Financial Fraud Detection using Quantum Graph Neural Networks

Nouhaila Innan[1,2]‡, Abhishek Sawaika[3], Ashim Dhor[4], Siddhant Dutta[5], Sairupa Thota[6], Husayn Gokal[7], Nandan Patel[8], Muhammad Al-Zafar Khan[2,9]§, Ioannis Theodonis[10] and Mohamed Bennai[1]

[1] Quantum Physics and Magnetism Team, LPMC, Faculty of Sciences Ben M'sick, Hassan II University of Casablanca, Morocco
[2] Quantum Formalism Fellow, Zaiku Group Ltd, Liverpool, United Kingdom
[3] Indian Institute of Technology, Bombay, India
[4] Department of Data Science and Engineering, Indian Institute of Science Education and Research Bhopal, India
[5] Dwarkadas J. Sanghvi College of Engineering
[6] Guru Nanak Institutions Technical Campus, Hyderabad, India
[7] ASTI Academy, Dubai, UAE
[8] Vellore Institute of Technology, Chennai, India
[9] Robotics, Autonomous Intelligence, and Learning Laboratory (RAIL), School of Computer Science and Applied Mathematics, University of the Witwatersrand, 1 Jan Smuts Ave, Braamfontein, Johannesburg 2000, Gauteng, South Africa
[10] Department of Physics, School of Applied Mathematical and Physical Sciences, National Technical University of Athens, Greece.

Abstract. Financial fraud detection is essential for preventing significant financial losses and maintaining the reputation of financial institutions. However, conventional methods of detecting financial fraud have limited effectiveness, necessitating the need for new approaches to improve detection rates. In this paper, we propose a novel approach for detecting financial fraud using Quantum Graph Neural Networks (QGNNs). QGNNs are a type of neural network that can process graph-structured data and leverage the power of Quantum Computing (QC) to perform computations more efficiently than classical neural networks. Our approach uses Variational Quantum Circuits (VQC) to enhance the performance of the QGNN. In order to evaluate the efficiency of our proposed method, we compared the performance of QGNNs to Classical Graph Neural Networks using a real-world financial fraud detection dataset. The results of our experiments showed that QGNNs achieved an AUC of 0.85, which outperformed classical GNNs. Our research highlights the potential of QGNNs and suggests that QGNNs are a promising new approach for improving financial fraud detection.

Keywords: Quantum Machine Learning, Quantum Graph Neural Network, Variational Quantum Circuit, Fraud Detection

‡ nouhaila.innan-etu@etu.univh2c.ma
§ muhammadalzafark@gmail.com

1. Introduction

In today's dynamic technological landscape, the rapid surge in digital transactions and financial activities has ushered in unprecedented conveniences. However, this digital transformation has also fostered a parallel escalation in financial fraud, posing significant threats to global industries and economies. As financial operations increasingly adopt digital complexities, the susceptibility to exploitation from malicious actors increases, leading to the pursuit of innovative fraud detection methods [1].

The importance of fraud detection extends beyond a single industry, resonating across diverse sectors, including Banking, e-Commerce, Healthcare, FinTech, and Insurance. The integrity of financial transactions is pivotal for optimal functionality in these domains, making robust fraud detection systems indispensable. The banking sector, exemplifying the forefront of this challenge, confronts expanding attack surfaces due to rising online and mobile banking adoption [2]. Conventional rule-based approaches, while partially effective, struggle to match the ever-evolving tactics of fraudsters. This necessitates advanced, adaptable techniques in order to keep pace with the dynamic landscape of financial fraud [3].

The research community is constantly exploring new ways to detect fraud, as the methods used by fraudsters become increasingly sophisticated. An emerging avenue combines Quantum Machine Learning (QML), and Graph Neural Networks (GNNs); a groundbreaking approach poised to revolutionise financial fraud detection [4]. By harnessing the distinct attributes of Quantum Computing (QC) — superposition and entanglement — and integrating them with the effectiveness of GNNs, this hybrid paradigm endeavours to tackle the intricate and ever-changing realm of financial fraud. In this paper, we investigate the challenges of financial fraud detection and the potential of Quantum Graph Neural Networks (QGNNs) as a transformative approach to address these challenges. We conduct comprehensive analyses and experiments to demonstrate the efficacy and viability of QGNNs for fraud detection. Our findings contribute to the discourse on advanced fraud detection techniques and pave the way for enhanced security in the financial ecosystem.

QML emerges at the intersection of QC and conventional Machine Learning (ML) techniques, offering promising avenues for solving complex problems in various domains. In the context of financial fraud detection, QML presents an intriguing approach that harnesses the inherent power of quantum computations to address the intricate challenges posed by fraudulent activities within the financial sector. By leveraging QGNNs, the combination of quantum and graphical techniques offers a novel framework for analysing and classifying financial transactions.

In financial fraud detection, QML encodes transactional data into quantum states. This can improve the efficiency and accuracy of fraud detection by enabling efficient parallel processing and enhanced feature representation. The unique property of quantum entanglement facilitates the capture of intricate relationships among transactional features. This fosters a more comprehensive understanding of fraudulent patterns.

However, this emerging paradigm also poses significant challenges, including the interpretability of quantum computations and concerns over quantum system security. Nonetheless, the promising outcomes showcased in this study highlight the potential of QML to significantly impact fraud detection by improving accuracy and scalability, opening the door to a new era of enhanced financial security. The inherent ability of quantum systems to process and model complex relationships among variables aligns well with the intricacies of financial transactions.

In the realm of fraud detection, conventional methods often struggle to navigate the intricate tapestry of relationships and intricate patterns that underlie fraudulent activities. As the complexity and diversity of fraudulent schemes continue to evolve, the demand for adaptable and nuanced detection techniques becomes increasingly apparent. In response, graph-based methods have emerged as a promising avenue, harnessing the intrinsic strength of graph structures to capture and dissect the multifaceted interconnections that define fraudulent behaviours.

Built upon the foundation of the mathematical study of graph theory, these methods provide an intuitive and comprehensive framework to model, depict, and analyse the labyrinthine-like relationships among entities. From individuals and accounts to transactions and organisations, a broad spectrum of entities can be seamlessly encapsulated within interconnected nodes and edges. This unique arrangement empowers graph structures to illustrate the intricate dynamics inherent in fraudulent schemes vividly. Diverging from conventional approaches, which rely on isolated data points and predetermined rules, graph-based methods can unveil concealed connections and dependencies, thereby unveiling anomalous or suspicious behaviours that might go unnoticed [5].

The true effectiveness of graph structures in fraud detection lies in their unparalleled ability to uncover latent patterns that elude customary detection techniques. Fraudulent endeavours rarely occur in isolation; rather, they unravel through intertwined actions involving multiple entities. These connections, often obscured by noise and obfuscation, are revealed through meticulous graph analysis. By representing and exploring these relationships, graph-based methods empower fraud detection systems to transcend the confines of isolated data points, capturing the holistic context in which fraudulent activities manifest and thrive.

This quantum potential extends seamlessly to the detection of financial fraud, where quantum-inspired techniques can transform the landscape. By integrating the strengths of quantum mechanics, QC could potentially revolutionise classical ML algorithms used for fraud detection. Techniques such as Quantum Neural Networks (QNNs) can harness the property of quantum parallelism to process and analyse financial data more efficiently. Exploiting superposition, these networks can simultaneously examine multiple data points, accelerating tasks like pattern recognition and data classification, crucial elements in detecting fraudulent activities. Furthermore, quantum feature mapping enhances the ability to identify intricate patterns in financial transactions that might evade classical methods. This intersection of QML and financial fraud detection promises a future where quantum-inspired techniques amplify the accuracy and efficiency of identifying and combating fraudulent behaviours.

The integration of quantum properties into graph-based fraud detection methodologies is motivated by the imperative to confront the escalating sophistication of fraudulent activities and the inherent limitations of classical computing paradigms. In today's landscape of intricate financial systems, fraudulent behaviours manifest intricate and interwoven patterns that defy conventional computational approaches. These conventional methods struggle to unravel the complexities inherent in fraud networks, marked by exponential data growth and non-linear dynamics. As a result, a critical need arises for innovative techniques that transcend classical computing boundaries, leading us to explore the distinctive attributes of QC.

While graph-based fraud detection methods have shown their ability to capture the intricate relationships within fraudulent activities, they face limitations in handling the explosive data growth and complex dynamics of fraud networks [6]. The conventional algorithms grounded in classical computing struggle to navigate the intricate patterns that define these networks, often characterised by non-linear structures and extensive solution spaces. This disparity underscores a compelling necessity for pioneering methodologies capable of transcending the confines of classical computation. By delving into quantum properties, we seek to bridge this gap, illuminating concealed fraud schemes and redefining the boundaries of fraud detection in the context of modern financial systems [7].

The primary objective of this study is to conduct a comprehensive investigation into the effectiveness of QGNNs for financial fraud detection. Specifically, we aim to comprehensively assess and analyse the potential advantages and enhancements that QGNNs offer in contrast to classical GNNs, specifically concerning their ability to identify and mitigate fraudulent activities within intricate networks of financial transactions. By meticulously evaluating and comparing the performance, adaptability, and overall effectiveness of QGNNs benchmarked against classical GNNs, we intend to shed light on the transformative possibilities of leveraging QC for refining fraud detection methodologies.

Through this research endeavour, we aspire to comprehensively understand QGNNs potential to enhance the landscape of fraud detection strategies [8]. By closely examining the strengths and constraints of QGNNs *vis-á-vis* classical GNNs, our study seeks to contribute meaningful insights to the QC and fraud detection disciplines.

The structure of this paper is as follows:

- Sec. 2, we present a comprehensive literature review on financial fraud detection, graph neural networks, and QML. We highlight the fundamental concepts of GNNs and emphasise the limitations of conventional methods.
- In Sec. 3, we examine the theory of GNNs and different models, focusing on `GraphSAGE` architecture.
- In Sec. 4, we conduct a comprehensive study of our proposed quantum graph neural network model for financial fraud detection.
- In Sec. 5, we provide a comprehensive understanding of our selected dataset characteristics, including attribute highlighting, class imbalance, and preprocessing steps.
- In Sec. 6, we evaluate the performance of QGNN and GNN on a real-world financial fraud dataset and discuss the results of our experiments.
- In Sec. 7, we summarise the key findings of our study and suggest directions for future research.

2. Literature Review

Transitioning our inquiry from the domain of conventional fraud detection techniques, we direct our attention towards the emerging promise of QGNNs. This transition represents a notable departure from established methodologies and carries the potential to reshape our understanding and implementation of fraud detection strategies. In adopting this advanced technological paradigm, we aim to uncover novel insights and methodologies that may prove instrumental in addressing the complexities of contemporary fraud detection challenges.

In the classical computing area, there exists a plethora of noteworthy research. We highlight some of the outstanding application-based research works below.

Ma *et al.*, 2020 [9] provide an in-depth exploration of using Deep Learning (DL) techniques to detect anomalies in graph data. It emphasises the importance of anomaly detection in domains such as security, finance, and medicine. The paper highlights how conventional methods struggle with graph data complexities, leading to the rise of DL approaches. The authors systematically categorise and analyse contemporary techniques for graph anomaly detection based on their ability to identify different types of anomalous graph objects. The survey underscores the challenges inherent in this field and identifies future research directions.

In today's rapidly evolving technological landscape, we are witnessing a growing variety of ways in which fraudulent transactions take place. This makes it crucial to understand how we detect fraud and to determine which approach works best to get accurate results. Researchers are on a widespread quest to explore different QML models that fit well for spotting fraud.

In [10], Ahmad *et al.*, 2021 explore the combination of QC and Deep Neural Networks (DNNs) to address network security through anomaly detection. Emphasising the transformative potential of QC, it develops a DNN-based quantum autoencoder to process quantum-encoded data. Applying dimensionality reduction techniques, the research enhances the model's ability to detect subtle anomalies within network data. The results reveal impressive accuracy rates of 100% during training and 99% during testing, demonstrating the framework's efficacy in capturing patterns and its potential to revolutionize anomaly detection in network security. This innovative research establishes a symbiotic relationship between QC and DL, offering a novel avenue for cybersecurity advancements.

In [11], Zheng *et al.*, 2021 introduce a Quantum Graph Convolutional Neural Network (QGCNN) model to address the challenge of applying a QNN to non-Euclidean spatial data, particularly graph-structured data. The QGCNN model efficiently captures graph topology and node features using quantum parametric circuits, making graph-level classification tasks effective. The study bridges the gap in QML research by extending QNNs to handle irregular spatial structures in graph data. The proposed QGCNN model showcases the potential of QC in enhancing graph-based ML tasks, contributing to the advancement of QML research.

QGNNs offer a transformative approach to fraud detection in the financial sector, addressing the escalating complexity of fraudulent schemes. By leveraging their capacity for intricate pattern recognition, enhanced data representation, and adaptability to evolving strategies, these networks hold the potential to significantly enhance the accuracy of fraud detection while mitigating false positives.

The study by Kyriienko *et al.*, 2022 [12] focuses on utilising QML to enhance the detection of internal fraud in the financial sector, particularly fraudulent activities conducted by individuals within an organisation. The study employs quantum kernels generated through quantum feature maps based on Instantaneous Quantum Polynomial (IQP) circuits to transform classical data from credit card transactions into a higher-dimensional space for analysis. Applying the one-class support vector machine (OC-SVM) algorithm within this quantum feature space enables to detection of abnormal transactions. Simulations confirm the effectiveness of this quantum kernel-based approach in detecting internal fraud, highlighting the potential of QC to improve fraud detection accuracy in complex financial data.

Recent research efforts, such as [13], Innan *et al.*, 2023 investigate the utility of QML models in enhancing fraud detection within the financial sector. The authors address the pressing need for accurate fraud detection and propose leveraging QML techniques to improve system efficiency. Through a comparative analysis of four QML models — Quantum Support Vector Classifier, Variational Quantum Classifier, Estimator Quantum Neural Network, and Sampler Quantum Neural Network — the study assesses their performances in detecting financial fraud. Employing a synthetic dataset derived from BankSim, the authors employ data preprocessing, logical analysis, and visualisations to identify key discriminatory features. The QSVC emerges as the most effective model, yielding high F1 scores, exceeding 98% for both fraud and non-fraud classes. The study highlights the potential of QML in fraud detection, providing insights into its capabilities and implications while acknowledging areas for future advancement.

In [14], Xiang *et al.*, 2023 address the challenge of credit card fraud detection by introducing a novel semi-supervised approach utilising attribute-driven graph representation and a Gated Temporal Attention Network (GTAN). The proposed method effectively enhances fraud detection performance, especially with limited labelled data, according to the study's key findings. The authors constructed a temporal-transaction graph based on transaction records, incorporating temporal transactions and interactions, and utilised GTAN for message passing and learning transaction representations. Risk patterns are further modelled through risk propagation amongst transactions. The research showcases that GTAN outperforms state-of-the-art baselines in detecting fraud across multiple datasets, demonstrating its ability to perform well with only a small fraction of labelled data. The study contributes a comprehensive framework that amalgamates attribute-driven embeddings, temporal attention mechanisms, and risk propagation to advance fraud detection methodologies, offering promising avenues for future research and innovation in the field.

In [15], Schetakis *et al.*, 2023 explore the application of fraud detection for small and medium-sized enterprises. The study employs a hybrid approach, comparing quantum classifiers with classical Artificial Neural Networks (ANN), demonstrating more efficient training and potential advantages in credit scoring. The study focuses on a classical-quantum hybrid model's performance using quantum classifiers and classical ANNs, revealing significant efficiency improvements with promising outcomes in credit scoring for SMEs.

In [16], Beer *et al.*, 2023 delve into the realm of QML by investigating the use of quantum neural networks for processing graph-structured quantum data. Addressing challenges and opportunities in the Noisy Intermediate-Scale Quantum (NISQ) devices era, the study introduces a novel concept of associating quantum states with graph vertices and correlations with edges. The paper proposes using modified loss functions and quantum training algorithms to learn and characterise complex graph structures with QNNs. The research contributes to the taxonomy of QML approaches, emphasising the significance of correlations and graph structure in advancing quantum information processing capabilities for various applications.

3. Classical Graph Neural Networks

In this section, we present the theoretical foundations of classical graph neural networks, focusing on the type used in this study. Firstly, we briefly describe the mechanics of ANNs and thereafter describe the GNNs.

Artificial Neural Networks are a class of ML algorithms whose structure is derived from the workings of the Biological neurons in sentient beings. Analogous to how the neurological systems of these beings transmit information from one neuron to another, the ANN consists of different layers of neurons that transmit information. Below, we clarify further.

- **Neurons:** These are the basic units of computation in the ANN. Each neuron consists of an input, hidden, and output layer.
  - The input layers ingest the initial data.
  - The hidden layer transforms the data received by the input layer and performs certain computations.
  - The output layer then receives the transformed data from the hidden layer and returns the predictive result.
- **Weights and Biases:** Each connection between neurons has an associated weight, $w_{ij} \in [0, 1]$, between nodes i and j respectively, which indicates the strength of connections between neurons and is learned and updated during the training process. The actual interpretation of weights is different for different types of ANNs. In an ANN, a bias term is added to the weighted sum of the inputs to a neuron before the activation function is applied. The bias term allows the neuron to learn a non-linear decision boundary, which is necessary for ANNs to learn complex functions.

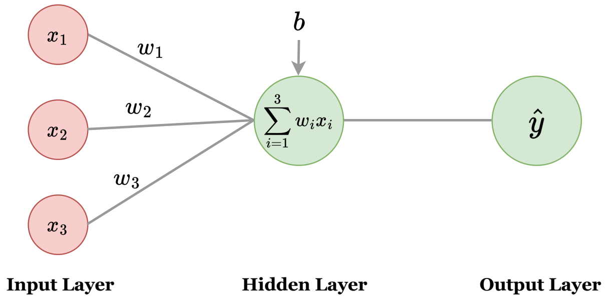

Figure 1. The basic structure of an ANN shows the weights and bias along with the respective layers.

In Fig. 1, the inputs are represented by x_1, x_2, x_3, and the weights are represented by w_1, w_2, w_3 shown in a summation form in the second layer. In terms of the bias, b, the output layer is expressed as

$$\hat{y} = \sum_i w_i x_i + b = w_1 x_1 + w_2 x_2 + w_3 x_3 + b. \quad (1)$$

- **Activation Functions:** The ANNs consist of an activation function that transforms the given output non-linearly. Several examples of activation functions include the sigmoid, the ReLU and its variations, the hyperbolic tangent, the swish, etc., as shown in Fig. 2. The choice of each of these functions is an artefact of the problem at hand that one is trying to solve, and each has its advantages and disadvantages.

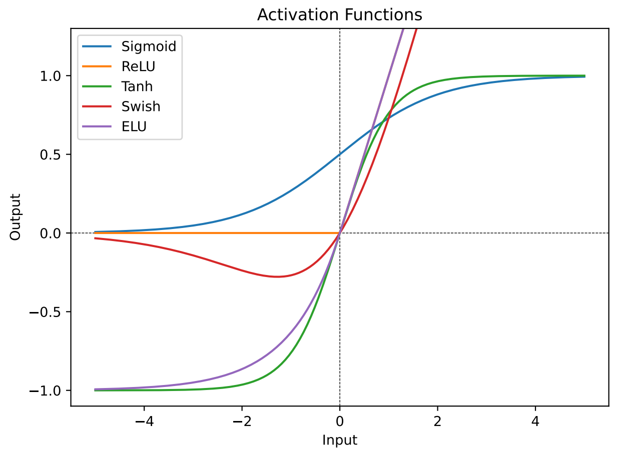

Figure 2. Plot showing the ReLU, Sigmoid, and Tanh activation functions along with vertical and horizontal lines at $x = 0$ and $y = 0$, respectively.

- **Final Output:** The final layer of a basic ANN is the output layer, which yields the output of the computations from the hidden layers.

As we discussed, artificial neural networks are ML models that can be used to learn from data. However, ANNs cannot capture the relationships between entities in a sequence or a grid, as they cannot consider the complex relationships between entities. To solve this problem, we use the where graph neural networks.

GNNs are a type of ANN specifically designed to process data structured as a graph. A graph is a group of nodes and edges representing relationships between them. The nodes in a GNN can represent entities such as people, objects, or places, and the edges can represent relationships such as friendships, kinship, or spatial proximity.

GNNs can process graph-structured data by iteratively aggregating information from neighbouring nodes. This allows GNNs to learn the complex relationships between entities in a graph, making them a powerful tool for node classification, link prediction, and graph classification tasks.

To express the edges and nodes of a graph as inputs to a neuron in an ANN, we first construct an adjacency matrix **A**. It is, essentially, a matrix of dimensions $n \times n$ where n is the number of nodes in a graph. The entries in the matrix are 1 if an edge exists between the nodes and 0 if there is no edge. Mathematically for each $A_{ij} \in \mathbf{A}$, and $1 \leq i, j \leq n$, we have that

$$A_{ij} = \begin{cases} 1, \text{if there exists an edge between nodes } i \text{ and } j, \\ 0, \text{otherwise.} \end{cases} \quad (2)$$

The adjacency matrix approach has several advantages: If the graph is undirected, the adjacency matrix

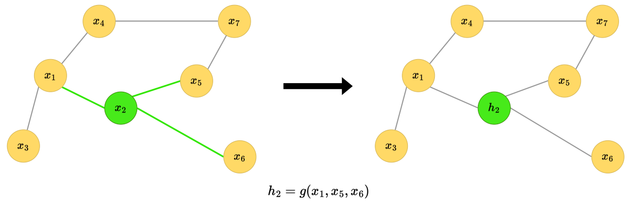

Figure 4. A graphical representation showing the dependency of the feature of nodes on the features of its neighbouring nodes.

becomes symmetric and has several interesting properties, including the diagonals representing self-loop vertices, the degree of the graph can be inferred from the sum of any row or column, the number of edges in the graphs (the number of connections required in the ANN) is just half the count of the number of nonzero entries, and properties relating the eigenvalues of **A**.

In Tab. 1, we observe no self-loops in the graph structure; thus, the diagonal entries of the adjacency matrix will be 0, This is illustrated in Fig. 3, which shows a directed graph with no self-loops.

Table 1. Adjacency matrix representing directed edges between nodes A, B, C, D, and E.

	A	B	C	D	E
A	0	1	0	0	0
B	0	0	1	1	0
C	0	0	0	0	1
D	1	0	0	0	0
E	0	0	0	1	0

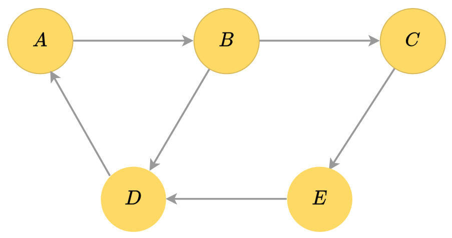

Figure 3. A directed graph. The arrows indicate the direction of flow. The graph consists of nodes labelled A, B, C, D, and E.

Below, we elaborate on the critical components of GNNs.

- **Graph Convolution** This is the key concept underpinning GNN architectures. Graph convolution predicts the features of the node in the next layer as a function of the neighbours' features. It transforms the node's features x_i in a latent space h_i that can be used for various reasons, as shown in Fig. 4.
- **Node classification** For a GNN, the objective is firstly to generate latent, from which we would

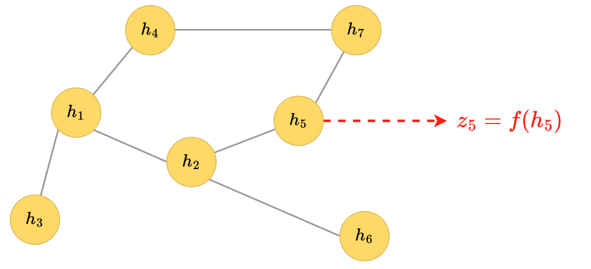

Figure 5. Graphical representation illustrating node classification. Nodes are associated with expressions $Z_i = f(h_i)$, where i ranges from 1 to 7, capturing relationships between features Z_i and hidden features h_i.

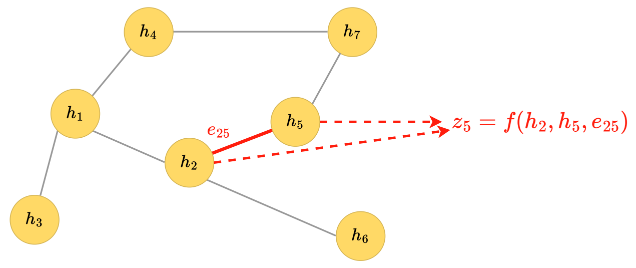

Figure 6. Graphical representation showcasing edge classification

then be able to work on a wide variety of standard tasks. Mathematically, we have that

$$Z_i = f(h_i), \tag{3}$$

where h_i is the feature vector associated with node i, and it can also be represented in the Fig. 5.

- **Edge Classification:** Similarly, we can use it to classify edges based on their features. To accomplish this, we generally need both the adjacent node vectors and the edge features, if they exist. Mathematically, we have that

$$Z_{ij} = f(h_i, h_j, e_{ij}), \tag{4}$$

where h_i, h_j are the feature vectors associated with nodes i and j respectively, and e_{ij} is the edge connecting nodes i and j as shown in Fig. 6.

3.1. Types of Graph Neural Networks

There are a variety of graph neural networks. Below, we focus on some of the important types of GNNs:

3.1.1. Feed-forward GNNs: Also known as "Deep feedforward Networks" or "Multi-layer Perceptrons". In these model architectures, input data is propagated through a graph of neurons to produce output by applying the transfer function at the edge weights between each node as represented in Fig. 7. Feedforward graph neural networks are simplified

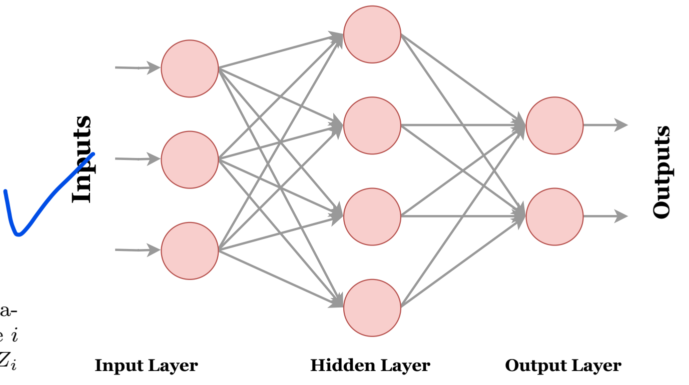

Figure 7. Basic structure of Feed-Forward GNNs. The network consists of an input layer with 3 nodes, a hidden layer with 4 nodes, and an output layer.

versions of GNNs that work straightforwardly. They process graph data in just one pass and update node information using neighbouring node data.

3.1.2. Graph Recurrent Neural Networks (GRNNs): GRNNs relay information from previous to neighbouring nodes. At each step, the GRNN uses a hidden state as a memory of what the network has encountered thus far in the sequence, as represented in Fig. 9. The network is updated with each new step based on the current input and the previous hidden state. When working with sequences, the RNN "unrolls" itself through time, treating each step as a separate layer of the network. This helps the GRNN handle sequences of different lengths. In [17], it was observed that Recurrent neural networks allow for loops and cycles, giving rise to dynamical systems and flexible behaviour in computation.

In [18], it has been observed that graph neural Networks are an evolution of Recurrent Neural Networks (RNNs) that address the limitations of RNNs by combining gated RNN architectures with graph convolutions.

3.1.3. Graph Convolutional Neural Networks (GCNNs): These architectures are one of the most popular and widely used GNNs in the literature, and they can be represented as shown in Fig. 8. The main property of GCNNs is their ability to maintain stability under stochastic perturbations. This means that GCNNs are less likely to be affected by noise or randomness in the data, which can be a major advantage in many applications. In [19], a new RES and a GFT model were employed over a GCNNs to test the robustness of the graph filter. GCNN is stable to stochastic perturbations with a factor proportional to the link loss probability. In particular, a wider and deeper GCNN decreases the stability while improving the performance,

(GCNNs) possess a notable characteristic of remaining stable and consistent even when exposed to random disturbances or variations in the input data

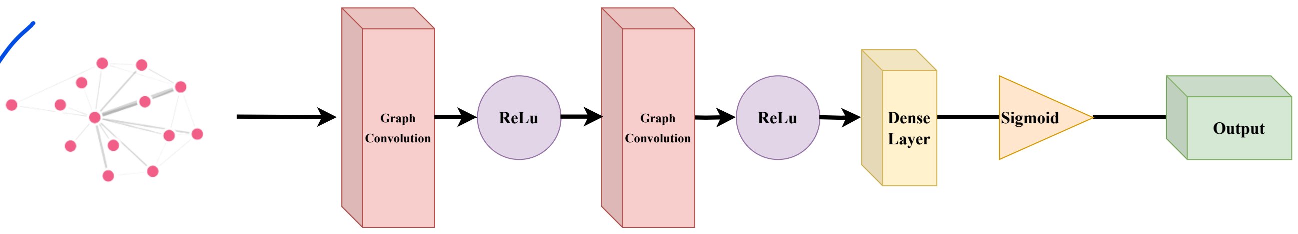

Figure 8. Architecture of Graph Convolutional Neural Network. The network includes a convolution layer followed by a ReLU activation function, a subsequent convolution layer with another ReLU activation function, an intermediate dense layer, and finally, a sigmoid activation function before the output layer.

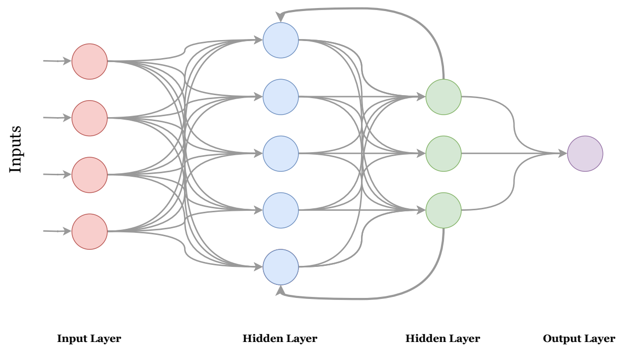

Figure 9. Structure of a Graph Recurrent Neural Network. The GRNN consists of an input layer with 4 nodes, a hidden layer with 5 nodes, and an additional layer with 3 nodes. The architecture is completed with an output layer.

indicating a trade-off between these two factors. In [20], the study leverages the stability property of GNNs to seek stable representations within a distribution.

3.1.4. `GraphSAGE` (Graph Sample and Aggregated) Networks: `GraphSAGE` is a node classification algorithm that samples and aggregates information from a node's local neighbourhood in a graph.

It improves over GCNNs by allowing nodes to aggregate information from a sampled set of neighbours, making `GraphSAGE` more scalable and generalisable to different graph structures.

In [21], Lo and *et al.* proposed `E-GraphSAGE`, a GNN-based intrusion detection system which outperforms state-of-the-art methods in evaluating network-based intrusion detection systems.

These networks use a message-passing scheme to update node representations by aggregating and transforming information from the neighbourhood. They have been shown to be effective in capturing both the edge features and topological information of a graph, making them suitable for analysing interconnected patterns in network flows. The modified `E-GraphSAGE` and `E-ResGAT` algorithms integrate residual learning into GNNs to address the challenge of class imbalance and improve performance in minority classes.

The `GraphSAGE` model architecture includes two graph convolution layers and uses the mean aggregation method to effectively capture node information from neighbours in the graph, as shown in Fig. 10.

For homogeneous graphs, the feature update rule is the mean aggregator, where the aggregation of features of the neighbours of the node v are given by

$$h^i_{N(v)} = \frac{1}{||N(v)||} D_p \left[h^{i-1}_u\right], \forall u \in N(v), \tag{5}$$

where the forward passes through layer k can be added in the future nodes and is given by

$$h^i_v = \sigma\left(\text{concat}\left[W^k_{\text{self}} D_p \left[h^{i-1}_v\right], W^i_{\text{neigh}} h^i_{N(v)}\right] + b^k\right). \tag{6}$$

In Eq.(5) $D_p[\ldots]$ is a random dropout with probability p applied to its argument vector, and b^k is an optional bias.

Eqs.(5) and (6) are the rules deciding how the features get updated in the GNN. The `GraphSAGE` algorithm was implemented on the dataset to perform predictions using `StellarGraph`. From the original graph, 30% of edges were removed to form a test set, 65% acted as the train set, and 5% acted as the validation set where the internal parameters were optimised over 10 epochs using the ADAM optimiser. The classifier was then applied to the test set; Scores were generated for the classifier's predictions and are provided in the Sec. 6. In the following, we describe the steps of the `GraphSAGE` algorithm.

The algorithm 1 first assigns unique node identifiers to each sample in the dataset. A graph is then constructed using the `StellarGraph` library, where nodes are identified by their unique identifiers. The `GraphSAGENodeGenerator` is then used to generate a stream of node samples and their corresponding labels. During the training phase, the `GraphSAGE` model is instantiated by specifying the sizes of the layers and activation functions. The model includes a dense output layer with a sigmoid activation function for generating predictions. After defining the input and output tensors, a `Keras` model is created

✓

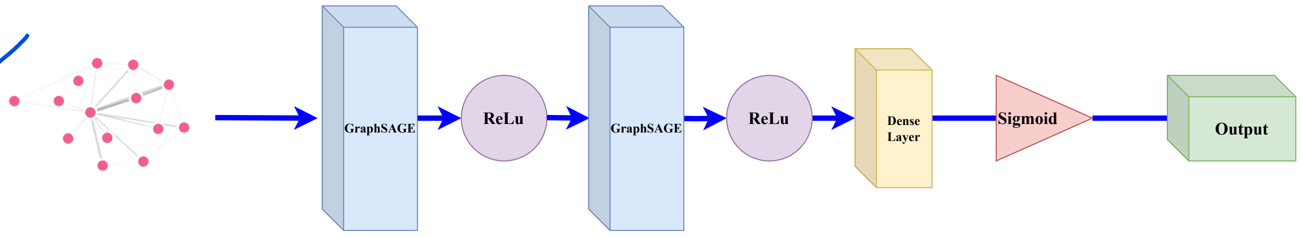

Figure 10. Architecture of the GraphSage. It comprises successive `GraphSAGE` layers, each followed by a ReLU activation function. An intermediate dense layer bridges the structure, culminating in a sigmoid activation function in the output layer.

Algorithm 1 `GraphSAGE`

```
Require: data (training, validation)
  for all data in training, validation do
    Assign unique node identifier to sample
  end for
  Create a StellarGraph $G$ using data with source column as "node_id"
  Initialise a GraphSAGENodeGenerator with graph $G$, batch size 5, and number of samples $[2, 32]$
  Extract node labels $y$ from data
  Generate a flow using the GraphSAGENodeGenerator and $y$ as targets
  if data is training then
    Initialise a GraphSAGE model with layer sizes $[128, 128]$ and activations ["relu", "relu"]
    Define input tensor and output tensor using the GraphSAGE model
    Add a Dense layer with 1 unit and "sigmoid" activation to obtain predictions
    Create a Keras model with input and output tensors
    Compile the model with "adam" optimiser, "binary_crossentropy" loss, and "accuracy" metric
    for epoch in 1 to 10 do
      Train the model using the flow and perform validation for 1 epoch
    end for
  end if
  if data is validation then
    Predict on the flow using the trained model.
    Calculate mode threshold and convert predictions to binary
    Calculate precision-recall curve points and AUC-PR using true labels and predicted scores
  end if
```

and compiled using the Adam optimiser, binary cross-entropy loss, and accuracy metric. The model is trained over a number of epochs using the generated data stream.

For non-training data, the trained model is used to predict labels. The predictions are then post-processed by calculating the mode threshold and converting the predicted scores to binary form. Precision-recall curves are then computed to evaluate the model's performance, and the area under the precision-recall curve (AUC-PR) is used to quantify its effectiveness.

4. Quantum Graph Neural Networks

Our research methodology draws inspiration from the pioneering work of Hu *et al.*, 2022 [22] in quantum graph neural network development. Our approach unfolds in the following manner:

- **Step 1: Graph Construction and Representation**
  Our initial step involves constructing the graph, where each graph corresponds to dataset transactions. This dataset encompasses 30 encoded attributes, predominantly processed via Principal Component Analysis (PCA) techniques, except for two attributes – transaction amount and time. In this framework, nodes represent encoded features. At the same time, edges symbolise their fusion to get novel graph configurations, and each graph in our feature grouping employs interconnected nodes, totalling 28 nodes per graph.
- **Step 2: Topological Data Analysis (TDA) for Graph Projection**
  Utilising topological data analysis, the graph is embedded into a one-dimensional scatter plot. This technique discriminates between graphs based on their distinct elements, facilitating differentiation between fraudulent and non-fraudulent cases. Following scatter plot creation, node clustering reduces nodes while encapsulating the initial 28 features, minimising quantum resource requirements. A final 28-dimensional feature vector representing the positional attributes is generated.
- **Step 3: Encoding within Quantum Circuit and Linear Transformation**
  Upon acquiring the graphical representation of each transaction, the subsequent step entails encoding it into a quantum state utilising the angle-encoding technique, employing a range of qubits from 1 to 28. To initiate, the encoding

process is applied to all node-level features ($N \times 28$), resulting in a dimensionally apt vector ($N \times q$) facilitated by a fully connected linear layer. This vector is subsequently channelled into the designated quantum circuit with q qubits as defined

$$|x\rangle = \bigotimes_{i=1}^{N} \cos(x_i)\,|0\rangle + i\sin(x_i)\,|1\rangle\,. \tag{7}$$

- **Step 4: Enhancing Node Features with Variational Quantum Circuits**
  Following the previous steps, a sequence of two-local ansatz constructs orchestrates the implementation of a multi-layered Variational Quantum Circuit (VQC) calibrated for the transactional dataset. Fig. 11 visually portrays a circuit encompassing a 6-qubit with the angle encoding alongside 2 concealed VQC layers. Notably, the ansatz capitalises on `Rx`, `Ry`, and `CNOT` operations while adopting `Rx` rotations for angle encoding, as explained in the 3rd step. Subsequently, circuit measurement culminates in using Pauli-`Z` expectation. This strategic circuit design is a conscientious outcome of considerations concerning circuit expressibility and intricate node-node interactions embedded within graphical data.
  We enhance node features using the VQC, emulating the operational principles of conventional classical graph neural networks with fully connected layers. VQC entails the successive application of variational layers. A single layer transformation L is depicted as

$$L : |\psi(x)\rangle \rightarrow |\psi(y)\rangle = U(w)\,|\psi(x)\rangle\,, \tag{8}$$

  where $U(w)$ represents the VQC with N layers, defined as

$$U(w) = T\bigotimes_{k=1}^{N} R_y(w_k)\,|\psi(x)\rangle\,. \tag{9}$$

- **Step 5: Integration of Unitary Gates and Classification**
  In order to meet the requirements of a quantum graph neural network, it is necessary to have unitary gates that act on two states simultaneously while preserving the others. As we explained, rotational gates are essential to integrate the graph into the quantum circuit. Once these gates are in place, a linear layer can classify the graph as fraudulent or non-fraudulent. The comprehensive procedure outlined in Fig. 12 sheds light on the architecture and operational aspects of the QGNN, highlighting the distinct components and their produced interplay in fraud detection. Additionally, Algorithm 4 provides a step-by-step algorithm description.

Algorithm 2 Quantum Graph Neural Networks

```
Inputs: Graph data (training, validation)
Initialise QGNN
for each epoch in training do
    for each graph batch in training data do
        Extract graph features and edges
        Apply angle encoding to the graph features
        Initialise Quantum Circuit for VQC
        Generate tunable parameters for VQC
        Update Quantum Circuit using inputs and VQC parameters
        Run the Quantum circuit (VQC) to obtain quantum output
        Perform average pooling for graph-level features
        Make predictions using a linear layer
        Compute loss and update parameters using Adam optimiser
    end for
    Evaluate QGNN on validation data
end for
Calculate optimal threshold for binary classification
Convert predictions to binary using the threshold
Calculate Precision-Recall curve and AUC-PR
Return Classification metrics, loss values
```

5. The Dataset

We used the publicly available credit card fraud detection dataset [23] for our analysis. This dataset contains credit card transactions of European cardholders with encrypted feature labels obtained from principal component analysis on actual user data. Each transaction is labelled as either fraud (1) or not fraud (0) and contains 28 encrypted features along with amount and time values. The dataset is highly imbalanced, with only 492 frauds out of 284 807 transactions. To address this imbalance, we used random undersampling on the non-fraud transactions.

For a comprehensive understanding of the dataset characteristics, we utilised visual representations. In Fig. 13, we present the correlation matrix ($v_1 - v_{28}$) of the dataset. The figure effectively illustrates the relationships between the features v_1 to v_{28} by displaying their correlation values. This information offers insights into how these features interact and influence one another.

Additionally, Fig. 14 offers histograms that vividly depict the distribution of all features in the dataset. These histograms visually represent the data distribution patterns, assisting us in identifying potential trends and outliers. Furthermore, our

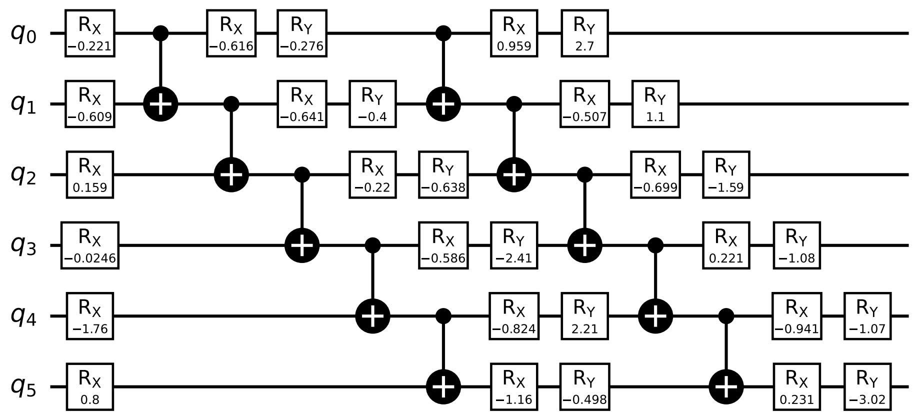
q_0 R_X −0.221 R_X −0.616 R_Y −0.276 R_X 0.959 R_Y 2.7
q_1 R_X −0.609 R_X −0.641 R_Y −0.4 R_X −0.507 R_Y 1.1
q_2 R_X 0.159 R_X −0.22 R_Y −0.638 R_X −0.699 R_Y −1.59
q_3 R_X −0.0246 R_X −0.586 R_Y −2.41 R_X 0.221 R_Y −1.08
q_4 R_X −1.76 R_X −0.824 R_Y 2.21 R_X −0.941 R_Y −1.07
q_5 R_X 0.8 R_X −1.16 R_Y −0.498 R_X 0.231 R_Y −3.02

Figure 11. Quantum circuit for angle encoding and VQC in the QGNN architecture with RY, RX, and CNOT gates.

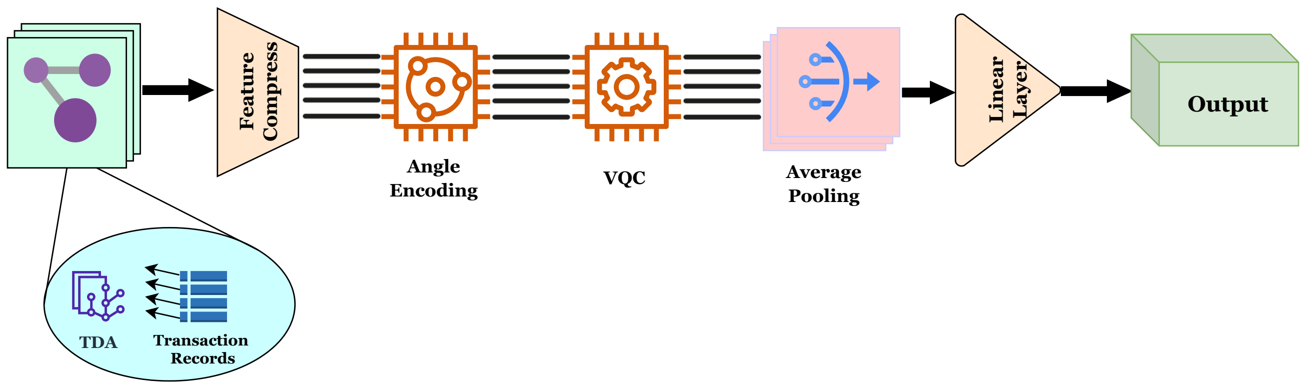

Figure 12. Architecture of the quantum graph neural network. The QGNN starts with a graph as input, followed by a feature compression step. Angle encoding prepares the data for the VQC processing. The output of the VQC is subjected to average pooling, then fed through a linear layer, and finally to the output layer

approach involved translating each transaction as a graph using topological data analysis [24]. This technique allowed us to capture intricate relationships among the various feature vectors, enhancing our ability to analyse the data effectively. In the following steps, we explain how to apply and implement this technique.

Given a transaction described by a vector

$$\mathbf{V} = [\text{Time}, V_1, V_2, \ldots, V_{28}, \text{Amount}], \tag{10}$$

We create the graph $G = (N, E)$, where $N \in \{R^1, R^2, \ldots, R^{28}\}$ represents the feature vector, and E represents the edge list, where, $V_1, V_2, \ldots, V_{28}$ denotes the encrypted feature labels in the dataset.

Next, we focus on converting the transaction, $\{T_i\}_{i=1}^{284\,807}$, into its graphical representations G_i via the following steps using TDA:

(i) Create a set $\{S_i\}$ of 3D data points, where $S_i = \{P_j \mid P_j = [\text{Time}_i, V_j, \text{Amount}_i], \forall j \in [1, 28]\}$. This helps in capturing the correlation of V_j with time and amount.

(ii) Project all data points $P_j \in S_i$ onto a 1D plane. We denote these projected points as f_j.

(iii) Draw covers around all the projected data points $\{f_j\}_{j=1}^{28}$ and cluster them using a suitable clustering algorithm, here, we have used DBSCAN. Let these clusters form a set $\{C_i\}$, where $C_i = \{c_k \mid \exists_{j,k} : f_j \in c_k, \forall j \in [1, 28] \textit{ and } 1 < k < |C_i|\}$

(iv) Draw edges between clusters with common data points to form the graph G_i.

Here, a node $N_k \in G_i$, represents a cluster $c_k \in \{C_i\}$, s.t. $N_k = (0 \ldots V_j \ldots 0)_{28}$, $\forall f_j \in c_k$. Thus, the relation between different features is captured and grouped using the clustering algorithm of high-dimensional data via a more representative graphical structure. Each graph represents a transaction labelled

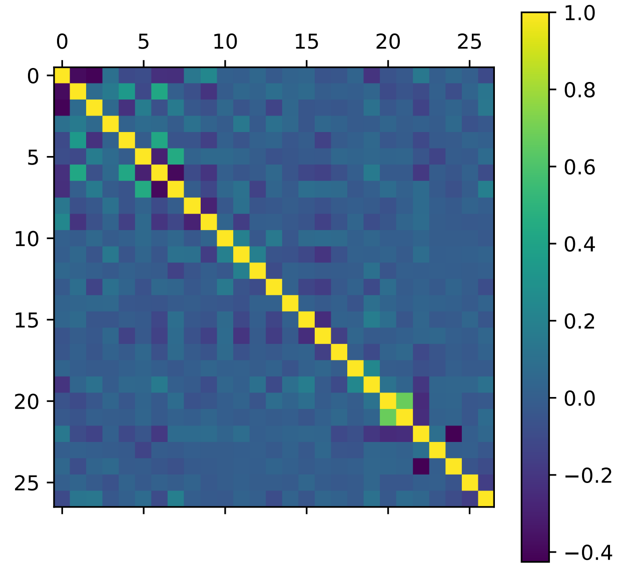

Figure 13. Correlation matrix ($v_1 - v_{28}$) of the dataset. The figure illustrates the relationships between the features v_1 to v_{28} by displaying their correlation values.

fraud or non-fraud for the classification task.

6. Results and Discussion

We tested the performance of our two classical and quantum algorithms on the same dataset described in Sec. 5. The results showed that the `GraphSAGE` model achieved promising results on the fraud classification task. The model was trained for 10 epochs using the Adam optimiser with a batch size of 5 using Keras, as shown in Tab. 2. The aggregation method was set to mean. These hyperparameters were chosen based on preliminary experiments and empirical observations to balance model complexity and generalisation performance.

Table 2. Hyperparameters and training details for the `GraphSAGE` model.

Hyperparameter	Value
Model Architecture	`GraphSAGE`
Number of Epochs	10
Optimiser	Adam
Batch Size	5
Aggregation Method	Mean

The receiver operating characteristic (ROC) curve presented in Fig. 15 shows the trade-off between the true positive rate (TPR) and the false positive rate (FPR) for the `GraphSAGE` model. The Area Under Cover (AUC) score measures how well a model can distinguish between positive and negative instances. A higher AUC score indicates a better model. The `GraphSAGE` model achieves an AUC of 0.77, within the acceptable range for fraud classification tasks. The performance metrics for the `GraphSAGE` model are also within the acceptable range, with an accuracy of 92.3%, and F1 score of 0.83. These scores are reported in Tab. 3.

The loss curves in Fig. 16 show that the training loss decreases steadily over time while the validation loss decreases more slowly, which suggests that the model is well-fitted on both the training and validation data.

We performed experiments on the quantum graph neural network using qiskit [25] under different scenarios, namely qubit count and hidden layers. The results in Tab. 4 show that a dense encoding scheme of node features using fewer qubits performed better than using more qubits and sparser representations, and this is evident as the model with 6 qubits, with an accuracy of 94.5% and F1 score of 0.86, performed best compared to that with 16 qubits, with an accuracy of 92.0% and F1-score of 0.81.

The AUC for the precision-recall curve from Fig. 17 is estimated to be roughly 0.85, which shows high tolerance in our method towards biased datasets with high class imbalances, apart from having highly accurate predictions. A smoother transition curve leads to a better F1 score, and the loss curves show a decrease in the loss function over time for both training and validation, which shows that our model is well-fitted, as presented in Fig. 18.

These results are observed on a basic network architecture of quantum circuits and outperform various classical approaches. This is due to the enhanced representation of graphical structures through quantum circuits and efficient feature relations using topological data analysis. We can see an improvement of $\approx$ 3% over classical GNN, such as `GraphSAGE`, across various metrics. However, it was observed that graphical techniques, in general, help enhance the precision and accuracy of these classification systems more than other non-graphical approaches.

Careful analysis of quantum techniques suggests that better representation, deeper search space, state entanglement, and qubit parallelism are key factors contributing to the supremacy of quantum circuits for classification tasks. In our experiments, we observed the same with a smaller system with 6 qubits and ≈200 parameters, which makes our model QGNN outperform the classical GNN. These results suggest that with more advanced features and complex quantum systems, we can achieve a real quantum advantage in solving problems such as fraud detection

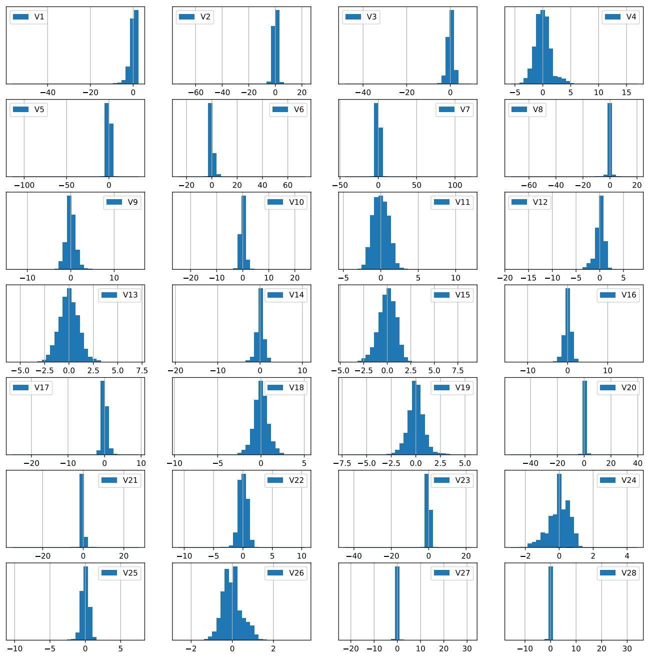

Figure 14. Histograms illustrating the distribution of all features in the dataset.

Table 3. Performance test metrics of `GraphSAGE`

Accuracy (%)	Precision (%)	Recall (%)	F1 score
92.3	95.2	76.3	0.83

Table 4. Performance test metrics of QGNN. The table presents results for various configurations, including the number of qubits and layers.

Num Qubits	Layers	Accuracy (%)	Precision (%)	Recall (%)	F1 score
6	1	94.5	96.1	79.5	0.86
16	1	92.06	86.5	77.3	0.81
6	2	91.5	93.3	65.6	0.76
16	2	89.2	86.6	72	0.77

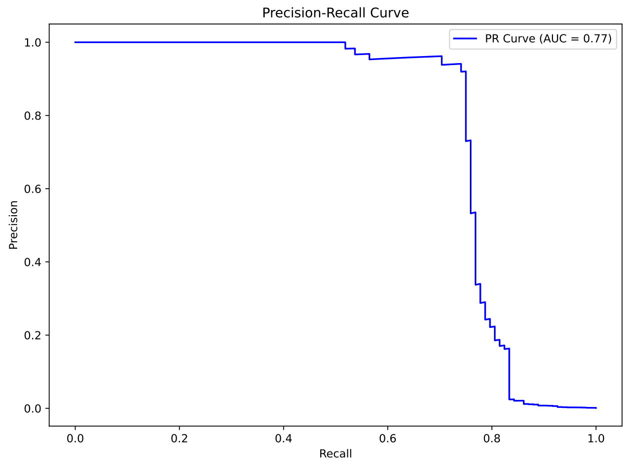

Figure 15. Receiver Operating Characteristic Curve for the `GraphSAGE` model with an AUC of 0.77.

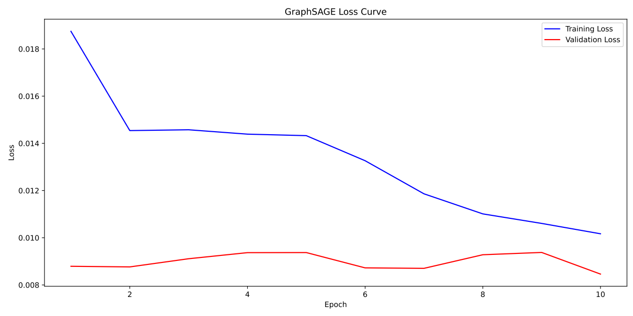

Figure 16. Training and validation loss curves for the `GraphSAGE` model.

in highly imbalanced datasets.

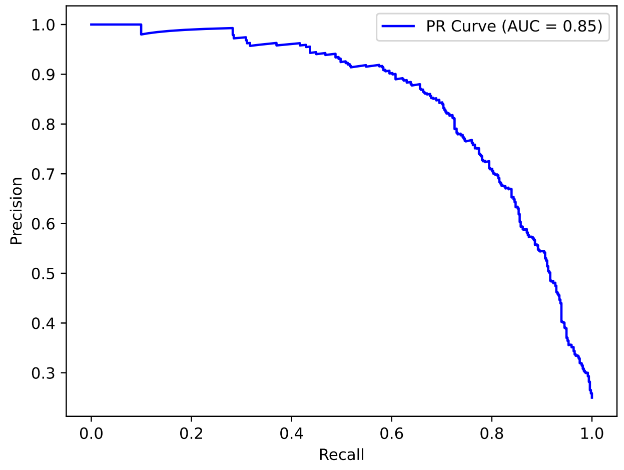

Figure 17. Receiver Operating Characteristic curve for QGNN model with 6 qubits and 1 hidden layer, with an AUC of 0.85

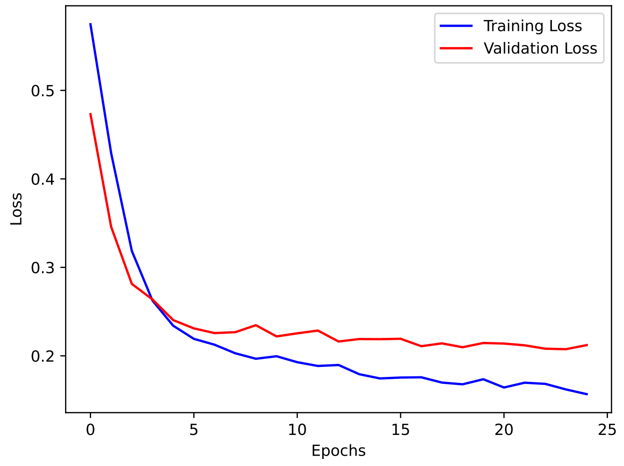

Figure 18. Training and validation loss curves for the QGNN model with 6 qubits and 1 hidden layer.

7. Conclusion

Our research has demonstrated the potential of quantum graph neural networks in detecting financial fraud. We have compared the performance of QGNNs with classical graph neural networks and found that QGNNs outperformed GNNs in terms of accuracy and efficiency. Our experiments on a real-world financial dataset show that QGNNs can detect fraud with a remarkable accuracy of 94.5%, which is significantly higher than GNNs' accuracy of 92.3%.

Moreover, we have also studied the influence of the number of qubits and layers on QGNNs' performance. Our research indicates that QGNNs with 6 qubits and one layer exhibit the best performance, with an accuracy of 94.5%. This can be attributed to the fact that fewer qubits are less prone to noise, and a single layer is sufficient to grasp the essential features of the data.

Our research shows that QGNNs have the potential to revolutionise fraud detection and improve the security of financial systems. It also highlights the need for further research to address the challenges of implementing QC in real-world applications.

Reference

[1] Ali A, Abd Razak S, Othman S H, Eisa T A E, Al-Dhaqm A, Nasser M, Elhassan T, Elshafie H and Saif A 2022 *Applied Sciences* **12** 9637

[2] Sadgali I, Sael N and Benabbou F 2019 *Procedia computer science* **148** 45–54

[3] Zojaji Z, Atani R E, Monadjemi A H *et al.* 2016 *arXiv preprint arXiv:1611.06439*

[4] Tüysüz C, Rieger C, Novotny K, Demirköz B, Dobos D, Potamianos K, Vallecorsa S, Vlimant J R and Forster R 2021 *Quantum Machine Intelligence* **3** 1–20

[5] Pourhabibi T, Ong K L, Kam B H and Boo Y L 2020 *Decision Support Systems* **133** 113303

[6] Yu S, Ren J, Li S, Naseriparsa M and Xia F 2022 *Frontiers in Artificial Intelligence* **5** 922589

[7] Grossi M, Ibrahim N, Radescu V, Loredo R, Voigt K, Von Altrock C and Rudnik A 2022 *IEEE Transactions on Quantum Engineering* **3** 1–12

[8] Ghosh S and Reilly D L 1994 Credit card fraud detection with a neural-network *System Sciences, 1994. Proceedings of the Twenty-Seventh Hawaii International Conference on* vol 3 (IEEE) pp 621–630

[9] Ma X, Wu J, Xue S, Yang J, Zhou C, Sheng Q Z, Xiong H and Akoglu L 2021 *IEEE Transactions on Knowledge and Data Engineering*

[10] Ahmad Z, Shahid Khan A, Nisar K, Haider I, Hassan R, Haque M R, Tarmizi S and Rodrigues J J 2021 *Applied Sciences* **11** 7050

[11] Zheng J, Gao Q and Lü Y 2021 Quantum graph convolutional neural networks *2021 40th Chinese Control Conference (CCC)* (IEEE) pp 6335–6340

[12] Kyriienko O and Magnusson E B 2022 *arXiv preprint arXiv:2208.01203*

[13] Innan N, Khan M A Z and Bennai M 2023 *arXiv preprint arXiv:2308.05237*

[14] Xiang S, Zhu M, Cheng D, Li E, Zhao R, Ouyang Y, Chen L and Zheng Y 2023 Semi-supervised credit card fraud detection via attribute-driven graph representation *Proceedings of the AAAI Conference on Artificial Intelligence* vol 37 pp 14557–14565

[15] Schetakis N, Aghamalyan D, Boguslavsky M, Rees A, Raktomalala M and Griffin P 2023 *arXiv preprint arXiv:2308.03575*

[16] Beer K, Khosla M, Köhler J, Osborne T J and Zhao T 2023 *Physical Review A* **108** 012410

[17] Siegelmann H T 2005 *Computer Science Today: Recent Trends and Developments* 29–45

[18] Kiran Kadambari S and Prabhakar Chepuri S 2020 *arXiv e-prints* arXiv–2001

[19] Gao Z, Isufi E and Ribeiro A 2021 *Signal Processing* **188** 108216

[20] Cerviño J, Ruiz L and Ribeiro A 2022 Training stable graph neural networks through constrained learning *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (IEEE) pp 4223–4227

[21] Lo W W, Layeghy S, Sarhan M, Gallagher M and Portmann M 2022 E-graphsage: A graph neural network based intrusion detection system for iot *NOMS 2022-*

2022 IEEE/IFIP Network Operations and Management Symposium (IEEE) pp 1–9

[22] Hu Z, Li J, Pan Z, Yang S Z L, Ding C, Khan O, Geng T and Jiang W 2022 On the design of quantum graph convolutional neural network in the nisq-era and beyond *IEEE 40th International Conference on Computer Design (ICCD)* pp 3–5

[23] ULB M L G 2013 Credit card fraud detection URL `https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud`

[24] Chazal F and Michel B 2021 *Frontiers in artificial intelligence* **4** 108

[25] Aleksandrowicz G and et al 2019 Qiskit: An Open-source Framework for Quantum Computing URL `https://doi.org/10.5281/zenodo.2562111`