3D-QAE: Fully Quantum Auto-Encoding of 3D Point Clouds (Supplementary Material)

Lakshika Rathi[1,2,†]
lrathi@wisc.edu
Edith Tretschk[2]
edithtretschk.github.io
Christian Theobalt[2]
theobalt@mpi-inf.mpg.de
Rishabh Dabral[2]
rdabral@mpi-inf.mpg.de
Vladislav Golyanik[2]
golyanik@mpi-inf.mpg.de

[1] Indian Institute of Technology Delhi, New Delhi, India
[2] Max Planck Institute for Informatics, SIC, Saarbrücken, Germany

†work done during an internship at MPI

This supplementary material provides extensive background on gate-based quantum computing (Sec. A), mathematical details on our normalisation scheme (Sec. B) and a visualisation of the circuit design (Sec. C).

A Background

A.1 Preliminaries on Quantum Computing

A *qubit* is the unit of information in a quantum system. Like a classical bit's binary states, a qubit has two basis states written in the Dirac's bra-ket notation as $|0\rangle$ and $|1\rangle$.

Superposition is one of the key advantages of quantum computing over classical computing. Rather than being restricted to the basis states, a qubit can also be in a state $|\psi\rangle$ that is a linear combination of the basis states: $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, where $\alpha, \beta \in \mathbb{C}$ and $|\alpha|^2 + |\beta|^2 = 1$. If we write the basis states as orthogonal vectors $|0\rangle = \begin{bmatrix}1 & 0\end{bmatrix}^T \in \mathbb{C}^2$ and $|1\rangle = \begin{bmatrix}0 & 1\end{bmatrix}^T \in \mathbb{C}^2$, we see that they span a complex Hilbert space. A state of a single qubit can be visualised on the *Bloch sphere*; see Fig. 1.

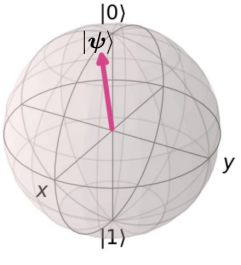

Figure 1: The one-qubit state $|\psi\rangle = (-0.91 - 0.39j)|0\rangle + (-0.11 - 0.05j)|1\rangle$ on the Bloch sphere.

Measuring the state $|\psi\rangle$ irreversibly forces (or *collapses*) it into one of the basis states, *i.e.* it yields either $|0\rangle$ or $|1\rangle$. The probability of collapsing to $|0\rangle$ or $|1\rangle$ is $|\alpha|^2$ and $|\beta|^2$, respectively. Because of this property, $|\alpha|$ and $|\beta|$ are also called *probability amplitudes*.

Quantum entanglement is the second key advantage over classical computing. The state of a collection (or *system*) of

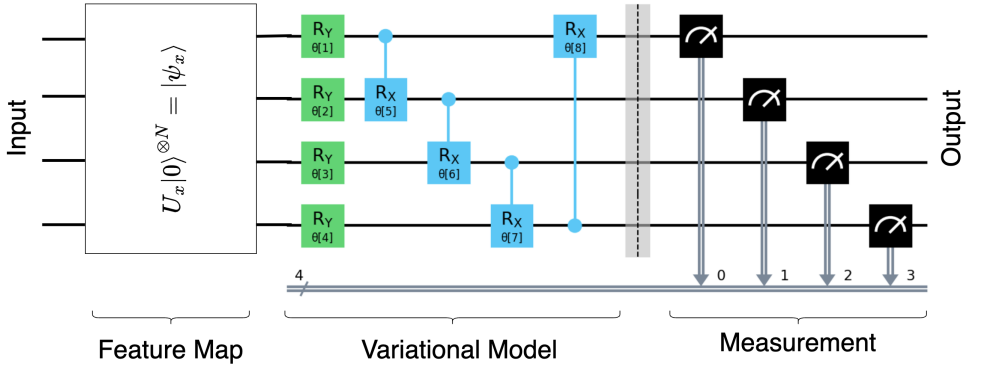

Figure 2: An exemplary quantum circuit. A feature map encodes a classical input into a state vector of N=4 qubits, which is then transformed by the parametrised quantum gates before being measured to obtain N classical output bits.

classical bits is described fully by knowing the state of each bit. However, qubits can be so strongly correlated that the state of the system cannot be described anymore as a mere collection of per-qubit states. Rather, the entire system can be in a joint superposition. For example, a system of two entangled qubits has a state $|\psi\rangle = \alpha\,|00\rangle + \beta\,|01\rangle + \gamma|10\rangle + \delta\,|11\rangle$, with $\alpha,\beta,\gamma,\delta \in \mathbb{C}$, $|\alpha|^2+|\beta|^2+|\gamma|^2+|\delta|^2 = 1$, and $|ij\rangle \in \mathbb{C}^2 \otimes \mathbb{C}^2$ are basis states covering all combinations of the two qubits, where "$\otimes$" is the tensor product. In other words, the state of a classical system is a *single* combination of N bits, while the state of an entangled system is a *distribution* over *all* combinations. More generally, an N-qubit system can exist in any superposition of the 2^N basis states: $|\psi\rangle = \mathring{a}_{i=0}^{2^N-1}\,\alpha_i\,|i\rangle$, where i enumerates all combinations of the N qubits (*i.e.,* all basis states $|i\rangle \in (\mathbb{C}^2)^{\otimes N}$) and $\mathring{a}_{i=0}^{2^N-1}|\alpha_i|^2 = 1$. In vector notation, we obtain the *state vector*: $|\psi\rangle = (\alpha_0, \alpha_1, \ldots, \alpha_{2^N-1})$. Thus, the state of N qubits is given by specifying 2^N-1 many degrees of freedom (DoF), while a classical system is given by N DoF. Therefore, entanglement allows to encode exponentially many real numbers in N many qubits, improving the processing speed of quantum computers and achieving exponential speed-up over classical systems.

A.2 Quantum Circuits

Like a classical circuit acts on classical bits, a *quantum circuit* transforms the state of a given N-qubit system (performs a computation with qubits). A quantum circuit consists of three broad steps, as Fig. 2 shows: input encoding, applying parameterised quantum gates, and output measuring. First, the classical input data (3D poses in our case) needs to be encoded into an initial N-qubit state vector. As its main operation, the quantum circuit then applies a unitary operation (the complex analogue of an orthogonal matrix) to this initial state vector. We thus obtain a transformed state vector as output. Subsequent measuring collapses the qubits to basis states, yielding an N-dimensional bit string.

Input Encoding. We first need to convert our classical input data into an initial quantum state vector. To that end, a *feature map* takes the classical input into the N-qubit Hilbert space. While there are many ways of encoding classical information (*e.g.* angle encoding),

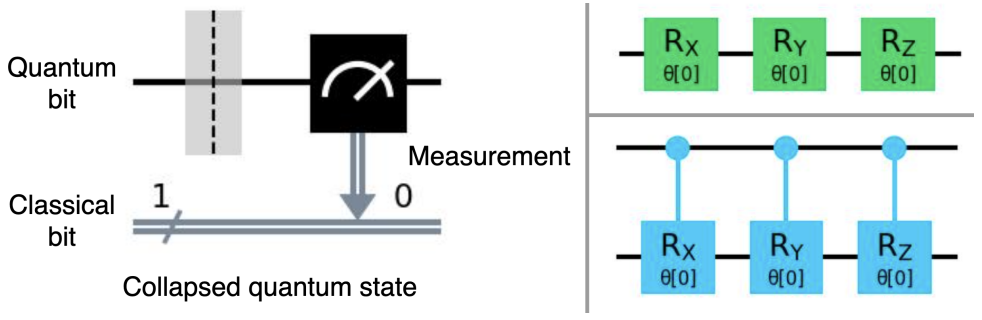

Figure 3: Circuit Notation. (Left:) Measurement: The black wire is a qubit and the grey double wire is a classical bit; the black box indicates measurement. (Top-right:) Rotation gates. (Bottom-right:) Controlled rotation gates. The top qubit is the control qubit for all three gates, while the bottom qubit acts as the target qubit.

we choose to use *amplitude encoding* since it takes the most advantage of the exponentially large state space. It encodes 2^N distinct floating-point values as the amplitudes of the state, thereby requiring only N qubits. Specifically, amplitude encoding takes a classical data vector $\mathbf{x} = (x_0, x_1, \ldots, x_{2^N-1}) \in [0,1]^{2^N}$ that is normalised to $\|\mathbf{x}\|_2 = 1$ and encodes it into the N-qubit quantum state $|\psi\rangle = \sum_{i=0}^{2^N-1}(x_i + 0j)\,|i\rangle$, where j is the imaginary unit.

Parametrised Quantum Gates. At its core, a quantum circuit applies a unitary transform $U_\theta \in \mathbb{C}^{2^N \times 2^N}$ (with parameters θ) to this initial state vector. In practice, U is implemented by sequentially applying *quantum gates*: $U = U_G \cdots U_2 U_1$. A quantum gate implements a simple unitary transform U_k that maps a state vector $|\psi\rangle$ to a new state vector $|\phi\rangle = U_k\,|\psi\rangle$. During training, we optimise for the best set of parameters θ of these gates.

We next discuss *hardware-efficient* gates that can be directly realised in quantum hardware. The parameter-free I, X-, Y-, and Z-Pauli gates act on a single qubit. The I gate is an identity map, while the X, Y, and Z gates rotate the qubit by 180° around the corresponding axis. The parametrised Pauli rotation gates R_X, R_Y, R_Z rotate the qubit by a given angle around the corresponding axis. For example, $R_X(\theta{=}20°)$ rotates the qubit by $\theta{=}20°$ around the X-axis, where θ is the parameter of the gate.

The controlled rotation gates CR_X, CR_Y, and CR_Z work on two qubits and they thus modify the entanglement of these qubits. One qubit acts as the *control qubit*: If it is in state $|0\rangle$, then the identity map is applied to the other qubit (called the *target qubit*); and if it is in state $|1\rangle$, then R_X, R_Y, or R_Z are applied to the target qubit. Importantly, if the control qubit is in a superposition, both operations are applied to the target qubit according to the superposition.

Fig. 3 shows the notation of these gates. Note that they form a *universal* set of gates [2]: They are enough to approximate *any* unitary transformation arbitrarily well as a finite sequence of them. A gate can occur more than once in the sequence and can be applied to any of the qubits each time. We also note that since gates are unitary transforms, they are reversible. In fact, each of these gates happens to be its own inverse (*e.g.* $(CR_X(20°))^\dagger = CR_X(-20°)$).

Output Measurement. Lastly, we measure each of the N qubits in a particular computational basis (typically along the Z-axis, which is also the basis we use in Sec. A.1), collapsing

the state vector into a binary vector of length N.

A.3 Circuit Design and Barren Plateaus

One widespread challenge when optimising the parameters θ of a quantum circuit are barren plateaus [1]. In some cases, the quantum nature of the task considered induces a particular circuit design and parameter choices, *e.g.* in quantum chemistry. Unfortunately, this approach is not feasible for generic tasks that are not inherently quantum-related. Instead, we need to follow the heuristic approach of designing generic circuits that can be efficiently implemented in hardware. In this heuristic setting, we usually follow a classical gradient-based optimisation routine that uses a cost function to iteratively compute updates to the parameters. Crucially, using more and more gates or qubits in generic designs leads to a loss landscape that is virtually flat (a barren plateau) in most places. The resulting per-parameter derivatives are essentially random, with smaller and smaller magnitudes and variances. These uninformative, vanishing gradients hinder the optimisation and thus restrict the size of the quantum circuits, limiting the expressibility.

A.4 Optimising Quantum Circuits

A further challenge unique to quantum circuits is that gradient-based optimisation inherently scales badly on real quantum hardware. Ultimately, this is because measurement irreversibly collapses the state. Thus, the *parameter shift* update rule requires evaluating the loss twice per parameter, which scales much worse than, for example, back-propagation on classical hardware. We avoid this issue in practice by resorting to simulation on classical hardware. This allows us to recover from the measurement process without having to re-run the circuit and we can thus apply back-propagation. In addition, simulation avoids noise, which remains prominent in contemporary quantum hardware.

B Normalisation Scheme

The dataset is a set of N classical 3D point clouds $[\{\mathbf{v}_i^j \in \mathbb{R}^3\}_{i=0}^{V-1}]_{j=0}^{N-1}$, each with V vertices. The probability vector containing the amplitudes restricts the output vector to the positive octant. To combat this, we take several steps to keep the raw input data within the positive octant as well.

We first define an axis-aligned bounding box $(\mathbf{v}_{min}, \mathbf{v}_{max})$ across the entire dataset. This is done by defining the minimum and maximum values along each axis:

$$v_{min,a} = \min_{\substack{j=0,\dots,N-1 \\ i=0,\dots,V-1}} v_{i,a}^j, \tag{1}$$

$$v_{max,a} = \max_{\substack{j=0,\dots,N-1 \\ i=0,\dots,V-1}} v_{i,a}^j, \tag{2}$$

where $v_{i,a}^j \in \mathbb{R}$ is the coordinate of vertex $\mathbf{v}_i^j$ along axis $a \in \{x,y,z\}$.

To achieve isotropic re-scaling, we turn the bounding box into a cube with side length $\mathbb{R} \ni s = \max_{a\in x,y,z} v_{max,a} - v_{min,a}$. We first shift the data and then re-scale it to get the final normalised dataset:

$$\left[\{\tilde{\mathbf{v}}_i^j\}_{i=0}^{V-1}\right]_{j=0}^{N-1} = \left[\left\{\frac{\mathbf{v}_i^j - \mathbf{v}_{min}}{s}\right\}_{i=0}^{V-1}\right]_{j=0}^{N-1}. \tag{3}$$