

Quantum machine learning of graph-structured data

Kerstin Beer^{1,2,*}, Megha Khosla³, Julius Köhler¹, Tobias J. Osborne¹, and Tianqi Zhao³

¹*Institut für Theoretische Physik, Leibniz Universität Hannover, 30167 Hannover, Germany*

²*School of Mathematical and Physical Sciences, Macquarie University, Sydney, New South Wales 2109, Australia*

³*Department of Intelligent Systems, Delft University of Technology, 2628 Delft, Netherlands*



(Received 18 January 2023; accepted 21 June 2023; published 10 July 2023)

Graph structures are ubiquitous throughout the natural sciences. Here we develop an approach that exploits the quantum source's graph structure to improve learning via an arbitrary quantum neural network (QNN) ansatz. In particular, we devise and optimize a self-supervised objective to capture the information-theoretic closeness of the quantum states in the training of a QNN. Numerical simulations show that our approach improves the learning efficiency and the generalization behavior of the base QNN. On a practical note, scalable quantum implementations of the learning procedure described in this paper are likely feasible on the next generation of quantum computing devices.

DOI: [10.1103/PhysRevA.108.012410](https://doi.org/10.1103/PhysRevA.108.012410)

I. INTRODUCTION

Quantum machine learning (QML), whereby classical machine learning (ML) is generalized to the quantum realm, has enjoyed a recent renaissance, leading to a dizzying array of formulations and applications (see [1–3] and references therein for a cross section). Broadly speaking, one has the following taxonomy [4]: (i) quantum speedups for classical ML [5–8], (ii) classical ML to characterize quantum systems [9–11], or (iii) quantum devices to learn quantum data (full QML) [12–22]. Our focus here is on the last category, as it is this scenario where quantum speedups are not only most likely, but also most urgently required owing to the aforementioned exponential difficulty of tomography [23].

A variety of quantum architectures for QML have been considered, from variational quantum circuits [19,24] to quantum analogs of artificial neural networks [15,17,18,20,21,25]. We believe that the quantum neural network (QNN) architecture introduced in [21] offers the most promising platform for full QML. For example, such QNNs have been exploited recently as quantum autoencoders to carry out the denoising of entangled quantum states [26]. Additionally, these QNNs appear to offer an architecture, when the quantum neurons are sufficiently local and sparse [27], which might potentially be exploited to avoid the “barren plateau” problem [28]. Finally, these QNNs have been found to reach the fundamental information-theoretic limits on quantum learning [12,16,29–31] imposed by the quantum no-free-lunch theorem [32–34], a bound on the performance of quantum learning of generic unstructured quantum data sources.

Quantum data sources will never be generic and unstructured because the devices producing them always have structure. Indeed, causal and spatial order manifest themselves in correlations between the states produced by nearby local

data sources. So it is that physics is even possible: Without causal locality, we could never have characterized the laws of physics. To quantify such correlations it is most convenient to introduce a graph structure via a finite (or infinite) graph $G = (V, E)$, where V denotes the set of vertices and E the set of edges.

There have already been several investigations exploiting graph structure for QML [35,36]. Here the emphasis has so far been on building the graph structure into the neural network ansatz itself. However, a critical open challenge facing QML is to teach a complex QNN the *a priori* variable graph structure of the quantum source itself. Here an approach that includes the graph adjacency structure in the variational network ansatz faces difficulties. It is the crucial challenge of exploiting a quantum source's graph structure to improve QML with an arbitrary QNN, which we take aim at here: Our main contribution is a general method to improve the learning efficiency, as well as the generalization behavior, of QML via an arbitrary QNN ansatz, by exploiting graph structure.

The archetypal problem we consider here is that of a distributed set of quantum information processors, associated with the vertices of a graph G . A processor at a vertex or site j takes as input a state ρ_j . The edges E of the graph encode the correlations induced between, e.g., by the spatial vicinity, these processors. The goal is to optimally learn input-output relations for this distributed set of processors: We are given a training set $\{(\rho_j, \sigma_j) \mid j = 1, 2, \dots, S\}$ of ideal outputs σ_j corresponding to an input ρ_j for a processor at vertex or site j . Such a scenario flexibly models a wide variety of physically relevant situations ranging from distributed networks of atomic clocks to quantum NISQ device clusters.

In this paper we initiate the study of graph-structured quantum data sources. Our emphasis is on learning and characterizing the graph structure of noisy and unreliable quantum data sources. Section II provides a brief overview of related works. We commence in Sec. III with a general discussion of quantum sources with graph structure and the design of

*kerstin.beer@mq.edu.au

appropriate loss functions for their characterization. This discussion is then followed in Sec. IV with the description of a training algorithm for a quantum neural network ansatz. The results of this algorithm's numerical investigations are then presented in Sec. V. The main contributions of this paper are (i) the design of an information-theoretic loss function to capture the graph structure of quantum data sources, (ii) the development of a (quantum) training algorithm applicable to QNNs to optimize the loss function as mentioned earlier, and (iii) proof-of-principle numerical simulations of the training algorithm developed.

II. RELATED WORK

Here we briefly review classical and quantum machine learning approaches for learning graph-structured data. The key challenge in this area is to encode graph structure into continuous low-dimensional representations, or embeddings, in order to exploit classical machine learning techniques. Unsupervised methods [37–40] train vertex representations or embeddings while preserving the topological structure of the graph. These representations are then exploited for downstream tasks such as missing link, or vertex label, prediction. Initial investigations of graph-based semisupervised learning [41,42] considered the addition of explicit graph-based regularizations such as Laplacian regularization to the supervised loss term. Recently semisupervised approaches based on graph-convolution networks [43–46] have exhibited state-of-the-art performance for node classification and graph classification tasks. Instead of using an additional graph regularization term in the loss function, these methods encode graph structure directly in the latent representations using neighborhood aggregation techniques. For a comprehensive overview and comparison of unsupervised and semisupervised techniques for graph-structured data, we refer the interested reader to [47,48].

There has already been a variety of investigations of QML for graph-structured classical and quantum data. First, quantum algorithms for classical graph-structured data using a quantum generalization of the random walk were presented in [49]. Semantic knowledge graphs were the subject of [50], where a sampling-based quantum algorithm was proposed. Another direction where graph structure has played a crucial role is in quantum generalizations of convolutional neural networks [29,35]. Here tensor networks with a hierarchical structure have been used to study many-body systems. The approach of incorporating a graph structure into a neural-network ansatz was also explored in [36], leading to generalizations of recurrent neural networks and convolutional neural networks.

III. GRAPH-STRUCTURED QUANTUM DATA

Correlations, both spatial and temporal, are ubiquitous throughout the natural sciences. Capturing the relationships implied by correlations is most naturally achieved in terms of graph structure. This section introduces the notion of graph-structured quantum data, which is the central object of study in this paper.

We commence by introducing some notation. We assume that we have access to a quantum system whose kinematics are characterized by a Hilbert space \mathcal{H} . There is no harm in assuming that \mathcal{H} is finite dimensional and comprised of a collection of m qubits, i.e., $\mathcal{H} \cong (\mathbb{C}^2)^{\otimes m}$. The extension to infinite dimensions does not present too many difficulties. We imagine that we have some source, for example, a quantum device from an (untrusted) commercial purveyor, of quantum states for the quantum system: The source produces an (uncharacterized or untrusted) quantum state ρ on demand. The quantum state produced by the device is assumed to be distributed according to some probability distribution over a set $\mathcal{S} = \{\rho_{v_1}, \rho_{v_2}, \dots, \rho_{v_n}\}$ of possible quantum states. Thus we write $\{(p_{v_j}, \rho_{v_j})\}_{j=1}^n$ for the source. So far, this is completely general and characterizes both unstructured and structured quantum data.

To go further we introduce a graph structure on the quantum data as follows. Suppose that the quantum states ρ_v are associated with the nodes of a graph $G = (V, E)$, i.e., we introduce a map

$$\rho : V \rightarrow \mathcal{D}(\mathcal{H})$$

from the vertices or nodes of the graph G to the set of density operators on \mathcal{H} . The connectivity structure of the data is captured by the edge set E and quantifies the information-theoretic closeness, or correlations, between neighboring states. More precisely, two states ρ_v and ρ_w are neighboring with corresponding edge $(v, w) \in E$ if they are close according to an information metric, i.e., $d(\rho_v, \rho_w) \leq \epsilon$. Note that we assume we are given a kind of graph structure of the data, i.e., we know which pairs of states are close. Information about the underlying metric of closeness is not needed to apply the presented algorithm.

To gain some intuition for this definition, we consider three examples. The first concerns a quantum simulation device which is claimed to simulate some interesting quantum system with Hamiltonian $H \in \mathcal{B}(\mathcal{H})$ for some period of time $t \in \{0, \epsilon, 2\epsilon, \dots, (n-1)\epsilon\}$, that is, we have quantum states $|\psi_t\rangle \equiv e^{itH}|0\rangle$, where $|0\rangle \in \mathcal{H}$ is some fiducial initial state. Here we associate the path graph P_n on n vertices with this data set; the vertices label the time associated with $|\psi_t\rangle$. The second example also concerns many-body physics: Here we presume a commercial vendor has produced a quantum device that can supposedly prepare a many-body system into a state with a particle localized at a given position in a lattice (the picture to have in mind here is that of a scanning tunneling microscope). Now the output states ρ_v are labeled by locations on a lattice graph G . The third example pertains to irregular graphs with a distribution of vertex degrees and connectivity, namely, a quantum device which emits low-energy eigenstates of disordered quantum systems such as Sachdev-Ye-Kitaev-type models, which have recently received considerable attention in the high-energy physics literature in the context of holography [51].

Given graph-structured quantum data $\{(p_v, \rho_v)\}_{v \in V}$, where ρ_v occurs with probability p_v , we turn to the goal of learning and modeling the (network of) quantum information processors. Note that our graph-based loss functions can be used with many kinds of graph structure. The structure can, for example, only describe the input states of the network or the

desired outputs. We use the latter case in our numerics in Sec. V. We assume that the uncharacterized quantum information processor(s) are described by a completely general completely positive (CP) map \mathcal{F} (this CP map provides the complete description of the entire network of processors). The graph structure manifests itself on the outputs of the processor(s) $\sigma_v = \mathcal{F}(\rho_v)$. Because two inputs ρ_v and ρ_w which are physically close (i.e., associated with neighboring vertices) should lead to correlated results when processed by \mathcal{F} , we assume that the output states are information-theoretically close, written $\sigma_v \sim \sigma_w$. Quantifying and exploiting this information-theoretic closeness is the main goal of this paper.

Putting aside the precise learning architecture [be it a quantum approximate optimization algorithm (QAOA) or QNN or something completely different] for the moment, we focus first on motivating and defining physically meaningful success metrics. To begin this discussion, we simply assume that our learning architecture is described by a variational class \mathcal{V} of CP maps $\mathcal{E} : \mathcal{D}(\mathcal{H}_{\text{in}}) \rightarrow \mathcal{D}(\mathcal{H}_{\text{out}})$ which take a quantum state ρ associated with a vertex and process it into some posterior output state $\mathcal{E}(\rho)$. We explain the loss functions we use to train and, after training, test our network in the following sections.

A. Supervised loss

At first we focus on how to subject a subset of the vertices of the graph to supervision. To simplify the description of the loss function in this case, we assume that the supervised vertices are required to be pure states (this restriction can be lifted with a little work). The full set of data (whereof subsets are used for training and testing) is

$$\{(\rho_1, |\phi_1^{\text{sv}}\rangle\langle\phi_1^{\text{sv}}|), \dots, (\rho_S, |\phi_S^{\text{sv}}\rangle\langle\phi_S^{\text{sv}}|), (\rho_{S+1}, |\phi_{S+1}^{\text{test}}\rangle\langle\phi_{S+1}^{\text{test}}|), \dots, (\rho_N, |\phi_N^{\text{test}}\rangle\langle\phi_N^{\text{test}}|)\}. \quad (1)$$

where, without loss of generality, we have listed the S supervised (labeled) vertices first followed by the $N - S$ unsupervised (unlabeled) vertices. For the supervised training process we use only the first S training pairs.

The key operational input required to build a meaningful success metric, or loss function, is a way to measure the information distance between two arbitrary quantum states ρ and σ . Here the fidelity $F(\rho, \sigma) \equiv \text{tr}(\sqrt{\sqrt{\rho}\sigma\sqrt{\rho}})$ is the natural choice [23]. The supervised part of our loss function is then

$$\mathcal{L}_{\text{sv}} \equiv \frac{1}{S} \sum_{u=1}^S \langle \phi_u^{\text{sv}} | \mathcal{E}(\rho_u^{\text{in}}) | \phi_u^{\text{sv}} \rangle.$$

B. Graph-based self-supervised loss

The supervised states are pure; however, the output states of our network are in general mixed. Although the fidelity is also defined for mixed states, the excessive computational complexity required to evaluate it metric means that it is often convenient to instead exploit the Hilbert-Schmidt distance

$$d_{\text{HS}}(\rho, \sigma) \equiv \text{tr}[(\rho - \sigma)^2].$$

We assume that, additionally to the quantum data described in Eq. (1), we are given an adjacency matrix A describing the

graph structure G of the problem. To say that the learning architecture \mathcal{E} has correctly captured the graph structure G of the source and supplied us with a faithful embedding, we introduce the loss function

$$\mathcal{L}_G \equiv \sum_{v,w \in V} [A]_{vw} d_{\text{HS}}(\mathcal{E}(\rho_v), \mathcal{E}(\rho_w)),$$

where $[A]_{vw}$ denotes the matrix element of A corresponding to vertices v and w . This loss function is minimized precisely when the processed output states of neighboring vertices in the graph are mapped to informationally close states.

C. Training loss

The full loss function is now specified as the combination of supervised and graph-based loss, with the graph part controlled by a factor γ :

$$\mathcal{L}_{\text{sv}+G} = \mathcal{L}_{\text{sv}} + \gamma \mathcal{L}_G. \quad (2)$$

The training task is thus to maximize $\mathcal{L}_{\text{sv}+G}$ with $\gamma \leq 0$. Recall that two quantum states are closest when the fidelity $F(\rho, \sigma)$ is maximum. Generically, the maximum depends on γ . In particular, by tuning γ , one can weight the importance of the graph structure.

It is important here to stress the role played by the graph-based loss \mathcal{L}_G : In a semisupervised learning setting \mathcal{L}_G provides the core mechanism which allows the QNN to interpolate between supervised vertices. If \mathcal{L}_G were not present, then the QNN would have no mechanism to exploit the graph structure to interpolate the action of \mathcal{E} on unobserved vertices.

A crucial feature of our loss function is that it is agnostic of the QNN architecture \mathcal{E} : It applies equally to any variational ansatz from a QAOA to dissipative QNNs.

D. Testing loss

The testing data set is supplied as a complete list of input and output states, containing both the supervised output states and the output states which were so far hidden from the QNN:

$$\{(\rho_{S+1}, |\phi_{S+1}^{\text{test}}\rangle\langle\phi_{S+1}^{\text{test}}|), \dots, (\rho_N, |\phi_N^{\text{test}}\rangle\langle\phi_N^{\text{test}}|)\}.$$

After training the network with the loss function (2), it is important to check how well the network generalizes, and this means how well it predicts the unsupervised outcomes. We use the following testing loss for this task:

$$\mathcal{L}_{\text{usv}} = \frac{1}{N - S} \sum_{u=S+1}^N \langle \phi_u^{\text{test}} | \mathcal{E}(\rho_u^{\text{in}}) | \phi_u^{\text{test}} \rangle.$$

IV. QUANTUM NEURAL NETWORK ANSATZ TO LEARN GRAPH-STRUCTURED QUANTUM DATA

We are particularly interested in scenarios where the input and output Hilbert spaces have different dimensions, which captures scenarios from classification to device characterization. This is most flexibly modeled via the dissipative variational quantum neural network ansatz based on [21]. Note that this QNN ansatz is universal for quantum computation so that it can equally model unitary processes along with general CP maps. A more detailed description can be found in the Supplemental Material [52].

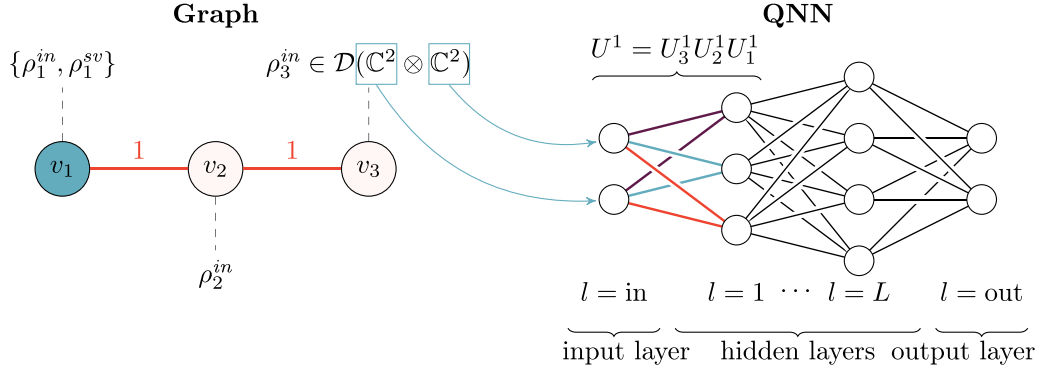


FIG. 1. Graph and QNN. Illustration of the semisupervised learning of a graph-structured quantum source (supervised nodes are shaded) via a quantum feedforward neural network (QNN). Neighboring vertices in the graph are associated with similar output states ρ_x^{sv} or ρ_x^{test} . The QNN consists of input, an output, and L hidden layers. The order of application of the perceptron unitaries is indicated with over- or undercrossings.

The QNN ansatz is built from quantum perceptrons, which are general unitary operators U acting simultaneously on the input and output qubits. The input qubits are assumed in a state ρ^{in} and the output qubits in a product state $|0 \cdots 0\rangle$. The output of one layer of perceptrons is then

$$\rho^{\text{out}} = \text{tr}_{\text{in}}[U_{\text{in,out}}(\rho^{\text{in}} \otimes |0 \cdots 0\rangle_{\text{out}} \langle 0 \cdots 0|)U_{\text{in,out}}^\dagger],$$

where $U_{\text{in,out}}$ is the product of all unitaries in that layer. We concentrate, for simplicity, on the case where the quantum perceptrons act on several input qubits and only a single output qubit. The general QNN is then described as follows: It consists of an input layer, L hidden layers, and an output layer. See Fig. 1 for an illustration. The QNN is a special class of quantum circuit comprised only of quantum perceptrons: The output state of the QNN with L hidden layers is then given by

$$\rho^{\text{out}} = \text{tr}_{\text{in,hid}}[U_{\text{out}}^L \cdots U^1(\rho^{\text{in}} \otimes |0 \cdots 0\rangle_{\text{hid,out}} \langle 0 \cdots 0|) \times U^{1\dagger} \cdots U^{L\dagger} U_{\text{out}}^{\dagger}],$$

where U^l are the layer unitaries, which are comprised of a product of quantum perceptrons acting on the qubits in layer $l-1$ and l ,

$$U^l = U_{m_l}^l U_{m_l-1}^l \cdots U_1^l,$$

where m_l is the number of qubits in layer l .

Since a quantum perceptron is an arbitrary unitary operator, the perceptrons do not in general commute. This is indicated in the figures with over- and undercrossings. Nevertheless, QNNs still inherit many of the crucial properties of classical neural networks. Most particularly, the network output is given by the composition of a sequence of CP layer-to-layer transition maps \mathcal{E}^l ,

$$\rho_x^{\text{out}} = \mathcal{E}_s^{\text{out}}(\mathcal{E}^L(\cdots \mathcal{E}^2(\mathcal{E}^1(\rho_x^{\text{in}})) \cdots)),$$

with the channel going from layer $l-1$ to l being

$$\mathcal{E}^l(X^{l-1}) = \text{tr}_{l-1}[U_{m_l}^l \cdots U_1^l(X^{l-1} \otimes |0 \cdots 0\rangle_l \langle 0 \cdots 0|) \times U_1^{l\dagger} \cdots U_{m_l}^{l\dagger}],$$

where m_l is the number of perceptrons in layer l .

With the loss functions and QNN ansatz in hand, we can explain how training proceeds. To optimize the loss function, we exploit gradient descent by allowing the perceptron unitaries to depend on a parameter s . We then update the component unitaries of the QNN by the following procedure:

$$U_j^l(s + \epsilon) = e^{i\epsilon K_j^l(s)} U_j^l(s).$$

Here $K_j^l(s)$ are Hermitian matrices that are chosen to optimize the loss function. The update matrix for a QNN trained with pure states $|\phi_u^{\text{sv}}\rangle$ as supervised vertices (and without using any known graph structure) is

$$K_j^l(s) = \frac{\eta 2^{m_l-1} i}{S} \sum_u \text{tr}_{\text{rest}}[M_{j\{u\}}^l(s)],$$

where

$$M_{j\{u\}}^l(s) = [U_j^l(s) U_{j-1}^l(s) \cdots U_1^l(s) (\rho_u^{\text{in}} \otimes |0 \cdots 0\rangle_1 \langle 0 \cdots 0|) \times U_1^{1\dagger}(s) \cdots U_{j-1}^{l\dagger}(s) U_j^{l\dagger}(s), U_{j+1}^{l\dagger}(s) \cdots U_{m_{\text{out}}}^{\text{out}\dagger}(s) \times (\mathbb{I}_{\text{in,hid}} \otimes |\phi_u^{\text{sv}}\rangle \langle \phi_u^{\text{sv}}|) U_{m_{\text{out}}}^{\text{out}}(s) \cdots U_{j+1}^l(s)].$$

This is shown in [21].

To explain how the QNN treats graph-structured quantum data and processes, see Fig. 1, where we have depicted the graph structure on the left (a path graph on three vertices) and the QNN on the right. Note, particularly, that the topology of the QNN need not have anything to do with the graph structure of the source. Here the source states are all input states for the QNN and belong to the set of density operators on two qubits. Supervised vertices (in this case one) are shaded and the corresponding supervised input and output are displayed as a pair $(\rho_i^{\text{in}}, \rho_i^{\text{sv}})$.

Theorem. The update matrix for a QNN trained with a graph structure between output states $\{\rho_v^{\text{out}}, \rho_w^{\text{out}}\}$ encoded in an adjacency matrix $[A]_{vw}$ (and without any task supervision) is

$$K_j^l(s) = \eta 2^{m_l-1+1} i \sum_{v \sim w} [A]_{vw} \text{tr}_{\text{rest}}[M_{j\{v,w\}}^l(s)],$$

where

$$M_{j\{v,w\}}^l(s) = \{U_j^l(s)U_{j-1}^l(s) \cdots U_1^l(s) \\ \times [(\rho_v^{\text{in}} - \rho_w^{\text{in}}) \otimes |0 \cdots 0\rangle_1 \langle 0 \cdots 0|] U_1^{l\dagger}(s) \cdots \\ \times U_{j-1}^{l\dagger}(s) U_j^{l\dagger}(s), U_{j+1}^{l\dagger}(s) \cdots U_{m_{\text{out}}}^{\text{out}\dagger}(s) \\ \times [\mathbb{I}_{\text{in,hid}} \otimes (\rho_v^{\text{out}} - \rho_w^{\text{out}})] U_{m_{\text{out}}}^{\text{out}}(s) \cdots U_{j+1}^l(s)\}.$$

See the Supplemental Material [52] for the proof.

Corollary. For a QNN trained with supervised vertices, as well as with graph structure, the update matrix is

$$K_j^l(s) = \frac{\eta 2^{m_{l-1}} i}{S} \sum_u \text{tr}_{\text{rest}}[M_{j\{u\}}^l(s)] \\ + \gamma \eta 2^{m_{l-1}+1} i \sum_{v \sim w} [A]_{vw} \text{tr}_{\text{rest}}[M_{j\{v,w\}}^l(s)].$$

The expression for the update matrices is involved; however, the matrices exhibit a particularly striking structure: One can calculate the updates iteratively, layer by layer, retaining only the reduced state for two layers at a time. This is reminiscent of the update rules arising in the backpropagation algorithm for classical feedforward neural networks.

V. RESULTS AND DISCUSSION

This section describes the results of numerical pilot studies for the semisupervised learning of graph-structured quantum sources on QNNs, with and without the use of graph structure. To use the proposed algorithm, we always assume we have access to training data in the form of input and output pairs and an adjacency matrix describing the graph structure of the problem. With this information we optimize a quantum neural network in a way that it provides correct output data for unseen input data.


Note that the scenario described here is a so-called transductive learning setting, where the same graph is used for training and testing. Some of the labels are hidden during the training and aimed to be predicted using the algorithm. In contrast, using an inductive learning setting aims to transfer the knowledge extracted from one graph to another, and learning and testing is done on different graphs. We leave the construction of a suiting algorithm for inductive learning on graph-structured quantum data as an open problem for future research.

The aim of the following three examples is to demonstrate that there are cases where the usage of the graph information in the transductive learning setting leads to better training of the QNN. These pilot studies were carried out using an exact simulation of the quantum systems on a classical computer. Due to the exponential scaling of the Hilbert space dimension with qubit number, we were limited to small quantum systems. Note, however, that the learning algorithms described here give rise to scalable quantum algorithms suitable for execution on the next generation of quantum computing devices described in [53].

A. Example I: Connected clusters

For the first numerical study, we construct a graph of $N = 8$ pairs of quantum states in the form of two connected clusters. The resulting structure is depicted in Fig. 2. The evolution of the testing loss during the training is depicted in Fig. 3(a). One can easily observe that the network performs better during the testing procedure, where the graph structure was exploited during training.

In the first experiment in Fig. 3(a), three of the eight vertices were supervised (used for training). We study how the number of supervised (labeled) vertices affects the training process. We repeat our experiments 30 times with a randomly chosen train-test split. In particular, we randomly choose $S < N$ of the $N = 8$ training pairs to be supervised before

every run and trained the  network for 1000 training epochs. (The symbol describes a QNN with a three-qubit input, no hidden layers, and a one-qubit outcome.) After 30 independent runs for randomly sampled training-test splits, we build the mean of the loss value after training. These testing loss means are displayed against $S \in \{0, N\}$ in Fig. 3(b).

As may be readily observed from the figures, the QNN is able to interpolate the action of the learned operation on unobserved vertices. One can observe that the test loss is higher when $\mathcal{L}_{\text{sv}+G}$ was optimized during the training compared to when \mathcal{L}_{sv} was optimized. This shows that using the additional information about the graph structure of the problem leads to better results. The code used for all numerical examples in this work is available on Github [54].

B. Example II: Line

For the second example we choose $N = 10$ pairs of quantum states so that the correlation structure is encoded in a line graph (see Fig. 4). As in example I, we plot the testing loss for one training in Fig. 5(a). We also vary the number of supervised states $S < 10$. The results are displayed in Fig. 5(b).

In this example, one observes that one can achieve a testing loss of over 0.6 with only five of ten supervised vertices when the graph structure is exploited. These numerical results demonstrate that graph-structure information provides powerful side information for training.

C. Example III: Synthetic graphs with nonrandom inputs

Based on the multilabel graph generator model developed in [55], which is inspired by the social distance attachment model in [56], we construct a synthetic graph such that the node features and the graph structure are correlated with the label outputs. We start by randomly assigning labels to nodes such that the average number of labels per node is a constant.

Here $d(\mathbf{x}_i, \mathbf{x}_j)$ denotes the hamming distance between the label vectors for nodes i and j , α denotes the level of homophily, and b is the characteristic distance. Then we generate an edge between two nodes i and j with the probability

$$p_{ij} = \frac{1}{1 + [b^{-1}d(\mathbf{x}_i, \mathbf{x}_j)]^\alpha}.$$

Note that the higher α is, the higher the chance would be that nodes with similar labels are connected. Then

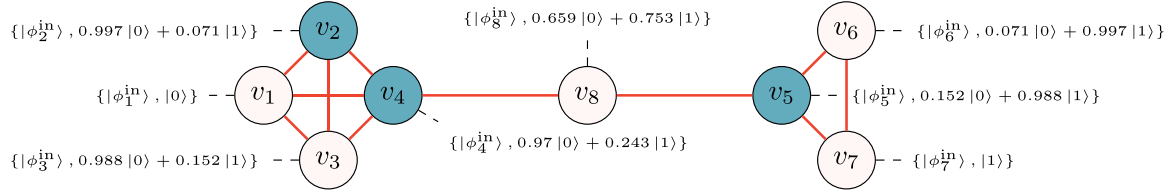


FIG. 2. Connected clusters. The output states of this training data set comprise a graph with two clusters. The states are chosen in a way such that v_1, \dots, v_4 form one cluster, v_8 connects the two clusters, and the three remaining vertices form the second cluster. The coefficients are only rounded to three decimal places. Note that only S of these states are used for training. In the figure $S = 3$ example supervised vertices are shaded. The input states are generally taken to be unstructured. In our case the states $|\phi_i^{in}\rangle$ are random three-qubit states built via a normal (Gaussian) distribution.

b is the characteristic distance between the node labels at which the connection probability for the nodes is 0.5

The graph contains 32 nodes, and every node can be assigned to one or more labels. We fix the total number of possible labels to 8. The average number of labels per node is 3. Further, every vertex is assigned to an embedding vector $\vec{e} = \{\alpha, \beta, \gamma, \delta\}$. The embedding vector is computed using DeepWalk [37] using walks of length 1 and the number of walks per vertex as 10. These embedding vectors will be used to construct input quantum states. By construction, the input states are informative of the graph structure and therefore the output labels or states.

We now construct the corresponding quantum data, namely, the quantum input and output states of the vertices. We adopt the graph structure and change the labels in the following way. Every node is assigned to a pair of quantum states: a two-qubit state $|\phi_i^E\rangle$ based on the embeddings and a three-qubit state $|\phi_i^L\rangle$ build based on the labels. Here $|\phi_i^E\rangle$ is the normalized superposition $\alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle$. To build $|\phi_i^L\rangle$ we link the states $\{|000\rangle, \dots, |111\rangle\}$ to the eight possible labels. The output state assigned to a

specific vertex is now built as the superposition of these basis states assigned to the vertex labels. If, for example, a vertex has the labels 2, 3, and 8, the output state would be a superposition of $|001\rangle$, $|010\rangle$, and $|111\rangle$.

The generalization analysis in Fig. 6 shows that for every number of supervised vertices S the training including the graph structure leads to better results.

VI. CONCLUSION

In this paper we have considered the learning of graph-structured quantum sources using dissipative QNNs. We have explained how to exploit the graph structure by designing information-theoretic loss functions. The optimization of the loss functions via QNNs was described, leading to analytic formulas for the update rules. Finally, proof-of-principle numerical simulations of the developed training algorithms were carried out, demonstrating the remarkable ability of trained QNNs to interpolate between supervised vertices and infer unobserved vertex labels.

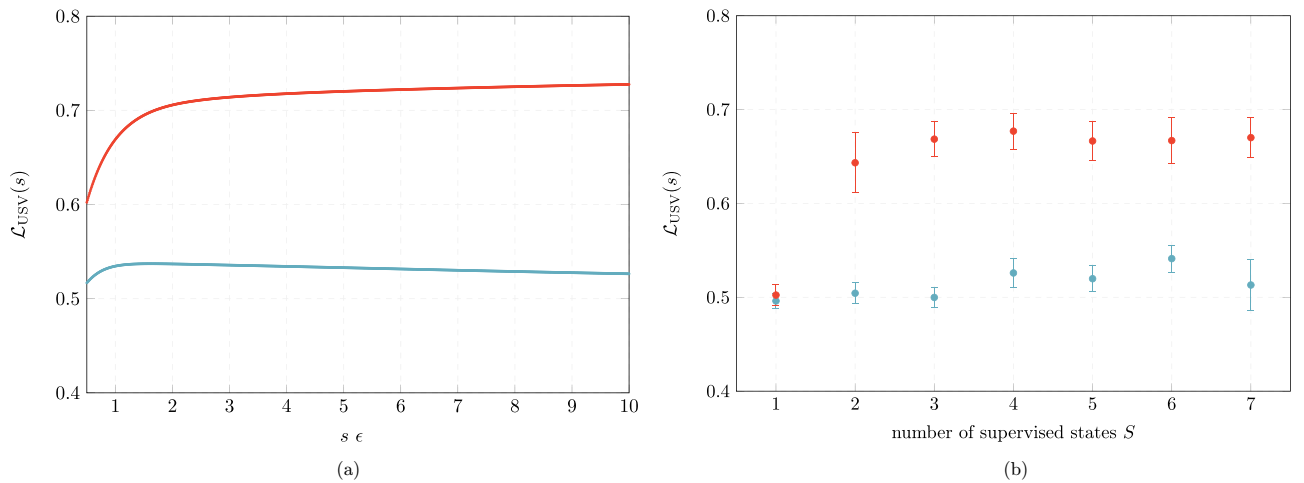



FIG. 3. Numerical results: connected clusters. We trained a  network with three supervised training pairs from Fig. 2 and the graph structure presented there. (a) Testing loss during 1000 training epochs ($\epsilon = 0.01$) optimizing $\gamma = 0$ supervised (blue circles) and $\gamma = -0.5$ supervised plus graph-based unsupervised objectives (red circles). (b) Testing loss after 1000 training epochs with the same data and parameters as in (a) but averaged over 30 random initializations and randomly chosen train-test splits plotted for different supervised pairs S . The error bars represent one standard error of the mean.

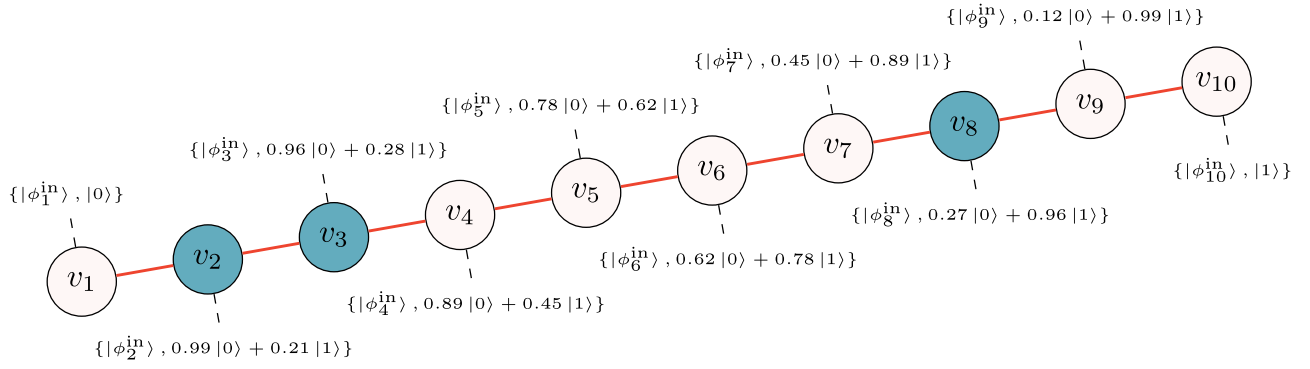


FIG. 4. Line. The output states of this training data set form a line graph. The states were chosen to be, according to the fidelity, evenly spaced along a line between the states $|0\rangle$ and $|1\rangle$ associated with the endpoints. Again, note that only S of these states are used for training. (Here $S = 3$ example vertices are shaded.) The input states are again unstructured: $|\phi_i^{in}\rangle$ are random three-qubit states built from a normal (Gaussian) distribution.

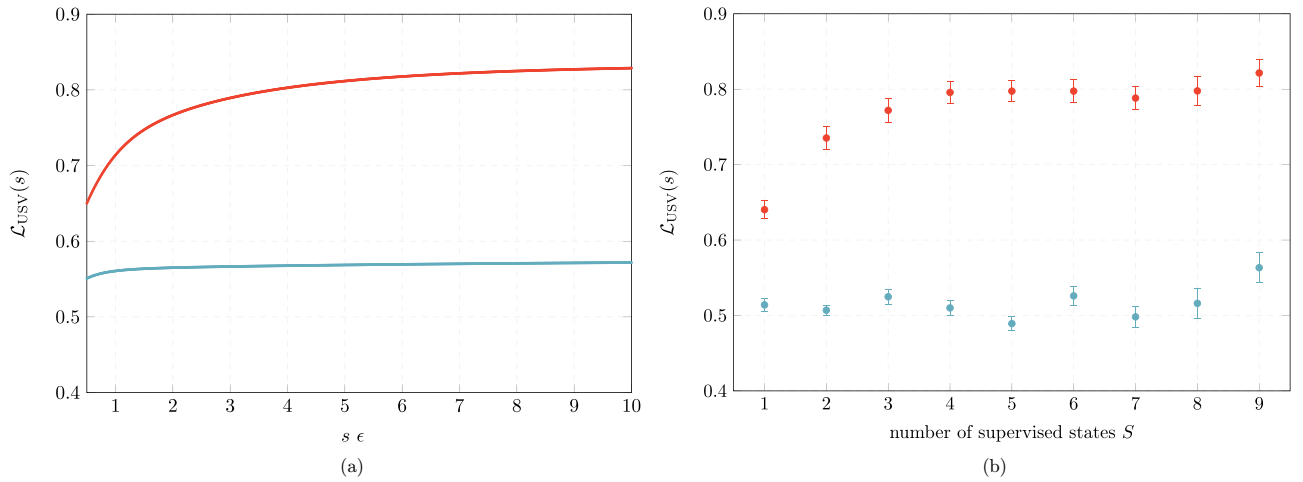





FIG. 5. Numerical results: line. We trained a  network with three supervised training pairs from Fig. 4 and the graph structure presented there. (a) Testing loss during 1000 epoches of training ($\epsilon = 0.01$) with $\gamma = 0$ semisupervised (blue ) and $\gamma = -1$ semisupervised plus graph information (red ). (b) Testing loss after 1000 epoches of training with the same data and parameters as in (a) but averaged over 30 shots plotted for different S (supervised pairs). The error bars represent one standard error of the mean.

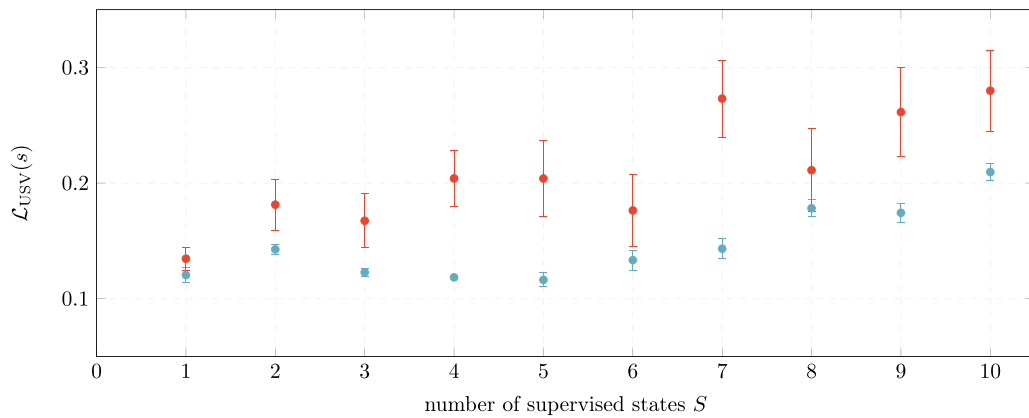





FIG. 6. Numerical results: synthetic graph with nonrandom input states. The plot describes the generalization behavior of a  network (2000 training epoches $\epsilon = 0.01$) trained without (blue ) and with (red ) using the graph structure of a graph with 32 vertices produced by a classical deep walk. Each data point demonstrates an average over ten independent training attempts. The error bars represent one standard error of the mean.

ACKNOWLEDGMENTS

Helpful correspondence and discussions with D. Bondarenko, T. Farrelly, P. Feldmann, A. Hahn, G. Müller, J. Hendrik Pfau, R. Salzmänn, D. Scheiermann, V. Schmiesing, M. Schwiering, C. Struckmann, and R. Wolf are gratefully

acknowledged. This work was supported in part by the Quantum Valley Lower Saxony, the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) through SFB 1227 (DQ-mat), the RTG 1991, and DFG under Germany's Excellence Strategy EXC-2123 QuantumFrontiers Grant No. 390837967.

-
- [1] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, *Nature (London)* **549**, 195 (2017).
 - [2] C. Ciliberto, M. Herbster, A. D. Ialongo, M. Pontil, A. Rocchetto, S. Severini, and L. Wossnig, *Proc. R. Soc. A* **474**, 20170551 (2018).
 - [3] M. Schuld, I. Sinayskiy, and F. Petruccione, *Contemp. Phys.* **56**, 172 (2015).
 - [4] E. Aïmeur, G. Brassard, and S. Gambs, in *Advances in Artificial Intelligence*, edited by L. Lamontagne and M. Marchand, Lecture Notes in Computer Science Vol. 4013 (Springer, Berlin, 2006), pp. 431–442.
 - [5] E. Aïmeur, G. Brassard, and S. Gambs, *Mach. Learn.* **90**, 261 (2013).
 - [6] G. D. Paparo, V. Dunjko, A. Makmal, M. A. Martin-Delgado, and H. J. Briegel, *Phys. Rev. X* **4**, 031002 (2014).
 - [7] M. Schuld, I. Sinayskiy, and F. Petruccione, *Quantum Inf. Process.* **13**, 2567 (2014).
 - [8] N. Wiebe, A. Kapoor, and K. M. Svore, in *Advances in Neural Information Processing Systems* 29, edited by D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (Curran, Red Hook, 2016).
 - [9] N. B. Lovett, C. Crosnier, M. Perarnau-Llobet, and B. C. Sanders, *Phys. Rev. Lett.* **110**, 220501 (2013).
 - [10] G. Carleo and M. Troyer, *Science* **355**, 602 (2017).
 - [11] M. Tiersch, E. J. Ganahl, and H. J. Briegel, *Sci. Rep.* **5**, 12874 (2015).
 - [12] M. Sasaki and A. Carlini, *Phys. Rev. A* **66**, 022303 (2002).
 - [13] S. Gambs, *arXiv:0809.0444*.
 - [14] G. Sentís, J. Calsamiglia, R. Muñoz-Tapia, and E. Bagan, *Sci. Rep.* **2**, 708 (2012).
 - [15] V. Dunjko, J. M. Taylor, and H. J. Briegel, *Phys. Rev. Lett.* **117**, 130501 (2016).
 - [16] A. Monràs, G. Sentís, and P. Wittek, *Phys. Rev. Lett.* **118**, 190503 (2017).
 - [17] U. Alvarez-Rodriguez, L. Lamata, P. Escandell-Montero, J. D. Martín-Guerrero, and E. Solano, *Sci. Rep.* **7**, 13645 (2017).
 - [18] M. H. Amin, E. Andriyash, J. Rolfe, B. Kulchytskyy, and R. Melko, *Phys. Rev. X* **8**, 021050 (2018).
 - [19] Y. Du, M.-H. Hsieh, T. Liu, and D. Tao, *Phys. Rev. Res.* **2**, 033125 (2020).
 - [20] G. Sentís, A. Monràs, R. Muñoz-Tapia, J. Calsamiglia, and E. Bagan, *Phys. Rev. X* **9**, 041029 (2019).
 - [21] K. Beer, D. Bondarenko, T. Farrelly, T. J. Osborne, R. Salzmänn, D. Scheiermann, and R. Wolf, *Nat. Commun.* **11**, 808 (2020).
 - [22] G. Verdon, J. Pye, and M. Broughton, *arXiv:1806.09729*.
 - [23] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, Cambridge, 2000).
 - [24] E. Farhi, J. Goldstone, S. Gutmann, and H. Neven, *arXiv:1703.06199*.
 - [25] V. Dunjko and H. J. Briegel, *Rep. Prog. Phys.* **81**, 074001 (2018).
 - [26] D. Bondarenko and P. Feldmann, *Phys. Rev. Lett.* **124**, 130502 (2020).
 - [27] K. Sharma, M. Cerezo, L. Cincio, and P. J. Coles, *Phys. Rev. Lett.* **128**, 180505 (2022).
 - [28] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven, *Nat. Commun.* **9**, 4812 (2018).
 - [29] S. Arunachalam and R. de Wolf, *SIGACT News* **48**, 41 (2017).
 - [30] S. Gammelmark and K. Mølmer, *New J. Phys.* **11**, 033017 (2009).
 - [31] M. Sasaki, A. Carlini, and R. Jozsa, *Phys. Rev. A* **64**, 022317 (2001).
 - [32] K. Poland, K. Beer, and T. J. Osborne, *arXiv:2003.14103*.
 - [33] K. Sharma, M. Cerezo, Z. Holmes, L. Cincio, A. Sornborger, and P. J. Coles, *Phys. Rev. Lett.* **128**, 070501 (2022).
 - [34] A. Bisio, G. Chiribella, G. M. D'Ariano, S. Facchini, and P. Perinotti, *Phys. Rev. A* **81**, 032324 (2010).
 - [35] I. Cong, S. Choi, and M. D. Lukin, *Nat. Phys.* **15**, 1273 (2019).
 - [36] G. Verdon, T. McCourt, E. Luzhnica, V. Singh, S. Leichenauer, and J. Hidar, *arXiv:1909.12264*.
 - [37] B. Perozzi, R. Al-Rfou, and S. Skiena, *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Association for Computing Machinery, New York, 2014), pp. 701–710.
 - [38] J. Qiu, Y. Dong, H. Ma, J. Li, K. Wang, and J. Tang, *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining* (Association for Computing Machinery, New York, 2018), pp. 459–467.
 - [39] S. Liu, M. F. Demirel, and Y. Liang, in *Advances in Neural Information Processing Systems* 32, edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Curran, Red Hook, 2019), pp. 8466–8478.
 - [40] M. Khosla, J. Leonhardt, W. Nejdl, and A. Anand, in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, edited by U. Brefeld, E. Fromont, A. Hotho, A. Knobbe, M. Maathuis, and C. Robardet, Lecture Notes in Computer Science (Springer, Cham, 2019), Vol. 11906, pp. 395–411.
 - [41] X. Zhu, Z. Ghahramani, and J. Lafferty, *Proceedings of the Twentieth International Conference on International Conference on Machine Learning* (AAAI, Washington, DC, 2003), pp. 912–919.
 - [42] M. Belkin, P. Niyogi, and V. Sindhwani, *J. Mach. Learn. Res.* **7**, 2399 (2006).
 - [43] T. N. Kipf and M. Welling, *Fifth International Conference on Learning Representations, Toulon, 2017* (ICLR, La Jolla, 2017).
 - [44] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, in *Sixth International Conference on Learning Representations Vancouver, 2018*, edited by Y. Bengio and Y. LeCun (ICLR, La Jolla, 2018).

- [45] W. L. Hamilton, R. Ying, and J. Leskovec, in *Advances in Neural Information Processing Systems 30*, edited by I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Curran, Red Hook, 2017).
- [46] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, *Seventh International Conference on Learning Representations, New Orleans, 2019* (ICLR, La Jolla, 2019).
- [47] M. Khosla, V. Setty, and A. Anand, *IEEE Trans. Knowl. Data Eng.* **33**, 1807 (2019).
- [48] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, *IEEE Trans. Neural Netw. Learn. Syst.* **32**, 4 (2021).
- [49] S. Dernbach, A. Mohseni-Kabir, S. Pal, M. Gepner, and D. Towsley, *Appl. Netw. Sci.* **4**, 76 (2019).
- [50] Y. Ma and V. Tresp, *Assoc. Comput. Mach.* **2**, 2643 (2021).
- [51] J. Maldacena and D. Stanford, *Phys. Rev. D* **94**, 106002 (2016).
- [52] See Supplemental Material at <http://link.aps.org/supplemental/10.1103/PhysRevA.108.012410> for details.
- [53] K. Beer, D. List, G. Müller, T. J. Osborne, and C. Struckmann, *arXiv:2104.06081*.
- [54] https://github.com/qigitphannover/QNN_GraphStructuredData.
- [55] T. Zhao, N. T. Dong, A. Hanjalic, and M. Khosla, *arXiv:2304.10398*.
- [56] S. Talaga and A. Nowak, *JASSS* **23**, 6 (2020).