electronics

MDPI

Article

An Evaluation of Hardware-Efficient Quantum Neural Networks for Image Data Classification

Tuyen Nguyen, Incheon Paik, Yutaka Watanobe and Truong Cong Thang *

Department of Computer Science and Engineering, The University of Aizu, Aizuwakamatsu 965-8580, Japan; s1262008@u-aizu.ac.jp (T.N.); paikic@u-aizu.ac.jp (I.P.); yutaka@u-aizu.ac.jp (Y.W.)
* Correspondence: thang@u-aizu.ac.jp

Abstract: Quantum computing is expected to fundamentally change computer systems in the future. Recently, a new research topic of quantum computing is the hybrid quantum–classical approach for machine learning, in which a parameterized quantum circuit, also called quantum neural network (QNN), is optimized by a classical computer. This hybrid approach can have the benefits of both quantum computing and classical machine learning methods. In this early stage, it is of crucial importance to understand the new characteristics of quantum neural networks for different machine learning tasks. In this paper, we will study quantum neural networks for the task of classifying images, which are high-dimensional spatial data. In contrast to previous evaluations of low-dimensional or scalar data, we will investigate the impacts of practical encoding types, circuit depth, bias term, and readout on classification performance on the popular MNIST image dataset. Various interesting findings on learning behaviors of different QNNs are obtained through experimental results. To the best of our knowledge, this is the first work that considers various QNN aspects for image data.

Keywords: quantum machine learning; parameterized quantum circuit; quantum neural network; image classification

Citation: Nguyen, T.; Paik, I.; Watanobe, Y.; Thang, T.C. An Evaluation of Hardware-Efficient Quantum Neural Networks for Image Data Classification. *Electronics* **2022**, *11*, 437. https://doi.org/10.3390/electronics11030437

Academic Editor: Sang Wook Han

Received: 30 November 2021
Accepted: 28 January 2022
Published: 1 February 2022

1. Introduction

Quantum computing is a novel type of computing that is expected to fundamentally change future computer systems. Because quantum mechanics is a more general model of physics than classical mechanics, it leads to a more general model of computing, which has the ability to tackle problems that classical computing cannot [1]. Unlike classical computers, which utilize the binary bits 0 and 1 to store or manipulate information, quantum computers use quantum bits, also known as "*qubits*". Based on the quantum properties of "*superposition*" and "*entanglement*", n qubits can act as a group rather than exist in isolation, thus resulting in exponentially greater information density than a classical computer. Quantum computing's applications are virtually limitless, such as artificial intelligence, chemical simulation, molecular modeling, cryptography, optimization, and financial modeling [1–3]. Although quantum computers offer a considerable performance advantage, system fidelity is still an issue. In practice, qubits are extremely sensitive to environmental disturbances, making them prone to errors [1]. To cope with this problem, the practical applications of noisy intermediate scale quantum (NISQ) devices are being investigated [2].

Recently, hybrid quantum–classical approaches for machine learning have been proposed to support NISQ devices [3,4]. A hybrid quantum–classical system can benefit from both quantum computing methods and classical machine learning methods. In this trend, parameterized quantum circuits (PQCs) have been introduced for ease of circuit control. In fact, PQCs are considered as the counter part of classical neural networks (NNs), so often called quantum neural networks (QNNs) [5]. In a hybrid quantum–classical system, the parameters of a PQC are optimized by popular classical routines (e.g., gradient-based

optimization). This approach has been successfully applied in various practical applications [5,6]. Furthermore, the potential of QNN with respect to comparable classical NN has recently been proved in [7].

Currently, the number of qubits supported by existing quantum computers is still limited. However, due to the enormous potential of quantum computing in general [1] and QNN in particular [7], many studies have been carried out to investigate the different structures and characteristics of PQCs for machine learning tasks. Furthermore, to facilitate the design of hybrid quantum–classical methods, several simulation tools have been developed [8–10].

As pointed out in [11], there is still a lack of understanding regarding PQCs for machine learning. Recently, some studies have evaluated how different factors of PQCs affect classification performance [12,13]. Specifically, various quantum circuits with different complexities are compared in [12], where it is found that circuits of higher complexities generally have better classification performances. However, different encoding types are not considered in [12]. The evaluation in [13], which is closest to our work, compares two typical data encoding types, namely qubit encoding and amplitude encoding. It is shown is that amplitude encoding results in the best performance [13]. However, only one quantum circuit is used in the comparison. Moreover, it should be noted that the datasets in these evaluations are just hand-crafted 2-dimensional data.

Some previous studies have shown that simple (and even random) QNNs can be used for image classification [14–16]. Furthermore, QNNs are being attempted within classical deep neural network architectures. For example, QNNs can be used to replace fully connected layers [17] or convolution operations in CNNs [18–20]. However, as discussed in detail in the next section, the impacts of various factors on the learning behaviors and performances of QNNs have not been extensively studied. In this paper, we will evaluate both different encoding types and circuits, using a practical dataset. More specifically, the novel aspects of our evaluation are as follows:

- We employed a practical dataset of image contents, namely the popular Modified National Institute of Standards and Technology (MNIST) dataset;
- The performances of encoding types are investigated with hardware-efficient quantum circuits of increasing complexities;
- We consider not only shallow circuits having lower circuit depths, but also a deep (and complex) QNN which is proposed recently [21];
- No layers of classical neurons are included in the considered neural networks. This helps to see the actual performances of quantum circuits;
- Rather than simply comparing the accuracy values, we will investigate the learning curves over a large number of epochs, which further reveal the behaviors of the circuits;
- In addition, we also study the impacts of some parameters such as the number of layers, bias terms, readout with ancilla qubit.

To the best of our knowledge, these multiple aspects have not been studied in previous works. It should be noted that, rather than directly using a complex quantum circuit, we will start from a basic and practical circuit and gradually increase its complexity. Interestingly, this evaluation strategy helps discover the specific behaviors of different data encoding types when the number of circuit layers is increased.

The contributions of this paper include various new findings about the impacts of number of qubits, circuit depth, encoding types, bias terms, and readout on image classification performance—especially considering that some settings result in no barren plateaus. However, there are possibly multiple barren plateaus on the learning curves of certain QNNs. This could result in low performance when the training process is terminated early during the first plateau. As previously mentioned, our focus in this study was to investigate the learning characteristics of QNNs with image data. The comparison with state-of-art CNNs is not considered because the number of qubits of current quantum computers or simulation tools is very small compared to the modern classical computers.

The rest of this paper is organized as follows. In Section 2, the related work will be discussed. The background of quantum computing for machine learning is presented in Section 3. In Section 4, we will describe the settings of the evaluation. The evaluation results and discussions will be provided in Section 5. Finally, conclusions and future work are given in Section 6.

2. Related Work

Quantum machine learning has been investigated with different types of data or signals such as analytical functions and scalar classes [22]. Ref. [14] was one of the earliest studies to employ quantum circuits to classify images. At the input of a circuit, each pixel value of an image is encoded into a qubit (i.e., using qubit encoding). The circuit structure can be random, and one ancilla qubit of the circuit is reserved for the output readout [14]. This circuit is implemented as an example method in TensorFlow Quantum [8]. Grant et al. [15] also classified images, by hierarchical circuits rather than random circuits. However, the pixel values are not directly encoded into qubits. Instead, various features, e.g., some of the most significant components based on principal component analysis (PCA), are first extracted from each image. The features are then encoded (by qubit encoding) into the qubits. In Ref. [16], the classification of CT scan images is carried out with only the baseline QNN provided by Tensorflow Quantum. Furthermore, the impacts of different factors on learning behaviors are not studied. Moreover, though this work compares with the accuracy values reported in previous studies of classical deep neural networks, the large CT scan images are downscaled to the 4×4 resolution to fit the baseline QNN. This operation would destroy the fine details of complex CT scan images. In Ref. [17], a neural network architecture consisting of a classical deep network and a simple quantum neural network is employed. Here, the classical deep neural network is employed to extract features as in CNN. However, this combination cannot reveal the characteristics of pure QNNs.

In Ref. [23], a hardware-efficient (HE) heuristic approach is proposed to build quantum circuits which are suitable to the limitations of current quantum computers. The basic block of the approach is an element layer with single-qubit rotation and two-qubit entanglement operations. A quantum circuit is then composed of multiple element layers (illustrated by circuit depth in [23]). Interestingly, it is shown that such a circuit of a higher depth can result in better performance [23].

The above multi-layer approach is further extended to the so-called quantum deep neural network (denoted by QDNN) [21], which is similar to classical deep neural networks. Each layer of the QDNN is composed of three quantum parts, namely the encoder, transformation, and output. The encoder is a special PQC that embeds classical data into a quantum state [21]. Though this encoder makes use of entanglement among qubits, it is not of the amplitude encoding type. The transformation is another PQC that is linearly applied to the prepared quantum state. Here, each PQC in the encoder or transformation is based on the HE multi-layer structure of [23]. Therefore, the encoding type of QDNN will be called HE encoding in our evaluation. The output part of a QDNN layer obtains classical data outputs by using quantum measurements. Similarly to classical deep learning neural networks, a bias term is added to each output. Thus, essentially each layer of QDNN plays a role similar to a layer of a classical neural network.

In another approach, some studies have focused on constructing quantum artificial neurons [24–26]. Similarly to artificial neurons in classical neural networks, quantum artificial neurons are basic building blocks to systematically develop QNNs of highly complex architectures. This concept has been proved with implementation on a real quantum computer [27]. Furthermore, the complexity of quantum artificial neurons can be reduced by techniques such as shortcuts to adiabaticity [28]. However, this approach is still in its early stage and current QNNs based on quantum artificial neurons are very difficult to be trained [5], so our evaluation will only consider hardware-efficient QNNs.

Recently, there have been some evaluations of different aspects of QNNs [11–13]. In a theoretical study of Sim et al. [11], the goodness of QNNs is investigated by some computational measures of expressibility and entangling capability. The authors show that, in general, these theoretical measures increase when a circuit has more layers. As experiments with real quantum computers are very burdensome and expensive, some simulation tools have been developed to design and evaluate quantum circuits [8–10]. Using TensorFlow Quantum [8], the authors in [13] focused on the comparison of two kinds of encodings, namely qubit encoding and amplitude encoding. It is found that the amplitude encoding always provides better performances. However, the data in [13] were computer generated and the employed circuit is just a basic circuit of TensorFlow Quantum. In [12], a number of circuits with different complexities was compared using PennyLane library [9]. The classification performances of the circuits are evaluated, also with different hand-crafted datasets. In general, greater circuit complexity indicates improved classification performance [12]. However, neither different encoding types nor the learning behaviors of quantum circuits were studied in [12].

In addition, there were some studies that used quantum circuits to replace the convolution operation of classical deep learning networks [18–20]. In our evaluation of QNNs, we will not consider layers of classical neurons. The purpose of this choice was to see the actual behaviors and performances of quantum neural networks. As mentioned, our evaluation is unique in using high-dimensional spatial data, different encoding types, and circuits with increasing complexities. Moreover, rather than simply comparing the accuracy values, we will look into the learning behaviors of the QNNs. This provides interesting insights into the relationships between encoding types, circuit depths, and the number of training steps.

3. Background

This section presents a background of quantum machine learning, including a brief introduction of qubits, quantum states, quantum circuits, and hybrid quantum–classical systems.

3.1. Qubit and Quantum State

Quantum bit (or qubit) is a fundamental information unit used in quantum computers. It can be compared to a bit that is used in a classical computer. A qubit, in more technical terms, is a two-dimensional quantum system. The state of a qubit can be expressed as

$$|\phi\rangle = \alpha|0\rangle + \beta|1\rangle, \tag{1}$$

where α and β are complex numbers and $|\alpha|^2 + |\beta|^2 = 1$. In the *ket*$-$*notation* or the *Dirac notation*, $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ are used to represent the basis states of a two-dimensional vector space. Therefore, Equation (1) indeed shows the state of a qubit as the two-dimensional complex vector $\begin{pmatrix} \alpha \\ \beta \end{pmatrix}$. The difference with classical bit is that a qubit cannot be measured without changing it. Measuring a qubit, or its state specified by Equation (1), will produce the classical value of either zero ($|0\rangle$) with a probability of $|\alpha|^2$ or one ($|1\rangle$) with a probability of $|\beta|^2$.

In addition, $\langle\phi|$ is the conjugate transpose of $|\phi\rangle$ and is a row vector (known as *bra*) with two components: $\langle 0| = (1\ 0)$ and $\langle 1| = (0\ 1)$. From *bra* and *ket*, we can compute the inner product or outer product of the vectors. Given $|u\rangle$ and $|v\rangle$, their inner product is $\langle u|v\rangle$ $(= \langle u||v\rangle)$, which is a scalar. For example, $\langle 0|0\rangle = \langle 1|1\rangle = 1$ and $\langle 0|1\rangle = \langle 1|0\rangle = 0$. The outer product is obtained by $|u\rangle\langle v|$, which is an operator in the matrix form. If $|0\rangle\langle 0|$ is operated on $|\phi\rangle$, the result of $\alpha|0\rangle$ is obtained. This means that the operator $|0\rangle\langle 0|$ will extract the $|0\rangle$ component from $|\phi\rangle$ or $|\phi\rangle$ is measured in the $|0\rangle$ direction. Similarly, the operator $|1\rangle\langle 1|$ extracts the $|1\rangle$ component from $|\phi\rangle$.

3.2. Quantum Circuits

Quantum algorithms are frequently shown in the form of quantum circuits, consisting of quantum gates that process input qubits. In the circuit representation, qubits are represented by horizontal lines. Quantum gates or operators are then applied to the qubits. This is performed in a clockwise direction from left to right. A qubit's initial state is indicated at each of the qubit lines. The gates are shown in the sequence of their applications from left to right. The states of qubits in a quantum circuit change over time. Different quantum gate combinations can perform certain quantum algorithms, and the results are obtained through quantum measurements. Mathematically, unitary matrices are used to represent quantum gates and the number of qubits in a gate's input and output must be equal. To measure the qubits in the computational basis, a measurement gate is applied and denoted by a special gate with a meter symbol as shown in the figures of the following section.

3.3. Hybrid Quantum–Classical Systems

A hybrid quantum–classical system, as shown in Figure 1, generally consists of a quantum part and a classical part [5]. The first step of the quantum part is a (fixed) circuit for state preparation, which is called an encoder circuit. The two most popular encoding types are qubit encoding, where data are encoded into individual qubits, and amplitude encoding, where data are encoded into the amplitudes of an entangled set of qubits [13]. The most important step of the quantum part is a parameterized quantum circuit (PQC, also called a QNN) with parameters to be adjusted by the classical part. The outputs of the quantum part are classical values which are obtained by measurements. The classical part applies an optimization algorithm (e.g., gradient-based methods) to adjust the parameters of the quantum circuit. In addition, some pre-processing and/or post-processing of classical data may be carried out in the classical part. Thus, a hybrid quantum–classical system can take advantage of both quantum computing and classical processing/optimization techniques.

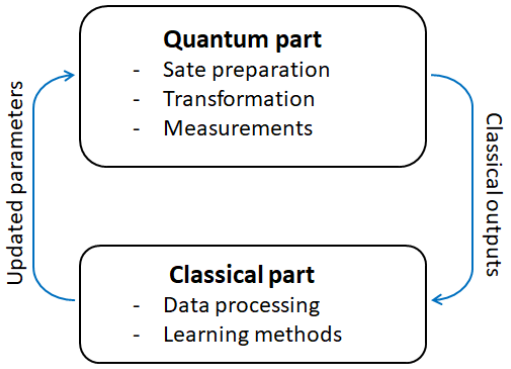


Figure 1. Illustration of a hybrid quantum–classical system.

4. Methods and Experimental Settings

In our evaluation, various QNNs (or quantum circuits) of different designs are employed. A set of circuits were built up from a basic block, which was proposed in the hardware-efficient circuit of [23]. In the following, the details of QNNs and their settings will be described.

4.1. Baseline QNN

The baseline QNN in our evaluation is the QNN proposed by Farhi et al. [14] and implemented as an example QNN in Tensorflow Quantum [8]. The implemented structure of this circuit is illustrated in Figure 2. This structure reserves an ancilla qubit for the purpose of output readout, which represents the probability of the true label. In each step, the readout qubit is entangled with one input qubit. The whole process is applied twice

before the output measurement is performed. Thus, the readout qubit acts as a collector that obtains information from other qubits in the circuit.

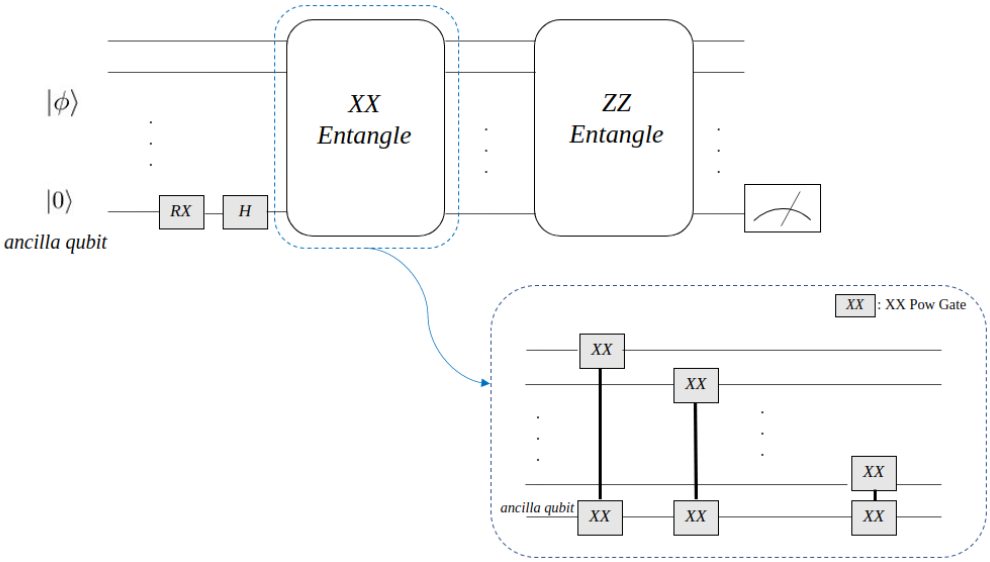


Figure 2. The design of baseline QNN [8].

4.2. Hardware-Efficient (HE) QNN

The QNNs of this type are based on the hardware-efficient circuits proposed by Kandala et al. [23]. An HE circuit has multiple element layers, each of which essentially consists of an entanglement block and single-qubit rotation operations as illustrated in Figure 3. A measurement gate is then used to calculate the expectation of the outcome pertaining to the two Hamiltonians' operators $\{M\}_{i=1}^{2}$ to form outputs for our classification problem. To determine the final probability value, we use a softmax activation function at the end. This output style is in fact similar to that of [22]. It is shown that the performance of an HE circuit would become higher if the number of element layers is increased [11,12,23]. As this type of HE circuit has a well-defined structure and has been widely applied in different scenarios, we will employ HE circuits to compare the impacts of encoding types and circuit depths.

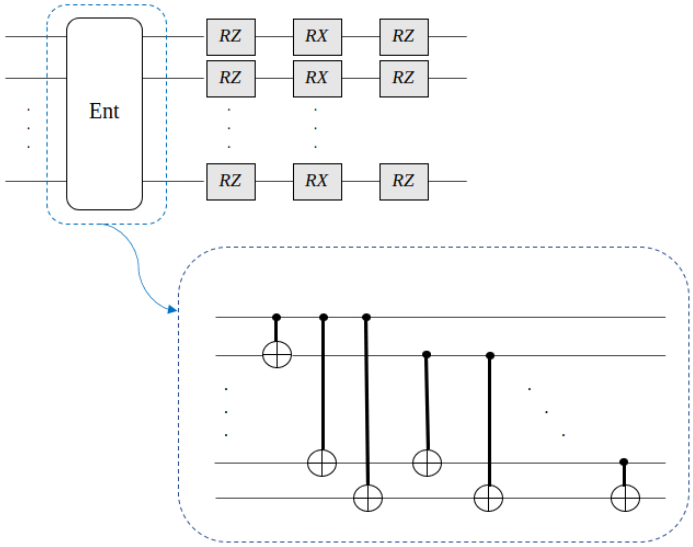


Figure 3. Element layer of an HE circuit [23].

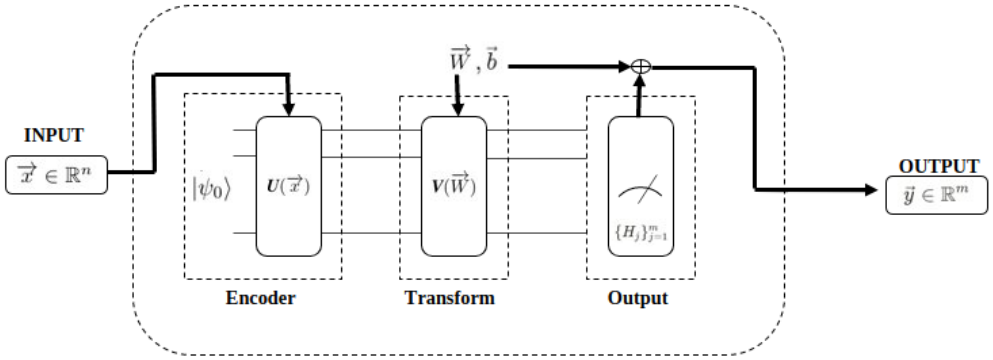


Figure 4. The architecture of a QDNN layer. Reprinted from ref. [21].

4.3. Deep QNN

For deep QNN, Zhao et al. [21] introduced the concept of QNN layer (or QNNL), which is itself a complete circuit consisting of three parts: the encoder, the transformation, and the output.

- The encoder: this part takes the input data $\vec{x} \in \mathbb{R}^n$ and encodes them using a PQC $U(\vec{x})$. Basically, this step modifies an initial quantum state $|\psi_0\rangle$ into $|\psi(\vec{x})\rangle$. The encoding process can be defined as

$$|\psi(x)\rangle = exp(-i\frac{x}{2}X)|\psi_0\rangle, \tag{2}$$

  where X is the Pauli matrix X.
- The transformation: this part applies a linear transformation to the input quantum state $|\psi_0\rangle$ using another PQC $V(\vec{W})$ with parameters $\vec{W}$;
- The output: Finally, the output will be computed by m Hamiltonians $H_j, j = 1, \ldots, m$ and output $\vec{y}$. Each output component will be added a bias term b_j as in classical deep neural networks:

$$y_j = \left\langle \psi(\vec{x})|V^{\dagger}(\overrightarrow{W})H_jV(\vec{W})|\psi(\vec{x})\right\rangle + b_j. \tag{3}$$

  Here, m is the same as the number of hidden dimensions of the model.

An encoder part or transformation part of a QNNL contains multiple element layers. The element layer of [23], which consists of an entanglement block and single-qubit rotation operations, as shown in Figure 4, is also used in this approach. As a QDNN may have multiple QNNLs, a QDNN actually contains a large number of element layers [21]. Between two QNNLs, there is a conversion between quantum state and classical state. Furthermore, one special design described in [21] is the way of encoding the input data. Specifically, to encode an 8×8 image, this encoder uses only eight input qubits. Eight 8-dimensional vectors of an image are then embedded by using two HE element layers. Therefore, as mentioned previously, we refer to this type of encoding as "HE encoding".

4.4. Implementation Details

All evaluated QNNs are implemented using the TensorFlow Quantum library [8]. Due to the limitations of the current simulation tool, the maximum number of input qubits in a circuit is 16. All our experiments were run with the popular MNIST dataset. For simplicity, we only used digits 0 and 1 to form a binary classification. There were 12,665 images for training and 2115 images for validation. Note that existing studies (e.g., [12,13,15,21]) also aim to perform binary classifications. Furthermore, the original image resolution of 28×28 will be downscaled to different resolutions depending on the number of qubits and encoding type of the evaluated circuits. We used the Adam optimizer with a learning rate of 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and binary cross-entropy as the loss function [29]. All

experiments were run in 200 epochs with a batch size of 240. The evaluation is carried out on a computer with Core i9-11900F CPU, 32 GB RAM, and RTX3090 GPU.

As discussed above, the possible encoding methods are qubit encoding, amplitude encoding, and HE encoding. However, as the complexity of state preparation for amplitude encoding is exponential in the number of qubits, it is not appropriate for images, which are high-dimensional data. In our preliminary experiments, the simulation with amplitude encoding either simply could not run or took an excessively long time (e.g., several days), which is impractical. Note that, in [13], the complexity is not a problem because the evaluated circuit has only two qubits. Therefore, in our evaluation, we will only explore qubit encoding and HE encoding.

In this study, in total of 13 QNNs were evaluated. As mentioned above, among these QNNs, the HE QNNs will be implemented with both qubit encoding and HE encoding. The detailed settings of the QNNs are as follows.

- The baseline QNN: data are input into the circuit using qubit encoding. This circuit has 16 qubits for input data. Images are thus downscaled to the 4×4 resolution, of which each pixel is encoded into one qubit. The output of the circuit is an ancilla qubit which is reserved for this purpose (Figure 2).
- The qubit-encoded HE QNNs: Similarly to the baseline QNN, 16 qubits are used to encode the input data. Input images are also downscaled to the 4×4 resolution. Narrower QNNs (i.e., using less than 16 qubits) are not considered because the corresponding resolutions of images would be too small. As an HE QNN is a sequence of element layers, in this paper, we will evaluate three circuit depths (or three numbers of element layers), namely 3, 5, and 8. These selected values are consistent with typical circuit depth values for low-depth QNNs, which are up to 5 in [11] and 8 in [23].
- The HE-encoded HE QNNs: In the case of HE encoding, the number of pixels can be n^2 where n is the number of qubits. Thus, we consider two options for the number of qubits, namely 8 qubits and 16 qubits, which correspond to the scaled resolutions of 8×8 and 16×16. Note that the options of more qubits are not included because the current simulation tool cannot support such complexity. For each number of qubits, there are three cases of circuit depth, namely 3, 5, and 8, which is similar to qubit-encoded HE QNNs. Thus, there are in total 6 HE-encoded HE QNNs.
- The QDNN and its variants: The implementation of this model is based on the original design described in [21], using the Yao.jl library [10]. For our evaluation, QDNN is re-implemented using Tensorflow Quantum. This QNN is investigated with three options. The first option is the original design, denoted by "QDNN (original)" in the results. In order to see the impact of bias terms, the second option is the variant without bias parameters, denoted by "QDNN (no bias)". Note that all the other QNNs above have no bias terms in their circuits. The third option is a modified QDNN that employs an ancilla qubit for readout, which is denoted by "QDNN (ancilla)". As in the case of the baseline QNN, the ancilla qubit is entangled with other qubits in the circuit at the end of each QNNL. This ancilla qubit is the only one that is used to compute the classification results. For these QNNs, images are scaled to the 8x8 resolution, which is the same as the original design of [21].

5. Evaluation Results

In this section, two evaluations will be carried out. In the first evaluation, we employed the HE QNNs to compare two encoding types. In the second one, the focus is on comparing QNNs with different designs and complexities, including the quantum deep neural network. To understand the learning behaviors and performances of the QNNs, we will present the learning curves in both training and validation.

5.1. Comparison of Encoding Types

In this subsection, we will explore how the performance of a QNN is affected by encoding choices and circuit depths. In the resulting tables and figures, qubit encoding and

HE encoding are, respectively, denoted by "Qu-enc" and "HE-enc". The accuracy values at the end of the training process are provided in Table 1 and depicted in Figure 5. For each combination of encoding and circuit depth, Figure 6a,b show the learning curves in training when the number of epochs is increased. The corresponding performances in validation are shown in Figure 7a,b.

Table 1. Training and validation accuracy of qubit-encoded and HE-encoded QNNs.

QNNs	Depth = 3		Depth = 5		Depth = 8	
	Train	Validation	Train	Validation	Train	Validation
Qu-enc HE (16 qubits)	0.83	0.836	0.84	0.853	0.85	0.856
HE-enc HE (8 qubits)	0.70	0.686	0.82	0.823	0.87	0.864
HE-enc HE (16 qubits)	0.705	0.697	0.69	0.679	0.76	0.769

From Table 1 and Figure 5, we can see that, when the circuit depth is increased, the performance generally becomes higher. However, with qubit encoding, the accuracy values corresponding to different circuit depths are very close, specifically 0.83, 0.84, and 0.85 for depths of 3, 5, and 8, respectively. Meanwhile, with HE encoding, the impact of circuit depth is varying, with accuracy values ranging widely from 0.70 to 0.87.

Furthermore, the learning curves in Figures 6 and 7 show that the behaviors of the circuits are different. Specifically, with qubit encoding, the accuracy is initially higher than that of HE encoding; however, it soon becomes saturated (Figure 6b or Figure 7b). Furthermore, only in the case of the circuit depth equal to 8, the loss curve is always decreasing, without reaching any plateaus. The circuit depths of 3 or 5 initially seem to saturate at low accuracy values (Figure 6b or Figure 7b). However, when the number of epochs increases, their performances jump to approach the accuracy of the circuit depth of 8. The barren plateau phenomenon in training QNNs has been pointed out in [30]. However, the behavior of the case of qubit encoding with the circuit depth of 3 or 5 shows that the performance may reach a plateau for many epochs and then jump to another better plateau. This is an interesting fact that will be further studied in our future work.

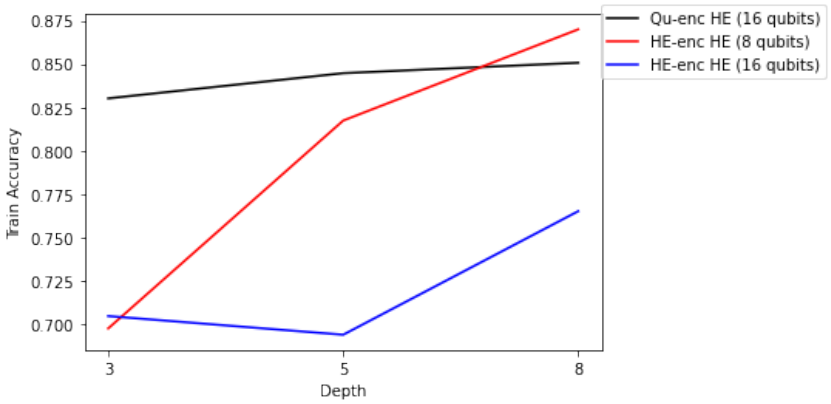
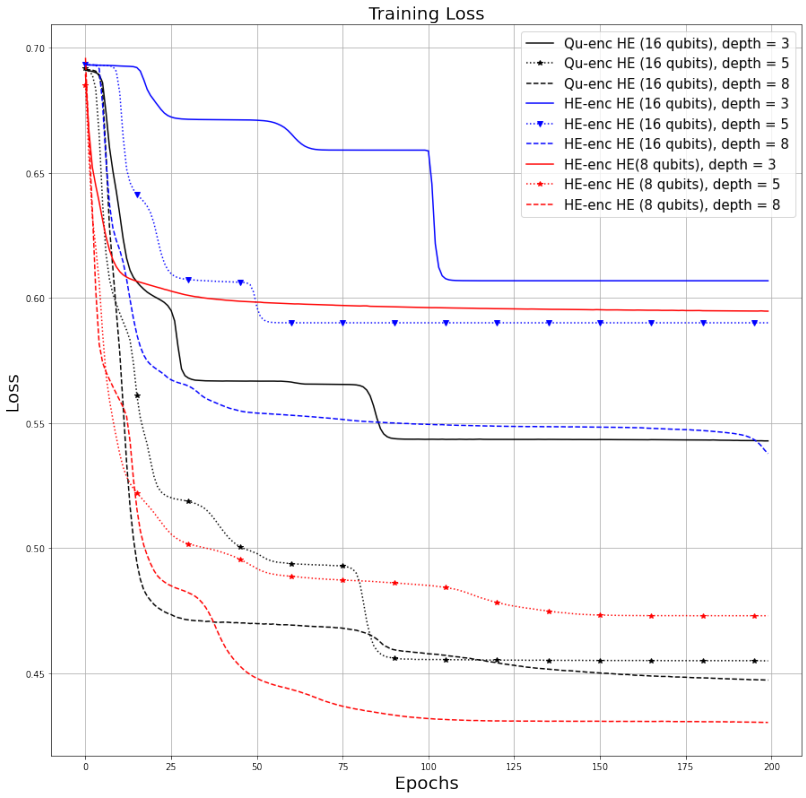
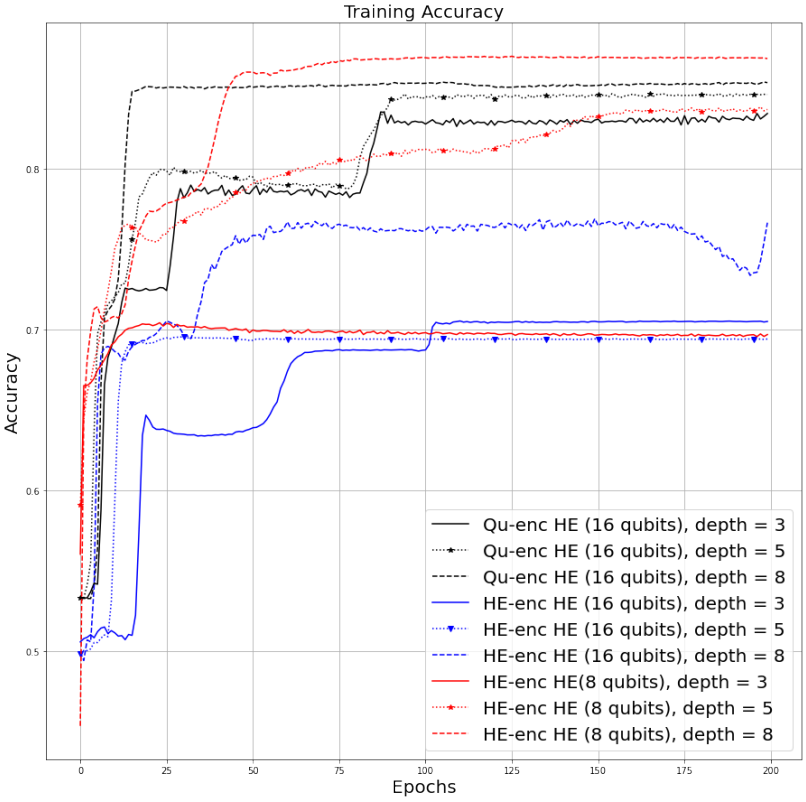


Figure 5. Comparison between the qubit-encoding and HE-encoding methods at different depth values.

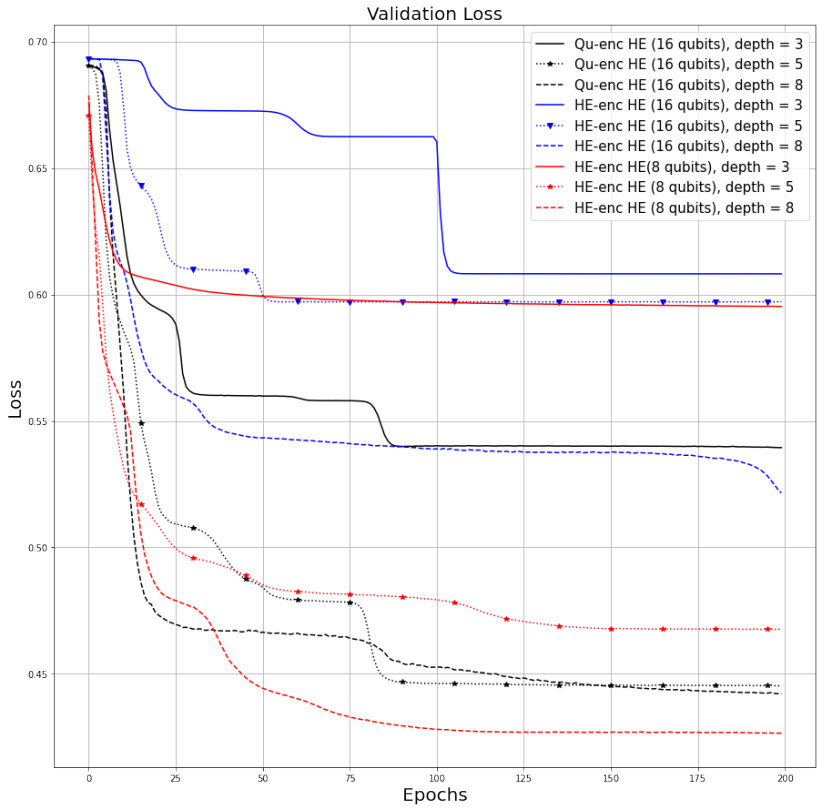


(**a**) Training loss

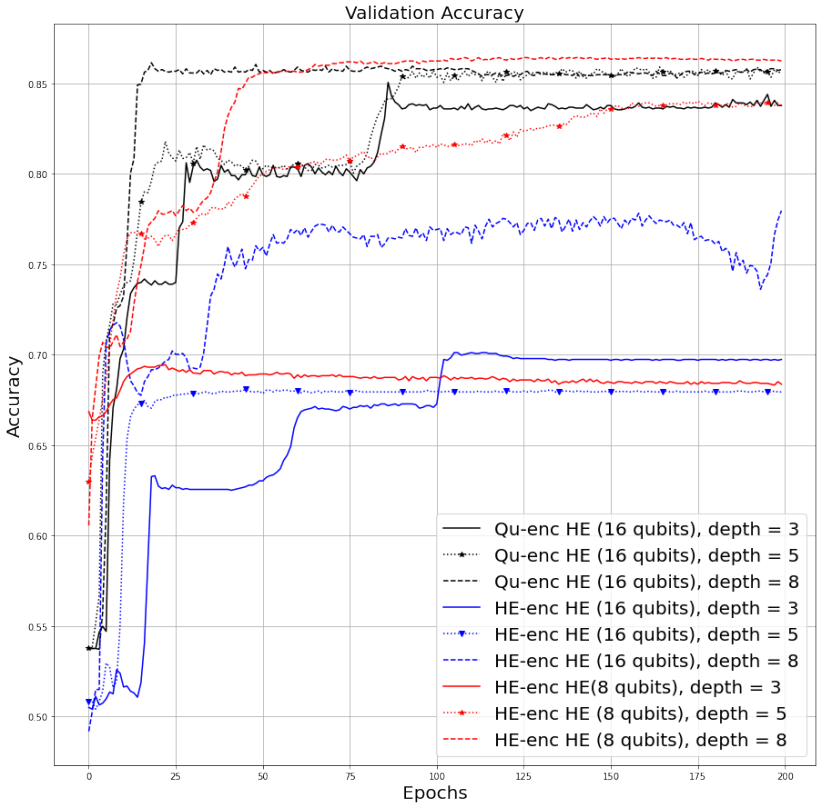


(**b**) Training accuracy

Figure 6. Learning curves of the training of HE circuits.



(**a**) Validation loss



(**b**) Validation accuracy

Figure 7. Learning curves in the validation of HE circuits.

With HE encoding, given the same number of qubits, the performance differences of the three circuit depths are very significant. When the number of qubits is 16, the learning curves are not stable and the performances are low. The reason could be that there are many parameters to be trained in this setting, and so the circuits are overfitted. In fact, when the number of qubits is 8, the learning curves and performances become much better. Specifically, when the number of element layers is increased from 3 to 8, the accuracy is quickly improved, from 0.70 to 0.864, which is higher than the best result of qubit encoding. Especially when the circuit depth is 5 or 8, the learning curves do not encounter any barren plateaus, which is a very useful behavior. In addition, it should be noted that the state preparation circuit of HE encoding is not complex, so given a sufficient circuit depth (here depth = 8), HE encoding is a promising choice between the extreme options of qubit encoding and amplitude encoding.

5.2. Comparison of Different QNNs

In the above subsection, we can see that the behaviors of HE-encoded QNNs varying quite noticeably across different settings. To prevent complexity in the resulting figures and discussion, we will exclude HE-encoded HE QNNs in this subsection. Figure 8 shows the accuracy values of different QNNs. The learning curves in the training of the QNNs are provided in Figure 9, and the corresponding curves in their validation are shown in Figure 10.

First, the most noticeable feature shown in Figure 9a or Figure 10a is the loss curve of the original QDNN which is significantly lower than those of the other QNNs. As mentioned, this method's architecture consists of three QNN layers (QNNLs), each composed of encoder, transform, and output parts which are enhanced by bias terms. That is, the QDNN is similar to a classical deep neural network. Thus, the learning curves of QDNN and its variants do not have problems of barren plateaus. Compared to the other QNNs that do not require conversion and a separate readout, the accuracy of QDNN is the highest (Figure 9b or Figure 10b). Indeed, the accuracy of the original QDNN is only lower than the cases of using an ancilla qubit readout, which are the baseline QNN and the QDNN (ancilla). However, the other variant, which is QDNN without bias, has lower performance than the original QDNN. This indicates that using bias terms in QDNN is in fact effective to improve the performance. Anyway, the QDNN without bias still performs better than HE QNNs. This could be because the QDNN has much more layers than HE QNNs. We can conclude that, although the QDNN has a complex architecture, it results in a stable and high performance in both training and validation. The HE QNNs, which could be considered as simplified versions of the QDNN without the state conversion between layers, provide lower and more unstable performances in both training and validation. HE QNNs with the depth of 3 or 5 have especially stair-like learning curves.

It results that the best accuracy is in fact provided by the baseline QNN, not the QDNN, though the baseline QNN's structure is quite simple. This could be due to the fact that, in the baseline QNN's structure, all input qubits are entangled in the single ancilla qubit which is used for readout. Meanwhile, in HE circuits, entanglement is just applied to pairs of adjacent qubits for efficient hardware implementation. This could also explain the (small) improvement in QDNN with the ancilla readout compared to the original QDNN as seen in Figure 9b or Figure 10b.

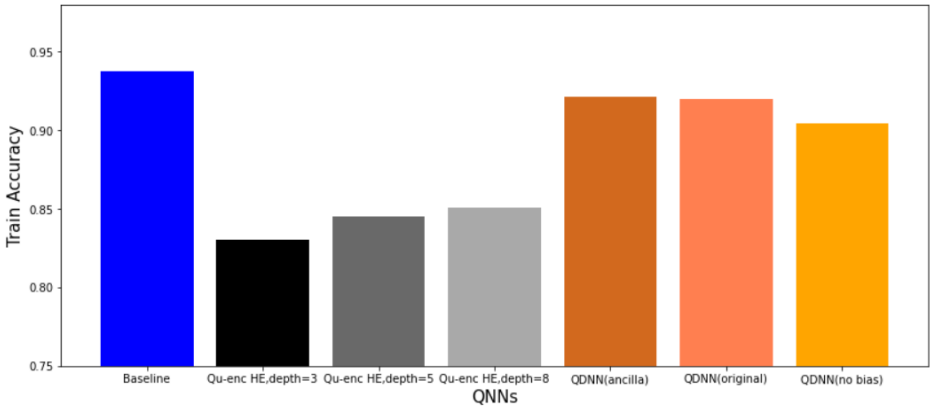
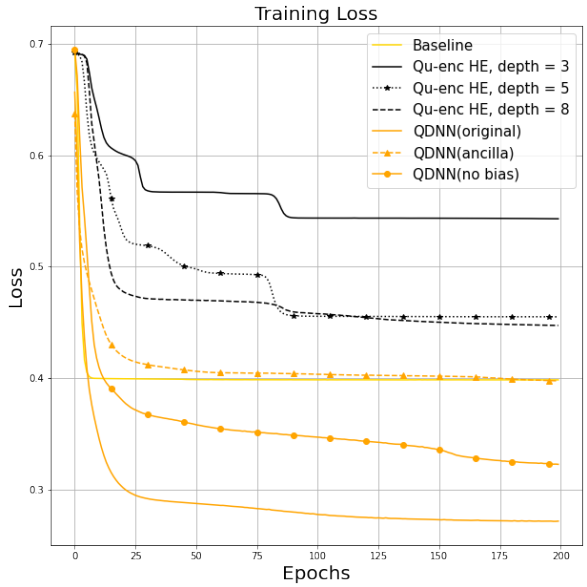
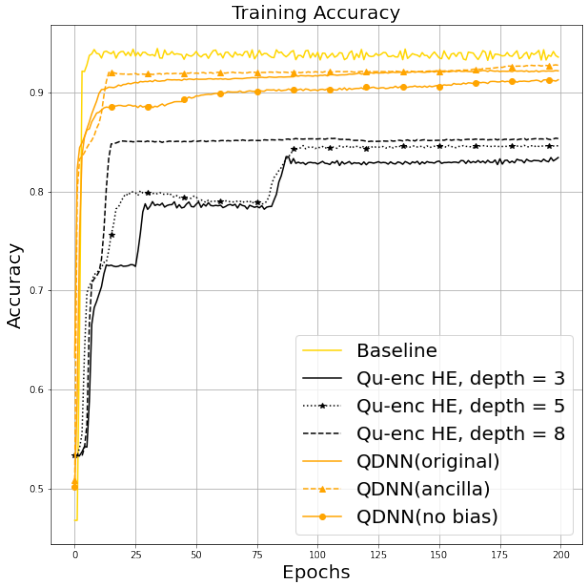


Figure 8. Accuracy values of different QNNs.

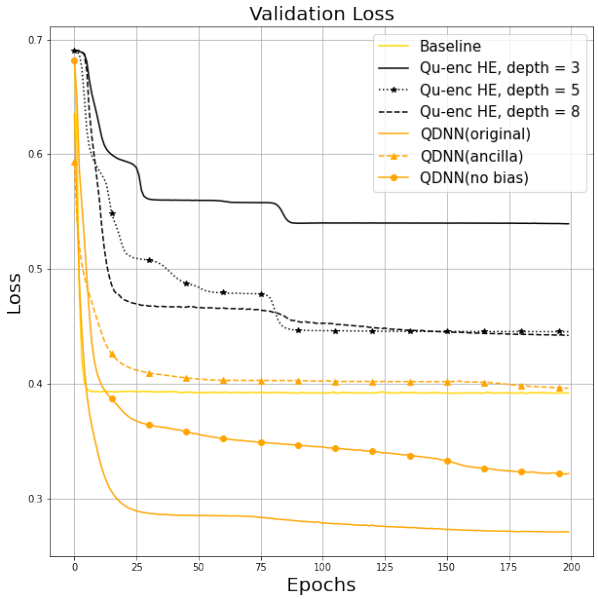


(**a**) Training loss

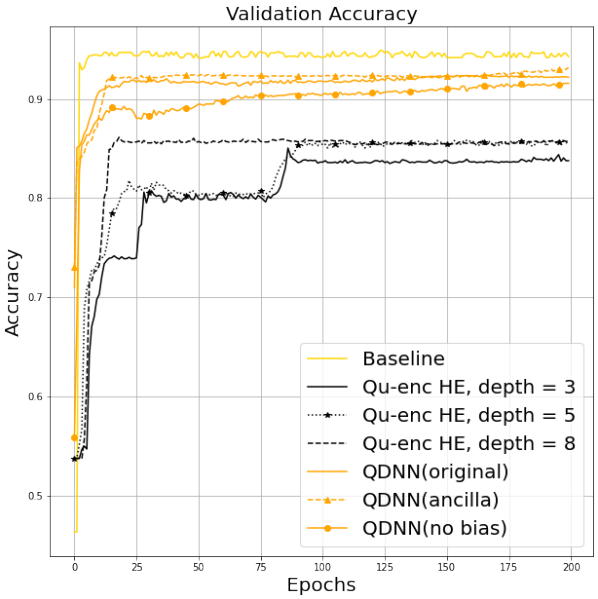


(**b**) Training accuracy

Figure 9. Learning curves in the training of different QNNs.



(**a**) Validation loss



(**b**) Validation accuracy

Figure 10. Learning curves in the validation of different QNNs.

The average training time (seconds per epoch) of QNNs is shown in Table 2. It can be seen that the training time is strongly dependent on the number of qubits. With 8-qubit HE-encoded QNNs, the training time per epoch is just 10∼16 s. For the QDNN model and its variants, which also employ 8 qubits, the training time is increased to 25∼26 s due to the deep architecture. However, when the number of qubits is 16, the training time is significantly increased by approximately 16∼28 times. Interestingly, given the same number of qubits, the training time is almost linear with respect to the circuit depth, as shown in Figure 11.

Table 2. Average training time (seconds per epoch) of different QNNs.

QNNs	QDNN			Qu-enc HE (16 qubits)			HE-enc HE (8 qubits)			HE-enc HE (16 qubits)			Base.
	Orig.	No bias	Ancil.	Depth 3	Depth 5	Depth 8	Depth 3	Depth 5	Depth 8	Depth 3	Depth 5	Depth 8	
Time	25	26	26	182	293	457	10	12	16	179	282	447	157

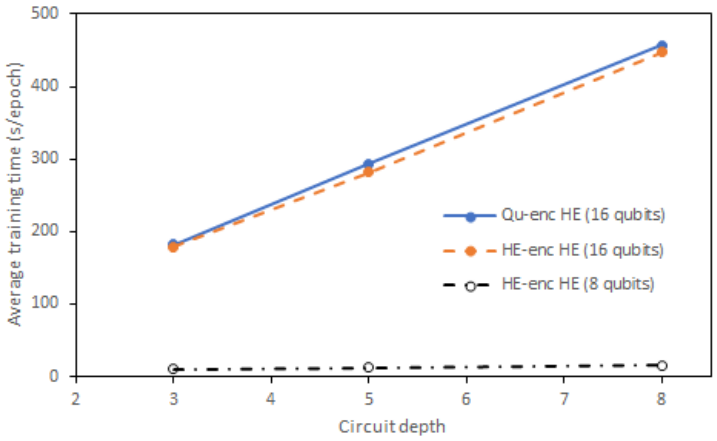


Figure 11. Relationship of training time and circuit depth.

5.3. Discussions

It is obvious that, thanks to the strategy of gradually increasing the basic HE circuit's depth, we can see the behaviors of different data-encoding types when the number of circuit layers is increased. In fact, the choice of encoding type should depend on the depth of a circuit, on the number of qubits, as well as on the type of data. In a recent evaluation [13], amplitude encoding was found to provide better results than qubit encoding. However, in [13], a data sample is a hand-crafted scalar value and there are only two qubits in the quantum circuit. In our evaluation, a sample is an actual image, which is high-dimensional data. To support this kind of data, many more qubits would be needed. As the complexity of state preparation for amplitude encoding is exponentially increased with respect to the number of qubits, this kind of encoding is not practical for image data. We found that HE encoding is actually very promising, with good performances and natural learning curves when (1) the circuit width is not high (e.g., 8 in our evaluations) and (2) the circuit depth is not too low (here 5 or 8). In addition, the experimental results show that qubit encoding can provide good performances in image classification across different circuit depths. Regarding the output, an ancilla readout qubit can provide some boost in performance. However, the problem of this method is being restricted by only binary classification task. In order to support multiple classes, more ancilla qubits are required, which increases the cost of circuits. Furthermore, when a circuit has high complexity (e.g., the number of qubits is 16 and the depth is 8), low performances could be due to overfitting and/or barren plateaus. However, as seen in the learning curves of HE-encoding 16-qubits QNNs, the problems of barren plateaus are much more prominent than overfitting because the learning curves quickly encounter a barren plateau and are then stuck at a low performance state.

To deal with the barren plateau problem, some recent studies have investigated different cost functions [31] and layer-wise training strategy [32]. From the above, we can see that this problem can also be tackled by selecting appropriate settings (namely circuit width and depth) or using the deep network structure as in QDNN.

From the above analysis, the key findings in this evaluation can be summarized as follows.

- Using image data in this evaluation, it is confirmed that a higher circuit depth helps improve the classification performance. However, the impact of circuit depth depends on the number of qubits and the encoding type of the QNN. For example, with qubit

encoding, the performances of different circuit depths are very close—whereas with HE encoding, the performances are widely varying;
- Amplitude encoding does not efficiently deal with image data. This is different from the finding in [13], which deals with two-feature data;
- HE encoding has good performance and encounters no barren plateaus when the circuit's depth and width are appropriately set. However, when low complexity (i.e., low circuit depth) is the first priority, qubit encoding should be used;
- QNNs based on qubit encoding could achieve good performance (over 0.85 with HE QNNs and over 0.95 with baseline QNN). However, there exist barren plateaus in the learning curves. Importantly, qubit-encoded HE circuits with a depth of 3 or 5 have jumps from one plateau to another plateau. This could be a big problem if the training process decides to terminate early, for example, due to time constraints;
- Interestingly the baseline QNN provides the best performance. This could be thanks to the highly entangled structure of this circuit. However, the implementation of such a complex structure is not as easy as HE circuits;
- QDNN has good performance and quite natural learning curves (without barren plateaus). This is because this method is closest to the existing architecture of classical deep neural networks;
- The bias terms in QDNN significantly contribute to the classification performance. Furthermore, even QDNN without bias is still better than the evaluated HE basic circuits. This could be explained by the fact that QDNN has much more layers than the HE circuits;
- The use of a separate (ancilla) qubit as the readout qubit helps improve the performance. Nonetheless, using ancilla qubits would make a circuit more complex and expensive. A more detailed investigation of different output styles will be reserved for our future work.

This study still has a number of limitations, which are reserved for our future work. First, we did not consider certain QNNs from the literature, especially the QNNs based on quantum artificial neurons. Furthermore, some efficient data embedding algorithms have been proposed to reduce the complexity of amplitude encoding (e.g., [33]). It is necessary to evaluate these algorithms against the basic data encoding types and the HE encoding used in this study. Currently, existing studies mostly employ the Adam optimizer for training. In our work, this optimizer is also used to maintain consistent settings with others. In fact, in a hybrid quantum–classical system, there are many choices in both the quantum part and the classical part. However, this study just focuses on the quantum part. To improve the training process, different optimizers, cost functions [31], training strategies (e.g., layer-wise training [32]), and many other factors will be extensively explored. Moreover, current evaluation results are obtained by simulations. To see the practical impacts of different factors on classification performance, some simple but effective QNNs will be selected and implemented on a real quantum computer.

6. Conclusions and Future Work

In this study, we evaluated different QNNs for the task of image classification. The simulation was based on the hybrid quantum–classical architecture, where gradient-based optimization in the classical part helps train a QNN in the quantum part. Based on empirical results, various new findings were obtained. We showed that qubit encoding and HE encoding could be efficient and effective to embed classical data into QNNs. Interestingly, when the circuit's depth and width are appropriately set, the HE-encoded QNNs do not reach any plateaus, i.e., the circuit always learns. Overall, the HE-encoded QNN with 8 qubits and a depth equal to 8 could be considered the best thanks to its simple implementation, fast training time, good learning curves, and high accuracy. When the implementation of a high circuit depth is not a problem, the QDNN should be the method of choice due to its very high performance. When circuit depth should be minimized, the qubit-encoded circuit with depth equal to 3 is the best one. However, this circuit encounters

multiple barren plateaus, which is problematic when the training process stops too early. In addition, the differences of QDNN with other designs are the quantum–classical conversion and the use of bias terms. Therefore, more work is necessary to understand how conversion between quantum data and classical data can reduce the detriment of barren plateaus.

Author Contributions: Conceptualization, T.N., I.P. and T.C.T.; investigation, T.N., I.P., Y.W. and T.C.T.; methodology, T.N., I.P. and T.C.T.; project administration, I.P. and T.C.T.; software, T.N., I.P. and T.C.T.; validation, T.N., I.P., Y.W. and T.C.T.; visualization, T.N., I.P., Y.W. and T.C.T.; writing—original draft, T.N., I.P. and T.C.T. All authors have read and agreed to the published version of the manuscript.

Acknowledgments: This research is partially supported by the competitive fund of the University of Aizu.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Trabesinger, A. Quantum computing: Towards reality. *Nature* **2017**, *543*, S1. [CrossRef]
2. Preskill, J. Quantum computing in the NISQ era and beyond. *Quantum* **2018**, *2*, 79. [CrossRef]
3. Biamonte, J.; Wittek, P.; Pancotti, N.; Rebentrost, P.; Wiebe, N.; Lloyd, S. Quantum machine learning. *Nature* **2017**, *549*, 195–202. [CrossRef]
4. Havlíček, V.; Córcoles, A.D.; Temme, K.; Harrow, A.W.; Kandala, A.; Chow, J.M.; Gambetta, J.M. Supervised learning with quantum-enhanced feature spaces. *Nature* **2019**, *567*, 209–212. [CrossRef]
5. Benedetti, M.; Lloyd, E.; Sack, S.; Fiorentini, M. Parameterized quantum circuits as machine learning models. *Quantum Sci. Technol.* **2019**, *4*, 043001. [CrossRef]
6. Chen, S.Y.C.; Yoo, S. Federated quantum machine learning. *Entropy* **2021**, *23*, 460. [CrossRef]
7. Abbas, A.; Sutter, D.; Zoufal, C.; Lucchi, A.; Figalli, A.; Woerner, S. The power of quantum neural networks. *Nat. Comput. Sci.* **2021**, *1*, 403–409. [CrossRef]
8. Broughton, M.; Verdon, G.; McCourt, T.; Martinez, A.J.; Yoo, J.H.; Isakov, S.V.; Massey, P.; Halavati, R.; Niu, M.Y.; Zlokapa, A.; et al. Tensorflow quantum: A software framework for quantum machine learning. *arXiv* **2020**, arXiv:2003.02989.
9. Bergholm, V.; Izaac, J.; Schuld, M.; Gogolin, C.; Alam, M.S.; Ahmed, S.; Arrazola, J.M.; Blank, C.; Delgado, A.; Jahangiri, S.; et al. Pennylane: Automatic differentiation of hybrid quantum-classical computations. *arXiv* **2018**, arXiv:1811.04968.
10. Luo, X.Z.; Liu, J.G.; Zhang, P.; Wang, L. Yao.jl: Extensible, efficient framework for quantum algorithm design. *Quantum* **2020**, *4*, 341. [CrossRef]
11. Sim, S.; Johnson, P.D.; Aspuru-Guzik, A. Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms. *Adv. Quantum Technol.* **2019**, *2*, 1900070. [CrossRef]
12. Hubregtsen, T.; Pichlmeier, J.; Stecher, P.; Bertels, K. Evaluation of parameterized quantum circuits: On the relation between classification accuracy, expressibility, and entangling capability. *Quantum Mach. Intell.* **2021**, *3*, 1–19. [CrossRef]
13. Sierra-Sosa, D.; Telahun, M.; Elmaghraby, A. Tensorflow quantum: Impacts of quantum state preparation on quantum machine learning performance. *IEEE Access* **2020**, *8*, 215246–215255. [CrossRef]
14. Farhi, E.; Neven, H. Classification with quantum neural networks on near term processors. *arXiv* **2018**, arXiv:1802.06002.
15. Grant, E.; Benedetti, M.; Cao, S.; Hallam, A.; Lockhart, J.; Stojevic, V.; Green, A.G.; Severini, S. Hierarchical quantum classifiers. *NPJ Quantum Inf.* **2018**, *4*, 1–8. [CrossRef]
16. Sengupta, K.; Srivastava, P.R. Quantum algorithm for quicker clinical prognostic analysis: An application and experimental study using CT scan images of COVID-19 patients. *BMC Med. Inform. Decis. Mak.* **2021**, *21*, 1–14. [CrossRef]
17. Abdel-Khalek, S.; Algarni, M.; Mansour, R.F.; Gupta, D.; Ilayaraja, M. Quantum neural network-based multilabel image classification in high-resolution unmanned aerial vehicle imagery. *Soft Comput.* **2021**, 1–12. [CrossRef]
18. Henderson, M.; Shakya, S.; Pradhan, S.; Cook, T. Quanvolutional neural networks: Powering image recognition with quantum circuits. *Quantum Mach. Intell.* **2020**, *2*, 1–9. [CrossRef]
19. Liu, J.; Lim, K.H.; Wood, K.L.; Huang, W.; Guo, C.; Huang, H.L. Hybrid quantum-classical convolutional neural networks. *Sci. China Phys. Mech. Astron.* **2021**, *64*, 1–8. [CrossRef]
20. Kerenidis, I.; Landman, J.; Prakash, A. Quantum algorithms for deep convolutional neural networks. *arXiv* **2019**, arXiv:1911.01117.
21. Zhao, C.; Gao, X.S. QDNN: Deep neural networks with quantum layers. *Quantum Mach. Intell.* **2021**, *3*, 1–9. [CrossRef]
22. Mitarai, K.; Negoro, M.; Kitagawa, M.; Fujii, K. Quantum circuit learning. *Phys. Rev. A* **2018**, *98*, 032309. [CrossRef]
23. Kandala, A.; Mezzacapo, A.; Temme, K.; Takita, M.; Brink, M.; Chow, J.M.; Gambetta, J.M. Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. *Nature* **2017**, *549*, 242–246. [CrossRef]
24. Cao, Y.; Guerreschi, G.G.; Aspuru-Guzik, A. Quantum neuron: An elementary building block for machine learning on quantum computers. *arXiv* **2017**, arXiv:1711.11240.
25. Torrontegui, E.; García-Ripoll, J.J. Unitary quantum perceptron as efficient universal approximator. *EPL (Europhys. Lett.)* **2019**, *125*, 30004. [CrossRef]

26. Kristensen, L.B.; Degroote, M.; Wittek, P.; Aspuru-Guzik, A.; Zinner, N.T. An artificial spiking quantum neuron. *NPJ Quantum Inf.* **2021**, *7*, 1–7. [CrossRef]
27. Tacchino, F.; Macchiavello, C.; Gerace, D.; Bajoni, D. An artificial neuron implemented on an actual quantum processor. *NPJ Quantum Inf.* **2019**, *5*, 1–8. [CrossRef]
28. Ban, Y.; Chen, X.; Torrontegui, E.; Solano, E.; Casanova, J. Speeding up quantum perceptron via shortcuts to adiabaticity. *Sci. Rep.* **2021**, *11*, 1–8.
29. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
30. McClean, J.R.; Boixo, S.; Smelyanskiy, V.N.; Babbush, R.; Neven, H. Barren plateaus in quantum neural network training landscapes. *Nat. Commun.* **2018**, *9*, 1–6. [CrossRef]
31. Cerezo, M.; Sone, A.; Volkoff, T.; Cincio, L.; Coles, P.J. Cost function dependent barren plateaus in shallow parametrized quantum circuits. *Nat. Commun.* **2021**, *12*, 1–12. [CrossRef]
32. Skolik, A.; McClean, J.R.; Mohseni, M.; van der Smagt, P.; Leib, M. Layerwise learning for quantum neural networks. *Quantum Mach. Intell.* **2021**, *3*, 1–11. [CrossRef]
33. Marin-Sanchez, G.; Gonzalez-Conde, J.; Sanz, M. Quantum algorithms for approximate function loading. *arXiv* **2021**, arXiv:2111.07933.