

# Quantum Graph Attention Networks: Trainable Quantum Encoders for Inductive Graph Learning

Arthur M. Faria<sup>1</sup>, Mehdi Djellabi<sup>2</sup>, Igor O. Sokolov<sup>1,2</sup>, and Savvas Varsamopoulos<sup>1</sup>

<sup>1</sup>Pasqal NL, Fred. Roeskestraat 100, 1076 ED Amsterdam, Netherlands

<sup>2</sup>Pasqal SAS, 24 Av. Emile Baudot, 91120 Palaiseau, France

We introduce Quantum Graph Attention Networks (QGATs) as trainable quantum encoders for inductive learning on graphs, extending the Quantum Graph Neural Networks (QGNN) framework introduced in [1]. QGATs leverage parameterized quantum circuits to encode node features and neighborhood structures, with quantum attention mechanisms modulating the contribution of each neighbor via dynamically learned unitaries. This allows for expressive, locality-aware quantum representations that can generalize across unseen graph instances. We evaluate our approach on the QM9 dataset, targeting the prediction of various chemical properties. Our experiments compare classical and quantum graph neural networks—with and without attention layers—demonstrating that attention consistently improves performance in both paradigms. Notably, we observe that quantum attention yields increasing benefits as graph size grows, with QGATs significantly outperforming their non-attentive quantum counterparts on larger molecular graphs. Furthermore, for smaller graphs, QGATs achieve predictive accuracy comparable to classical GAT models, highlighting their viability as expressive quantum encoders. These results show the potential of quantum attention mechanisms to enhance the inductive capacity of QGNN in chemistry and beyond.

Arthur M. Faria: [arthur.faria2@gmail.com](mailto:arthur.faria2@gmail.com)

Mehdi Djellabi: [mehdi.djellabi@pasqal.com](mailto:mehdi.djellabi@pasqal.com)

Igor O. Sokolov: [i.sokolov@pasqal.com](mailto:i.sokolov@pasqal.com)

Savvas Varsamopoulos: [svarsamo@gmail.com](mailto:svarsamo@gmail.com)

## 1 Introduction

Graphs are a fundamental data structure for modeling relational systems, where entities (nodes) are connected by pairwise interactions (edges). This representation naturally arises in a wide range of domains, including chemistry (molecular structures) [2], social networks [3], Transportation & Logistics [4], Electrical Grids & Circuits [5], Communication Networks, Finance and Economics [6], and many more. Traditional machine learning models often struggle to process such non-Euclidean data due to their irregular topology. To address this, Graph Neural Networks (GNNs) have emerged as a powerful class of models that learn over graph-structured inputs by iteratively aggregating and transforming information from a node’s neighborhood. By capturing both local structure and node features, GNNs enable tasks such as node classification, link prediction, and graph-level regression. Among their many variants, models like Graph Convolutional Networks (GCNs) [7], Graph Attention Networks (GATs) [8], and GraphSAGE [9] have demonstrated strong performance in both transductive and inductive settings, making GNNs a key building block in modern geometric deep learning.

While classical GNNs have achieved remarkable success, their scalability and expressiveness can be limited by the classical nature of their computation [10], especially when modeling systems with inherent quantum structure, such as molecules or quantum materials. Quantum Graph Neural Networks (QGNNs) aim to address this by leveraging the computational power of parameterized quantum circuits to encode and process graph data in a quantum-enhanced latent space. By embedding node and edge features into quantum states and performing learnable unitary transformations, QGNNs can cap-

ture complex, high-dimensional correlations that may be intractable for classical models [11, 12].

At our previous work [1], we adapted the core idea of the classical GraphSAGE framework to the quantum setting. In this architecture, each node’s feature vector is encoded into a quantum state, and message passing is performed via parameterized quantum circuits that aggregate information from neighboring nodes. The quantum circuit realizes a Quantum Graph Convolutional Network (QGCN), inspired by prior quantum convolutional architectures [13]. The aggregation follows a fixed scheme, such as mean or sum, mirroring classical GraphSAGE, but the transformation and embedding of the aggregated message is executed within a quantum processor using trainable unitary operations. This quantum aggregation procedure is repeated across multiple layers, enabling the model to capture multi-hop dependencies in the graph while remaining inductive in nature. Although this architecture does not incorporate any dynamic weighting of neighbors, it establishes a strong and well-defined foundation for evaluating the effect of quantum-enhanced representation learning.

Attention mechanisms are fundamental in classical [8, 14–17] and quantum [18–25] machine learning, demonstrating broad effectiveness. Building on the QGNN framework [1], we extend our model by incorporating attention mechanisms, inspired by the effectiveness of Graph Attention Networks (GATs) in classical deep learning [8]. GATs improve graph neural networks by computing learnable, context-dependent attention coefficients that weight the influence of neighboring nodes during message aggregation. This allows the model to differentiate between structurally similar but semantically distinct neighborhoods, addressing a key limitation of standard GNNs that treat all neighbors uniformly or heuristically. In our approach however, we adapt this concept to the quantum setting by introducing Quantum Graph Attention Networks (QGATs)—an architecture that integrates self-attention mechanisms with parameterized quantum circuits. During training, QGATs dynamically learn to assign attention weights to neighboring nodes based on their features and their relationship to the target node. These attention weights are then used to modulate the behavior of trainable quantum circuits responsible

for aggregating and updating node representations. By doing so, QGATs enable the quantum model to selectively emphasize informative substructures within the graph, enhancing its capacity to model complex dependencies and localized interactions. This combination of quantum processing and attention-driven adaptability makes QGATs especially effective for learning over molecular graphs, where both node-specific properties and relational context are critical for accurate prediction.

The main contributions of this work are:

1. We propose QGAT, a novel framework that augments QGNNs with attention mechanisms and implement it in PYTORCH [26] using QADENCE [27]. In this approach, graph-structured data is embedded through tunable feature maps in quantum circuits, parameterized by attention weights.
2. We evaluate QGAT on the QM9 molecular property prediction benchmark, reporting training results for classical GNNs and GATs alongside QGNNs and our proposed framework. We consider two aggregation strategies: (i) a single shared model across all steps and (ii) step-specific models. Our results demonstrate that QGNNs and QGATs can achieve predictive accuracy on par with their classical counterparts.
3. In our numerical experiments, we show that QGATs, through the incorporation of attention mechanisms, consistently achieve higher accuracy than QGNNs on the QM9 dataset.

The remainder of this paper is structured as follows. Section 2 introduces the problem setting and describes the dataset used in our experiments. Section 3 provides an overview of GATs. Section 4 details the architecture of the proposed quantum counterpart, QGATs. Section 5 presents the experimental results and performance analysis. Finally, Section 6 concludes the paper with a summary of our findings and discusses potential directions for future work.

## 2 Molecular Dataset: QM9

The QM9 dataset [28] is a widely used benchmark in computational chemistry for evaluating machine learning models. It was constructed from

the GDB-17 database [29] and contains small organic molecules with up to nine heavy atoms such as carbon, oxygen, nitrogen, and fluorine. The dataset provides a range of properties computed using Density Functional Theory (DFT) at the B3LYP/6-31G(2df,p) [30, 31] level of theory, including molecular geometries, total energies, electronic properties, and thermodynamic quantities such as atomization energies, dipole moments, and HOMO–LUMO gaps. Molecular properties such as LUMO (Lowest Unoccupied Molecular Orbital) and HOMO (Highest Occupied Molecular Orbital) energies are indicators of a molecule’s chemical reactivity and stability. As these are global molecular-level properties, we frame LUMO energy prediction as a node regression task using atomic features.

In this investigation, each molecule is represented as a graph, where nodes encode atomic properties, and edges indicate connectivity between atoms. For each molecule in the QM9 dataset, we extract atom features and the adjacency matrix. However, only the atom features are used as classical input to the model, while the adjacency matrix specifies the set of atom features to be used. In fact, our model uses seven atomic features: atomic number, chirality, degree, formal charge, radical electrons, hybridization state, and scaled mass (indices 0-5, 10). We omit four properties (indices 6-9: aromaticity, hydrogen count, ring membership, and valence) as their values are negligible (effectively zero) in our samples. This framework successfully connects local atomic characteristics with emergent molecular behavior.

### 3 Classical Graph Neural Networks

GNNs are deep learning models that operate on graph-structured data, leveraging node features and topology to learn representations. Unlike traditional neural networks, GNNs handle non-Euclidean structures. Many follow the Message Passing Neural Network (MPNN) framework, where each node aggregates information from neighbors through message and update functions over several steps. A readout function may then compute graph-level outputs. Notable models like GCNs [7], GraphSAGE [9], GATs [8], and GINs [32] fall under this framework, with applications in chemistry [33], social networks, and rec-

ommendation systems [34].

Inspired by convolutional neural networks for image processing [35], GNNs generalize the convolution operation to graph-structured data. The layer-wise propagation rule for Graph Convolutional Networks (GCNs), introduced in [7]. However, GCNs scale poorly to large graphs due to the cost of multiplying the normalized adjacency matrix with node features at each layer. They operate in a *transductive* setting, requiring the full graph in memory, thus preventing mini-batch training and limiting generalization. These constraints make GCNs impractical for large-scale applications, prompting the development of more scalable models.

Among these alternatives, Ref. [9] proposes GraphSAGE, a general inductive framework that addresses GCN limitations by enabling inference on unseen graph data. Essentially, it samples a fixed number of neighbors at each hop and applies an aggregation function to generate local embeddings, which can be concatenated with node features for richer representations. By stacking hops, the method captures multi-hop structural information while supporting mini-batching and GPU training. This sampling–aggregation scheme preserves local structure and scales to large graphs as seen in Fig. 1. Hence, we adopt GraphSAGE as the classical GNN baseline against which to compare our quantum-enhanced models. We next present the mathematical formulation of GraphSAGE, focusing on embedding generation and propagation.

#### 3.1 The GraphSAGE Model

In GraphSAGE, embedding generation involves sampling and aggregation. At layer  $l = 0$ , the message for node  $v$  is  $\mathbf{h}_v^{(0)} = \mathbf{x}_v$ , where  $\mathbf{x}_v$  are its input features. The sampling step gathers features from neighbors  $u$  of  $v$  using the computational graph. For inductive learning, the message at layer  $l$  for neighbor  $u$  of  $v$  is given by

$$\mathbf{h}_{\mathcal{N}(v)}^{(l)} = \text{AGG}^{(l)} \left( \{ \mathbf{h}_u^{(l-1)}, \forall u \in \mathcal{N}(v) \} \right). \quad (1)$$

Here, AGG denotes a general vector function [36, 37]. We adopt the MEAN aggregator for its simplicity and permutation invariance. It computes the element-wise mean of each neighbor node embedding  $\mathbf{h}_u^{(l-1)}$ , yielding  $\mathbf{h}_{\mathcal{N}(v)}^{(l)} = \text{MEAN}(\{ \mathbf{h}_u^{(l-1)} \mid$

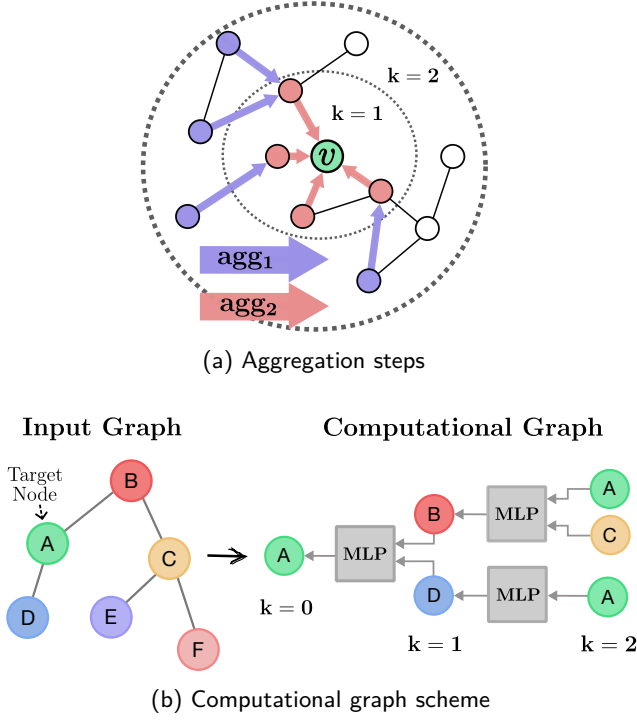


Figure 1: Illustration of the GNN framework. (a) Message passing with aggregations (*agg1*, *agg2*) to the target node  $v$  over  $k = 2$  hops. (b) Transformation of an input graph (left) into a computation graph (right), where nodes represent feature embeddings and edges denote information flow. Gray boxes indicate Multi-Layer Perceptrons (MLPs) that aggregate neighborhood features through modular transformations.

$\forall u \in \mathcal{N}(v)$ }, similar to the transductive setting [7]. The target node embedding is then formed by concatenating  $\mathbf{h}_v^{(l-1)}$  with  $\mathbf{h}_{\mathcal{N}(v)}^{(l)}$ , unlike standard GCNs. The final embedding of  $v$  is given by

$$\mathbf{h}_v^{(l)} = \sigma \left( \mathbf{W}^{(l)} \cdot [\mathbf{h}_v^{(l-1)} \parallel \mathbf{h}_{\mathcal{N}(v)}^{(l)}] \right) \quad (2)$$

where  $[\cdot \parallel \cdot]$  denotes concatenation. At the final layer  $L$ , the embedding  $\mathbf{h}_v^{(L)}$  captures both the node’s features and its structural context. By leveraging feature information, GraphSAGE learns the node’s role in the graph. This process can be enhanced with attention mechanisms [14], which weigh neighbors by relevance, as in ref. [8], discussed in the next section.

### 3.2 Graph Attention Networks (GATs)

GATs address the uniform weighting in early GNNs by introducing masked self-attention [14], allowing nodes to assign data-dependent importance to neighbors. This emphasizes relevant con-

nections while down-weighting noisy ones. Attention is computed only over existing edges, retaining the parallelism and receptive-field control of Transformers, while avoiding expensive eigendecompositions. The approach scales linearly with the number of edges and supports inductive learning. Formally, for each neighbor  $u \in \mathcal{N}(v)$ , attention coefficients are computed as [8]

$$e_{vu}^{(l)} = a(\mathbf{h}_v^{(l)}, \mathbf{h}_u^{(l)}), \quad (3)$$

where the constant function  $a(\cdot, \cdot)$  defines attentional mechanism between embeddings  $\mathbf{h}_v^{(l)}$  and  $\mathbf{h}_u^{(l)}$ . The attention coefficients are then normalized across all neighbors of  $v$  using the softmax function as

$$\alpha_{vu}^{(l)} = \frac{\exp(e_{vu}^{(l)})}{\sum_{w \in \mathcal{N}(v)} \exp(e_{vw}^{(l)})}, \quad (4)$$

Fig. 2 schematically illustrates how attention weights are assigned to the neighboring nodes of a target node. The updated embedding of the weighted aggregation of neighbors of  $v$  is hence obtained as

$$\mathbf{h}_{\mathcal{N}(v)}^{(l)} = \sigma \left( \sum_{u \in \mathcal{N}(v)} \alpha_{vu}^{(l)} \mathbf{h}_u^{(l-1)} \right) \quad (5)$$

Here again,  $\sigma(\cdot)$  denotes a nonlinearity such as ReLU. The updated representation  $\mathbf{h}_v^{(l)}$  is obtained by concatenating the outputs from all attention heads  $\mathbf{h}_{\mathcal{N}(v)}^{(l)}$  with the self-embedding  $\mathbf{h}_v^{(l-1)}$ , as shown in Eq. (2). This extends naturally to multi-head attention, where each head uses its own projection matrix  $\mathbf{W}^{(l)}$ .

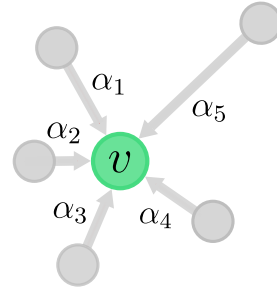


Figure 2: Schematic representation of attention weight assignment to the neighbors of a target node. Each neighboring node is assigned a weight  $\alpha_1, \dots, \alpha_7$  indicating its relative importance during message aggregation.

GATs leverage masked self-attention to down-weight noisy or high-degree nodes and provide



interpretable importance scores. Since updates rely only on local neighborhoods, GATs support inductive learning across different graphs. To scale further, GraphSAGE style neighborhood sampling can be combined with GATs, forming an implicit ensemble that stabilizes training. This enables efficient mini-batching and reduces memory usage, maximizing GPU efficiency. Next, we discuss our quantum version of the GAT model.

## 4 Quantum GATs

QGATs leverages quantum GraphSAGE, introduced in Ref. [1], by combining a trainable quantum feature map, serving as an attention mechanism, with a QGCN-inspired Ansatz. The feature map uses input-dependent unitaries to encode classical data into quantum states with learnable neighbor weights, which are then processed by quantum convolution and pooling layers [1].

### 4.1 Trainable quantum feature map as attention mechanism

A quantum model is trained on the basis of the classical dataset that is provided. The quantum model consists of a variational quantum circuit defined on a qubit register. The qubit register is initialized in the zero state  $|0\rangle^{\otimes N}$  and a quantum feature map (FM)  $\mathcal{F}$  is used to map coordinates from classical input values  $\mathbf{x}$  to a distinct place in the Hilbert space of the qubit register.

By assigning attention coefficients  $\alpha_{vu}$  to each neighbor  $u$  of a target node  $v$ , the classical node feature  $\mathbf{x}_u$  is weighted by  $\alpha_{vu}$ . The feature map  $\mathcal{F}$  is then realized by a parametrized unitary  $U_{\mathbf{x}^{\alpha_{vu}}}$  acting on the initial state  $|0\rangle^{\otimes N}$ . Concretely,

$$\mathbf{x}_u^{\alpha_{vu}} \mapsto |\mathcal{F}(\mathbf{x}_u^{\alpha_{vu}})\rangle = U_{\mathbf{x}^{\alpha_{vu}}} |0\rangle^{\otimes N}, \quad (6)$$

with  $\mathbf{x}_u^{\alpha_{vu}} \equiv \alpha_{vu}\mathbf{x}_u$ . We refer to the state  $|\mathcal{F}(\mathbf{x}_u^{\alpha_{vu}})\rangle$ , generated by the feature map  $\mathcal{F}$ , as the quantum attention layer, with trainable coefficient  $\alpha_{vu}$ . Note that, the layer index ( $l$ ) is omitted here, since the weights only parametrize the FM in Eq. (5), which has depth 1. Hence, the initial state becomes

$$\rho_u^{\alpha_{vu}} = |\mathcal{F}(\mathbf{x}_u^{\alpha_{vu}})\rangle\langle\mathcal{F}(\mathbf{x}_u^{\alpha_{vu}})| \quad (7)$$

Trainable feature maps have previously been explored in [38, 39]. Unlike prior QGNNs [1], the

input state  $\rho^\alpha$  is parametrized by trainable attention weights  $\alpha$ . This formulation allows each node to adaptively attend to its neighbors by assigning distinct probabilities to their edges. This enables the Ansatz to perform flexible aggregation that naturally adapts to varying node degrees without assuming fixed or uniform connectivity.

For a set of  $N$  neighbors of node  $v$ , the overall input state becomes

$$\rho_{\text{in}}^{\alpha_v} = \bigotimes_{w=1}^N \rho_w^{\alpha_{vw}}, \quad (8)$$

where  $\alpha_v := \{\alpha_{vw}\}_{w=1}^N$  and  $\rho_w^{\alpha_{vw}}$  represents the state of the neighboring node  $w$  of  $v$  parametrized by the corresponding attention weight  $\alpha_{vw}$ . The feature map  $\mathcal{F}$  is applied uniformly across all neighbors.

### 4.2 Ansatz-QGCN circuit

Inspired by Ref. [1, 13], the QGCN circuit is described as a composition of quantum convolutional and pooling channels as

$$\Phi_{\theta_v} = \bigcirc_{l=1}^L \left( P_l \circ C_l^{\theta_v^l} \right), \quad (9)$$

where  $\theta_v := \{\theta_v^l\}_{l=1}^L$  and  $\circ$  denotes a single function composition, and  $\bigcirc_{l=1}^L$  represents the composition of  $L$  functions applied sequentially. Each QGCN layer  $l$  comprises a quantum convolutional layer  $C_l^{\theta^l}$ , with  $\theta^l$  being the convolution parameters, followed by a quantum pooling layer  $P_l$ . The alternating structure of the QGCN circuit processes and simplifies the input quantum state, starting with  $C_1$  and ending with  $P_L$ . The convolutional layer acts on an even number of qubits and preserves the size of the quantum register defined as

$$C_l^{\theta^l}(\cdot) = \bigcirc_{j=1}^r \left( \bigotimes_{(i,i+1) \in S(j)} W_l^{(i,i+1)}(\theta^l) \right) (\cdot), \quad (10)$$

with  $W_l$  processing adjacent qubit pairs  $(i, i+1)$ , as displayed in Fig. 3a. For  $n_l$  qubits in the layer  $l$ , the operator  $W_l$  acts on nearest-neighbor pairs in an alternating fashion: at even steps  $j$ , it applies to pairs  $(0, 1), (2, 3), \dots$ ; at odd steps  $j$ , it applies to  $(1, 2), (3, 4), \dots$ . Formally, the pairs are  $S(j) = \{(i, i+1) \mid i \equiv j \pmod{2}, 0 \leq i \leq n_l - 2\}$ . Using the decomposition proposed in Ref. [40],  $W_l$  is

composed of 3 CZ gates, 3 single-qubit rotations, and 4 general  $R^G$  gates, where

$$R^G(\theta^l) = e^{-iX\theta_1^l/2}e^{-iZ\theta_2^l/2}e^{-iX\theta_3^l/2}. \quad (11)$$

Each  $C_l$  layer has depth  $r$  (e.g.,  $r_1, r_2$  in a two-layer QGCNs). Conversely, the pooling layer  $P_l$  reduces the size of the quantum register by tracing out selected qubits as

$$P_l = \text{Tr}_i[(\cdot)], \quad (12)$$

where  $i$  denotes the qubits traced out at layer  $l$ . In practice, this discards half of the qubits at each pooling step, i.e.,  $i = \{j \in \{0, 1, \dots, n_l - 1\} \mid j \equiv 0 \pmod{2}\}$ . Generally, the reduction is implemented by measuring a subset of qubits and applying classically controlled Pauli gates on their neighbors. As a design choice, the principle of deferred measurement [41] is applied. Thus, the pooling layers first employ controlled operations, while (partial) measurements are postponed until the end of the computation. At the final QGCN layer  $L$ , the output state produced by the quantum aggregation,  $\rho_{\text{out}}$ , is defined as

$$\rho_{\text{out}}^{\alpha_v, \theta_v} = \Phi_{\theta_v}(\rho_{\text{in}}^{\alpha_v}) \quad (13)$$

$$= \bigotimes_{w=1}^N \Phi_{\theta_v}(\rho_{\text{in}}^{\alpha_{vw}}) \quad (14)$$

For the full circuit architecture, see Fig. 3b.

### 4.3 Embedding generation and model training

By employing the trainable FM to initialize the weighted qubit states and the QGCN as the aggregator, the classical embedding for the neighbors  $u$  of  $v$  is obtained through measurements of the observable  $\mathcal{O}$ , expressed as

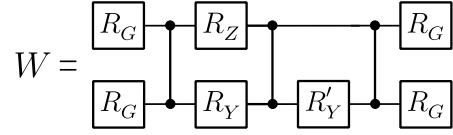
$$\mathbf{h}_{\mathcal{N}(v)}^{(L)}(\alpha_v, \theta_v) = \text{Tr}[\rho_{\text{out}}^{\alpha_v, \theta_v} \mathcal{O}] \quad (15)$$

$$= \langle \mathcal{O} \rangle_{\rho_{\text{out}}^{\alpha_v, \theta_v}}. \quad (16)$$

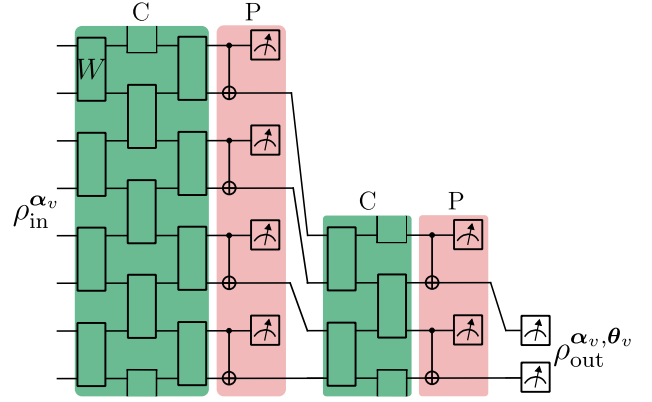
Note that the embedding is generated by the same QGCN regardless of the number of neighboring nodes  $u$ . It can be further concatenated with the self-embedding of node  $v$  to yield its updated representation as

$$\mathbf{h}_v^{(L)}(\alpha_v, \theta_v) = \sigma \left( \left[ \mathbf{h}_{\mathcal{N}(v)}^{(L)} \parallel \mathbf{h}_v^{(1)} \right] \right). \quad (17)$$

As the QGCN circuit has  $L$  layers, the initial embedding  $\mathbf{h}_v^{(1)}$  can be only concatenated after



(a)  $W$  definition.



(b) QGCN circuit.

Figure 3: Schematic representation of the QGCN circuit and convolutional unitary  $W$  acting on neighboring qubits. (a) Definition of the convolutional cell  $W$ , which consists of CZ gates applied to alternating qubit pairs, enclosed by parameterized single-qubit rotations  $R_G$ . The single qubit gates:  $R_Z, R_Y$ , and  $R'_Y$  also are parametrized with different tunable angles. (b) The QGCN processes an  $n$ -qubit input state  $\rho_{\text{in}}^{\alpha_v}$  through  $L$  layers, each consisting of alternating convolution (C) and pooling (P) operations ( $L = 2$  in this case). The C layers apply unitary transformations  $W$  to qubit pairs and P layers reduce dimensionality by tracing out half the qubits, ultimately yielding the output state  $\rho_{\text{out}}^{\alpha_v, \theta_v}$ .

the quantum measurement. In addition, the concatenation operation used here is a simple vector operation that does not involve any learnable parameters. Fig. 4 summarizes the proposed QGAT framework.

For completeness, in a multi-head setting, where multiple target nodes  $\nu := \{v_k\}_{k=1}^K$  are considered, the output embedding reads

$$\mathbf{h}_\nu(\alpha, \theta) = \sigma \left( \left[ \bigparallel_{k=1}^K \mathbf{h}_{v_k}^{(L)} \right] \right), \quad (18)$$

where  $\theta := \{\theta_{v_k}\}_{k=1}^K$  and  $\alpha := \{\alpha_{v_k}\}_{k=1}^K$ . For simplicity, we assume target nodes have the same number of neighbors and that all QGCN Ansätze employed have a uniform depth  $L$  across hops.

To this point, we have focused on the unsupervised node-level task within the graph structure. By repeatedly propagating information until reaching the designated target node, the re-

sulting embedding captures both the intrinsic features of the node and its structural dependencies within the graph. The final embedding yields the predicted graph-level output  $\bar{y}(\tilde{\theta}_v)$ , where  $\tilde{\theta}_v = \{\alpha_v, \theta_v\}$  represents the tunable parameters of the QGAT model. This allows us to run a supervised learning, by comparing the prediction to the true target value  $y$ . The training error is quantified by the loss function  $\mathcal{L}$ , and the parameters  $\tilde{\theta}_v$  are optimized by minimizing  $\mathcal{L}(\bar{y}(\tilde{\theta}_v), y)$  as

$$\tilde{\theta}_v^* = \arg \min_{\tilde{\theta}_v} \mathcal{L}(y(\tilde{\theta}_v), y). \quad (19)$$

The parameters are efficiently minimized using standard gradient-based optimization algorithms. Successful training results in a negligible prediction error, i.e.,  $|\bar{y}(\tilde{\theta}_v) - y| \approx 0$ . We next benchmark our proposed framework on the QM9 dataset, evaluating its performance relative to its classical counterpart in molecular property prediction.

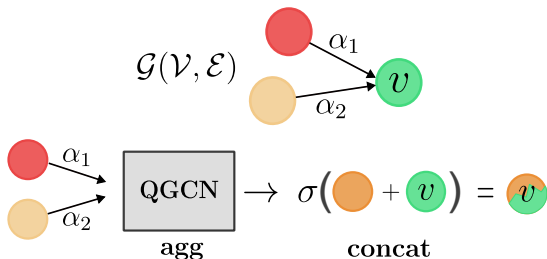


Figure 4: Schematic of the proposed QGAT framework using a QGCN circuit as quantum aggregator, which contains  $L$  sets of C and P layers. Given a graph  $\mathcal{G}(\mathcal{V}, \mathcal{E})$ , node features from connected nodes (red, yellow) are sampled and processed by the QGCN to generate the neighbor embeddings (orange)  $\mathbf{h}_{\mathcal{N}(v)}^{(L)}(\alpha_1, \alpha_2, \theta_v)$ . These are concatenated classically with the target node’s self-embedding (green) through a simple vector operation and transformed by a non-linear function  $\sigma$  to yield the updated embedding  $\mathbf{h}_v^{(L)}(\alpha_1, \alpha_2, \theta_v)$ .

## 5 Results

We now present the key findings from our implementation. Our evaluation compares the performance of quantum-enhanced graph neural networks (QGNNs, QGATs) against their classical counterparts (GNNs, GATs) across molecules of varying sizes. We use GNN to refer to the GraphSAGE model.

### 5.1 Training details

The performance of each quantum and classical model is evaluated using Smooth L1 Loss and the  $R^2$  score in separate experiments on the same small subset of molecular samples. The Smooth L1 Loss is selected for its robustness to outliers and stability, particularly advantageous for small, high-variability datasets. Training employs the Adam optimizer ( $\beta_1 = \beta_2 = 0.9$ ) with a decaying learning rate initialized at 0.03.

Molecules are processed in two stages: (i) an unsupervised atom-level aggregation, where the model aggregates connected node features according to the adjacency matrix of the molecule to form a molecular representation, and (ii) a supervised molecule-level step, where this representation is used to predict the target value. Here, the loss function compares predictions with true targets, guiding the training of the model to capture both local and global structural information. Algorithm 1 presents the high-level implementation of the previous steps. In the case of the quantum models  $\mathcal{M}$ , the FM and Ansatz are incorporated directly into their definition

---

#### Algorithm 1 (Q)GATs on the QM9 dataset

---

**Require:** Batch of molecules  $G$  with adjacencies  $A$ , atom features  $F$ , targets  $T$ , model  $\mathcal{M}$ , attention weights  $\alpha$

**Ensure:**  $T_{\text{true}}, T_{\text{pred}}, \text{loss}$

```

1:  $\text{loss} \leftarrow 0.0, T_{\text{pred}} \leftarrow [], T_{\text{true}} \leftarrow []$ 
2: for each mol  $(A, F, T, \alpha)$  in batch do
3:    $\text{prev\_out} \leftarrow \mathbf{0}, \text{atom\_outs} \leftarrow []$ 
4:   for each node  $v$  in  $G$  do
5:      $a\_feat \leftarrow F[A[v]] \times \alpha[v]$ 
6:      $x \leftarrow \text{concat}(a\_feat, \text{prev\_out})$ 
7:      $\text{out} \leftarrow \mathcal{M}(x)$ 
8:      $\text{prev\_out} \leftarrow \text{out}$ ,
9:     Append  $\text{out}$  to  $\text{atom\_outs}$ 
10:  end for
11:   $\text{mol\_out} \leftarrow \text{mean}(\text{atom\_outs})$ 
12:  Append  $\text{mol\_out}$  to  $T_{\text{pred}}$ 
13:  Append  $T$  to  $T_{\text{true}}$ 
14:   $\text{loss} += \text{criterion}(\text{mol\_out}, T) / |\text{batch}|$ 
15: end for
16: return  $T_{\text{true}}, T_{\text{pred}}, \text{loss}$ 
```

---

In the proposed algorithm, all hops  $k$  up to the number of atoms are considered; for instance, a 9-atom molecule undergoes 9 hops, with em-

Model	samples	inputs/qubits	FM	$r$ /hidden layers	params
Classical	30	8	-	[8]	209
Quantum	30	8	Fourier	[3, 1, 1]	218
Classical (multi)	40	8	-	[2, 2]	1675
Quantum (multi)	40	8	Fourier	[1]	1700

Table 1: The table presents the hyperparameters of the classical and quantum models. For each model, we have: number of training samples, number of input features (classical) or qubits (quantum), feature map type (quantum only), either the  $r$  configuration (quantum) or hidden layer architecture (classical), and total trainable parameters (params). The "multi" suffix denotes models combining multiple sub-models, e.g., quantum (multi) uses separate circuits for each graph hop, and similarly for classical (multi). Sample sizes were determined by model complexity: single-model architectures used 30 samples, whereas multi-model configurations employed 40.

beddings propagated from lower- to higher-index atoms, yielding a single-branch computational graph with one final head. Classical GNNs and GATs are implemented in PyTorch [26], while QGNNs and QGATs use QADENCE [27]. The number of inputs/qubits are 8, matching the 7 node feature dimension plus one (accounting for the mean self-embedding of the target node)<sup>1</sup>. QM9 data are encoded via either a trainable or standard Fourier Feature Map [42, 43], depending on attention layers usage. Measurements assess local magnetization with tunable parameters  $\omega := \{\omega_i\}_{i=1}^N$ , and the average total magnetization is defined as an expectation value:

$$\langle \mathcal{O} \rangle = \frac{1}{N} \sum_{i=1}^N \omega_i \langle Z^{(i)} \rangle. \quad (20)$$

The hyperparameters of the quantum and classical models used in the results are detailed in Table 1. Unless otherwise specified, the models hyperparameters and evaluation metrics: Smooth L1 Loss and  $R^2$  score, remain unaltered.

## 5.2 Discussion

We evaluate the performance of four graph learning frameworks: GNNs, GATs, QGNNs, and our proposed QGATs, on subsets of the QM9 dataset grouped by molecular size: molecules with up to 9, 16, 20, and 25 atoms. For each subset, we randomly sample 30/40 molecules and report the average performance in terms of training loss and

$R^2$  score. Two aggregation strategies are considered: (i) using a single shared model across all aggregation steps (or hops), and (ii) employing a distinct model at each hop, thereby increasing the modularity of the architecture. Table 2 presents the complete results for all frameworks under both aggregation strategies and across the defined molecular size regimes.

The following analysis examines performance across frameworks and highlights the benefits of attention mechanisms and trainable FMs, particularly in quantum contexts. In the single-model setting, where a single GCN or QGCN is reused across all aggregation steps, classical models consistently outperform quantum ones. As shown in Figure 5, GATs achieve near-perfect scores across all molecule sizes, with GCNs also performing strongly, albeit with a slight performance drop as molecule size increases. Quantum models, by contrast, exhibit a more noticeable decline in performance as molecular complexity increases. QGCNs, in particular, struggle to generalize under this configuration. In contrast, QGATs consistently outperform QGCNs across all size regimes. This gap highlights the value of attention mechanisms in both classical and, more importantly, quantum graph learning models. The ability to adaptively weigh neighbor contributions appears to guide the quantum encoding process more effectively by emphasizing structurally or chemically relevant information.

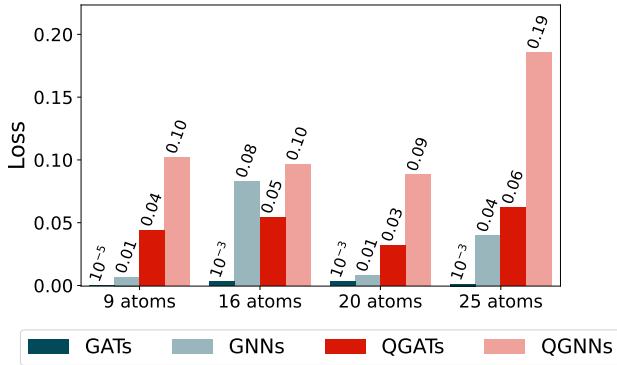
In the multiple-model setting, where each aggregation step is assigned a distinct, shallow classical or quantum models, we observe substantial performance improvements across all architectures. As illustrated in Figure 6, both classical GATs and GNNs achieve  $R^2$  scores of 1.0, not showing a big improvement in the classical sce-

<sup>1</sup>Features such as atomic number, chirality, degree, formal charge, radical electrons, hybridization, and scaled mass are kept, while aromaticity, hydrogen count, ring membership, and valence are excluded as they are consistently zero in the selected samples.

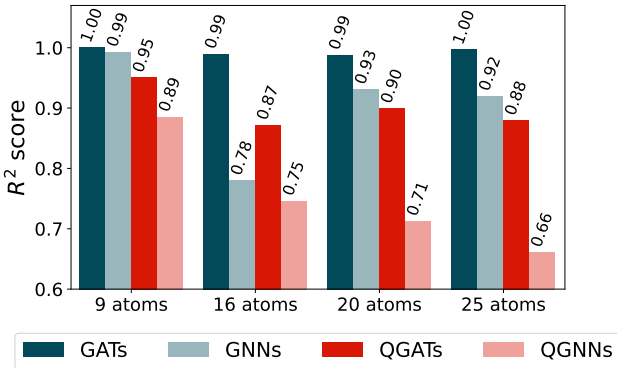


		GNNs		QGNNs		GATs		QGATs	
	Metric	single	multi	single	multi	single	multi	single	multi
$\leq 9$	Loss	0.00676	0.00009	0.1021	0.00032	0.00003	0.00005	0.0441	0.00008
	R <sup>2</sup> score	0.99280	0.99980	0.8853	0.99840	1.00000	0.99920	0.9504	0.99990
$\leq 16$	Loss	0.08291	0.00001	0.0966	0.02753	0.00325	0.00002	0.0544	0.00398
	R <sup>2</sup> score	0.78000	1.00000	0.7463	0.93700	0.98960	0.99990	0.8717	0.98090
$\leq 20$	Loss	0.00810	0.00001	0.0882	0.03209	0.00337	0.000003	0.0321	0.00002
	R <sup>2</sup> score	0.93080	0.99970	0.7127	0.86760	0.98800	1.00000	0.8992	0.99980
$\leq 25$	Loss	0.03968	0.00003	0.1854	0.04720	0.00118	0.00001	0.0620	0.00610
	R <sup>2</sup> score	0.91990	0.99970	0.6610	0.92790	0.99670	0.99990	0.8793	0.98920

Table 2: Comparison of single-model versus multi-model approaches across different graph neural network architectures (GNNs, QGNNs, GATs, QGATs). The table shows loss values and R<sup>2</sup> scores for molecules of varying sizes: from  $\leq 9$  to  $\leq 25$  atoms.

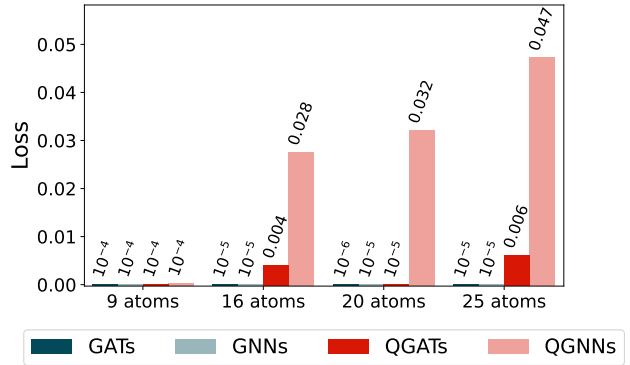


(a) Loss.

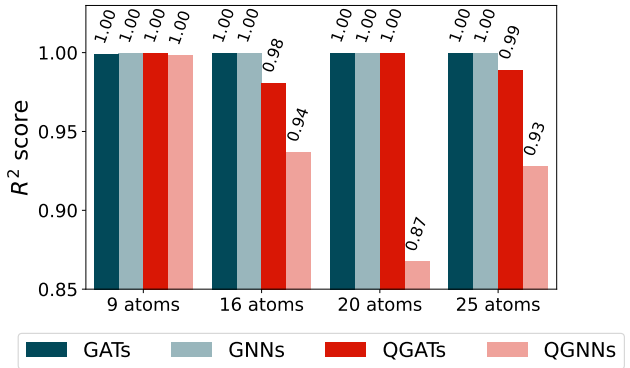


(b) R<sup>2</sup> score.

Figure 5: Training results of single-model approach across different graph neural network architectures (GNNs, QGNNs, GATs, QGATs). The subfigure (a) shows loss values whereas (b) R<sup>2</sup> scores for molecules of varying sizes: from  $\leq 9$  to  $\leq 25$  atoms.



(a) Loss.



(b) R<sup>2</sup> score.

Figure 6: Training results of multi-model approach across different graph neural network architectures (GNNs, QGNNs, GATs, QGATs). The subfigure (a) shows loss values and the subfigure (b) displays R<sup>2</sup> scores for molecules of varying sizes: from  $\leq 9$  to  $\leq 25$  atoms.

nario. However, the quantum frameworks again experience the most significant gains from this modular approach. QGNNs, which previously

underperformed, show now strong improvements across all molecule sizes. QGATs also benefit substantially, confirming the synergy between atten-

tion mechanisms and modular quantum design. Hence, by employing shallow circuits that are less prone to barren plateaus and easier to optimize, it can enhance both the stability and efficiency of the training process in quantum graph machine learning.

## 6 Conclusion

This work introduced QGAT, a quantum graph attention framework that integrates attention mechanisms into quantum graph neural networks for inductive learning via tunable feature maps, where the quantum circuit scales independently of the number of nodes. Through extensive experiments on the QM9 dataset, we demonstrated that attention not only improves classical models but is particularly beneficial in the quantum setting, where it enhances expressivity and task adaptability, enabling QGATs to surpass our QGNN architectures. Furthermore, we showed that modular architectures employing multiple shallow circuits per aggregation step significantly outperform single deep-circuit designs, offering a practical solution to the optimization challenges of quantum models. Together, these findings highlight a promising path toward more scalable and trainable quantum graph learning models.

## References

- [1] Arthur M. Faria, Ignacio F. Graña, and Savvas Varsamopoulos. Inductive graph representation learning with quantum graph neural networks. *arXiv:2503.24111 [quant-ph]*, 2025.
- [2] A. I. Gircha, A. S. Boev, K. Avchaciov, P.O. Fedichev, and A. K. Fedorov. Hybrid quantum-classical machine learning for generative chemistry and drug design. *Scientific Reports*, 13(1), 2023.
- [3] Shashank Sheshar Singh, Sumit Kumar, Sunil Kumar Meena, Kuldeep Singh, Shivansh Mishra, and Albert Y. Zomaya. Quantum social network analysis: Methodology, implementation, challenges, and future directions. *Information Fusion*, 117:102808, 2025.
- [4] Chence Niu, Elnaz Irannezhad, Casey Myers, and Vinayak Dixit. Quantum computing in transport science: A review. *arXiv:2503.21302 [quant-ph]*, 2025.
- [5] Md Habib Ullah, Rozhin Eskandarpour, Honghao Zheng, and Amin Khodaei. Quantum computing for smart grid applications. *IET Generation, Transmission & Distribution*, 16(21):4239–4257, 2022.
- [6] Dylan Herman, Cody Googin, Xiaoyuan Liu, Alexey Galda, Ilya Safro, Yue Sun, Marco Pistoia, and Yuri Alexeev. A survey of quantum computing for finance. *arXiv:2201.02773 [quant-ph]*, 2022.
- [7] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.
- [8] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.
- [9] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.
- [10] S. Abadal, A. Jain, R. Guirado, J. Lopez-Alonso, , and E. Alarcón. Computing graph neural networks: A survey from algorithms to accelerators. *ACM Computing Surveys (CSUR)*, 54(9):1–38, 2021.
- [11] Amira Abbas, David Sutter, Christa Zoufal, Aurélien Lucchi, Alessio Figalli, and Stefan Woerner. The power of quantum neural networks. *Nat Comput Sci.*, 1:403–409, 2021.
- [12] Vojtěch Havlíček, Antonio D. Córcoles, Kristan Temme, Aram W. Harrow, Abhinav Kandala, Jerry M. Chow, and Jay M. Gambetta. Supervised learning with quantum-enhanced feature spaces. *Nature*, 567:209–212, 2019.
- [13] Iris Cong, Soonwon Choi, and Mikhail D. Lukin. Quantum convolutional neural networks. *Nat. Phys.*, 15:1273–1278, 2019.
- [14] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Gar-

- nett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.
- [15] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014.
- [16] Zhaoyang Niu, Guoqiang Zhong, and Hui Yu. A review on the attention mechanism of deep learning. *Neurocomputing*, 452:48–62, 2021.
- [17] Yoon Kim, Carl Denton, Luong Hoang, and Alexander M. Rush. Structured attention networks. In *International Conference on Learning Representations*, 2017.
- [18] Guangxi Li, Xuanqiang Zhao, and Xin Wang. Quantum self-attention neural networks for text classification. *Sci. China Inf. Sci.*, 67:142501, 2024.
- [19] Yuichi Kamata, Quoc Hoan Tran, Yasuhiro Endo, and Hirotaka Oshima. Molecular quantum transformer. *arXiv:2503.21686 [quant-ph]*, 2025.
- [20] Ren-Xin Zhao, Jinjing Shi, and Xuelong Li. QKSAN: A Quantum Kernel Self-Attention Network. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 46(12):10184–10195, December 2024.
- [21] Dominic Widdows, Willie Aboumrad, Do-hun Kim, Sayonee Ray, and Jonathan Mei. Quantum natural language processing. *arXiv:2403.19758 [quant-ph]*, 2024.
- [22] Ethan N. Evans, Matthew Cook, Zachary P. Bradshaw, and Margarite L. LaBorde. Learning with sasquatch: a novel variational quantum transformer architecture with kernel-based self-attention. *arXiv:2403.14753 [quant-ph]*, 2024.
- [23] Naixu Guo, Zhan Yu, Matthew Choi, Aman Agrawal, Kouhei Nakaji, Alán Aspuru-Guzik, and Patrick Rebentrost. Quantum linear algebra is all you need for transformer architectures. *arXiv:2402.16714 [quant-ph]*, 2024.
- [24] Yeqi Gao, Zhao Song, Xin Yang, and Ruizhe Zhang. Fast quantum algorithm for attention computation. *arXiv:2307.08045 [quant-ph]*, 2023.
- [25] El Amine Cherrat, Iordanis Kerenidis, Natansh Mathur, Jonas Landman, Martin Strahm, and Yun Yvonna Li. Quantum Vision Transformers. *Quantum*, 8:1265, February 2024.
- [26] Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, Peter Bell, David Berard, Evgeni Burovski, Geeta Chauhan, Anjali Chourdia, Will Constable, Alban Desmaison, Zachary DeVito, Elias Ellison, Will Feng, Jiong Gong, Michael Gschwind, Brian Hirsh, Sherlock Huang, Kshiteej Kalambarkar, Laurent Kirsch, Michael Lazos, Mario Lezcano, Yanbo Liang, Jason Liang, Yinghai Lu, CK Luk, Bert Maher, Yunjie Pan, Christian Puhersch, Matthias Reso, Mark Saroufim, Marcos Yukio Siraiichi, Helen Suk, Michael Suo, Phil Tillet, Eikan Wang, Xiaodong Wang, William Wen, Shunting Zhang, Xu Zhao, Keren Zhou, Richard Zou, Ajit Mathews, Gregory Chanan, Peng Wu, and Soumith Chintala. PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation. In *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS ’24)*. ACM, April 2024.
- [27] Dominik Seitz, Niklas Heim, João Moutinho, Roland Guichard, Vytautas Abramavicius, Aleksander Wennersteen, Gert-Jan Both, Anton Quelle, Caroline Groot, Gergana Velikova, Vincent Elfving, and Mario Dagrada. Qadence: a differentiable interface for digital and analog programs. *IEEE Software*, PP:1–14, 01 2025.
- [28] Raghunathan Ramakrishnan, Pavlo O Dral, Matthias Rupp, and O Anatole Von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Sci. data*, 1(1):1–7, 2014.
- [29] Lars Ruddigkeit, Ruud Van Deursen, Lorenz C Blum, and Jean-Louis Reymond. Enumeration of 166 billion organic small molecules in the chemical universe database gdb-17. *Journal of chemical information and modeling*, 52(11):2864–2875, 2012.
- [30] Axel D Becke. Density-functional thermochemistry. iii. the role of exact exchange. *J. Chem. Phys.*, 98(7):5648–5652, 1993.
- [31] RHWJ Ditchfield, Warren J Hehre, and John A Pople. Self-consistent molecular-

- orbital methods. ix. an extended gaussian-type basis for molecular-orbital studies of organic molecules. *J. Chem. Phys.*, 54(2):724–728, 1971.
- [32] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.
- [33] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1263–1272. PMLR, 06–11 Aug 2017.
- [34] Le Wu, Xiangnan He, Xiang Wang, Kun Zhang, and Meng Wang. A survey on accuracy-oriented neural recommendation: From collaborative filtering to information-rich recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 35(5):4425–4445, 2023.
- [35] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [36] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [37] R. Qi Charles, Hao Su, Mo Kaichun, and Leonidas J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 77–85, Los Alamitos, CA, USA, July 2017. IEEE Computer Society.
- [38] Seth Lloyd, Maria Schuld, Aroosa Ijaz, Josh Izaac, and Nathan Killoran. Quantum embeddings for machine learning. *arXiv:2001.03622v2 [quant-ph]*, 2020.
- [39] Ben Jaderberg, Antonio A. Gentile, Youssef Achari Berrada, Elvira Shishenina, and Vincent E. Elfving. Let quantum neural networks choose their own frequencies. *Phys. Rev. A*, 109:042421, Apr 2024.
- [40] Farrokh Vatan and Colin Williams. Optimal quantum circuits for general two-qubit gates. *Phys. Rev. A*, 69:032315, 2004.
- [41] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2010.
- [42] Maria Schuld, Ryan Sweke, and Johannes Jakob Meyer. Effect of data encoding on the expressive power of variational quantum-machine-learning models. *Phys. Rev. A*, 103:032430, Mar 2021.
- [43] Oleksandr Kyriienko, Annie E. Paine, and Vincent E. Elfving. Solving nonlinear differential equations with differentiable quantum circuits. *Phys. Rev. A*, 103:052416, May 2021.