A hybrid quantum–classical neural network with deep residual learning

Yanying Liang a,b,*, Wei Peng a, Zhu-Jun Zheng b,c, Olli Silvén a, Guoying Zhao a

a Center for Machine Vision and Signal Analysis, University of Oulu, Oulu 90570, Finland
b School of Mathematics, South China University of Technology, Guangzhou 510641, China
c Laboratory of Quantum Science and Engineering, South China University of Technology, Guangzhou 510641, China

A R T I C L E   I N F O

Article history:
Received 10 January 2021
Received in revised form 18 May 2021
Accepted 25 May 2021
Available online 2 June 2021

Keywords:
Quantum computing
Quantum neural networks
Deep residual learning

A B S T R A C T

Inspired by the success of classical neural networks, there has been tremendous effort to develop classical effective neural networks into quantum concept. In this paper, a novel hybrid quantum–classical neural network with deep residual learning (Res-HQCNN) is proposed. We firstly analyse how to connect residual block structure with a quantum neural network, and give the corresponding training algorithm. At the same time, the advantages and disadvantages of transforming deep residual learning into quantum concept are provided. As a result, the model can be trained in an end-to-end fashion, analogue to the backpropagation in classical neural networks. To explore the effectiveness of Res-HQCNN, we perform extensive experiments for quantum data with or without noisy on classical computer. The experimental results show the Res-HQCNN performs better to learn an unknown unitary transformation and has stronger robustness for noisy data, when compared to state of the arts. Moreover, the possible methods of combining residual learning with quantum neural networks are also discussed.

© 2021 The Author(s). Published by Elsevier Ltd. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

1. Introduction

Artificial neural networks (ANNs) stand for one of the most prosperous computational paradigms (Nielsen, 2015). Early neural networks can be traced back to the McCulloch–Pitts neurons in 1943 (McCulloch & Pitts, 1943). Over the past few decades, taking advantage of a number of technical factors, such as new and scalable software platforms (Bergstra et al., 2010; Jia et al., 2014; Maclaurin et al., 2015; Paszke et al., 2017, 2019) and powerful special-purpose computational hardware (Chetlur et al., 2014; Jouppi et al., 2017), the development of machine learning techniques based on neural networks makes progress and breakthroughs. Currently, neural networks achieve significant success and have wide applications in various machine learning fields like pattern recognition, video analysis, medical diagnosis, and robot control (Amato et al., 2013; Bishop et al., 1995; Mitchell & Thrun, 1993; Nishani & Çiço, 2017). One notable reason is the increasingly passion for exploring new neural architectures, including manual ways by expert knowledge and automatic ways by auto machine learning (He et al., 2016a; Peng et al., 2020a; Pham et al., 2018). Particularly, in 2016, deep residual networks (ResNets) are proposed with extremely deep architectures showing excellent accuracy and nice convergence behaviors (He et al., 2016a, 2016b). Their result won the 1st place on the ImageNet Large Scale Visual Recognition Challenge 2015 classification task for ImageNet classification and ResNet (and its variants) achieves revolutionary success in many research and industry applications.

In parallel with the development of ANNs, quantum neural networks (QNNs) appear with the potential to evade the limitation of the classical computation due to the use of quantum computing. Compared with classical computing, quantum computing has possible advantages with the properties of quantum mechanics, such as quantum entanglement, quantum superposition and massive parallelism (Biamonte et al., 2017; Rønnow et al., 2014). Earlier quantum neural networks (Ezhov & Ventura, 2000; Purushothaman & Karayiannis, 1997) can even trace back to two decades ago. As the current quantum technologies and devices developing, there is a huge passion to combine quantum computing with ANNs for quantum data with or without noisy (Alvarez-Rodriguez et al., 2017; Beer et al., 2020; Bisio et al., 2010; Chen et al., 2021; Dunjko & Briegel, 2018; Dunjko et al., 2016; Li & Wang, 2020; Patterson et al., 2021; Sasaki & Carlini, 2002; Schuld et al., 2014; Sedlák et al., 2019; Sentis et al., 2019; Skolik et al., 2021; Verdon et al., 2018; Zhao et al., 2019).

Among these works, Beer et al. (2020) caught our attention. On one hand, it provides the readers with accessible code. On the other hand, the paper presents a truly quantum neurons forming
Fig. 1. Numerical results of QNNs in Beer et al. (2020).
impress us a lot. In detail, Bisio et al. (2010) addressed this task and found optimal strategy to store and retrieve an unknown unitary transformation on a quantum memory. Soon after, Sedlák et al. (2019) designed an optimal protocol of unitary channels, which generalizes the results in Bisio et al. (2010). Moreover, Beer et al. (2020) proposed a quantum neural network with remarkable generalization behavior for the task of learning an unknown unitary quantum transformation. We hope the ability of learning unknown unitary transformation can be improved due to the new idea we propose.

2.2. Residual scheme in neural networks

Proposed in 2012, AlexNet (Krizhevsky et al., 2017) became one of the most famous neural network architecture in deep learning era. This was treated as the first time that deep neural network was more successful than traditional, hand-crafted feature learning on the ImageNet (Deng et al., 2009). Since then, researchers pay many efforts to make the network deeper, as deeper architecture could potentially extract more important semantic information. But deeper networks are more difficult to train, due to the notorious vanishing/exploding gradient problem. Residual scheme in ResNet (He et al., 2016a) is one of the most successful strategy of improving current neural networks. ResNet makes it possible to train up to hundreds or even thousands of layers and still achieves compelling performance. Basically, this scheme reformulates the layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions. Based on this, a residual neural network builds on constructs known from pyramidal cells in the cerebral cortex, utilizing skip connections, or shortcuts to jump over some layers. This simple but efficient strategy largely improves the performance of current neural architectures in many fields. For instance, in image classification tasks, many variants of ResNets (Chen et al., 2018; Dong et al., 2020; He et al., 2016a; Korpi et al., 2020; Xie et al., 2017) are proposed and get the state-of-the-art performance. This scheme is even introduced into graph convolutional networks. For instance, works from Peng et al. (2020a, 2020b) and Yan et al. (2018) also endow graph convolutional network with the residual connections, capturing a better representations for skeleton graphs. Nevertheless, as far as we know, there is no work introducing the residual scheme in the field of quantum neural networks. In this paper, we will make the first attempt to do this and present an efficient Res-HQCNN which can be trained with an end-to-end fashion.

3. Preliminaries

3.1. Qubits and quantum operators

Analogous to the role bit is the smallest unit of classical computing, qubit is the smallest unit in quantum computing. The notation \( | \rangle \) is called a ket which is used to indicate that the object \( | \rangle \) is a column vector. The complex conjugate transpose of \( | \rangle \) is \( \langle \rangle \), which is called a bra (Nielsen & Chuang, 2002). A two-level quantum system in a two-dimensional Hilbert space \( \mathbb{C}^2 \) with basis \( \{|0\rangle, |1\rangle \} \) is a single qubit, which can be written from the superposition principles:

\[
|\psi\rangle = \alpha |0\rangle + \beta |1\rangle
\]

\[
|\alpha|^2 + |\beta|^2 = 1, \alpha, \beta \in \mathbb{C}.
\]

The quantum operators used in this paper mainly include tensor product operator, reduced density operator and partial trace. Tensor product is a way to extend the dimension of vector spaces through putting vector space together (Nielsen & Chuang, 2002). The symbol for tensor product is denoted by \( \otimes \). Assume \( A \) is an \( m \times n \) matrix, \( B \) is a \( p \times q \) matrix, then \( A \otimes B \) is an \( m \times n \times p \times q \) matrix:

\[
A \otimes B = \begin{bmatrix}
A_{11}B & A_{12}B & \cdots & A_{1n}B \\
A_{21}B & A_{22}B & \cdots & A_{2n}B \\
\vdots & \vdots & \ddots & \vdots \\
A_{m1}B & A_{m2}B & \cdots & A_{mn}B
\end{bmatrix}.
\]

Another important operator used in this paper is the reduced density operator (Nielsen & Chuang, 2002). It is often used to get the desired subsystems of a composite quantum system. Assume \( A \) and \( B \) are two physical systems. The state in \( A \otimes B \) is described by density matrix \( \rho_{AB} \). The reduced density operator for \( A \) is given by

\[
\rho_A = \text{tr}_B(\rho_{AB}),
\]

where \( \text{tr}_B \) is a map of operators called partial trace over \( B \). Here partial trace is defined as

\[
\text{tr}_B(|a_1\rangle\langle a_2| \otimes |b_1\rangle\langle b_2|) = |a_1\rangle\langle a_2| \text{tr}(|b_1\rangle\langle b_2|).
\]

where \( |a_1\rangle \) and \( |a_2\rangle \) are any vectors in \( A \), \( |b_1\rangle \) and \( |b_2\rangle \) are any vectors in \( B \). For example, if \( A \) and \( B \) are both two-dimensional complex vector space \( \mathbb{C}^2 \), then

\[
\rho_A = \text{tr}_B(|0\rangle_A\langle 0| \otimes |0\rangle_B\langle 0|) = |0\rangle_A\langle 0| \text{tr}(|0\rangle_B\langle 0|) = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix},
\]

\[
\rho_B = \text{tr}_A(|0\rangle_A\langle 0| \otimes |1\rangle_B\langle 1|) = |1\rangle_B\langle 1| \text{tr}(|0\rangle_A\langle 0|) = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}.
\]

Here \( |0\rangle_A \) means \( |0\rangle \) in physical system \( A \).

3.2. Deep residual learning

Training a deep neural network can be computationally costly. He et al. (2016a) proposed neural networks with deep residual learning framework, which is easier to optimize and obtain high accuracy from increased depth. In a residual block structure, there is a shortcut pathway connecting the input and output of a block structure. Specifically, residual learning chooses to fit the residual mapping \( F(x) := H(x) - x \), rather than approximating the desired underlying mapping \( H(x) \) directly. The final mapping of a residual block structure is \( F(x)+x \), which is equivalent to \( H(x) \), see Fig. 2.

The degradation problem tells us that the deeper network has higher training error and test error. But compared with shallower counterpart, the performance of deeper network should not be worse. This suggests that the network might have difficulties in approximating identity mapping. So optimizing the residual mapping \( F(x) \) towards zero is relatively easier than approximating \( H(x) \) into identity mapping.

For ANNs, deep residual learning has the advantage of solving the problem of vanishing/exploding gradients and the degradation problem. This paper is mainly based on the QNNs in Beer et al. (2020), which is absence of a barren plateau in the cost function landscape. So we expect the QNNs with residual learning...
can get improved performance of the cost function both for clean and noisy data. We hope that the deeper the Res-HQCNN, the more effective it will be.

4. The model of Res-HQCNN

In this section, we define the architecture of Res-HQCNN based on the QNNs in Beer et al. (2020). According to the mechanism of the defined Res-HQCNN, we explain its training algorithm in different cases.

4.1. The architecture of Res-HQCNN

We firstly define a residual block structure in Res-HQCNN. Then the architecture of Res-HQCNN with multiple layers is presented. For better understanding the mechanism, we show an example of Res-HQCNN with one hidden layer. Finally, we give an analysis about the differences between former QNNs and Res-HQCNN.

Based on the residual block structure in Fig. 2, a new residual block structure in Res-HQCNN is defined as follows. For convenience, we put forward a few assumptions and notations at the beginning.

Res-HQCNN has $L$ hidden layers. The perceptron nodes of each layer represent single qubits. Denote $m_l$ as the number of nodes in each layer $l$ and we assume $m_{l-1} \leq m_l$ for $l = 1, 2, \ldots, L$. Here $l = 0$ represents the input layer and $l = L + 1$ corresponds to output layer. $\rho^{in}$ and $\rho^{out}$ represent the input and output state of layer $l$ in Res-HQCNN. Then the residual block structure can be designed in Fig. 3.

Mathematically, we set the input mapping and final mapping of the residual block structure in Res-HQCNN as $\rho^{in}$ and $\rho^{out}$, respectively. The residual mapping is chosen as $\rho^{l+1}_{out}$. It is important to note that the output of the former layer is not the exact input of the next layer. As shown in Fig. 3, the new input of layer $l+1$ is the addition of the output state and the input state in layer $l$. Here the additive operation corresponds to the matrix element-wise addition. Due to $m_{l-1} \leq m_l$, we apply tensor product to $\rho^{in}$ and $|0 \cdots 0\rangle_{\Delta m_l} |0 \cdots 0\rangle$ to keep the same dimension.

Next, we go on investigating the method to merge the residual block structure and quantum neural network together. Define quantum perceptron in layer $l$ of Res-HQCNN to be an arbitrary unitary operator with $m_{l-1}$ input qubits and one output qubit. For example, as presented in Fig. 4, quantum perceptron $U^j_l$ is a $(m_{l-1}+1)$-qubit unitary operator for $j = 1, 2, \ldots, m_l$. The Res-HQCNN is made up of quantum perceptrons with $L$ hidden layers. It acts on an input state $\rho^{in}_l$ of input qubits and obtains a mixed state $\rho^{l+1}_{out}$ for the output qubits based on the layer unitary operator $U^j_l$ in the form of a matrix product of quantum perceptrons: $U^j_l = U^j_{m_l} U^j_{m_l-1} \cdots U^j_{1}$. Here $U^j_{m_l}$ acts on the qubits in layer $l-1$ and $j$ for $j = 1, 2, \ldots, m_l$. The unitary operators are arbitrary, and they do not always commute, so the order of the layer unitary is important. During processing information from $\rho^{1}_{in}$ to $\rho^{L+1}_{out}$, the residual block structure produces the new input state for layer $l+1$ through adding the input state with the output state of layer $l$ for $l = 1, 2, \ldots, L$.

To facilitate the understanding, we give an example of the mechanism for Res-HQCNN with one hidden layer, see Fig. 5.

Define the layer unitary between the input layer and the hidden layer as $U^1 = U^1_1 U^1_2 U^1_1$, which is in the form of a matrix product of quantum perceptrons. Analogously we define the layer unitary $U^2 = U^2_2 \cdots U^2_2$ between the hidden layer and the output layer. For the first step, we apply the quantum perceptrons layer-wise from top to bottom, then the output state $\rho^{1}_{out}$ of the hidden
layer is
\[ \rho_{\text{out}}^{l} = \text{tr}_{in}(U^{l}(\rho_{\text{in}}^{l}\otimes|000\rangle_{\text{hid}}|000\rangle_{\text{in}}U^{\dagger l})). \]

Next, we apply residual block structure to \( \rho_{\text{in}}^{l} \) and \( \rho_{\text{out}}^{l} \) in order to get a new input state for the output layer:
\[ \rho_{\text{in}}^{l+1} = \rho_{\text{in}}^{l} + \left(\rho_{\text{in}}^{l}\otimes|0\rangle\langle0|\right). \]

In the third step, we get the final output state for this Res-HQCNN in Fig. 5:
\[ \rho_{\text{out}}^{L} = \text{tr}_{\text{in}}(U^{L}(\rho_{\text{in}}^{L}\otimes|000\rangle_{\text{out}}|000\rangle_{\text{in}}U^{\dagger L}). \]

Compared the former QNNs in Beer et al. (2020) with Res-HQCNN, we find that the trace value of the input state \( \rho_{\text{in}}^{l+1} \) for some \( l \) changes due to the addition operation in the residual block structure. For example, if we set \( L = 2 \), \( \rho_{\text{in}}^{2} = (\rho_{\text{in}}^{1}\otimes|0\rangle_{\text{in}}-m_{1}\otimes|0\rangle_{\text{in}}) + \rho_{\text{in}}^{1} \), and \( \rho_{\text{in}}^{1} = (\rho_{\text{in}}^{0}\otimes|0\rangle_{\text{in}}-m_{0}\otimes|0\rangle_{\text{in}}) + \rho_{\text{in}}^{0} \), then the trace values of \( \rho_{\text{in}}^{2} \) and \( \rho_{\text{in}}^{1} \) are 2 and 4, respectively. In theory, \( \rho_{\text{in}}^{2} \) and \( \rho_{\text{in}}^{1} \) are not density matrix, and we cannot apply the training algorithm in quantum computer. However, every coin has two sides. The residual block structure improves the performance of cost function, especially for deeper network, which can be demonstrated in the experiment part.

One may also notice that we can apply the residual block structure for all the hidden layers, but not for the last output layer. Since we have assumed \( m_{l-1} \leq m_{l} \) for \( l = 1, 2, \ldots, L \) and \( m_{0} = m_{L+1} \), then in general, the qubits in the last output layer \( L+1 \) are no more than the qubits in layer \( L \). If we use the residual block structure to the last output layer \( L+1 \), then the final output of the network will be \( \rho_{\text{out}}^{L+1} = \rho_{\text{in}}^{L+1} + \rho_{\text{out}}^{L+1} \). The dimension of \( \rho_{\text{in}}^{L+1} \) is no less than the dimension of \( \rho_{\text{out}}^{L+1} \), so we should apply partial trace to \( \rho_{\text{in}}^{L+1} \) in order to keep the rule of matrix addition. As we mentioned before, the residual block structure in Res-HQCNN also has difficulty in approximating identity mapping. However, we will lose some information of \( \rho_{\text{in}}^{L+1} \) in this way, which is contradicted with the goal of using residual scheme. We also provide an experimental result to show the inefficiency when applying residual block structure to the last output layer.

4.2. The training algorithm of Res-HQCNN

We randomly generate \( N \) pairs training data which are possibly unknown quantum states in the form of \( (|\phi_{\text{in}}^{x}\rangle, |\phi_{\text{out}}^{x}\rangle) \) with \( x = 1, 2, \ldots, N \). It is also allowed to use enough copies of training pair \( (|\phi_{\text{in}}^{x}\rangle, |\phi_{\text{out}}^{x}\rangle) \) of specific \( x \) so that we can overcome quantum projection noise when computing the derivative of the cost function. Here for simplicity, we do not allow input states interacting with environment to produce output states (e.g., thermalization). We choose to consider the desired output \( |\phi_{\text{out}}^{x}\rangle \) as \( |\phi_{\text{out}}^{x}\rangle = V|\phi_{\text{in}}^{x}\rangle \) with \( V \) an unknown unitary operation.

The cost function we choose is based on the fidelity between the output of Res-HQCNN and the desired output averaged over all training data. But due to the definition of residual block structure and linear feature of fidelity, we define that the cost function of Res-HQCNN should divide \( 2^{t} \), where \( t \) is the number of residual block structure in Res-HQCNN:
\[ C(s) = \frac{1}{2^{TN}} \sum_{x=1}^{N} \text{tr}(\rho_{\text{out}}^{x}|\phi_{\text{out}}^{x}(s)\rangle\langle\phi_{\text{out}}^{x}|). \]

Since we want to know how close the network output state and the desired output state, and the closer they are, the bigger fidelity is. If the cost function comes to 1, we judge the Res-HQCNN performs best, otherwise 0 the worst. So our goal is to maximize the cost function in the training process.

For each layer \( l \) of Res-HQCNN, denote \( \rho_{\text{in}}^{l} \) as the input state of layer \( l \) and \( \rho_{\text{out}}^{l} \) as the output state of layer \( l \) with \( l = 1, 2, \ldots, L \) and \( x = 1, 2, \ldots, N \). We firstly consider the case that each layer \( l \) is added with a residual block structure with no skipping layer, then \( t = L \). The training algorithm for this kind of Res-HQCNN is given by the following steps:

I. Initialize:

1. Set step \( s = 0 \).
2. Choose all unitary \( U_{j}^{l}(0) \) randomly, \( j = 1, 2, \ldots, m_{l} \), where \( m_{l} \) is the number of nodes in layer \( l \).

II. For each layer \( l \) and each training pair \( (|\phi_{\text{in}}^{x}\rangle, |\phi_{\text{out}}^{x}\rangle) \), do the following steps:

II1. Feedforward:

1.1a. Tensor the input state \( \rho_{\text{in}}^{l}(s) \) to the initial state of layer \( l \),
\[ \rho_{\text{in}}^{l}(s) = |\phi_{\text{in}}^{x}\rangle\langle\phi_{\text{in}}^{x}|. \]

1.1b. Apply the layer unitary between layer \( l-1 \) and \( l \),
\[ U_{l}^{l}(s) = U_{l}^{l}(s)U_{l-1}(s)\cdots U_{1}(s)\rho_{\text{in}}^{l}(s)\otimes|0\rangle\langle0|\cdots|0\rangle\langle0|. \]

1.1c. Trace out layer \( l-1 \) and obtain the output state \( \rho_{\text{out}}^{l}(s) \) of layer \( l \),
\[ \rho_{\text{out}}^{l}(s) = \text{tr}_{l-1}(U_{l}^{l}(s)). \]

II2. Residual learning:

2.2a. Apply the residual block structure in Fig. 3 to \( \rho_{\text{in}}^{l}(s) \) and \( \rho_{\text{out}}^{l}(s) \) to obtain the new input state of layer \( l+1 \),
\[ \rho_{\text{in}}^{l+1}(s) = \rho_{\text{in}}^{l}(s) + \left(\rho_{\text{in}}^{l}(s)\otimes|0\rangle\langle0|\cdots|0\rangle\langle0|\right). \]

2.2b. Store \( \rho_{\text{in}}^{l+1}(s) \).

III. Update parameters:

III1. Compute the cost function:
\[ C(s) = \frac{1}{2^{TN}} \sum_{x=1}^{N} \text{tr}(\rho_{\text{out}}^{x}|\phi_{\text{out}}^{x}(s)\rangle\langle\phi_{\text{out}}^{x}|). \]

III2. Update the unitary of each perceptron via
\[ U_{j}^{l}(s+\epsilon) = e^{i\theta K_{j}^{l}(s)}U_{j}^{l}(s). \]
Here \( K_{j}^{l}(s) \) is the parameters matrix.

III2a. If Res-HQCNN has three layers with one hidden layer. When \( l = 1 \), we can get the analytical expression for \( K_{j}^{1}(s) \) after calculation:
\[ K_{j}^{1}(s) = \frac{2m_{l-1}}{N} \sum_{x=1}^{N} \text{tr}_{\text{est}}M_{j}^{l}, \]
where the trace is over all qubits of Res-HQCNN which are not affected by \( U_{l}^{l} \), \( \eta \) is the learning rate and \( N \) is the number of training pairs. Moreover, \( M_{j}^{l} \) is made up of two parts of the commutator:
\[ M_{j}^{l}(s) = U_{j}^{l}(s)\cdots U_{j}^{1}(s)(\rho_{\text{in}}^{x}(s)\otimes|0\rangle\langle0|\cdots|0\rangle\langle0|). \]

137

Neural Networks 143 (2021) 133–147
III2b. If Res-HQCNN has four layers with two hidden layers.

When \( l = 1 \), then

\[
K_j(l) = \eta \frac{2^{m_l-1}}{N} \sum_{s=1}^{N} \text{tr}_{\text{rest}}(M_j^l + N_j^l),
\]

where \( M_j^l \) is Eq. (3) and

\[
N_j^l(s) = \sum_{i=1}^{N} \langle \phi_{\text{out}}(s) | \phi_{\text{out}}(s) \rangle U_j^i(s) \rho_{\text{out}}^i(s) \otimes |0 \ldots 0 \rangle_{\text{dim}_l} |0 \ldots 0 \rangle \otimes |0 \ldots 0 \rangle_{\text{dim}_{l+1}} |0 \ldots 0 \rangle \otimes
\]

\[
|0 \ldots 0 \rangle (0 \ldots 0) U_j^i(s) \cdots U_j^i(s),
\]

\[
U_j^{i+1}(s) \cdots U_j^{i+1}(s) (0 \ldots 0) U_j^{i+1}(s),
\]

\[
(\text{id}(m_1) \otimes | \phi_{\text{out}}(s) \rangle \langle \phi_{\text{out}}(s) |) U_j^{i+1}(s) \cdots U_j^{i+1}(s). \tag{5}
\]

When \( l = 2 \), then

\[
K_j(l) = \eta \frac{2^{m_l-1}}{N} \sum_{s=1}^{N} \text{tr}_{\text{rest}}(M_j^l + Q_j^l),
\]

where \( M_j^l \) is Eq. (3) and

\[
Q_j^l(s) = \sum_{i=1}^{N} \langle \phi_{\text{out}}(s) | \phi_{\text{out}}(s) \rangle U_j^i(s) \rho_{\text{out}}^i(s) \otimes |0 \ldots 0 \rangle_{\text{dim}_l} |0 \ldots 0 \rangle \otimes
\]

\[
|0 \ldots 0 \rangle |0 \ldots 0 \rangle U_j^i(s) \cdots U_j^i(s),
\]

\[
|0 \ldots 0 \rangle \otimes (\text{id}(m_2) \otimes | \phi_{\text{out}}(s) \rangle \langle \phi_{\text{out}}(s) |) U_j^{i+1}(s) \cdots U_j^{i+1}(s). \tag{6}
\]

III2c. If Res-HQCNN has more than five layers, we can calculate the corresponding parameters matrices \( K_j(s) \) with the method in Appendix.

There is no fixed formula for \( K_j(s) \). It changes with the depth of network and the parameter \( l \).

III3. Update \( s = s + \epsilon \).

IV. Repeat steps II and III until reaching the maximum of the cost function.

As for other cases of Res-HQCNN, we can get \( t \leq L \) due to the skipping connection. The calculation method is the same as the one in Appendix, and the complexity of calculation decrease compared with the case \( t = L \). So we do not give the training algorithm for other cases in detail. Here we can automatically update
Finally, we generalize the experiments into noisy ent types, such as skipping one layer to connect the residual block structure. For example, in Beer et al. (2020), we find that the residual block structure shown in Fig. 3 that acts on the hidden layers, we plus a tilde on the top of the natural numbers. For example, a 1-2-1 quantum neural network in Beer et al. (2020) can be denoted as $[1, 2, 1]$ and a 1-2-1 quantum neural network with our residual block structure in this paper will be written as $[1, 2, 1]$. If there is a skipping connection, we plus a hat on the top of the number representing the skipped layer. For example, $[2, 3, \bar{3}, 2]$ denotes $\rho_x^{3u}(s) = \rho_x^{2us}(s) + \left(\rho_x^{1ins}(s) \otimes |0\rangle\langle 0|\right)$.

5.1. Elementary tests

We consider Res-HQCNN $[1, \bar{2}, 1]$ and $[2, \bar{3}, 2]$ with $\eta = 1/1.8$ and $\epsilon = 0.1$ for elementary tests, see Fig. 7(a). We find that the solid line is higher than the dashed one in the same color and both lines converge to 1 as the training rounds increase to 250. Compared the solid lines with dashed lines in the same color, we find that the solid lines have higher rate of convergence than the dashed ones before reaching 1.

We then test the performance of $[1, \bar{2}, \bar{1}]$ and $[2, \bar{3}, \bar{2}]$ with $\eta = 1/2$ and $\epsilon = 0.1$, which is the case that applying the residual block structure to the last output layer, see Fig. 7(b). Since the residual block structure is used twice in $[1, \bar{2}, 1]$ and $[2, \bar{3}, 2]$, the update parameters matrix $k_l$ in Eq. (4) will increase by adding one more term after computation. The method of computation is in Appendix. So we set $\eta = 1/2$ to decrease the learning rate a bit. In Fig. 7(b), the solid lines in blue and red cannot reach the maximum of the cost function. Compared the solid and dashed lines in the same color, the rate of convergence of the solid lines is not always higher than the one of dashed lines. These results give agreement with the theoretical analysis in the last paragraph of Section 4.1. Therefore we will not use residual block structure to the last output layer for Res-HQCNN in the following.

5.2. Big networks

In this subsection, we consider Res-HQCNN with deeper layers to test the advantages of deep residual learning. We firstly select Res-HQCNN $[2, 3, \bar{3}, 2]$ and $[2, \bar{3}, \bar{4}, \bar{2}]$ for 10 training pairs. The corresponding simulation results of them are presented in Figs. 8 and 9.

In Fig. 8(a), we test all possible four cases for QNNs $[2, 3, 3, 2]$ with residual block structure. Since $[2, 3, \bar{3}, 2]$ has two residual block structures, the parameters matrix $k_{l}^{\prime}$ for $[2, \bar{3}, 3, 2]$ would be larger than the ones for $[2, \bar{3}, \bar{3}, 2], [2, \bar{3}, \bar{3}, 2]$ and $[2, \bar{3}, \bar{3}, 2]$. Choosing $\eta = 1/5$ is suitable for $[2, \bar{3}, \bar{3}, 2], [2, \bar{3}, \bar{3}, 2]$ and $[2, \bar{3}, \bar{3}, 2], [2, \bar{3}, \bar{3}, 2]$...
To test the power of skipping connection, we plot the performance of QNNs with residual block structure for 10 training pairs and 600 training rounds, as shown in Fig. 8(a). From this figure, we can see that all possible four cases for QNNs with residual block structure perform well and the case with no skipping connection performs best. Therefore, we find that the skipping connection here is helpful in improving the performance of cost function.

In order to test the power of skipping connection, we plot the performance of QNNs with residual block structure for 10 training pairs and 600 training rounds, as shown in Fig. 8(a). From this figure, we can see that all possible four cases for QNNs with residual block structure perform well and the case with no skipping connection performs best. Therefore, we find that the skipping connection here is helpful in improving the performance of cost function.

We then try to change QNNs with residual block structure to Res-HQCNN with all possible cases, as shown in Fig. 9(a). Analogous to the results in Fig. 8(a), the blue line is above the other lines. The red and green lines are still unstable due to a smaller learning rate. We still test the skipping connection for Res-HQCNN with 10 training pairs and 600 training rounds, as shown in Fig. 9(a). We find that the skipping connection here is helpful in improving the performance of cost function.

5.3. Generalization: the robustness to noisy data

In this subsection, we begin examining the robustness of Res-HQCNN to noisy quantum data. For comparison conveniently, we consider the best case of Res-HQCNN with residual block structure in different cases for 10 training pairs and 600 training rounds.
Fig. 10. Numerical results of \([2, \tilde{3}, \tilde{3}, 2]\) and \([2, \tilde{3}, \tilde{3}, \tilde{3}, 2]\) with 5 training pairs for 1000 training rounds and 2500 training rounds, respectively.

(a) \(\eta = 1/15\) and \(\epsilon = 0.1\)  
(b) \(\eta = 1/35\) and \(\epsilon = 0.1\)

Fig. 11. Behaviors of \([2, \tilde{3}, \tilde{3}, 2]\) and \([2, \tilde{3}, 4, 5, 2]\) to noisy training data. Here \(\eta = 1/1.8\) and \(\epsilon = 0.1\), which is the same setting in Fig. 7(a). The step-sizes between two adjacent dots in Fig. 11(a) and Fig. 11(b) are 3 and 5, respectively. We also plot the variance of cost function between the green dots and red dots on the right. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

(a) 30 training pairs and 50 training rounds

(b) 100 training pairs and 200 training rounds
employ the same rule in Beer et al. (2020) to test the robustness. We firstly generate $N$ good training pairs $\langle |\phi^{in}\rangle, V|\phi^{in}\rangle \rangle$ and then destroy $n$ of them by replacing them with noisy training pairs. Every time the replaced subset is chosen randomly. The cost function is assessed for all good test pairs.

We choose Res-HQCNN $[2, \tilde{3}, 2]$ with $\eta = 1/1.8$ and $\epsilon = 0.1$ as an example and their behaviors are presented in Fig. 11. In the left sub-figures of Fig. 11, the green dots are the results of Res-HQCNN $[2, 3, 2]$ and the red ones are the corresponding results in Beer et al. (2020). And in the right sub-figures of Fig. 11, we also plot the variation of the cost function between the green dots and the red ones. The x-axis of Fig. 11 represents how many good training pairs are replaced by noisy pairs.

When the number of the training rounds and training pairs are small, such as 50 training rounds and 30 training pairs in Fig. 11(a), the values of cost for $[2, 3, 2]$ and $[2, 3, 2]$ both decrease as the number of noisy pairs increase and the variation of the cost value is always positive. This shows the superiority of $[2, 3, 2]$ for noisy training data with small training rounds and small training pairs.

When the number of training rounds and training pairs are big, such as 200 training rounds and 100 training pairs in Fig. 11(b), we find that if the number of noisy pairs are small, such as less than 35, Res-HQCNN and QNNs in Beer et al. (2020) both have strong robustness to noisy quantum data. As the number of noisy pairs continues to increase, the values of cost for green dots and red dots both begin to decrease.

When the number of noisy pairs exceeds 60, the variation of cost becomes increasing, reaching maximum value when the number of noisy pairs are 70. There are three unstable points (55, $-0.0115$), (90, $-0.0161$) and (100, $-0.0012$) that the variation is negative. As we can see, there are 21 pairs of green dots and red dots in Fig. 11. Then the training data are generated 21 times. For each time, the good training data $\langle |\phi^{in}\rangle, V|\phi^{in}\rangle \rangle$ and noisy training data $\langle |\phi^{in}\rangle, |\theta^{out}\rangle \rangle$ are generated randomly. The elements of $|\phi^{in}\rangle$ and $|\theta^{out}\rangle$ are randomly picked out of a normal distribution before normalization. So we think the randomness of training data causes some unstable points, which can be seen as comparable results. As a whole, Res-HQCNN $[2, 3, 2]$ shows its stronger robustness to noisy data than QNNs $[2, 3, 2]$.

We also detect deeper network $[2, \tilde{3}, \tilde{4}, 2]$ to noisy data, see Fig. 12. When the number of training rounds and training pairs are small, such as 150 training rounds and 30 training pairs in
Fig. 12(a), we find that the improvement is obvious. The variances of cost function are always positive. Roughly speaking, the variation decreases with the number of noisy pairs increasing. When the number of training rounds and training pairs is big, such as 600 training rounds and 100 training pairs in Fig. 12(b), it is pleased to find that all variances of cost function are always positive without unstable points. The maximum value of variance is more than 0.35, while the one in Fig. 11(b) is no more than 0.1. So for noisy data, deeper [2, 3, 4, 2] shows better improvement than [2, 3, 2].

Up to now, we have gone through the experiments for Res-HQCNN with or without noisy and obtained the improvement of performance for cost function compared with the QNNs in Beer et al. (2020). Although we do not show the results for Res-HQCNN with four or more hidden layers, we conjecture that deeper Res-HQCNN would bring better improvement of performance for cost function due to the mechanism of its training algorithm.

6. Conclusion and discussion

In this paper, we have developed a hybrid quantum-classical neural network with deep residual learning to improve the performance of cost function for deeper networks. A new residual block structure in quantum concept has been designed based on QNNs in Beer et al. (2020). We have presented how to connect residual block structure with QNNs. The corresponding training algorithm of Res-HQCNN has also been given for different cases. From the perspective of propagating information, the residual block structure allows information to propagate from input layer to any deeper layer, which is similar to the mechanism of ANNs with deep residual learning. The simulations have illustrated the power of Res-HQCNN at the cost of only running on classical computer.

There is another method to design residual block structure in quantum concept:

\[ \rho(\lambda) = \rho_{\text{in}} \ast (\rho_{\text{out}} \otimes |0 \cdots 0\rangle\langle 0 \cdots 0|) + (1 - p) \ast \rho_{\text{out}}. \]

It is a convex combination of \( \rho_{\text{in}} \) and \( \rho_{\text{out}} \). \( \Delta \rho = \rho_{\text{in}} - \rho_{\text{out}} \). Such an operation is possible to implement on a quantum computer. A quantum device would choose the one state with some probability \( p \) and the other with \( 1 - p \) for \( 0 \leq p \leq 1 \). For convenience, we call quantum neural networks with this kind of deep residual learning as p-ResQNN. The training algorithm of p-ResQNN is also changed. For example, if p-ResQNN has one hidden layer, then \( \rho(\lambda) = \rho_{\text{in}} \ast (\rho_{\text{out}} \otimes |0 \cdots 0\rangle\langle 0 \cdots 0|) + (1 - p) \ast \rho_{\text{out}}. \]

We choose QNNs [1, 2, 1] and [2, 3, 2] to have a test. When \( p = 0 \), it is just the case of former QNNs. Randomly choose \( p = 0.3, p = 0.6, p = 0.9 \), and \( p = 0.1 \). From Fig. 13, we find that only the result of \( p = 1 \), i.e. \( \rho(\lambda) = \rho_{\text{in}} \ast (\rho_{\text{out}} \otimes |0 \cdots 0\rangle\langle 0 \cdots 0|) + (1 - p) \ast \rho_{\text{out}}. \)
This paper is mainly focus on how to combine deep residual learning with QNNs to improve the ability of QNNs learning an unknown unitary transform with or without noisy. The readers can learn the advantages and disadvantages from our analysis of the model Res-HQCNN. This is just the beginning of combining quantum neural network with residual block structure. Related work is none. We hope our paper could be a useful reference in this area, which is exactly the point of this paper.

One future investigation of this field could be achieved by processing classical data, such as images and sounds. A suitable data encoding rule between quantum states and real world data should be designed. At present, scholars have carried out many methods on quantum image representation (Yan et al., 2017; Yan et al., 2016; Yao et al., 2017). Due to the combination of quantum computing and deep residual learning, we believe the exploration of Res-HQCNN with real world data will be interesting on some computer vision tasks. Another possible future research is to connect and compare the current results with quantum process tomography (Mohseni et al., 2008; O’Brien et al., 2004). Since the main goal in the whole field of quantum process tomography is to reconstruct quantum processes. Finding out the connection and difference between Res-HQCNN and quantum process tomography may be helpful in characterization of quantum dynamical systems.

Acknowledgments

This work is supported by the Academy of Finland for ICT 2023 project (grant 328115), and Ministry of Education and Culture of Finland for AI forum project. This work is also supported by the Key Research and Development Project of Guangdong Province, China under Grant No. 2020B0303300001, the Guangdong Basic and Applied Basic Research Foundation, China under Grant No. 2020B1515310016. This work is also supported by the Chinese Scholarship Council.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix

Here we present the training algorithm of a Res-HQCNN with one hidden layer in Fig. 5, which can be denoted as [2, 3, 2]. In this training algorithm, we also show the method to calculate the update parameters matrix $K_j(s)$ in detail.

I. Initialize:

I1. Set step $s = 0$.

I2. Choose all unitary $U_j(0)$ and $U_q(0)$ randomly, $j = 1, 2, 3$.

II. For each element $(|\phi_{in}^{\text{im}}, |\phi_{out}^{\text{out}}\rangle)$ from the set of training data, do the following steps:

II1. Feedforward:

II1a. Tensor the input state $\rho_{\text{in}}^{\text{in}} = |\phi_{\text{in}}^{\text{im}}\rangle\langle\phi_{\text{in}}^{\text{im}}| \otimes |000\rangle_{\text{hid}} (000)$.

II1b. Apply the layer unitary $U_1(s) = U_1 U_2 U_1^\dagger(s)$.

II2. Compute the cost function:

$$C(s) = \frac{1}{2N} \sum_{x=1}^{N} (\langle\rho_{\text{out}}^{\text{out}} | \rho_{\text{out}}^{\text{out}}(s)\rangle_{\text{hid}} |\phi_{\text{out}}^{\text{out}}\rangle)$$

II3. Repeat Step II.1 for $\rho_{\text{out}}^{\text{out}}$, and get the output of layer with $l = 2$.

II4. Update $s = s + \epsilon$.

IV. Repeat Step II and Step III until reaching the maximum of the cost function.

In the following, we will derive a formula for $K_j(s)$ to update the perceptron unitaries $U_j(s)$ with $l = 1, 2$. We assume the unitaries can always act on its current layer, such as $U_j(s)$ is actually $U_j(0) \otimes |000\rangle\langle000|$ in Res-HQCNN [2, 3, 2]. Inspired by the method from Beer et al. (2020), we present a different way to calculate $K_j(s)$ with the residual block structure.

We begin with a mathematical derivation of derivative function for the cost function: $\frac{d}{ds} = \lim_{\epsilon \to 0} \frac{C(s + \epsilon) - C(s)}{\epsilon}$, which leads us to find an analytical expression of $C(s + \epsilon)$. According to the definition of $C(s + \epsilon)$, we focus on the output state of the updated unitary $U_j(s + \epsilon)$.

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

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

Firstly, we consider the output state of the updated unitary $U_j(s + \epsilon)$ in the hidden layer. For convenience, we omit to write the parameter $s$ in $K_j(s)$ and $U_j(s)$ in the following with $j = 1, 2, 3$. Then using Eq. (A.1), we have

$$\rho_{\text{out}}^{\text{out}}(s + \epsilon) = \text{tr}_{\text{hid}}(e^{i K_j U_j e^{i K_j U_j e^{i K_j U_j}}} \langle\phi_{\text{in}}^{\text{im}}| \otimes |000\rangle_{\text{hid}} (000))$$

$$= \rho_{\text{out}}^{\text{out}}(s) + i e \text{tr}_{\text{hid}}(U_j U_j U_j (\phi_{\text{in}}^{\text{im}})\langle\phi_{\text{in}}^{\text{im}}) \otimes |000\rangle_{\text{hid}} (000))$$

$$+ U_j U_j U_j (\phi_{\text{in}}^{\text{im}})\langle\phi_{\text{in}}^{\text{im}}) \otimes |000\rangle_{\text{hid}} (000) + U_j U_j U_j (\phi_{\text{in}}^{\text{im}})\langle\phi_{\text{in}}^{\text{im}}) \otimes |000\rangle_{\text{hid}} (000)$$

$$+ U_j U_j U_j (\phi_{\text{in}}^{\text{im}})\langle\phi_{\text{in}}^{\text{im}}) \otimes |000\rangle_{\text{hid}} (000) + U_j U_j U_j (\phi_{\text{in}}^{\text{im}})\langle\phi_{\text{in}}^{\text{im}}) \otimes |000\rangle_{\text{hid}} (000)$$

$$+ U_j U_j U_j (\phi_{\text{in}}^{\text{im}})\langle\phi_{\text{in}}^{\text{im}}) \otimes |000\rangle_{\text{hid}} (000) + U_j U_j U_j (\phi_{\text{in}}^{\text{im}})\langle\phi_{\text{in}}^{\text{im}}) \otimes |000\rangle_{\text{hid}} (000)$$

$$+ U_j U_j U_j (\phi_{\text{in}}^{\text{im}})\langle\phi_{\text{in}}^{\text{im}}) \otimes |000\rangle_{\text{hid}} (000) + U_j U_j U_j (\phi_{\text{in}}^{\text{im}})\langle\phi_{\text{in}}^{\text{im}}) \otimes |000\rangle_{\text{hid}} (000).$$
\[
\rho_n^{\text{out}}(s + \epsilon) = \mathfrak{tr}\sum_{\alpha} U_{\text{hid}}^{\text{out}}(00)\rho_n^{\text{out}}(00)|\langle 00 | e^{-ik_2^\text{out}}U_2^\text{out} e^{ik_2^\text{out}}|00 \rangle |^2 + o(\epsilon^2) + R(\epsilon) + o(\epsilon^2).
\]

Based on the new input state for layer with \(l = 2\), we then go on computing the updated final output state of Res-HQCNN [2, 3, 2] in terms of \(U^2(\delta + \epsilon)\) in Eq. (A.2),

\[
\rho_n^\text{out}(s + \epsilon) = \mathfrak{tr}\sum_{\alpha} U_{\text{hid}}^{\text{out}}(00)\rho_n^\text{out}(00)|\langle 00 | e^{-ik_2^\text{out}}U_2^\text{out} e^{ik_2^\text{out}}|00 \rangle |^2 + o(\epsilon^2) + R(\epsilon) + o(\epsilon^2).
\]

in which

\[
A_1 = \mathfrak{tr}\sum_{\alpha} U_{\text{hid}}^{\text{out}}(00)\rho_n^\text{out}(00)|\langle 00 | e^{-ik_2^\text{out}}U_2^\text{out} e^{ik_2^\text{out}}|00 \rangle |^2 + o(\epsilon^2) + R(\epsilon) + o(\epsilon^2).
\]

So the mathematical derivative function of the cost function can be calculated as

\[
\frac{\partial C}{\partial s} = \lim_{\epsilon \to 0} \frac{C(s + \epsilon) - C(s)}{\epsilon} = \frac{1}{N} \sum_{i=1}^{N} \mathfrak{tr}[\rho_n^\text{out}(s + \epsilon)|\phi_n^\text{out}|(s + \epsilon)](|\langle k_2^\text{out} | U_2^\text{out} | 00 \rangle |^2 + o(\epsilon^2) + R(\epsilon) + o(\epsilon^2)).
\]

\[
K_j^1(s) = \sum_{\alpha_1, \alpha_2, \beta} K_j^{\alpha_1 \alpha_2 \beta}(s)(\sigma^{\alpha_1} \otimes \sigma^{\alpha_2} \otimes \sigma^{\beta}),
\]

where \(\sigma\) is Pauli matrix in single qubit. \(\alpha_1\) and \(\alpha_2\) represent the qubits in the input layer and \(\beta\) represents the current qubit of unitary \(U^n\) in the hidden layer. We choose to use of a Lagrange multiplier \(\lambda\), which is a real number, to find a finite solution. When \(j = 1\), the analysis above leads us to solve a maximization.
After going through the similar process for $k_{1, a_1, a_2, \beta}$, it represents the current qubit of unitary $K_1$.

Reference

Alvarez-Rodriguez, U., Lamata, L., Escandell-Montero, P., Martin-Guerrero, J. D., & Solano, E. (2017). Supervised quantum learning without measurements. Scientific Reports, 7(1), 1–6.

Amato, F., López, A., Peña-Méndez, E. M., Vañhara, P., Hampel, A., & Havel, J. (2013). Artificial neural networks in medical diagnosis.

Beer, K., Bondarenko, D., Farrell, T., Osborne, T. J., Salzmann, R., Schiermann, D., & Wolf, R. (2020). Training deep quantum neural networks. Nature Communications, 11(1), 1–6.

Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D., & Bengio, Y. (2010). Theano: A CPU and GPU math compiler in Python. In Proc. 9th Python in science conf (vol. 1) (pp. 3–10).

Biamonte, J., Wittek, P., Pancotti, N., Rebentrost, P., Wiebe, N., & Lloyd, S. (2017). Quantum machine learning. Nature, 549(7671), 195–202.

Bishop, C. M. (1995). Neural networks for pattern recognition. Oxford University Press.

Bisio, A., Chiribella, G., D’Ariano, G. M., Facchini, S., & Perinotti, P. (2010). Optimal quantum learning of a unitary transformation. Physical Review A, 81(3), Article 032324.

Chen, R. T., Rubanova, Y., Bettencourt, J., & Duvenaud, D. K. (2018). Neural ordinary differential equations. Advances in Neural Information Processing Systems, 31, 6571–6583.

Chen, H., Wossning, L., Severini, S., Neven, H., & Mohseni, M. (2021). Universal discriminative quantum neural networks. Quantum Machine Intelligence, 3(1), 11–11.

Chehrler, S., Woolley, C., Vandersurch, P., Cohen, J., Tran, J., Catanzaro, B., & Shelhamer, E. (2014). Cudnn: Efficient primitives for deep learning. arXiv preprint arXiv:1410.0759.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition (pp. 248–255). Ieee.

Dong, C., Liu, L., Li, Z., & Shang, J. (2020). Towards adaptive residual network training: A neural-ode perspective. In International conference on machine learning (pp. 2616–2626). PMLR.

Dunjko, V., & Briegel, H. J. (2018). Machine learning & artificial intelligence in the quantum domain: a review of recent progress. Progress on in Physics, 81(7), Article 074001.

Dunjko, V., Taylor, J. M., & Briegel, H. J. (2016). Quantum-enhanced machine learning. Physical Review Letters, 117(13), Article 130501.

Ezhov, A. A., & Ventura, D. (2000). Quantum neural networks. In Future directions for intelligent systems and information sciences (pp. 213–235). Springer.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770–778).

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Identity mappings in deep residual networks. In European conference on computer vision (pp. 630–645). Springer.

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., & Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. In Proceedings of the 22nd ACM international conference on multimedia (pp. 675–678).

Jouppi, N. P., Young, C., Patil, N., Patterson, D., Agraval, G., Bajwa, R., Bates, S., Bhatia, S., Boden, N., & Borchers, A. (2017). In-datacenter performance analysis of a tensor processing unit. In Proceedings of the 44th annual international symposium on computer architecture (pp. 1–12).

Korpi, D., Honkalä, M., Huttunen, J. M., & Starck, V. (2020). DeepXPS: MIMO: Convolutional MIMO detection with learned multiplicative transformations. arXiv preprint arXiv:2010.16283.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. Communications of the ACM, 60(6), 84–90.

Li, P., & Wang, B. (2020). Quantum neural network models based on swap test and phase estimation. Neural Networks, 130, 152–164.

Maclaurin, D., Duvenaud, D., & Adams, R. P. (2015). Autograd: Effortless gradients in numpy. In ICM 2015 AuToM workshop (vol. 238) (p. 5).

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. The Bulletin of Mathematical Biophysics, 5(4), 115–133.

Mitchell, T. M., & Thrun, S. B. (1993). Explanation-based neural network learning for robot control. In Advances in neural information processing systems (pp. 287–294).

Mohseni, M., Rezakhani, A., & Lidar, D. (2008). Quantum-process tomography: Resource analysis of different strategies. Physical Review A, 77(3), Article 032322.

Nielsen, M. A. (2015). Neural networks and deep learning (vol. 2018). Determination Press San Francisco, CA.

Nielsen, M. A., & Chuang, I. (2002). Quantum computation and quantum information.

Nishani, E., & Çiço, B. (2017). Computer vision approaches based on deep learning and neural networks: Resource analysis of different strategies. Physical Review A, 77(3), Article 032322.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., & Lerer, A. (2017). Automatic differentiation in pytorch. In NIPS workshop.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., & Antiga, L. (2019). Pytorch: An imperative style, high-performance deep learning library. In Advances in neural information processing systems (pp. 8026–8037).
Patterson, A., Chen, H., Wossnig, L., Severini, S., Browne, D., & Rungger, I. (2021). Quantum state discrimination using noisy quantum neural networks. *Physical Review Research, 3*(1), Article 013063.

Peng, W., Hong, X., Chen, H., & Zhao, G. (2020). Learning graph convolutional network for skeleton-based human action recognition by neural searching. *AAAI*.

Peng, W., Shi, J., Xia, Z., & Zhao, G. (2020). Mix dimension in poincaré geometry for 3D skeleton-based action recognition. *ACM Multimedia*.

Pham, H., Guan, M. Y., Zoph, B., Le, Q. V., & Dean, J. (2018). Efficient neural architecture search via parameter sharing. *arXiv preprint arXiv:1802.03268*.

Purushothaman, G., & Karayiannis, N. B. (1997). Quantum neural networks (QNNs): inherently fuzzy feedforward neural networks. *IEEE Transactions on Neural Networks, 8*(3), 679–693.

Rønnow, T. F., Wang, Z., Job, J., Boixo, S., Isakov, S. V., Wecker, D., Martinis, J. M., Lidar, D. A., & Troyer, M. (2014). Defining and detecting quantum speedup. *Science, 345*(6195), 420–424.

Sasaki, M., & Carlini, A. (2002). Quantum learning and universal quantum matching machine. *Physical Review A, 66*(2), Article 022303.

Schuld, M., Sinayskiy, I., & Petruccione, F. (2014). The quest for a quantum neural network. *Quantum Information Processing, 13*(11), 2567–2586.

Sedláček, M., Bisio, A., & Ziman, M. (2019). Optimal probabilistic storage and retrieval of unitary channels. *Physical Review Letters, 122*(17), Article 170502.

Sentís, G., Monràs, A., Muñoz-Tapia, R., Calsamiglia, J., & Bagan, E. (2019). Unsupervised classification of quantum data. *Physical Review X, 9*(4), Article 041029.

Skolik, A., McClean, J. R., Mohseni, M., van der Smagt, P., & Leib, M. (2021). Layerwise learning for quantum neural networks. *Quantum Machine Intelligence, 3*(1), 1–11.

Verdon, G., Pye, J., & Broughton, M. (2018). A universal training algorithm for quantum deep learning. *arXiv preprint arXiv:1806.09729*.

Xie, S., Girshick, R., Dollár, P., Tu, Z., & He, K. (2017). Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1492–1500).

Yan, F., Iliyasu, A. M., & Le, P. Q. (2017). Quantum image processing: a review of advances in its security technologies. *International Journal of Quantum Information, 15*(03), Article 1730001.

Yan, F., Iliyasu, A. M., & Venegas-Andraca, S. E. (2016). A survey of quantum image representations. *Quantum Information Processing, 15*(1), 1–35. [http://dx.doi.org/10.1007/s11128-015-1195-6](http://dx.doi.org/10.1007/s11128-015-1195-6).

Yan, S., Xiong, Y., & Lin, D. (2018). Spatial temporal graph convolutional networks for skeleton-based action recognition. *arXiv preprint arXiv:1801.07455*.

Yao, X.-W., Wang, H., Liao, Z., Chen, M.-C., Pan, J., Li, J., Zhang, K., Lin, X., Wang, Z., & Luo, Z. (2017). Quantum image processing and its application to edge detection: theory and experiment. *Physical Review X, 7*(1), Article 031041.

Zhao, J., Zhang, Y.-H., Shao, C.-P., Wu, Y.-C., Guo, G.-C., & Guo, G.-P. (2019). Building quantum neural networks based on a swap test. *Physical Review A, 100*(1), Article 012334.