Quantum Topological Graph Neural Networks for Financial Anomaly Detection Review

Martino Papa

January 15, 2026

Abstract

In this article I review the paper "Quantum topological graph neural networks for detecting complex fraud patterns" [2] which proposes a novel approach to detect financial anomalies by combining graph neural networks, topological data analysis and quantum machine learning techniques. A special focus is given to the definition of the loss function since the original paper doesn't define it explicitly. A simulated implementation and evaluation of the model is also presented to analyze its performance and efficiency compared to classical methods.

1 Introduction

Nowadays detecting financial anomalies is further complicated by the increasing volume of transactions, the sophistication of fraudulent schemes and the diversity of the transaction data (temporal, relational and multimodal features). Latest theoretical advances in machine learning, quantum computation and topology offer promising tools to address these challenges.

Graph neural networks (GNNs) have shown great potential in modeling financial transaction due to their ability to capture relational information and complex patterns. Albeit that GNNs struggle to capture long-range dependencies, global graph properties and non-linear correlations which are critical for identifying fraudulent behaviors [6]. Furthermore these models usually require significant computational resources and large labeled datasets. In addition most deep learning models are black-boxes, they provide very limited interpretability which is a key requirement in financial applications due to regulatory and ethical considerations [1].

Quantum machine learning (QML) has latlely emerged, promising an exponential speedup in embedding and optimization tasks [4] and better explainability through quantum phenomena such as superposition and entanglement.

Furthermore Variational Quantum Circuits (VQC) have demonstrated applicability in classification, clustering, and pattern recognition problems. Hybrid quantum-classical architectures offer solutions for resource-limited NISQ devices [10].

Definition 1 (NISQ devices)**.** *NISQ (Noisy Intermediate-Scale Quantum) devices are quantum computers that operate with a limited number of qubits and are prone to errors due to noise and decoherence.*

Parallelly, Topological Data Analysis (TDA) provides tools to extract invariant features that characterize the global structure of data. An example is persistence homology, which remain stable under noise or minor perturbations [5].

In this report I will explore the paper [2] which proposes a QTGNN (Quantum Topological Graph Neural Network), combining both GNNs with TDA and quantum machine learning techniques to improve anomaly detection in financial transactions. This new model is designed to balance theoretical rigor with practical fesability, the key contributions of the paper are:

- Leveraging quantum embeddings for richer relational representation;
- Incorporating topological invariants for robustness and interpretability (such as persistence homology);
- Ensuring scalability and practicality on NISQ devices.

2 Model description

During the years the effectiveness of graph-based models in fraud detection brought them to be the standard approach in the field [8]. For our project we will consider a financial transaction graph $G = (V, E)$, $|V| \doteq N$, where each node $v \in V$ represents an account and each edge $(u, v) \in E$ represents a transactions between accounts u and v. Each edge is associated with a weight $W(u, v) = w_{uv}$ defined as

$$w_{uv} \doteq \sum_{t \in T_{uv}} \text{amount}(t) \tag{1}$$

where T_{uv} is the set of transactions between accounts u and v. Node centrality is also taken in consideration and it's computed as

$$\alpha_i \doteq \frac{\deg(i)}{\sum_{j \in V} \deg(j)} \tag{2}$$

Before moving forward the graph is filtered to remove edges with low weights

$$E \leftarrow \{(u, v) \in E \mid w_{uv} > \tau_{\text{filter}}\} \tag{3}$$

2.1 Generalized Amplitude Encoding with Entanglement Seeding (GAES)

GAES is performed to encode the graph structure and features into quantum states. The idea behind standard amplitude encoding is to represent data as amplitudes of a quantum state. Given a vector $v \in \mathbb{R}^N, N = 2^n$ with $\|v\| = 1$, the corresponding quantum state is

$$|\psi\rangle = \sum_{i=0}^{2^n-1} v_i |i\rangle \tag{4}$$

where ψ is a quantum state with n qubits.

In our case we want to encode not only node features but also graph structure. To achieve this the authors propose to use entanglement seeding, which introduces entanglement between qubits to capture relationships between nodes.

$$|\psi_G\rangle = \frac{1}{\sqrt{Z}} \sum_{i,j} A_{ij} \mathcal{E}_{ij}(\theta_e) |i\rangle |j\rangle \tag{5}$$

where

$$A_{ij} \doteq \begin{cases} w_{ij} & \text{if } i \neq j \\ \alpha_i & \text{if } i = j \end{cases} \tag{6}$$

$$Z \doteq \sum_{i,j} |A_{ij}|^2 \tag{7}$$

and

$$\mathcal{E}_{ij}(\theta_e) \doteq \exp(-i\theta_e \sigma_i^x \sigma_j^x) \tag{8}$$

is a quantum gate that applies entanglement between qubits i and j, σ_i^A with $A \in \{X, Y, Z\}$ are the Pauli operators acting on qubit i and θ_e are parameters that will be optimized during training so that the state physically embodies the interactions defined by $H(G)$.

$H(G)$ is the graph Hamiltonian and it includes higher-order interactions to capture multi-party transactions. The original paper define it as

$$H(G) \doteq \sum_{(i,j) \in E} w_{ij}(\sigma_i^x \sigma_j^x + \sigma_i^y \sigma_j^y + \sigma_i^z \sigma_j^z) + \sum_{i \in V} \alpha_i \sigma_i^z + \sum_{(i,j,k) \in \Delta} \gamma_{ijk} \sigma_i^z \sigma_j^z \sigma_k^z \tag{9}$$

where γ_{ijk} captures the three-way interactions between nodes i, j, and k. The original paper doesn't provide a specific definition for γ_{ijk}, just the description of its purpose. A possible interpretation is the geometric mean of the weights of the edges forming the triangle.

$$\gamma_{ijk} \doteq \sqrt[3]{w_{ij} \cdot w_{jk} \cdot w_{ki}} \tag{10}$$

From my prespective there is a big flaw in the definition of the Hamiltonian. From equation 9 we can see that the Paoli gates are being applied to the i-th qubits, this index i layes in the range $[0, N-1]$ where N is the number of nodes in the graph. But from equation 5 we can see that after the encoding we will have $2n$ qubits, where $n = \log_2(N)$. This is a contradiction and obviously we cannot afford to have N qubits since in real world applications N can be in the order of millions.

My solution is to redefine the Hamiltonian using lernable parameters that will capture the interactions between nodes without explicitly mapping them to qubits. The new definition becomes

$$H \doteq H_{eff}(\Theta) = \underbrace{\sum_{q=0}^{n-1} \phi_q \sigma_q^z}_{\text{Global Bias}} + \underbrace{\sum_{q<r}^{n-1} \theta_{qr} \sigma_q^z \sigma_r^z}_{\text{Qubit Correlations}} + \underbrace{\sum_{q<r<s}^{n-1} \psi_{qrs} \sigma_q^z \sigma_r^z \sigma_s^z}_{\text{Higher-Order Entanglement}} \tag{11}$$

This definition is supported by the paper "*A Quantum Approximate Optimization Algorithm*" [3] and it will be used in the Variational Quantum Graph Convolution section.

After the encoding we will end up with a quantum state ψ_G composed by $2n$ qubits, where $n \doteq \log_2(N)$. Usually for amplitude encoding we would need only n qubits, the other n are defined as *enviorment* and are introduced to perform the entaglement seeding. In the equation 5 those correspond to the second register $|j\rangle$.

$$|\psi_G\rangle \in \mathcal{H}_{\text{source}} \otimes \mathcal{H}_{\text{env}} \tag{12}$$

Since we are interested only in the source register, we will trace out the environment qubits to obtain a reduced density matrix ρ_G. The use of the density matrix alows to represent mixed states and capture statistical properties of the quantum system.

$$\rho_{\text{total}} = |\psi_G\rangle\langle\psi_G| \tag{13}$$

$$\rho_G = \text{Tr}_{\text{env}}(\rho_{\text{total}}) \tag{14}$$

where

$$\text{Tr}_{\text{env}}(\rho_{\text{total}}) \doteq \sum_{m \in \text{env}} (\mathbb{I} \otimes \langle m|)\rho_{\text{total}}(\mathbb{I} \otimes |m\rangle) \tag{15}$$

In the end we will obtain a quantum state ρ_G composed by n qubits that encapsulates the information of the original graph G by effectively incorporating the entanglement effects introduced during the encoding process. Although the final representation needs only n qubits, a quantum computer with $2n$ qubits will be needed to perform the encoding.

2.2 Variational Quantum Graph Convolution with Non-Linear Dynamics

During this step parameterized quantum circuits perform non-linear graph convolutions, incorporating higher-order neighborhood interactions to model transaction flows evolving through the layers with the non linear operator

$$\rho^{(0)} = \rho_G \tag{16}$$

$$\rho^{(l+1)} = \mathcal{N}_\theta^{(l)}(U_\theta^{(l)} \rho^{(l)} U_\theta^{(l)\dagger}) \tag{17}$$

where $\mathcal{N}_\theta^{(l)}(\rho)$ is the non-linear quantum channel defined as

$$\mathcal{N}_\theta^{(l)}(\rho) = \sum_k p_k(\theta) \Pi_k \rho \Pi_k \tag{18}$$

$U_\theta^{(l)}$ is th unitary operator

$$U_\theta^{(l)} = \exp\left(-i \sum_{(i,j)\in E} \theta_{ij}^{(l)} H_{ij} - i \sum_{i \in V} \phi_i^{(l)} \sigma_i^z - i \sum_{(i,j,k)\in\Delta} \psi_{ijk}^{(l)} \sigma_i^z \sigma_j^z \sigma_k^z \right) \tag{19}$$

In the original paper the Kraus operators Π_k and the probabilities $p_k(\theta)$ are not explicitly defined. In my implementation, in regard to preserve the node information (diagonal terms of the density matrix) while filtering the complex relational patterns, i chose to implement the *Phase-Damping (Dephasing) Channel*. This should break unitarity, enabling the model to learn complex, non-linear boundaries between legitimate and fraudulent transactions [7]. The mathematical definition of Kraus operators becomes

$$\Pi_1 = I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \Pi_2 = \sigma^z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{20}$$

Notice that they both satisfy the completeness relation

$$\sum_k E_k^\dagger E_k = I \tag{21}$$

The associated probabilities depend on the tunable parameters θ_1, θ_2 and are defined as

$$p_0(\theta) = \frac{e^{\theta_0}}{e^{\theta_0} + e^{\theta_1}}, \quad p_1(\theta) = \frac{e^{\theta_1}}{e^{\theta_0} + e^{\theta_1}} \tag{22}$$

The bigger will be the value of θ_1 the more noise will be added to the system. θ should be initialized in a way that $p_0(\theta) \gg p_1(\theta)$, the network will learn to increase the "noise" (phase damping) in deeper layers to strip away irrelevant correlations and focus on the core topological features of the fraud graph.

$\theta_{ij}^{(l)}$, $\phi_i^{(l)}$ and $\psi_{ijk}^{(l)}$ are tunable parameters that captures transaction dynamics, account-specific dynamics and multi-party (hyperedge) interactions respectively. The quantum feature embedding after L layers will be

$$Z_Q = Tr_{anc}\left(\prod_{l=1}^{L} \mathcal{N}_\theta^{(l)}(U_\theta^{(l)} \rho_G U_\theta^{(l)\dagger})\right) \tag{23}$$

Where Tr_{anc} is defined as for Tr_{env} and it traces out ancilla qubits that were introduced to apply the Kraus operators.

We also compute the correlation entropy which captures multi-scale transaction interactions

$$C_Q = \sum_{i \neq j} Tr(\rho_{ij}^{(L)} log\ \rho_{ij}^{(L)} - \rho_i^{(L)} log\ \rho_i^{(L)} - \rho_j^{(L)} log\ \rho_j^{(L)}) \tag{24}$$

where $\text{Tr}(\rho) = \sum_k \rho_{kk}$.

Both Z_Q and C_Q will be used as features in the final classification.

2.3 Topological Quantum Signature Extraction with Higher-Order Invariants

In this section we will extract Quantum Betti numbers and persistent Euler characteristic from the quantum-encoded graph.

Definition 2 (Betti numbers)**.** *Betti numbers act as a "counter" for different types of holes in a topological space or object. Each Betti number β_i corresponds to the number of i-cells in the CW-complex (graph). In particular, β_0 is the number of connected components, β_1 counts the number of one-dimensional holes (Cycles), β_2 counts the number of two-dimensional voids (cavities). In our methods higher-order Quantum Betti numbers are computed to extend the analysis beyond simple loops.*

Definition 3 (Euler's characteristic)**.** *Let τ be a CW-complex composed by β_i i-cells. We define Euler's characteristic as*

$$\chi(\tau) = \sum_{i=0}^{\infty} (-1)^i \beta_i \tag{25}$$

The Euler's characteristic is a topological invariant that combines information from all Betti numbers into a single scalar value.

The original paper doesn't specify how many Betti numbers are computed, in my implementation I decided to compute up to β_3 to capture complex topological features.

From now on we define $\rho \doteq \rho^{(L)}$ from the last layer of the Variational Quantum Graph Convolution. Persistent homology is extended to compute higher-order topological invariants over a multi-scale quantum distance landscape, capturing structural anomalies in transaction networks. The quantum distance matrix uses a weighted metric defined as follows.

$$d_\rho(i,j) = \sqrt{1 - F_w(\rho_i, \rho_j)} \tag{26}$$

where

$$F_w(\rho_i, \rho_j) = \frac{(Tr\sqrt{\sqrt{\rho_i} W \rho_j \sqrt{\rho_i}})^2}{Tr(W)} \tag{27}$$

and W is a positive definite weight matrix encoding priors like transaction frequency and account centrality. The quantum Vietoris–Rips complex is defined as the simplicial complex formed by connecting nodes that are within a distance ϵ of each other.

$$\mathcal{VR}_\epsilon(\rho) = \{\sigma \subseteq V \mid d_\rho(i,j) \leq \epsilon, \forall i,j \in \sigma\} \tag{28}$$

Then the Quantum Betti numbers $\beta_k(\epsilon)$ are computed as

$$\beta_k^Q(\epsilon) = \dim H_k(\mathcal{VR}_\epsilon(\rho); \mathbb{Q}) \tag{29}$$

where H_k is the k-th homology group with rational coefficients defined as

$$H_k(\mathcal{VR}_\epsilon(\rho); \mathbb{Q}) = \frac{\ker \partial_k}{\operatorname{im} \partial_{k+1}} \tag{30}$$

with $\partial_k : C_k \to C_{k-1}$ is the boundary operator, a linear map that takes a k-dimensional shape and returns its $(k-1)$-dimensional boundary.

Once the Quantum Betti numbers are computed we can define the Persistent Euler Characteristic (PEC) as

$$\chi^Q(\epsilon) = \sum_k (-1)^k \beta_k^Q(\epsilon) \tag{31}$$

The value of ϵ should be tuned and, since $d_\rho(i,j) \in [0, \sqrt{2}]$, we should choose values in that range. More then one value can be choosed and used as features for the final classification. Adding those topological features to the quantum features Z_Q and C_Q we obtain the final feature vector

$$\phi(G) = [Z_Q, C_Q, \{\beta_k^Q(\epsilon)\}_{k,\epsilon}, \{\chi^Q(\epsilon)\}_\epsilon] \tag{32}$$

2.4 Quantum-Classical Hybrid Anomaly Learning with Adaptive Optimization

In this section a hybrid quantum-classical architecture is proposed to perform the final anomaly detection.

Problem: the definition of S_{normal} In the original paper a weight vector S_{normal} is required as input and it's described as a representation of normal transaction patterns. The exact definition was not provided but from the definition of the loss function it's clear that S_{normal} should contain as values the feature vectors $\phi(G)$ of graphs that are known to be legitimate. This, from my prospective, is a big flaw in the original paper. In fact, the value of $\phi(G), G \in$ legitimate cannot be known before the training and therfore it cannot be passed as input to the model. I'm sure that in their implementation they found a way to compute an estimate of S_{normal} and maybe then update it during the traing, but I think that it should have been mentioned in the paper.

To overcome this issue i firstly tried a method inspired by "Deep One-Class Classification" [9] that collapses normal data into a single mean vector (see Appendix A). Unfortunately, the assumption of using a hypersphere to model normal transactions turned out to be too restrictive and the model struggled to generalize to unseen fraud patterns.

By implementing S_{normal} as a Memory Bank we maintain a dynamic collection of feature vectors from the most recent legitimate graphs. This allows the model to represent the "normal" class as a multimodal manifold, effectively capturing distinct clusters of valid behavior (e.g., high-frequency small trading vs. low-frequency large transfers) without averaging them into an inaccurate center.

We define the Memory Bank at training step t as a set $\mathcal{M}^{(t)}$ with fixed capacity K:

$$\mathcal{M}^{(t)} \doteq \{\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_K\} \tag{33}$$

where each $\mathbf{m}_k \in \mathbb{R}^d$ is a stored feature vector $\phi(G)$ of a previously processed legitimate graph. The bank is updated via a First-In-First-Out (FIFO) queue mechanism. Let $V^{(t)}_{\text{normal}}$ be the set of feature vectors from legitimate graphs in the current batch.

To ensure the unsupervised loss is well-defined at the start of training, the memory bank is initialized using a "warm-up" phase. Let $\mathcal{G}_{\text{init}} \subset \mathcal{G}_{\text{normal}}$ be a subset of K legitimate graphs sampled before training begins. We define the initial state $\mathcal{M}^{(0)}$ as:

$$\mathcal{M}^{(0)} \doteq \{\phi(G; \Theta^{(0)}) \mid G \in \mathcal{G}_{\text{init}}\} \tag{34}$$

where $\Theta^{(0)}$ represents the randomly initialized network parameters. This ensures that the bank contains a representative distribution of the untrained feature space, preventing model collapse during the early training iterations. Afterwards the update rule is:

$$\mathcal{M}^{(t+1)} \leftarrow \left(\mathcal{M}^{(t)} \setminus \mathcal{M}_{\text{oldest}}\right) \cup V^{(t)}_{\text{normal}} \tag{35}$$

where $\mathcal{M}_{\text{oldest}}$ represents the set of vectors removed to make space for the new entries.

The anomaly detection is formulated as a hypothesis test based on the distance to the *nearest* neighbor in the memory bank, rather than a single center:

$$\mathcal{H}_0 : \min_{\mathbf{m} \in \mathcal{M}} ||\phi(G) - \mathbf{m}||^2 \leq \tau \quad \text{(Normal)} \tag{36}$$

$$\mathcal{H}_1 : \min_{\mathbf{m} \in \mathcal{M}} ||\phi(G) - \mathbf{m}||^2 > \tau \quad \text{(Anomaly)} \tag{37}$$

The loss function is defined as

$$\mathcal{L} = \mathcal{L}_{sup}(\phi(G), y; \Theta) + \lambda_1 \mathcal{L}_{\text{unsup}}(\phi(G), \mathcal{M}) + \lambda_2 \mathcal{R}(\Theta) \tag{38}$$

where the supervised term remains:

$$\mathcal{L}_{sup}(\phi(G), y; \Theta) = -\left[y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})\right] \tag{39}$$

with $\hat{y} = \sigma(W^\top \phi(G) + b)$. The unsupervised loss term is redefined to minimize the distance between a normal graph's embedding and its closest prototype in the bank:

$$\mathcal{L}_{\text{unsup}}(\phi(G), \mathcal{M}) = \min_{\mathbf{m} \in \mathcal{M}} ||\phi(G) - \mathbf{m}||^2 \tag{40}$$

$$\mathcal{R}(\Theta) = \sum_l ||\Theta^{(l)}||_2^2 \tag{41}$$

λ_1, λ_2 are hyperparameters controlling the trade-off between supervision, manifold consistency, and regularization. Θ represent the adaptive parameters optimized via quantum gradient flows.

$$\Theta^{(t+1)} = \Theta^{(t)} - \eta \nabla_\Theta \mathcal{L}\left(\mathcal{N}\left(e^{-iH_{eff}(\Theta)} \rho_{in} e^{iH_{eff}(\Theta)}\right)\right) \tag{42}$$

This should lead to a robust learning of normal and fraudulent transaction patterns across quantum and classical domains.

After the training we define S_{normal} using a validation set W composed by M non fraudolent graphs as

$$S_{\text{normal}} = \{\phi(G^*) \mid G^* \in W\} \tag{43}$$

S_{normal} will not be modified during the inference phase.

2.5 Decision and Interpretability

The final decision is made based on the distance from the learned normal pattern S_{normal}.

$$s(G) = \min_{\phi(G^*) \in S_{\text{normal}}} ||\phi(G) - \phi(G^*)||_2^2 \tag{44}$$

The decision rule is

$$\text{Fraud}(G) \doteq \begin{cases} 1 & \text{if } s(G) > \tau \\ 0 & \text{otherwise} \end{cases} \tag{45}$$

Where τ is a threshold that can be tuned based on the desired trade-off between false positives and false negatives. My implementation is slightly different from the one described in the original paper since I used a different definition for S_{normal}.

3 Results and implementation notes

The algorithm was implemented in Python using *PennyLane* and *Pytorch*. It was tested on the *PaySim* dataset "PS_20174392719_1491204439457_log.csv" as suggested in the original paper.

3.1 Graph sampling

After building the transaction graf I ran an analysis on it and I found out that it's a very sparse graph mostly composed by nodes of degree 1. The analysis was done in with the script "calculate_density.py" and these are the results

```
    --- Summary ---
    Max Degree:            113
    Nodes with Degree = 1: 8,604,623 (94.83%)
    Nodes with Degree >= 2:469,277 (5.17%)
```

This strongly negativly affects the performance of our algorithm. Since our goal is to use topological features to identify frauds having loats of subgraphs, both "fraudolent" and "normal", with the same simple structure makes it impossible to correctly classify them with our algorithm.

To try and overcome this problem, before running the main algorithm, I run a small script "find_pool_of_components.py" to find the biggest weakly connected components. This turned out to be usefull but not as much as I thought because, even in those rare components, loops were extreamly rare. This, as we will show in the final results, compleatly invalidates the purpose of the Betti numbers.

In the main algorithm "main.py" the sampling is done using a random-walk starting from the Hubs (Nodes with a high degree) of the biggest connected components found befere. The algorithm was implmented as described in the previous section with the following hyperparameters.

```
TRAINING_DIM = 700 # Number of graphs for training
VALIDATION_DIM = 10 # Number of graphs for validation (S_normal)
TEST_DIM = 100 # Number of graphs for testing
FRAUD_RATIO = 0.3 # Target percentage of Fraud Graphs (0.3 = 30%)
SAMPLED_EDGES = 8 # Minimum edges in sampled subgraphs

TOTAL_GRAPHS = TRAINING_DIM + VALIDATION_DIM + TEST_DIM
NUM_NODES = 16
NUM_QUBITS = int(np.log2(NUM_NODES))
TOTAL_QUBITS = 2 * NUM_QUBITS

BATCH_SIZE = 16

NUM_LAYERS = 3
```

```
N_BETTI = 3
LATENT_DIM = 2 + (N_BETTI + 1) * 3

K_MEMORY_BANK = 10
BANK_PERCENTAGE_UPDATE = 0.1
WARMUP_STEPS = 5

EPOCHS = 20
LEARNING_RATE = 0.005

LAMBDA_1 = 0.1 #unsup lambda
LAMBDA_2 = 0.01 #regularization lambda

# Probabilieties for the Kraus matrixes
THETA_0_INIT = 1.0
THETA_1_INIT = -5.0

TAU_THRESHOLD = 0.4  #threshold used in the final classification
```

As we espected from the previous considerations the result are the following:

```
    Final Accuracy: 70.0%
    Confusion Matrix: TP=3, FP=0, TN=84, FN=13
```

What conforts me is that the network is capturing the topological features correctly, albeit the results aren't great. In fact almost all the subgraphs have the following value for $\Phi(G)$ due to the identical topological structure.

```
Graph 117 | NORMAL | Score: 0.000000 | Pred: 0
Phi(117) = tensor([ 0.1628,  2.1829,  1.0000,  0.0000,  0.0000,  1.0000,  1.0000,  0.0000,
         0.0000,  1.0000,  0.0625,  1.0625,  0.0000, -1.0000])
Graph 002 | FRAUD  | Score: 0.000009 | Pred: 0
Phi(002) = tensor([ 0.1641,  2.1785,  1.0000,  0.0000,  0.0000,  1.0000,  1.0000,  0.0000,
         0.0000,  1.0000,  0.0625,  1.0625,  0.0000, -1.0000])
```

What's conforting me even more is that, the frauds that are being correctly recognised actually belong to graphs that have a slightly different topology.

```
Graph 068 | FRAUD  | Score: 0.406291 | Pred: 1
Phi(068) = tensor([ 0.1571,  2.1855,  1.0000,  0.0000,  0.0000,  1.0000,  1.0000,  0.0000,
         0.0000,  1.0000,  0.1875,  0.6875,  0.0000, -0.5000])
Graph 016 | FRAUD  | Score: 2.115981 | Pred: 1
Phi(016) = tensor([0.1239, 2.3273, 1.0000, 1.0000, 0.0000, 1.0000, 1.0000, 0.0000, 0.0000,
        1.0000, 0.3125, 0.1875, 0.0000, 0.1250])
```

That beeing said, a further deterministic analysis confirmed that the "PaySim" dataset has very few cycles and its subgraphs share a very similar topology. It's important to remember that "PaySim" is a syntethic dataset that might not represent the actual reality. The authors of the original paper said they trained their algorithm on a dataset that was like PaySim. My conclusion is that almost centrality it wasn't the classical PaySim or at least not only it.

4 Conclusions

Despite my implementation of the algorithm failed to precisly detect frauds in the PaySim dataset the results showed that the proposed architecture correctly estimates the Betti numbers and the Euler's number of the graphs.

Unfortunately the authors of the original paper have not published their code yet. For now what I can conlude is that this results prove that the algorithm has some potential and it could work in a real life scenario where the transaction graphs have different and complicated topological structures.

A A previous implementation of S_{normal}

My first implementation of S_{normal} tourned out to be ineffective; the assumption of using an hypersphere to model normal transactions was too restrictive and the model struggled to generalize to unseen fraud patterns. I mention it here for completeness.

The idea was to compute S_{normal} as the mean of the feature vectors of all the legitimate graphs in the current batch. This technique is inspired by the paper "*Deep One-Class Classification*" [9]. This changes the network the network to map all "normal" data points into a hypersphere of minimum volume surrounding a center c. By doing so the model should learn common factors of variation in normal transactions, making it better at detecting "unknown" types of fraud that fall far from that center.

The new definition of S_{normal} becomes

$$S^{(0)}_{\text{normal}} = \mu_i \doteq \frac{1}{|\mathcal{G}^{(i)}_{\text{normal}}|} \sum_{G \in \mathcal{G}^{(i)}_{\text{normal}}} \phi(G) \tag{46}$$

$$S^{(i)}_{normal} = \alpha \cdot S^{(i-1)}_{normal} + (1-\alpha) \cdot \mu_i \tag{47}$$

where $\mathcal{G}^{(i)}_{\text{normal}}$ is the set of legitimate graphs in the i-th batch and $\alpha \in [0,1]$ is a hyperparameter. If I would have followd the implementation described in the paper "*Deep One-Class Classification*" [9] S_{normal} would have been computed as the mean of the normal sample in the batch; our implementation compute it as a moving average over batches as this idea is proven to improve robustness [11].

The anomaly detection is still formulated as an hypothesis test

$$\mathcal{H}_0 : ||\phi(G) - S^{(i)}_{\text{normal}}||^2 \leq R^2 \quad \text{(Normal)} \tag{48}$$

$$\mathcal{H}_1 : ||\phi(G) - S^{(i)}_{\text{normal}}||^2 > R^2 \quad \text{(Anomaly)} \tag{49}$$

The loss function is defined as

$$\mathcal{L} = \mathcal{L}_{sup}(\phi(G), y; \Theta) + \lambda_1 \mathcal{L}_{\text{unsup}}(\phi(G), \Theta) + \lambda_2 \mathcal{R}(\Theta) \tag{50}$$

where

$$\mathcal{L}_{sup}(\phi(G), y; \Theta) = -\left[y \log(\hat{y}) + (1-y)\log(1-\hat{y})\right] \tag{51}$$

where $\hat{y} = \sigma(W^\top \phi(G) + b)$ is the predicted probability of being an anomaly, with σ being the sigmoid function, W and b are learnable parameters, and $y \in \{0,1\}$ is the ground truth label (0 for normal, 1 for anomaly).

$$\mathcal{L}_{\text{unsup}}(\phi(G), \Theta) = ||\phi(G) - S^{(i)}_{\text{normal}}||^2 \tag{52}$$

$$\mathcal{R}(\Theta) = \sum_l ||\Theta^{(l)}||_2^2 \tag{53}$$

λ_1, λ_2 are hyperparameters that control the relative importance of the unsupervised, and regularization terms respectively. Θ are adaptive parameters optimized via quantum gradient flows

References

[1] S. Alarfaj F. K. Shahzadi. "Enhancing fraud detection in banking with deep learning: Graph neural networks and autoencoders for real-time credit card fraud prevention". In: *Journal of Financial Data Science* (2024).

[2] Mohammad Doost and Mohammad Manthouri. “Quantum topological graph neural networks for detecting complex fraud patterns”. In: *arXiv preprint arXiv:2512.03696* (2025). URL: https://arxiv.org/abs/2512.03696v1.

[3] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. “A Quantum Approximate Optimization Algorithm”. In: (2014). arXiv: 1411.4028 [quant-ph]. URL: https://arxiv.org/abs/1411.4028.

[4] Hdaib M. and L. Rajasegarar S. Pan. “Quantum deep learning-based anomaly detection for enhanced network security”. In: *Quantum Machine Intelligence* 6.1 (2024).

[5] M. J. Bastian N. D. Monkam G. F. De Lucia. “A topological data analysis approach for detecting data poisoning attacks against machine learning based network intrusion detection systems”. In: *Computers Security* 144 (2024).

[6] B. Motie S. Raahemi. “Financial fraud detection using graph neural networks: A systematic review”. In: *Expert Systems with Applications* 240 (2024).

[7] María Laura Olivera-Atencio, Lucas Lamata, and Jesús Casado-Pascual. “Impact of Amplitude and Phase Damping Noise on Quantum Reinforcement Learning: Challenges and Opportunities”. In: (2025). arXiv: 2503.24069 [quant-ph]. URL: https://arxiv.org/abs/2503.24069.

[8] T. Pourhabibi et al. “Fraud detection: A systematic literature review of graph-based anomaly detection approaches”. In: *Decision Support Systems* 133 (2020), p. 113303.

[9] Lukas Ruff et al. “Deep One-Class Classification”. In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, Oct. 2018, pp. 4393–4402. URL: https://proceedings.mlr.press/v80/ruff18a.html.

[10] Y. Umeda, J. Kaneko, and H. Kikuchi. “Topological data analysis and its application to time-series data analysis”. In: *Fujitsu Scientific and Technical Journal* 55 (2019), pp. 65–71.

[11] Xingbao Zhang, Wei Li, and Yue Zhao. “One-class Anomaly Detection with Redundancy Reduction and Momentum Mechanism”. In: (2022), pp. 1–6. DOI: 10.1109/DOCS55193.2022.9967719.