QSpeech: Low-Qubit Quantum Speech Application Toolkit

Zhenhou Hong, Jianzong Wang*, Xiaoyang Qu, Chendong Zhao, Wei Tao and Jing Xiao

Ping An Technology (Shenzhen) Co., Ltd., Shenzhen, China

Emails: {hongzhenhou168,wangjianzong347,qxiaoyang343,zhaochendong343,taowi085,xiaojing661}@pingan.com.cn

Abstract—Quantum devices with low qubits are common in the Noisy Intermediate-Scale Quantum (NISQ) era. However, Quantum Neural Network (QNN) running on low-qubit quantum devices would be difficult since it is based on Variational Quantum Circuit (VQC), which requires many qubits. Therefore, it is critical to make QNN with VQC run on low-qubit quantum devices. In this study, we propose a novel VQC called the low-qubit VQC. VQC requires numerous qubits based on the input dimension; however, the low-qubit VQC with linear transformation can liberate this condition. Thus, it allows the QNN to run on low-qubit quantum devices for speech applications. Furthermore, as compared to the VQC, our proposed low-qubit VQC can stabilize the training process more. Based on the low-qubit VQC, we implement QSpeech, a library for quick prototyping of hybrid quantum-classical neural networks in the speech field. It has numerous quantum neural layers and QNN models for speech applications. Experiments on Speech Command Recognition and Text-to-Speech show that our proposed low-qubit VQC outperforms VQC and is more stable.

Index Terms—Quantum Neural Networks Library, Hybrid Quantum-classical Neural Networks, Low-qubit, Variational Quantum Circuit, Speech Application

I. INTRODUCTION

Recent progress in the development and commercialization of quantum computation has significantly impacted the landscape of Quantum Neural Network (QNN) [1], [2]. These quantum algorithms designed for the Noisy Intermediate-Scale Quantum (NISQ) [3] computer are programmed with a Variational Quantum Circuit (VQC) [4] and a classical optimizer. The approach is similar to the parameter updating in many machine learning algorithms. The QNN has been developed by some researchers in various applications [5], [6].

The number of qubits in the NISQ era [3] is 5 to 50 qubits. Several quantum devices have less than 10 qubits. The application of quantum algorithms in these quantum devices is a key challenge. Mainstream research has focused on running the quantum algorithm in low-qubit (2 to 10) quantum devices [7], [8]. Using a scalable trapped atomic ion system [9], the Grover Search algorithm [7] can be implemented on 3 qubits quantum device. The Quantum Principal Component Analysis (QPCA) [10] is developed to the Resonant QPCA [11], which can be conducted in a low-qubit quantum device using a resonant analysis algorithm.

However, it can be challenging when QNN or hybrid quantum-classical neural networks are implemented in speech applications [12], [13]. First, these algorithms with VQC in speech applications need many qubits, making them difficult to implement. Second, with numerous qubits, QNN will have the Barren Plateau problem [14], [15] in the training process. When restricting the number of qubits, it is challenging to assure model performance.

![Diagram](image-url)
long short term memory [20], Quantum Gate Recurrent Unit (QGRU), and Quantum Attention (QAttention). Thus, the QSpeech can easily implement the hybrid quantum-classical neural network.

Our main contributions are as follows:

- Low-qubit VQC has been proposed, allowing QNN to be used in a low-qubit quantum device. It can stabilize the training process compared to VQC.
- Based on low-qubit VQC, we design a library for the rapid prototyping of hybrid quantum-classical neural networks in speech applications, named QSpeech.
- In the experiments, hybrid quantum-classical neural networks were successfully applied in speech applications.

II. RELATED WORKS

A. Quantum Convolutional Neural Networks

Cong et al. [21] propose a quantum CNN based on a quantum circuit model that incorporates the idea of convolutional and pooling layers from classical CNNs. The proposed architecture is similarly layered; however, it applies 1D convolutions to the input quantum state (contrary to 2D/3D convolutions on images). On the input state density \( \rho = \text{i} \mu \), the convolutional layer is modeled as a quasi-local unitary operation. This unitary operator, denoted by \( U_i \), is applied to several successive sets of input qubits up to a predefined depth. The pooling layer is implemented by measuring some qubits and applying unitary rotations \( V_i \) to the nearby qubits. The observations on the qubits determine the rotation operation. This combines the functionalities of dimensionality reduction (the output of \( V_i \) is of lower dimension) and nonlinearity (due to the partial measurement of qubits). After the required number of convolutional blocks and pooling units, the unitary \( F \) implements the fully connected layer. A final measurement on the output of \( F \) yields the network output.

Recently, Kerendis et al. [22] identified the relationship between convolutions and matrix multiplications and proposed the first quantum algorithm to compute a CNN’s forward pass as a convolutional product. They also proposed a quantum backpropagation algorithm to learn network parameters through gradient descent. In an application of QCNNs, Zhang et al. [23] and Melnikov et al. [24] propose special convolutional neural networks for extracting features from graphs to identify those showing the quantum advantage. Hong et al. [25] propose a novel structure of QCNN that outperforms other DNN algorithms in predicting protein distance and contact.

B. Quantum Recurrent Neural Networks

There have also been several interesting suggestions for developing quantum variants of RNN. Hiba-Alallah et al. [26] propose a quantum variant of RNNs using variational wave functions to learn the approximate ground state of a quantum Hamiltonian. Roth [27] propose an iterative retraining approach using RNNs for simulating bulk quantum systems via mapping translations of lattice vectors to the RNN time index. Hopfield Networks [28] were a popular early type of a recurrent NN for which quantum variants have been developed in several works [29].

C. Quantum Learning for Speech Processing

Li et al. [30] propose a QNN that combines fuzzy theoretic principles with continuous digits inference. Their results show that more than 15% error reduction is obtained on a speaker-independent continuous digits recognition task. Li et al. [31] propose a QNN used in speech enhancement. The QNN can be trained on the backpropagation algorithm. Its simulation results show that the QNN outperforms the traditional spectral subtraction in speech enhancement. Fu et al. [32] propose an improved particle swarm optimization (IPSO), called IPSO-QNN, for training the QNN. Its experimental results in speech recognition are better than other approaches. Yang et al. show that the QCNN can use in privacy-preserving learning [6]. It uses the QCNN to encode the speech feature to quantum state for data protection.

III. PRELIMINARIES

The entire quantum variational circuit [4] consists of three components: Data Encoding Layer, Variational Layer, and Quantum Measurement Layer. Notably, the number of qubits and measurements can be changed to fit the problem of interest, and the variational layer can contain many dashed boxes to increase the number of parameters, all subject to the capacity and capability of the quantum machines used in the experiments.

Data Encoding Layer. We first encode a classical input vector into a quantum state in this layer, which is required for further processing. A general \( N \)-qubit quantum state can be represented as the following:

\[
|\phi\rangle = \sum_{(q_1, q_2, ..., q_N) \in \{0, 1\}^N} c_{q_1, q_2, ..., q_N} |q_1\rangle \otimes |q_2\rangle \otimes |q_3\rangle \otimes \cdots \otimes |q_N\rangle
\]

(1)

where \( c_{q_1, q_2, ..., q_N} \in \mathbb{C} \) is the amplitude of each quantum state and \( q_i \in \{0, 1\} \). The square of the amplitude \( c_{q_1, q_2, ..., q_N} \) is the probability of measurement with the post-measurement state in \( |q_1\rangle \otimes |q_2\rangle \otimes |q_3\rangle \otimes \cdots \otimes |q_N\rangle \), and the total probability should sum to 1, i.e.,

\[
\sum_{(q_1, q_2, ..., q_N) \in \{0, 1\}^N} \|c_{q_1, q_2, ..., q_N}\|^2 = 1
\]

(2)

For the quantum gates, the input will first be flattened and transformed into rotation angles. In this study, \( f_e(\cdot) \) represent the data encoder.

Variational Layer. After encoding the classical values into a quantum state, it will be subject to a series of unitary transformations. The variational layer consists of two parts, one for entanglement and the other for rotation, which include numerous single-qubit unitary rotations \( U \) parameterized by \( 1 \) to \( m \) parameters \( \theta_1, ..., \theta_m \). The parameters labeled by \( \theta_1, ..., \theta_m \) are the ones that will be updated by the optimization procedure. In this study, \( f_u(\theta_1, ..., \theta_m; \cdot) \) represent the variational circuit.
Quantum Measurement(Decode) Layer. We performed a quantum measurement layer to obtain the transformed data from VCs. The VCs’ ensemble samplings (expectation values) were considered. Quantum simulation software (for example, PennyLane or IBM Qiskit) can be used to calculate this value deterministically. While implementing on a real quantum computer, it is necessary to prepare the same system and repeatedly use the measurements to gain adequate statistics. In this study, $f_d()$ represent the measurement layer.

IV. PROPOSED METHODS

A. Challenges of VQC in Low-qubit Quantum Device

The VQC consists of data encoder, variational circuit, and measurement [4], [33]. It only processes the 1D array and necessitates qubits based on the size of one dimension input. It is difficult to use the VQC in a high-dimensional input $x \in \mathbb{R}^{1 \times n}$ ($n$ is 128 or 256 or even larger) because it requires $n$ qubits to encode and transform the input in VQC, which exceeds the number of qubits. Furthermore, with multiple qubits, it will be easy to cause the Barren Plateaus problem [14], [15] during the training process.

B. Problem Statement

Similar to deep learning [34]–[36] in speech application, using the QNN $f_Q(.)$ to model a function can generate $y$, which represents the classify of audio or generate speech from data input $x$.

$$y = f_q(W, f_v(\theta), x)$$  \hspace{1cm} (3) \\
\mathcal{L}(W, f_v(\theta), x, y_{true}) = \mathcal{L}(f_q(W, f_v(\theta), x), y_{true})$$  \hspace{1cm} (4)

In VQC $f_v(\theta)$, we use the $\text{RY}$ gate $R_y(.)$ in Data Encoding Layer, $y_{true}$ is the true label in data.

The optimization for QNN in speech application, is similar with DNN:

$$\hat{\theta} = \arg \min_{\theta} \mathcal{L}(f_q(W, f_v(\theta), x), y_{true})$$  \hspace{1cm} (5) \\
\hat{W} = \arg \min_{W} \mathcal{L}(f_q(W, f_v(\theta), x), y_{true})$$

where $\hat{\theta}$ and $\hat{W}$ are optimized parameters in $f_q(W, f_v(\theta), x)$. Based on PennyLane setting, VQC $f_v(\theta)$ can perform the optimization using gradient descent approaches. In this study, we follow the [37], using the Adam optimization method [38] to minimize the loss $\mathcal{L}(f_q(W, f_v(\theta), x), y_{true})$.

C. Low-qubit Variational Quantum Circuit(VQC)

As shown in Fig. 2, the two purple blocks are the new parts that we added in VQC. In low-qubit VQC, we add two linear transformations unlike in previous VQC [20], [21].

Consider $x$ as input, let $x = \{x_1, x_2, ..., x_n\}, x \in \mathbb{R}^{1 \times n}$. LT $f_{lt}(W_{\cdot})$ is linear transformation. $W_{\cdot}$ is the learnable parameter in $f_{lt}(W_{\cdot})$. $f_{lt}(W_{\cdot}): \mathbb{R} \rightarrow y$. $q_1$, $q_2$, and $q_3$ are representative quantum states. $y_1$ and $y_2$ are the outputs of first and second $f_{lt}$, $y = \{y_1,y_2,...,y_j\}$. Then the formula of low-qubit VQC follows:

$$y_1 = f_{lt}^1(W; x)$$  \hspace{1cm} (6) \\
q_1 = f_c(y_1)$$  \hspace{1cm} (6) \\
q_2 = f_a(\theta_1, ..., \theta_m; c(q_1))$$  \hspace{1cm} (6) \\
q_3 = f_d(q_2)$$  \hspace{1cm} (6) \\
y_2 = f_{lt}^2(W; q_3)$$  \hspace{1cm} (6)$$

Where $\theta_1, ..., \theta_m$ is the parameters in $U(\theta; \cdot)$, $m$ is a positive integer. $f_c$, $f_a$, and $f_d$ are the $\text{encoder}$, $\text{variational circuit}$, and measurement operation. $c(.)$ is the $\text{CNOT}$ gate. $f_{lt}^1$ and $f_{lt}^2$ is the first and second linear transformation.

The advantage of adding the linear transformation is that it can squeeze high-dimensional input in the data encoding layer or expand the output filters after measurement. Several qubits are not required since they will be squeezed with the linear transformation before the data encoding and variational layers. It can make convergence stable in the training and need fewer resources to simulate the quantum computation. Additionally, it makes low-qubit VQC process multiple channels simultaneously while extracting the features from input. It is beneficial for the QConv and QAttention layers. The QAttention layer can allow high-dimensional units such as 128 or 256 with few qubits. Thus, the Transformer with QAttention can be used in the low-qubit quantum devices for speech applications [39] such as Text-to-Speech (TTS).

D. Analysis of the Low-qubit VQC

We found the $f_c(y_1)$ contains the $\text{arctan}(\cdot)$, its derivative may cause the gradient vanish. $f_a(y_1) = R_y(\text{arctan}(y_{11}))|0\ldots R_y(\text{arctan}(y_{11}))|0)$. Its derivative results of $R_y(\text{arctan}(y_{11}))$ as follow:

$$\frac{\partial R_y(\text{arctan}(y_{11}))}{\partial w_i} = -R_y(\text{arctan}(y_{11})) \frac{\partial \text{arctan}(y_{11})}{\partial w_i}$$  \hspace{1cm} (7) \\
\frac{\partial \text{arctan}(y_{11})}{\partial w_i} = \frac{1}{1 + (y_{11})^2} \frac{\partial y_{11}}{\partial w_i} = \frac{1}{1 + (y_{11})^2} x_i$$

Where $y_{11}$ is the element in $y_1, y_{11} = w x$. $w$ is a vector in $W$.

We analysis the $x_i/(1 + (y_i)^2)$. To observe the output, we fix the $y_i$ and change the $x_i$. We can see that the $y_i$ cannot be too big or too small, or the gradient will vanish. Therefore, we add the output clip operation in the first linear transformation. It is possible to optimize the weight of linear transformation. The output clip’s range is $[-3,3]$.

The LT with learnable parameters can be implemented using FC. We compare the linear transformation with other
subsampling approaches like bilinear interpolation. The linear transformation has two advantages: first, it can reduce the dimension and extract the features efficiently with learnable parameters; second, it can extract the features adapt to the VQC with learnable parameters. Bilinear interpolation simply reduces the dimension, causing the gradient vanish based on the derivative result.

E. Framework of QSpeech

Fig. 3 shows the software architecture of QSpeech. In the QSpeech, the main parts of QNN training and inference are written in python, which calls PennyLane and PyTorch by switching the backend option. PennyLane allows differentiable programming of quantum computers. We also provide complete recipes for conducting speech application experiments. As shown in Fig. 3, the QSpeech is based on the PyTorch and PennyLane. The functional or layers backend are qcircuit, qconv, qlstm, qgru, and qattention. The functional backend builds up the model libraries: Quantum Transformer-TTS (QTransformer-TTS), Quantum M5 (QM5), and Quantum Tacotron (QTacotron). The Quantum Layers (QLayers) are the functional backend, and the Quantum Models (QModels) are the model libraries in the major libraries in QSpeech. Using the PennyLane as a Quantum Machine Learning (QML) backend can easily embed the code into Python and PyTorch.

V. IMPLEMENTED QUANTUM NEURAL NETWORK IN SPEECH APPLICATION

A. Quantum M5 For SCR

![Fig. 4. The model architecture of QM5 compared with M5.](image)

The M5 [40] model is a deep convolutional neural network (CNN) that directly uses time-domain waveforms as inputs. To extract the feature from the raw speech data, the M5 model in speech command uses one dimension convolutional layer (Conv1d). Thus, we first implement the quantum one dimension convolutional (QConv1d) layer that can use in speech commands and other speech applications [41], [42]. Then, we use the M5 model with the QConv1d layer in the speech command task, called quantum M5(QM5).

Fig. 5 shows the QConv1d extract feature from raw speech data or feature map. We propose a QConv1d that uses the low-qubit VQC and help extract from a feature map, while avoiding source in the quantum simulator.

B. Quantum Tacotron For TTS

Tacotron [43] is an end-to-end generative TTS model that synthesizes speech directly from characters. It has an Attention Decoder part, which includes a pre-net, an attention recurrent neural networks (RNN), and a decoder RNN. The RNN component uses the GRU. As shown in Fig. 6, we change the original GRU to QGRU in attention GRU and decoder GRU. The others are maintained.

![Fig. 6. The attention decoder with GRU compared with the attention decoder with quantum GRU.](image)

To construct the basic unit of the proposed QGRU architecture, a QGRU cell, we stack the low-qubit VQC blocks as mentioned earlier together. In Fig. 7, each of the low-qubit VQC...
VQC is described in the previous section (see also Fig. 2). In a QGRU cell, there are five low-qubit VQC. For three low-qubit VQC in Fig. 7 below, the input is the concatenation $v_t$ of the hidden state $H_{t-1}$ from the previous time step and the current input vector $X_t$, and the output is three vectors obtained from the measurements at the end of each low-qubit VQC. The output value can then be further processed with two low-qubit VQC in Fig. 7 top right, to get $Y_t$ and $H_t$.

C. Quantum Transformer For TTS

Transformer-TTS [44] is an architecture that uses two parts (Encoder and Decoder) to transform one sequence into another. Here we implement the quantum self-attention with a single head and multihead. As shown in Fig. 8(a), adding two VQC blocks in the input and output, others are the same with Scaled-Dot Product Attention [44]. The quantum self-attention with multihead as shown in Fig. 8(b), the $Q$, $V$, $K$ computed by the VQC, input to the quantum self-attention with a single head. The final output is also computed by the VQC. The others are the same as the Multihead Attention [44].

![Quantum self-attention with single head.](image)

![Quantum self-attention with multihead.](image)

Fig. 8. The proposed quantum self-attention architecture with single head and multihead. The low-qubit VQC as shown in Figure 2.

With the quantum self-attention with multihead, the quantum Transformer-TTS (QTransformer-TTS) will also be implemented. As shown in Fig. 9, change the Multihead Attention to the quantum self-attention with multihead.

![Quantum self-attention with multihead as shown in Figure 8(b).](image)

Fig. 9. The proposed QTransformer-TTS architecture. The quantum self-attention with multihead as shown in Figure 8(b).

A. Dataset

**LJSpeech [45].** This dataset consists of 13,100 short audio clips of a single speaker reading passages from seven nonfiction books. The texts are normalized and inserted with the beginning character (space) and end character (a period), e.g., “There are 16 apples” is converted to “there are sixteen apples.”. The LJSpeech dataset is randomly divided into two sets: 12,600 samples for training and 500 samples for testing.

**Google Speech Command-V2 [46].** For spoken word recognition, we use the 35 class setting, including the following frequent speech commands: ‘left’, ‘go’, ‘yes’, ‘down’, ‘up’, ‘on’, ‘right’, ‘no’, ‘off’, ‘stop’, and so on with a total of 84,843 training examples and 11,005 testing examples.

B. Settings

**Quantum Environment.** The simulation of quantum computation runs on the CPU. We limit the number of qubits to small numbers (2 to 10). The $m$ of $\theta_1, ..., \theta_m$ is equal to 3 in VQC and low-qubit VQC.

**Model Architecture.** The QTransformer-TTS has two layers. It replaces self-attention with QAttention, the other settings follow [44]. The QGRU is a substitute for GRU in QTacotron. QTacotron has the same parameter as as [43]. The QM5 replaces the 1D convolution (Conv1d) with 1D QConv (QConv1d). The QTransformer-TTS, QM5, and QTacotron use the low-qubit VQC. As we discussed in Sec. IV-C, VQC requires similar qubits as the input dimensions, and it is considered large for quantum devices. Therefore, the low-qubit VQC in QTransformer-TTS and QTacotron, cannot be compared with VQC. The QConv1d with VQC or low-qubit VQC of QM5, in which the number of qubits is the same as the kernel size, follows the [20], [21]. The linear transformation in low-qubit VQC forms a fully connected layer.

**Training.** Adam is the optimizer in all experiments, with $\beta_1 = 0.9$, $\beta_2 = 0.99$. The batch size in the training process of QTransformer-TTS and QTacotron is 2. QM5 can be trained in a large batch size (e.g., 64, 256) using QConv1d with our proposed low-qubit VQC.

**Compared Schemes.** In the experiments, the QTransformer-TTS, QM5, and QTacotron models are compared with the original Transformer-TTS, M5, and Tacotron. The setups of training for the QTransformer-TTS, QM5, and QTacotron are the same as the original Transformer-TTS, M5, and Tacotron. It will also compare the results of different qubits in the experiments of QTransformer-TTS, QM5, and QTacotron.

**Evaluation.** In Speech Command Recognition (SCR) experiments, the metric of accuracy is the same as [40]. We use the mean opinion score (MOS) to measure generated voices’ qualities in TTS experiments. Each sentence is judged by ten males and ten females native English speakers.

C. Results and Analysis

**Q5M.** In Fig. 10(a), the yellow curve is the training loss of QM5 with VQC. Notably, it cannot stably converge as the iterative process increases. The black curve, which is
the QM5 with low-qubit VQC, can train in convergence and stability, indicating that using low-qubit VQC can handle the Barren Plateaus challenge. In Fig. 10(b), the training of QM5 with low-qubit VQC can improve convergence and stability compared to the M5. With more qubits in QM5, the QM5 model will converge faster.

![Feature map of M5](image)
![Feature map of QM5](image)

Fig. 10. The feature map output from M5 and QM5. (a) The feature map output from Conv1d in M5. (b) The feature map output from QConv1d in QM5.

A similar loss convergence. However, it shows that with more qubits, the QTacotron can converge better and with less loss.

Table II shows the MOS with Tacotron and QTacotron. QTacotron’s performance will improve with more qubits of the low-qubit VQC. Therefore, the QTacotron can achieve better results when using the low-qubit VQC. With eight qubits of the low-qubit VQC, the MOS of QTacotron is about 2.67, indicating that the QTacotron with low-qubit VQC can run in the TTS task.

In addition to the training process and performance, we provide the mel-spectrogram obtained from the Tacotron and QTacotron. As shown in Fig. 13(a)(b), similar mel-spectrograms are obtained from the Tacotron and QTacotron. We can observe that the QTacotron with low-qubit can achieve the approximate performance as Tacotron. QTransformer-TTS. In the training process, QTransformer-TTS outperforms Transformer-TTS, as shown in Fig. 12(b). Compared to the training loss curve in Fig. 12(b), the QTransformer-TTS converges faster than the Transformer-TTS.
between 0% and 40% of the training process. Furthermore, QTransformer-TTS, like the QTacotron, gives similar loss convergence in different qubits. With more qubits of the low-qubit VQC, the QTransformer-TTS obtains lower loss in the training process, indicating that the low-qubit VQC can work well in QAttention, like the low-qubit VQC in QConv1d and QGRU.

Table II also shows the MOS with Transformer-TTS and QTransformer-TTS. We can see that QTransformer-TTS performs better when using the low-qubit VQC. With more qubits of the low-qubit VQC, QTransformer-TTS will get better performance. With eight qubits of the low-qubit VQC, the MOS of QTransformer-TTS is about 2.52. It means that the QTransformer-TTS with low-qubit VQC can run in the TTS task with a low-qubit quantum device.

Furthermore, in Fig. 13(c)(d), we show the mel-spectrogram generated by the Transformer-TTS and QTransformer-TTS. The mel-spectrogram of the Transformer-TTS and QTransformer-TTS is similar. We conclude that QTransformer-TTS with low-qubit VQC can achieve the approximate performance as Transformer-TTS.

**Ablation Study.** The low-qubit VQC is compared in two scenarios: first, the LT is implemented by FC layer or bilinear interpolation; second, the LT layer with or without output clip operation. Table III shows the results.

**TABLE III**

| Model         | Methods of LT               | Results      |
|---------------|------------------------------|--------------|
| QM5           | use bilinear interpolation   | 37.51%       |
|               | use FC                       | 43.41%       |
| QTacotron     | use bilinear interpolation   | 2.21         |
|               | use FC                       | 2.47         |
| QTransformer-TTS | use bilinear interpolation | 2.24         |
|               | use FC                       | 2.32         |
| QM5           | without output clip          | 43.21%       |
|               | with output clip             | 50.11%       |
| QTacotron     | without output clip          | 2.41         |
|               | with output clip             | 2.67         |
| QTransformer-TTS | without output clip     | 2.39         |
|               | with output clip             | 2.52         |

**Implemented methods of LT.** In QM5, QTacotron, and QTransformer-TTS, the LT is implemented by FC layer or the LT implemented by bilinear interpolation. As shown in Table III, the results of using the FC layer are better than bilinear interpolation. In low-qubit VQC, using the FC layer outperforms the bilinear interpolation when implementing the LT.

**With or without output clip.** In QM5, QTacotron, and QTransformer-TTS, the results of LT with output clip operation are compared with LT without output clip operation. As shown in Table III shows, the results of using output clip are better than without output clip. In low-qubit VQC, the results of LT with output clip operation outperform LT without output clip operation.

**VII. CONCLUSION**

This study introduced a novel VQC, called low-qubit VQC. It can address the Barren Plateaux challenge and lower the requirement of qubits compared to VQC. It is crucial because VQCs are used in low-qubit quantum devices in the NISQ era. Low-qubit VQC can be used to implement QNN models in low-qubit quantum devices. Based on low-qubit VQC, we propose the QSpeech toolkit. QSpeech fully uses PyTorch and PennyLane as a QML engine and significantly simplifies the implementation and training of the entire speech application. Our QSpeech provided various quantum neural layers and hybrid quantum-classical neural networks. The experimental results showed that the hybrid quantum-classical neural networks with low-qubit VQC can be successfully used in speech applications.

**ACKNOWLEDGMENT**

This paper is supported by the Key Research and Development Program of Guangdong Province under grant No.2021B0101400003. Corresponding author is Jianzong Wang from Ping An Technology (Shenzhen) Co., Ltd (jzwang@188.com).
