Kashif and Al-Kuwari *EPJ Quantum Technology* (2024) 11:4
https://doi.org/10.1140/epjqt/s40507-023-00216-8

EPJ Quantum Technology
a SpringerOpen Journal

RESEARCH Open Access

ResQNets: a residual approach for mitigating barren plateaus in quantum neural networks

Muhammad Kashif[1*] and Saif Al-Kuwari[1]

*Correspondence:
mkashif@hbku.edu.qa
[1]Division of Information and Computing Technology, College of Science and Engineering, Hamad Bin Khalifa University, Qatar Foundation, Doha, Qatar

Abstract

The barren plateau problem in quantum neural networks (QNNs) is a significant challenge that hinders the practical success of QNNs. In this paper, we introduce residual quantum neural networks (ResQNets) as a solution to address this problem. ResQNets are inspired by classical residual neural networks and involve splitting the conventional QNN architecture into multiple quantum nodes, each containing its own parameterized quantum circuit, and introducing residual connections between these nodes. Our study demonstrates the efficacy of ResQNets by comparing their performance with that of conventional QNNs and plain quantum neural networks through multiple training experiments and analyzing the cost function landscapes. Our results show that the incorporation of residual connections results in improved training performance. Therefore, we conclude that ResQNets offer a promising solution to overcome the barren plateau problem in QNNs and provide a potential direction for future research in the field of quantum machine learning.

Keywords: Quantum neural networks; Barren plateaus; Parameterized quantum circuits; Residual learning

1 Introduction

The Noisy Intermediate-Scale Quantum (NISQ) devices are a new generation of quantum computers capable of executing quantum algorithms. However, NISQ devices still suffer from significant errors and limitations in terms of the number of qubits and coherence time [1]. Despite these limitations, NISQ devices are an important stepping stone towards the development of fault-tolerant quantum computers, as they provide a platform for exploring and evaluating basic quantum algorithms and applications [2, 3]. Research in the NISQ era is focused on developing algorithms and techniques that are resilient to noise and errors, and can run effectively on NISQ devices [4]. This includes algorithms for quantum error correction [5], quantum optimization [6], and quantum machine learning (QML) [7].

QML is an interdisciplinary field that combines the concepts and techniques from quantum computing and machine learning (ML). It aims to leverage the unique properties of quantum systems, such as superposition, entanglement, and interference, to develop new algorithms and approaches for solving complex machine learning problems [8, 9].

© The Author(s) 2024. **Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

QML is increasingly becoming an exciting application in the NISQ era [2]. The anticipation here is that quantum models (by exploiting the exponentially large Hilbert space) would achieve a computational advantage over their classical counterparts [10, 11], particularly for quantum datasets [12–14]. As we continue to witness advances in quantum hardware [15], quantum algorithms [16], and quantum error correction [17], the future of QML seems bright, and quantum computing will likely play a significant role in the field of machine learning. A wide range of ML algorithms is being explored in the quantum realm, including quantum neural networks (QNNs) [18], quantum support vector machines [19, 20], quantum principal component analysis [21], and quantum reinforcement learning [22–24]. These approaches have been shown to be effective in various domains, such as image classification [12, 25–29], natural language processing [30–32], and recommendation systems [33].

QNNs are a promising area of research that aims to combine the power of quantum computing and neural networks to solve complex computational problems [34, 35]. Unlike classical neural networks, QNNs use quantum-inspired representations and operations to encode and process data [36–38]. This allows for the exploration of exponential solution space and the exploitation of quantum parallelism, potentially leading to faster and more accurate results [8, 14, 39, 40]. QNNs can be considered as a subclass of variational quantum algorithms, which aim to optimize parameters (θ) of a parameterized quantum circuit (PQC)[1] $U(\theta)$ to minimize the cost function $\mathcal{C}$. PQC utilizes tunable parameters to optimize quantum algorithms through classical computation. An example of a QNN architecture is the quantum Boltzmann machine [41, 42], which uses quantum circuits to model complex probability distributions and perform unsupervised learning tasks. In addition to unsupervised learning, QNNs have shown potential in various applications such as quantum feature detection [20], quantum data compression and denoising [43, 44], and quantum reinforcement learning [45, 46]. QNNs can also be used for quantum-enhanced image recognition [7, 47] and quantum molecular simulations [48].

However, despite their potential, QNNs are still in the early stages of development and face several technical and practical challenges. In particular, training and optimizing the parameters in QNNs pose significant challenges. To address these challenges, the research community has been developing quantum landscape theory [49] that explores the properties of cost function landscapes in QML systems. Consequently, interesting results have been obtained in the study of QNN's training landscapes, including the occurrence of barren plateaus (BP) [50], the presence of sub-optimal local minima [51], and the impact of noise on cost function landscapes [52–55]. These findings provide important insights into the properties of QNNs and their training dynamics, and can inform the development of new algorithms and strategies for training and optimizing QNNs.

In particular, the BP problem refers to a phenomenon in which the circuit's expressiveness, as measured by its ability to approximate a target unitary operation, is severely limited as the number of qubits in the circuit increases [50], which is mainly due to vanishing gradients in the parameter space. The phenomenon of BP in QNNs is a significant challenge that impedes the advancement and widespread implementation of QNNs. To mitigate the BP, various strategies have been proposed, including the use of clever parameter initialization techniques [56], pre-training [57], examination of the dependence on the

[1] we will use the terms "PQC" and "quantum layers" interchangeably.

cost function [58, 59], implementation of layer-wise training of QNNs [60], and initialization parameters drawn from the beta distribution [61]. The trainability vs expressibility analysis of QNNs from the aspect of BP is conducted in [62], where a trade-off between quantum layers width and depth has been observed for a better learning performance. These solutions aim to overcome the limitations posed by the BP in QNNs and facilitate the full realization of their potential. However, it is important to note that the solution that works best for one QNN architecture may not work for another, as the BP problem can be highly dependent on the specific problem being solved and the quantum architecture being used.

Related work In recent research efforts, the concept of utilizing the residual approach in QNNs has gained traction. One such work, proposed in [63], introduces a hybrid quantum-classical neural network with deep residual learning. The authors explore the integration of the residual block structure into QNN architecture and highlight its potential benefits. Specifically, they emphasize that connecting residual blocks with QNNs can enhance robustness against noise, a crucial consideration in quantum computing applications.

In contrast, our research focuses on a different aspect of the residual approach in QNNs. We aim to transform the traditional QNN architecture into a residual structure by dividing the conventional QNN architecture into multiple quantum nodes (each containing its own PQC), and investigate its effectiveness in mitigating the BP phenomenon. BP are a challenge that arises as the number of qubits in quantum circuit increases, leading to deep quantum circuits. Our primary objective is to address this issue and improve the training performance of QNNs.

Furthermore, a related study in [64] employs the residual approach in the optimization of an IoT platform. The authors also conclude that the residual approach in QNNs exhibits greater robustness against noisy data and better performance in learning unitary functions.

In a different context, [65] explores the residual approach in QNNs but with a focus on shallower quantum circuits rather than deep ones. The authors aim to achieve comparable performance with shallower circuits, and they suggest manipulating data encoding strategies to improve accuracy. Our work, however, entirely concentrates on quantum circuit width, i.e., the number of qubits, as a means to study and address the barren plateaus phenomenon, independent of the data encoding technique.

Overall, these research efforts collectively contribute to the growing body of knowledge on the residual approach in QNNs, highlighting its potential benefits for various quantum computing applications.

Contribution In this paper, we propose a novel solution to mitigate the issue of barren plateaus (BP) in quantum neural networks (QNNs). Our approach is based on the concept of residual neural networks, which were previously introduced as a means of overcoming the problem of vanishing gradients in classical neural networks. In this context, we introduce the concept of residual quantum neural networks (ResQNets) by incorporating residual connections between two quantum layers of variable depths. Our findings suggest that the utilization of ResQNets substantially enhances the training process of QNNs as compared to their non-residual counterparts, denoted as PlainQNets. To substantiate

the efficacy of our proposed ResQNets, we undertake a systematic comparison involving an analysis of the cost function landscapes and an assessment of their training performance with that of PlainQNets. The results obtained from our experimental investigations elucidate that the incorporation of residual connections in QNNs (ResQNets) effectively mitigate the adverse effects of BP and result in improved overall training performance.

Organization The rest of the paper is organized as follows: Sect. 2 provides an overview of classical and quantum residual neural networks and motivates their application. Section 3 discusses parameterized quantum circuits and elaborates on how multiple PQCs can be cascaded. This section also introduces the residual approach in cascaded PQCs. The methodology we adopt in this paper while conducting the various experiments is provided in Sect. 4. Section 5 presents the results we obtained on both the simulation environment and real quantum devices. Finally, the paper concludes in Sect. 6 with a few concluding remarks and pointers to possible extensions to this work.

2 Residual neural networks

Residual Neural Networks (ResNets) are a type of deep neural network architecture that aims to improve the training process by addressing the problem of vanishing gradients. The basic idea behind ResNets is to introduce residual connections between layers in the network, allowing easier optimization as the network gets deepens. The residual connections allow the network to learn residual mapping rather than trying to fit the target function directly. This helps prevent the vanishing gradient problem, where the gradients in the backpropagation process become very small, making it difficult to update the parameters effectively. ResNets were first introduced in [66], where the authors showed that ResNets outperformed traditional deep neural networks on benchmark image recognition tasks and demonstrated that ResNets could accommodate significantly deeper architectures than previous networks without sacrificing accuracy.

Residual connections in ResNets have been shown to be effective in training very deep neural networks, with hundreds or even thousands of layers. This has drastically improved the performance in several computer vision and natural language processing tasks. A typical structure of a residual block is depicted in Fig. 1a.

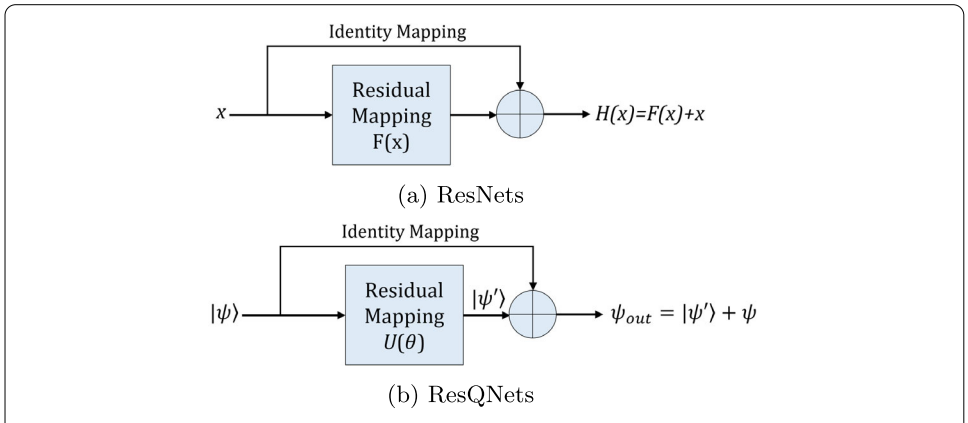

Figure 1 Residual block structure

Given an input feature map x, the basic building block of a ResNet can be defined as:

$$H(x) = F(x, W_i) + x,$$

where $H(x)$ is the output of the block, F is a non-linear function represented by a series of neuron and activation layers with parameters W_i, and x is the input feature map that is added back to the output (the residual connection). The model is trained to learn the function F such that it approximates the residual mapping $y - x$, where y is the desired output. By introducing residual connections, ResNets can address the vanishing gradient problem in deep neural networks, allowing for deeper architectures to be trained effectively.

In this paper, we introduce the quantum counterpart of ResQNet, namely residual quantum neural network (ResQNet), a QNN architecture combining the principles of classical ResNets with QNNs. The basic idea is to add a residual connection between the output of one layer of quantum operations and the input of the next layer. This helps to mitigate the vanishing gradient problem, a.k.a. BP, which is a major challenge in QNNs and arises as the number of qubits in the systems increases. Figure 1b directs how ResQNets is compared to ResNets.

In ResQNets, the residual connection is mathematically represented as:

$$\psi_{\text{out}}(\theta) = \psi(\theta) + U(\theta)\psi(\theta),$$

where $\psi(\theta)$ is the input to the quantum circuit, $U(\theta)$ is the unitary operation defined by the PQC, and $\psi_{\text{out}}(\theta)$ is the output.

3 Parameterized quantum circuits

QNN is a type of Parameterized Quantum Circuit (PQC), which is a quantum circuit that has tunable parameters that can be optimized to perform specific tasks. In a QNN, the parameters are typically optimized using classical optimization algorithms to learn a target function or perform a specific task. The PQC architecture of a QNN allows for the representation and manipulation of quantum data in a manner that can be used for various applications, such as QML and quantum control. The mathematical derivation of PQC involves the representation of quantum states and gates as matrices and the composition of these matrices to form the overall unitary operator for the circuit.

A quantum state can be represented by a column vector in a Hilbert space, where the elements of the vector are complex numbers that satisfy the normalization constraint:

$$|\psi\rangle = \begin{bmatrix} \alpha & \beta \end{bmatrix}, \quad |\alpha|^2 + |\beta|^2 = 1.$$

A quantum gate is represented by a unitary matrix, which preserves the norm of the vector, i.e., the inner product of the transformed vector with itself is equal to the inner product of the original vector with itself:

$$U^\dagger U = UU^\dagger = I,$$

where $U^\dagger$ is the conjugate transpose of U and I is the identity matrix. A PQC can be modeled as a sequence of gates, each represented by a unitary matrix based on classical

parameters. The overall unitary operator of the circuit can be obtained by composing the matrices of the individual gates in the correct order:

$$U_{\text{circuit}} = U_n(\theta_n) \cdots U_2(\theta_2) U_1(\theta_1),$$

where $U_i(\theta_i)$ is the unitary matrix representing the i-th gate and θ_i is a classical parameter.

The final quantum state after applying the PQC to an initial state can be obtained by matrix-vector multiplication:

$$|\psi_{\text{final}}\rangle = U_{\text{circuit}} |\psi_{\text{initial}}\rangle.$$

The parameters $\theta_1, \ldots, \theta_n$ can be optimized using classical optimization algorithms to achieve a desired quantum state or to maximize an objective function such as the expected value of a measurement outcome. The optimization problem can be written as:

$$\theta^* = \arg\max_{\theta} \left| \langle \psi_{\text{desired}} | U_{\text{circuit}}(\theta) | \psi_{\text{initial}} \rangle \right|^2.$$

Solving this optimization problem provides the optimal set of parameters θ^* that produce the desired outcome.

3.1 Cascading PQCs

In the proposed ResQNets, we encapsulate PQC/QNNs into a quantum node (QN) and arrange multiple QNs in a series, so that the output of one QN serves as the input of the next. This structure enables us to introduce the residual learning approach in a manner that allows the PQCs to work together to achieve the desired outcome. The process of cascading PQCs involves feeding the output of each PQC into the input of the next, creating a layered structure where each layer represents a single PQC. In this case, each PQC can build on the outputs of the previous ones, leading to a more complex and sophisticated computation. The residual learning approach is used to ensure that the overall computation remains stable, where the output of each PQC is combined with the input of the next in a specified manner.

We now present the mathematical formulation for connecting multiple PQCs in sequence. We will refer to each PQC as U_i where i denotes the QN it is encapsulated in.

3.1.1 2-cascaded PQC

Consider two PQCs denoted as $U_1(\theta_1)$ and $U_2(\theta_2)$, where θ_1 and θ_2 are classical parameters. The first PQC $U_1(\theta_1)$ is applied to an initial quantum state $|\psi_{\text{initial}}\rangle$ to obtain an intermediate quantum state $|\psi_{\text{intermediate}}\rangle$:

$$|\psi_{\text{intermediate}}\rangle = U_1(\theta_1) |\psi_{\text{initial}}\rangle.$$

The second PQC $U_2(\theta_2)$ is then applied to the intermediate state $|\psi_{\text{intermediate}}\rangle$ to obtain the final quantum state $|\psi_{\text{final}}\rangle$:

$$|\psi_{\text{final}}\rangle = U_2(\theta_2) |\psi_{\text{intermediate}}\rangle.$$

The overall unitary operator of the two cascaded PQCs can be obtained by composing the matrices of the individual PQCs in the correct order:

$$U_{\text{circuit}} = U_2(\theta_2)U_1(\theta_1).$$

The final quantum state after applying the two cascaded PQCs to an initial state can be obtained by matrix-vector multiplication:

$$|\psi_{\text{final}}\rangle = U_{\text{circuit}}|\psi_{\text{initial}}\rangle.$$

The parameters θ_1 and θ_2 can be optimized using classical optimization algorithms to achieve a desired quantum state or to maximize an objective function such as the expected value of a measurement outcome. The optimization problem can be written as:

$$\begin{aligned}\theta_1, \theta_2 &= \arg\max_{\theta_1,\theta_2} \left|\langle\psi_{\text{desired}}|U_{\text{circuit}}(\theta_1,\theta_2)|\psi_{\text{initial}}\rangle\right|^2 \\ &= \arg\max_{\theta_1,\theta_2} \left|\langle\psi_{\text{desired}}|U_2(\theta_2)U_1(\theta_1)|\psi_{\text{initial}}\rangle\right|^2.\end{aligned}$$

Solving this optimization problem returns the optimal set of parameters (θ_1, θ_2) that produce the desired outcome.

3.1.2 n-cascaded PQCs

Similarly, for n cascaded PQCs, where each PQC takes the output of the previous one as its input, the intermediate states can be described as follows:

$$|\psi_{\text{intermediate},i}\rangle = U_i(\theta_i)|\psi_{\text{intermediate},i-1}\rangle,$$

where $i = 1, 2, \ldots, n$ and $|\psi_{\text{intermediate},0}\rangle = |\psi_{\text{initial}}\rangle$. The overall unitary operator of the n cascaded PQCs can be obtained by composing the matrices of the individual PQCs in the correct order:

$$U_{\text{circuit}} = U_n(\theta_n)\cdots U_2(\theta_2)U_1(\theta_1).$$

The final quantum state after applying the n cascaded PQCs to an initial state can be obtained by matrix-vector multiplication:

$$|\psi_{\text{final}}\rangle = U_{\text{circuit}}|\psi_{\text{initial}}\rangle.$$

The parameters $\theta_1, \theta_2, \ldots, \theta_n$ can be optimized using classical optimization algorithms to achieve a desired quantum state or to maximize an objective function such as the expected value of a measurement outcome. The optimization problem can be written as:

$$\begin{aligned}\theta_1, \theta_2, \ldots, \theta_n &= \arg\max_{\theta_1,\theta_2,\ldots,\theta_n} \left|\langle\psi_{\text{desired}}|U_{\text{circuit}}(\theta_1,\theta_2,\ldots,\theta_n)|\psi_{\text{initial}}\rangle\right|^2 \\ &= \arg\max_{\theta_1,\theta_2,\ldots,\theta_n} \left|\langle\psi_{\text{desired}}|U_n(\theta_n)\cdots U_2(\theta_2)U_1(\theta_1)|\psi_{\text{initial}}\rangle\right|^2.\end{aligned}$$

Solving this optimization problem returns the optimal set of parameters $(\theta_1, \theta_2, \ldots, \theta_n)$ that produce the desired outcome.

3.2 Residual PQCs

We now introduce residual blocks in the cascaded PQCs encapsulated in QNs which we call ResQNets. In ResQNets, the output of the previous PQC is added to its input and fed as an input to the next PQC. The residual block is inserted to facilitate efficient information flow and improved performance. The primary objective of incorporating residual blocks in QNNs here is to overcome the difficulties associated with BP and thereby improve the learning process. Furthermore, the proposed method aims to harness the strengths of both residual learning and quantum computing to tackle complex problems more effectively.

To mathematically formulate our proposed ResQNets, we start by considering the case of two PQCs, and extend the approach to the general case of cascading n PQCs with n residual blocks. We will refer to each PQC as U_i where i denotes the QN it is encapsulated in.

3.2.1 1-*residual block*

ResQNet with a single residual block contains a maximum of two PQCs of arbitrary depth enclosed in two separate QNs. The first QN serves as a residual block whose input is added to its output before passing it as input to the PQC in the next QN. In the context of the NISQ era, hybrid QNNs have gained considerable traction. These models exhibit a distinctive architecture wherein the input data, characterized by its classical nature, necessitates an initial encoding process. This encoding procedure plays a vital role in preparing the data for processing on quantum computer. It is important to note that in this paper we exclusively employ a configuration wherein classical datasets are not utilized. Instead, our approach involves the initialization of qubits in ground states and the gates in PQC are randomly parameterized prior to the training phase. Nevertheless, here we present a comprehensive mathematical framework that accommodates the broader context of hybrid systems. This formalism is especially pertinent in scenarios where classical datasets form an integral component of the computational process. An illustrative configuration featuring a pair of Quantum Nodes (QNs), where the initial node functions as the residual block, is depicted in Fig. 2.

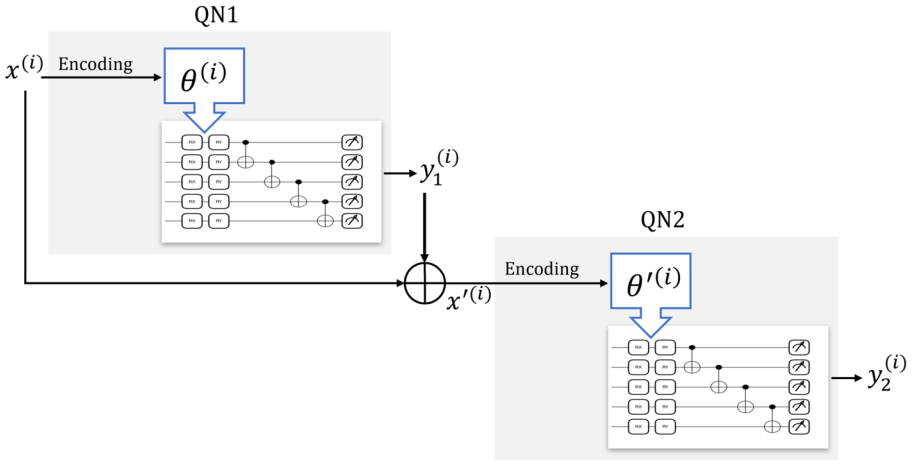

Figure 2 Illustration of residual approach in hybrid quantum neural networks

The two QNs will have two PQCs denoted as $U_1(\theta_1)$ and $U_2(\theta_2)$, where θ_1 and θ_2 are classical parameters encoded in such a way that the quantum circuit can process them. The classical dataset is a set of data points, i.e., $\mathcal{D} = \{x^{(i)}\}$, where $x^{(i)}$ represents the ith datapoint. The next step is to encode the classical data Each data point $x^{(i)}$, is encoded using an encoding method (e.g., angle or amplitude encoding [67]). The angle-encoded data for the ith data point can be denoted as $\theta^{(i)}$. Each encoded data point $\theta^{(i)}$ is used as parameters for the PQC in the first quantum node (QN1). This PQC processes the encoded data and upon measurement, it generates a classical result $y_1^{(i)}$. The classical result $y_1^{(i)}$ from QN1 is added to the original data $x^{(i)}$ element-wise to obtain a new modified classical dataset denoted by $\mathcal{D}' = \{x^{(i)} + y_1^{(i)}\}$. Each data point in $\mathcal{D}'$ is then encoded again to obtain a new set of encoded data denoted by $\theta'^{(i)}$ which is then used as input for a PQC in the second quantum node (QN2). The PQC in second QN processes this encoded data and upon measurement it generates classical result $y_2^{(i)}$ which, in case of two QNs, is the final output of the network used for cost function optimization.

The mathematical formulation for a single residual block in two QN setting starts with preparing and initializing the qubits. We initialize the qubits in ground state:

$$|\psi_{\text{initial}}^{(1)}\rangle = |0\rangle^{\otimes n}$$

where n is the number of qubits and the superscript denotes the PQC number number, i.e., 1 here denotes the qubit initialization in the first PQC. After the qubit initialization, the next step is to encode the classical data features into quantum space:

$$\theta^{(i)} = f_{\text{encode}}\left(x^{(i)}\right),$$

where f_{encode} is the encoding function which maps the classical input features to quantum space. The first PQC $U_1(\theta_1)$ is applied to the encoded input features to obtain an intermediate quantum state $|\psi_{\text{intermediate}}\rangle$:

$$|\psi_{\text{intermediate}}\rangle = U_1(\theta^{(i)})|\psi_{\text{initial}}^{(1)}\rangle.$$

Upon measurement the intermediate quantum state $|\psi_{\text{intermediate}}\rangle$ collapses and returns the classical result:

$$y_1 = \mathcal{M}|\psi_{\text{intermediate}}\rangle,$$

where $\mathcal{M}$ denotes the qubit measurement. The qubits in second PQC are also prepared in ground state:

$$|\psi_{\text{initial}}^{(2)}\rangle = |0\rangle^{\otimes n}$$

where the superscript (2) denotes the PQC number. Now, the input of the second PQC $U_2(\theta_2)$ is not just the output of QN1 but the sum of the original input ($x^{(i)}$) and the intermediate result (y_1).

$$x'^{(i)} = x^{(i)} + y_1^{(i)}.$$

We again have to encode the new data points obtained after addition of original input and intermediate result before passing it to the PQC in QN2.

$$\theta'^{(i)} = f_{\text{encode}}\left(x'^{(i)}\right).$$

The final quantum state obtained after the second QN would be:

$$|\psi_{\text{final}}\rangle = U_2(\theta'^{(i)})|\psi^{(2)}_{\text{initial}}\rangle.$$

After measuring the qubits in second QN, the final quantum state collapses and we get the final classical result:

$$y_2 = \mathcal{M}|\psi_{\text{final}}\rangle.$$

The parameters θ can be optimized using classical optimization algorithms to achieve a desired quantum state or to maximize an objective function such as the expected value of a measurement outcome.

3.2.2 2-residual blocks

In ResQNets with two residual blocks, up to three PQCs can be incorporated within three QNs. There are three potential configurations for the residual blocks in this setup:

1. utilizing only the first QN as a residual block,
2. combining the first two QNs to form a single residual block,
3. utilizing both the first and second QNs individually as separate residual blocks.

For our mathematical formulation, only the third configuration will be considered since it is the general setting for the case of two residual blocks; other configurations effectively contain a single residual block, which has already been mathematically derived in Sect. 3.2.1. However, we will conduct experiments that examine all three configurations to determine which configuration performs the best. Let $U_1(\theta_1)$, $U_2(\theta_2)$, and $U_3(\theta_3)$ be PQCs enclosed in three QNs, where θ_1, θ_2, and θ_3 are the quantum-encoded classical parameters. The qubits in the first PQC are intialized in ground state:

$$|\psi^{(1)}_{\text{initial}}\rangle = |0\rangle^{\otimes n}$$

The intial classical input features are encoded into qubit rotation angles:

$$\theta^{(i)}_1 = f_{\text{encode}}\left(x^{(i)}\right).$$

The first PQC $U_1(\theta_1)$ takes the initial quantum state $|\psi_{\text{initial}}\rangle$ as its input and produces an intermediate quantum state $|\psi_{\text{intermediate}}\rangle$:

$$|\psi_{\text{intermediate}}\rangle = U_1(\theta^{(i)}_1)|\psi^{(1)}_{\text{initial}}\rangle.$$

The $|\psi_{\text{intermediate}}\rangle$ is the output of $U_1(\theta_1)$ before the measurement. Upon measuring the qubits the quantum state $|\psi_{\text{intermediate}}\rangle$ collapses and produces a classical result:

$$y_1 = \mathcal{M}|\psi_{\text{intermediate}}\rangle.$$

Before passing the output of QN1 (y_1) as input to QN2, it is first added element-wise with the original input $x^{(i)}$. Since, both y_1 and $x^{(i)}$ are classical values, therefore an encoding function is again applied in order for the PQC in QN2 to process them:

$$x'^{(i)} = x^{(i)} + y_1^{(i)},$$
$$\theta_2^{(i)} = f_{\text{encode}}\left(x'^{(i)}\right).$$

The encoded data is then passed to second PQC, yielding another intermediate quantum state $|\psi'_{\text{intermediate}}\rangle$:

$$\left|\psi'_{\text{intermediate}}\right\rangle = U_2(\theta_2^{(i)})\left|\psi^{(2)}_{\text{initial}}\right\rangle.$$

where $|\psi^{(2)}_{\text{initial}}\rangle$ denotes the ground state initialization of qubits in second PQC. The quantum state $|\psi'_{\text{intermediate}}\rangle$ is the result before measurement and upon measuring the PQC in QN2, we get the classical result:

$$y_2 = \mathcal{M}\left|\psi'_{\text{intermediate}}\right\rangle.$$

Finally, the third PQC $U_3(\theta_3)$ takes the sum of output of QN1 (which also is the input of QN2) and output of QN2 as input and produces the final quantum state $|\psi_{\text{final}}\rangle$:

$$x''^{(i)} = y_1^{(i)} + y_2^{(i)}.$$

Since $x''^{(i)}$ is again classical, so it has to encoded before passing it to the $U_3(\theta_3)$:

$$\theta_3^{(i)} = f_{\text{encode}}\left(x''^{(i)}\right).$$

The final quantum state obtained after the action of $U_3(\theta_3)$ will be:

$$|\psi_{\text{final}}\rangle = U_3(\theta_3^{(i)})\left|\psi^{(3)}_{\text{input}}\right\rangle.$$

where $\left|\psi^{(3)}_{\text{input}}\right\rangle$ denotes the ground state initialization of qubits in third PQC. The final quantum state collapses into a classical vector after measurement, which will be the final result of the network used for further optimization.

$$y_3 = \mathcal{M}|\psi_{\text{final}}\rangle.$$

The same procedure can be extended for n-residual blocks with different residual configurations.

Given a set of n PQCs, $U_1(\theta_1), U_2(\theta_2), \ldots, U_n(\theta_n)$ and an initial quantum state $|\psi_{\text{initial}}\rangle$, the objective is to find the set of parameters $\boldsymbol{\theta} = \theta_1, \theta_2, \ldots, \theta_n$ that maximizes (or minimizes) some cost function $C(\boldsymbol{\theta})$ associated with the final quantum state $|\psi_{\text{final}}\rangle$ produced by the cascaded PQCs. The optimization problem can be formulated as:

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} C(\boldsymbol{\theta})$$

or

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} C(\boldsymbol{\theta}),$$

where $\boldsymbol{\theta}^*$ represents the optimal set of parameters that maximizes (or minimizes) the cost function. The cost function $C(\boldsymbol{\theta})$ can be defined based on the desired behavior of the quantum circuit and can be calculated from the measurement result of final quantum state $|\psi_{\text{final}}\rangle$.

4 Methodology

In classical NNs, residual neural networks (ResNets) were proposed to overcome the problem of vanishing gradients and were very useful for enabling deep learning in classical machine learning. In this paper, we propose a Residual Quantum Neural Networks (ResQNets), to enable deep learning in QNNs by mitigating the effect of BP as a function of the number of layers.

The conventional approach to constructing QNNs contains an arbitrarily deep PQC, which takes some input and yields some output. Such an architecture typically has a single QN, as depicted in Fig. 3a. In this paper, we refer to this traditional QNN architecture as "Simple PlainQNet".

To construct our proposed ResQNets, we need to further split the traditional QNN architecture into two QNs, where every QN contains arbitrary deep quantum layers. Since our proposed ResQNets contain at least two QNs and the traditional way of constructing QNNs contains a single QN, we construct a slightly modified version of simple PlainQNet, which we call "PlainQNet" and includes two or more QNs, with each QN containing PQCs of arbitrary depth, as shown in Fig. 3b. In PlainQNets, the output of the previous QN is fed to the next QN. The purpose of constructing PlainQNet is to have a fair comparison with our proposed ResQNets because ResQNets need two or more QNs to work. An example

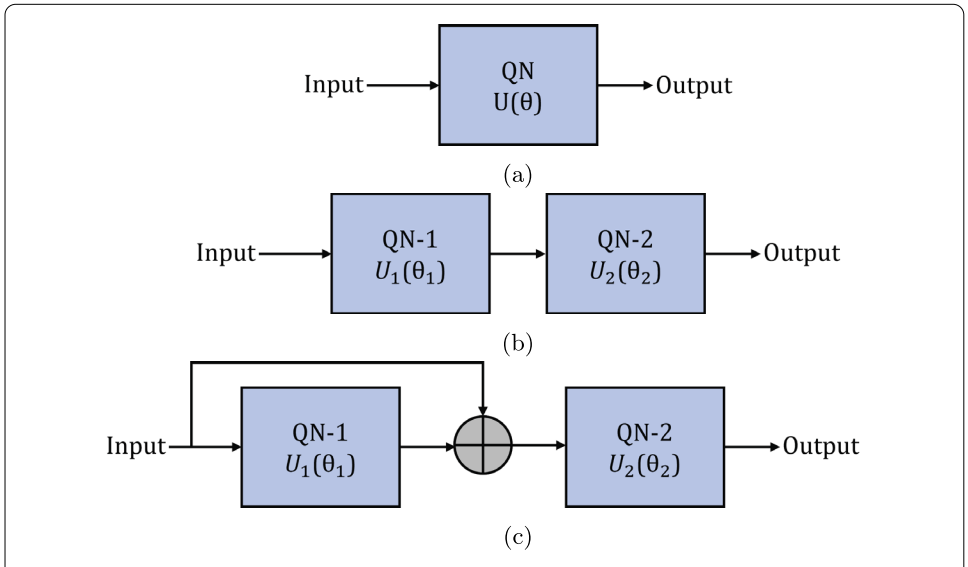

Figure 3 QNN architecture used in this paper (a) Simple PlainQNet (b) PlainQNet and (c) ResQNet. The internal architure and working of QN is shown Fig. 2

of ResQNet architecture with two QNs is shown in Fig. 3c. The PlainQNet architecture is similar to general QNN split into two QNs, whereas in the case of ResQNet, the first QN serves as the residual block, i.e., the input of the first QN is added to its output and then passed as input to the second QN.

It should be noted that ResQNets can comprise multiple QNs with various arrangements of residual blocks. For instance, the ResNet from Fig. 3c can be extended to have three QNs, in which case three potential configurations can be employed. These include having the first and second QNs acting as individual residual blocks, combining the first and second QNs to serve as a single residual block, and only the first QN functioning as the residual block. The possibility of these three configurations has been taken into consideration. We also consider the case of three QNs with these configurations.

4.1 Quantum layers design

For the design of quantum layers, we use a periodic structure containing two single-qubit unitaries (RX and RY) per qubit. These unitaries are randomly initialized in the range $[0, \pi]$. Furthermore, a two-qubit gate, i.e., $CNOT$-gate is used to entangle qubits, and every qubit is entangled with its neighboring qubit. Figure 4 shows the example design of the quantum layers we used (5 qubits). All the QNs in our experiments have the same quantum layers desgin.

4.2 Depth of quantum layers

The impact of the quantum layer depth in examining the existence of BP in the cost function landscape of a QNN is significant. Effective depth (the longest path within the quantum circuit until the measurement) is crucial in this regard. For convenience, We introduce two depth parameters: layer depth (D_L) and effective depth (D_E). The layer depth D_L refers to the combined number of repetitions of the quantum layer illustrated in Fig. 4 in both QNs, while the effective depth D_E represents the overall depth. For our quantum layers design, the following equation can be used to calculate the effective depth.

$$\textit{Total Effective Depth} = D_E = 4 \times D_L + k, \tag{1}$$

where $k = 2, 3, 4, 5 \ldots$ for $5, 6, 7, 8 \ldots$ qubits, respectively. Since the quantum layers are split into two separate QNs, and the depth per QN can be crucial to achieving better per-

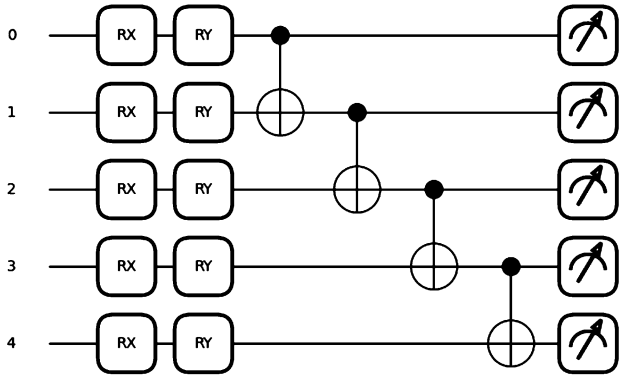

Figure 4 Quantum Layers Design

Table 1 Depth combinations per QN

D_L in QN-1	D_L in QN-2	In-text representation
1	5	(1, 5)
5	1	(5, 1)
2	4	(2, 4)
4	2	(4, 2)
3	3	(3, 3)

formance, it is important to calculate D_E of each QN individually and then add them to obtain the final D_E. Failure to calculate the depth in each QN separately could result in an effective depth different from the sum of the effective depths of each QN, i.e., $D_L/QN1 + D_L/QN2 \neq D_E$. For example, with $D_L = 2$, the total effective depth would be 10 without considering the splitting into two QNs. However, if D_L is split into two QNs with $D_L/QN = 1$, the effective depth would be 12. A modified version of Eq. (1) should be used to calculate the D_E per QN, as described below.

$$\textit{Effective Depth per } QN = D_E/QN = 4 \times D_L/QN + k. \tag{2}$$

4.3 Depth distribution per QN

As previously discussed, ResQNets and PlainQNets consist of multiple QNs, which results in different depth splits for a given depth of quantum layers. According to the definition of BP, the gradient vanishes as a function of the number of qubits; hence, we fix the depth of quantum layers to $D_L = 6$, and only vary the number of qubits. Table 1 summarizes the different depth per QN combinations for $D_L = 6$, and all these depth combinations are tested for different numbers of qubits. Column 3 of Table 1 represents the depth split in the form of ordered pairs (we refer to this form in the rest of the paper whenever we discuss depth split per QN). For instance, (1, 5) denotes $D_L = 1$ in the first QN and $D_L = 5$ in the second QN. The depth per QN combination can be extended to more than two QNs in a similar manner.

4.4 Cost function definition

For training our proposed ResQNet, we consider a simple example of learning the identity gate. In such a scenario a natural cost function would be the difference of 1 minus the probability of measuring an all-zero state, which can be described by the following equation.

$$C = \left\langle \psi(\theta) \right| \left(I - |0\rangle\langle 0| \right) \left| \psi(\theta) \right\rangle = 1 - p_{|0\rangle}.$$

We consider the global cost function setting, i.e., we measure all the qubits in the network. Therefore, the above cost function definition will be applied across all the qubits according to the following equation.

$$C = \left\langle \psi(\theta) \right| \left(I - |00\ldots0\rangle\langle 00\ldots0| \right) \left| \psi(\theta) \right\rangle = 1 - p_{|00\ldots0\rangle}. \tag{3}$$

For cost function optimization, we use Adam optimizer (with a stepsize of 0.1), which is a gradient-based optimization method for optimization problems. The Adam optimizer updates the parameters of a model iteratively based on the gradient of the loss function

with respect to the parameters. The Adam optimizer uses an exponentially decaying average of the first and second moments of the gradients to adapt the learning rate for each parameter. Let g_t be the gradient of the loss function with respect to the parameters at iteration t. The first moment, m_t, and the second moment, v_t, are computed as follows:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t,$$
$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2,$$

where β_1 and β_2 are the decay rates for the first and second moments, respectively. The bias-corrected first moment and second moment are then computed as:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t},$$
$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}.$$

Finally, the parameters are updated using the following equation:

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t,$$

where α is the learning rate and ϵ is a small constant to prevent division by zero.

5 Results and discussion

In order to investigate the issue of BP in both PlainQNets and ResQNets, we maintain a constant depth of quantum layers, $D_L = 6$, which comprises 100 quantum gates and 60 parameters. The quantum layer depth distribution is varied among different combinations, as discussed in Table 1. The D_E per QN can then be calculated using Eq. (2). The performance of both networks is evaluated by comparing their cost function landscapes and training results for the problem specified in Eq. (3).

5.1 PlainQNet and simple PlainQNet

In this paper, the construction of the proposed ResQNets involves the incorporation of a minimum of two QNs, whereas traditionally QNNs development entails the use of a single QN (referred to as "simple PlainQNets in this paper). To ensure a fair performance comparison of QNNs with no residual connections and our proposed ResQNets, we modify the architecture of simple PlainQNets by dividing it into two QNs (referred to as "PlainQNets" in this paper). This architectural modification is primarily aimed to have a similar architecture of PlainQNets (QNNs with no residual connection) and ResQNets before comparing their performance.

Given the modification introduced to the conventional QNN architecture, as stated above, it is necessary to comparatively analyze the performance of the unaltered simple PlainQNets and the adapted PlainQNets. This preliminary comparison aims to identify any potential consequences arising from the structural modification. If this architectural modification results in minimal disruptions to performance, it would establish a basis for conducting a subsequent comparative analysis between PlainQNets and ResQNets with confidence.

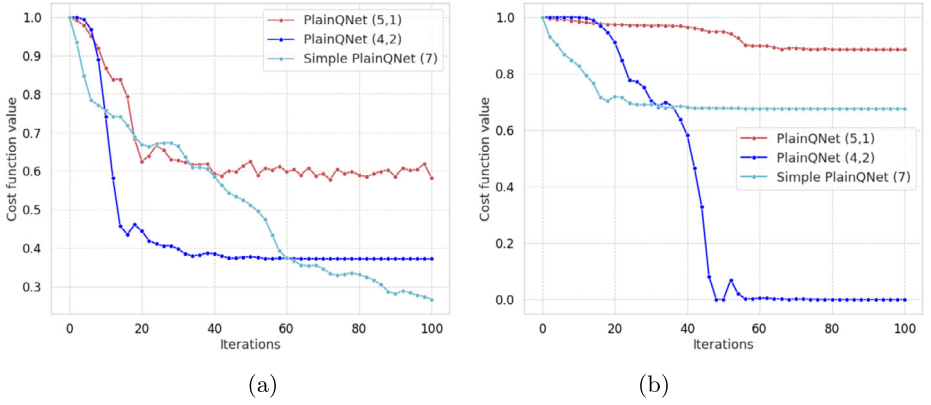

Figure 5 Cost vs. iterations of PlainQNets and simple PlainQNets (a) for 6 qubits (b) for 7 qubits. The parentheses denote the D_L per QN

The simple PlainQNets and PlainQNets are compared for 6-qubit and 7-qubit quantum layers with a constant depth of $D_L = 6$. In the case of PlainQNets, the depth distribution per QN can vary, but we use the depth combinations of $(5, 1)$ and $(4, 2)$, where the first entry represents the depth of the first QN and the second entry represents the depth of the second QN, as shown in Table 1. We choose deeper quantum layers on the first QN and relatively shallow depth on the second QN primarily because such a configuration of depths per QN leads to a better performance, which will be discussed in more detail in the subsequent sections. For 6-qubit quantum layers, the effective depth (D_E) for PlainQNets for both depth combinations mentioned above is 30 (as defined in Eq. (2)). The closest possible D_E for simple PlainQNets using the quantum layers considered in this paper (shown in Fig. 4) is 31 with an overall D_L of 7 (as defined in Eq. (1)), which was used in the comparison. Similarly, for 7-qubit quantum layers, the D_E for PlainQNets is 32 for both depth combinations per QN. The closest D_E in the case of simple PlainQNets is obtained for $D_L = 7$.

Both PlainQNets and simple PlainQNets are then trained for the problem specified in Eq. (3). The training results are displayed in Fig. 5. It can be observed that for 6-qubit layers, both PlainQNets and simple PlainQNets exhibit comparable performance. However, when the number of qubits increases to 7, the performance of simple PlainQNets decreases significantly due to BP, while PlainQNets improves. Based on these observations, we can infer that it is appropriate to compare the performance of PlainQNets with that of our proposed ResQNets. Hence, for the remainder of the paper, we will compare the performance of PlainQNets, which are QNNs containing two (or more) QNs, with that of ResQNets.

5.2 ResQNet with shallow width quantum layers

In this section, we perform a comparative analysis of the incidence of BP in both PlainQNets and ResQNets. Both PlainQNets and ResQNets consist of two QNs, with a maximum of one residual block in the case of ResQNets. To facilitate a fair comparison, we consider shallow depth quantum layers with $D_L = 6$ and incrementally vary the number of qubits from 6 to 10.

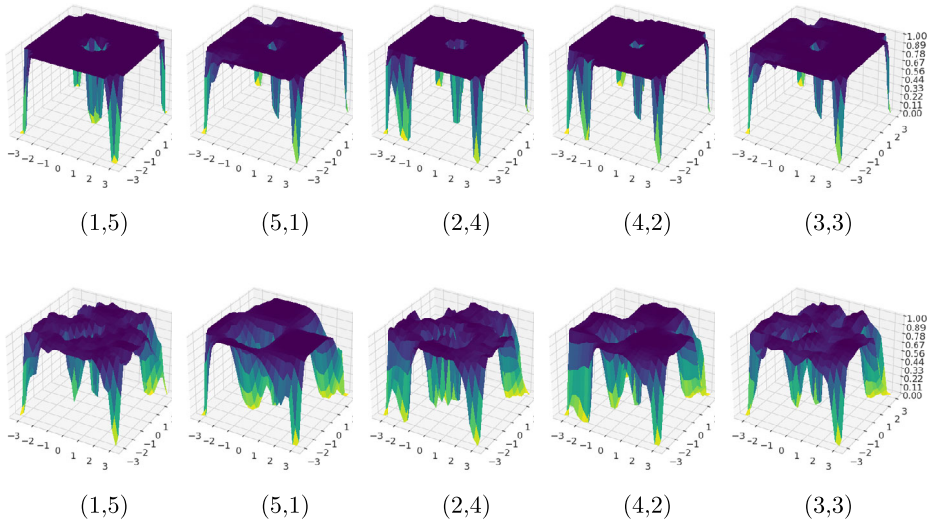

Figure 6 Cost function landscapes of PlainQNet (upper panel) and ResQNet (lower panel) for 6 Qubits. The parentheses denotes the D_L per QN

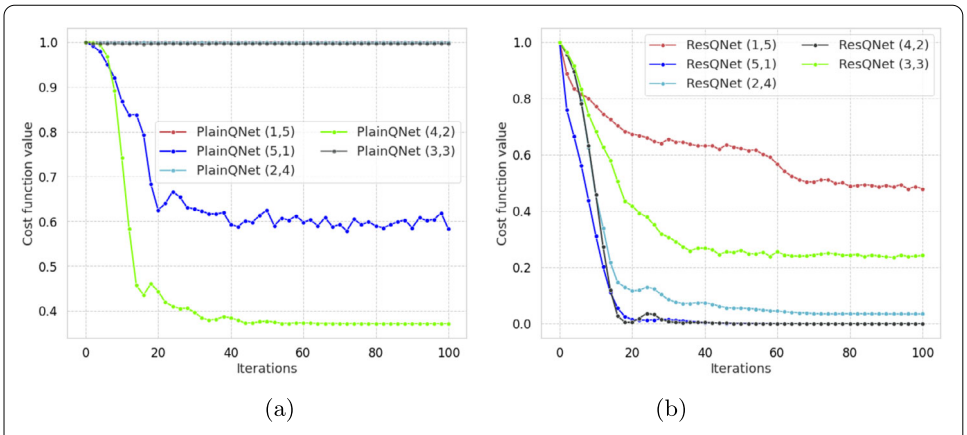

Figure 7 Cost vs. iterations of (a) PlainQNets (b) and ResQNets for 6 qubits. The parentheses denote the D_L per QN

5.2.1 6-qubit circuit

In this setting, we experiment with a total of 6 qubits. The cost function landscapes for both PlainQNet and ResQNet were analyzed and compared, as shown in Fig. 6. The results demonstrate that a significant portion of the cost function landscapes of the PlainQNet for almost all the depth combinations are flat and have a narrow region containing the global minimum. On the other hand, the cost function landscapes of ResQNets are less flat and have a wider region containing the global minimum, which makes ResQNet more suitable for optimization.

The training of PlainQNets and ResQNets was performed for the problem defined in Eq. (3). The results of the training are depicted in Fig. 7. When the depth of the second QN is equal to or greater than the depth of the first QN, it was observed that the PlainQNets do not undergo successful training. This can be attributed to the flat cost function landscape, i.e., the BP, as depicted in Fig. 6. For the similar depth distribution per QN

(depth in second QN ≥ depth in first QN), the ResQNets were observed to effectively undergo training. However, they struggled to reach an optimal solution due to the presence of multiple local minima in their cost function landscape. In instances where the depth of the first QN is greater than the second QN, both PlainQNets and ResQNets underwent successful training, but ResQNets outperformed PlainQNets.

5.2.2 8-qubit ciruit

We now conduct experiments on both PlainQNets and ResQNets with 8-qubit layers, and examine the cost function landscapes of both PlainQNets and our proposed ResQNets. The overall layer depth is set to 6, and all depth combinations are analyzed. The results presented in Fig. 8, reveal that approximately 90% of the cost function landscape for PlainQNets remains flat irrespective of the depth distribution per QN, making them unsuitable for optimization. In contrast, the cost function landscapes of ResQNets are still not flat for all the depth combinations, and thus are more favorable for optimization.

We conduct training experiments for both PlainQNets and ResQNets with 8 qubit quantum layers to solve the problem defined in Eq. (3). The training results are presented in Fig. 9, which shows that as we increase the number of qubits from 6 to 8, the PlainQNets get trapped in the flat cost function landscape (i.e., BP) for all the depth combinations per QN and fail to train effectively for the specified problem.

On the other hand, the ResQNets demonstrate successful training across all the depth combinations, surpassing the performance of PlainQNets. Notice that ResQNets exhibit superior learning outcomes when the depth of the first QN is much greater than that of the second QN (D_E in $QN1 > > > > D_E$ in $QN2$), such as in the case of (5, 1). This is because in such scenarios the cost function landscape has fewer and wider regions leading to the global minimum. Conversely, when the depth of the second QN is equal to or greater than that of the first QN, the cost function landscape is characterized by multiple local minima, making it less suitable for optimization as the optimizer becomes trapped in local minima. This phenomenon can be attributed to the presence of residual blocks in ResQNets. In the

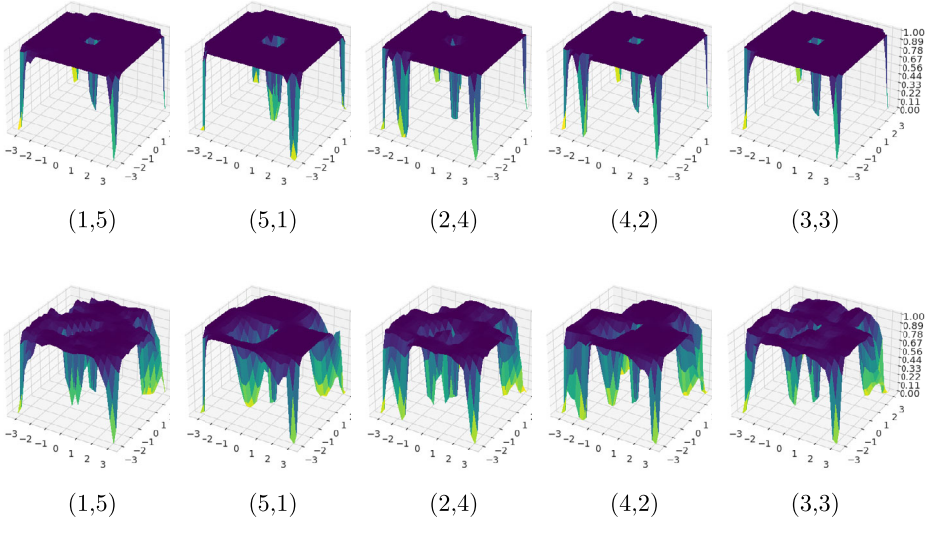

Figure 8 Cost function landscapes of PlainQNet (upper panel) and ResQNet (lower panel) for 8 Qubits. The parentheses denote the D_L per QN

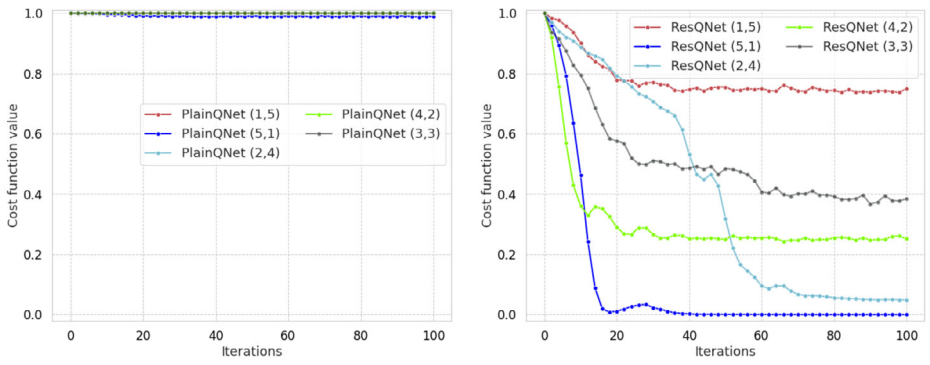

Figure 9 Cost vs. iterations of (a) PlainQNets and (b) ResQNets for 8 qubits. The parentheses denote the D_L per QN

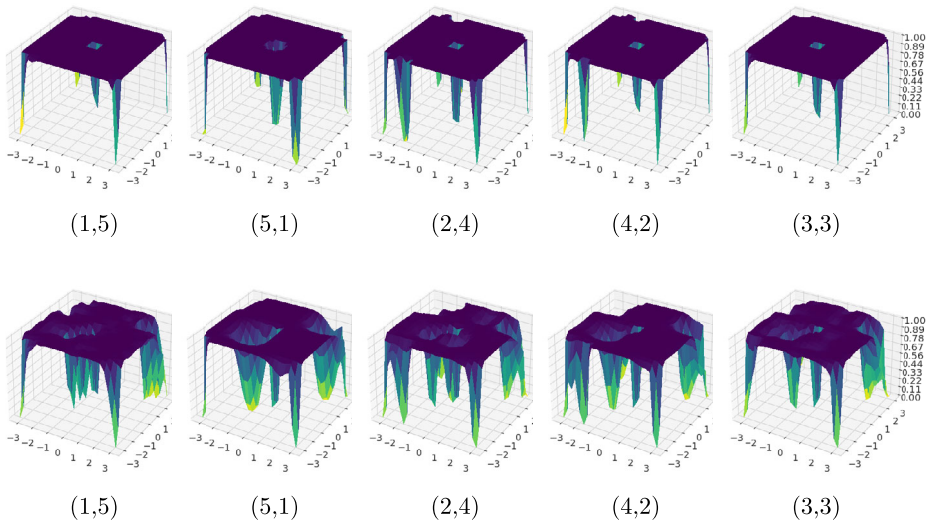

Figure 10 Cost function landscapes of PlainQNet (upper panel) and ResQNet (lower panel) for 10 Qubits. The parentheses denotes the D_L per QN

case of two QNs, a residual connection is introduced only after the first block. This helps in mitigating the issue of BP. However, if the second QN is deep enough, it can still result in BP. In such scenarios, the cost function landscape still contains multiple local minima and fewer paths to reach the global minimum, which makes the optimization process more prone to becoming stuck in a local minimum. Despite this, ResQNets still demonstrate superior training performance compared to PlainQNets.

5.2.3 10-*qubit circuit*

To expand our study further, we increased the number of qubits to 10 and performed the same experiments as with quantum layers of 6 and 8 qubits. The cost function landscapes were then analyzed for both PlainQNets and ResQNets, as shown in Fig. 10. Similar to the case of 8 qubit layers, a substantial portion of the cost function landscape of PlainQNets was found to be flat, indicating the presence of BP and making it unsuitable for optimiza-

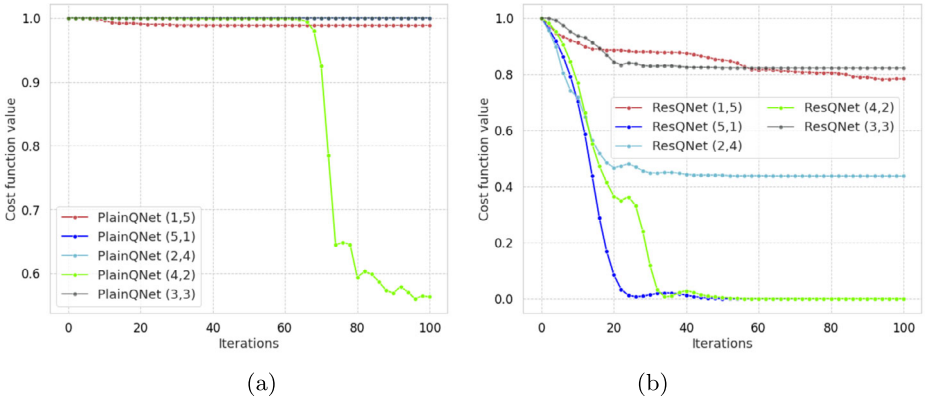

Figure 11 Cost vs. iterations of (a) PlainQNets (b) and ResQNets for 10 qubits. The parantheses denotes the D_L per QN

tion. Conversely, the cost function landscape of ResQNets remained more favorable for optimization as it was characterized by multiple paths leading to the global minimum, thus avoiding the occurrence of BP.

Subsequently we trained the 10 qubit quantum layers to address the problem defined in Eq. (3). The results of these experiments are depicted in Fig. 11. Our analysis indicates that PlainQNets did not exhibit successful training outcomes for nearly all depth combinations, with the exception of (4, 2), which showed considerable performance improvement. When we examined its cost function landscape in Fig. 10, we observed that there exist one or two narrow regions that contain the solution and may be found by the optimizer to converge to the solution. However, these narrow regions are unlikely to be encountered and thus the performance, despite being optimal, is not considered suitable for general optimization problems. Therefore, it can still be concluded that the PlainQNets are severely affected by the problem of BP. On the other hand, ResQNets effectively overcame the issue of BP and demonstrated successful training outcomes for all depth combinations. Our observations for 10 qubit quantum layers align with our previous findings for 6 and 8 qubit layers in that ResQNets are more effective when the depth after the residual connection is less. This suggests that a shallower depth of quantum layers after the residual connection in ResQNets is more favorable for optimization and mitigating the impact of BP.

Our results conclusively demonstrate that PlainQNets are heavily impacted by the issue of BP as the number of qubits increases, which significantly hinders their performance and ability to optimize the cost function. The previous results have demonstrated the advantage of our proposed ResQNets over PlainQNets in mitigating the phenomenon of BP. Therefore, in the next section, we will conduct experiments solely with ResQNets.

5.3 ResQNets with wider quantum layers

To analyze the scalability of ResQNets for larger quantum circuits, we consider quantum layers with a larger number of qubits, i.e., 15 and 20. The depth of the quantum layers, D_L, is kept constant at 6. As the cost function landscapes are known to have a direct impact on the training results, as shown in Sect. 5.2. Consequently, we only present the training results for the 15 and 20-qubit quantum layers.

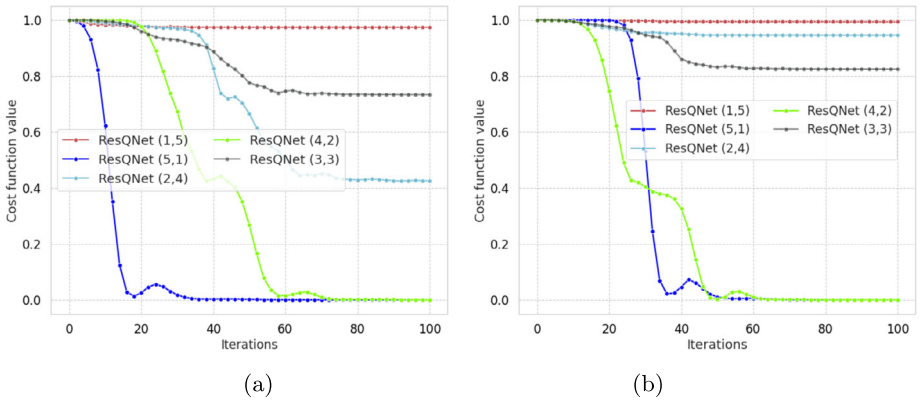

Figure 12 Cost vs. iterations of ResQNets for (a) 15 qubits and (b) 20 qubits. The parentheses denote the D_L per QN

5.3.1 15-*qubit circuit*

We train the 15 qubit quantum layers to optimize the problem defined in Eq. (3). The training results are shown in Fig. 12a. It can be observed that the ResQNets are effectively trained. Additionally, analogous to the case of shallow width quantum layers, the performance is substantially better when the depth in the first QN (before the residual point) is bigger than the second QN.

5.3.2 20-*qubit circuit*

We now train the ResQNets for 20-qubit layers for the problem defined in Eq. (3), with a total layer depth of $D_L = 6$. It can be observed that even with 20 qubit layers, the ResQNets are effectively trained, as shown in Fig. 12b. Furthermore, similar to the previously shown results, the ResQNets for 20-qubit layers also perform significantly better when the depth after the residual point (second QN) is lesser than the depth before the residual point (first QN).

From the results in Fig. 12, it is evident that the ResQNets are capable of working with wider quantum layers. The results demonstrate that analogous to the case of shallow-width quantum layers, the training performance is better with optimal results being achieved for a larger depth in the first QN and a smaller depth in the second QN.

It should be noted that our experiments are limited by the memory constraints of our local computer and we cannot go beyond 20 qubits. However, based on our findings, we believe that the proposed ResQNets would still train effectively even beyond 20 qubits.

5.4 ResQNets with 3-QN

From the analysis presented in previous sections, it can be observed that the ResQNets consisting of two QNs with a maximum of one residual block can effectively address the problem of BP and significantly improve the training performance of QNNs. In this section, we show that increasing the number of QNs in ResQNets can enhance the performance of ResQNets even further. As discussed in Sect. 4, for three QNs we can have multiple configurations of residual blocks. We consider all of these configurations for our experiments with 20-qubit quantum layers and a fixed quantum layer depth of $D_L = 6$. The results of the experiments conducted in this section will provide valuable insights into the optimal configuration of residual blocks for ResQNets with three or more QNs.

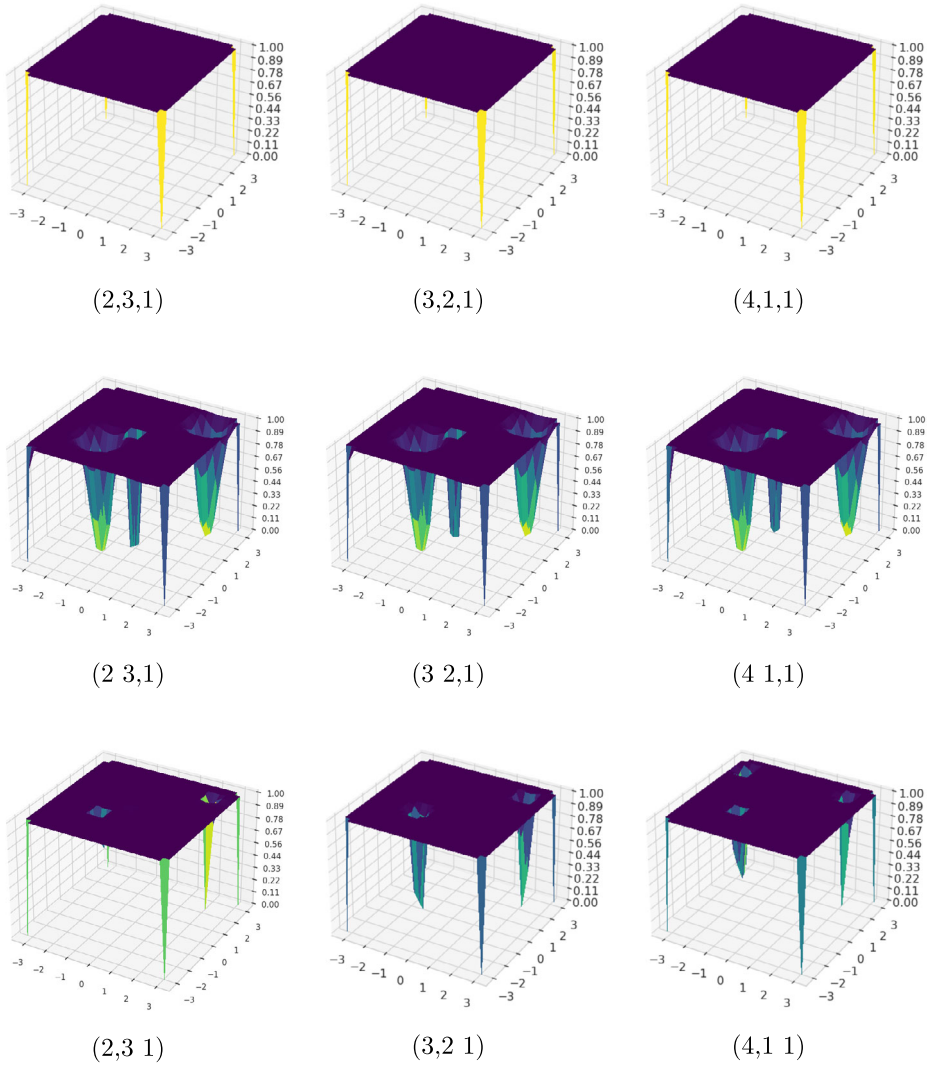

Figure 13 Cost function landscapes of ResQNets for 20 Qubits and 3-QNs. Residual after every QN (Top panel), Residual after two QNs (middle panel) and residual only after the first QN (bottom panel). The parentheses denote the D_L per QN and the comma denotes the residual point

The cost function landscapes of various residual block configurations in ResQNets with three QNs were analyzed, as presented in Fig. 13. The results indicate that the optimal placement of residual blocks has a significant impact on the performance of ResQNets. When the residual block is added after every QN, the cost function landscape quickly flattens irrespective of the depth per QN, suggesting that this configuration leads to equivalent or suboptimal performance compared to PlainQNets, which is not at all suitable for optimization.

On the other hand, when the residual block is added after two QNs, the cost function landscape shows multiple and wider regions containing the global minimum, which makes this configuration more suitable for optimization. Moreover, this configuration exhibits a consistent cost function landscape regardless of the depth per QN combination, implying

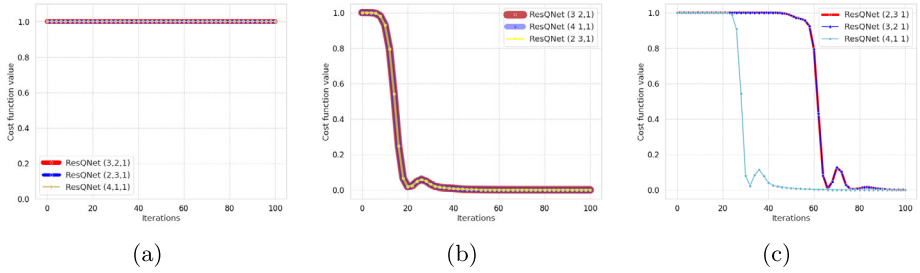

Figure 14 Training results of ResQNets with three QNs with 20 qubit layers. (a) Residual after every QN (b) Residual after two QNs and (c) Residual after the first QN. The parentheses denote the D_L per QN and the comma denotes the residual point

that this particular residual block arrangement is more robust to BP and supports a wide range of depths and QN combinations.

For the case of adding the residual only after the first QN, with two QNs after the residual block, the results show that the cost function landscape is better than the case of adding the residual block after every QN, but not as good as the case where there is a gap of two QNs while adding the residual.

We then trained ResQNets with three QNs for all the configurations while varying the depth for each QN combination on the problem defined in Eq. (3). The training results are shown in Fig. 14. These results align with the behavior of the cost function landscape, where the residual block configuration skipping two QNs outperforms other configurations. It can be observed that the residual block configuration after every QN does not train at all, while the residual block configuration after the first QN does converge for all the depth per QN combinations, but with significantly slower convergence compared to the residual block configuration after two QNs.

5.5 3-QN vs. 2-QN ResQNet

In this section, we compare the performance of ResQNets with 2 and 3-QNs to demonstrate the impact of increasing the number of QNs. The analysis was conducted for 20 qubit layers considering the best-performing depth combinations for both 2 and 3-QNs.

For 2-QNs, the results from Fig. 12b indicate that the depth combinations of (5,1) and (4,2) performed better than other depth combinations. On the other hand, for three QNs, the results from Fig. 14b and 14c show that the depth combinations of (4 1, 1) and (4, 1 1) outperformed other depth combinations. A closer examination of the best-performing depth combinations reveals that the D_L before and after the residual block for the depth per QN combination of (5, 1) in 2-QN ResQNet is equivalent to depth per QN combination of (4 1, 1) for 3-QN ResQNet. Similarly, the combination (4, 2) in the 2-QN ResQNet is equivalent to (4, 1 1) in the 3-QN ResQNet. Despite these similarities, as demonstrated in Fig. 15, the ResQNets with 3-QNs exhibit superior performance, as they converge to the optimal solution more efficiently compared to the ResQNets with 2-QNs.

5.6 Real quantum device

The results presented so far were obtained by running ResQNets and PlainQNets on a simulation platform. In this section, we carry out some experiments on real quantum devices. In particular, we trained both ResQNets and PlainQNets with 2-QNs on a 5-qubit

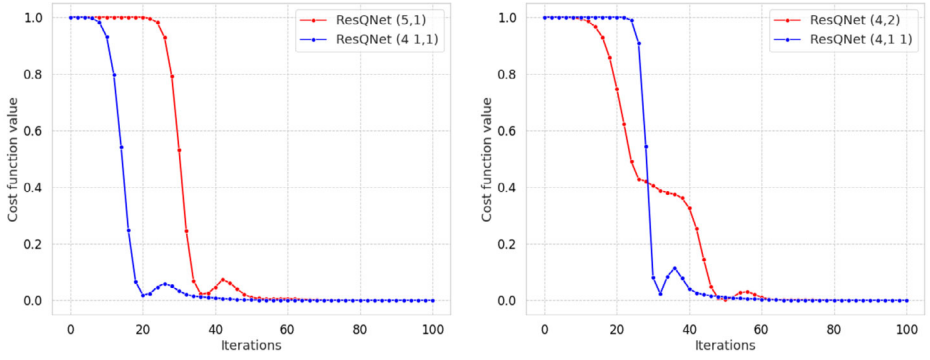

Figure 15 Training comparison of 2-QN and 3-QN ResQNets for 20 qubit layers. The parentheses denote the D_L per QN and the comma denotes the residual point

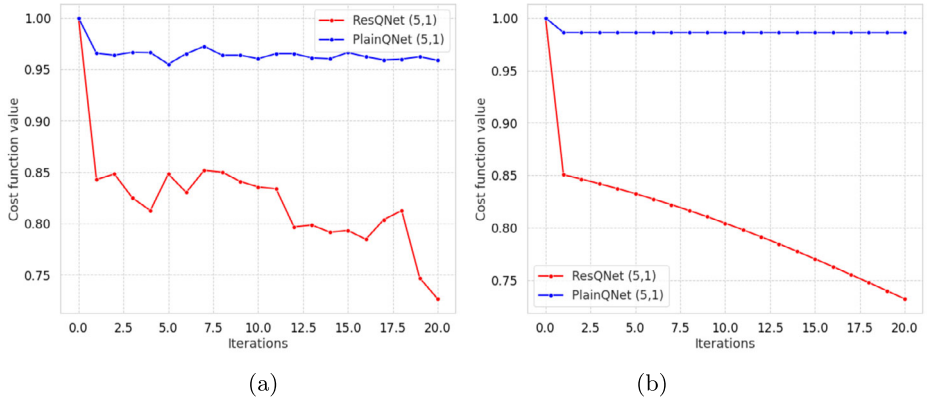

Figure 16 Training comparison ResQNets and PlainQNets on (a) real quantum device and (b) simulator. The values in parentheses denote the depth per QN

quantum layer with 20 epochs using an IBM's quantum device, namely *ibmq_lima*. The quantum layers depth was fixed to $D_L = 6$ with $D_L = 5$ in the first QN, and $D_L = 1$ in the second QN. This depth combination was chosen considering all the results discussed previously. We note that due to the limited number of publicly available quantum devices, the queue times for executing the jobs are considerably long. Therefore, to minimize the training time, we chose to reduce the number of epochs for real-device training. We trained both PlainQNets and ResQNets for only 20 epochs on real devices instead of 100 epochs as in the case of simulation. The training results are illustrated in Fig. 16.

The results presented in Fig. 16a reveal that ResQNets have been trained successfully on a real device, whereas PlainQNets have not been trained on a real device. The same trend is observed when both networks are executed on the simulator, as depicted in Fig. 16b. However, when both PlainQNets and ResQNets are trained on a real device, a slight fluctuation is observed while approaching the optimal solution due to hardware noise, as compared to the simulation results. Despite the presence of noise, the rate of decrease in the loss value for ResQNets is almost identical for both simulation and real experiments. According to [52], hardware noise can potentially cause BP. However, our results demonstrate that our

proposed ResQNets are somewhat resilient against hardware noise, as they achieve similar performance to that of the simulator (though with some fluctuations).

6 Conclusion

The problem of barren plateaus (BP) in quantum neural networks (QNNs) is a critical hurdle on the road to the practical realization of QNNs. There have been several attempts to resolve this issue, but the impact of BP can still vary greatly depending on the application and the architecture of the quantum layers. Thus, it is essential to have multiple solutions for BP to cover a wide range of problems.

In this paper, we propose residual quantum neural networks (ResQNets) to address the issue of BP in QNNs. Our approach is inspired by classical residual neural networks (ResNets), which were introduced to overcome the problem of vanishing gradients in classical neural networks.

In traditional QNNs, a single parameterized quantum circuit (PQC) with arbitrary depth is included within a single quantum node (QN). To create ResQNets, we split the conventional QNN architecture into multiple QNs, each of which contains its own PQC with varying depths. Splitting the QNNs allows us to introduce the residual connections between the QNs, forming our proposed ResQNets. In simple QNNs without residual connections (referred to as PlainQNets), the output from the previous QN serves as the input to the next. On the other hand, in ResQNets, one or multiple QNs can serve as residual blocks, with the output from a previous residual block being added to its input before it is passed on to the next QN.

In our study, we first demonstrate the efficacy of the proposed splitting of the conventional QNN architecture into multiple QNs (PlainQNets) by comparing their performance to that of conventional QNNs (simple PlainQNets). The comparison results indicated that the PlainQNets perform better than or equivalent to that of conventional QNNs. Subsequently, we compare the performance of PlainQNets with that of our proposed ResQNets through several training experiments. Our analysis of the cost function landscapes for quantum layers of increasing qubits shows that incorporating residual connections results in improved training performance.

Based on our findings, we conclude that the proposed ResQNets provide a promising solution to overcome the problem of BP in QNNs and offer a potential direction for further research in the field of quantum machine learning.

Acknowledgements
The authors thank Qatar National Library for supporting the open access publication of this article.

Funding
Open Access funding provided by the Qatar National Library.

Abbreviations
QNN, Quantum Neural Network; BP, Barren Plateaus; QN, Quantum Node; ResNet, Classical Residual Neural Network; ResQNet, Quantum Residual Neural Network; PQC, Parameterized Quantum Circuit.

Data availability
The code that was used to generate the numerical results in this work can be provided on request.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Both authors have approved the publication. The research in this work did not involve any human, animal or other participants.

Competing interests
The authors declare no competing interests.

Author contributions
Both authors contributed to the work described in this paper. M.K conceived the idea and performed the numerical experiments, and prepared the manuscript. S.A.K reviewed the manuscript and approved the submission.

Received: 9 March 2023 Accepted: 25 December 2023 Published online: 10 January 2024

References

1. Bharti K, Cervera-Lierta A, Kyaw TH, Haug T, Alperin-Lea S, Anand A, Degroote M, Heimonen H, Kottmann JS, Menke T, Mok W-K, Sim S, Kwek L-C, Aspuru-Guzik A. Noisy intermediate-scale quantum algorithms. Rev Mod Phys. 2022;94:015004. https://doi.org/10.1103/RevModPhys.94.015004.
2. Preskill J. Quantum computing in the NISQ era and beyond. Quantum. 2018;2:79. https://doi.org/10.22331/q-2018-08-06-79.
3. Schuld M, Bergholm V, Gogolin C, Izaac J, Killoran N. Evaluating analytic gradients on quantum hardware. Phys Rev A. 2019;99(3):032331.
4. Lau JWZ, Lim KH, Shrotriya H et al. NISQ computing: where are we and where do we go? AAPPS Bull. 2022;32(1):27. https://doi.org/10.1007/s43673-022-00058-z.
5. Roffe J. Quantum error correction: an introductory guide. Contemp Phys. 2019;60(3):226–45. https://doi.org/10.1080/00107514.2019.1667078.
6. Moll N, Barkoutsos P, Bishop LS, Chow JM, Cross A, Egger DJ, Filipp S, Fuhrer A, Gambetta JM, Ganzhorn M, Kandala A, Mezzacapo A, Müller P, Riess W, Salis G, Smolin J, Tavernelli I, Temme K. Quantum optimization using variational algorithms on near-term quantum devices. Quantum Sci Technol. 2018;3(3):030503. https://doi.org/10.1088/2058-9565/aab822.
7. Biamonte J, Wittek P, Pancotti N, Rebentrost P, Wiebe N, Lloyd S. Quantum machine learning. Nature. 2017;549(7671):195–202. https://doi.org/10.1038/nature23474.
8. Schuld M, Sinayskiy I, Petruccione F. An introduction to quantum machine learning. Contemp Phys. 2014;56(2):172–85. https://doi.org/10.1080/00107514.2014.964942.
9. Mitarai K, Negoro M, Kitagawa M, Fujii K. Quantum circuit learning. Phys Rev A. 2018;98:032309. https://doi.org/10.1103/PhysRevA.98.032309.
10. Liu Y, Arunachalam S, Temme K. A rigorous and robust quantum speed-up in supervised machine learning. Nat Phys. 2021;17(9):1013–7. https://doi.org/10.1038/s41567-021-01287-z.
11. Huang H-Y, Broughton M, Cotler J, Chen S, Li J, Mohseni M, Neven H, Babbush R, Kueng R, Preskill J, McClean JR. Quantum advantage in learning from experiments. Science. 2022;376(6598):1182–6. https://doi.org/10.1126/science.abn7293. https://www.science.org/doi/pdf/10.1126/science.abn7293.
12. Cong I, Choi S, Lukin MD. Quantum convolutional neural networks. Nat Phys. 2019;15(12):1273–8. https://doi.org/10.1038/s41567-019-0648-8.
13. Schatzki L, Arrasmith A, Coles PJ, Cerezo M. Entangled datasets for quantum machine learning. 2021. https://doi.org/10.48550/ARXIV.2109.03400. arXiv:2109.03400.
14. Caro MC, Huang H-Y, Cerezo M, Sharma K, Sornborger A, Cincio L, Coles PJ. Generalization in quantum machine learning from few training data. Nat Commun. 2022;13(1):4919. https://doi.org/10.1038/s41467-022-32550-3.
15. de Leon NP, Itoh KM, Kim D, Mehta KK, Northup TE, Paik H, Palmer BS, Samarth N, Sangtawesin S, Steuerman DW. Materials challenges and opportunities for quantum computing hardware. Science. 2021;372(6539):2823. https://doi.org/10.1126/science.abb2823. https://www.science.org/doi/pdf/10.1126/science.abb2823.
16. Lloyd S, Mohseni M, Rebentrost P. Quantum algorithms for supervised and unsupervised machine learning. 2014. https://doi.org/10.48550/ARXIV.1307.0411. arXiv:1307.0411.
17. Linke NM, Gutierrez M, Landsman KA, Figgatt C, Debnath S, Brown KR, Monroe C. Fault-tolerant quantum error detection. Sci Adv. 2017;3(10):1701074. https://doi.org/10.1126/sciadv.1701074.
18. Abel S, Criado JC, Spannowsky M. Completely quantum neural networks. Phys Rev A. 2022;106:022601. https://doi.org/10.1103/PhysRevA.106.022601.
19. Rebentrost P, Mohseni M, Lloyd S. Quantum support vector machine for big data classification. Phys Rev Lett. 2014;113(13):130503. https://doi.org/10.1103/physrevlett.113.130503.
20. Havlíček V, Córcoles AD, Temme K, Harrow AW, Kandala A, Chow JM, Gambetta JM. Supervised learning with quantum-enhanced feature spaces. Nature. 2019;567(7747):209–12. https://doi.org/10.1038/s41586-019-0980-2.
21. Lloyd S, Mohseni M, Rebentrost P. Quantum principal component analysis. Nat Phys. 2014;10(9):631–3. https://doi.org/10.1038/nphys3029.
22. Dunjko V, Taylor JM, Briegel HJ. Advances in quantum reinforcement learning. In: 2017 IEEE international conference on systems, man, and cybernetics (SMC). 2017. p. 282–7. https://doi.org/10.1109/SMC.2017.8122616.
23. Meyer N, Ufrecht C, Periyasamy M, Scherer DD, Plinge A, Mutschler C. A survey on quantum reinforcement learning. 2022. arXiv:2211.03464.
24. Lockwood O, Si M. Reinforcement learning with quantum variational circuit. In: Proceedings of the AAAI conference on artificial intelligence and interactive digital entertainment. vol. 16. 2020. p. 245–51.
25. Farhi E, Neven H. Classification with quantum neural networks on near term processors. 2018. https://doi.org/10.48550/ARXIV.1802.06002. arXiv:1802.06002.
26. Mari A, Bromley TR, Izaac J, Schuld M, Killoran N. Transfer learning in hybrid classical-quantum neural networks. Quantum. 2020;4:340.
27. Mathur N, Landman J, Li YY, Strahm M, Kazdaghli S, Prakash A, Kerenidis I. Medical image classification via quantum neural networks. arXiv preprint. 2021. arXiv:2109.01831.

28. Pesah A, Cerezo M, Wang S, Volkoff T, Sornborger AT, Coles PJ. Absence of barren plateaus in quantum convolutional neural networks. Phys Rev X. 2021;11:041011. https://doi.org/10.1103/PhysRevX.11.041011.
29. Chen SY-C, Wei T-C, Zhang C, Yu H, Yoo S. Quantum convolutional neural networks for high energy physics data analysis. Phys Rev Res. 2022;4:013231. https://doi.org/10.1103/PhysRevResearch.4.013231.
30. Meichanetzidis K, Gogioso S, de Felice G, Chiappori N, Toumi A, Coecke B. Quantum natural language processing on near-term quantum computers. Discret Math Theor Comput Sci. 2021;340:213–29. https://doi.org/10.4204/eptcs.340.11.
31. Coecke B, de Felice G, Meichanetzidis K, Toumi A. Foundations for near-term quantum natural language processing. arXiv preprint. 2020. arXiv:2012.03755.
32. Di Sipio R, Huang J-H, Chen SY-C, Mangini S, Worring M. The dawn of quantum natural language processing. In: ICASSP 2022–2022 IEEE international conference on acoustics, speech and signal processing (ICASSP). 2022. p. 8612–6. https://doi.org/10.1109/ICASSP43922.2022.9747675.
33. Gao S, Yang Y-G. A novel quantum recommender system. Phys Scr. 2022;98(1):010001. https://doi.org/10.1088/1402-4896/aca4a8.
34. Wan KH, Dahlsten O, Kristjánsson H, Gardner R, Kim MS. Quantum generalisation of feedforward neural networks. npj Quantum Inf. 2017;3(1):36. https://doi.org/10.1038/s41534-017-0032-4.
35. Killoran N, Bromley TR, Arrazola JM, Schuld M, Quesada N, Lloyd S. Continuous-variable quantum neural networks. Phys Rev Res. 2019;1:033063. https://doi.org/10.1103/PhysRevResearch.1.033063.
36. Zoufal C, Lucchi A, Woerner S. Quantum generative adversarial networks for learning and loading random distributions. npj Quantum Inf. 2019;5(1):103. https://doi.org/10.1038/s41534-019-0223-2.
37. Beer K, Bondarenko D, Farrelly T, Osborne TJ, Salzmann R, Scheiermann D, Wolf R. Training deep quantum neural networks. Nat Commun. 2020;11(1):1–6.
38. Kashif M, Al-Kuwari S. Design space exploration of hybrid quantum classical neural networks. Electronics. 2021;10(23):2980. https://doi.org/10.3390/electronics10232980.
39. Du Y, Hsieh M-H, Liu T, Tao D. Expressive power of parametrized quantum circuits. Phys Rev Res. 2020;2:033125. https://doi.org/10.1103/PhysRevResearch.2.033125.
40. Kashif M, Al-Kuwari S. Demonstrating quantum advantage in hybrid quantum neural networks for model capacity. In: 2022 IEEE international conference on rebooting computing (ICRC). 2022. p. 36–44. https://doi.org/10.1109/ICRC57508.2022.00011.
41. Amin MH, Andriyash E, Rolfe J, Kulchytskyy B, Melko R. Quantum Boltzmann machine. Phys Rev X. 2018;8:021050. https://doi.org/10.1103/PhysRevX.8.021050.
42. Zoufal C, Lucchi A, Woerner S. Variational quantum Boltzmann machines. Quantum Mach Intell. 2021;3(1):7. https://doi.org/10.1007/s42484-020-00033-7.
43. Romero J, Olson JP, Aspuru-Guzik A. Quantum autoencoders for efficient compression of quantum data. Quantum Sci Technol. 2017;2(4):045001. https://doi.org/10.1088/2058-9565/aa8072.
44. Bondarenko D, Feldmann P. Quantum autoencoders to denoise quantum data. Phys Rev Lett. 2020;124:130502. https://doi.org/10.1103/PhysRevLett.124.130502.
45. Kwak Y, Yun WJ, Jung S, Kim J-K, Kim J. Introduction to quantum reinforcement learning: theory and pennylane-based implementation. In: 2021 international conference on information and communication technology convergence (ICTC). 2021. p. 416–20. https://doi.org/10.1109/ICTC52510.2021.9620885.
46. Chen SY-C, Yang C-HH, Qi J, Chen P-Y, Ma X, Goan H-S. Variational quantum circuits for deep reinforcement learning. IEEE Access. 2020;8:141007–24.
47. Banchi L, Zhuang Q, Pirandola S. Quantum-enhanced barcode decoding and pattern recognition. Phys Rev Appl. 2020;14:064026. https://doi.org/10.1103/PhysRevApplied.14.064026.
48. Grimsley HR, Economou SE, Barnes E, Mayhall NJ. An adaptive variational algorithm for exact molecular simulations on a quantum computer. Nat Commun. 2019;10(1):3007. https://doi.org/10.1038/s41467-019-10988-2.
49. Arrasmith A, Holmes Z, Cerezo M, Coles PJ. Equivalence of quantum barren plateaus to cost concentration and narrow gorges. Quantum Sci Technol. 2022;7:045015.
50. McClean JR, Boixo S, Smelyanskiy VN, Babbush R, Neven H. Barren plateaus in quantum neural network training landscapes. Nat Commun. 2018;9(1):4812. https://doi.org/10.1038/s41467-018-07090-4.
51. Wierichs D, Gogolin C, Kastoryano M. Avoiding local minima in variational quantum eigensolvers with the natural gradient optimizer. Phys Rev Res. 2020;2:043246. https://doi.org/10.1103/PhysRevResearch.2.043246.
52. Wang S, Fontana E, Cerezo M, Sharma K, Sone A, Cincio L, Coles PJ. Noise-induced barren plateaus in variational quantum algorithms. Nat Commun. 2021;12(1):6961. https://doi.org/10.1038/s41467-021-27045-6.
53. Fontana E, Cerezo M, Arrasmith A, Rungger I, Coles PJ. Optimizing parametrized quantum circuits via noise-induced breaking of symmetries. 2020. https://doi.org/10.48550/ARXIV.2011.08763. arXiv:2011.08763.
54. Wang S, Czarnik P, Arrasmith A, Cerezo M, Cincio L, Coles PJ. Can error mitigation improve trainability of noisy variational quantum algorithms? arXiv preprint. 2021. arXiv:2109.01051.
55. Stilck França D, Garcia-Patron R. Limitations of optimization algorithms on noisy quantum devices. Nat Phys. 2021;17(11):1221–7.
56. Liu H-Y, Sun T-P, Wu Y-C, Han Y-J, Guo G-P. Mitigating barren plateaus with transfer-learning-inspired parameter initializations. New J Phys. 2023;25(1):013039. https://doi.org/10.1088/1367-2630/acb58e.
57. Verdon G, Broughton M, McClean JR, Sung KJ, Babbush R, Jiang Z, Neven H, Mohseni M. Learning to learn with quantum neural networks via classical neural networks. 2019. https://doi.org/10.48550/ARXIV.1907.05415. arXiv:1907.05415.
58. Cerezo M, Sone A, Volkoff T, Cincio L, Coles PJ. Cost function dependent barren plateaus in shallow parametrized quantum circuits. Nat Commun. 2021;12(1):1791. https://doi.org/10.1038/s41467-021-21728-w.
59. Kashif M, Al-Kuwari S. The impact of cost function globality and locality in hybrid quantum neural networks on nisq devices. Mach Learn: Sci Technol. 2023;4(1):015004. https://doi.org/10.1088/2632-2153/acb12f.
60. Skolik A, McClean JR, Mohseni M, van der Smagt P, Leib M. Layerwise learning for quantum neural networks. Quantum Mach Intell. 2021;3(1):5. https://doi.org/10.1007/s42484-020-00036-4.

61. Kulshrestha A, Safro I. Beinit: avoiding barren plateaus in variational quantum algorithms. In: 2022 IEEE international conference on quantum computing and engineering (QCE). 2022. p. 197–203. https://doi.org/10.1109/QCE53715.2022.00039.
62. Kashif M, Al-Kuwari S. The unified effect of data encoding, ansatz expressibility and entanglement on the trainability of hqnns. Int J Parallel Emerg Distrib Syst. 2023;38(5):362–400. https://doi.org/10.1080/17445760.2023.2231163.
63. Liang Y, Peng W, Zheng Z-J, Silvén O, Zhao G. A hybrid quantum–classical neural network with deep residual learning. Neural Netw. 2021;143:133–47. https://doi.org/10.1016/j.neunet.2021.05.028.
64. Abd El-Aziz RM, Taloba AI, Alghamdi FA. Quantum computing optimization technique for iot platform using modified deep residual approach. Alex Eng J. 2022;61(12):12497–509. https://doi.org/10.1016/j.aej.2022.06.029.
65. Dayang Q. Resnet-inspired hybrid quantum neural network. Graduate Institute of Physics, National Taiwan University. 2020. https://doi.org/10.6342/NTU202003705.
66. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2016. p. 770–8.
67. LaRose R, Coyle B. Robust data encodings for quantum classifiers. Phys Rev A. 2020;102:032420. https://doi.org/10.1103/PhysRevA.102.032420.

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Submit your manuscript to a SpringerOpen journal and benefit from:

► Convenient online submission
► Rigorous peer review
► Open access: articles freely available online
► High visibility within the field
► Retaining the copyright to your article

Submit your next manuscript at ► springeropen.com