Graph Learning on Tabular Data: Think Global And Local

Full Fusion and Interleaved architectures on IBM's Anti-Money Laundering Data

Andrei Ștefan[1]

Supervisor(s): Dr. Kubilay Atasu[1]**, Çağrı 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: Andrei Ștefan
Final project course: CSE3000 Research Project
Thesis committee: Dr. Kubilay Atasu, Çağrı Bilgi, Dr. Thomas Höllt

Abstract

As financial fraud becomes increasingly sophisticated, traditional detection methods struggle to uncover the complex relational patterns underlying illicit behavior. This paper investigates the effectiveness of combining Graph Neural Networks (GNNs) and Transformers for fraud detection on relational data transformed into graph structures. Focusing on the IBM Anti-Money Laundering (AML) dataset, two hybrid architectures are proposed: **Interleaved**, which alternates between GNNs and Transformers to exploit local and global information sequentially, and **Full-Fusion**, which fuses parallel GNN and Transformer representations at both feature and decision levels. The results show that integrating Transformers significantly boosts performance over standalone GNN baselines, with improvements up to 10% in the F1 score in small-scale datasets. It is also demonstrated that gating-based fusion strategies enhance model stability and accuracy, and further, that PEARL-based positional encodings do not result in any conclusive improvement of the models. These findings highlight the value of combining local message passing and global attention mechanisms for structured financial anomaly detection, and pave the way for more robust, adaptable graph-based solutions in fraud analytics and more.

1 Introduction

Machine Learning and Artificial Intelligence are evolving at a rapid pace, and with them comes the growing need for models that can not only perform well but also adapt to increasingly complex problems. This project explores how machine learning models, specifically Graph Neural Networks (GNNs) and Transformers, can be applied on relational data that has been transformed into graph structures.

The focus is on the financial sector, where fraud detection is becoming more critical than ever due to the increasing sophistication of fraudulent schemes. Traditional detection methods often fall short when it comes to uncovering hidden relationships in data, which is where graph-based approaches come in. Prior work, such as that by Akoglu et al. [1], has shown how effective graph-based anomaly detection can be in capturing subtle, structured patterns of fraud. More recently, Egressy et al. [2] achieved strong results using graph models, reinforcing the promise of this direction. That said, there is still a noticeable gap when it comes to broad, hands-on experimentation with different model architectures. This project aims to address that by testing a variety of graph-based learning techniques on sample banking fraud scenarios and evaluating their effectiveness.

The research question can be formally expressed as follows: *"How well does a Full Fusion or Interleaved architecture using Graph Transformers and GNNs on the edges of sampled subgraphs perform on the IBM Transactions for Anti Money Laundering (AML) [3] dataset, compared to existing literature?"* The focus is on the capacity of Graph Transformers to capture edge embeddings, and combining that with the node embeddings of the GNNs for a more accurate model.

Thus, the main contributions of this paper are threefold: **(1)** A complete implementation and detailed examination of the two proposed architectures (Full Fusion and Interleaved specifically) are presented. **(2)** An ablation study demonstrates that each architecture outperforms its individual components when evaluated in a vacuum, using models with equivalent parameter counts. **(3)** The relevance and utility of PEARL-based positional encodings are analyzed within both architectures and datasets.

2 Background and Related Work

In this section, a comprehensive overview of all concepts used to construct the architectures is given, from basic concepts to state-of-the-art models that inspired and contribute to the architecture's performance. This section also makes use of mathematical formulas, for which all relevant notation is mentioned and (briefly) explained in section *A.1* of the Appendix.

2.1 Background/Preliminaries

Graphs

Graphs and their representations stay at the cornerstone of all machine learning techniques to come. The financial network represented by the AML data can be structured as the multigraph $G = (V, \mathcal{E}, \mathbf{X}, \mathbf{E})$, a multigraph referring to a graph with the possibility of more than one (directed) edge connecting a node to another. In this case, V is a set of n nodes representing accounts and $\mathcal{E} \subseteq V \times V$ corresponds to a set of edges representing transactions between accounts. If the graph is node-attributed or edge-attributed, the node attribute matrix $\mathbf{X} \in \mathbb{R}^{n \times d_n}$ assigns attributes to each node, and the edge attribute tensor $\mathbf{E} \in \mathbb{R}^{n \times n \times d_e}$ assigns attributes to each edge.

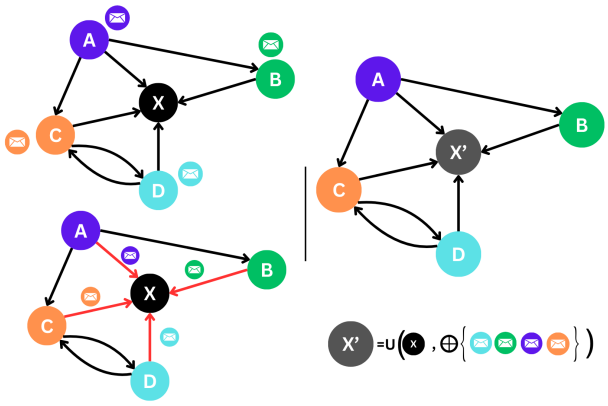

Figure 1: Illustration of the Message Passing framework's two steps: message passing (left) and aggregation (right). Focused only on the node labeled 'X'.

Graph Neural Networks

Graph Neural Networks (GNNs) are a machine learning tool which apply the message-passing (MPNN) paradigm to

graph-structured data. In each layer, a node updates its embedding by aggregating (e.g. summing or averaging) transformed features from its neighbors (messages).

One such layer can be represented as follows:

$$\text{GNN\_Layer}(X, E) = X^{(N)}, \quad \text{where } X^{(0)} = X, \tag{1}$$

$$X_i^{(t+1)} = U\left(X_i^{(t)}, \bigoplus_{j \in \mathcal{N}(i)} M\left(X_i^{(t)}, E_{j \to i}, X_j^{(t)}\right)\right) \tag{2}$$

However, this equation is based on the assumption that edge features stay constant throughout the layer. This assumption may lead to limit the power of the GNN to learn better latent representations of the graph. To address this, more recent work has resulted in the addition of an edge update module within the classic GNN architecture, where edge features are also updated at the same time as node features in the aggregation step, by using a Multi-Layer Perceptron on the edge embedding concatenated with the node embeddings. This approach has generally lead to an improvement in the GNN's performance across many datasets, and represent the current standard for most GNN architectures. As such, a new formulation for a GNN layer emerges:

$$\text{GNN\_LayerEU}(X, E) = (X^{(N)}, E^{(N)}), \tag{3}$$

$$\text{where } X^{(0)} = X, \quad E^{(0)} = E \tag{4}$$

$$E_{j \to i}^{(t+1)} = M_e\left(X_j^{(t)}, X_i^{(t)}, E_{j \to i}^{(t)}\right) \tag{5}$$

$$X_i^{(t+1)} = U\left(X_i^{(t)}, \bigoplus_{j \in \mathcal{N}(i)} M\left(X_i^{(t)}, E_{j \to i}^{(t+1)}, X_j^{(t)}\right)\right) \tag{6}$$

This general neighborhood-aggregation framework has led to many performant models such as GCN, GraphSage, PNA, and more. However, Xu et al. [4] show that most "vanilla" GNNs (e.g. simple graph convolutions, GraphSAGE) lack full expressivity and cannot distinguish certain graph structures, showcasing an inherent limitation of such a model on its own.

Transformers

Another way to generate embeddings for data was introduced in "Attention is All You Need" by Vaswany et al. [5]. There, the authors introduced the concept of self-attention as a powerful alternative to convolutions or recurrence, a key resource in transformer architectures as part of the encoder layer. Self-attention learns more global/long-range dependencies within the input, by processing data all together and letting all elements attend to all others. This discovery has led to many state-of-the-art results in fields such as Natural Language Processing.

Further, within the encoder layer of an architecture, besides the self-attention module, a Feed-forward Neural Network is employed per element, transforming features locally after the global information exchange, resulting in better overall embeddings.

The processing of data through an encoder layer can be expressed mathematically as such:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V \tag{7}$$

$$h_i = \text{Attention}(XW_i^Q, XW_i^K, XW_i^V) \tag{8}$$

$$\text{MultiHead}(X) = \text{Concat}(h_1, \ldots, h_h)W^O \tag{9}$$

$$X' = \text{LayerNorm}(X + \text{MultiHead}(X)) \tag{10}$$

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \tag{11}$$

$$X'' = \text{LayerNorm}(X' + \text{FFN}(X')) \tag{12}$$

$$\text{EncoderLayer}(X) = \text{LayerNorm}\big(\text{LayerNorm}(X + \text{MultiHead}(X)) + \text{FFN}(X')\big) \tag{13}$$

Where the LayerNorm function works to normalize input features' values to stabilize and accelerate the training process. The function re-centers and scales the values of the features to have a mean of 0 and standard deviation of 1, without changing the distribution's shape.

Recent work has extended the power of embeddings using self-attention and adapted it to graph-structured data. Graph Transformers further integrate structural biases inherent to a graph (such as edge connectivity) into self-attention, making them effective for graph representation learning.

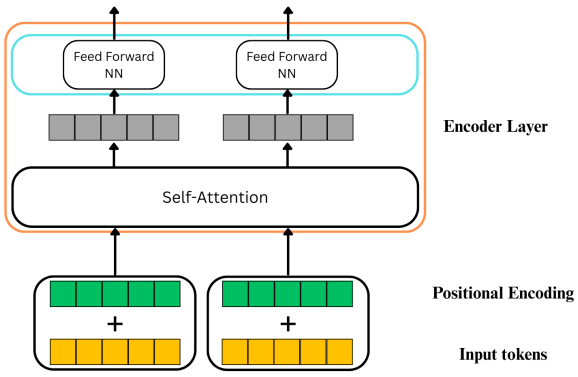

Figure 2: Illustration of the Transformer Encoder module, with Positional Encodings included.

Positional Encoding

Within the work done by Vaswany et al. [5], Positional Encoding was mentioned as one of the methods used by transformers to better parametrize the underlying structure and position of the tokens within the input.

Unlike NLP, within graph-structured data, there is no natural ordering of tokens that can be assumed. However, graphs do possess plenty of structural information, given by the node and edge structure, which can be used in order to generate

meaningful positional encodings. By incorporating encodings that respect the graph structure, transformer-based graph learning models can better capture the relationships between nodes and improve their performance.

Common approaches for graph positional encoding include spectral methods, such as Laplacian eigenvectors, that capture global graph structure, and distance-based encodings that use shortest-path or random walk statistics to reflect relational information between nodes. By incorporating these positional signals into the Transformer's self-attention mechanism, graph-based models can better capture both local and global dependencies.

Fusion layer

Fusion combines multiple feature sources or modalities in order to obtain a more adequate representation of data. In fraud graphs this can mean fusing node features, edge features, or outputs of different models. Fusion is often categorized by when the combination happens.

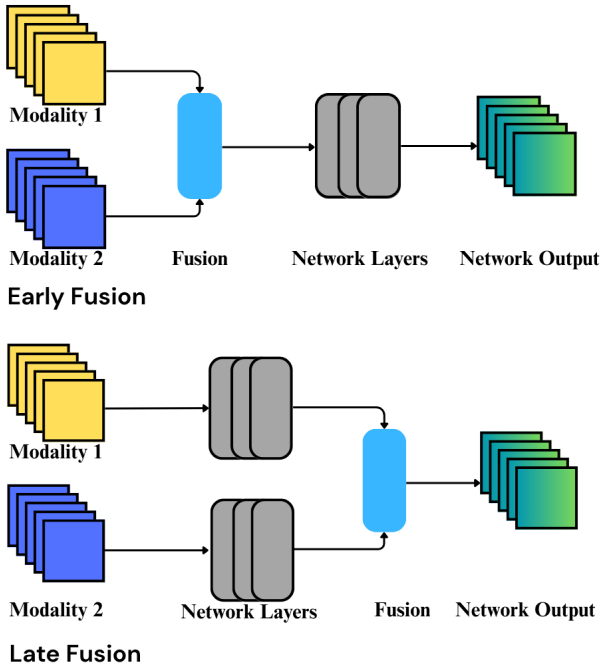

Figure 3: Illustration of early and late fusion methods used by multimodal fusion networks.

Early Fusion (Feature-level Fusion), where data from different modalities (or graph components) are combined at the input or early in the network. Early fusion "allows interactions between modalities to be captured early" [6]. A simple implementation is Concat-then-MLP, different modalities' features are concatenated and passed through MLP layers. This straightforward fusion can work well if the modalities are compatible. However, it may force the model to learn to separate modalities implicitly.

Late Fusion (Decision-level Fusion) , where modalities are processed separately and fused at the end. In this approach, one network branch might learn from node features and another from edge features, then their outputs are combined (e.g. averaged or via a classifier). Late fusion is useful when one modality might be missing at inference time, as each branch can operate independently.

Hybrid/Full Fusion combines early and late strategies. Hybrid fusion merges features at multiple points (both feature-level and decision-level). For instance, one might fuse node and edge information in intermediate layers (early) and also fuse final logits (late). Such multi-stage fusion can capture deeper interactions, but is more complex to design than the former.

2.2 Related Work

Graph Neural Networks

Starting from the foundation of the classic message-passing GNN, Xu et al. introduce the Graph Isomorphism Network (GIN)[4] to overcome its limmited expressive power, designing a GNN probably as powerful as the Weisfeiler-Lehman graph isomorphism test. Similarly, Schlichtkrull et al. proposed the Relational Graph Convolutional Network (R-GCN) [7] for learning over heterogeneous graphs with relation-specific parameters. While both models laid important groundwork in enhancing GNN capabilities, they are largely outclassed in the context of the proposed methods, which explores architectures that go beyond these earlier designs in both flexibility and representational power.

As such, the method used has a greater focus on more recent advances, such as:

Principal Neighbourhood Aggregation [8] Corso et al. argue that using multiple aggregators (mean, max, etc.) and degree-scalers improves a GNN's capacity, especially with continuous features. PNA combines several aggregation functions with learned scaling by node degree, generalizing the standard sum or mean. In their experiments, PNA outperforms prior GNNs on synthetic and real benchmarks. In financial graphs where transaction amounts and frequencies vary widely, PNA's multi-aggregator scheme can capture patterns across different degrees of node connectivity.

Mega-GNN [9] Currently the state-of-the-art in GNN-based AML performance, Bilgi et al propose MEGA-GNN. They improve over the classic GNN by employing a two-stage aggregation process in the message passing layers: "first, parallel edges are aggregated, followed by a node-level aggregation of messages from distinct neighbors" [9]. For the purposes of this work, the focus will be on the variant of MEGA with unidirectional Message Passing which is less powerful than the one integrating both Reverse MP and EgoIDs, but serves as a good comparison point for the PNA.

Overall, modern GNNs combine expressive aggregators, multi-relation handling and parameter scaling to capture complex relational patterns in fraud data.

Transformers

Ying et al. (2021) introduced Graphormer [10], demonstrating that Transformers can perform competitively on graph-structured tasks when enhanced with suitable structural en-

codings. By incorporating features such as shortest-path distances, node centralities, and edge attributes into attention biases, Graphormer adapts the standard Transformer to better capture graph topology. More recently, Lin et al. (2024) proposed FraudGT [11], a Transformer tailored for financial fraud detection. It incorporates edge-aware mechanisms, including message-passing gates and edge-based attention biases, to highlight key transaction features. FraudGT achieves strong performance gains, both in accuracy and computational throughput, on large transaction graphs.

Building on this motivation, the work adopts a standard Transformer encoder within both interleaved and full fusion architectures, leveraging its global attention capabilities while retaining the structural bias of GNNs. This design is informed by prior successes like Graphormer and FraudGT, but simplifies the architecture to focus on integration rather than specialized graph encodings.

Positional Encodings

Within the realm of positional encodings in graph tasks, Kanatsoulis et al. [12] introduce **PEARL**, a general and efficient framework for learnable graph positional encodings (PEs). PEARL leverages the insight that message-passing GNNs act as nonlinear functions of Laplacian eigenvectors, allowing it to approximate powerful, permutation-equivariant functions of graph structure with linear complexity.

Building on PEARL, the **R-PEARL** (Random PEARL) framework replaces deterministic initializations with random Gaussian vectors to learn expressive, eigenvector-like positional encodings. It enables tasks like substructure counting and matches or outperforms spectral methods (e.g., SignNet, SPE) on various benchmarks. While PEARL and R-PEARL demonstrate strong performance with learnable positional encodings, they require careful tuning of hyperparameters (e.g., number of samples, GNN layers) and are sensitive to choices like sampling distribution and architecture.

Fusion Layers

Common fusion mechanisms include simple concatenation plus an MLP or gating units.

Concat + MLP in PEARL The aforementioned positional encoding framework, PEARL, also employs effectively a concatenation plus MLP strategy: it concatenates the positional data with the original node features, and further passes it through a GNN, which has MLP layers. This approach, also adopted in R-PEARL, reinforces the viability of concat-based fusion as a practical and effective method for integrating positional information into graph models.

Gated Multimodal Unit (GMU) [13] learns to weigh modalities via multiplicative gates. A GMU takes two input modality vectors, uses a sigmoid gate to decide how much each modality influences the output, and combines them multiplicatively. Arevalo et al. show that GMUs outperform naive fusion on a movie genre prediction task.

Gating Units - Dynamic Feature Fusion [14] Sheng et al. introduce a novel dynamic feature fusion mechanism that combines global graph structures and local semantics using gated fusion, which enables the selective combination of features from different sources and improves the overall performance of the model. Compared to classic graph neural network methods, the proposed approach with gated fusion achieves better performance and robustness in detecting fraudulent activities in blockchain transactions, particularly in the presence of incomplete or missing data.

In summary, fusion layers let GNN or Transformer models combine multiple information sources. As fusion methods, Concatenation-then-MLP is simple and often effective for combining multiple features, and gated units like GMUs allow the model to learn when and how much to trust each modality.

3 Proposed Architectures

As stated previously, the aim of this work is to better comprehend whether the addition of a transformer layer on top of the MPNN paradigm for graph-structured data yields a sensible improvement. In that direction, this paper proposes two ways of fusing the global graph structure embedded by a Transformer with a local one provided by a Graph Neural Network. Each one is visualized in its own separate illustration, Figure 4 and 5 respectively.

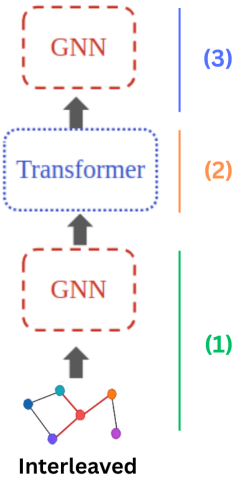

Figure 4: Illustration of the Interleaved architecture

3.1 The Interleaved Architecture

This general architecture capitalizes on the fact that Graph Neural Networks are able to embed local structure within both the nodes and the edges of the graph (1), which gives a good local perspective of the transactional graph for the transformer to learn from (2). On top of that, a second message-passing round more evenly distributes the information among neighboring nodes to capture even more of the graph's underlying structure(3).

In essence, the Interleaved architecture can be expressed through the following equations, with reference to equations 6 and 13 from Section 2.

$$(X^{(1)}, E^{(1)}) = \text{GNNLayerEU}(X, E) \tag{14}$$

$$E^{(2)}_{i\to j} = \text{EncoderLayer}(E^{(1)}_{i\to j}) \quad \forall (i,j) \in \mathcal{E} \tag{15}$$

$$(X^{(2)}, E^{(3)}) = \text{GNNLayerEU}(X^{(1)}, E^{(2)}) \tag{16}$$

The final graph embedding is, as such, $(X^{(2)}, E^{(3)})$, which is then fed through a prediction head consisting of an MLP that takes as input the edge features concatenated with both node features, and outputs a binary value, stating whether the model considers the transaction illicit or not.

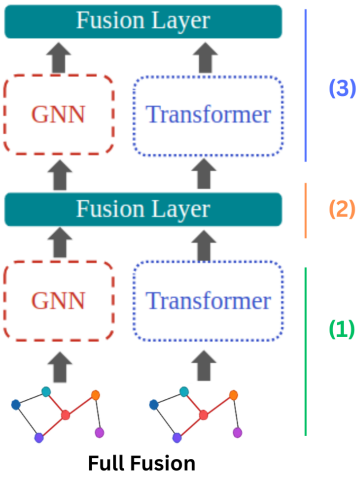

Figure 5: Illustration of the Full Fusion architecture

3.2 The Full Fusion Architecture

Conversely, the Full Fusion approach capitalizes on the fact that the models capture different modalities and may work with different embedding sizes, allowing for more expressive representations of data. The input is passed in parallel through both a Transformer and a GNN (1), the edge embeddings are then fused through an early fusion approach (2), which is then fed again through a Transformer and GNN and fused at the end to capture more of the graph structure before the prediction head (3). Besides the potential to collect more relevant and valuable relationships with more adequate embedding sizes, which Transformers especially benefit from, there is also the potential of controlling specifically how the combination of information is done: As specified in section 2, gating mechanisms may provide a learnable way for the model to consider how much of each representation should be kept in the final graph embedding.

Using the same notation as before, the fusion architecture can be written as an equation as expressed below. Note that the "Fuse" function works by taking two embedding tensors and returns one with both modalities merged.

$$E^{(1)}_{i\to j,T} = \text{EncoderLayer}(E_{i\to j}) \quad \forall (i,j) \in \mathcal{E} \tag{17}$$

$$(X^{(1)}, E^{(1)}_G) = \text{GNNLayerEU}(X, E) \tag{18}$$

$$E^{(1)}_{\text{fused}} = \text{Fuse}\big(E^{(1)}_T, E^{(1)}_G\big) \tag{19}$$

$$E^{(2)}_{i\to j,T} = \text{EncoderLayer}(E_{i\to j,\text{fused}}) \quad \forall (i,j) \in \mathcal{E} \tag{20}$$

$$(X^{(2)}, E^{(2)}_G) = \text{GNNLayerEU}\big(X^{(1)}, E^{(1)}_{\text{fused}}\big) \tag{21}$$

$$E^{(2)}_{\text{fused}} = \text{Fuse}\big(E^{(2)}_T, E^{(2)}_G\big) \tag{22}$$

Similarly to the Interleaved architecture, the final graph embedding $(X^{(2)}, E^{(2)}_{\text{fused}})$ is fed through the same prediction head, concatentating node and edge embeddings and being passed through a neural network with a binary output.

3.3 Integrations with Prior Work

In order to obtain greater results on the AML datasets, the architectures were refined with the techniques presented in the "Related Work" section. More precisely, the following is studied:

R-PEARL generates node-level positional encodings [12] which are applied prior to the first GNN layer. This choice reflects R-PEARL's design focus on generating structural encodings for nodes. These encodings are then propagated to the edges through the edge MLPs within the GNN layer. As a result, the subsequent transformer layer on edge embeddings can leverage structural information derived from R-PEARL, allowing it to better capture the underlying graph topology.

MEGA-GNN [9] is used as a core GNN layer in both architectures. MEGA-GNN is particularly well-suited for multigraphs due to its sophisticated edge update functions and multi-channel aggregation mechanisms. Additionally, it achieves strong standalone performance on AML datasets, approaching state-of-the-art F1 scores, making it a natural fit for this task.

GMU-based fusion layers are implemented within the Full Fusion architecture, inspired by prior work demonstrating their effectiveness in fraud detection tasks [14].

4 Experimental Setup

Dataset In order to quantify the results of the architectures and their possible improvements over their components, each model is tested on the IBM Anti Money Laundering (AML) datasets, more particularly the Small datasets, with both degrees of illicit activites. The aim of this dataset is (binary) edge classification: decide whether a transaction is illicit or not based on information pertaining to it, such as amount of money sent, identifier of the sender, identifier of the person who receives the money, and more. More particular details can be found on the dataset's page [3].

During initialization, the dataset is reformatted to the following table-like structure: `[EdgeID, from_id, to_id, Timestamp, Amount Sent, Sent Currency, Amount Received, Received`

`Currency, Payment Format, Is Laundering]` and then further converted to a *networkx* graph representation. For each epoch, a node is picked at random and then an n-hop neighbourhood is sampled with a fixed amount of edges chosen at random at each hop.

The dataset was partitioned into training, validation, and test sets using a 60%-20%-20% split respectively. Furthermore, the split is based on each day of transactions, such that all datasets contain information of transactions in all days, for higher accuracy. This allocation was chosen to ensure a sufficiently large training set for model learning while preserving adequate validation and test samples for reliable tuning and performance evaluation.

Implementation details The implementation and training of all models was facilitated by the use of the PyTorch and PyTorch Geometric frameworks. The models have been trained using the AdamW [15] optimizer for a maximum of 60 epochs on the AML-Small datasets. This threshold has been chosen as such because, through testing, no further improvements were achieved by going over this limit. Learning rates are adjusted for each model and change over epochs using a cosine with warmup scheduler to achieve better convergence on the datasets. To ensure stable training, gradient clipping with a maximum norm of 1.0 was applied.

Similarly, other hyperparameters such as dropout rates, activation functions have been tuned to the best values which were found empirically.

The only parameter that may be perceived as suboptimal is the batch size, which, due to the large amount of edges within a batch, was limited to an effective size of 4096 (for models that cannot accommodate the size stored in CUDA memory, batch accumulation was employed as a means to emulate it).

For reproducibility, all of the runs used in the process of gathering the data have been assigned a random seed, from which the results in the next section can be derived.

More implementation details can be found within the project repository (see Appendix A.2).

Metrics The F1-score is used as the main metric to quantify each model's capabilities. The average F1-score of the minority class (illicit transactions) on the test set is reported only for the model that achieved the highest validation performance. This metric was chosen because the AML datasets feature a high class imbalance, with legal transactions being prevalent and the task of classifying transactions based on their likelihood to be illicit is inherently binary in scope. The F1-score is computed as follows:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{23}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{24}$$

$$F_1\text{-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{25}$$

Furthermore, the sample standard deviation is computed and displayed to account for the randomness in initialization and edge sampling.

Baselines For all results, two main baselines have been chosen:
The PNA GNN by Corso et al. [8] with edge updates, serving as a reference point in most literature on the AML dataset [11], [9];
The MEGA-PNA GNN by Bilgi et al. [9], representing the state-of-the-art in GNN performance on edge classification on the AML dataset.

Hardware All models have been trained using the DAIC High-Performance Cluster [16], using NVIDIA A40 and L40 GPUs in equal proportions for all tests.

5 Results

Table 2 lists the results of each method across the datasets, with standard deviations calculated over a minimum of 3 and an average of 5 runs. Each cell has been shaded in accordance with the model's performance versus the highest measured value on the datasets: the darker the green color, the better the model's performance.

Furthermore, Table 1 represents the result of an ablation study performed in order to show the impact the architecture has, having parameter counts comparable to the baseline.

Performance Comparison Table 2 shows that both the Interleaved and Full Fusion architectures consistently outperform both the traditional PNA baseline and the state-of-the-art GNN proposed by Bilgi et al., with increases in F1 score as high as 10% on the Small_LI dataset. This improvement is less visible on the datasets with higher degrees of illicit activity, Small_HI in this case, where the increments reach only up to half as much (4.5% compared to the baseline).

Ablation	Small_LI	Small_HI	#Params
PNA+EU	22.29 ± 3.00	67.32 ± 3.22	68,007
Interleaved	$\mathbf{30.87 \pm 3.21}$	$\underline{\textit{68.27 \pm 3.27}}$	63,297
Fusion (C-MLP)	27.37 ± 1.91	66.98 ± 1.07	72,757
Fusion (GMU)	$\underline{\textit{29.34 \pm 2.13}}$	$\mathbf{69.71 \pm 2.02}$	71,137

Table 1: Ablation study of models with similar number of parameters relative to one another, showing the Minority-class F1 scores (%). The best two results are highlighted.

Transformer-based Edge Embedding Results These results underscore a clear trend: incorporating transformer modules into edge embeddings leads to consistent and controlled performance improvements. This is also solidified by the ablation study conducted in Table 1. In situations where parameter counts are similar in value, the expressive power of the transformer still brings a consistent improvement which is more visible on the datasets with low illicit activity.

Table 2: Minority-class F1-scores (%) of the models on the AML datasets. Best performing are shown in **boldface**, second-best in *underlined italics*.

Model — Dataset	Small_LI	Small_HI	#Params
PNA+EU	22.66 ± 2.97	65.29 ± 2.33	32,197
MEGA-PNA + EU	40.80 ± 2.95	73.20 ± 0.45	41,837
Interleaved	30.87 ± 3.21	68.27 ± 3.27	63,297
+ PEARL	30.14 ± 2.87	65.21 ± 4.03	64,463
+ MEGA	**46.05 ± 0.55**	74.36 ± 0.67	82,577
+ MEGA, PEARL	45.15 ± 1.58	*74.57 ± 1.00*	83,743
Full Fusion (Concat-MLP)	27.37 ± 4.08	66.98 ± 2.17	72,757
+ PEARL	32.78 ± 3.51	67.08 ± 2.68	73,923
+ GMU	29.34 ± 1.77	69.71 ± 2.11	71,137
+ GMU, PEARL	32.10 ± 3.07	68.62 ± 2.69	72,303
+ GMU, MEGA	*45.85 ± 1.59*	**74.97 ± 0.78**	90,417

GMU Fusion The GMU (Gated Multimodal Unit) fusion layer contributes significantly to the effectiveness of the Full Fusion architecture compared to the Concatenate-then-MLP approach. The addition of a gating mechanism on top of the classic MLP and weighted average approach allowed for a more robust method to prioritize embeddings differently in early and late fusion, resulting in increased robustness, reduced parameter counts, and close to no additional training overhead. In addition, there is a visible reduction in the standard deviation of the architectures employing the GMU compared to the ones without it, indicating that such a fusion increases consistency in gathering adequate results, no matter the initialization.

Limitations of R-PEARL Finally, Table 2 shows two different trends between the datasets: on the Small_HI dataset, the architectures using PEARL typically yield comparable or reduced performance compared to their counterparts. However, this contrasts with the low-illicit dataset, where PEARL indeed shows a perceptible improvement as an addition to the Full Fusion models, reaching as high as a 5% increase in F1 score. This pattern likely reflects the sensitivity of PEARL-based positional encodings to hyperparameters and data: Different initializations can lead to significant variance in outcomes.

6 Conclusions and Future Work

This paper addressed the research question of whether modifying the classic GNN-based edge classification architectures with Transformer models operating on edge embeddings leads to measurable improvements in performance on financial fraud detection tasks. Using the IBM AML dataset as a benchmark, two hybrid architectures, Interleaved and Full Fusion specifically, were evaluated and compared against SOTA and common benchmarks.

The results demonstrate that both proposed architectures significantly outperform standard GNN baselines, including the highly performant MEGA-GNN, with improvements reaching up to 10% in F1-score on datasets with sparse illicit activity. The Interleaved model, in particular, showed strong gains through sequential exploitation of local and global structure, while the Full Fusion model proved the value of multimodal integration, especially when enhanced with gated fusion strategies like the GMU.

However, the impact of PEARL-based positional encodings remains inconclusive, often resulting in neutral or negative performance compared to models without it. This outcome shows a need for more consistent hyperparameter optimization and deeper analysis of positional encoding schemes in future research.

Moreover, the modularity and task-agnostic nature of the proposed frameworks make them adaptable to other domains and datasets beyond financial fraud, though such generalization was not explored due to time constraints. Future work should investigate (1) broader and more diverse datasets, (2) improved neighborhood sampling techniques to reduce variance, and (3) further exploration and tuning of architectural components such as positional encodings and fusion methods.

Through this work, empirical evidence is provided that combining the structural bias of GNNs with the representational capacity of Transformers offers a promising direction for graph-based edge classification that is both performant and extensible.

7 Responsible Research

This piece of research has been conducted by making responsible use of AI technologies. The primary aim of this work was to improve a model created to protect consumers, while operating within the legal and ethical white lines. To that end, all data utilized in this study, the AML dataset, is synthetically generated rather than drawn from real individuals or organizations, which both gives applicability of the results and maintains the integrity and privacy of actual people's data.

To support reproducibility, this paper provides a detailed description of the methodology, including the project setup and tools used. The code behind this paper and all hyperparameters used have been open-sourced, see Appendix A.2. Any feedback, replication, and improvement of the existing

codebase to ensure higher robustness and transparency of the results is welcome.

8 Acknowledgements

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 author, not the DAIC team.

A Appendix

A.1 Mathematical Notation Table

Symbol	Explanation
$X_i^{(t)}$	Feature of node i at layer t
$X_i^{(t+1)}$	Updated feature after aggregation
$E_{j \to i}$	Edge feature from j to i
$\mathcal{N}(i)$	Neighbors of node i
N	Number of hops to sample
$M(\cdot)$	Message function
$U(\cdot)$	Update function
$m_{j \to i}^{(t)}$	Message sent from j to i at layer t
$\bigoplus$	Aggregation operator (e.g., concat)
d_i	Degree of node i
$\mathcal{A}$	Aggregator set (e.g., mean, max)
$\mathcal{S}$	Scaler set (e.g., identity, deg)
$s(d_i)$	Scaler applied to degree d_i
$\mu(\cdot), \sigma(\cdot)$	Mean and std aggregators
$\max(\cdot), \min(\cdot)$	Max and min aggregators
$\otimes$	Tensor product (outer product)
M_e	Edge update function (eMLP)
X	Input to Transformer encoder layer
X'	Output of multi-head attention + residual + norm
X''	Output of feedforward layer + residual + norm
W^Q, W^K, W^V, W^O	Learnable projection matrices for attention
Q, K, V	Query, key, and value matrices
d	Hidden (embedding) dimension of model
d_k	Dimension of query/key vectors
FFN$(\cdot)$	Feed-forward network
LayerNorm$(\cdot)$	Layer normalization function

Table 3: Notation used in equations, grouped by model.

A.2 Project Repository

This section covers the structure of the project repository used for acquiring the data presented in this research paper.

The repository can be found at the following GitHub link. To encourage further experimentation, all hyperparameters specified for the runs is present within the `/params` folder, in YAML format.

Besides hyperparameters, the repository contains implementations for the components used within the paper. The project itself is a combination of the MEGA-GNN [9], FraudGT [11] and R-PEARL [12] codebases, with the GMU being implemented by hand from the description provided by Arevalo et al. [13].

In addition to the components described above, the repository is organized to facilitate both reproducibility and extensibility. The directory structure separates model implementations, experimental parameters, and utility scripts, allowing researchers to easily navigate and modify the codebase. Detailed instructions for setting up the environment and reproducing results are provided in the README, alongside example scripts for running experiments. All dependencies are specified in a requirements file to ensure consistency across different systems.

To encourage further work, the codebase is modular: new models or datasets can be added with minimal changes, and all major components are documented to aid understanding and extension. The repository is publicly available and open to extend / fork.

A.3 Statement on the Use of Large Language Models

With respect to the policy on Generative AI use, Large Language models were leveraged in the creation of this report. The scope of application was limited, solely as a means to improve LaTeX code, ensure proper accreditation, and get constructive feedback on writing. The prompts for the aforementioned categories are, verbatim, as follows:

- "(Image) How legible are the numbers/results in this table?"
- "How can I make my references in latex to be in the correct order (i.e, numbered in the order they are cited)?"
- "(Paragraph) Give me some feedback for how I phrased this section."

References

[1] Leman Akoglu, Hanghang Tong, and Danai Koutra. Graph-based anomaly detection and description: A survey. *Data mining and knowledge discovery*, 29(3):626–688, 2015.

[2] 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.

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

[4] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks?, 2019.

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

[6] Songtao Li and Hao Tang. Multimodal alignment and fusion: A survey, 2024.

[7] Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks, 2017.

[8] Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. Principal Neighbourhood Aggregation for Graph Nets, December 2020. arXiv:2004.05718 [cs].

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

[10] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform bad for graph representation?, 2021.

[11] 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.

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

[13] John Arevalo, Thamar Solorio, Manuel Montes y Gómez, and Fabio A. González. Gated multimodal units for information fusion, 2017.

[14] Zhang Sheng, Liangliang Song, and Yanbin Wang. Dynamic feature fusion: Combining global graph structures and local semantics for blockchain fraud detection, 2025.

[15] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019.

[16] Delft AI Cluster (DAIC). The delft ai cluster (daic), rrid:scr_025091. https://doi.org/10.4233/rrid:scr_025091, 2024. Accessed: 2025-05-23.