Layered Quantum Architecture Search for 3D Point Cloud Classification

Natacha Kuete Meli[1] Jovita Lukasik[1] Vladislav Golyanik[2] Michael Moeller[1]
[1]University of Siegen [2]MPI for Informatics, SIC
vsa.informatik.uni-siegen.de 4dqv.mpi-inf.mpg.de

Abstract

We introduce layered Quantum Architecture Search (layered-QAS), a strategy inspired by classical network morphism that designs Parametrised Quantum Circuit (PQC) architectures by progressively growing and adapting them. PQCs offer strong expressiveness with relatively few parameters, yet they lack standard architectural layers (e.g., convolution, attention) that encode inductive biases for a given learning task. To assess the effectiveness of our method, we focus on 3D point cloud classification as a challenging yet highly structured problem. Whereas prior work on this task has used PQCs only as feature extractors for classical classifiers, our approach uses the PQC as the main building block of the classification model. Simulations show that our layered-QAS mitigates barren plateau, outperforms quantum-adapted local and evolutionary QAS baselines, and achieves state-of-the-art results among PQC-based methods on the ModelNet dataset [1].

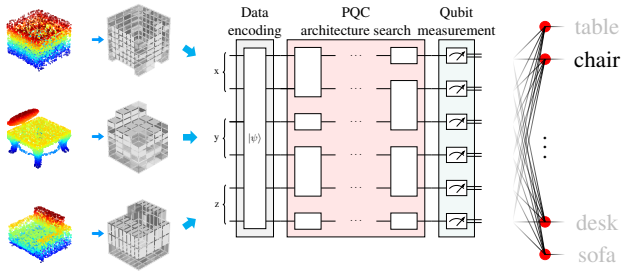

Figure 1. **Overview of our framework for 3D point classification using Parametrised Quantum Circuits (PQCs).** The input point cloud is voxelised and used to prepare the quantum system. We then use the new layered Quantum Architecture Search (layered-QAS) approach to engineer the PQC design that meaningfully learns features from the encoded point cloud. Lastly, qubits are measured to extract the learned features, which are used by an optionally learnable classical linear layer for classification.

1. Introduction

Quantum Machine Learning (QML) [3, 5, 26], through shallow-depth Parametrised Quantum Circuits (PQCs), is anticipated to pave the way for utility-scale quantum computing. PQCs operate in high-dimensional Hilbert spaces with comparably few parameters, leveraging superposition and entanglement to extract features beyond classical neural networks. However, designing effective PQC architectures for task-oriented feature extraction remains challenging. While classical models benefit from well-established architectural layers and inductive biases, PQCs lack such standardised building blocks. In addition, the same quantum properties that enable high expressiveness can also cause barren plateaus [30], where flattened loss landscapes hinder optimisation. Furthermore, PQCs are largely composed of linear transformations, missing the non-linear activations that underpin classical deep networks' representational power. In light of this, maximising PQC architectural design is crucial to ensure discriminative feature extraction while keeping the PQCs trainable and robust.

Quantum Architecture Search (QAS) offers a systematic way to address these design challenges by automatically exploring the space of PQC structures [32, 34, 59]. By replacing manual trial-and-error with guided search, QAS can discover expressive circuits for task-specific feature extraction, improve trainability, and mitigate barren plateaus. Existing QAS policies, however, face several limitations. Super-circuit and weight-sharing strategies [10, 47] not only search in very large spaces that are computationally demanding, but also often converge to suboptimal architectures, as candidate models can adversarially update the super-circuit parameters. Evolutionary and RL-based methods may struggle with scalability or training instability due to sensitivity to hyperparameters and reward design [39], while differentiable approaches [51] are prone to getting trapped in local optima. Moreover, none of these QAS methods account for PQC-specific considerations such as strategic training [11, 46] and careful initiali-

[1]project page: https://4dqv.mpi-inf.mpg.de/LQAS/

sation [15, 22, 48], which have been shown to mitigate barren plateaus, improve generalisation, and enhance feature learning.

In response to these limitations, we introduce *layered-QAS*. Building on ideas from the Lamarckian-based LEMONADE [12], our policy adds new layers to a pre-trained circuit and retains only those that provide the largest performance improvement. To further increase efficiency, we incorporate a pruning mechanism that removes gates operating near the identity, reducing unnecessary complexity without sacrificing expressivity. Together, these components make layered-QAS a robust framework for discovering high-performing PQCs.

We target 3D point cloud classification as a demanding yet structured exemplary application to evaluate our approach. The task involves assigning labels to objects represented by unordered sets of points, with applications in autonomous driving, robotics, and semantic segmentation [37, 41, 56]. Point clouds vary in density, resolution, and shape, while modern sensors produce increasingly large datasets. These characteristics make point cloud classification an ideal benchmark for QML, as it requires models to reason about spatial relationships in irregular, high-dimensional data. In this context, sQCNN-3D [1] applies quanvolutional filters to point cloud patches but delegates classification to a classical fully connected network—leaving the quantum model underutilised. In contrast, we pursue a fully quantum classification pipeline, minimising classical post-processing and improving feature extraction by the quantum circuit through architecture search. Applied to 3D multi-class point cloud classification, our pipeline enables near end-to-end quantum classification with only minimal classical components. An overview of our 3D classification framework is provided in Figure 1.

To summarise, our main contributions are as follows:

- A new layered-QAS policy for discovering improved and task-adapted PQC designs, avoiding manual trial-and-error approaches (Sec. 3; Sec. 3.2);
- A new framework for 3D point cloud classification based on PQCs and amplitude encoding of 3D data, leveraging layered-QAS for architectural design (Sec. 3.1).

We experiment on the ModelNet datasets [52] using a quantum computer simulator and show that our models outperform the existing sQCNN-3D quantum baseline [1] and are even competitive with a classical baseline of similar expressivity that we design. Our search strategies can find parameter-efficient PQC architectures. Moreover, our layered search achieves better performance than the evolutionary search and a QAS-adapted local search [49], a baseline that incrementally improves PQC architectures by making small localised changes to the PQC and retains the changes that enhance performance.

2. Related Work

2.1. PQC Training and QAS

Strategic PQC Training. Strategic training of PQCs is believed to dampen trainability issues. Skolik et al. [46] proposed a layer-wise training approach, which incrementally grows the circuit depth, freezes previously learned parameters, and optimises only the parameters of added layers. This layered training strategy not only speeds up the training but also experimentally proved to increase the generalisation error. Similarly, the method proposed by Duffy et al. [11] allows not only for the addition of new parametrised gates, but also feature-map encodings of the data to incrementally grow the circuits. Data re-uploading is proven to increase the expressivity of PQCs [43]. Both methods suggest that incrementally training PQCs is a promising approach to enhance PQCs' expressivity without compromising their trainability. Note that a layered training approach was also used by Krahn et al. [24] for binary networks with quantum annealed gradients.

Strategic PQC Initialisation. Random initialisation of PQCs was shown to be one cause of barren plateaus [17, 35]. Grant et al. [15] proposed a selective initialisation method that randomly selects only a subset of the initial parameter values and chooses the remaining ones so that the circuit is a sequence of shallow blocks that evaluate to the identity, limiting the circuit depth in the first parameter update. Wang et al. [48] proved that reducing the initial domain of each parameter inversely proportional to the square root of the circuit depth causes the magnitude of the cost gradient to decay at most polynomially with respect to the qubit count and the circuit depth. Kashif et al. [22] empirically showed that initialising parameters within smaller distribution ranges, with lower magnitudes, tends to perform better than using larger ranges with higher magnitudes.

Quantum Architecture Search (QAS). QAS methods often use a super-circuit to define the search pool of candidate circuits, as seen in early works by Du et al. [10] and Wang et al. [47], where training updates the super-circuit before selecting and fine-tuning the best candidate. Ma et al. [33] enhanced this with an evolutionary post-training process, while other evolutionary approaches forgo super-circuits, training architectures from scratch [6, 55]. Weight-sharing improves memory efficiency but may cause suboptimal convergence. Differentiable QAS methods, like those by Wu et al. [51] and Zhang et al. [57], optimize the search domain, while Reinforcement Learning (RL)-based methods [8, 27, 28, 39, 53] use neural agents to identify effective architectures. However, differentiable methods risk favoring local optima, and RL methods may face instability due to hyper-parameters and reward function designs.

Our LEMONADE-[12]-inspired *layered-QAS* combines informed initialisation with progressive layer-wise training. (i) Candidate layer architectures are briefly trained and ranked to discard weak options; (ii) when new layers are added, previously learned parameters remain trainable and continue to improve. This warm-starting lets child models inherit and refine the performance of their parents.

2.2. Classification using PQCs

PQC Classification and Architectures. Few methods have explored 3D point cloud classification with PQCs, as most prior QML works focused on binary classification on small datasets like Moons, Iris or heavily downsampled MNIST datasets; see Refs. [4, 13, 26] for an overview.

Binary and 2D classification have long served as standard PQC benchmarks. Farhi *et al.* [14] and Cong *et al.* [7] were among the first to design quantum neural networks for binary classification. Havlíček *et al.* [18] proposed a quantum variational classifier using a feature-map encoding to project data into high-dimensional spaces. Henderson *et al.* [19] introduced quanvolutional filters for PQC-based patch feature extraction. Salinas *et al.* [40] developed a universal classifier using data re-uploading, showing that a single-qubit PQC can serve as a universal classifier when target classes correspond to specific Bloch-sphere states—though in multi-class settings, class-state non-orthogonality induces correlations. More recent works [21, 29, 44] extended these models to image datasets such as MNIST [9], CIFAR [25], and GTSRB [20].

To the best of our knowledge, the work by Baek et al. [1] is the only PQC approach to 3D point cloud classification. Their method, sQCNN-3D, employs multiple PQCs that are trained in parallel and behave as filters for feature extraction. Those so-called quanvolution filters process small patches of the voxelised cloud encoded using angle encoding a data re-uploading. To mitigate barren plateaus, these PQCs remain small in size and depth. Similar to classical convolutional filters, each PQC uses the same parameters to process all the inputted patches. Features are extracted by locally measuring the qubits and concatenated into a feature vector that is further processed by an MLP, which has significantly more parameters than the PQCs. Due to this overwhelming classical part of the model, it is unclear whether the PQCs still play a role in the model's performance.

In contrast, we propose a quantum-driven alternative that minimises classical components: voxelised inputs are amplitude-encoded into PQCs, features are extracted directly via quantum measurements, and classification is performed with at most one final classical linear layer.

3. 3D Point Classification with Layered-QAS

This section presents our layered-QAS policy for PQC design. To set the stage, we first outline the classification pipeline in Section 3.1, followed by the presentation of our layered-QAS policy in Section 3.2. A brief overview of gate-based quantum computing is provided in Appendix A.

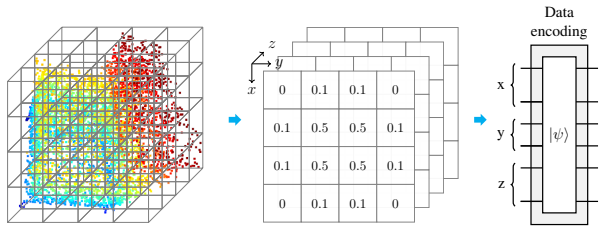

Figure 2. **Amplitude encoding of a 3D point cloud into a quantum state.** We partition the point cloud into voxels (left) and compute the normalised density of each voxel, *i.e.* the proportion of points within the voxel (middle). This creates a matrix that we vectorise and use as input state vector for the PQC (right). The granularity of the voxelisation, which defines the number of qubits used, is user-defined. Colours are added for visualisation purposes.

3.1. Workflow

3.1.1 3D Point Cloud Encoding

We use amplitude encoding as described in Figure 2 to encode the input point cloud in a quantum state vector.

Given is a point cloud $\tilde{\mathcal{P}} = \{\tilde{p}^i = (\tilde{p}^i_x, \tilde{p}^i_y, \tilde{p}^i_z)\}_{i=1}^N$ and a voxel granularity $k \in \mathbb{N}$. First, we normalise the point cloud $\tilde{\mathcal{P}}$ into $\mathcal{P} = \{p^i = (p^i_x, p^i_y, p^i_z)\}_{i=1}^N$ by fitting it into the 3D unit cube that we later scale by $2^k - 1$. We then partition this cube in each dimension into 2^k sub-cubes called voxels. Next, for each voxel that we index by integer coordinates (x, y, z), we compute the proportion or density

$$\delta_{xyz} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}\big(p^i_x \in V_x, p^i_y \in V_y, p^i_z \in V_z\big), \qquad (1)$$

of points within it, with $V_x = [x, x{+}1)$, $V_y = [y, y{+}1)$, and $V_z = [z, z+1)$ representing the voxel boundaries, and $\mathbb{I}(\cdot)$ returning 1 if the input condition is true, and 0 otherwise. Finally, we use the computed densities to form a normalised state vector

$$|\psi\rangle = \sum_{x,y,z} \sqrt{\delta_{xyz}}\, |xyz\rangle \qquad (2)$$

in which we prepare the quantum system before the PQC transformation, with $|xyz\rangle = |x\rangle \otimes |y\rangle \otimes |z\rangle$ being a composed system of three registers of k qubits each that encode the x, y and z coordinates of the voxels in the binary basis. Thus, the overall encoding consists of $3k$-many qubits. Since the point cloud occupies only a subregion of the cube, the state vector in Equation (2) is sparse within the 2^{3k}-dimensional Hilbert space, enabling a relatively efficient state-preparation [31, 42]. Example voxelised point clouds for different k are visualised in Appendix B.

3.1.2 PQC Transformation

The encoded quantum state vector undergoes several layers of parametrised quantum gates forming the PQC, see Figure 1. These gates transform the initial state vector into a final state that can be measured and post-processed. A PQC of ℓ layers can be densely represented as an operator $\mathbf{U}(\boldsymbol{\theta})$, with $\boldsymbol{\theta}$ being the set of parameters to be optimised. It transforms the initial state vector into a parametrised state

$$|\psi(\boldsymbol{\theta})\rangle = \mathbf{U}(\boldsymbol{\theta})\,|\psi\rangle = \mathbf{L}_\ell(\theta_\ell)\cdots\mathbf{L}_1(\theta_1)\,|\psi\rangle\,. \tag{3}$$

Each layer $\mathbf{L}_i(\theta_i)$ consists of a combination of single-qubit and/or multi-qubit gates. The single-qubit gates we use are Pauli $\mathrm{RX}, \mathrm{RY}, \mathrm{RZ}$ rotations:

$$\mathrm{RX}(2\theta) = \begin{pmatrix} \cos(\theta) & -i\sin(\theta) \\ -i\sin(\theta) & \cos(\theta) \end{pmatrix}, \tag{4}$$

$$\mathrm{RY}(2\theta) = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}, \quad \text{and} \tag{5}$$

$$\mathrm{RZ}(2\theta) = \begin{pmatrix} e^{-i\theta} & 0 \\ 0 & e^{i\theta} \end{pmatrix}. \tag{6}$$

Multi-qubit entangling gates introduce correlations between qubits. We use two-qubit controlled RX rotations

$$\mathrm{CRX}(2\theta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos(\theta) & -i\sin(\theta) \\ 0 & 0 & -i\sin(\theta) & \cos(\theta) \end{pmatrix}. \tag{7}$$

This set of gates is similar to the well-known universal set $\{\mathrm{RX}, \mathrm{RY}, \mathrm{RZ}, \mathrm{Phase}, \mathrm{CNOT}\}$ [38, 50]. The optimal parameters $\boldsymbol{\theta}$ for the classification task should be found by minimising the classification loss, while the circuit architecture will be designed by automated layered-QAS.

3.1.3 Measurement

We perform local qubit measurements by measuring each qubit individually. As the PQC uses the three Pauli $\mathrm{RX}, \mathrm{RY}, \mathrm{RZ}$ rotations to position the qubits on the Bloch sphere, we measure in the Pauli $\mathrm{X}, \mathrm{Y}, \mathrm{Z}$ bases.

The Pauli observables M_q, with $\mathrm{M} \in \{\mathrm{X}, \mathrm{Y}, \mathrm{Z}\}$ applied on the qth qubit with the identity to the remaining qubits, measure the probability distribution for outcomes in the M bases. The corresponding expectation values

$$\langle \mathrm{M}_q(\boldsymbol{\theta})\rangle = \langle\psi(\boldsymbol{\theta})|\mathrm{M}_q|\psi(\boldsymbol{\theta})\rangle \tag{8}$$

are returned for the three bases, with $|\psi(\boldsymbol{\theta})\rangle$ from Equation (3) being the state of the system after the PQC transformation. In total, the expectation values returned constitute a set of $3\cdot 3k$ learned features, with k being the number of qubits of each coordinate register.

Algorithm 1 Layered Quantum Architecture Search

Require: Layer architectures, number T of layer types, training and validation sets, ranking metric Best.

```
U_0 := I
for i = 0, 1, 2, ... do
    ArchList := [ ]                                  ▷ Empty list
    Layer type t := i mod T
    L_t = {Random set of layers of type t}
    for L_{i+1} ∈ L_t do
        U_candidate = L_{i+1} U_i
        Train U_candidate for a few epochs
        Append U_candidate to ArchList
    Update U_{i+1} = Best(ArchList)                  ▷ Ranking
Return U_{i+1}
```

3.1.4 The Loss Function

Let c be the number of class labels. As mentioned above, the QPC outputs $3\cdot 3k$ learned features, which may not correspond to the number c of classes. We use a classical linear layer to map these features into a length-c logit vector, which is then compared by the cross-entropy loss function

$$\mathcal{L}_{\mathrm{CE}} = -\sum_{j=1}^{c} y_j \log \hat{p}_j \tag{9}$$

to the one-hot encoded ground-truth label $\mathbf{y} = (y_1, \ldots, y_c)$, with $\hat{\mathbf{p}} = (\hat{p}_1, \ldots, \hat{p}_c)$ being the predicted probability distribution obtained by applying softmax to the logits.

Ideally, the linear layer is trainable so it can learn weights that map PQC outputs to the correct logits. To assess whether the PQCs themselves learn meaningful features, we also test a non-trainable linear layer implemented as a fixed random Gaussian projection, i.e., a randomly initialized matrix with entries drawn from a normal distribution.

3.2. Layered Quantum Architecture Search

We propose to find a suitable combination of the gates described in Section 3.1.2 for a PQC architecture via a new layered search methodology that is outlined in Algorithm 1 and illustrated in Figure 3. We denote by $\mathbf{U}_i(\boldsymbol{\theta}_i)$ the unitary operator of the PQC at generation i.

We begin with a trivial identity circuit $\mathbf{U}_0(\boldsymbol{\theta}_0) = \mathbf{I}$, which consists of data encoding followed by measurement. At each generation $(i+1)$, the existing circuit $\mathbf{U}_i(\boldsymbol{\theta}_i)$ is expanded by adding a new layer $\mathbf{L}_{i+1}(\theta_{i+1})$. To explore the best possible architecture, different designs of the same layer type are tested, each yielding a new PQC candidate:

$$\mathbf{U}_{\text{candidate}}(\boldsymbol{\theta}_{i+1}) = \mathbf{L}_{i+1}(\theta_{i+1})\mathbf{U}_i(\boldsymbol{\theta}_i). \tag{10}$$

The type of layer alternates across generations. For a fair comparison, the parameters of the previous generation remain unchanged in all candidate PQCs. Furthermore, newly

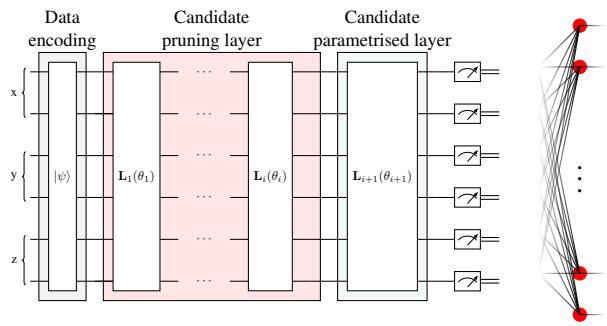

Figure 3. **Workflow of our layered-QAS.** At generation $(i+1)$, the algorithm expands the PQC with a layer $\mathbf{L}_{i+1}(\theta_{i+1})$, chosen from single-qubit, entangling, or pruning layers. Single-qubit and entangling layers add new parametrised gates (green), while pruning layers remove gates (red). Several candidate layers are evaluated, and the most effective circuit extension is retained.

added layers are designed to almost preserve the accuracy of $\mathbf{U}_i(\boldsymbol{\theta}_i)$ at the start of the training. This is achieved, for example, by initialising the parameters of added layers to zero, ensuring that they initially act as the identity.

Each PQC candidate $\mathbf{U}_{\text{candidate}}(\boldsymbol{\theta}_{i+1})$ undergoes a few training epochs. After training, candidates are ranked based on their highest performance on the validation set during the last training epoch. The most effective circuit

$$\mathbf{U}(\boldsymbol{\theta}_{i+1}) = \underset{\mathbf{U}\in\{\mathbf{U}_{\text{candidate}}\}}{\arg\max} \ \text{Best}(\mathbf{U}_{\text{candidate}}(\boldsymbol{\theta}_{i+1})), \qquad (11)$$

along with its optimised parameters $\boldsymbol{\theta}_{i+1}$, is selected for the next generation. Our ranking metric $\text{Best}(\cdot)$ is the classification accuracy on the validation data. The process iterates, progressively building a more and more expressive and effective PQC by systematically selecting the best-performing circuits from each generation.

Layer Architectures. The layered architectures we consider for the point cloud classification are illustrated in Figure 4. Three types of layers are considered, i.e., single-qubit layers, entangling layers, and pruning layers:

- **Single-qubit layers** apply single-qubit Pauli $\text{RX}, \text{RY}, \text{RZ}$ rotation gates, enabling the circuit to learn single-qubit transformations. Candidate layers at a given generation are architectures 0, 1 and 2.
- **Entangling layers** introduce controlled rotations between qubit pairs, allowing the circuit to learn entanglement patterns and enhance its expressivity [45]. We experiment with CRX gates. Although some architecture pairs in Figure 4 (e.g., 4&5, 6&7, 8&9) appear similar, they differ in how control is applied. In architecture 4, each qubit except the first is controlled by all its predecessors, whereas in architecture 5 it is controlled only by its immediate predecessor. At each generation, three entangling-layer candidates are sampled uniformly at random from architectures 3 to 9.
- **Pruning layers** randomly remove a proportion of variational gates from the previous circuit $\mathbf{U}_i(\boldsymbol{\theta}_i)$ if the absolute value of their parameters falls below a predefined dropout threshold. The intuition is that gates with small rotation angles contribute minimally to the computation while increasing circuit complexity. By eliminating them, we reduce computational cost without significantly sacrificing the accuracy. Candidate pruning layers at a given generation are three pruning layers that randomly select gates to prune among the ones with small angles.

Layer types are explored cyclically. We, however, note that different layer types could also go into the candidate dimen-

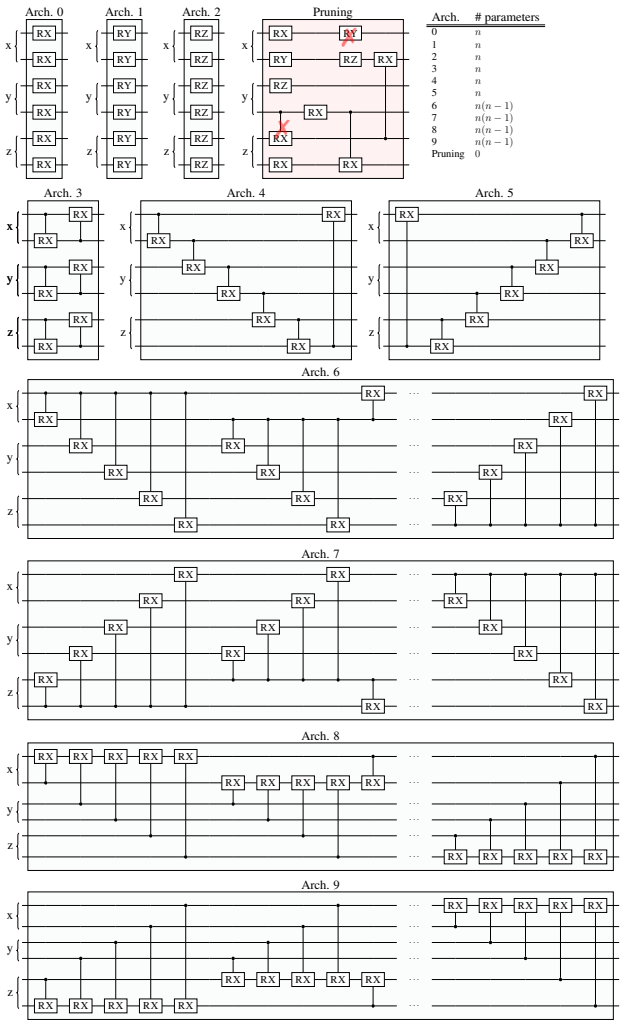

Figure 4. **Layers tested in our layered-QAS.** The number of parameters for each layer is shown at the top right. Special architectures include the pruning layer, which deletes gates, and architecture 3, which circularly applies CNOTs on coordinate registers.

sion. Our layered search enables a structured and principled exploration of the PQC architectures, progressively refining their performance while maintaining efficiency.

4. Experimental Results

In this section, we present the results of our layered-QAS strategy in the context of 3D point cloud classification. After presenting the implementation and evaluation details, we begin in Section 4.1, by summarising the 3D classification results of the layered-QAS to evaluate the performance of our quantum model in comparison with quantum and classical baselines. Next, in Section 4.2, we benchmark our layered search against alternative QAS policies. Finally, in Section 4.3, we justify key design choices through ablation studies on the voxel granularity and the gate pruning threshold, and briefly discuss runtimes.

Implementation Details. The code is written in Python using the PennyLane framework [2]. In PennyLane, the quantum tape —sequence of gates in a quantum circuit— is stored in a Python list that can be easily modified to alter the circuit architecture. All experiments are simulated and performed in the idealised noise-free setting, and gradients for training are computed with automatic differentiation. We use a NVIDIA GeForce RTX 4090 GPU.

Evaluation Methodology. We experiment on the ModelNet10 and ModelNet40 datasets [52], with 10 and 40 labels respectively, to classify the point clouds into. ModelNet samples are object triangle meshes, from which we uniformly sample 5000 3D points at random to generate object point clouds for classification. ModelNet10 originally has 3991 training and 908 test samples, while ModelNet40 has 9842 training and 2468 test samples. We take out 20% of the original training samples to form the validation set on each dataset and keep the test set intact for evaluation. The granularity of the voxelisation is $k = 3$ in each of the x, y and z dimensions, leading to 9 qubits on which the PQCs operate. We reduce the datasets by 90% during the architecture search phase, but use the full datasets for fine-tuning after the search. We observed that the search on full, large datasets causes the parameters in the evolutionary search in Section 4.2 to deeply adapt to the candidate models, resulting in highly conflicting updates to the super-circuit parameters (see the discussion in Appendix D). Conversely, the search on smaller datasets enables the selection of candidate models that generalise better. Benchmark classification results are reported based on the full datasets.

All models are optimised with the ADAM optimiser [23] and a learning rate $\text{lr} = 0.1$ in the search phase and $\text{lr} = 0.03$ in the fine-tuning phase. Each candidate model is trained for 5 epochs. The classification performance metric is the standard top-1 accuracy. The gate pruning threshold is $\pi/10$ in our layered-QAS.

4.1. Resuts on ModelNet10&40

We benchmark our approach in a shallow version obtained after 10-layer search iterations, as well as a deeper version after 20 search iterations on the full ModelNet10 and ModelNet40 datasets against the following competitors:

- The only prior work on PQCs for 3D point cloud classification, the sQCNN-3D method [1], with 2 quanvolution filters as tested in the original publication.
- A vanilla CNN with one 3D convolution, one ReLU layer and one fully-connected layer to keep the parameter count and expressivity of the models comparable. For instance, the 3D convolutional layer is a linear transformation as the PQCs, the ReLU activation is non-linear as the measurement of the quantum systems, and the last linear layer is used for classification.

We train the benchmark models for 20 epochs with a higher learning rate on the reduced dataset, and subsequently fine-tune all models over 10 more epochs with a lower learning rate on the full datasets.

As we can see from the results in Table 1, both variants of the proposed approach outperform the prior work on 3D point cloud classification with PQCs as well as the simple CNN baseline on both benchmark datasets in terms of validation and test accuracy. Remarkably, our reduced PQC model after 10 layered search iterations obtains high accuracy at less than half the number of learnable quantum parameters, trailing the deeper (20-iteration) model but surprisingly few percents only. To get insight into shapes that are challenging to classify, we show confusion matrices on ModelNet10 in Appendix D.3.

4.2. Benchmark with QAS Methods

We ablate the proposed layered search against two alternatives: 1) An adaptation of a classical local search algorithm to the setting of PQCs, and 2) an evolutionary search approach similar to other QAS methods [10, 33], making use of a super-circuit inspired by the classical one-shot architecture search approach in Guo et al. [16]. We evaluate the evolutionary search in two different variants: One where the parameters are directly taken from the super-circuit and one where the current architecture is fine-tuned. Evolutionary search models with fine-tuning are suffixed by "ft". Details as well as an algorithmic description for both approaches can be found in Appendix C.

Our layered-QAS is performed over 20 generations, which corresponds to 13 parametrised layers and 7 pruning layers added. In each generation, we train and evaluate 3 candidate models. The super-circuit for the evolutionary search also has 20 layers, but the number of parameters per layer may be different than in the layered case. The super-

	ModelNet10					ModelNet40				
	NPQ	NPC	TR.	VA.	TS.	NPQ	NPC	TR.	VA.	TS.
Layered$_{10}$ [Ours]	100	270	**92**%	92%	84%	105	1080	59%	56%	54%
Layered$_{20}$ [Ours]	279	270	**92**%	**93**%	**85**%	279	1080	**61**%	**59**%	**55**%
sQCNN-3D [1]	48	650	83%	79%	72%	48	2600	47%	45%	41%
Vanilla CNN	0	590	90%	88%	82%	0	1580	60%	55%	54%

Table 1. **Resources and performance comparison for the models.** Keys: "NP(Q/C)"=Number of parameters in the quantum or classical backbones; "TR./VA./TS."=Train/Validation/Test top-1 accuracy; The subscript in our layered approach refers to the number of layered-QAS iterations. Our proposed approach consistently outperforms the baselines in all settings.

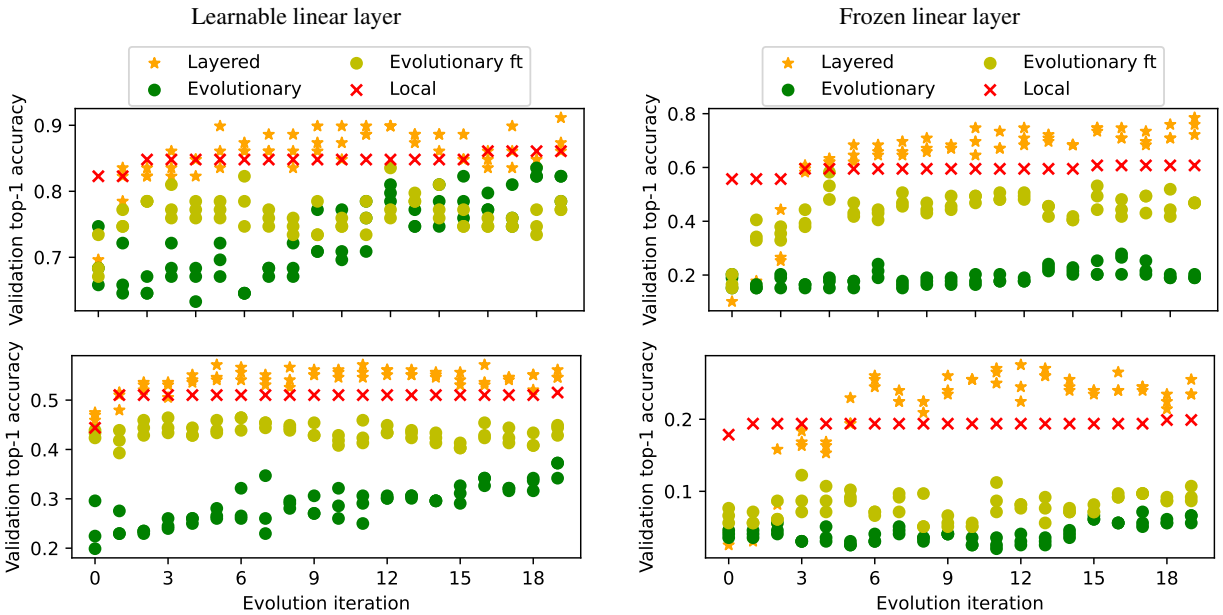

Figure 5. **Convergence of the search algorithms on the reduced ModelNet10 (top) and ModelNet40 (bottom) datasets for models with learnable (left) and frozen (right) linear layers.** Shown are the accuracies of the top three candidates for the layered and evolutionary searches, and the single candidate for the local search. Layered search steadily improves accuracy by increasing PQC expressivity. Evolutionary search converges faster with fine-tuning, while the version without fine-tuning can occasionally find better models but struggles when the linear layer is frozen. Local search exhibits a step-like behavior, as the model is updated only when an improvement is found.

circuit is trained with 100 randomly sampled architectures. The evolutionary search itself is performed over 20 generations for a population of size 10, from which only the top-5 architectures are considered to generate new ones.

Figure 5 presents the behaviours of the different search procedures on ModelNet10 and ModelNet40 over the course of the iterations, and in addition, Table 2 shows the training, validation and test accuracies for the best models found by each QAS for two different settings: Our standard setting in which the last (classical) linear layer is learnable as well as a setting with a frozen linear layer with random weights that gives an impression of how powerful the PQC alone (without classical components) is.

The results of the evolutionary search are from the evolutionary search itself, *i.e.*, after the super-circuit training. In Appendix D.2, we discuss the performance of the found models when trained from scratch after the search.

While all search strategies yield an increase in validation accuracy, one can see a clearly favourable behaviour of the layered search in both settings, with and without a learnable classical linear layer. It is interesting to observe the high jump of the layered-search accuracy of models with frozen linear layers at generation 2. This jump corresponds to the addition of the first entangled layer, which significantly increases the circuit's expressivity.

While the performance for all search strategies is higher in the learnable than in the frozen linear layer setting, it is remarkable that –at least in the case of ModelNet10– our proposed approach reaches almost 80% validation accuracy for a random linear layer, indicating a highly expressive PQC part. In comparison, we ran sQCNN-3D on ModelNet10 with a frozen MLP to obtain 17.4% validation and 15.0% test accuracy only, indicating a significant advantage in the expressiveness of our PQC architecture. In addition,

	ModelNet10					ModelNet40				
	NPQ(L/F)	NPC	TR.(L/F)	VA.(L/F)	TS.(L/F)	NPQ(L/F)	NPC	TR.(L/F)	VA.(L/F)	TS.(L/F)
Layered$_{10}$ [Ours]	100/207	270	**92**/**78**%	92/**78**%	84/67%	105/164	1080	59/**27**%	56/**26**%	54/**21**%
Layered$_{20}$ [Ours]	279/282	270	**92**/**78**%	**93**/**78**%	**85**/**69**%	279/220	1080	**61**/22%	**59**/22%	**55**/16%
Local search	163/165	270	90/75%	89/75%	83/65%	166/166	1080	60/20%	58/18%	54/13%
Evolutionary [16, 33]	162/163	270	88/68%	85/65%	80/53%	162/168	1080	59/18%	57/17%	52/12%
Evolutionary ft. [16, 33]	163/176	270	89/69%	85/68%	81/56%	163/166	1080	59/17%	58/16%	53/11%

Table 2. **Ablation study on the choice of QAS:** "NP(Q/C)"=Number of parameters in the quantum or classical backbones; "TR./VA./TS."=Train/Validation/Test top-1 accuracy; "L/F"=Search models with learnable and frozen linear layers; "ft."=Fine tuning. NPC is 0 for models with frozen linear layers. The subscript in the layered model refers to the number of generations.

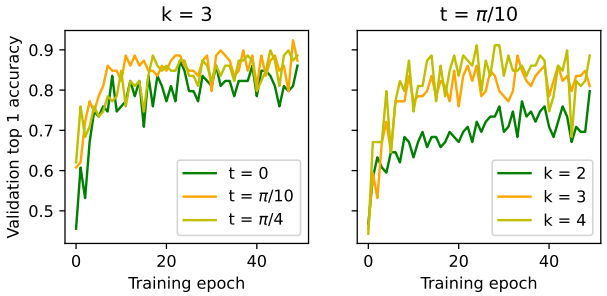

Figure 6. **Performance** of the layered search for different pruning thresholds t and voxelisation granularity k on the reduced ModelNet10. Pruning reduces the number of parameters without sacrificing accuracy. Increasing k increases the prediction accuracy.

our sQCNN-3D-implementation yields lower accuracy than originally reported [1], likely because the original work employs a fully connected network (with no architectural details provided) for classification, whereas we employ only a single fully connected layer. This simplification is sufficient to highlight the superior accuracy–parameter trade-off of our layered-QAS approach.

4.3. Ablation Studies and Runtimes

For the model with a learnable linear layer, Figure 6 shows the convergence behaviour for different pruning thresholds t and voxel granularity k on the reduced ModelNet10. We see that pruning gates with absolute parameter values below t do not worsen the prediction accuracy while reducing the PQCs' depths. Pruning at $t = \pi/10$ and $t = \pi/4$ reduced the number of gates from 126 to 86 and 189 to 148 respectively. We note that other pruning methods based on Fisher information exist that guarantee not to affect the PQC expressivity [17], but are more computationally expensive as they require the Fisher information matrix to be computed.

Finer voxel grids improve the prediction, as they allow capturing more details in the object shapes. However, the performance for $k = 4$ being only slightly better than for $k = 3$ suggests that other factors are much more significant than the voxel granularity beyond $k = 3$.

Runtimes. On the reduced ModelNet10, the layered search took 2 hours. Those times multiply by 3 for the search on the reduced ModelNet40, and by 10 on full datasets. Fine-tuning all search models took about 0.5 and 1 hours on the full ModelNet10 and ModelNet40 datasets, respectively. In comparison, the training of the sQCNN-3D and vanilla CNN took 10 and 4 hours on the full ModelNet10, which multiplies by 3 on the full ModelNet40.

5. Discussion and Conclusion

We evaluated PQCs discovered via our layered-QAS policy on 3D point cloud classification using the ModelNet datasets, combining amplitude encoding, QAS-found PQCs, and classical linear layers as principal components. Our models outperformed the existing quantum sQCNN-3D and, when matched in expressivity to a purely classical baseline, achieved competitive accuracy with far fewer parameters. Even with frozen classical layers, the QAS-discovered PQCs alone learned meaningful and discriminative features, underscoring the effectiveness of our architecture search. These results demonstrate that layered-QAS can identify task-specific quantum architectures that balance expressivity and trainability, making them viable for challenging, structured domains like 3D classification.

A fundamental remaining problem is the under-expressivity of the considered models, mainly consisting of linear PQC operations, with quantum measurements being the only non-linear functions. Future work should investigate expressive PQC designs, such as those incorporating non-linearities via intermediate measurements. Another interesting direction would be a binary encoding of the input in basis states, on which linear operations are provably expressive enough to approximate any output. This alternative would require far more qubits, making it a goal for future quantum hardware advancements.

Acknowledgements. This work was supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation), project number 534951134. NKM and MM acknowledge support by the Lamarr Institute for Machine Learning and Artificial Intelligence.

References

[1] Hankyul Baek, Won Joon Yun, Soohyun Park, and Joongheon Kim. Stereoscopic scalable quantum convolutional neural networks. *Neural Networks*, 165:860–867, 2023. 2, 3, 6, 7, 8

[2] Ville Bergholm, Josh Izaac, Maria Schuld, Christian Gogolin, Shahnawaz Ahmed, Vishnu Ajith, M Sohaib Alam, Guillermo Alonso-Linaje, B AkashNarayanan, Ali Asadi, et al. Pennylane: Automatic differentiation of hybrid quantum-classical computations. *arXiv preprint arXiv:1811.04968*, 2018. 6

[3] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning. *Nature*, 549(7671):195–202, 2017. 1

[4] Joseph Bowles, Shahnawaz Ahmed, and Maria Schuld. Better than classical? the subtle art of benchmarking quantum machine learning models. *arXiv preprint arXiv:2403.07059*, 2024. 3

[5] Marco Cerezo, Guillaume Verdon, Hsin-Yuan Huang, Lukasz Cincio, and Patrick J Coles. Challenges and opportunities in quantum machine learning. *Nature computational science*, 2(9):567–576, 2022. 1

[6] D Chivilikhin, A Samarin, V Ulyantsev, I Iorsh, AR Oganov, and O Kyriienko. Mog-vqe: Multiobjective genetic variational quantum eigensolver. *arXiv preprint arXiv:2007.04424*, 2020. 2

[7] Iris Cong, Soonwon Choi, and Mikhail D Lukin. Quantum convolutional neural networks. *Nature Physics*, 15(12): 1273–1278, 2019. 3

[8] Xin Dai, Tzu-Chieh Wei, Shinjae Yoo, and Samuel Yen-Chi Chen. Quantum machine learning architecture search via deep reinforcement learning. In *Quantum Computing and Engineering*, pages 1525–1534. IEEE, 2024. 2

[9] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012. 3

[10] Yuxuan Du, Tao Huang, Shan You, Min-Hsiu Hsieh, and Dacheng Tao. Quantum circuit architecture search for variational quantum algorithms. *npj Quantum Information*, 8(1): 62, 2022. 1, 2, 6

[11] Callum Duffy, Smit Chaudhary, and Gergana V Velikova. Quantum circuit training with growth-based architectures. *arXiv preprint arXiv:2411.16560*, 2024. 1, 2

[12] Thomas Elsken, Jan Metzen, and Frank Hutter. Efficient multi-objective neural architecture search via lamarckian evolution. *arXiv preprint arXiv:1804.09081*, 2018. 2, 3

[13] Fan Fan, Yilei Shi, Tobias Guggemos, and Xiao Xiang Zhu. Hybrid quantum-classical convolutional neural network model for image classification. *Neural networks and learning systems*, 2023. 3

[14] Edward Farhi and Hartmut Neven. Classification with quantum neural networks on near term processors. *arXiv preprint arXiv:1802.06002*, 2018. 3

[15] Edward Grant, Leonard Wossnig, Mateusz Ostaszewski, and Marcello Benedetti. An initialization strategy for addressing barren plateaus in parametrized quantum circuits. *Quantum*, 3:214, 2019. 2

[16] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. In *European conference on Computer visio*, pages 544–560. Springer, 2020. 6, 8, 2

[17] Tobias Haug, Kishor Bharti, and MS Kim. Capacity and quantum geometry of parametrized quantum circuits. *PRX Quantum*, 2(4):040309, 2021. 2, 8

[18] Vojtěch Havlíček, Antonio Córcoles, Kristan Temme, Aram Harrow, Abhinav Kandala, Jerry Chow, and Jay Gambetta. Supervised learning with quantum-enhanced feature spaces. *Nature*, 567(7747):209–212, 2019. 3

[19] Maxwell Henderson, Samriddhi Shakya, Shashindra Pradhan, and Tristan Cook. Quanvolutional neural networks: powering image recognition with quantum circuits. *Quantum Machine Intelligence*, 2(1):2, 2020. 3

[20] Sebastian Houben, Johannes Stallkamp, Jan Salmen, Marc Schlipsing, and Christian Igel. Detection of traffic signs in real-world images: The German Traffic Sign Detection Benchmark. In *International Joint Conference on Neural Networks*, number 1288, 2013. 3

[21] Yu Jing, Xiaogang Li, Yang Yang, Chonghang Wu, Wenbing Fu, Wei Hu, Yuanyuan Li, and Hua Xu. Rgb image classification with quantum convolutional ansatz. *Quantum Information Processing*, 21(3):101, 2022. 3

[22] Muhammad Kashif and Muhammad Shafique. The dilemma of random parameter initialization and barren plateaus in variational quantum algorithms. *arXiv preprint arXiv:2412.06462*, 2024. 2

[23] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[24] Maximilian Krahn, Michele Sasdelli, Fengyi Yang, Vladislav Golyanik, Juho Kannala, Tat-Jun Chin, and Tolga Birdal. Projected stochastic gradient descent with quantum annealed binary gradients. *BMVC*, 2024. 2

[25] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images., 2009. 3

[26] Natacha Kuete Meli, Shuteng Wang, Marcel Seelbach Benkner, Michele Sasdelli, Tat-Jun Chin, Tolga Birdal, Michael Moeller, and Vladislav Golyanik. Quantum-enhanced computer vision: Going beyond classical algorithms. *arXiv e-prints*, page arXiv, 2025. 1, 3

[27] Akash Kundu, Aritra Sarkar, and Abhishek Sadhu. Kanqas: Kolmogorov-arnold network for quantum architecture search. *EPJ Quantum Technology*, 11(1):76, 2024. 2

[28] En-Jui Kuo, Yao-Lung L Fang, and Samuel Yen-Chi Chen. Quantum architecture search via deep reinforcement learning. *arXiv preprint arXiv:2104.07715*, 2021. 2

[29] Sylwia Kuros and Tomasz Kryjak. Traffic sign classification using deep and quantum neural networks. In *International Conference on Computer Vision and Graphics*, pages 43–55. Springer, 2022. 3

[30] Martin Larocca, Supanut Thanasilp, Samson Wang, Kunal Sharma, Jacob Biamonte, Patrick J Coles, Lukasz Cincio, Jarrod R McClean, Zoë Holmes, and M Cerezo. A review of barren plateaus in variational quantum computing. *arXiv preprint arXiv:2405.00781*, 2024. 1, 2

[31] Lvzhou Li and Jingquan Luo. Nearly optimal circuit size for sparse quantum state preparation. In *International Colloquium on Automata, Languages and Programming*, 2024. 3

[32] Xudong Lu, Kaisen Pan, Ge Yan, Jiaming Shan, Wenjie Wu, and Junchi Yan. Qas-bench: rethinking quantum architecture search and a benchmark. In *International conference on machine learning*, pages 22880–22898. PMLR, 2023. 1

[33] QuanGong Ma, ChaoLong Hao, XuKui Yang, LongLong Qian, Hao Zhang, NianWen Si, MinChen Xu, and Dan Qu. Continuous evolution for efficient quantum architecture search. *EPJ Quantum Technology*, 11(1):54, 2024. 2, 6, 8

[34] Darya Martyniuk, Johannes Jung, and Adrian Paschke. Quantum architecture search: a survey. In *Quantum Computing and Engineering*, pages 1695–1706. IEEE, 2024. 1

[35] Jarrod R McClean, Sergio Boixo, Vadim N Smelyanskiy, Ryan Babbush, and Hartmut Neven. Barren plateaus in quantum neural network training landscapes. *Nature communications*, 9(1):4812, 2018. 2

[36] Kosuke Mitarai, Makoto Negoro, Masahiro Kitagawa, and Keisuke Fujii. Quantum circuit learning. *Physical Review A*, 98(3):032309, 2018. 2

[37] AAM Muzahid, Hua Han, Yujin Zhang, Dawei Li, Yuhe Zhang, Junaid Jamshid, and Ferdous Sohel. Deep learning for 3d object recognition: A survey. *Neurocomputing*, page 128436, 2024. 2

[38] Michael A Nielsen and Isaac L Chuang. *Quantum computation and quantum information*. Cambridge university press, 2010. 4

[39] Mateusz Ostaszewski, Lea M Trenkwalder, Wojciech Masarczyk, Eleanor Scerri, and Vedran Dunjko. Reinforcement learning for optimization of variational quantum circuit architectures. *Advances in Neural Information Processing Systems*, 34:18182–18194, 2021. 1, 2

[40] Adrián Pérez-Salinas, Alba Cervera-Lierta, Elies Gil-Fuster, and José I Latorre. Data re-uploading for a universal quantum classifier. *Quantum*, 4:226, 2020. 3

[41] Sushmita Sarker, Prithul Sarker, Gunner Stone, Ryan Gorman, Alireza Tavakkoli, George Bebis, and Javad Sattarvand. A comprehensive overview of deep learning techniques for 3d point cloud classification and semantic segmentation. *Machine Vision and Applications*, 35(4):67, 2024. 2

[42] Maria Schuld and Francesco Petruccione. Supervised learning with quantum computers. *Quantum science and technology*, 2018. 3

[43] Maria Schuld, Ryan Sweke, and Johannes Jakob Meyer. Effect of data encoding on the expressive power of variational quantum-machine-learning models. *Physical Review A*, 103 (3):032430, 2021. 2

[44] Arsenii Senokosov, Alexandr Sedykh, Asel Sagingalieva, Basil Kyriacou, and Alexey Melnikov. Quantum machine learning for image classification. *Machine Learning: Science and Technology*, 5(1):015040, 2024. 3

[45] Sukin Sim, Peter D Johnson, and Alán Aspuru-Guzik. Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms. *Advanced Quantum Technologies*, 2(12):1900070, 2019. 5

[46] Andrea Skolik, Jarrod R McClean, Masoud Mohseni, Patrick Van Der Smagt, and Martin Leib. Layerwise learning for quantum neural networks. *Quantum Machine Intelligence*, 3:1–11, 2021. 1, 2

[47] Hanrui Wang, Yongshan Ding, Jiaqi Gu, Yujun Lin, David Z Pan, Frederic T Chong, and Song Han. Quantumnas: Noise-adaptive search for robust quantum circuits. In *Symposium on High-Performance Computer Architecture (HPCA)*, pages 692–708, 2022. 1, 2

[48] Yabo Wang, Bo Qi, Chris Ferrie, and Daoyi Dong. Trainability enhancement of parameterized quantum circuits via reduced-domain parameter initialization. *Physical Review Applied*, 22(5):054005, 2024. 2

[49] Colin White, Sam Nolen, and Yash Savani. Exploring the loss landscape in neural architecture search. In *Uncertainty in Artificial Intelligence*, pages 654–664. PMLR, 2021. 2, 3

[50] Colin P Williams and Colin P Williams. Quantum gates. *Explorations in quantum computing*, pages 51–122, 2011. 4

[51] Wenjie Wu, Ge Yan, Xudong Lu, Kaisen Pan, and Junchi Yan. Quantumdarts: differentiable quantum architecture search for variational quantum algorithms. In *International Conference on Machine Learning*, pages 37745–37764. PMLR, 2023. 1, 2

[52] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *computer vision and pattern recognition*, pages 1912–1920, 2015. 2, 6

[53] Esther Ye and Samuel Yen-Chi Chen. Quantum architecture search via continual reinforcement learning. *arXiv preprint arXiv:2112.05779*, 2021. 2

[54] Kaicheng Yu, Christian Sciuto, Martin Jaggi, Claudiu Musat, and Mathieu Salzmann. Evaluating the search phase of neural architecture search. In *International Conference on Learning Representations*. OpenReview.net, 2020. 5

[55] Anqi Zhang and Shengmei Zhao. Evolutionary-based searching method for quantum circuit architecture. *Quantum Information Processing*, 22(7):283, 2023. 2

[56] Huang Zhang, Changshuo Wang, Shengwei Tian, Baoli Lu, Liping Zhang, Xin Ning, and Xiao Bai. Deep learning-based 3d point cloud classification: A systematic survey and outlook. *Displays*, 79:102456, 2023. 2

[57] Shi-Xin Zhang, Chang-Yu Hsieh, Shengyu Zhang, and Hong Yao. Differentiable quantum architecture search. *Quantum Science and Technology*, 7(4):045023, 2022. 2

[58] Yuge Zhang, Zejun Lin, Junyang Jiang, Quanlu Zhang, Yujing Wang, Hui Xue, Chen Zhang, and Yaming Yang. Deeper insights into weight sharing in neural architecture search. *arXiv preprint arXiv:2001.01431*, 2020. 5

[59] Weiwei Zhu, Jiangtao Pi, and Qiuyuan Peng. A brief survey of quantum architecture search. In *International conference on algorithms, computing and systems*, pages 1–5, 2022. 1