Quantum Feature Embeddings for Graph Neural Networks

Sascha Xu
German Research Center for
Artificial Intelligence (DFKI)
s8xgxuuu@stud.uni-saarland.de

Frank Wilhelm-Mauch
Forschungszentrum
Jülich (FZJ)
f.wilhelm-mauch@fz-juelich.de

Wolfgang Maass
German Research Center for
Artificial Intelligence (DFKI)
wolfgang.maass@dfki.de

Abstract

Quantum computing offers a promising avenue to reduce growing machine learning model complexity as required in large language models and simulation models for weather forecasts, financial forecasts, or engineering. Graph neural networks are a particular class of machine learning models that have garnered much attention for their ability to deal well with structured data. We investigate how to enhance existing GNNs and find through the inductive bias that quantum circuits are used best to encode node features. The proposed Quantum Feature Embeddings (QFEs) turn raw input features into quantum states, enabling non-linear and entangled representations. In particular, QFEs provide normalized, non-redundant weight matrices in an exponentially larger feature space and require much fewer qubits than fully quantum graph neural networks. On standard graph benchmark datasets, we showcase that for the same parameter count QFEs perform better than their classical counterpart, and are able to match the performance of an exponentially larger model. Finally, we study the potential benefit of using a hybrid quantum graph neural network over a classic alternative on a concrete use case, laser cutting. We find that the proposed model has the performance and thus the near-term potential to uplift these business applications.

Keywords: Quantum Machine Learning, GNNs

1. Introduction

Machine learning (ML) models have become increasingly complex, with vast amounts of data and parameters, requiring significant computational resources for training and inference (Patterson et al., 2021; Touvron et al., 2023). As a consequence, the practical implementation of these models on traditional computing platforms often becomes infeasible due to resource limitations. However, the emergence of quantum computing offers a promising avenue to address these challenges, as it has unlocked unprecedented computational capabilities to solve previously infeasible problems such as the integer factorization problem (Shor, 1999). These advancements have inspired researchers to explore the integration of quantum techniques in the realm of ML to mitigate the complexity of large-scale models (Beer et al., 2020; Lloyd et al., 2020).

The primary objective of this paper is to investigate the application of quantum computing to reduce the complexity of ML models, with a particular focus on Graph Neural Networks (GNNs). GNNs have garnered substantial attention due to their ability to generalize across various model classes and exhibit exceptional performance in tasks involving graph-structured data (Xu et al., 2019). By leveraging the unique characteristics of quantum computing, we aim to accelerate and enhance the capabilities of GNNs.

Rather than pursuing a fully quantum approach, which faces challenges related to scalability and hardware constraints, we propose a hybrid architecture that combines classical and quantum computing paradigms (De Luca, 2022). This approach harnesses the strengths of both classical and quantum systems, offering a practical solution that can deliver notable performance improvements without sacrificing scalability. In this paper, we will explore the motivation behind integrating quantum computing with GNNs, identifying the areas where quantum acceleration can be most effective. Furthermore, we demonstrate how a hybrid architecture can be composed and provide empirical evidence of its enhanced performance compared to conventional, purely classical GNN models. By focusing on the construction of quantum-enhanced GNNs, we aim to pave the way for more efficient and powerful ML models capable of addressing complex real-world problems.

2. Related Work

Quantum machine learning (QML) approaches can be categorized into two main types: fully quantum computing-based methods and hybrid approaches that combine quantum computing components with classical neural networks. In general, a QML model which works with classical data requires three distinct blocks (Benedetti et al., 2019): data encoding, a variational circuit, and the measurement stage. The data encoding maps classical input data onto qubits, making it accessible to the quantum circuit. The variational circuit incorporates tuneable operations that adjust its parameters to fit the desired function. Finally, the measurement stage transforms the quantum state back into a real vector. The exact implementation of each component is open for exploration. Schuld and Petruccione, 2021 propose optimizing the measurement basis to learn functions, while Lloyd et al., 2020 suggests optimizing the variational circuit to maximize the distance between classes of points for classification tasks. Here, Lloyd et al. demonstrate the accuracy of their approach by replacing the fully connected layer of a convolutional neural network. Both approaches enable the realization of parameterized, optimizable functions on quantum circuits.

In analogy to deep learning, deep QML models are increasingly investigated. There exists however a major roadblock, as revealed by McClean et al., 2018. The problem of barren plateaus in deep, densely connected quantum circuits leads to vanishing gradients (Hochreiter, 1998) and hence a non-optimizable function. To address the challenge of barren plateaus, Beer et al., 2020 introduce a deep learning-inspired approach that restricts entanglement between qubits, allowing for the training of deep quantum neural networks. Beer et al. show that deep quantum neural networks can be trained accurately even with limited data, as confirmed by Caro et al., 2022, who provide an upper bound on the generalization error of QML models, which show that QML models work well with a relatively small amount of data.

Finally, specialized QML methods have been developed for structured data such as images and graphs. In this paper, our focus lies on quantum graph neural networks (QGNNs). Verdon et al., 2019 provide a general formulation for QGNNs, which involves encoding the feature and adjacency matrices and processing them with a parameterized unitary, realizing convolutional and recurrent variants with appropriate constraints. Most closely related is the work of Chen et al., 2021, who propose a hybrid quantum-classical graph convolutional network, suggesting the replacement of the fully connected layer with a parameterized quantum circuit. However, the exact benefits of this augmentation in the architecture are not fully clear. Our objective is to identify the specific parts of a graph neural network that benefit most from QML by analyzing the inductive biases introduced by QML methods.

3. Quantum Feature Embeddings

Consider a homogeneous graph with N nodes and D node features, represented by a feature matrix $X \in \mathbb{R}^{N \times D}$ and an adjacency matrix A. Each row vector $x^{(i)}$ of the feature matrix X corresponds to the input features of node i. The graph has a label $Y \in \mathcal{Y}$ to be predicted. For binary classification, $\mathcal{Y}$ is defined as $\{0, 1\}$, while for tasks some tasks we may be interested in node-wise labels, e.g. a vector from $\mathcal{Y} := \mathbb{R}^N$.

Given a dataset of M labeled graphs $\{X_i, A_i, Y_i\}_{i=1}^{M}$, our objective is to fit a parameterized function f_θ by minimizing the loss between the predicted label $f_\theta(X_i, A_i) = \hat{Y}_i$ and the true label Y_i, achieved by updating the parameters θ. We focus on GNNs as the class of parameterized functions f_θ.

Graph Neural Networks (GNNs) are a powerful class of machine learning models designed to handle data structured as graphs (Wu et al., 2020; Zhou et al., 2020). GNNs typically consist of three main components: feature embedding, message passing, and decoding. In the feature embedding phase, graph nodes and edges are mapped to high-dimensional feature vectors, capturing their characteristics and relationships, shown on the left of Figure 1. During the message-passing phase, GNNs iteratively aggregate and propagate information between connected nodes, allowing for the incorporation of local and global graph information. The message passing step for a node is shown in the middle of Figure 1. Finally, in the decoding phase, the learned representations are used to make predictions or perform downstream tasks. GNNs have demonstrated impressive performance across various domains, including social network analysis and drug discovery.

In our proposed Quantum Feature Embedding (QFE) framework, we specifically focus on the feature embedding stage as the ideal application of QML. Traditional methods often employ multi-layer perceptrons (MLPs) as node embedders, which are universal approximators capable of representing any function f, as proven by Hornik et al., 1989. Often, this leads MLPs to overfit rather than generalize. To address this, special neural network architectures constrain the set of learnable functions, encoding an *inductive bias* that specifies desired properties of the function.

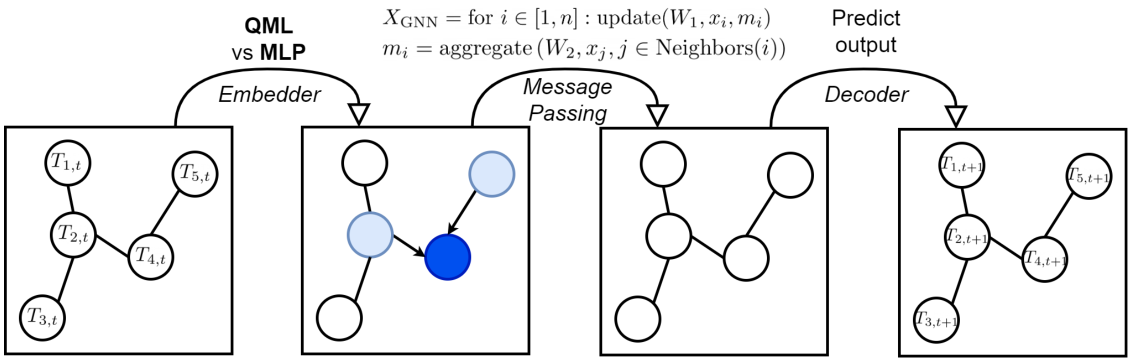


Figure 1: GNN flow: Node vectors are embedded individually by ML/QML component. Message Passing updates node embeddings considering features of the neighborhood. Decoder predicts labels from embeddings.

3.1. Inductive Bias

Inductive bias plays a crucial role in deep learning by guiding the learning process and influencing the model's generalization ability (Battaglia et al., 2018; Baxter, 2000; Zhang et al., 2021). It refers to the set of assumptions and biases encoded in the model architecture and learning algorithms that help it make predictions and learn from limited data. Inductive bias can take various forms, such as architectural choices, regularization techniques, or the use of specific loss functions. It allows deep learning models to effectively capture and exploit patterns in data and prioritize relevant features. For instance, in GNNs, the inductive bias allows updating node features using neighbors only, which allows the GNN to fit robust functions that generalize beyond the training dataset.

Our primary objective is to explore the inductive biases introduced by QML, particularly in relation to the feature embedding stage of GNNs, and discover the potential benefits of using Quantum Feature Embeddings instead. Here, we will be analyzing parameterized quantum circuits, described through their unitary operator $U(\boldsymbol{\theta})$, which transforms a quantum state $|x\rangle$ using specific gates and parameters. The unitary operator is represented as $U(\boldsymbol{\theta}) = U_n(\theta_n)U_{n-1}(\theta_{n-1})\ldots U_1(\theta_1)$, where each $U_i(\theta_i)$ is a quantum gate with its associated parameter θ_i, e.g. a rotation around the X-axis by degree θ_i. Goto et al., 2021 show that parameterized quantum circuits are also universal approximators and hence equivalent in that regard to conventional MLPs. A parameterized unitary is the quantum equivalent to the linear weight multiplication Wx in a classical MLP, but encodes additional constraints that we analyze now.

3.2. Quantum Feature Space

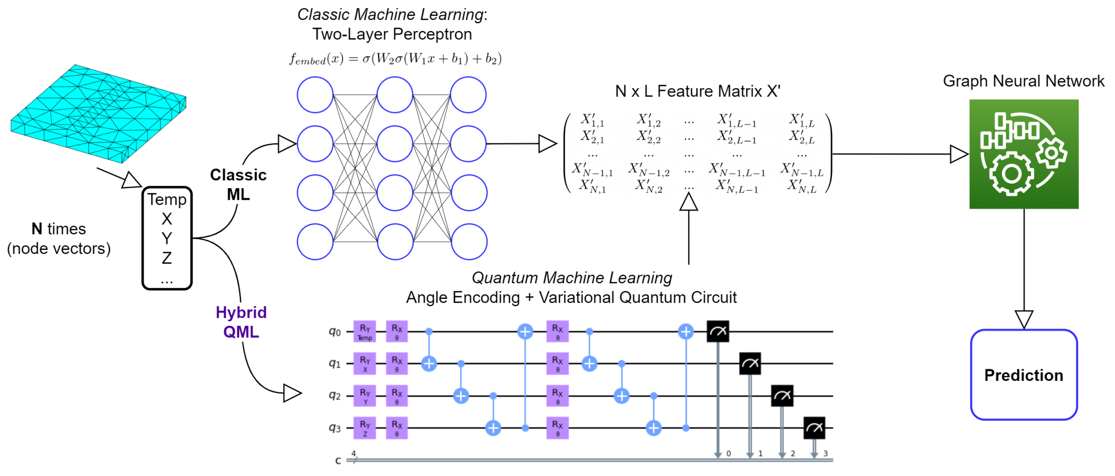
Quantum circuits have an exponentially sized feature space in relation to the number of qubits. Consequently, the dimensionality of the data representation $|x\rangle$ grows exponentially whilst the required amount of gates/parameters scales only polynomially. In contrast, the hidden representation of MLPs depends linearly on the number of neurons. Hence, to reproduce the feature space of a quantum circuit with an MLP, exponentially more neurons and parameters are required, which makes both computation and optimization challenging.

In addition, quantum states and transformations have the property of *normalization*, since unitaries are norm preserving, i.e $||U\,|x\rangle\,|| = ||\,|x\rangle\,||$. Normalization ensures similarity in scale and prevents a single feature with a larger magnitude from dominating the learning process. It improves the stability and convergence of gradient descent by providing a balanced landscape, as different scales among features can destabilize optimization (Ioffe and Szegedy, 2015). Finally, normalization reduces the impact of outliers, enhancing the model's robustness and reducing overfitting. The intermediate and final state vectors in QML models are inherently normalized, ensuring reliable computations without using costly batch normalization.

3.3. The Unitary Prior

QML models leverage unitaries, which are linear operators in the quantum state space, further constrained to model only invertible transformations. They possess several desirable properties, including:

1. Orthonormal Basis: The columns/rows of a unitary matrix form an orthonormal basis of the complex vector space. Additionally, U has full rank, ensuring a rich representation capacity.



Classic Machine Learning:
Two-Layer Perceptron
$f_{embed}(x) = \sigma(W_2\sigma(W_1x + b_1) + b_2)$
N x L Feature Matrix X'
Graph Neural Network
Classic ML
N times (node vectors)
Temp X Y Z ...
Hybrid QML
Quantum Machine Learning
Angle Encoding + Variational Quantum Circuit
Prediction

Figure 2: QFE-GNN framework with quantum feature embeddings: the node features of the input graph are embedded with a QML model (top) instead of an MLP (bottom), before being passed to a classical GNN.

2. Embedding Uniqueness: Unitary transformations are bijections. Consequently, it produces unique embeddings for unique inputs and covers the entirety of the output space.

In contrast, the linear weight matrices in traditional deep learning are not of full rank and contain redundancies, exploited for compression (Nakkiran et al., 2015) or acceleration (Denton et al., 2014). However, for feature embeddings, low-rank weight matrices and thus linear dependencies between embedded vector elements, is undesirable, as the node feature embedding aims to automatically learn relevant, not redundant features from the input vector of each node. When using a classic MLP, the optimization process based on gradient descent leads to a non-full-rank weight matrix W. In contrast, QML introduces an inductive bias so that the learned embedding must be linearly independent.

Furthermore, by representing only bijective functional mappings, the embedded representations of nodes can be uniquely mapped back to their original input features. This allows for a clear understanding of how the embedded representations relate to the original data. Bijectivity helps in preserving the richness of the node information during the embedding process. By ensuring that the embedding function is a one-to-one mapping, bijectivity prevents information loss or distortion. In GNNs, accurate representation of the nodes and their relationships is crucial for capturing the underlying graph structure. Non-bijective embeddings may result in the collapse of distinct nodes into the same representation, leading to the loss of discriminative power and potentially compromising the performance of downstream components. Overall, QML allows only learning unique, non-redundant representations for the node input features and thus ensures faithful and informative node embeddings in GNNs.

3.4. Framework

We introduce Quantum Feature Embeddings (QFEs) to enhance learned feature representations in graph neural networks (GNNs) using quantum machine learning. QFEs are integrated into a hybrid framework that combines quantum-inspired feature embedding with classic ML-based message passing. The inductive biases leading to improved feature representations are present for all parameterized quantum circuits and allow to flexibly instantiate different architectures as QFEs.

The differences between a traditional GNN and our proposed QFE-GNN are illustrated in Figure 2. We are given an input graph with N node vectors, representing a 3D laser cutting object for example. The QFE-GNN uses a QML model to embed node features, which are processed further by a classic GNN performing feature updates based on node neighborhood. The hybrid architecture combines the benefits of both classical and quantum approaches, allowing for improved feature extraction and representation learning compared to the classical MLP-powered approach.

The QFE-GNN framework is *hardware efficient*, as it requires relatively few qubits compared to a fully quantum GNN. QFEs embed individual node vectors only and not the entire graph, thereby requiring only small quantum hardware. Overall, by incorporating QFEs into a classic GNN architecture, we match the advantages of quantum machine learning by placing it in the best-suited role, feature embedding.

Accommodating hardware constraints, QFE-GNN aims to improve the performance and efficiency of graph representation learning in various domains.

3.5. Instantiation

We now present an architecture for Quantum Feature Embeddings to instantiate a QFE-GNN. We describe the key components necessary for a QML model processing classical data as described by Benedetti et al., 2019.

Input Encoding: First, the real-valued input vector $x = (x_1, \ldots, x_D)$ must be mapped onto quantum state $|x\rangle$ for the QML model.We want to make use of the exponential-sized Hilbert space and hence use *angle encoding* with D qubits for D input features. Each qubit is subjected to a rotation (e.g. around the Y-axis), defined by the rotation matrix:

$$U_{R_Y(x_i)} = \begin{pmatrix} \cos(\frac{x_i}{2}) & -\sin(\frac{x_i}{2}) \\ \sin(\frac{x_i}{2}) & \cos(\frac{x_i}{2}) \end{pmatrix} ,$$

where the angle x_i is the normalized raw input data. The resulting quantum state is given by $\phi_{\text{angle}}(x) = \otimes_{i=1}^{D} U_{R_Y(x_i)} |0\rangle$, and is an exponential size vector in $\mathbb{C}^{2^D}$. Angle encoding exhibits operational advantages by requiring only a single sequential gate per qubit.

Parameterized Gates: Once the input data x is embedded into a quantum state $|x\rangle$, we follow the design paradigm by Lloyd et al., 2020 and optimize the parameters of the variational circuit. We use a standard architectures that involves alternating parameterized rotations and entanglement of qubits using CNOT gates (Chen et al., 2021; Selig et al., 2021).

The parameterized rotation operation, denoted as $U_{R_X(\theta)}$, acts on each qubit and is defined as:

$$U_{R_X(\theta)} = \begin{pmatrix} \cos(\frac{\theta}{2}) & -i\sin(\frac{\theta}{2}) \\ -i\sin(\frac{\theta}{2}) & \cos(\frac{\theta}{2}) \end{pmatrix} .$$

By combining these rotation operations, we can construct a unitary operator for the entire layer as $U_{R_X\text{-Layer}(\theta)} = \otimes_{i=1}^{D} U_{R_X(\theta_i)}$. Subsequently, we introduce an *entanglement* operation to establish interactions between the encoded values on the qubits, by entangling neighboring qubits with Controlled-NOT (CNOT) gates. We repeat the rotation-entanglement procedure twice to limit circuit depth and avoid barren plateaus (McClean et al., 2018), for a total of $2D$ rotations, and $O(D)$ parameters.

Measurement: Finally, the processed data in form of a quantum state needs to be transformed back into a real vector to be processed further with the classic GNN. To this end, we use a fixed observable and measure *each* *individual* qubit using the Z basis with $M_0 = |0\rangle\langle 0|$ and $M_1 = |1\rangle\langle 1|$ to yield a real vector x', which is then used as the node embedding for the GNN.

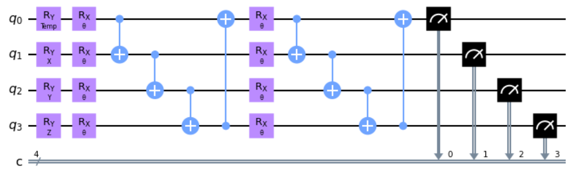


Figure 3: Full QML Model consisting of input encoding stage (left), the variational, parameterized circuit (middle) and measurement stage (right).

Full Model: The complete proposed QML architecture is displayed in Figure 3. Each node has a feature vector $x \in \mathbb{R}^D$. The corresponding embedding circuit uses D qubits in total which are in the initial state of $|0\rangle$. The classic data is encoded as angles on the R_Y rotations, represented by the first purple blocks on the left with a gate depth of one. Next follows the variational circuit consisting of a parameterized R_X rotation followed by CNOT entanglement with the direct neighbors. The circuit is repeated twice, which is the setup with the best observed empirical accuracy. The processed quantum state is then Z measured for each individual qubit. In the end, one obtains an QFE-embedded latent node vector $x' \in \mathbb{R}^D$ which is now ready for further processing by the classic GNN.

3.6. Model Training

The training of the QFE graph neural network is based on a chosen loss function, e.g. the MSE for regression or cross-entropy for classification. The loss evaluates the network's output and computes gradients with respect to the parameters. In the classical GNN component, the gradients are obtained with standard backpropagation. This allows the GNN to learn and refine its parameters based on the error signal provided by the loss function. Simultaneously, in the quantum part of the network, the embedding parameters that encode classical features into quantum states need to be optimized. The parameter-shift rule is commonly used to estimate gradients by evaluating the quantum circuit with slight perturbations in the embedding parameters (Crooks, 2019; Schuld et al., 2019). The resulting change in loss gives an accurate estimate of the gradient. These gradients are then used to update the embedding parameters through gradient descent (Bottou, 2010).

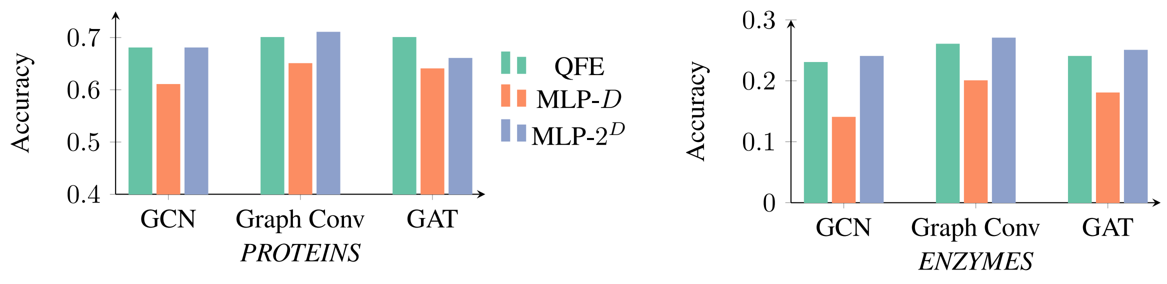


Figure 4: [Higher is better] Accuracy on *PROTEINS* (left) and *ENZYMES* (right) datasets with QFE and MLP embedding. The QFE outperforms MLP-D and matches the performance of the exponentially larger MLP-2^D.

4. Experiments

In this section, we evaluate the Quantum Feature Embeddings (QFEs) on both synthetic and real-world graph datasets. We compare the performance of our introduced QFE architecture, which uses D qubits for D input features, against a classic Multi-Layer Perceptron (MLP)-based node embedder. We consider two variations of the MLP: MLP-D with D hidden neurons (equally many parameters as QFE), and MLP-2^D with 2^D exponentially many neurons (same hidden dimensionality).

Each embedding is combined with three popular message-passing architectures: Graph Convolutional Network (GCN) (Welling and Kipf, 2016), GraphConv (Morris et al., 2019), and Graph Attention Networks (GAT) (Velickovic et al., 2017). These architectures provide different approaches for aggregating node activations and have been widely used in graph machine learning. We implemented all Hybrid Graph Neural Networks in Python and used Pennylane (Bergholm et al., 2018) to simulate the QML model.

4.1. Real-World Datasets

In order to evaluate the performance of quantum embeddings in real-world scenarios, we consider graph/node classification or regression tasks on real-world datasets. We utilize the *PROTEINS* (Dobson and Doig, 2003) and *ENZYMES* (Schomburg et al., 2004) graph classification datasets from the TUDataset of Morris et al., 2020. These datasets encode proteins as connected amino acids, and the task is to classify them as enzymes or not (*PROTEINS*) and to distinguish between six possible catalyzed chemical reactions (*ENZYMES*). Both benchmarks provide midsize graphs with low dimensional node features and allow to train of a QFE-GNN within a reasonable time frame (less than a week). Since both datasets involve classification, we trained the networks using the cross-entropy loss.

After running and evaluating all models with cross-validation, we obtain the following results. Figure 4 depicts the accuracy of all QFE and MLP GNN combinations on *PROTEINS* and *ENZYMES*. On the *PROTEINS* benchmark, MLP-D significantly underperforms the other two methods, achieving an accuracy of **0.61**. Both MLP-2^D and QFE achieve nearly identical rounded test accuracies of **0.68**, with a slight advantage for MLP-2^D. Similar observations hold when using the GraphConv layer as the GNN. Notably, when employing the GAT layer, the QML model outperforms both MLPs, achieving an overall accuracy of **0.70**. The best-performing model is the combination of GraphConv and MLP-2^D, with a close second place for GraphConv and QFE. As expected, GCN performs relatively poorly due to its limitations highlighted by the authors of GraphConv Morris et al., 2019. Given that identifying unique substructures is crucial in protein analysis, GraphConv capitalizes on its strengths, surpassing the graph attention network. These strengths become particularly relevant when addressing laser cutting and thermal simulation tasks.

A similar pattern is observed on the *ENZYMES* dataset. Once again, MLP-2^D and QFE perform comparably well, while MLP-D demonstrates weaker empirical performance. This trend persists across all three GNN layers when employing cross-validation. It appears that QML can achieve performance on par with MLPs using exponentially fewer parameters on this benchmark as well. GraphConv with MLP-2^D emerges as the top performer, followed by GraphConv with our QML embedding. The MLP struggles to achieve good performance with only linearly many parameters, whereas the QML model excels.

In conclusion, our empirical evaluation demonstrates the potential of quantum embeddings in graph classification tasks. The results consistently show that the QFE-GNN outperform their classical counterparts using the proposed parameterized quantum circuit design. The findings highlight the advantage of

	PROTEINS		*ENZYMES*	
	No Noise	Noisy	No Noise	Noisy
GCN	0.68	0.64	0.23	0.15
GraphConv	0.70	0.60	0.26	0.22
GAT	0.70	0.59	0.24	0.19

Table 1: [Higher is better] Accuracy of a simulated perfect quantum circuit versus a simulated noisy circuit.

	ENZYMES	*PROTEINS*
QML training run	2:20 h	3:10 h
ML training run	<0:01 h	<0:01 h
QML total time	35:00 h	46:00 h
ML total time	0:02 h	0:03 h

Table 2: [Lower is better] Training time of QML and ML models per training run and total time for cross-validation.

leveraging the inductive bias provided by quantum circuits in capturing complex graph structures. These promising results open up exciting possibilities for applying quantum embeddings to various domains.

4.2. Noisy Quantum Feature Embeddings

Current day quantum hardware is considered to be in the noisy, intermediate scale quantum era, widely known as NISQ. QFEs require only few parameters/qubits, and are well suited to the scale constraints of modern hardware. In the following, we will examine the effect of hardware noise. To this end, we use the trained hybrid models from the previous section, but in place of a perfect simulator use a simulator of an IBM Falcon r4T device and the device specific noise.

Table 1 shows the results of this experiment. Across all benchmarks, the performance including hardware noise lies slightly below the perfect simulation. Indeed, noise may affect the precision of the, especially when using fewer qubits. To remedy this problem, integrating the noise into the training process is a potential avenue of improvement, as using noise for data augmentation in neural network training is often beneficial to training (Shorten and Khoshgoftaar, 2019).

4.3. Time and Resource Consumption

Compared to traditional ML models, simulating QML models has significantly longer training times. Table 2 shows that simulating QML models is very time-consuming. For *ENZYMES* and *PROTEINS*, to process a graph in parallel, we require on average 40 nodes $*\ 3$ features $= 120$ qubits, and a total of 600 gate operations per graph with circuit depth of 5. In contrast, the classic model MLP-2^D needs 21.120 floating point operations. As of now, IBMs most advanced 400 qubit processor is said to have 15k CLOPS (circuit layer operations per second). Ignoring all possible overheads, we obtain for the training of the QFE-GNN on the *PROTEINS* dataset with 1000 graphs$*$ 600 gates $*\ 50$ epochs with 15 kCLOPS$/s$ compute power an estimated run time of 33 minutes.

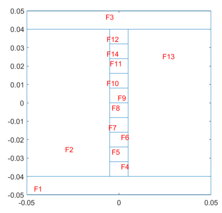
On the economic side, GPU hardware is currently magnitudes cheaper than quantum hardware. It remains to see, whether affordable quantum hardware and thus economic viability of QFEs will be achieved in the long term. Independent of specific hardware costs, the theoretical complexity of both models can be determined. The QML model needs only a linear amount of gates $O(D)$, whereas the MLP with exponentially more neurons that matches the performance, hence requiring exponentially many operations $O(2^D)$. This exponential relationship bodes well for the QFE approach. With increasing amounts of data that can be handled by quantum hardware, the balance tips further in favor of the quantum machine learning approach due to the exponential scaling.

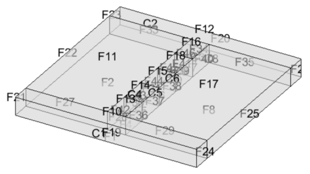
4.4. Case Study: Laser Cutting

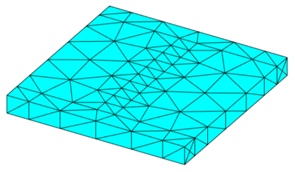
Finally, we analyze the potential of a QFE-GNN on a concrete use case, laser cutting. Laser cutting is a technology that enables to precisely and flexibly cut fine geometries on thin metal sheets and is an important part of automotive industrial production (Thomas et al., 2011). Currently, laser cutting machines operate fully automated, however they cannot operate autonomously and miss out on a lot of economic potential by being able to operate alone through the night. Due to the heat induced by the laser there is thermal expansion, which can cause the machine to be blocked and require human intervention. Simulating the induced heat accurately allows to enact appropriate countermeasures and prevent production standstills.

The state-of-the-art approach for heat distribution simulation involves using Partial Differential Equation solvers (Akhtar et al., 2014). However, these solvers are computationally intensive and slow when applied to the entire cutting process. Instead, Graph Neural Networks (Han et al., 2021; Pfaff et al., 2021) have demonstrated their potential as powerful alternatives to PDE solvers. The cut geometry, a mesh composed of many small simple shapes, is processed by a GNN, which learns to simulate the thermodynamics of laser cutting.

A dataset is necessary for training purposes, and we


(a) 2-D geometry of a straight cut into a metal plate.


(b) Vertically lifted 3-D geometry from the 2-D sketch.


(c) Mesh constructed from the provided 3-D geometry.

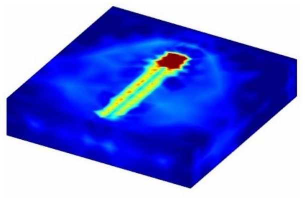

(d) Simulated temperature on the mesh during laser cutting. Predictions by the QFE-GNN.

Figure 5: From geometry to simulation on a mesh. Steps involved in simulating laser cutting.

	MLP-D	MLP-2^D	QFE
GCN	0.54	**0.50**	**0.50**
GraphConv	0.56	0.55	**0.39**
GAT	0.34	0.29	**0.21**

Table 3: [Lower is better] MSE after 20 training epochs on the *LASER* dataset.

generate this dataset using the PDE solver from Matlab. The simulation process is outlined in Figure 5. We test feasibility by modeling the simplest shape, straight cuts in a rectangular metal sheet, which we represent as a 2D geometry (Figure 5a). To extend the sheet geometry into 3D, we introduce a parameterizable height (Figure 5b), and then convert it into a mesh graph that is utilized by the GNN (Figure 5c). By training on these graphs, the GNN learns to predict the thermodynamics and simulate the heat spread during the cutting process (Figure 5d).

In total, we generate 60 different geometries. Each geometry is a graph formed by the nodes and edges of the mesh (see Figure 5c). On average there are 900 nodes, where each node vector carries information about X/Y/Z coordinates, current temperature, thermal conductivity and more (11 qubits per node). We embed each node vector with QFE/MLP respectively and use the combined GNN to predict the nodal temperature at the next timestep with 60 timesteps per simulation.

To evaluate the performance, we again test all hybrid/classical GNN variants of this dataset. The results are presented in Table 3. Among all the GNNs, the QFE-GNN variant performs the best, demonstrating the advantages of incorporating quantum elements in the appropriate sections of GNNs. Ultimately, the hybrid graph attention network with QFEs emerges as the top-performing model. It achieves an impressive MSE of **0.21** and outperforms other GNN and embedding configurations. By leveraging graph attention, this network automatically determines the most relevant nodes, especially when provided with high-quality representations from QFEs. By replacing the embedder component in this network with a QML-based solution, we achieve the lowest training error while utilizing exponentially fewer parameters compared to competing models. This empirical evidence further substantiates the efficacy of our QML models and demonstrates their advantages in real-world applications like laser cutting.

5. Conclusion

This article introduced the concept of Quantum Feature Embeddings for Graph Neural Networks to enhance and accelerate classical graph neural networks using quantum computing. By leveraging the inductive biases offered by quantum circuits, we propose the QFE framework for a hybrid quantum-classical GNN. We showed that by design QFEs produce informative and distinct embeddings, leading to improved empirical accuracy of GNN architectures across various benchmarks. Furthermore, we found that QFEs achieve comparable accuracy to exponentially larger embedder models, highlighting the significant advantage provided by the larger quantum feature space.

To illustrate the utility of QFE-GNNs, we presented a real-world use case involving laser cutting. The results showcased how a QFE-GNN could be leveraged in the near term to accelerate a business application, thanks to its computational advantages. QFE-GNNs are a promising tool for addressing real-world challenges with increased efficiency and accuracy.

In summary, the QFE approach contributes to the feature engineering and data pre-processing in machine learning models. We have shown that QFEs can be used to enhance the processing of graph neural networks and, thus, convolutional neural networks. They are applicable to any unstructured input features, as long as the number of features is close to the number of qubits that can be processed by the simulation system or quantum computer. This allows to leverage the quantum computing advantage on problems that can be transformed into graph representations with a reduced set of independent features.

5.1. Future Work

In our future work, we seek to understand the underlying mechanisms that contribute to the success of the QFE framework in improving GNN performance. We assume that exploring the theoretical foundations and conducting empirical analyses will shed light on the specific aspects of quantum feature embeddings that contribute to their informativeness and uniqueness. This investigation involves examining the impact of different quantum circuit structures and training strategies on the effectiveness of QFEs. Furthermore, scaling the QFE-GNN approach to larger datasets and more complex graph structures is an exciting avenue for future research. Exploring techniques for efficient computation, parallelization, and optimization can help overcome scalability challenges and enable the application of QFE-GNNs to real-world problems with massive graphs. Investigating both the theoretical underpinnings and scalability will pave the way for harnessing the full potential of QFE-GNNs and establishing them as reliable and efficient tools in the domains of machine learning and graph analysis.

Acknowledgements

This work is part of the research project QUASIM (grant number: 01MQ22001A, www.quasim-project.de), funded by the German Federal Ministry for Economic Affairs and Climate Action (BMWK).

References

Akhtar, S., Kardas, O. O., Keles, O., & Yilbas, B. S. (2014). Laser cutting of rectangular geometry into aluminum alloy: Effect of cut sizes on thermal stress field. *Optics and Lasers in Engineering*, *61*, 57–66.

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al. (2018). Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*.

Baxter, J. (2000). A model of inductive bias learning. *Journal of artificial intelligence research*, *12*, 149–198.

Beer, K., Bondarenko, D., Farrelly, T., Osborne, T. J., Salzmann, R., Scheiermann, D., & Wolf, R. (2020). Training deep quantum neural networks. *Nature communications*, *11*(1), 1–6.

Benedetti, M., Lloyd, E., Sack, S., & Fiorentini, M. (2019). Parameterized quantum circuits as machine learning models. *Quantum Science and Technology*, *4*(4), 043001.

Bergholm, V., Izaac, J., Schuld, M., Gogolin, C., Alam, M. S., Ahmed, S., Arrazola, J. M., Blank, C., Delgado, A., Jahangiri, S., et al. (2018). Pennylane: Automatic differentiation of hybrid quantum-classical computations. *arXiv preprint arXiv:1811.04968*.

Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *Proceedings of compstat'2010* (pp. 177–186). Springer.

Caro, M. C., Huang, H.-Y., Cerezo, M., Sharma, K., Sornborger, A., Cincio, L., & Coles, P. J. (2022). Generalization in quantum machine learning from few training data. *Nature communications*, *13*(1), 1–11.

Chen, S. Y.-C., Wei, T.-C., Zhang, C., Yu, H., & Yoo, S. (2021). Hybrid quantum-classical graph convolutional network. *arXiv preprint arXiv:2101.06189*.

Crooks, G. E. (2019). Gradients of parameterized quantum gates using the parameter-shift rule and gate decomposition. *arXiv preprint arXiv:1905.13311*.

De Luca, G. (2022). A survey of nisq era hybrid quantum-classical machine learning research. *Journal of Artificial Intelligence and Technology*, *2*(1), 9–15.

Denton, E. L., Zaremba, W., Bruna, J., LeCun, Y., & Fergus, R. (2014). Exploiting linear structure within convolutional networks for efficient evaluation. *Advances in neural information processing systems*, *27*.

Dobson, P. D., & Doig, A. J. (2003). Distinguishing enzyme structures from non-enzymes without alignments. *Journal of molecular biology*, *330*(4), 771–783.

Goto, T., Tran, Q. H., & Nakajima, K. (2021). Universal approximation property of quantum machine learning models in quantum-enhanced feature spaces. *Physical Review Letters*, *127*(9), 090506.

Han, X., Gao, H., Pfaff, T., Wang, J.-X., & Liu, L. (2021). Predicting physics in mesh-reduced space with temporal attention. *International Conference on Learning Representations*.

Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of*

Uncertainty, Fuzziness and Knowledge-Based Systems, *6*(02), 107–116.

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, *2*(5), 359–366.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *International conference on machine learning*, 448–456.

Lloyd, S., Schuld, M., Ijaz, A., Izaac, J., & Killoran, N. (2020). Quantum embeddings for machine learning. *arXiv preprint arXiv:2001.03622*.

McClean, J. R., Boixo, S., Smelyanskiy, V. N., Babbush, R., & Neven, H. (2018). Barren plateaus in quantum neural network training landscapes. *Nature communications*, *9*(1), 1–6.

Morris, C., Kriege, N. M., Bause, F., Kersting, K., Mutzel, P., & Neumann, M. (2020). Tudataset: A collection of benchmark datasets for learning with graphs. *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*. www.graphlearning.io

Morris, C., Ritzert, M., Fey, M., Hamilton, W. L., Lenssen, J. E., Rattan, G., & Grohe, M. (2019). Weisfeiler and leman go neural: Higher-order graph neural networks. *Proceedings of the AAAI conference on artificial intelligence*, *33*(01), 4602–4609.

Nakkiran, P., Alvarez, R., Prabhavalkar, R., & Parada, C. (2015). Compressing deep neural networks using a rank-constrained topology.

Patterson, D., Gonzalez, J., Le, Q., Liang, C., Munguia, L.-M., Rothchild, D., So, D., Texier, M., & Dean, J. (2021). Carbon emissions and large neural network training. *arXiv preprint arXiv:2104.10350*.

Pfaff, T., Fortunato, M., Sanchez-Gonzalez, A., & Battaglia, P. (2021). Learning mesh-based simulation with graph networks. *International Conference on Learning Representations*.

Schomburg, I., Chang, A., Ebeling, C., Gremse, M., Heldt, C., Huhn, G., & Schomburg, D. (2004). Brenda, the enzyme database: Updates and major new developments. *Nucleic acids research*, *32*(suppl_1), D431–D433.

Schuld, M., Bergholm, V., Gogolin, C., Izaac, J., & Killoran, N. (2019). Evaluating analytic gradients on quantum hardware. *Physical Review A*, *99*(3), 032331.

Schuld, M., & Petruccione, F. (2021). Quantum models as kernel methods. In *Machine learning with quantum computers* (pp. 217–245). Springer.

Selig, P., Murphy, N., Sundareswaran, A., Redmond, D., & Caton, S. (2021). A case for noisy shallow gate-based circuits in quantum machine learning. *2021 International Conference on Rebooting Computing (ICRC)*, 24–34.

Shor, P. W. (1999). Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, *41*(2), 303–332.

Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of big data*, *6*(1), 1–48.

Thomas, D. J., Whittaker, M. T., Bright, G. W., & Gao, Y. (2011). The influence of mechanical and co2 laser cut-edge characteristics on the fatigue life performance of high strength automotive steels. *Journal of Materials Processing Technology*, *211*(2), 263–274.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. (2023). Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., & Bengio, Y. (2017). Graph attention networks. *stat*, *1050*, 20.

Verdon, G., McCourt, T., Luzhnica, E., Singh, V., Leichenauer, S., & Hidary, J. (2019). Quantum graph neural networks. *arXiv preprint arXiv:1909.12264*.

Welling, M., & Kipf, T. N. (2016). Semi-supervised classification with graph convolutional networks. *J. International Conference on Learning Representations (ICLR 2017)*.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Philip, S. Y. (2020). A comprehensive survey on graph neural networks. *IEEE Transactions on neural networks and learning systems*, *32*(1), 4–24.

Xu, K., Hu, W., Leskovec, J., & Jegelka, S. (2019). How powerful are graph neural networks? *International Conference on Learning Representations*.

Zhang, C., Bengio, S., Hardt, M., Recht, B., & Vinyals, O. (2021). Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, *64*(3), 107–115.

Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., & Sun, M. (2020). Graph neural networks: A review of methods and applications. *AI open*, *1*, 57–81.