade and Interleaved with each PNA Variant

**PNA** When using vanilla PNA as the backbone of both Cascade and Interleaved we observe that the Cascade model is able to improve just on the Small-LI dataset. Furthermore, the Interleaved model is able to improve over the vanilla PNA with 5% on the Small-HI and a 10% increase on the Small-LI, meaning that using the final MPNN for refining local information, in the Interleaved model, is beneficial.

**PNA + PEARL** The addition of PEARL positional encodings leads to a decrease for the Cascade model in comparison to PNA + PEARL on Small-HI and a slight increase on Small-LI. Whilst for the Interleaved, there is a 9% increase on the Small-LI and a small increase on the Small-HI. Both the Cascade and Interleaved models have a decrease in performance on the Small-HI dataset when compared to their implementation with just vanilla PNA.

**PNA + MEGA** For the PNA + MEGA we can see steady improvements for both the Cascade and Interleaved models. The Cascade model is able to achieve a 2% increase on the Small-LI, and a small increase on Small-HI in comparison to the PNA + MEGA. The Interleaved + MEGA model achieves the best result for Small-HI, 75.31%, and the second best for Small-LI, 46.05%.

**PNA + MEGA + PEARL** For the PNA + MEGA + PEARL the Cascade model sees a slight decrease for the Small-HI dataset, but achieves the best result for the Small-LI one, 46.07%. In the meantime, the Interleaved + MEGA + PEARL model has a performance decrease on both datasets, compared to the results of both Cascade and PNA with the same enhancements, but also compared to Interleaved + MEGA as well.

## 6 Discussion

In this section, we present both the advantages and limitations of the Proposed Method.

### 6.1 Advantages of the Proposed Method

Our experimental results presented in the previous section demonstrate several clear benefits of both the proposed models and of the enhancements.

**PEARL adds expressive power at low cost.** The PEARL model that we use as a pre-processing step is only adding 1350 parameters and a small time overhead, yet, it is able to improve consistently the result for most of the models that use it.

**PNA (Unidirectional MP) + MEGA + PEARL achieves significant results.** As stated previously, in this research we use the PNA + MEGA with Unidirectional MP, meaning that we do not use EgoIDs and Reverse MP. Even though we do not make use of this additions, when using PEARL we get results similar to those reported in the MEGA paper for the model that does include them. Thus, showing that PEARL is capable to add, with small overhead, expressive power similar to that added by other, more complex solutions.

**Constant improvements with each enhancement for PNA and Cascade.** The PNA and Cascade models really benefit from each addition: PEARL, MEGA and MEGA + PEARL, in exactly this order. Their performance increases with almost each addition, except for Cascade + PEARL on Small-HI. Cascade is able to improve slightly on almost all the PNA results.

**Interleaved model improves performance of PNA significantly.** The Interleaved model is consistently performing better than the PNA and Cascade for each of the variants, except for MEGA + PEARL. Furthermore, the Interleaved + MEGA model achieves the highest F1 on Small-HI and the second best on Small-LI, thus making it one of the best models.

### 6.2 Limitations of the Proposed Method

**Unstable Cascade + PEARL on Small-HI.** The Cascade + PEARL model exhibited a very high standard deviation, 5.5%, for the SMALL-HI dataset driven by a run that scored only 50.89% F1 score. This suggest that our training method might be unstable, or that the addition of PEARL might cause instability.

**Overfitting in the Interleaved + MEGA + PEARL.** The Interleaved + MEGA + PEARL model results indicate that it might be suffering from overfiting, as it continues to improve on the training set, yet its validation and test performance peak early and then gradually decrease. This issue might be solvable by better tuning of hyperparameters.

## 7 Responsible Research

### 7.1 Reproducibility

To ensure reproducibility of the experiments, we open-source the codebase at https://github.com/hcagri/aml-sorin. In the repository we also included the configurations that achieved the best result for each model, for each dataset.

### 7.2 Ethics Statement

The financial datasets used in this research are publicly available and artificially generated without any personally identifiable information. Thus, they do not pose any privacy risks or discrimination concerns. Finally, by contributing methods combining GNNs and Transformers, and by integrating novel

techniques for positional encodings for this task, this work supports the global fight against financial fraud.

# 8 Conclusions and Future Work

## 8.1 Conclusions

In this work, we have shown that combining local message passing with global message passing on nodes can achieve improved results on the Small AML datasets compared to just a PNA. We began by showing that the Cascade and Interleaved models can improve a vanilla PNA. We then added MEGA to the PNA model that is the backbone for both models, a SOTA Multigraph Technique, which has improved the results for all the models.

There are three significant results that we obtained. First, enhancing PNA with both PEARL and MEGA has gained a 11% increase over vanilla PNA on Small-HI and a 24% on Small-LI. Second, the Cascade model has achieved the best results with the PEARL and MEGA enhancements as well, and has the best F1 score for Small-LI, 46.07%. Lastly, the Interleaved + MEGA model has achieved the best result for Small-HI, 75.31%, and the second best on Small-LI, 46.05%. This shows both that the enhancements we made, PEARL and MEGA, were effective and also that the Cascade and Interleaved models were able to improve over the PNA baselines.

Another interesting finding was that including learnable positional encodings, by using PEARL into our models increases their performance, and also adds a really small overhead to them, thus showing this method is worth considering for future work in fraud detection.

## 8.2 Limitations and Future Work

Despite the strong performance that the Cascade and Interleaved architectures achieved, our work was constrained by limited hyperparameter tuning due to time constraints.

Thus, future work could investigate what impact various sampling schemes could have on the performance of these models. Another interesting research direction would be to use the FraudGT transformer, which uses both node and edge features instead of just a Transformer Encoder on nodes in the Cascade and Interleaved models. Furthermore, as PEARL shows promise in improving the results of the models, at a really small cost, we consider that it would be worth experimenting more with its hyperparameters as well as trying to integrate it with other previous works, such as Multi-GNN, and FraudGT to see what results would it give.

# 9 Acknowledgement

Research reported in this work was partially or completely facilitated by computational resources and support of the Delft AI Cluster (DAIC) at TU Delft (RRID: SCR_025091), but remains the sole responsibility of the authors, not the DAIC team.

# References

[1] Ahsan Shehzad, Feng Xia, Shagufta Abid, Ciyuan Peng, Shuo Yu, Dongyu Zhang, and Karin Verspoor. Graph transformers: A survey, 2024.

[2] Bowen Jin, Gang Liu, Chi Han, Meng Jiang, Heng Ji, and Jiawei Han. Large language models on graphs: A comprehensive survey. *IEEE Transactions on Knowledge and Data Engineering*, 36(12):8622–8642, 2024.

[3] Matthias Fey, Weihua Hu, Kexin Huang, Jan Eric Lenssen, Rishabh Ranjan, Joshua Robinson, Rex Ying, Jiaxuan You, and Jure Leskovec. Position: Relational deep learning - graph representation learning on relational databases. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 13592–13607. PMLR, 21–27 Jul 2024.

[4] E. Altman, J. Blanuša, L. von Niederhäusern, B. Egressy, A. Anghel, and K. Atasu. Realistic synthetic financial transactions for anti-money laundering models. arXiv preprint arXiv:2306.16424, 2024. doi: 10.48550/arXiv.2306.16424.

[5] IBM Research. Ibm transactions for anti money laundering (aml). https://www.kaggle.com/datasets/ealtman2019/ibm-transactions-for-anti-money-laundering-aml, 2023.

[6] H. Çağrı Bilgi, Lydia Y. Chen, and Kubilay Atasu. Multigraph message passing with bi-directional multi-edge aggregations, 2024.

[7] Beni Egressy, Luc von Niederhäusern, Jovan Blanuša, Erik Altman, Roger Wattenhofer, and Kubilay Atasu. Provably powerful graph neural networks for directed multigraphs. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, volume 38 of *AAAI'24/IAAI'24/EAAI'24*, pages 11838–11846. AAAI Press, February 2024.

[8] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry, 2017.

[9] Junhong Lin, Xiaojie Guo, Yada Zhu, Samuel Mitchell, Erik Altman, and Julian Shun. Fraudgt: A simple, effective, and efficient graph transformer for financial fraud detection. In *Proceedings of the 5th ACM International Conference on AI in Finance*, ICAIF '24, page 292–300, New York, NY, USA, 2024. Association for Computing Machinery.

[10] Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. Principal neighbourhood aggregation for graph nets, 2020.

[11] Charilaos I. Kanatsoulis, Evelyn Choi, Stephanie Jegelka, Jure Leskovec, and Alejandro Ribeiro. Learning efficient positional encodings with graph neural networks, 2025.

[12] Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications, 2021.

[13] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning, 2018.

[14] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[15] Mitchell Black, Zhengchao Wan, Gal Mishne, Amir Nayyeri, and Yusu Wang. Comparing graph transformers via positional encodings, 2024.

[16] Vijay Prakash Dwivedi, Chaitanya K. Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks, 2022.

[17] Vijay Prakash Dwivedi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Graph neural networks with learnable structural and positional representations, 2022.

[18] Zhanghao Wu, Paras Jain, Matthew A. Wright, Azalia Mirhoseini, Joseph E. Gonzalez, and Ion Stoica. Representing long-range context for graph neural networks with global attention, 2022.

[19] Shuo Yin and Guoqiang Zhong. Lgi-gt: Graph transformers with local and global operators interleaving. In Edith Elkind, editor, *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence (IJCAI-23)*, pages 4504–4512. International Joint Conferences on Artificial Intelligence Organization, 8 2023. Main Track.

[20] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric, 2019.

# A Statement on the use of Large Language Models

In accordance with the *Course Policy on the Use of Large Language Models (LLMs) / ChatGPT*, LLMs were used for improving the clarity of paragraphs or fixing grammatical errors. The prompts used had the following patterns: "Is this sentence (...) clear ?", "Are there any grammatical errors in the following paragraph: (...)".