Equivariant Quantum Graph Circuits

Péter Mernyei [1 2] **Konstantinos Meichanetzidis** [3] **İsmail İlkan Ceylan** [1]

Abstract

We investigate quantum circuits for graph representation learning, and propose *equivariant quantum graph circuits (EQGCs)*, as a class of parameterized quantum circuits with strong relational inductive bias for learning over graph-structured data. Conceptually, EQGCs serve as a unifying framework for quantum graph representation learning, allowing us to define several interesting subclasses which subsume existing proposals. In terms of the representation power, we prove that the studied subclasses of EQGCs are universal approximators for functions over the bounded graph domain. This theoretical perspective on quantum graph machine learning methods opens many directions for further work, and could lead to models with capabilities beyond those of classical approaches. We empirically verify the expressive power of EQGCs through a dedicated experiment on synthetic data, and additionally observe that the performance of EQGCs scales well with the depth of the model and does not suffer from barren plateu issues.

1. Introduction

In recent years, the field of quantum computing has made significant steps towards practical usefulness, which has sparked increasing interest in many areas, including machine learning (Perdomo-Ortiz et al., 2018; Benedetti et al., 2019). The growing field of quantum machine learning has since led to the development of quantum analogues (or, generalizations) of many types of classical machine learning models, such as feedforward neural networks (Schuld et al., 2014), convolutional neural networks (Cong et al., 2019) and graph neural networks (Verdon et al., 2019).

[1]Department of Computer Science, University of Oxford, Oxford, UK. [2]Charm Therapeutics, London, UK. [3]Cambridge Quantum Computing and Quantinuum, Oxford, UK.. Correspondence to: Péter Mernyei <pmernyei@gmail.com>.

Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

Many existing quantum machine learning approaches seek advantages over classical methods by exploiting the facts that quantum states and processes take place in the exponentially large Hilbert space, that states can be superposed, and that quantum amplitudes can interfere. This, however, is still being explored: encoding useful quantum states efficiently and measuring them accurately are challenges that make straightforward speed-ups difficult (Aaronson, 2015). Furthermore, since existing quantum devices are very limited, empirical benchmarks are often impossible at the scales where quantum methods might lead to a real advantage. Due to these difficulties, theoretical analysis plays a fundamental role, and recent works focusing on characterizing the capabilities and limitations of potential quantum models have shown significant results (Schuld et al., 2021; Liu et al., 2021; Kübler et al., 2021; Goto et al., 2021).

The goal of this paper is to establish a unifying framework for learning functions over graphs using parameterized quantum circuits and characterize the capabilities and limitations of various circuit classes within this framework. Classical graph machine learning techniques have been the backbone of many key developments in machine learning, since graphs are used to encode various forms of relational data, including knowledge graphs (Bordes et al., 2011), social networks (Zhang & Chen, 2018), and importantly also molecules (Wu et al., 2018), which are a particularly promising application domain of quantum computing due to their inherent quantum properties.

Graph neural networks (GNNs) (Kipf & Welling, 2017; Veličković et al., 2018) are prominent models for graph representation learning, as they encode desirable properties such as permutation invariance (resp., equivariance) relative to graph nodes, enabling a strong inductive bias (Battaglia et al., 2018). While broadly applied, the expressive power of prominent GNN architectures, such as *message-passing neural networks (MPNNs)* (Gilmer et al., 2017), is shown to be upper bounded by the 1-dimensional Weisfeiler-Lehman graph isomorphism test (Xu et al., 2019; Morris et al., 2019). This motivated a large body of work aiming at more expressive models, including higher-order models (Morris et al., 2019; Maron et al., 2019a), as well as extensions of MPNNs with unique node identifiers (Loukas, 2020), or with random node features (Sato et al., 2021; Abboud et al., 2021).

In this paper, we investigate quantum analogues of GNNs and make the following contributions:

- We define criteria for quantum circuits to respect the invariances of the graph domain, leading to *equivariant quantum graph circuits* (EQGCs) (Section 4).
- We define *equivariant hamiltonian quantum graph circuits (EH-QGCs)* and *equivariantly diagonalizable unitary quantum graph circuits (EDU-QGCs)* as special subclasses of EQGCs, and relate these classes to existing proposals from the literature, providing a unifying perspective for quantum graph representation learning (Section 4.2).
- We characterize the expressive power of EH-QGCs and EDU-QGCs, proving that they are universal approximators for functions defined over the bounded graph domain. This result is achieved by showing a correspondence between EDU-QGCs and *MPNNs enhanced with random node initialization* which are universal approximators over bounded graphs (Abboud et al., 2021). Differently, our model does not require any extraneous randomization, and the result follows from the model properties (Section 5).
- We experimentally show that even simple EDU-QGCs go beyond the capabilities of popular GNNs, by empirically verifying that they can discern graph pairs, which are indiscernible by standard MPNNs (Section 6).

The rest of this paper is organized as follows. We first discuss related work in the field of quantum machine learning in Section 2. Following this, we give an overview of important methods and results in graph representation learning in Section 3. After these preliminaries, we present our proposed framework and discuss important subclasses in Section 4, show our theoretical results on model expressivity in Section 5 and provide empirical evaluation in Section 6. We finish with a discussion of our results and possible further directions in Section 7.

All proof details and technical constructions can be found in the appendix of this paper.

2. Related Work

The field of quantum machine learning includes a wide range of approaches. Early work had partial successes in speeding up important linear algebra subroutines (Harrow et al., 2009), but these methods usually came with caveats (e.g., requirements of the input being easy to prepare or being sparse, or approximate knowledge of the final state being sufficient) that made them hard to apply to large problem classes in practice (Aaronson, 2015). Recent approaches tend to use quantum circuits to mimic or replace larger parts of classical techniques: *quantum kernels* use a quantum computer to implement a fixed kernel function in a classical learning algorithm (Schuld & Killoran, 2019; Liu et al., 2021), while *parameterized quantum circuits* (PQCs) use tunable quantum circuits as machine learning models in a manner similar to neural networks (Perdomo-Ortiz et al., 2018; Benedetti et al., 2019). Lacking the possibility of standard backpropagation, there are alternative ways of calculating gradients (Schuld et al., 2019), and gradient-free optimization methods are also used (Ostaszewski et al., 2021). In this paper, we focus on PQCs.

There is also a growing body of work on the capabilities and limitations of such models. Ciliberto et al. (2018) and Kübler et al. (2021) give rigorous results about when we can and cannot expect the inductive bias quantum of kernels to give them an advantage over classical methods; Servedio & Gortler (2004) and Liu et al. (2021) demonstrate carefully chosen function classes that quantum kernels can provably learn more efficiently than any classical learner. PQCs have been harder to reason about due to their non-convex nature, but there have been important steps in showing conditions under which certain PQCs are universal function approximators over vector spaces (Schuld et al., 2021; Goto et al., 2021), similarly to multi-layer perceptrons in the classical world (Hornik et al., 1989). There has been also rigorous work on the PAC-learnability of the output distributions of local quantum circuits (Hinsche et al., 2021).

For learning functions over graphs, the literature is rather sparse: there are some proposals based on PQCs (Verdon et al., 2019; Zheng et al., 2021) or based on kernels (Henry et al., 2021) supported by small-scale experiments, but there is generally a lack of formal justification for the particular model choices. In particular, we are not aware of any theoretical work on the capabilities of these models. We propose a framework unifying PQC models that build a circuit for each example graph in a structurally appropriate way when running inference, such as Verdon et al. (2019), Zheng et al. (2021), and Henry et al. (2021). Importantly, such PQCs are recently also used as a building block by Ai et al. (2022), who apply them to subgraphs, thereby requiring fewer qubits and enabling scaling to larger graphs. We discuss possible choices for circuit classes, and investigate their expressive power.

It is worth nothing that there are also other quantum approaches that we do not cover, such as using edges primarily in classical pre- or post-processing steps of a PQC (Chen et al., 2021), or running a PQC for each node independently and optimizing a connectivity-based error term to ensure similarity of related nodes (Beer et al., 2021)

3. Graph Neural Networks

GNNs can be dated back to earlier works of Scarselli et al. (2009) and Gori et al. (2005), which is followed by a rich line of work, leading to very popular GNNs (Kipf & Welling, 2017; Xu et al., 2019; Veličković et al., 2018; Li et al., 2016). GNNs are designed to have a graph-based *inductive bias*, i.e., the functions they learn should be invariant to the ordering of the nodes or edges of the graph, since the ordering is just a matter of representation and not a property of the graph. This includes *invariant functions* that output a single value that should be unchanged on permuting nodes, and *equivariant functions* that output a representation for each node, and this output is reordered consistently as the input is shuffled (Hamilton, 2020). Formally, a function f is *invariant* over graphs if, for isomorphic graphs $\mathcal{G}, \mathcal{H}$ it holds that $f(\mathcal{G}) = f(\mathcal{H})$; a function f mapping a graph $\mathcal{G}$ with vertices $V(\mathcal{G})$ to vectors $\boldsymbol{x} \in \mathbb{R}^{|V(\mathcal{G})|}$ is *equivariant* if, for every permutation π of $V(\mathcal{G})$, it holds that $f(\mathcal{G}^{\pi}) = f(\mathcal{G})^{\pi}$.

Message-passing neural networks (MPNNs) (Gilmer et al., 2017) are a popular and highly effective class of GNNs that iteratively update the representations of each node based on their local neighborhoods. In an MPNN, each node v is assigned some initial state vector $\boldsymbol{h}_v^{(0)}$ based on its features. This state vector is iteratively updated based on the current state of its neighbors $\mathcal{N}(v)$ and its own state, as follows:

$$\boldsymbol{h}_v^{(k+1)} = \text{UPD}^{(k)}\Big(\boldsymbol{h}_v^{(k)}, \text{AGG}^{(k)}\big(\{\!\{\boldsymbol{h}_u^{(k)} \mid u \in \mathcal{N}(v)\}\!\}\big)\Big),$$

where $\{\!\{\cdot\}\!\}$ denotes a multiset, and $\text{AGG}^{k}(\cdot)$ and $\text{UPD}^{(k)}(\cdot)$ are differentiable functions.

The specific choice for the aggregate and update functions varies across approaches, e.g., *graph convolutional networks* (GCNs) (Kipf & Welling, 2017), *graph isomorphism networks* (GINs) (Xu et al., 2019), *graph attention networks* (GATs) (Veličković et al., 2018), and *gated graph neural networks* (GGNNs) (Li et al., 2016).

After several iteration (or, layers) have been applied, the final node embeddings are pooled to form a graph embedding vector to predict properties of entire graphs. The pooling often takes the form of simple averaging, summing or elementwise maximum.

One well-known limitation of MPNNs is their expressive power which is upper bounded by the 1-dimensional Weisfeiler-Lehman algorithm (1-WL) for graph isomorphism testing (Xu et al., 2019; Morris et al., 2019). Considering a pair of 1-WL indistinguishable graphs, such as those shown in Figure 1, any MPNN will learn the exact same representations for these graphs, yielding the same prediction for both, irrespectively of the target function to be learned. In particular, this means that MPNNs cannot learn functions such as counting cycles, or detecting triangles.

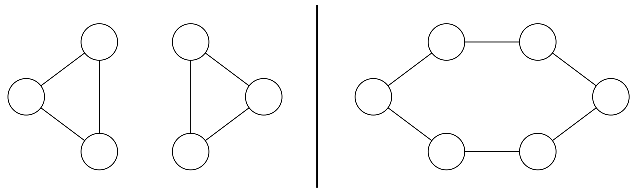


Figure 1. Two graphs indistinguishable by 1-WL: $\mathcal{G}_1$ consisting of two triangles (left), and $\mathcal{G}_2$ being a single 6-cycle (right).

The limitations in the expressive power of GNNs motivated a large body of work. Xu et al. (2019) proposed the graph isomorphism networks, as maximally expressive MPNNs, and showed this model to be as powerful as 1-WL, owing to its potential of learning injective aggregate-update functions. To break the expressiveness barrier, some approaches considered unique node identifiers (Loukas, 2020), or random pre-set color features (Dasoulas et al., 2020), and alike, so as to make graphs discernible by construction (since 1-WL can distinguish graphs with unique node identifiers), but these approaches suffer in generalization. Other approaches are based on higher-order message passing (Morris et al., 2019), or higher-order tensors (Maron et al., 2019b;a), and typically have a prohibitive computational complexity, making them less viable in practice.

Rather recently, MPNNs enhanced with random node initialization (Sato et al., 2021; Abboud et al., 2021) are shown to increase the expressivity without incurring a large computational overhead, and while preserving invariance properties in expectation. Sato et al. (2021) showed that such randomized MPNNs can detect any fixed substructure (e.g., a triangle) with high probability, and Abboud et al. (2021) proved that randomized MPNNs are *universal approximators* for functions over bounded graphs, building on an earlier logical characterization of MPNNs (Barceló et al., 2020). Intuitively, random node initialization assigns unique identifiers to different nodes with high probability and the model becomes robust via more sampling, leading to strong generalization. However, it is harder to train these models, since they need to see many different random labelings to eventually become robust to this variation. The extent of this effect can be mitigated by using fewer randomized dimensions (Abboud et al., 2021).

4. Equivariant Quantum Graph Circuits

In this section, we describe the class of models we are considering and formalize the requirement of respecting the graph structure in our definition of *equivariant quantum graph circuits*. We then discuss two subclasses and their relation to each other as well as their relation to existing models.

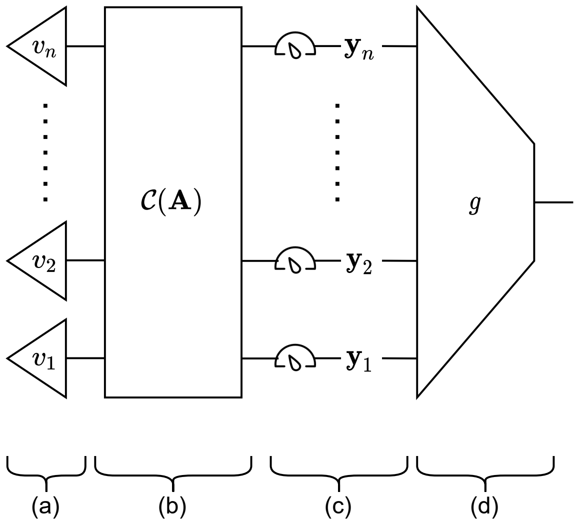


Figure 2. Overview of our model setup. **(a)** A product state is prepared based on individual nodes, **(b)** a parameterized circuit $\boldsymbol{C}$ is applied based on the adjacency matrix $\boldsymbol{A}$, **(c)** the nodes states are measured and **(d)** aggregated by some classical function g.

4.1. Model Setup

Let $\mathbb{G}^n$ be the set of graphs up to size n. Consider a graph $\mathcal{G} \in \mathbb{G}^n$, with adjacency matrix $\boldsymbol{A} \in \mathbb{B}^{n\times n}$ and a node feature vector $\boldsymbol{x}_i$ for each node $i \in \{1 \dots n\}$. We consider a broad class of models with the following simple structure, as shown in Figure 2:

1. For each node with features $\boldsymbol{x}_i$, a quantum state $|v_i\rangle = |\rho(\boldsymbol{x}_i)\rangle \in \mathbb{C}^s$ is prepared via some fixed feature map $\rho(\cdot)$. The dimensionality of this state is $s = 2^q$ when using q qubits per node.

2. The node states are composed with the tensor product to form the product state $|\boldsymbol{v}\rangle = \bigotimes_{i=1}^n |v_i\rangle \in \mathbb{C}^{s^n}$.

3. We apply some circuit encoding a unitary matrix $\boldsymbol{C_\theta}(\boldsymbol{A}) \in \mathbb{C}^{s^n\times s^n}$, dependent on the adjacency matrix $\boldsymbol{A}$ and tunable parameters $\boldsymbol{\theta}$, to the initial state of the system.

4. Each node state is measured in the computational basis, leading to a one-hot binary vector $|y_i\rangle \in \mathbb{B}^s$ for each node. Over the entire system, we measure any $|\boldsymbol{y}\rangle = \bigotimes_{i=1}^n |y_i\rangle \in \mathbb{B}^{s^n}$ with probability $P(\boldsymbol{y}) = |\langle \boldsymbol{y}| \boldsymbol{C_\theta}(\boldsymbol{A}) |\boldsymbol{v}\rangle|^2$ as dictated by the Born rule. This means the probability of any specific measurement is given by the magnitude of a single element in the final state vector $\boldsymbol{C_\theta}(\boldsymbol{A})|\boldsymbol{v}\rangle \in \mathbb{C}^{n^s}$.

5. These are aggregated by some permutation-invariant parameterized classical function $g_{\boldsymbol{\theta}'}$ to provide a prediction $g_{\boldsymbol{\theta}'}(\boldsymbol{y})$.

While this setup rules out certain possibilities such as using mixed-state quantum computing with mid-circuit measurements, or somehow aggregating the node states inside the quantum circuit, it still leaves a broad and powerful framework that subsumes existing methods (as we will discuss in Section 4.2).

We do not consider details of how to design the classical aggregator $g_{\boldsymbol{\theta}'}$ – for questions of expressivity, we will simply assume that it is a universal approximator over multisets, which is known to be achievable by combining multi-layer perceptrons with sum aggregation (Zaheer et al., 2017; Xu et al., 2019). The choice of the feature map ρ does have to be made upfront, but our proofs all use simple constructions encoding the data in the computational basis.

Our focus is instead on the circuit $\boldsymbol{C_\theta}(\boldsymbol{A})$, and how it should behave in order to interact well with the graph. As in the case of classical GNNs, we want to make sure the ordering of nodes and edges does not matter. In our case, this means that for any input, reordering the nodes and edges should reorder the probabilities of all measurements appropriately.

Example 1. With $n = 3$ nodes represented by a single qubit each ($s = 2$), the probability of observing some output $\langle y_1y_2y_3|$ is $p = \langle y_1y_2y_3| \boldsymbol{C_\theta}(\boldsymbol{A}) |v_1v_2v_3\rangle$. If we cycle the nodes around to form the input state $|v_2v_3v_1\rangle$, and also use an appropriately reordered adjacency matrix $\boldsymbol{A}'$, we should find the probability of the reordered observation $\langle y_2y_3y_1| \boldsymbol{C_\theta}(\boldsymbol{A}') |v_2v_3v_1\rangle$ to be p as well.

This brings us to the definition of *equivariant quantum graph circuits* (EQGCs):

Definition 1. Let $\boldsymbol{A} \in \mathbb{B}^{n\times n}$ be an adjacency matrix, $\boldsymbol{P} \in \mathbb{B}^{n\times n}$ a permutation matrix representing a permutation p over n elements, and $\tilde{\boldsymbol{P}} \in \mathbb{B}^{s^n\times s^n}$ a larger matrix that reorders the tensor product, mapping any $|v_1\rangle |v_2\rangle \dots |v_n\rangle$ with $|v_i\rangle \in \mathbb{C}^s$ to $|v_{p(1)}\rangle |v_{p(2)}\rangle \dots |v_{p(n)}\rangle$.

An EQGC is an arbitrary parameterized function $\boldsymbol{C_\theta}(\cdot)$ mapping an adjacency matrix $\boldsymbol{A} \in \mathbb{B}^{n\times n}$ to a unitary $\boldsymbol{C_\theta}(\boldsymbol{A}) \in \mathbb{C}^{s^n\times s^n}$ that behaves equivariantly for all $\boldsymbol{\theta}$:

$$\boldsymbol{C_\theta}(\boldsymbol{A}) = \tilde{\boldsymbol{P}}^T \boldsymbol{C_\theta}(\boldsymbol{P}^T \boldsymbol{A} \boldsymbol{P}) \tilde{\boldsymbol{P}} \tag{1}$$

In the following sections, we will generally leave the parameter $\boldsymbol{\theta}$, and sometimes also $\boldsymbol{A}$, as implicit when they are clear from context.

In accordance to our model setup, an EQGC $\boldsymbol{C_\theta}(\cdot)$ represents a probabilistic model over graphs only when combined with a fixed feature map $\rho(\cdot)$ to prepare each node state, as well as measurement and classical aggregation $g_{\boldsymbol{\theta}'}$ at the end of the circuit. Putting these together, we can formally speak of the capacity of EQGCs in representing functions.

Definition 2. We say that a (Boolean or real) function f defined on $\mathbb{G}^n$ *can be represented by an EQGC $\boldsymbol{C_\theta}$ with error* *probability* ϵ if there is some feature map ρ and invariant classical aggregation function $g_{\boldsymbol{\theta}}$, such that for any input graph $\mathcal{G} \in \mathbb{G}^n$ the model's output is $f(\mathcal{G})$ with probability $1-\epsilon$. In the special case, where $\epsilon = 0$, we simply say that the function f can be represented by an EQGC $\boldsymbol{C}_{\boldsymbol{\theta}}$.

Remark 1 (A note on directedness)**.** Unlike many works on GNNs, our definition of EQGCs allows us to consider *directed* graphs naturally, and this will also be true for the subclasses we consider later. Of course, we can still easily operate on undirected data by either adding edges in both directions, or placing extra restrictions on our models. For the purposes of expressivity, we will still focus on classifying graphs in the undirected case, as this is better explored in previous works on classical methods.

4.2. Subclasses of EQGCs

Note that we cannot and should not aim to use all possible EQGCs as a model class. If we did, the prediction of our models on any graph would not restrict their behavior on other, non-isomorphic graphs in any way. This would not only make such a class impossible to characterize with a finite set of parameters $\boldsymbol{\theta}$, but the models would also have no way to generalize to unseen inputs. Therefore EQGCs should be seen as a broad framework, and we investigate more restricted subclasses that do not have such problems.

We are particularly interested in subclasses that scale well with the number of nodes in a graph, so in the following sections we discuss approaches based on uniform single-node operations and two-node interactions at edges[1]. All of the following models are parameterized by identical operations being applied for each node or for each edge, ensuring that a single model can efficiently learn about graphs of various sizes. It is also a useful starting point for ensuring equivariance, although as we will see, we also have to make sure that the ordering of these operations does not affect our results.

4.2.1. Parameterization by Hamiltonians

Operations on the quantum states of nodes or pairs of nodes can be easily represented as unitaries, but these are tricky to parameterize directly: e.g., a linear combination of unitaries is not unitary generally. One alternative is to use the fact that any unitary $\boldsymbol{U}$ can be expressed using its Hamiltonian $\boldsymbol{H}$, a Hermitian matrix of the same size such that $\boldsymbol{U} = \exp(-i\boldsymbol{H})$. We can let the Hamiltonian depend linearly on the adjacency matrix, with Hermitian operators applied based on the structure of the graph:

Definition 3. An *equivariant hamiltonian quantum graph circuit* (EH-QGC) is an EQGC given by a composition of finitely many layers $\boldsymbol{C}_{\boldsymbol{\theta}}(\boldsymbol{A}) = \boldsymbol{L}_{\boldsymbol{\theta}_1}(\boldsymbol{A}) \circ \cdots \circ \boldsymbol{L}_{\boldsymbol{\theta}_k}(\boldsymbol{A})$, with each $\boldsymbol{L}_{\boldsymbol{\theta}_j}$ for $1 \leq j \leq k$ given as:

$$\boldsymbol{L}_{\boldsymbol{\theta}}(\boldsymbol{A}) = \exp\Big(-i\big(\sum_{\boldsymbol{A}_{jk}=1} \boldsymbol{H}^{(\text{edge})}_{j,k} + \sum_{i=1}^{n} \boldsymbol{H}^{(\text{node})}_{i}\big)\Big), \quad (2)$$

where $\boldsymbol{H}^{(\text{edge})}$ and $\boldsymbol{H}^{(\text{node})}$ are learnable Hermitian matrices over one and two-node state spaces comprising the parameter set $\boldsymbol{\theta}$, and the indexing $\boldsymbol{H}^{(\text{edge})}_{j,k}, \boldsymbol{H}^{(\text{node})}_{v}$ refers to the same operators applied at the specified node(s) – i.e., one EH-QGC layer is fully specified by a single one-node Hamiltonian and a single two-node Hamiltonian.

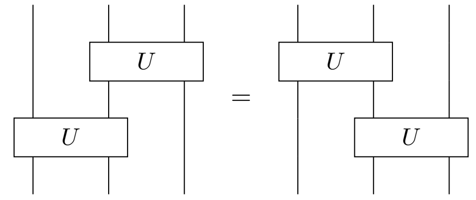
This means that if the graph is permuted, the operators will be applied at changed positions appropriately. There is also no sequential ordering of operations in a summation, so the model is equivariant. For example, $\boldsymbol{H}^{(\text{node})}_3 = \boldsymbol{I} \otimes \boldsymbol{I} \otimes \hat{\boldsymbol{H}}^{(\text{node})} \otimes \boldsymbol{I}$ in the case of $n = 4$ nodes.

EH-QGCs is closely related to the approach taken by Verdon et al. (2019) for their *quantum graph convolutional neural network* (QGCNN) model as well as the parameterized *quantum evolution kernel* of Henry et al. (2021). They both define operations in terms of Hamiltonians based on the graph structure, although their models are restricted to Hamiltonians of specific forms with fewer parameters. This helps for efficiently compiling small circuits (which is important on today's noisy machines), and allows better scaling to a larger number of qubits per node (which should be possible on future hardware). For our purposes, working with the broader class of arbitrary Hamiltonians lends itself better to theoretical analysis, and we leave it to future work to investigate circuit classes with better scaling in the number of qubits.

4.2.2. Parameterization by Commuting Unitaries

A similar, but more direct approach would be to consider two-node unitaries instead of Hamiltonians and apply a single learned unitary for each edge of the graph. As before, this ensures the number of operations scales linearly with the number of edges in a graph. This is also the approach taken by Zheng et al. (2021), but we need to add extra conditions that they do not consider to ensure equivariance.

Specifically, we need to enforce that the order we apply these unitaries in does not matter. This gives us the following commutativity condition for a two-node unitary $\boldsymbol{U}$:

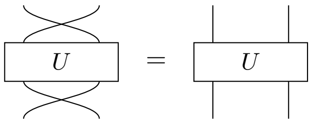


(3)

If the graphs are undirected, we should ensure the following to make sure the direction of the edge representation does

[1]We also considered the case, where $\boldsymbol{C}_{\boldsymbol{\theta}}(\cdot)$ depends only on the graph size rather than the adjacency matrix, and we report these findings in Appendix E as they are not central to our main results.

not affect our predictions:

 (4)

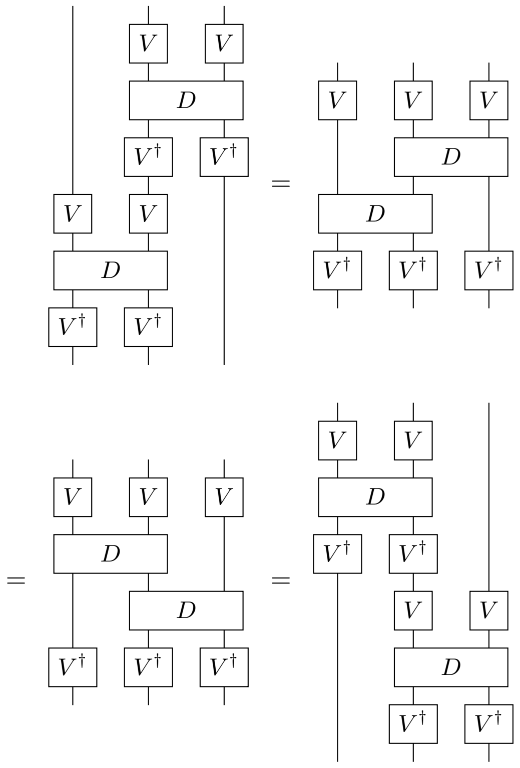
In the case of directed graphs, there are further conditions we need to satisfy instead of the above, which we detail in Appendix A.1.

It is not clear whether we can parameterize the space of all such commuting unitaries, but we can focus on a subclass.

Definition 4. An *equivariantly diagonalizable unitary* (EDU) is a unitary that can be expressed in the form $\boldsymbol{U} = (\boldsymbol{V}^\dagger \otimes \boldsymbol{V}^\dagger)\boldsymbol{D}(\boldsymbol{V} \otimes \boldsymbol{V})$ for a unitary $\boldsymbol{V} \in \mathbb{C}^{s\times s}$ and diagonal unitary $\boldsymbol{D} \in \mathbb{C}^{s^2\times s^2}$.

Note that all unitaries can be diagonalized in the form $\boldsymbol{U} = \boldsymbol{P}^\dagger\boldsymbol{D}\boldsymbol{P}$ for some other unitary $\boldsymbol{P}$ and diagonal unitary $\boldsymbol{D}$. The above is simply the case when $\boldsymbol{P}$ decomposes as $\boldsymbol{V}\otimes\boldsymbol{V}$ for one single-node unitary $\boldsymbol{V}$.

Using the facts that $\boldsymbol{I}\otimes\boldsymbol{D}$ is still a diagonal matrix and that diagonal matrices commute, we can see that equivariantly diagonalizable unitaries satisfy Equation 3:

 (5)

Furthermore, a square matrix is unitary if and only if all of its eigenvalues (the diagonal elements of $\boldsymbol{D}$) have absolute value 1. We can therefore parameterize these unitaries by combining arbitrary single-node unitaries $\boldsymbol{V}$ with diagonal matrices $\boldsymbol{D}$ of unit modulus numbers[2].

This allows us to parameterize the following class of EQGCs:

Definition 5. An *equivariantly diagonalizable unitary quantum graph circuit* (EDU-QGC) is an EQGC expressed as a composition of *node layers* $\boldsymbol{L}_{\text{node}}$ and *edge layers* $\boldsymbol{L}_{\text{edge}}$ given as follows on a graph with node and edge sets $(\mathcal{V}, \mathcal{E})$:

$$\boldsymbol{L}_{\text{node}} = V^{\otimes|\mathcal{V}|} \tag{6}$$

$$\boldsymbol{L}_{\text{edge}} = \prod_{(j,k)\in\mathcal{E}} U_{jk} \tag{7}$$

In short, we either apply the same single-node unitary to all nodes, or we apply the same EDU appropriately for each edge. Since both types of layers are equivariant by construction, so is their composition, hence EDU-QGCs are a valid EQGC class.

It can be shown that EDU-QGCs are a subclass of the Hamiltonian-based EH-QGCs discussed in Section 4.2.1. This is particularly useful for investigating questions of expressivity: we also get a result about the expressivity of EH-QGCs by showing the existence of EDU-QGC constructions representing some function.

Theorem 1. *Any EDU-QGC can be expressed as an EH-QGC.*

To show this result, we consider node layers and edge layers separately and show that both can be represented by one or more EH-QGC layers. We first prove the case for node layers, then diagonal edge layers; finally, we build on these two to prove the case for all edge layers, completing the proof. The details are provided in Appendix A.2.

5. Expressivity Results

In this section, we analyse the expressivity of the EQGCs discussed in Section 4.2: Hamiltonian-based EH-QGCs and EDU-QGCs defined using commuting unitaries.

Quantum circuits operate differently from MPNNs and other popular GNN architectures, so one might hope that they are more expressive. Since current classical methods with high expressivity are either computationally expensive (like higher-order GNNs) or require a large number of training samples to converge (like GNNs with random node initialization), this could in principle lead to a form of quantum advantage with sufficiently large-scale quantum computers.

We first show that EDU-QGCs subsume MPNNs: a class of MPNNs, including maximally expressive architectures, can

[2]To add the inductive bias of undirected graphs, we can set $\boldsymbol{D}\,|e_1e_2\rangle = \boldsymbol{D}\,|e_2e_1\rangle$ for any computational basis vectors $|e_1\rangle, |e_2\rangle$, approximately halving the number of free parameters.

be 'simulated' by a suitable EDU-QGC configuration. We then prove that they are in fact universal models for arbitrary functions on bounded-size graphs, building on prior results regarding randomized MPNNs. Since we have proven EDU-QGCs to be a subclass of EH-QGCs in Theorem 1, the results immediately follow for EH-QGCs as well.

5.1. Simulating MPNNs

Recall that MPNNs are defined via aggregate and update functions in Equation 3. In this section, we focus on MPNNs where the aggregation is of the form $\text{AGG}^{(k)}(\{\!\{\boldsymbol{h}_i\}\!\}) = \sum_i \boldsymbol{h}_i$, which includes many common architectures.

Remark 2. We consider MPNNs node states with real numbers represented in fixed-point arithmetic. Although GNNs tend to be defined with uncountable real vector state spaces, these can be approximated with a finite set if the data is from a bounded set.

We show that EDU-QGCs can simulate MPNNs with sum aggregation in the following sense:

Theorem 2. *Any (Boolean or real) function over graphs that can be represented by an MPNN with sum aggregation, can also be represented by an EDU-QGC.*

We prove this result, by giving an explicit construction to simulate an arbitrary MPNN with sum aggregation, detailed in Appendix B.1. In particular, our construction for Theorem 2 implies that for an MPNN with k layers with an embedding dimensionality of w, with a fixed-point real representation of b bits per real number, this EDU-QGC needs $(2k+1)wb$ qubits per node.

Since MPNNs with sum aggregation (e.g., GINs) can represent any function learnable by any MPNN (Xu et al., 2019), we obtain the following corollary to Theorem 2:

Corollary 2.1. *Any (Boolean or real) function that can be represented by any MPNN can also be represented by some EDU-QGC.*

5.2. Universal Approximation

We build on results about randomization in classical MPNNs, discussed in Section 3 (Sato et al., 2021; Abboud et al., 2021), to show that our quantum models are universal.

We simulate classical models that randomize some part of the node state by putting some qubits into the uniform superposition over all bitstrings, then operating in the computational basis. Unlike in the classical case, where this randomization had to be explicitly added to extend model capacity, we can do this *without modifying our model definition* – our results apply to EDU-QGCs and their superclasses. Analogously to the universality of MPNNs with random features, this allows us to prove the following theorem:

Theorem 3. *For any real function f defined over $\mathbb{G}^n$, and any $\epsilon > 0$, an EDU-QGC can represent f with an error probability ϵ.*

We cannot directly rely on the results of either Abboud et al. (2021) or Sato et al. (2021): although our theorem is analogous to that of Abboud et al. (2021), they used MPNNs extended with *readouts at each layer*, which our quantum models cannot simulate. Sato et al. (2021) used MPNNs without readouts, but did not quite prove such a claim of universality. Therefore, we give a novel MPNN construction that is partially inspired by Sato et al. (2021), but relies solely on the results of Xu et al. (2019), and use it to show Theorem 3.

Briefly, we use the fact that for bounded-size graphs individualized by random node features, a GIN can in principle assign final node states that injectively depend on the isomorphism class of each node's connected component. These node embeddings can be pooled to give a unique graph embedding for each isomorphism classes of bounded graphs, which an MLP can map to any desired results. All of this can be simulated on an EDU-QGC, hence they are universal models. The details are given in Appendix B.2.

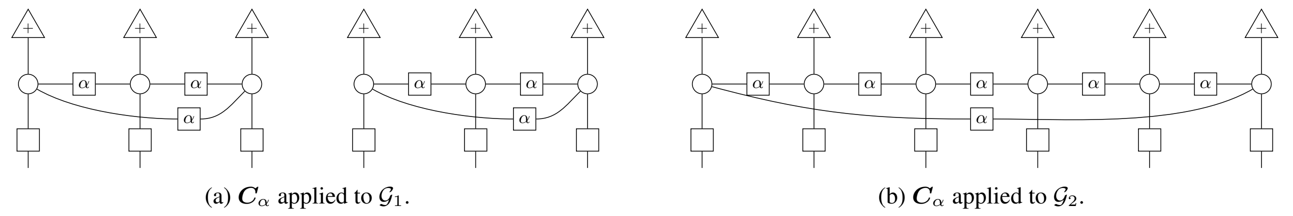
6. Empirical Evaluation

While our primary focus is theoretical, and it is challenging to execute experiments large enough to give interesting results, we performed two small experiments as well. We first look at a very restricted EDU-QGC model and observe that it can the graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ with nontrivial probability (which is beyond the capabilities of MPNNs), and also reason about this simple case analytically. After this, we construct a small classification dataset of cycle graphs in a way that MPNNs could achieve no more than 50% accuracy, and we successfully train deeper EDU-QGCs to high performance.

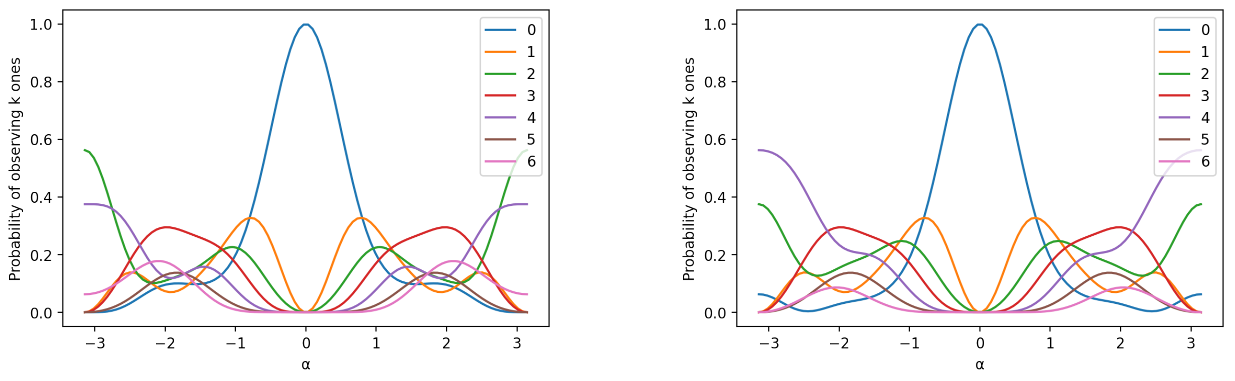
6.1. Testing expressivity beyond 1-WL

We performed a simple experiment to verify that EDU-QGC models can give different outputs for graphs that are indistinguishable by deterministic classical MPNNs. As our inputs, we used the two graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ shown in Figure 1 without node features (i.e., fixed initial node states in our quantum circuit), the simplest example where MPNNs fail. Our models should identify which graph is input. Using a *single* qubit per node, we expect our accuracy to be better than 50%, but far from perfect.

Experimental setup. To keep the experiment as simple as possible, we used a very simple subset of EDU-QGCs parameterized by a single variable α, similar to *instantaneous quantum polynomial* circuits (Bremner et al., 2016):


(a) $\boldsymbol{C}_\alpha$ applied to $\mathcal{G}_1$.

(b) $\boldsymbol{C}_\alpha$ applied to $\mathcal{G}_2$.



(c) Measurement probabilities for $\mathcal{G}_1$.

(d) Measurement probabilities for $\mathcal{G}_2$.

Figure 3. The two circuits in the experiment in top-to-bottom ZX-diagram notation, with the α-box between white spiders representing a $CZ(\alpha)$ gate (a standard ZX-calculus shorthand (Coecke & Kissinger, 2018)), followed by probabilities of observing given number of $|1\rangle$s as a function of $\alpha \in [-\pi, \pi]$ for each circuit. The two distributions differ most visibly when α is near $\pm\pi$.

- Each node state $|v_i\rangle$ is initialized as the $|+\rangle = H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ state on one node-qubit ($q = 1$). By $H = \frac{1}{\sqrt{2}}\left(\begin{smallmatrix} 1 & 1 \\ 1 & -1 \end{smallmatrix}\right)$ we denote the Hadamard gate.
- We apply an *edge layer* as given by Equation 7, with a $CZ(\alpha) = \mathrm{diag}(1, 1, 1, \exp(-i\alpha))$ gate as the applied unitary acting on two neighboring node-qubits.
- We apply a *node layer* with an H gate at each node.
- After a single measurement, we measure k nodes as a $|1\rangle$ state and $6 - k$ as $|0\rangle$. For each value of k, the aggregator $g_\alpha(\cdot)$ can map this to a different prediction.

Using ZX-diagram notation (Coecke & Kissinger, 2018), Figure 3 (top) shows the circuits we get for our choice of $\boldsymbol{C}$ in the case of $\mathcal{G}_1$ and $\mathcal{G}_2$. The probabilities of observing k $|1\rangle$s for each graph and all possible values of k as a function of our single parameter α is also shown in Figure 3 (bottom).

We find that as α gets near $\pm\pi$, the distributions of the number of $|1\rangle$s measured do differ, and an accuracy of 0.625 is achievable. This would naturally get better as we increase the number of qubits used, but this already shows an expressivity exceeding that of deterministic MPNNs.

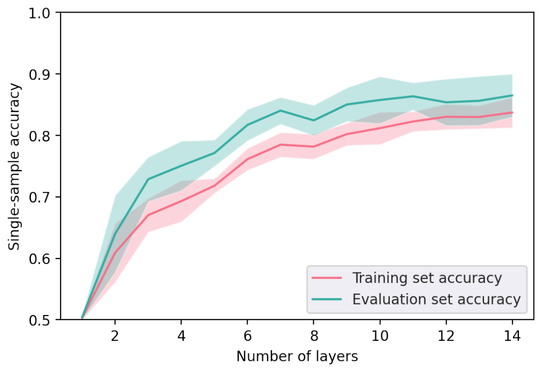
Further theoretical analysis of this setup is included in Appendix C, using the ZX-calculus (Coecke & Kissinger, 2018) to analytically derive the probabilities of all observations when applying this EDU-QGC with $\alpha = \pm\pi$ to arbitrary-size cycle graphs.

6.2. Synthetic dataset of cycle graphs

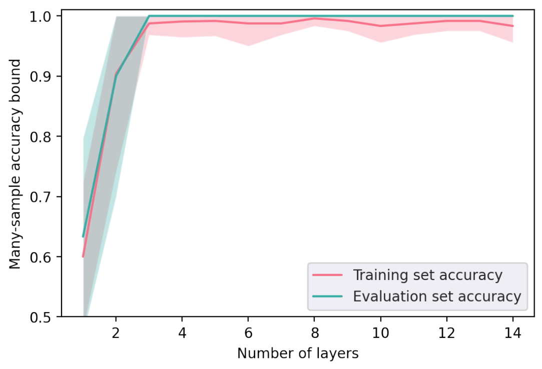
We created a synthetic dataset of 6 to 10-node graphs consisting of either a single cycle, or two cycles. The single-cycle graphs were oversampled to create two equally-sized classes for a binary classification task. 8-cycle graphs were reserved for evaluation, while all others were used for training.

We trained EDU-QGC models of various depths with a single qubit per node on this dataset. Each node state was initialized as $|+\rangle = \frac{1}{\sqrt{2}}$, then an equal number $k \in \{1, \ldots, 14\}$ general node and edge layers were applied alternatingly. After measurement, the fraction of observed $|1\rangle$s was used to predict the input's class through a learnable nonlinearity. Exact probabilities of possible outcomes were calculated, and the Adam optimizer was used to minimize the expected binary cross-entropy loss for 100 epochs, with an initial learning rate of 0.01 and an exponential learning rate decay with coefficient 0.99 applied at each epoch.

Results are shown in Figure 4. We report the one-sample accuracy (the average probability of a correct prediction across the dataset), and the highest achievable many-sample accuracy (the fraction of the dataset where a model was right with at least 50% probability). Importantly, we observe a consistent benefit from increasing depth, in contrast with the



(a) Single-sample accuracy by number of layers.



(b) Highest many-sample accuracy by number of layers.

Figure 4. Accuracies of EDU-QGC models on the synthetic cycles dataset. The many-sample accuracy bound is calculated as the fraction of examples in the dataset where the model was correct with more than 50% accuracy. Results are based on an average of 10 runs, with the shaded region representing standard deviation.[3]

oversmoothing problems of GNNs (Li et al., 2018). We also did not experience any issues with the near-zero gradients or 'barren plateaus' that make it challenging to optimize many PQC models (McClean et al., 2018). The model was also able to fit this dataset with a very small number of parameters: after accounting for redundancy, the model contains only 6 real-valued degrees of freedom for each pair of node and edge layers (see Appendix D).

Interestingly, the model performs better on the evaluation set than the training set. This is due to the fact that it is hard for the model to reliably correctly classify 9 and 10-node graphs containing two cycles when these contain subgraphs that are in the one-cycle class. For example, the model associates a high number of measured $|1\rangle$s with single-cycle graphs, then a 6-cycle will lead to many $|1\rangle$s. Since a disjoint union of a 6-cycle and a 3-cycle contains this subgraph, it will also have a relatively high fraction of $|1\rangle$s, leading to an incorrect prediction. Clearly, this would not be an issue if more qubits per node could be used (which may be feasible in future): the size of a cycle could be encoded exactly in the larger set of possible observations, and this could be easily aggregated invariantly to count the number of cycles.

7. Conclusions, Discussions, and Outlook

In this paper, we proposed equivariant quantum graph circuits, a general framework of quantum machine learning methods for graph-based machine learning, and explored possible architectures within that framework. Two subclasses, EH-QGCs and EDU-QGCs, were proven to have desirable theoretical properties: they are universal for functions defined up to a fixed graph size, just like randomized MPNNs. Our experiments were small-scale due to the computational difficulties of simulating quantum computers classically, but they did confirm that the distinguishing power of our quantum methods exceeds that of deterministic MPNNs.

By defining the framework of EQGCs and their subclasses, many questions can be raised that we did not explore in this paper. EDU-QGCs and EH-QGCs have important limitations: using arbitrary node-level Hamiltonians or unitaries allowed us to show expressivity results, but they are not feasible to scale to a large number of qubits per node, since the space of parameters grows exponentially. Perhaps a small number of qubits will already turn out to be useful, but EQGC classes with better scalability to large node states should also be investigated.

There are also design choices beyond the EQGC framework that might be interesting. For example, rather than measuring only at the end of the circuit, mid-circuit measurements and quantum-classical computation might offer possibilities that we have not analyzed.

Ultimately, the biggest questions in the field of quantum computing are about quantum advantage: what useful tasks can we expect quantum computers to speed up, and what kind of hardware do these applications require? Recent work on the theoretical capabilities of quantum machine learning architectures is already contributing to this: it has been shown that we can carefully engineer artificial problems that provably favor quantum methods (Kübler et al., 2021; Arute et al., 2019; Liu et al., 2021), but this is yet to be seen for practically significant problem classes. At the same time, there are convincing arguments that quantum computers will be useful for computational chemistry tasks such as simulating molecular dynamics, where EQGCs could be useful, which is a direction worth exploring.

[3]One of 10 runs with was dropped as an outlier in the case of 4 layers. Through some unlucky initialization, the model failed to learn anything and stayed at 50% accuracy in this single run.

References

Aaronson, S. Read the fine print. *Nature Physics*, 11(4): 291–293, 2015.

Abboud, R., Ceylan, İ. İ., Grohe, M., and Lukasiewicz, T. The surprising power of graph neural networks with random node initialization. In *IJCAI*, 2021.

Ai, X., Zhang, Z., Sun, L., Yan, J., and Hancock, E. Decompositional quantum graph neural network. *arXiv preprint arXiv:2201.05158*, 2022.

Arute, F., Arya, K., Babbush, R., Bacon, D., Bardin, J. C., Barends, R., Biswas, R., Boixo, S., Brandao, F. G. S. L., Buell, D. A., Burkett, B., Chen, Y., Chen, Z., Chiaro, B., Collins, R., Courtney, W., Dunsworth, A., Farhi, E., Foxen, B., Fowler, A., Gidney, C., Giustina, M., Graff, R., Guerin, K., Habegger, S., Harrigan, M. P., Hartmann, M. J., Ho, A., Hoffmann, M., Huang, T., Humble, T. S., Isakov, S. V., Jeffrey, E., Jiang, Z., Kafri, D., Kechedzhi, K., Kelly, J., Klimov, P. V., Knysh, S., Korotkov, A., Kostritsa, F., Landhuis, D., Lindmark, M., Lucero, E., Lyakh, D., Mandrà, S., McClean, J. R., McEwen, M., Megrant, A., Mi, X., Michielsen, K., Mohseni, M., Mutus, J., Naaman, O., Neeley, M., Neill, C., Niu, M. Y., Ostby, E., Petukhov, A., Platt, J. C., Quintana, C., Rieffel, E. G., Roushan, P., Rubin, N. C., Sank, D., Satzinger, K. J., Smelyanskiy, V., Sung, K. J., Trevithick, M. D., Vainsencher, A., Villalonga, B., White, T., Yao, Z. J., Yeh, P., Zalcman, A., Neven, H., and Martinis, J. M. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, Oct 2019. ISSN 1476-4687.

Barceló, P., Kostylev, E. V., Monet, M., Pérez, J., Reutter, J. L., and Silva, J. P. The logical expressiveness of graph neural networks. In *ICLR*, 2020.

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. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Beer, K., Khosla, M., Köhler, J., and Osborne, T. J. Quantum machine learning of graph-structured data. *arXiv preprint arXiv:2103.10837*, 2021.

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

Bordes, A., Weston, J., Collobert, R., and Bengio, Y. Learning structured embeddings of knowledge bases. In *AAAI*, 2011.

Bremner, M. J., Montanaro, A., and Shepherd, D. J. Average-case complexity versus approximate simulation of commuting quantum computations. *Physical review letters*, 117(8):080501, 2016.

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

Ciliberto, C., Herbster, M., Ialongo, A. D., Pontil, M., Rocchetto, A., Severini, S., and Wossnig, L. Quantum machine learning: a classical perspective. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 474(2209):20170551, 2018.

Coecke, B. and Kissinger, A. Picturing quantum processes. In *International Conference on Theory and Application of Diagrams*, pp. 28–31. Springer, 2018.

Cong, I., Choi, S., and Lukin, M. D. Quantum convolutional neural networks. *Nature Physics*, 15(12):1273–1278, 2019.

Dasoulas, G., Santos, L. D., Scaman, K., and Virmaux, A. Coloring graph neural networks for node disambiguation. In *IJCAI*, 2020.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *ICML*, pp. 1263–1272. PMLR, 2017.

Gori, M., Monfardini, G., and Scarselli, F. A new model for learning in graph domains. In *IJCNN*, 2005.

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

Hamilton, W. L. Graph representation learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 14(3):1–159, 2020.

Harrow, A. W., Hassidim, A., and Lloyd, S. Quantum algorithm for linear systems of equations. *Physical review letters*, 103(15):150502, 2009.

Henry, L.-P., Thabet, S., Dalyac, C., and Henriet, L. Quantum evolution kernel: Machine learning on graphs with programmable arrays of qubits. *Physical Review A*, 104 (3):032416, 2021.

Hinsche, M., Ioannou, M., Nietner, A., Haferkamp, J., Quek, Y., Hangleiter, D., Seifert, J.-P., Eisert, J., and Sweke, R. Learnability of the output distributions of local quantum circuits, 2021.

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

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

Kübler, J. M., Buchholz, S., and Schölkopf, B. The inductive bias of quantum kernels. *arXiv preprint arXiv:2106.03747*, 2021.

Li, Q., Han, Z., and Wu, X.-M. Deeper insights into graph convolutional networks for semi-supervised learning. In *AAAI*. AAAI Press, 2018.

Li, Y., Tarlow, D., Brockschmidt, M., and Zemel, R. S. Gated graph sequence neural networks. In *ICLR*, 2016.

Liu, Y., Arunachalam, S., and Temme, K. A rigorous and robust quantum speed-up in supervised machine learning. *Nature Physics*, 17(9):1013–1017, 2021.

Loukas, A. What graph neural networks cannot learn: depth vs width. In *ICLR*, 2020.

Maron, H., Ben-Hamu, H., Serviansky, H., and Lipman, Y. Provably powerful graph networks. In *NeurIPS*, 2019a.

Maron, H., Ben-Hamu, H., Shamir, N., and Lipman, Y. Invariant and equivariant graph networks. In *ICLR*, 2019b.

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

Morris, C., Ritzert, M., Fey, M., Hamilton, W. L., Lenssen, J. E., Rattan, G., and Grohe, M. Weisfeiler and leman go neural: Higher-order graph neural networks. In *AAAI*, 2019.

Ostaszewski, M., Grant, E., and Benedetti, M. Structure optimization for parameterized quantum circuits. *Quantum*, 5:391, 2021.

Perdomo-Ortiz, A., Benedetti, M., Realpe-Gómez, J., and Biswas, R. Opportunities and challenges for quantum-assisted machine learning in near-term quantum computers. *Quantum Science and Technology*, 3(3):030502, 2018.

Sato, R., Yamada, M., and Kashima, H. Random features strengthen graph neural networks. In *SDM*, pp. 333–341. SIAM, 2021.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. The graph neural network model. *Trans. Neur. Netw.*, 20(1):61–80, 2009.

Schuld, M. and Killoran, N. Quantum machine learning in feature hilbert spaces. *Physical review letters*, 122(4): 040504, 2019.

Schuld, M., Sinayskiy, I., and Petruccione, F. The quest for a quantum neural network. *Quantum Information Processing*, 13:2567–2586, 2014.

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

Schuld, M., Sweke, R., and Meyer, J. J. Effect of data encoding on the expressive power of variational quantum-machine-learning models. *Physical Review A*, 103(3), 2021.

Servedio, R. A. and Gortler, S. J. Equivalences and separations between quantum and classical learnability. *SIAM Journal on Computing*, 33(5):1067–1092, 2004.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In *ICLR*, 2018.

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

Wu, Z., Ramsundar, B., Feinberg, E., Gomes, J., Geniesse, C., Pappu, A. S., Leswing, K., and Pande, V. Moleculenet: a benchmark for molecular machine learning. *Chemical Science*, 9(2):513–530, 2018. ISSN 2041-6539.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In *ICLR*, 2019.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Póczos, B., Salakhutdinov, R., and Smola, A. Deep sets. In *NIPS*, 2017.

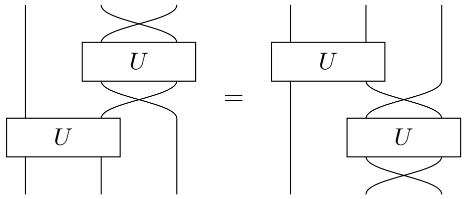
Zhang, M. and Chen, Y. Link prediction based on graph neural networks. In *NIPS*, volume 31, pp. 5165–5175, 2018.

Zheng, J., Gao, Q., and Lv, Y. Quantum graph convolutional neural networks. *arXiv preprint arXiv:2107.03257*, 2021.

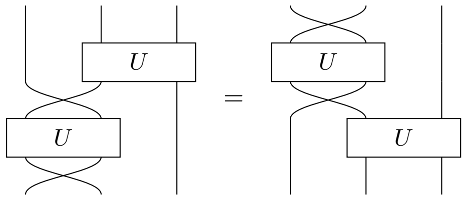
A. Details on Commuting Unitaries

A.1. Commutativity Conditions in the Directed Case

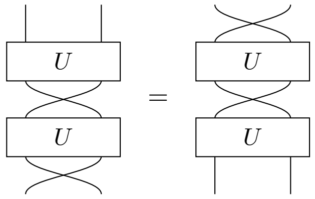
If our we are applying a two-node unitary for each edge of a directed graph, Equation 4 need not apply, but 3 is also not sufficient in itself, since we need to consider cases where the unitary might be applied in different directions. Specifically, we need to ensure the following extra conditions:



(8)

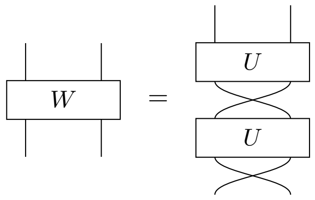


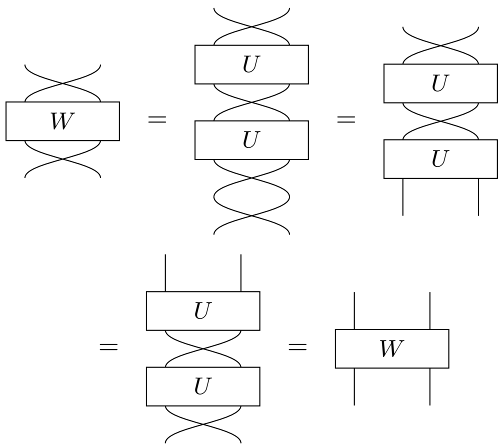
(9)



(10)

Of course, such a directed unitary can also be used for directed graphs by applying it in both directions: in fact, if Equation 10 is satisfied, this composition itself satisfies the undirected Equation 4:





(11)

In the directed case, the commutativity conditions are also satisfied, since $\boldsymbol{V} \otimes \boldsymbol{V}$ and $\boldsymbol{V}^{\dagger} \otimes \boldsymbol{V}^{\dagger}$ commute with the swap, and then analogous derivations to the undirected case apply.

A.2. Proof of Theorem 1: EDU-QGCs are a subclass of EH-QGCs

To prove this result, we initially consider EDU-QGC node layers and EDU-QGC edge layers separately and show that both can be represented by (one or more) EH-QGC layers, and afterwards combine these layers to show EH-QGCs can represent any EDU-QGC.

The proof is structured as follows: we first prove the case for node layers (Lemma 3.1), then diagonal edge layers (Lemma 3.2); and, finally, we build on these two to prove the case for all edge layers (Lemma 3.3), completing the proof of Theorem 1.

Lemma 3.1. *Any node layer $\boldsymbol{L}_{node} = \boldsymbol{V}^{\otimes|\mathcal{V}|}$ (as defined in Equation 6) can be expressed as an EH-QGC layer.*

Proof. Let $|\mathcal{V}| = n$ and let $\boldsymbol{R}$ be the Hamiltonian for $\boldsymbol{V}$. Then, $\boldsymbol{H} = \boldsymbol{R}^{\otimes n} = \sum_{v\in\mathcal{V}} \boldsymbol{R}_v$ is an appropriate EH-QGC Hamiltonian (of the form defined in Equation 2). We can easily show $\boldsymbol{H}$ is then the Hamiltonian for the EDU-QGC layer $\boldsymbol{V}^{\otimes n}$:

$$
\begin{aligned}
\exp(-i\boldsymbol{H}) &= \sum_{k=0}^{\infty} \frac{(-i\boldsymbol{H})^k}{k!} \\
&= \sum_{k=0}^{\infty} \frac{(-i)^k(\boldsymbol{R}^{\otimes n})^k}{k!} \\
&= \sum_{k=0}^{\infty} \frac{((-i\boldsymbol{R})^k)^{\otimes n}}{k!} \\
&= \Big(\sum_{k=0}^{\infty} \frac{((-i\boldsymbol{R})^k)}{k!}\Big)^{\otimes n} \\
&= \exp(-i\boldsymbol{R})^{\otimes n} \\
&= \boldsymbol{V}^{\otimes n}
\end{aligned}
$$

□

Lemma 3.2. *Any diagonal edge layer $\boldsymbol{L}_{diag} = \prod_{(j,k)\in\mathcal{E}} \boldsymbol{D}_{jk}$, with a* diagonal *unitary applied for each edge, can be expressed as an EH-QGC layer.*

Proof. A diagonal unitary $\boldsymbol{D}$ has a diagonal Hamiltonian $\boldsymbol{R}$, where $\boldsymbol{D}_{jj} = \exp(-i\boldsymbol{R}_{jj})$. Using the fact that $\exp(\boldsymbol{A})\exp(\boldsymbol{B}) = \exp(\boldsymbol{A}+\boldsymbol{B})$ for commuting matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, and that all diagonal matrices commute, we will derive that applying the Hamiltonian $\boldsymbol{R}$ for each edge at the same time has the effect of applying the unitary $\boldsymbol{D}$ for each.

Consider two edges $\{(v_1, u_1), (v_2, u_2)\}$. The overall unitary we apply, with implicit identities on all other nodes, is

$$\begin{aligned}\boldsymbol{D}_{v_2u_2}\boldsymbol{D}_{v_1u_1} &= \exp(-i\boldsymbol{R}_{v_2u_2})\exp(-i\boldsymbol{R}_{v_1u_1})\\ &= \exp(-i(\boldsymbol{R}_{v_2u_2} + \boldsymbol{R}_{v_1u_1}))\end{aligned}$$

This generalizes easily to n nodes: the Hamiltonian of the overall unitary is $\sum_{(j,k)\in\mathcal{E}} \boldsymbol{R}_{jk}$ as required. $\square$

Lemma 3.3. *Any edge layer $\boldsymbol{L}_{edge} = \prod_{(j,k)\in\mathcal{E}} \boldsymbol{U}_{jk}$ (as defined in Equation 7), with any equivariantly diagonizable unitary $\boldsymbol{U}$, can be expressed as an EH-QGC layer.*

Proof. This relies on Lemmas 3.1 and 3.2. We can show that a layer of equivariantly diagonalizable unitaries can expressed as a layer of diagonal unitaries sandwiched between two layers of single-node unitaries. Each of these can be represented as an EH-QGC layer by the previous lemmas, therefore giving us a 3-layer EH-QGC construction for this statement.

Consider an equivariantly diagonalizable unitary $\boldsymbol{U} = (\boldsymbol{V}^\dagger \otimes \boldsymbol{V}^\dagger)\boldsymbol{D}(\boldsymbol{V} \otimes \boldsymbol{V})$ applied for each edge in a layer $\prod_{(j,k)\in\mathcal{E}} \boldsymbol{U}_{jk}$. From the perspective of each node involved in edges, this decomposes as follows:

- a single-node unitary $\boldsymbol{V}$
- some number of two-node diagonal matrices separated by $\boldsymbol{V}^\dagger \times \boldsymbol{V} = \boldsymbol{I}$, which can be ignored
- a single-node unitary $\boldsymbol{V}^\dagger$

For nodes that are not part of any edge, we have the identity matrix that can be written as $\boldsymbol{V}^\dagger \times \boldsymbol{V}$. So we can rewrite the layer:

$$\prod_{(j,k)\in\mathcal{E}} \boldsymbol{U}_{jk} = \Big((\boldsymbol{V}^\dagger)^{\otimes n}\Big)\Big(\prod_{(j,k)\in\mathcal{E}} \boldsymbol{D}_{jk}\Big)\Big(\boldsymbol{V}^{\otimes n}\Big)$$

This is of the 3-layer form we discussed, proving the lemma. $\square$

Given these, we can prove the result:

Proof of Theorem 1. Putting together Lemma 3.1 and 3.3 completes the proof: both types of EDU-QGC layers given by Equations 6 and 7 can be represented by one or more EH-QGC layers, so a sequence of EH-QGC layers can represent any EDU-QGC. $\square$

B. Proofs of Expressivity Results

B.1. Proof of Theorem 2: Simulating MPNNs

We give an explicit construction to simulate an arbitrary MPNN with sum aggregation, i.e., an arbitrary MPNN where the aggregation is of the form:

$$\text{AGG}^{(k)}(\{\!\!\{\boldsymbol{h}_i\}\!\!\}) = \sum_i \boldsymbol{h}_i.$$

The node states will be conceptually split into registers representing fixed-point real numbers in two's complement in the computational basis. We first need to establish that we can perform addition on these registers using unitary transformations.

Lemma 3.4. *Consider two node states with two registers each, storing unsigned integers: $|a_1, a_2\rangle \otimes |b_1, b_2\rangle$, with $a_i, b_i \in \{0, \dots, 2^b - 1\}$ for some b. Let $\boldsymbol{U}$ map $|a_1, b_1\rangle \otimes |a_2, b_2\rangle$ to $|a_1, b_1 + a_2\rangle \otimes |a_2, b_2 + a_1\rangle$, with standard overflowing addition. Then, $\boldsymbol{U}$ is an equivariantly diagonalizable unitary and satisfies the undirected symmetry condition in Equation 4.*

Proof. Let $\boldsymbol{S}_a$ be a single-node unitary that increments integers encoded in the computational basis by a. Note that $\boldsymbol{S}_a = \boldsymbol{S}_1^a$. Diagonalize $\boldsymbol{S}_1$ as $\boldsymbol{V}^\dagger\boldsymbol{D}\boldsymbol{V}$, then $\boldsymbol{S}_a = (\boldsymbol{V}^\dagger\boldsymbol{D}\boldsymbol{V})^a = \boldsymbol{V}^\dagger\boldsymbol{D}^a\boldsymbol{V}$.

Now $\boldsymbol{U}$ can be represented by applying $\boldsymbol{V}$ to the second register of each node, conditionally applying $\boldsymbol{D}$ to the the second register of each node some number of times depending on the value of the first register, and finally applying $\boldsymbol{V}^\dagger$ to the second registers. The controlled application of a diagonal matrix is still diagonal, so this decomposition diagonalizes $\boldsymbol{U}$ equivariantly with $(\boldsymbol{I} \otimes \boldsymbol{V})^{\otimes 2}$.

The undirected symmetry Equation 4 can be seen easily from the definition of $\boldsymbol{U}$: swapping a_1 with a_2 and b_1 with b_2 results in swapping the values in the output. $\square$

Lemma 3.5. *Consider two node states with two registers each, storing fixed point unitaries in two's complement: $|a_1, a_2\rangle \otimes |b_1, b_2\rangle$, with $a_i, b_i \in \{(-2^{b-1} + 1) \times 2^{-k}, \dots, 2^{b-1} \times 2^{-k}\}$ for some b, k. Let $\boldsymbol{U}$ map $|a_1, b_1\rangle \otimes |a_2, b_2\rangle$ to $|a_1, b_1 + a_2\rangle \otimes |a_2, b_2 + a_1\rangle$, with standard overflowing addition. Then, $\boldsymbol{U}$ is an equivariantly diagonalizable unitary.*

Proof. As far as the bit-level operations are concerned, this is exactly the same as Lemma 3.4: with two's complement, standard overflowing addition of unsigned integers can represent addition of signed integers, and fixed point reals are essentially integers interpreted with a multiplication of 2^{-k}. $\square$

Having established Lemma 3.5, we are ready to prove the result:

Proof of Theorem 2. Let M be an MPNN with k layers and width w, where the initial states are $\boldsymbol{h}_1 \dots \boldsymbol{h}_n$. We define an EDU-QGC $\boldsymbol{C}$ which computes the same final node embeddings as M, based on M's iterated message-passing and node update procedure.

In the following, we conceptually divide the qubits for each node v into $(k+1) \times w$ registers $\boldsymbol{h}_v^{(0,0)}, \dots, \boldsymbol{h}_v^{(k,w-1)}$ of b qubits each, and $k \times w$ registers $\boldsymbol{a}_v^{(1,0)}, \dots, \boldsymbol{a}_v^{(k,w-1)}$ of b qubits each. This is a total of $(k+1)w \times b + kw \times b = (2k+1)wb$ qubits as expected. The $\boldsymbol{s}_v^{(0)}$ registers are initialized to the initial MPNN node states $\boldsymbol{h}_v$, and all other qubits are set to $|0\rangle$.

Then, for each MPNN layer, we first simulate its message-passing phase with two-node unitaries for all edges, and afterwards, we simulate the update functions with single-node unitaries. Specifically, for the k-th message-passing layer of M, we apply a unitary $U^{(k)}$ for each edge (v, u) that should have the effect of adding the value of $\boldsymbol{h}_v^{(k-1,i)}$ to $\boldsymbol{a}_u^{(k,i)}$ and vice versa for each $i \in \{0 \dots w-1\}$. This results in the $\boldsymbol{a}_v^{(k,\cdot)}$ registers eventually storing the sum of their neighbors' states from the previous layer, which simulates the sum aggregation. This is an equivariantly diagonalizable unitary acting well on undirected graphs by Lemma 3.5, so applying it for each edge is a valid EDU-QGC layer.

For the k-th update layer, a unitary is applied to each node that XORs the result of the MPNN's update function, $\text{UPDATE}^{(k)}(\boldsymbol{h}_v^{(k-1,\cdot)}, \boldsymbol{a}_v^{(k)})$ onto the set of registers $\boldsymbol{h}_v^{(k,\cdot)}$, which are until this point still initialized to all zeros. This is a permutation and therefore a unitary, so applying it for each node is a valid EDU-QGC layer.

At the end of the circuit, we measure all qubits, which will include the final node states $\boldsymbol{h}_v^{(k,\cdot)}$. We can classically aggregate in the same way the MPNN pools its results to give our prediction. This will match the MPNN's output for all inputs with 0 error probability. □

B.2. Proof of Theorem 3: Universality Result

We show that EDU-QGCs are universal approximators for (real and Boolean) functions over bounded graph domains, by showing EDU-QGCs can simulate MPNNs extended with random node initialization.

B.2.1. From Boolean to real-valued functions

We will prove Theorem 3 by first looking at the case of Boolean-valued functions over graphs, and show that the case for real functions follows by the same argument as Abboud et al. (2021).

Lemma 3.6. *For any Boolean function f defined over $\mathbb{G}^n$, and any $\epsilon > 0$, there is an EDU-QGC that calculates $f(\mathcal{G})$ with probability $(1-\epsilon)$ for any graph $\mathcal{G}$.*

Let us start by showing how Theorem 3 follows from Lemma 3.6:

Proof of Theorem 3 given Lemma 3.6. Consider the outputs of any real-valued function f over graphs of size n expressed in binary decimal form, in the form of zeros and ones assigned to different positions. Since there is a finite number of such graphs, there is a finite number k of different decimal places where the result differs for any two graphs. For each of these, a binary classifier can be represented by EDU-QGCs by Lemma 3.6 that gives the correct prediction with probability $1 - \frac{\epsilon}{k}$.

Say the i-th binary classifier predicts an output $F_i(G) \in \{0, 1\}$ for any bounded-size graph G that represents the bit at position $k_i \in \mathbb{Z}$ of the desired real-number output. Running these 'next to each other' is also a valid EDU-QGC, and their results can then be combined by an MLP to calculate the real output:

$$F(G) = \Big(\sum_i F_i(G) \times 2^{k_i}\Big) + C$$

By the union bound, the total probability of any classifier making a mistake is ϵ, so with probability $(1-\epsilon)$ our prediction can be as accurate as allowed by our representation of real numbers. □

B.2.2. Individualizing graphs

Abboud et al. (2021) prove their results about the power of MPNNs as follows: say a graph is *individualized* if all nodes are extended with unique features. They construct MPNNs that accurately model any function from a large class assuming the input graph is individualized. And for any graph of n nodes and arbitrarily small desired error rate ϵ, if we randomize some node features appropriately, the result will be individualized with probability at least $(1-\epsilon)$.

In the case of EDU-QGCs, if we assume some part of all node states is initialized to all $|0\rangle$s, we can have the first EDU-QGC layer apply a unitary on all nodes consisting of Hadamard gates on the appropriate qubits. This maps them to the uniform superposition over all bitstrings. If we then use the construction from Theorem 2 that acts classically on the computational basis, and then measure the results, we get the same result as running the MPNN with a randomized initial state. The following lemma bounds the number of qubits required for this:

Lemma 3.7. *Putting n sets of $b \geq 2\log(n) + \log(1/\epsilon)$ qubits each in the uniform superposition and measuring* *them leads to n unique bitstrings with probability at least $(1-\epsilon)$.*

Proof. We are effectively just randomizing b classical bits uniformly. If we randomize b individual bits of node state uniformly at random, each pair of nodes would get the same label with just 2^{-b} probability. This applies for each of the $n(n-1)/2 < n^2$ pairs of nodes, so by the union bound, the total probability of any match is at most $2^{-b}n^2$. This is less than ϵ if $b \geq 2\log(n) + \log(1/\epsilon)$ bits are randomized. □

B.2.3. Achieving universality

As noted in Section 5.2, we cannot directly rely on the results of either Abboud et al. (2021) or Sato et al. (2021), and instead give a novel MPNN construction that is partially inspired by Sato et al., but relies solely on the results of Xu et al. about their graph isomorphism networks (2019).

We essentially rely on the following about graph isomorphism networks which follows directly from Corollary 6 of Xu et al.:

Lemma 3.8. *Let $\mathcal{X}$ be a countable set of vectors, and let $\mathcal{P}_k(\mathcal{X})$ be the set of multisets of elements of $\mathcal{X}$ with size at most k. The aggregate-update function of GINs applied to inputs from $(\mathcal{X} \times \mathcal{P}_k(\mathcal{X}))$ (representing a node's previous state and the multiset of its neighbors' previous states) can learn injective functions over such an input space.*

From this result, we build up to MPNNs that can injectively encode the connected subgraph of each node into their final states if the initial features are unique. To formalize this, we need the following auxiliary definition:

Definition 6. For a graph G with initial node features $\boldsymbol{h}_v$ for each node v, a node u in G and $k \in \mathbb{Z}^+$, define

$$T(G,u,l) = \begin{cases} \{\!\{\boldsymbol{h}_u\}\!\} & \text{if } k=0 \\ \big(\boldsymbol{h}_u, \{\!\{T(G,v,k-1) & \text{if } k>0 \\ \qquad \mid v \in \mathcal{N}(u)\}\!\}\big) & \end{cases}$$

where $\mathcal{N}(u)$ represents the set of neighbors of a node u.

Following Sato et al., we call this a *level-k tree*, and it represents total information propagated to a node in k message-passing steps.

We show that GINs with k layers can injectively encode the level-k tree of a node:

Lemma 3.9. *Let $GIN_{\boldsymbol{\theta}}(G)_v$ represent the final node features of node v in a graph G after applying a graph isomorphism network with parameters $\boldsymbol{\theta}$. There is some configuration $\boldsymbol{\theta}^*$ of a k-layer GIN such that for any nodes v_1, v_2 in degree-bounded graphs G_1, G_2 respectively, with initial node features chosen from a countable space, if $T(G_1,v_1,k) \neq T(G_2,v_2,k)$ then $GIN_{\boldsymbol{\theta}^*}(G_1)_{v_1} \neq GIN_{\boldsymbol{\theta}^*}(G_2)_{v_2}$.*

Proof. By induction. The base case $k=1$ follows directly from Lemma 3.8. The inductive step follows from the same claim, since the outputs of a GIN layer applied to a countable input space still form a countable space: the set of bounded-size multisets from a countable space is still countable, and so is any image of this set under some function. □

Furthermore, we show that the level-n tree of a node in a graph of n nodes identifies the isomorphism class of the node's connected component:

Lemma 3.10. *Let G_1, G_2 be two non-isomorphic graphs with node sets V_1, V_2 of size n with node feature vectors $\boldsymbol{h}_v$ unique within each graph, and take any $v_1 \in V_1, v_2 \in V_2$. Then, the following statements hold:*

- *If the graphs G_1 and G_2 are connected, then $T(G_1,v_1,n) \neq T(G_2,v_2,n)$*
- *If the graphs G_1 and G_2 are not connected, then $T(G_1',v_1,n) \neq T(G_2',v_2,n)$, where G_1' and G_2' are the induced connected components of v_1 and v_2 in G_1 and G_2, respectively, representing the isomorphism classes.*

Proof. We first prove the case where the graphs are connected. Let $\{\boldsymbol{v}_1,\ldots,\boldsymbol{v}_n\}$ be the unique node feature vectors in G_1. Note that all of these will appear in $T(G_1,v_1,n)$, because the features of any nodes at distance d from v_1 will appear in $T(G_1,v_1,d)$ by induction, and a connected graph of n nodes has a diameter at most $(n-1)$. Therefore, if G_2 contains a different set of unique node features, we get $T(G_1,v_1,n) \neq T(G_2,v_2,n)$ immediately.

Otherwise for each i, we can denote $v_i^{(1)}$ as the node in G_1 with feature vector $\boldsymbol{v}_i$, and $v_i^{(2)}$ as the node in G_2 with the same vector. These are unique by the uniqueness of feature vectors. From $T(G_1,v_1,n)$, we can extract the sets $\mathcal{N}_1(\boldsymbol{v}_i) = \{\boldsymbol{h}_u \mid u \in \mathcal{N}(v_i^{(1)})\}$, i.e., the features of nodes adjacent to the node with the feature vector $\boldsymbol{v}_i$. This also follows by induction: $T(G_1,v_1,k)$ recursively includes a tuple $(\boldsymbol{h}_u, \{\!\{T(G_1,w,k-d-1) \mid w \in \mathcal{N}(u)\}\!\})$ for any u at $d \leq k-1$ steps from v_1, and $T(G_1,w,k-d-1)$ gives us $\boldsymbol{h}_w$ for any k,d. Similarly, from $T(G_2,v_2,n)$, we can extract $\mathcal{N}_2(\boldsymbol{v}_i) = \{\boldsymbol{h}_u \mid u \in \mathcal{N}(v_i^{(2)})\}$. If $T(G_1,v_1,n) = T(G_1,v_2,n)$, then $\mathcal{N}_1(\boldsymbol{v}_i) = \mathcal{N}_2(\boldsymbol{v}_i)$ for all i, which gives an isomorphism between G_1 and G_2: the nodes $v_i^{(1)}$ and $v_i^{(2)}$ are in correspondence.

This can be extended to the case for disconnected graphs because $T(G,v,n) = T(C,v,n)$ for a any graph G with a node v in a connected component C, and then the same derivation applies. □

These results finally allow us to prove Lemma 3.6 and thereby also complete the proof of Theorem 3:

Proof of Lemma 3.6. We start by initializing a sufficient number of qubits of each node to $|+\rangle$ such that with probability $(1-\epsilon)$, observing all n initial node states leads to n unique measurements. By Lemma 3.7, $\lceil 2\log(n)+\log(1/\epsilon)$ quits suffice. We apply an n-layer GIN to this input, which our EDU-QGC can simulate by Theorem 2. By combining Lemmas 3.9 and 3.10, with appropriate parameterization the GIN, the final node states will be an injective function of the node's connected component.

Since there is a finite number of such graphs, the set of the GIN's outputs is bounded, so an MLP applied to the node state can turn this into a vector of indicator variables for each isomorphism class within some required accuracy: let an indicator $I_C^{(v)}$, part of the node state for node v, be between $1-\frac{1}{3n}$ and 1 if the v's component is isomorphic to a graph C (without regard for the random features) and between 0 and $\frac{1}{3n}$ otherwise. Since the update function in the GIN architecture is an MLP, this computation can be built into its final layer, which our EDU-QGC can simulate.

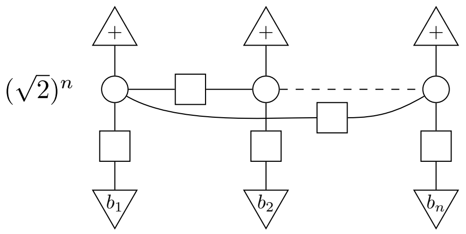
We can then pool the node states by summing them into graph-level indicators: for each isomorphism class C of at most n-node graphs, the pooled embedding will contain a summed value N_C encoding the number of nodes whose connected component is in that isomorphism class. For each I_C, the total error is at most $\frac{1}{3}$, so graphs with a different multiset of connected components will be mapped to different vectors. Since the set of graphs of size n is finite, the space of these vectors is bounded, and we can apply an MLP to these values to learn any Boolean function over bounded graphs. If we construct an MLP with accuracy 0.4, the output is always more than 0.6 if the correct answer is 1 and always less than 0.4 if the correct answer is 0. This can be mapped to discrete values in $\{0,1\}$ with perfect accuracy via a continuous function easily representable by further MLP layers. Therefore, the output of the model will be exactly correct as long as observing the $|+\rangle$ states leads to a unique initial state for each node, which has probability at least $(1-\epsilon)$ as required. $\square$

C. Analysis of the IQP Experiment

In an effort to better understand the power of such circuits, we focused on analyzing the most well-behaved special case of the above EDU-QGC, with $CZ(\pi)$ rotations.

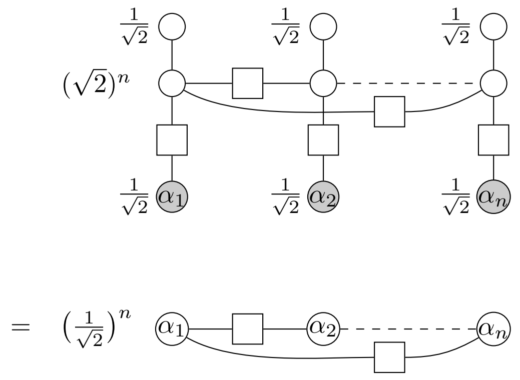
Using the ZX-calculus, we show that applying it to any n-cycle graph results in a uniform distribution over certain measurement outcomes, give a simple algorithm to check for a given n-length bitstring whether it is one of these possible outcomes, and prove that the number of measured $|1\rangle$s always has the same parity as the size n of the graph.

With $\alpha=\pi$, the α-boxes representing the CZ-gates in Figure 3 turn into simple Hadamard. So for any specific bitstring $|b_1\ldots b_n\rangle$, we can get the probability of measuring it by simplifying the following scalar:



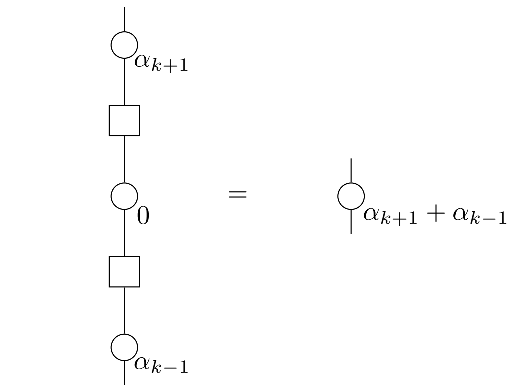
where the numerical term comes from normalizing each CZ-gate with a factor of $\sqrt{2}$.

We can substitute the appropriate white and gray spiders for the $|+\rangle$, $|0\rangle$ and $|1\rangle$ states to apply ZX-calculus techniques (Coecke & Kissinger, 2018): a white spider with phase 0 for the $|+\rangle$ state, and gray spiders with 0 and π phases respectively for $|0\rangle$ and $|1\rangle$. All of these need to be normalized with a factor of $\frac{1}{\sqrt{2}}$. Due to the Hadamard gates, these all turn into white spiders that can be fused together, so this is equal to a simple trace calculation:



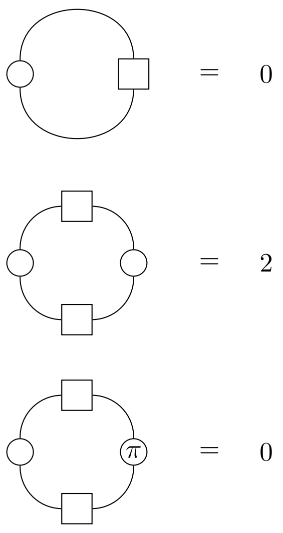
where $\alpha_i=0$ if $b_i=0$ and π if $b_i=1$.

This can be simplified step by step. Firstly, as long as there are any spiders with $\alpha_i=0$ and two distinct neighbors (i.e., there are at least 3 nodes in total), we can remove them and fuse their neighbors:



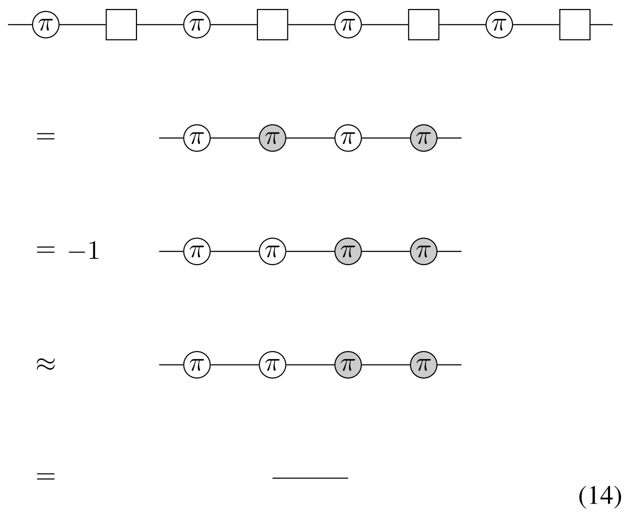
(12)

After repeating this, we get one of two outcomes. Firstly, we might end up with one of 3 possible circuits with that still have some $\alpha_i = 0$ but less than 3 nodes, which we can evaluate by direct calculation of their matrices:



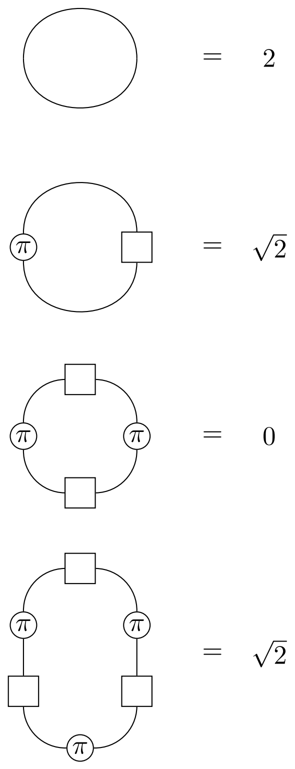
(13)

Or all the remaining spiders have $\alpha_i = \pi$, we can repeatedly eliminate them in groups of 4:



(14)

On repeating this, we end up with 0 to 3 nodes with $\alpha_i = \pi$, which we can evaluate directly:



(15)

Observe that during the simplifications, we only introduced phases with an absolute value of 1, which do not affect measurement probabilities. Furthermore, we always decreased the number of nodes involved by 2 or 4, hence the parity is unchanged. This means for odd n, we will always end up with one of the odd-cycle base cases with a trace of 0 or $\pm\sqrt{2}$, while for even n, we get to the even-cycle base cases with traces of 0 or 2.

Combining with the initial coefficient of $\left(\frac{1}{\sqrt{2}}\right)^n$ and taking squared norms, we get that *for odd n, each bitstring is observed with probability* 0 *or* $\frac{1}{2^{n-1}}$ (so half of all possible bitstrings are observed), while *with even n, each bitstring is observed with probability* 0 *or* $\frac{1}{2^{n-2}}$ (so we see only a quarter of all bitstrings).

Furthermore, to check which bitstrings are observed, we can summarize the ZX-diagram simplification as a simple algorithm acting on cyclic bitstrings (where the first and last bits are considered adjacent):

- As long as there is a 0 in the bitstring and the length of the bitstring is more than 2, remove the zero along with its two neighbors, and replace them with the XOR of the neighbors.
- If you end up with just $|00\rangle$, the state has a positive

probability to be observed. If you end up with $|0\rangle$ or $|01\rangle$, it has 0 probability.

- When there are only $|1\rangle$s remaining, if the number of these is 2 mod 4, the input has 0 probability to be observed, otherwise positive.

This shows us why the observed number of $|1\rangle$s always has the same parity as n: at each step, both the parity of $|1\rangle$s and the parity of the bitstring's length is unchanged. The only even-length base case with an odd number of ones is $|01\rangle$, which corresponds to states with 0 probability; and similarly the only odd-length base case with an even number of ones is $|0\rangle$, which has the same outcome.

We can also derive the specific probabilities observed in the experiment. It's easy to see from this that in the case of a triangle, the observable states are $|001\rangle, |010\rangle, |100\rangle, |111\rangle$. This allows us to calculate the probabilities observed for the case of two triangles. For the 6-cycle, the observable states are $|000000\rangle$, six rotations of $|000101\rangle$, six rotations of $|001111\rangle$, and three rotations of $|101101\rangle$, giving the expected probabilities as well.

D. The number of parameters of a single-qubit EDU-QGC layer

In our second experiment, we train EDU-QGC models with alternating node and edge layers. After accounting for redundancy, this setup contains merely 6 real-valued degrees of freedom for each pair of node and edge layers:

- The node layer is given by an arbitrary single-qubit unitary, which can be given by 3 Euler-angle rotations of the Bloch sphere.
- The edge layer can involve an arbitrary equivariantly diagonalizable unitary $(\boldsymbol{V} \otimes \boldsymbol{V})\boldsymbol{D}(\boldsymbol{V}^\dagger \otimes \boldsymbol{V}^\dagger)$ as given in Definition 4. However, the $\boldsymbol{V}$ is redundant when surrounded by two node layers applying single-node unitaries $\boldsymbol{U}_1, \boldsymbol{U}_2$ everywhere: modifying these to be $\boldsymbol{V} \times \boldsymbol{U}_1$ and $\boldsymbol{U}_2 \times \boldsymbol{V}^\dagger$ respectively would have the same effect. Hence it suffices to consider the diagonal unitary $\boldsymbol{D}$, which applies some phase in each of the $|00\rangle, |01\rangle, |10\rangle$ and $|11\rangle$ cases. To satisfy the undirected graph constraint of Equation 4, the phases for $|01\rangle$ and $|10\rangle$ need to be the same. This leaves us with 3 real parameters for each of the phases.

Note that in order to have an efficient implementation, we implemented edge layers as just diagonal unitaries over two nodes. This is justified by the above argument regarding their redundancy for all layers except the last, which is not surrounded by node layers – in this case it could slightly affect the performance of the model in principle.

E. Characterising Equivariant Unitaries

While investigating the behavior of EQGCs, we have considered what happens if we restrict $\boldsymbol{C_\theta}(\cdot)$ to only depend on the graph size rather than the adjacency matrix. In this case, for each n it must apply a unitary that treats each node the same. These unitaries are of interest because they could be considered PQCs with an inductive bias for learning functions over *sets* rather than graphs, and they are also the unitaries that any EQGC must assign if given a graph that is either empty or complete.

Definition 7. Let EU_s^n be the subset of $\mathbb{C}^{s^n \times s^n}$ corresponding to equivariant unitaries over n nodes of dimensionality s, i.e. unitaries that satisfy Equation 1 in place of $\boldsymbol{C_\theta}(\cdot)$.

These are the matrices that could serve as the value of $\boldsymbol{C_\theta}(n)$ in an EQGC that did not depend on the adjacency matrix $\boldsymbol{A}$. In this appendix, we prove upper and lower bounds on the dimensionality of this set, and show some necessary and some sufficient conditions for an $s^n \times s^n$ matrix to be in EU_s^n. We show that the dimensionaity of EU_s^n grows without a bound in n. This implies that contrary to the closely related *invariant and equivariant networks* studied by Maron et al. (2019b), even for our restricted EQGCs, no finite parameterization could achieve all allowed unitaries for arbitrarily high n.

We focus on the case $s = 2$, but also discuss how one would generalize our results to larger node states.

E.1. An upper bound: equivariant linear layers

The unitarity constraint is tricky to analyse, so in this section we will focus on a superset of EU_s^n with simpler structure:

Definition 8. Let $\mathsf{EU}_s^{n,+}$ be the subset of $\mathbb{C}^{s^n \times s^n}$ corresponding to arbitrary complex matrices that satisfy Equation 1 in place of $\boldsymbol{C_\theta}(\cdot)$.

First let us consider the case when $s = 2$, so each node is assigned a single qubit which is in a superposition of $|0\rangle$ and $|1\rangle$, and the action of any matrix $\boldsymbol{L}$ in $\mathsf{EU}_2^{n,+}$ can be represented as mapping bitstrings of length n (i.e., computational basis vectors in $\mathbb{C}^{2^n}$) to linear combinations of such bitstrings. The general case for $s > 2$ is conceptually analogous, but this is easier to state and prove clearly.

Theorem 4. *A matrix $\boldsymbol{L} \in \mathbb{C}^{2^n \times 2^n}$ is in $\mathsf{EU}_2^{n,+}$ if and only if it can be expressed by weights $w_{ijk} \in \mathbb{C}$ for $0 \leq i \leq n$, $0 \leq j \leq i$ and $0 \leq k \leq n - i$ as follows: for computational basis states $|\psi\rangle, |\theta\rangle$, $\langle\theta| \boldsymbol{L} |\psi\rangle = w_{ijk}$ if the bitstring representing $|\psi\rangle$ contains $|1\rangle$s in i different positions, the bitstring representing $|\theta\rangle$ contains j $|1\rangle$s at positions where $|\psi\rangle$ had $|1\rangle$s and k $|1\rangle$s at positions where $|\psi\rangle$ had $|0\rangle$s.*

Example 2. For $n = 3$,

$$\begin{aligned}\boldsymbol{L}\,|100\rangle =& w_{100}\,|000\rangle + \\ & w_{101}\big(\,|001\rangle + |010\rangle\,\big) + \\ & w_{102}\,|011\rangle + \\ & w_{110}\,|100\rangle + \\ & w_{111}\big(\,|101\rangle + |110\rangle\,\big) + \\ & w_{112}\,|111\rangle\end{aligned} \tag{16}$$

This shows that $\langle 001|\,\boldsymbol{L}\,|100\rangle = \langle 010|\,\boldsymbol{L}\,|100\rangle = w_{101}$ since $\langle 001|$ and $\langle 010|$ both contain one $\langle 1|$ in a position where where $|100\rangle$ has a $|0\rangle$, and no $\langle 1|$ where $|100\rangle$ has a $|1\rangle$; and $\langle 101|\,\boldsymbol{L}\,|100\rangle = \langle 110|\,\boldsymbol{L}\,|100\rangle = w_{111}$, because they both contain one $\langle 1|$ for the $|0\rangle$s in $|100\rangle$ and one $\langle 1|$ for the single $|1\rangle$ in $|100\rangle$. The other inner products involving $|100\rangle$ all differ in how many $\langle 1|$s meet $|1\rangle$s and $|0\rangle$s, so they can be chosen independently of each other. (Note however that they are not independent of other values of the matrix $\boldsymbol{L}$ such as those in the vectors $\boldsymbol{L}\,|001\rangle$ and $\boldsymbol{L}\,|010\rangle$, as we will see in the proof.)

For further clarity, consider representing the following EQSCs with such weights:

Example 3 ($CZ(\alpha)$-gates between all pairs of nodes)**.** Consider a circuit $\boldsymbol{L}$ consisting of controlled Z-rotations with a parameter α applied between each pair of qubits. For computational basis states $|e_1\rangle\,, |e_2\rangle$, this only applies phases, therefore we have a diagonal matrix and $\langle e_1|\,\boldsymbol{L}\,|e_2\rangle = 0$ if $|e_1\rangle \neq |e_2\rangle$. The phase applied is $e^{-i\alpha}$ for each pair of qubits that are both set to one, so if the input contains i ones then we get a phase of $e^{-i(i-1)\alpha/2}$ in total. Therefore $\boldsymbol{L}$ is represented by $w_{ijk} = e^{-i(i-1)\alpha/2}$ if $j = i, k = 0$ and 0 otherwise.

Example 4 (Arbitrary single-qubit unitaries applied everywhere)**.** Let $\boldsymbol{U} = \left(\begin{smallmatrix} u_{0,0} & u_{0,1} \\ u_{1,0} & u_{1,1}\end{smallmatrix}\right)$. Then, for $x, y \in \{0, 1\}$, we have $\langle x|\,\boldsymbol{U}\,|y\rangle = u_{x,y}$. Suppose we apply this unitary to all n qubits. Then, for two computational basis states $|e_1\rangle\,, |e_2\rangle$, $\langle e_1|\,\boldsymbol{U}^{\otimes n}\,|e_2\rangle$ is of the form $u_{0,0}^a \times u_{0,1}^b \times u_{1,0}^c \times u_{1,1}^d$, where a and d are the number of overlapping $|0\rangle$s and $|1\rangle$s respectively in the bitstring representation of $|e_1\rangle\,, |e_2\rangle$, b is the number of positions where $|e_1\rangle$ contains a $|0\rangle$ and $|e_2\rangle$ contains a $|1\rangle$, and c is the same in the other direction.

This lets us express the w_{ijk} parameters representing $\boldsymbol{U}^{\otimes n}$ from inner products of computational basis states $\langle e_1|\,\boldsymbol{U}\,|e_2\rangle$ and expressing a, b, c, d as above:

- d, the number of overlapping ones, is just j.
- c, the number of ones in $\langle e_1|$ meeting zeros in $|e_2\rangle$, is just k.
- We can get b, the number of zeros in $\langle e_1|$ meeting ones in $|e_2\rangle$ as $i - j$, subtracting the overlapping ones from the number of ones in the input.
- We can get a, the number of overlapping zeros as $(n - i) - k$, getting the number of zeros in $|e_2\rangle$ as $(n - i)$ and then subtracting the k positions where $\langle e_1|$ has a one.

So we get that $\boldsymbol{U}^{\otimes n}$ is represented by $w_{ijk} = u_{0,0}^{n-i-k} \times u_{0,1}^{i-j} \times u_{1,0}^{k} \times u_{1,1}^{j}$.

We will prove this theorem through two simple lemmas.

Lemma 4.1. *Any matrix $\boldsymbol{L} \in \mathsf{EU}_2^{n,+}$ is entirely characterized by its output on $|s_0\rangle = |00\ldots00\rangle\,, |s_1\rangle = |00\ldots01\rangle\,, \ldots, |s_{n-1}\rangle = |01\ldots11\rangle\,, |s_n\rangle = |11\ldots11\rangle$.*

Proof. Consider any the computational basis vector $|e\rangle \in \mathbb{C}^{2^n}$. This corresponds to some string of zeros and ones. Then, for the $|s_i\rangle$ containing the same number of zeros and ones, there is some permutation of indices $\tilde{\boldsymbol{P}} \in \mathbb{C}^{2^n \times 2^n}$ such that $|e\rangle = \tilde{\boldsymbol{P}}\,|s_i\rangle$ and therefore $\boldsymbol{L}\,|e\rangle = \boldsymbol{L}\tilde{\boldsymbol{P}}\,|s_i\rangle$. Multiplying by $\tilde{\boldsymbol{P}}^T$ gives $\tilde{\boldsymbol{P}}^T\boldsymbol{L}\,|e\rangle = \tilde{\boldsymbol{P}}^T\boldsymbol{L}\tilde{\boldsymbol{P}}\,|s_i\rangle = \boldsymbol{L}\,|s_i\rangle$ by equivariance, so $\boldsymbol{L}\,|e\rangle = \tilde{\boldsymbol{P}}\boldsymbol{L}\,|s_i\rangle$. So knowing $\boldsymbol{L}\,|s_i\rangle$ for each $|s_i\rangle$ determines $\boldsymbol{L}\,|e\rangle$ for all computational basis vector, hence determining it entirely. ☐

Lemma 4.2. *We must have $\langle e_1|\,\boldsymbol{L}\,|s_i\rangle = \langle e_2|\,\boldsymbol{L}\,|s_i\rangle$ for computational basis vectors $|e_1\rangle\,, |e_2\rangle$ which can be transformed to each other by permuting over indices that have the same value (0 or 1) in $|s_i\rangle$.*

Proof. Consider the permutation of indices $\tilde{\boldsymbol{P}}$ that turns $|e_1\rangle$ to $|e_2\rangle$. Note that $\tilde{\boldsymbol{P}}\,|s_i\rangle = |s_i\rangle$ by the given premise, so by equivariance we have $\langle e_1|\,\boldsymbol{L}\,|s_i\rangle = \langle e_1|\,\tilde{\boldsymbol{P}}^T\boldsymbol{L}\tilde{\boldsymbol{P}}\,|s_i\rangle = \langle e_1|\,\tilde{\boldsymbol{P}}^T\boldsymbol{L}\,|s_i\rangle = \langle e_2|\,\boldsymbol{L}\,|s_i\rangle$. ☐

Proof of Theorem 4. Lemma 4.2 showed that $\boldsymbol{L}\,|s_i\rangle$ expressed in the computational basis will have the same weight for any basis vector with j ones where $|s_i\rangle$ had ones, and k ones where $|s_i\rangle$ had zeros. Denote this weight w_{ijk}. By Lemma 4.1, these parameters uniquely characterize the equivariant linear layer.

This proves the theorem in the forward direction: any matrix in $\mathsf{EU}_2^{n,+}$ can be characterized by weights w_{ijk}. Now we show the other direction, that any linear transformation characterized by an arbitrary choice of w_{ijk} satisfies Equation 1 and therefore is in $\mathsf{EU}_2^{n,+}$. Consider an arbitrary $\boldsymbol{L} \in \mathbb{C}^{2^n \times 2^n}$ given in this form. It suffices to show that it behaves correctly with respect to swap permutations and input states in the computational basis: more complex permutations can be built by composing swaps, and more complex states by linear combinations of basis states. For any bitstring input $|e\rangle$, we can have two kinds of swaps:

- In the first case, we swap two indices with the same digit in the bitstring (both $|0\rangle$ or $|1\rangle$). The input to $\boldsymbol{L}$ is unchanged, and equivariance is respected because the same coefficients from w_{ijk} are multiplying pairs of output vectors that should be swapped.

- In the second case, the digits at the two indices differ. The inputs passed to $\boldsymbol{L}$ on the two sides of the equation are different, and equivariance is ensured by the number of overlapping $|1\rangle$s changing in a way that the w_{ijk} coefficients get swapped consistently.

□

As a consequence, we can easily see that the dimensionality of the set $\mathsf{EU}_2^{n,+}$ is unbounded in terms of n, as opposed to the equivariant layers studied by Maron et al. (2019b), so we cannot hope to uniformly parameterize the entire space for unbounded n.

Corollary 4.1. *The dimensionality of the set $\mathsf{EU}_2^{n,+}$ with a single qubit per node over n nodes is:*

$$\sum_{i=0}^{n}(i+1)(n-i+1) = \frac{1}{6}n(n+1)(n+5)$$

Proof. The left-hand side follows from the above by considering the number of (i,j,k) triples with $0 \le i \le n$, $0 \le j \le i$, $0 \le k \le n-i$. We get the closed form on the right using the formula for pyramid numbers and simplifying. □

E.1.1. Generalizing to larger node states

An analogous result holds for $\mathsf{EU}_s^{n,+}$ with $s > 2$. Say we have s possible node basis states $\{|0\rangle, \dots, |s-1\rangle\}$. In this case, a single matrix element $\langle\theta|\,\boldsymbol{L}\,|\psi\rangle$ for computational basis states $|\psi\rangle, |\theta\rangle$ is depends on the entire set of how many $|i\rangle$ appear in $|\psi\rangle$ in positions where $|\theta\rangle$ contains a $|j\rangle$, for all $i, j \in \{0 \dots s-1\}$.

To prove this, similarly to Lemma 4.1, we can show that it suffices to specify $\boldsymbol{L}\,|\psi\rangle$ for each distribution of input node states; and similarly to Lemma 4.2, we can show that $\langle\theta|\,\boldsymbol{L}\,|\psi\rangle$ is invariant to changing $\langle\theta|$ in a way that does not change the number of any $\langle i|$ to $|j\rangle$ 'matches' as described above.

E.1.2. Implications for EU_s^n

Corollary E.1 has implications for our original set of equivariant unitaries EU_s^n – it gives an upper bound for the dimensionality of the set.

E.2. A lower bound: diagonal equivariant unitaries

To see whether the size of the space of EQSCs grows with the size of the set, we can investigate a more restricted space as a lower bound: *diagonal* unitaries satisfying the equivariance condition in Equation 1.

A general diagonal unitary can apply an arbitrary phase to each computational basis state independently. The equivariance condition restricts us to applying the same phase for inputs that could be transformed to each other by permuting the indices, i.e. inputs that contain the same distribution of node states (the same number of $|0\rangle$s and $|1\rangle$s when using one qubit per node). This gives a lower bound of $n+1$ on the dimensionality of equivariant unitaries over sets of size n using a single qubit per node, which is still unbounded in n. More generally, for n nodes with s possible states each, the lower bound is the number of unique s-tuples of non-negative integers that sum to n, which is given by $\binom{n+s-1}{s-1}$. This is also a lower bound on the dimensionality of EU_s^n.

E.3. Comparison with classical invariant/equivariant graph networks

In their paper on invariant and equivariant graph networks, Maron et al. ask similar questions to characterize and implement classical equivariant/invariant models operating on tensors representing relational data (2019b). While the questions we investigate were partly inspired by them, and our data can also be seen as high-order tensors, there are significant differences in our setting.

Most importantly, the order of k the tensors they dealt with was fixed and independent of the size n of the input graph, while the size of the tensors along each of those k dimensions depended n. For example, their input included the adjacency matrix, a tensor in $\mathbb{R}^{n^2}$. For EQGCs, this is the other way around. Adding more nodes means working with a larger tensor product, but each dimension is of a fixed size s. For example, with a single qubit per node, our state is in $\mathbb{C}^{2^n}$. This matters for the notion of equivariance/invariance: applying a permutation p brings the element at an index $(i_1, i_2, \dots, i_n)$ to $(i_{p(1)}, i_{p(2)}, \dots, i_{p(n)})$ for us, instead of $(p(i_1), p(i_2), \dots, p(i_n))$ as in the previous work.

Finally, there are a few more obvious differences: due to the quantum context, we are working with complex numbers rather than reals, and we are interested in the extra condition of unitarity rather than arbitrary linear layers.