Received 12 November 2023; revised 12 February 2024; accepted 2 March 2024; date of publication 6 March 2024; date of current version 18 April 2024.

Digital Object Identifier 10.1109/TQE.2024.3373903

Application of Quantum Recurrent Neural Network in Low-Resource Language Text Classification

WENBIN YU[1,2,4] (Member, IEEE), LEI YIN[1], CHENGJUN ZHANG[2,3,4] (Member, IEEE), YADANG CHEN[3] (Member, IEEE), AND ALEX X. LIU[5] (Fellow, IEEE)

[1]School of Software, Nanjing University of Information Science and Technology, Nanjing 210044, China
[2]Nanjing University of Information Science and Technology, Wuxi Institute of Technology, Wuxi 214000, China
[3]School of Computer Science, Nanjing University of Information Science and Technology, Nanjing 210044, China
[4]Jiangsu Collaborative Innovation Center of Atmospheric Environment and Equipment Technology, Nanjing University of Information Science and Technology, Nanjing 210044, China
[5]Shandong Provincial Key Laboratory of Computer Networks, Shandong Computer Science Center (National Supercomputer Center in Jinan), Qilu University of Technology (Shandong Academy of Sciences), Jinan 250014, China

Corresponding author: Chengjun Zhang (e-mail:zhangcj5@gmail.com).

This work was supported in part by the National Natural Science Foundation of China under Grant 62071240, in part by the Natural Science Foundation of Jiangsu Province under Grant BK20231142, and in part by the Innovation Program for Quantum Science and Technology under Grant 2021ZD0302901.

ABSTRACT Text sentiment analysis is an important task in natural language processing and has always been a hot research topic. However, in low-resource regions such as South Asia, where languages like Bengali are widely used, the research interest is relatively low compared to high-resource regions due to limited computational resources, flexible word order, and high inflectional nature of the language. With the development of quantum technology, quantum machine learning models leverage the superposition property of qubits to enhance model expressiveness and achieve faster computation compared to classical systems. To promote the development of quantum machine learning in low-resource language domains, we propose a quantum–classical hybrid architecture. This architecture utilizes a pretrained multilingual bidirectional encoder representations from transformer (BERT) model to obtain vector representations of words and combines the proposed batch upload quantum recurrent neural network (BUQRNN) and parameter nonshared batch upload quantum recurrent neural network (PN-BUQRNN) as feature extraction models for sentiment analysis in Bengali. Our numerical results demonstrate that the proposed BUQRNN structure achieves a maximum accuracy improvement of 0.993% in Bengali text classification tasks while reducing average model complexity by 12%. The PN-BUQRNN structure surpasses the BUQRNN structure once again and outperforms classical architectures in certain tasks.

INDEX TERMS Natural language processing (NLP), quantum machine learning, quantum recurrent neural network.

I. INTRODUCTION

As one of the classical subfields of machine learning [1], [2, pp. 14–16], [3, pp. 5–25], natural language processing (NLP) [4], [5] has been a hot research topic in recent years. Text sentiment analysis (SA) [6], as a subtask of NLP, aims to classify text into positive and negative sentiment categories by detecting the polarity of the text. SA has been applied in various domains, including lexicon-based SA [7], machine-learning-based SA [8], and deep-learning-based SA [9, pp. 2–31]. Remarkable results have been achieved in SA for high-resource languages, such as English and Chinese [10], [11]. However, due to the complexity of language grammar, limited usage, and expensive computational resources, SA in low-resource languages has not been extensively explored. With the development of the Internet, a large influx of textual comments has made SA in low-resource languages feasible. In general, effective SA tasks can be achieved by combining good word-embedding models with efficient feature extraction models. In the case of studying word embeddings for Bengali texts, a significant challenge lies in capturing the rich expressions of sentiment present in the Bengali language. Due to the complexity of grammar rules, the extraction of features for sentiment classification becomes intricate as sentiment information may be expressed differently in sentences. The emergence of pretrained language models [12], [13] has improved feature extraction in Bengali sentiment classification tasks, as they can effectively capture sentiment information within Bengali sentences by learning rich language representations and contextual understanding.

Despite the positive role of pretrained language models in Bengali sentiment classification tasks, traditional feature extraction models still face efficiency challenges. Quantum deep learning [14], [15] combines the concepts of quantum computing and deep learning, leveraging the parallelism advantage [16] of quantum computing to accelerate the training and inference processes of models. By utilizing quantum neural networks (QNNs) and quantum gate operations, quantum deep learning models can handle complex sentiment classification tasks more efficiently [17]. A quantum–classical hybrid recurrent neural network (QRNN) model based on the variational quantum circuit (VQC) core was proposed in [18]. Such networks have been successfully applied as feature extractors in text classification tasks for high-resource languages, demonstrating better performance compared to their classical counterparts. Considering the characteristics of low-resource languages, this sparks the idea of using quantum algorithms to improve low-resource text SA tasks. However, previous studies have shown that QRNNs may struggle to effectively capture semantic information and may even result in information loss when dealing with longer sequences. The current challenges can be summarized as follows.

1) Due to the limitations of current noisy intermediate-scale quantum (NISQ) devices [19], it is necessary to match the dimensionality of the input sequence with the number of qubits. Previous QRNN models employed parameter-sharing linear layers to reduce the dimensionality of the input data, which may potentially result in the loss of semantic information to some extent.
2) Previous QRNN models did not optimize the VQC specifically but instead utilized parameter-sharing linear layers for optimization across all the VQCs.

In response to the first situation mentioned above, we designed and utilized a batch uploading quantum neural network (BUQNN), which is essentially a structure that incorporates VQCs. The BUQNN divides the input feature sequence into batches and loads them into the circuit to obtain the complete semantic information. We refer to the QRNN model that utilizes this BUQNN as the batch uploading quantum recurrent neural network (BUQRNN). By adopting this approach, we can alleviate the semantic information loss caused by previous methods with an S number of qubits.

Regarding the second situation mentioned, in [20], a nonparameter-sharing linear layer was applied after the VQC to enhance the expressiveness of the circuit. We followed this idea and made improvements by using nonparameter-sharing linear layers both before and after each VQC. This allows for independent optimization of each VQC, which is advantageous for the model.

Our contributions are as follows.

1) We proposed a BUQRNN specifically designed for sequential data. This method requires only an S number of qubits and mitigates the loss of semantic information caused by previous approaches.
2) We introduced a parameter nonshared batch upload quantum recurrent neural network (PN-BUQRNN) that employs independent linear layers for each VQC, enabling independent optimization of each VQC in the model structure.
3) For the first time, we applied quantum algorithms to address sentiment classification tasks in low-resource languages, such as Bengali, which holds significant importance for advancing the development of quantum in low-resource languages.

The rest of this article is organized as follows. Section II focuses on the text classification process and discusses word embedding techniques and the QRNN for low-resource language text classification. Section III presents the specific implementation approach to address the aforementioned issues. Section IV describes the numerical simulation results. Finally, Section V concludes this article, providing insights into future directions and prospects.

II. RELATED WORK

Sazzed et al. [21] employed a combination of multilingual bidirectional encoder representations from transformers (MBERT) embeddings and RNN for text classification tasks in Bengali. In this section, we will introduce an improved quantum–classical hybrid model based on its architecture. We will discuss various methods for text classification in the low-resource language domain and provide an overview of Bengali-specific word-embedding techniques and the QRNN model.

A. WORD EMBEDDING

In the domain of high-resource languages, there are generally two methods for generating word vectors: contextual word-embedding techniques and noncontextual word-embedding techniques. BERT, as a context-based pretrained language model, can also be employed for word embedding. In [22], BERT is utilized to obtain word vectors for English text, followed by feature extraction using the quantum temporal convolutional network (TCN). The results demonstrate the high performance of BERT while also indicating the feasibility of related quantum models in text classification tasks. Similar to the high-resource language domain, low-resource languages can be categorized into two main types in terms of word-embedding methods. The first type is noncontextual word-embedding methods. Al-Amin et al. [23] used the Word2vec model to generate Bengali word vectors. However, Word2vec does not handle subword information, such as affixes, and does not consider spelling characteristics of words. In contrast to Word2Vec, GloVe [24] considers both local context and global statistical information to generate word vectors. FastText [25], an open-source word-embedding and text classification library developed by Facebook AI Research, differs from Word2Vec and GloVe in that it considers subword information within words. In contrast to noncontextual word embeddings, the second type, contextual word embeddings, takes into account the context of words in specific sentences or texts, enabling better capture of word semantics. MBERT [26], as a multilingual version of the BERT model, has been pretrained on Wikipedia texts from 104 languages. An important feature of MBERT is its bidirectional self-attention mechanism using the Transformer model, which allows for a better understanding of the complex semantics of words within their context. Experimental results [21] demonstrate that compared to traditional word embeddings, such as FastText and GloVe, MBERT performs better in Bengali word embeddings. In addition to employing classical methods for text word embedding as mentioned above, Li et al. [27] propose a novel technique for text word embedding using a quantum language model. The results indicate that word vectors mapped by the quantum language model can achieve performance comparable to their classical counterparts in downstream tasks. In this article, we will choose MBERT as the word embedding model for Bengali text. The sentence vector x can be represented as $\mathbf{x} = (x_1, x_2, x_3, \ldots, x_t)$ after tokenization, where x_t represents the tth word in the sentence. After passing through the MBERT model, the corresponding BERT vector representation can be obtained as follows:

$$\mathbf{MBERT}(x_1, x_2, x_3, \ldots, x_t) = (e_1, e_2, e_3, \ldots, e_t). \quad (1)$$

Here, each e_t represents the MBERT vector representation of the sentence with index j at time t.

B. QUANTUM–CLASSICAL HYBRID RECURRENT NEURAL NETWORK

Quantum long short-term memory (QLSTM) [28] is a variation of the QRNN that incorporates a VQC [29] to construct quantum-gated units. The VQC adjusts the initial states of qubits through a series of rotation gates, performs quantum entanglement operations using a series of CNOT gates [30], and finally collapses them into classical states through measurement operations. The parameters of the rotation gates need to be updated through gradient descent. Fig. 1 illustrates the structure of the VQC included in the QLSTM network. In this case, we assume that the circuit is simulated with four qubits. The input word vector sequence $\vec{v_t} = (v_t^1, v_t^2, \ldots, v_t^n)$, where $\vec{v_t}$ is composed of the current MBERT output e_t and the previous hidden layer feature h_{t-1}. Initially, a linear layer is used to match the number of

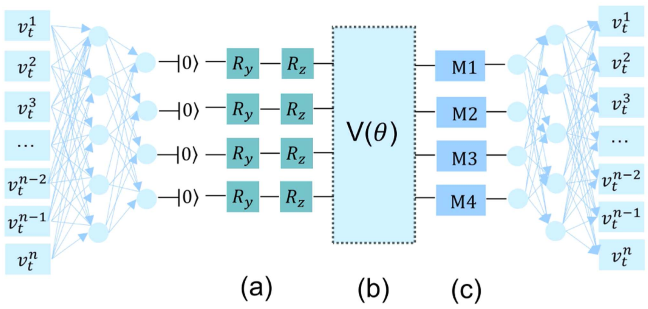


FIGURE 1. VQC structure used in QLSTM consists of (a) an encoding layer, (b) a variational layer, and (c) a measurement layer. The encoding layer is composed of a series of rotation gates used for encoding.

qubits, and the encoding layer utilizes angle encoding [31] to embed the input sequence into the circuit. The variational layer optimizes the rotation angles of the qubits on the Bloch sphere [32] using a series of rotation gates with updatable parameters. The measurement part employs Pauli-Z gates to measure the states of qubits. Measuring in the Pauli-Z basis $\sigma_z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$ means projecting the state onto one of the eigenstates of the Pauli-Z matrix, namely, $|0\rangle$ or $|1\rangle$. The measured classical bits are then expanded to match the size of the hidden layer through another linear layer. However, when $I > n$ (where n is the number of qubits and I is the length of the feature sequence), the current approach typically reduces the input dimensionality to match the number of qubits through a linear layer, which may result in the loss of semantic information to some extent. We draw inspiration from a quantum algorithm for image classification on the MNIST handwritten dataset [33], which differs from the previous data reuploading approach [34]. The former method slices the image horizontally and uploads the value of each pixel to the quantum circuit, while the latter repeatedly uploads the feature vector to the circuit. Due to the limitations of current NISQ devices, the latter approach requires significant computational resources for processing high-dimensional feature sequences, making it difficult to implement. The former approach, by batch uploading, reduces the dependence on the number of qubits, and fully loads the feature sequence into the circuit. We aim to address the first issue mentioned in the introduction of traditional QRNN using this method.

In addition, the parameters of the linear embedding layer before each VQC and the linear expansion layer after each VQC in the traditional QRNN are shared. Taking QLSTM as an example in [28], the model uses four VQCs to replace the classical network layers, and these four VQCs share the embedding and expansion layers. However, this shared linear layer cannot be effectively optimized for different VQCs. In [20], nonshared linear layers with separate parameters were added after each VQC in QLSTM to achieve independent optimization of the circuits. We were inspired by this and made improvements by incorporating nonshared parameter linear layers before and after each VQC, along with the aforementioned batch uploading approach, to achieve more

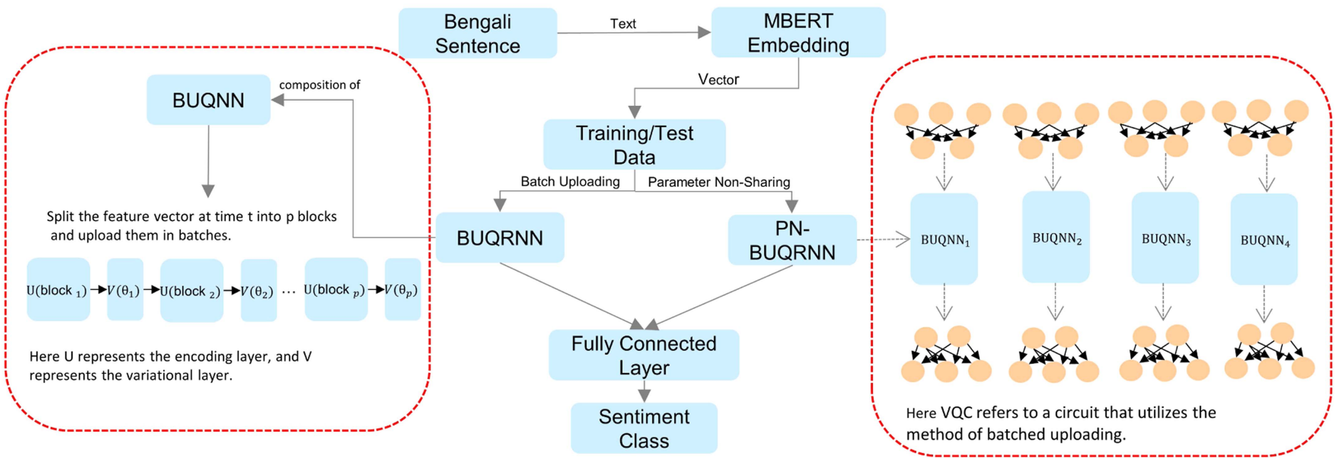

Bengali Sentence
Text
MBERT Embedding
Vector
Training/Test Data
Batch Uploading
Parameter Non-Sharing
BUQRNN
PN-BUQRNN
Fully Connected Layer
Sentiment Class
BUQNN
composition of
Split the feature vector at time t into p blocks and upload them in batches.
U(block 1) → V(θ1) → U(block 2) → V(θ2) ··· U(block p) → V(θp)
Here U represents the encoding layer, and V represents the variational layer.
BUQNN1
BUQNN2
BUQNN3
BUQNN4
Here VQC refers to a circuit that utilizes the method of batched uploading.

FIGURE 2. Framework includes two red dashed sections, showcasing the proposed structures designed to address the mentioned issues.

comprehensive circuit optimization and further improve the performance of the QRNN.

III. QRNN WE PROPOSE

Regarding the first issue mentioned in Section I, in Sections III-A and III-B, we demonstrate the replacement of classical neural networks in classical RNNs with a BUQNN and apply it to the QRNN, proposing a BUQRNN based on a classical-quantum hybrid framework. Unlike the traditional QRNN, which requires matching the number of qubits to the feature vectors, our approach provides universality and effectiveness for handling higher dimensional feature vectors. The BUQNN divides input features into batches according to a predefined number of qubits and passes them through a VQC, forming an n-layer encoding-variational hybrid structure. This approach allows for processing sequence data without reducing the dimensionality of feature vectors and does not require a large number of qubits to handle the data. Addressing the second issue mentioned in Section I, Section III-C provides a specific solution. The workflow for the sentiment classification task in Bengali language is shown in Fig. 2. The input text is transformed into word embeddings using MBERT, and then, the word vectors are fed into the proposed BUQRNN or PN-BUQRNN for feature extraction. Finally, the extracted features are input into a fully connected layer for classification. We will now describe the implementation details of the BUQRNN and the PN-BUQRNN.

A. BATCH UPLOADING QUANTUM NEURAL NETWORK

In the QNN, the encoding gate, decoding gate, and variational gate are further divided into encoding layers, decoding layers, and variational layers, respectively. The selection of encoding gates is based on the chosen encoding method and the number of input features. The optimal choice of encoding method is crucial for successful learning of the QNN model. We implement the BUQNN using a multilayer encoding-variational structure. The left portion of Fig. 2 illustrates the BUQNN structure that we employ. We divide the features

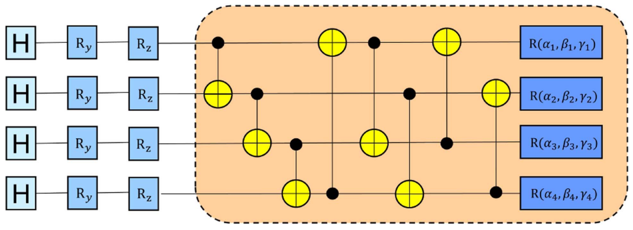


FIGURE 3. Encoding layer circuit and the variational layer structure that we use.

$\vec{v_t} = (v_t^1, v_t^2, \ldots, v_t^n)$ into batches based on the number of qubits, such that the number of batches is $n/N = p$. Here, N represents the number of qubits, and n represents the dimensionality of the input word vector features. This assumes a scenario where the division is evenly divisible. If it is not divisible, the number of batches p needs to be increased by 1, and the remaining space is padded with zero elements. After the division, $\vec{v_t} = (\vec{batch}_1, \vec{batch}_2, \ldots, \vec{batch}_p)$, where $\vec{batch}$ is a vector containing N features. In this demonstration, we use four qubits, and there is one variational layer between any two encoding layers. This restriction is just a design choice, and alternative design schemes can be chosen. Each feature batch undergoes angle encoding embedding circuit and then a variational layer. One feature batch is uploaded to the circuit at a time, and after loading p times, all the feature vectors can be embedded in the circuit. It is important to note that the essence of the BUQNN is a VQC using batch uploading, which differs from the structure of the VQC included in traditional QLSTM networks. It can directly embed feature vectors into the circuit without the need for additional linear layers. The BUQNN with linear layers will be mentioned in Section III-D of this chapter.

Fig. 3 illustrates the encoding layer circuit and the variational layer structure that we utilize. In the simulation with n qubits, let us consider the example of a batch vector $\vec{batch}1 = (v_t^1, v_t^2, \ldots, v_t^N)$, where $1 \leq i \leq N$. For each v_t^i, we generate angles $\theta_{i,1} = \arctan(v_t^i)$ and $\theta_{i,2} = \arctan(v_t^{i^2})$, resulting in a total of $2i$ rotation angles. $\theta_{i,1}$ is applied using the $Ry(\theta_{i,1})$

gate for rotation around the y-axis, while $\theta_{i,2}$ is applied using the $Rz(\theta_{i,2})$ gate for rotation around the z-axis. The encoded data are in a quantum state and undergo a series of unitary operations, including multiple CNOT gates and single-qubit rotation gates. $R(\theta, \alpha, \gamma)$ represents a general parameterized single-qubit rotation gate and can be expressed as

$$\mathbf{R}(\theta_1, \theta_2, \theta_3) = \begin{pmatrix} e^{i\theta_2}\cos(\theta_1) & e^{i\theta_3}\sin(\theta_1) \\ -e^{i\theta_3}\sin(\theta_1) & e^{-i\theta_2}\cos(\theta_1) \end{pmatrix}.$$

Assume that a quantum circuit consists of four qubits. The aforementioned process of quantum state changes can be summarized as follows.

1) First, the initial state of four qubits in the quantum circuit is denoted as $|\psi_0\rangle$, which is typically a pure state with all the qubits in the ground state $|0\rangle$, i.e., $|\psi_0\rangle = |0000\rangle$.
2) Next, $\vec{batch}_1$ is encoded into $|\psi_0\rangle$ to form a new quantum state $|\psi_1\rangle$. The encoding operation uses R_Y and R_Z gates, the rotation angles of which are determined by the elements in $\vec{batch}_1$. We denote this operation as $U(\vec{batch}_1)$, so the quantum state becomes $|\psi_1\rangle = U(\vec{batch}_1)|\psi_0\rangle$.
3) In this step, additional quantum operations, $V(\theta_1)$, are applied to $|\psi_1\rangle$ to introduce more complex quantum correlations, generating a new quantum state $|\psi_1'\rangle = V(\theta_1)|\psi_1\rangle = V(\theta_1)U(\mathbf{e}_1)|\psi_0\rangle$. Here, $V(\theta_1)$ is a variational layer controlled by the parameter θ_1.
4) Repeat steps 2 and 3, encoding $(\vec{batch}_2, \ldots, \vec{batch}_p)$ into the respective quantum states and performing unitary operation $V(\theta_i)$ after each encoding, where $i \in 1, 2, \ldots, p$. The resulting series of quantum states are $|\psi_2\rangle, \ldots, |\psi_p\rangle$ and their variational states $|\psi_2'\rangle, \ldots, |\psi_p'\rangle$. For example, for $\vec{batch}_2$, the encoded state is

$$\begin{aligned} |\psi_2\rangle &= U\left(\vec{batch}_2\right)|\psi_1'\rangle \\ &= U\left(\vec{batch}_2\right)V(\theta_1)U\left(\vec{batch}_1\right)|\psi_0\rangle \end{aligned}$$

then applying the unitary operation of the variational layer $V(\theta_2)$ results in $|\psi_2'\rangle = V(\theta_2)|\psi_2\rangle = V(\theta_2)U(\vec{batch}_2)V(\theta_1)U(\vec{batch}_1)|\psi_0\rangle$.
5) Finally, perform the measurement operation M on the quantum state $|\psi_p'\rangle$, and compute the expectation value of the measurement result, which is usually obtained by random sampling in the Pauli basis. The expectation E can be represented as $E = \langle\psi_p'|M|\psi_p'\rangle$, and the obtained expectation value is used for subsequent calculations.

B. BATCH UPLOADING QUANTUM LONG SHORT-TERM MEMORY (BUQLSTM)

Similar to QLSTM, we replace the classic neural network in LSTM with the aforementioned BUQNN. Fig. 4 shows the BUQLSTM network we proposed, which consists of four BUQNNs. The expectation values output by the BUQNN are combined in the long short-term memory (LSTM) network after passing through nonlinear activation functions, such as tanh and sigmoid, to update the values of various gating units. The calculation process of four BUQNN units is as follows:

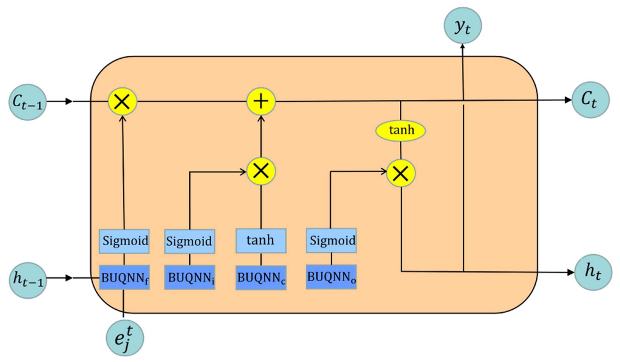


FIGURE 4. **Our proposed BUQLSTM.**

$$f_t = \sigma(BUQNN_f) \tag{2}$$
$$i_t = \sigma(BUQNN_i) \tag{3}$$
$$\tilde{c}_t = \tanh(BUQNN_c) \tag{4}$$
$$c_t = f_t * c_{t-1} + i_t * \tilde{c}_t \tag{5}$$
$$o_t = \sigma(BUQNN_o) \tag{6}$$
$$h_t = o_t * \tanh(c_t). \tag{7}$$

The BUQLSTM that we propose uses four BUQNNs, represented by $BUQNN_n (n \in f, i, c, o)$. Through the above calculations, the LSTM network can obtain the hidden state h_t and the cell state c_t at time step t.

Algorithm 1 outlines the numerical computation process of BUQLSTM. Initially, within each gate unit, the input vector "inputs" is partitioned into several "batch" vectors. Subsequently, these vectors are sequentially embedded into the quantum circuit following the encoding-variational order. The "weights" will be updated as part of the subsequent optimization process. Finally, the expectation values of Pauli-Z operators on the relevant qubits are calculated, and the results are returned in the form of a list for further computations. Here is the explanation for the four BUQNN modules used in BUQLSTM.

1) $BUQNN_f$: $BUQNN_f$ obtains the vector f_t by combining a sigmoid function and maps the expectation value to the interval [0,1]. f_t is a crucial component of the BUQLSTM network, with its output shown in (2). It operates on c_{t-1} based on $f_t * c_{t-1}$, meaning that it decides whether to "forget" or "retain" the corresponding elements from the previous cell state c_{t-1}. For instance, values of 1 or 0 indicate that the corresponding elements will be entirely retained (forgotten). Typically, the f_t vector affects cell state values between 0 and 1, indicating that only part of the information will

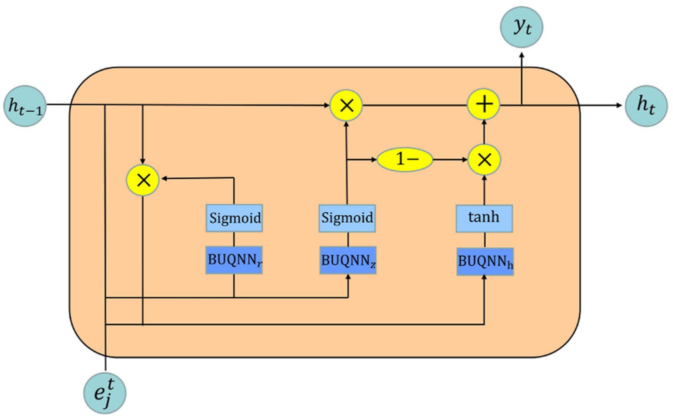


FIGURE 5. Our proposed BUQGRU.

be retained. Its function is vital for learning and modeling time dependence.

2) $BUQNN_i$ and $BUQNN_c$: First, the $BUQNN_i$ module processes the input data v_t and outputs a set of values between 0 and 1 through a Sigmoid function, determining which information can be added to the current cell state. Simultaneously, the $BUQNN_c$ module also processes the same input data and generates a new cell state candidate $\widetilde{c_t}$ through a tanh function [as shown in (4)]. Equation (5) combines the output of the input gate with the forget gate f_t, the cell state of the previous moment c_{t-1}, and the new cell state candidate $\widetilde{c_t}$, and the resulting vector will be used to update the current cell state. In other words, the output of the input gate (a real number between 0 and 1) determines how much of the new information $\widetilde{c_t}$ is added to the current cell state c_t. This mechanism allows LSTM to better remember long-term dependencies, avoiding the vanishing gradient problem of ordinary RNNs.

3) $BUQNN_o$: The goal of $BUQNN_o$ is to generate the output of the cell. In (6), O_t obtains its output through the sigmoid function after obtaining the expectation value from $BUQNN_o$. In (7), the output o_t is multiplied elementwise with the output of the update gate c_t (which is processed through a tanh activation function), generating the new hidden state vector h_t, which will be passed to the next time step for calculation.

C. BATCH UPLOADING QUANTUM GATED RECURRENT UNIT (BUQGRU)

Above, we have detailed the structure of BUQLSTM. Similar in principle to BUQLSTM, in this section, we only provide a brief introduction to the BUQGRU network. Fig. 5 depicts our BUQGRU network, where we replace the classical neural network of the gated recurrent unit (GRU) model with the BUQNN. Compared to BUQLSTM, it only requires three BUQNNs. The current input e_j^t and the previous moment's h_{t-1} are fed into the network. The reset gate, composed of

Algorithm 1: Algorithm for BUQLSTM.

```
BUQLSTM(INPUT_SIZE, HIDDEN_SIZE)
inputs = **concatenate**(input_size, hidden_size)
**forget gate** : device$_f$ = **device**(backend, wires = $w_f$)
#circuit_forget(inputs, weights):
  split inputs into$p$ batches
  for batch in $p$:
    **encoding**(batch, wires = $w_f$)
    **variation**(weights$_f$, wires = $w_f$)
  return ([**Expectation**(**PauliZ**(wire) for each wire])
**input gate** : device$_i$ = **device**(backend, wires = $w_i$)
#circuit_input(inputs, weights):
  split inputs into $p$ batches
  for batch in $p$:
    **encoding**(batch, wires = $w_i$)
    **variation**(weights$_i$, wires = $w_i$)
  return ([**Expectation**(**PauliZ**(wire) for each wire])
**update gate** : device$_c$ = **device**(backend, wires = $w_c$)
#circuit_update(inputs, weights):
  split inputs into $p$ batches
  for batch in $p$:
    **encoding**(batch, wires = $w_c$)
    **variation**(weights$_c$, wires = $w_c$)
  return ([**Expectation**(**PauliZ**(wire) for each wire])
**output gate** : device$_o$ = **device**(backend, wires = $w_o$)
#circuit_output(inputs, weights):
  split inputs into $p$ batches
  for batch in $p$:
    **encoding**(batch, wires = $w_o$)
    **variation**(weights$_o$, wires = $w_o$)
  return ([**Expectation**(**PauliZ**(wire) for each wire])
```

$BUQNN_r$ and the sigmoid activation function, determines how much of the previous moment's hidden state information should be used when calculating the current candidate hidden state. The update gate, comprised of $BUQNN_z$ and the sigmoid function, determines how much of the previous moment's hidden state information should be preserved when calculating the current hidden state. Next, $BUQNN_z$ combined with the tanh activation function is used to calculate the candidate hidden state $\widetilde{h_t}$. Finally, (11) decides whether the new hidden state h_t should fully accept the candidate hidden state, retain the previous moment's hidden state, or be a compromise between the two

$$r_t = \sigma(BUQNN_r) \tag{8}$$

$$z_t = \sigma(BUQNN_z) \tag{9}$$

$$\widetilde{h_t} = \tanh(BUQNN_h) \tag{10}$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \widetilde{h_t}. \tag{11}$$

Here, $BUQNN_i(i \in r, z, h)$ represent the reset gate circuit, update gate circuit, and candidate hidden state circuit, respectively. It is important to note that, due to the characteristics

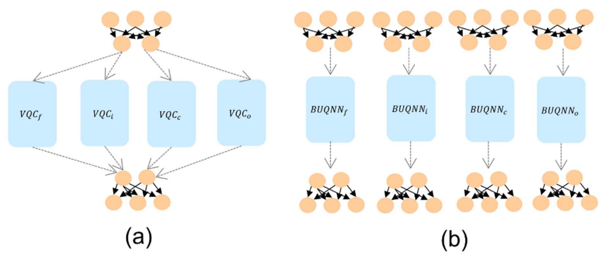


FIGURE 6. (a) Parameter-shared VQC. (b) Proposed PN-BUQNN.

TABLE 1. Datasets Used in the Experiment are the BOOK-Reviews Dataset and the YouTubeB-S Dataset

Dataset	Positive	Negative	All
BOOK-Reviews	996	1004	2000
YouTubeB-S	1005	995	2000

TABLE 2. Comparison of the Number of Model Parameters Used

Model	Number of Quantum Gates	Number of Classical Parameters	All
LSTM	0	224	224
GRU	0	168	168
QLSTM	96	72	168
QGRU	72	72	144
BUQLSTM	144	20	164
BUQGRU	108	20	128

TABLE 3. Accuracy of Structures with BUQRNN and Baseline Structures on BOOK-Reviews and YouTubeB-S Validation Sets

Model	BOOK-Reviews	YouTubeB-S
MBERT+LSTM	84.231	92.394
MBERT+QLSTM	82.235	91.266
MBERT+BUQLSTM	**83.121**	**92.037**
MBERT+GRU	84.524	92.214
MBERT+QGRU	83.857	90.735
MBERT+BUQGRU	**84.227**	**91.728**

of the GRU network, the input to $BUQNN_h$ differs from other $BUQNN$. In the computation of (10), output r_t of the reset gate is multiplied by the hidden state output h_{t-1} from the previous time step. This product determines how much information from the previous time step can be utilized. The resulting product is concatenated with e_j^t and serves as input to $BUQNN_h$. After passing through the tanh activation function, a new candidate hidden state is obtained.

D. PARAMETER NONSHARING BUQNN

In this section, we will introduce the proposed PN-BUQNN to enhance the learning capability of the BUQNN. As shown in Fig. 6(a), in traditional QLSTM networks, the linear embedding layers before and after each VQC are nonshared parameters. While this reduces the parameter count of the model, it may introduce some issues. By using the same linear transformation for all the VQCs, the model's ability to learn different input data features could be limited. For instance, if the forget gate and the input gate need to extract different features from the input data, a model with shared parameters may struggle to simultaneously satisfy the requirements of both the gates. Furthermore, during the training process, all the VQCs backpropagate through the same linear layer. This can lead to gradient explosion or vanishing gradients. If the gradient of a particular VQC is exceptionally large, it may "overwhelm" the gradients of other circuits, making it difficult for the entire network to learn effectively. To address the aforementioned issues, we propose the PN-BUQNN, as depicted in Fig. 6(b).

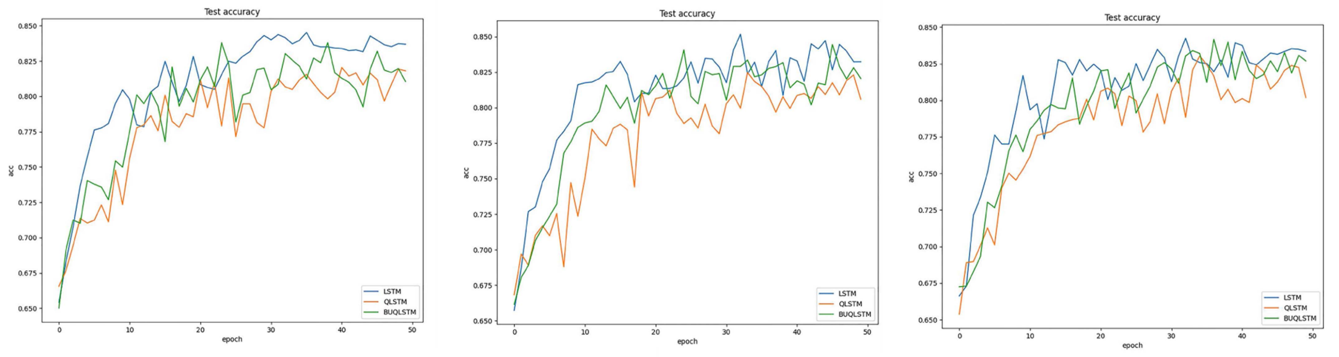
TABLE 4. Accuracy of the Structure With PN-BUQRNN Compared to the Baseline Structure on the YouTubeB-S and BOOK-Reviews Validation Sets

Model	BOOK-Reviews	YouTubeB-S
MBERT+LSTM	84.231	92.394
MBERT+PN-QLSTM	84.266	91.416
MBERT+PN-BUQLSTM	**84.738**	**92.681**
MBERT+GRU	84.524	92.214
MBERT+PN-QGRU	84.343	90.762
MBERT+PN-BUQGRU	**85.304**	**92.690**

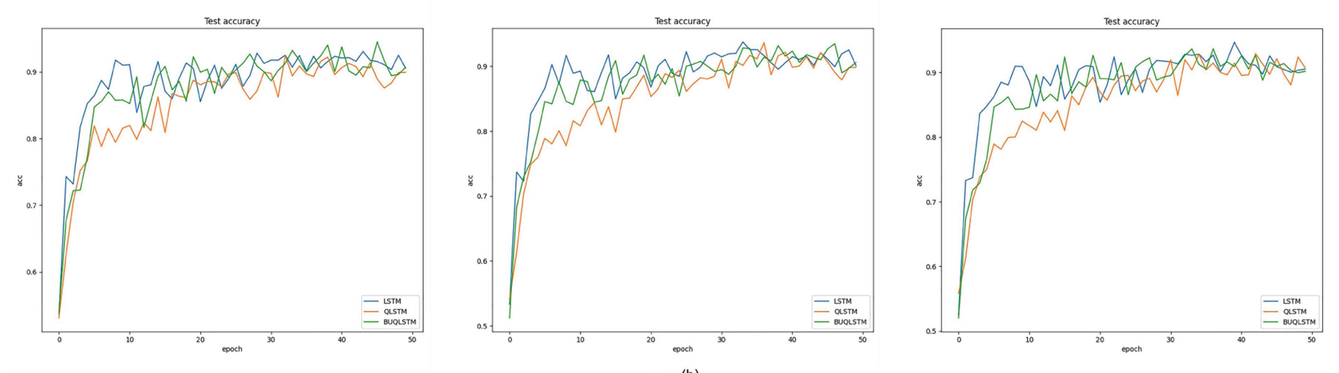
TABLE 5. PN-BUQRNN Structure With Two Types of Parameter-Unshared Circuits, Where Superscript (*a*) Refers to (a) in Fig. 9 and Superscript (*b*) Refers to (b) in Fig. 9

Model	BOOK-Reviews	YouTubeB-S
PN-BUQLSTM$^{(a)}$	**84.547**	**92.036**
PN-BUQGRU$^{(a)}$	**85.047**	**91.644**
PN-BUQLSTM$^{(b)}$	84.232	89.925
PN-BUQGRU$^{(b)}$	82.435	91.356

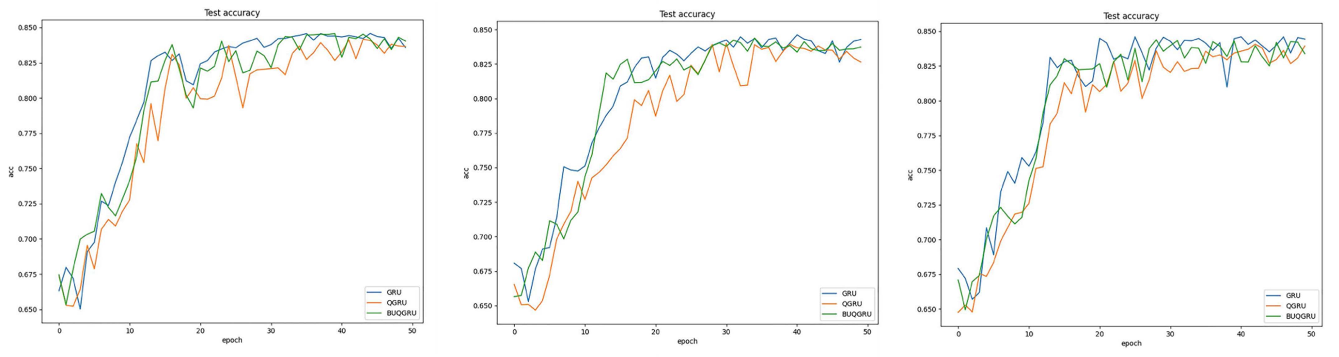
We employ linear layers before and after each BUQNN, which means that they can learn more appropriate input and output transformations for themselves. The independent linear layers imply that each BUQNN can learn and extract different features, with optimizations being specific to each BUQNN. During the training process, since each BUQNN has its own linear layer, its gradient updates will no longer affect each other. Furthermore, adding a linear layer before $BUQNN$ can, to some extent, alleviate the issue of gradient vanishing. The issue of gradient vanishing is discussed in low-qubit VQCs [35], and a similar problem exists in the encoding layer of $BUQNN$. The input feature vector v_t^i generates different rotation angles for the parameters of R_y and R_z gates using the arctan function. For the input $\vec{batch}1$, the encoding layer of $BUQNN$ can be represented as: $encoding(\vec{batch}_1) = R_z\left(\arctan(v_t^1)^2\right) R_y\left(\arctan\left(v_t^1\right)\right)|0\rangle \ldots R_z\left(\arctan\left(v_t^N\right)^2\right) R_y\left(\arctan\left(v_t^N\right)\right)|0\rangle$. It can be observed that the encoding layer of the BUQNN involves the use of arctan, whose derivative may lead to gradient vanishing. For instance, the



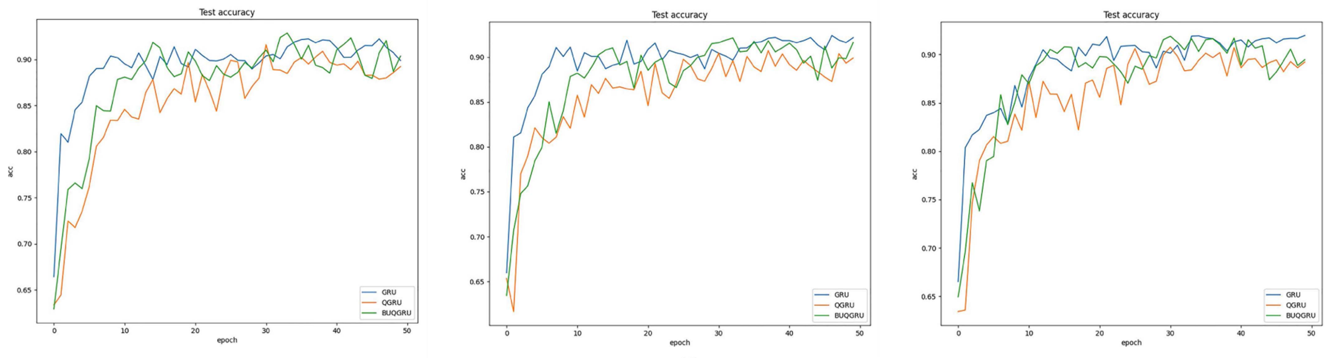
(a)



(b)



(c)



(d)

FIGURE 7. Experimental result graph. (a) Comparison results of different LSTM models on BOOK-Reviews. (b) Comparison results of different LSTM models on YouTubeB-S. (c) Comparison results of different GRU models on BOOK-Review. (d) Comparison results of different GRU models on YouTubeB-S.

derivative of $R_y(\arctan(v_t^1))$ is given as follows:

$$\frac{\partial R_y\left(\arctan\left(v_t^1\right)\right)}{\partial w_i} = -R_y\left(\arctan\left(v_t^1\right)\right)^T \frac{\partial \arctan\left(v_t^1\right)}{\partial w_i} \tag{12}$$

$$\frac{\partial \arctan\left(v_t^1\right)}{\partial w_i} = \frac{1}{1+\left(v_t^1\right)^2}\frac{\partial v_t^1}{\partial w_i}. \tag{13}$$

Here, v_t^1 is an element in $\vec{batch}_1$, and w_i denotes the updatable rotation gate parameters. From (13), it is evident that the value of v_t^1 should not be too large or too small, as it may lead to gradient vanishing. Therefore, a clip operation on the output can be added after the linear layer preceding BUQNN, restricting the output within a specified range. Through this approach, it is possible to optimize the weights of the linear transformation effectively, thus mitigating gradient vanishing to a certain extent. In subsequent experiments, we will use such a linear layer to control the output within the range of $[-3, 3]$.

IV. DATASET AND EXPERIMENTAL RESULT

A. DATASET

The experiments utilized two Bengali text classification datasets. The BOOK-Reviews dataset [36] is a collection of Bengali book reviews gathered from the Internet (such as blogs, Facebook, and e-commerce websites). This dataset is a binary classification dataset (with positive and negative classes) containing 2000 book reviews. The other dataset used in the experiment is YouTube-B [21], a collection of comments on Bengali dramas collected from the YouTube site. It contains 11 807 comments, of which 8500 are positive and 3307 are negative. Given the limitations of current NISQ devices on the efficiency of quantum algorithms in classification tasks, we selected 2000 entries from the YouTube-B dataset for the experiment, with 1700 for training and 300 for testing. We named the modified YouTubeB dataset as YouTubeB-S. The average number of words per sentence in the BOOK-Reviews and YouTubeB-S datasets is 46 and 21, respectively. As shown in Table 1, to prevent long-tail distribution in the data, the number of data entries in each category was kept approximately equal. The dataset we used is available online.[1]

In the subsequent experimental process, we will use these two datasets to test the two structures shown on both the sides of Fig. 2.

B. STRUCTURE WITH THE BUQRNN

In this section, we conducted experiments on the BUQRNN architecture shown in Fig. 2. We employed the BUQNN to define BUQLSTM and BUQGRU and conducted experiments using a four-qubit circuit. For comparison, we also constructed a traditional VQC with four qubits. In (1), the input vector is passed through the MBERT model, resulting in the embedding vector e_j^t. To facilitate quantum simulation, we combined MBERT with a linear layer to control the feature size to eight dimensions. We also compared it with a classical LSTM network with an input size of eight dimensions and 224 parameters. All the experiments used a hidden layer size of four dimensions, and the feature dimensions combined with the hidden layer dimensions were fed into the model. For a fair comparison, the depth of the VQC in the traditional QRNN was set to 2. Zhang and Zhuang [37] mention that when the architecture of the VQC is extensive—when the circuit depth does not decrease the number of gates—the quantum data classification errors of VQC typically decrease exponentially with the increase in circuit depth. This rapid error suppression ends when reaching the final Helstrom limit of quantum state discrimination. However, considering the limitations of NISQ devices, it is challenging to train more parameters. In future work, we plan to further explore the impact of VQC depth on classification accuracy by improving experimental design and adopting more advanced quantum devices. The model parameter counts used in the experiments are shown in Table 2. The experiments were conducted using the AdamW optimizer and the cross-entropy loss function. The PennyLane framework was used for modeling the quantum circuits, which includes multiple built-in simulators to meet different task requirements.

All the aforementioned experiments were conducted under the conditions of a learning rate of 0.01, 50 epochs, and the use of linear warm-up optimization method. To enhance the persuasiveness of the experimental results, we conducted ten experiments for each trial using datasets divided into different partitions to obtain the mean accuracy. Due to space constraints, we present only the first three results for each experiment. Fig. 7 presents the comparison results of different models on two datasets, where the highest accuracy is shown in Table 3. Overall, while quantum structures have disadvantages compared to classical structures, the BUQRNN-based structures show improved accuracy compared to traditional QRNN structures on both the datasets. This is because the traditional QRNN utilizes linear layers to reduce the dimensionality of the input feature sequence during circuit simulation, which may result in a loss of semantic information to some extent.

In contrast, the BUQRNN adopts the approach of uploading the complete sequence into the circuit in batches. On the YouTube dataset, the BUQLSTM structure and the BUQGRU structure achieve improvements of 0.771% and 0.993%, respectively, compared to the QLSTM structure and the quantum gated recurrent unit (QGRU) structure. On the BOOK-Reviews dataset, the BUQLSTM structure and the BUQGRU structure achieve improvements of 0.886% and 0.370%, respectively, compared to the QLSTM structure and the QGRU structure. Although the improvements in accuracy are limited, our proposed BUQRNN model has fewer parameters compared to the QRNN and the classical RNN, making it more suitable for low-resource language domains.

[1]https://github.com/nuistyl/Bengali-dataset

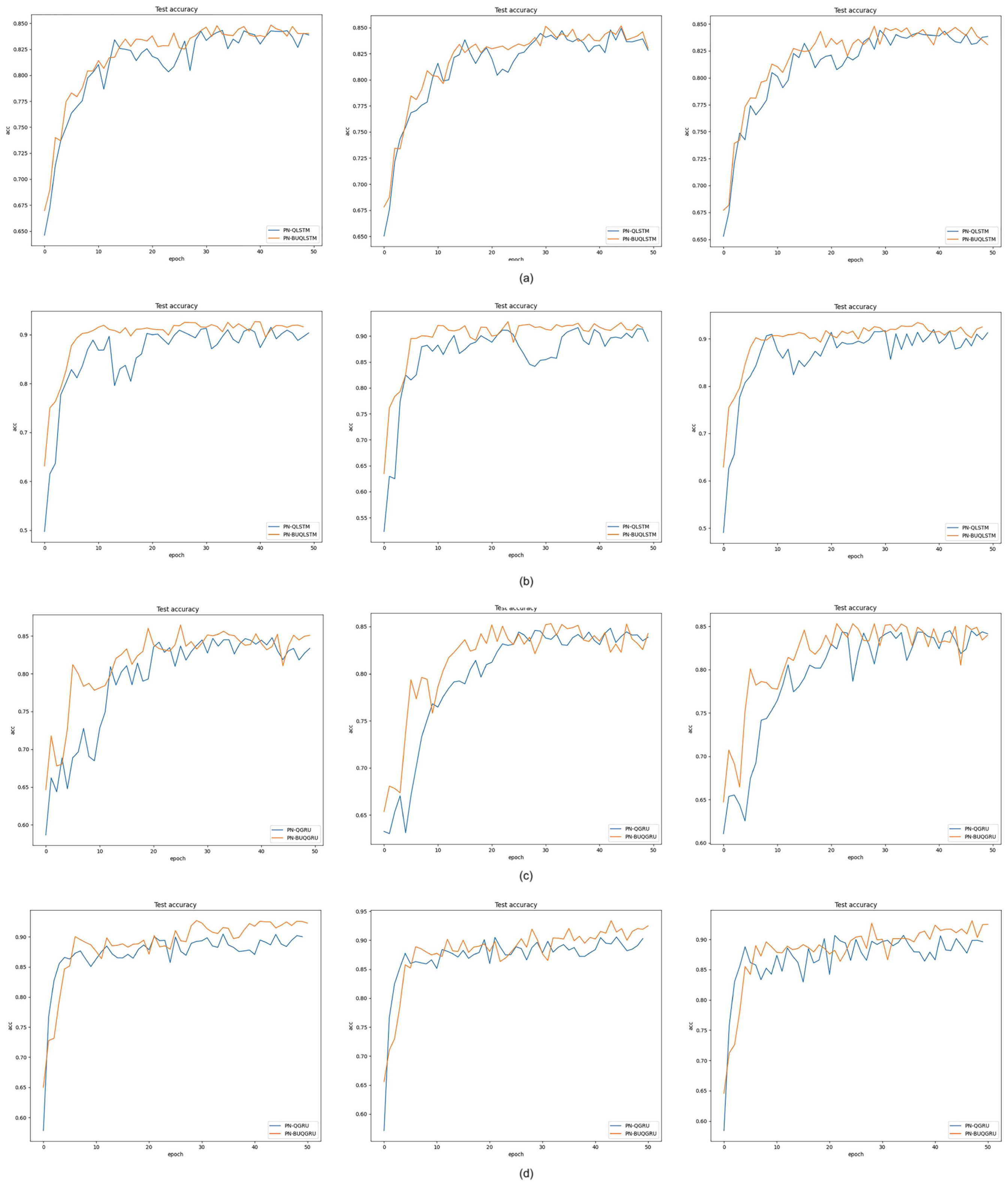


FIGURE 8. Experimental result graph. (a) Comparison results of LSTM models with parameter nonsharing on BOOK-Review. (b) Comparison results of LSTM models with parameter nonsharing on YouTubeB-S. (c) Comparison results of LSTM models with parameter nonsharing on BOOK-Review. (d) Comparison results of LSTM models with parameter nonsharing on YouTubeB-S.

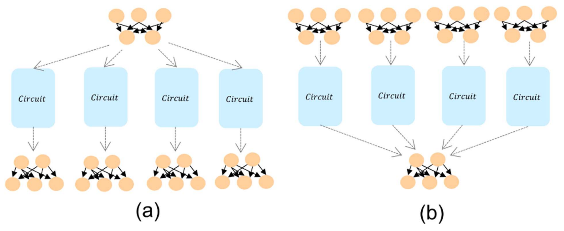


FIGURE 9. (a) and (b) Two modes for parameter nonshared circuits.

C. STRUCTURE WITH THE PN-BUQRNN

In the previous section, we presented the results of BUQRNN. In this section, we tested the structure equipped with the PN-BUQRNN, in which independent linear layers were used before and after each BUQNN. For comparative experiments, to maintain consistency with the structure of the PN-BUQRNN, we set the depth of the VQC in the traditional QRNN to 3 and also used independent linear layers before and after each VQC, denoted as PN-QRNN.

The experimental settings remained consistent with the previous section. Fig. 8 illustrates the comparison results of using the parameter nonshared LSTM model and the GRU model on the two datasets. The highest accuracy is shown in Table 4. Due to space constraints, we present only the results of the first three runs for each experiment. Overall, the PN-BUQRNN structure outperforms the classical RNN structure and the PN-QRNN structure on both the datasets. After applying the parameter nonshared VQC, the PN-QRNN structure even performs worse in certain tasks. This is due to the information loss caused by the dimensionality reduction of the linear layer, which prevents it from working effectively. In contrast, in the PN-BUQRNN structure, we fully load the information into the BUQNN circuit and optimize each BUQNN circuit independently, resulting in better results.

D. ABLATION STUDY ON PARAMETER NONSHARING CIRCUITS

In previous experiments, we constructed models with parameter nonsharing by using separate linear layers before and after each quantum circuit. However, there is still a question to be verified. As shown in Fig. 9, what would be the result if we only use parameter nonsharing linear layers at the input end of the quantum circuit, or only at the output end?

We tested the PN-BUQRNN structure with two modes shown in Fig. 9(a) and (b). Fig. 10 and Table 5 present the experimental results, indicating a decreasing trend in accuracy for both the modes on the two datasets. Among them, mode (a) performs better than mode (b). When using mode (a), the accuracy slightly decreases compared to the case where both the input and output ends use parameter nonshared in the previous section, while mode (b) significantly affects the experimental results. This verifies the correctness of our structure, which is that it is necessary to add parameter nonshared linear layers at both the input and output ends of the quantum circuit, and it will better optimize the quantum circuit.

This confirms the correctness of our structure, i.e., adding parameter nonsharing linear layers at both the input and output ends of the quantum circuit is necessary and will better optimize the quantum circuit. Therefore, we believe that it is possible to select the appropriate model structure based on different task requirements. That is, if accuracy is pursued, the BUQRNN with parameter nonsharing linear layers before and after can be used. If both accuracy and efficiency are pursued, it is also feasible to only use the parameter nonsharing structure after the BUQNN.

V. CONCLUSION

To address the brute force approach taken by traditional QRNNs when handling feature dimensions larger than the number of qubits, we propose a solution that breaks feature vectors into batches and passes them through the circuit, thereby increasing the available information. As such, in this article, we design a novel incremental QNN, termed BUQNN, and apply it to LSTM and GRU networks, forming BUQRNN. Experimental results on the Bengali corpus demonstrate that, compared to traditional QRNNs, our proposed BUQRNN improves accuracy by up to 0.993% while reducing model complexity on average by 12%. Given that traditional QRNNs are unable to independently optimize the linear layers before and after the quantum circuit, thereby limiting their adaptability to VQC, we are inspired to combine the aforementioned BUQNN design with an RNN with nonshared parameters, resulting in a model class called PN-BUQRNN. QNNs constructed using this approach perform better in experiments, surpassing both classical neural networks and traditional QRNNs on two Bengali text datasets. As an attempt in the field of low-resource language QNNs, we demonstrate the feasibility of applying quantum algorithms to address practical issues in the low-resource text domain. Considering the limited computational resources in low-resource regions, our method allows for circuit simulation with an S number of qubits, aligning with the characteristics of the low-resource language domain. Finally, our goal is to introduce our proposed model to NLP tasks in more low-resource regions in order to address a broader range of real-world issues.

APPENDIX A
FEASIBILITY EXPLORATION OF THE BUQRNN AS A WORD-EMBEDDING MODEL

In the discussion of the aforementioned related work, Li et al. [27] mentioned an approach that utilizes a QNN as a word-embedding model. The authors employed QLSTM as the pretraining model and then utilized the obtained word embeddings for downstream tasks. Due to the similarity with this work, in this section, we explore the applicability of the proposed BUQRNN as an embedding layer model on Bengali text corpora. The experimental data consist of the aforementioned BOOK-Reviews and YouTubeB-S datasets, and the specific methodology is outlined as follows.

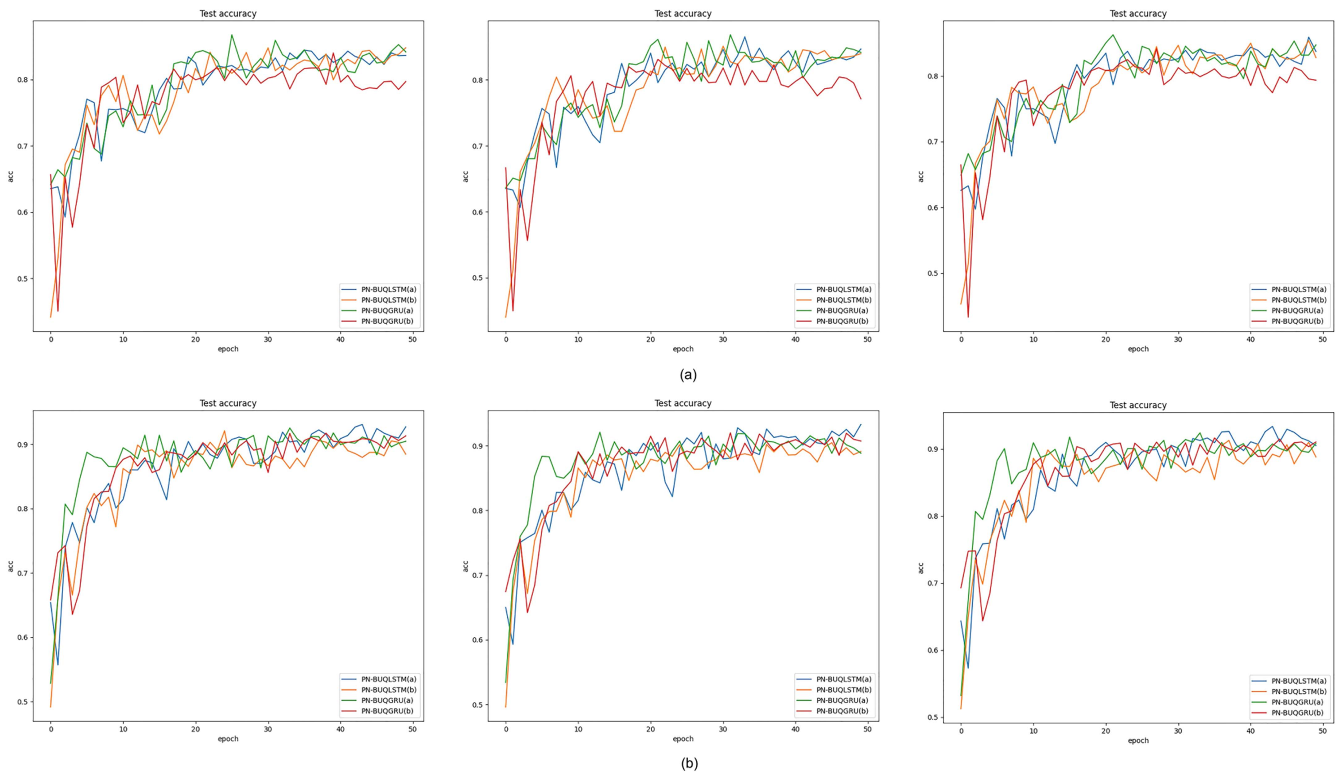


FIGURE 10. Experimental results of the two modes are presented on the datasets BOOK-Reviews and YouTubeB-S. (a) Comparison results of the above two methods on the BOOK-Reviews. (b) Comparison results of the above two methods on the YouTubeB-S.

1) *Pretraining quantum models:* We selected the proposed BUQRNN and QRNN mentioned in Section IV-B as comparative models. To ensure a fair comparison, we kept the experimental parameters consistent with those in Section IV-B, with word-embedding sizes of 8 for both the BUQRNN and the QRNN and vocabulary size as the output size.
2) *Pretraining:* The aforementioned models were separately trained on the BOOK-Reviews and YouTubeB-S datasets until convergence. During the language model training, sentiment labels were disregarded.
3) *Model evaluation:* We assessed the utility of the pretrained embedding vectors in downstream SA tasks. Specifically, we used the pretrained word embeddings from the BUQRNN and the QRNN to train a linear-layer-based classifier. The choice of a linear layer as the classifier aimed for experimental convenience. Consistent with the discussed methods in related work, we employed a random subset of language model training data for evaluation to avoid introducing additional noise into the model.

We conducted ten experiments to obtain the mean performance. Table 6 presents our experimental results. The word embeddings trained by the BUQRNN demonstrate higher accuracy compared to the QRNN, indicating the high utility of our proposed BUQRNN as a word-embedding model in the low-resource language domain. The primary reason for BUQRNN's superior results lies in its ability to fully load input vectors into the circuit, thus avoiding the loss of semantic information. This experimental outcome also reinforces our viewpoint discussed earlier. In future explorations, we anticipate using the BUQRNN as a word-embedding model for various downstream tasks in the low-resource language domain.

TABLE 6. Results of Word Vector Representations Obtained Using BUQRNN and QRNN on the BOOK-Reviews and YoutubeB-S Datasets

Model	BOOK-Reviews	YouTubeB-S
QLSTM	76.314	82.246
QGRU	77.241	82.436
BUQLSTM	**77.243**	**83.357**
BUQGRU	**78.923**	**83.143**

APPENDIX B
DIFFERENCE BETWEEN BUQNN AND VQC USING AMPLITUDE ENCODING

In the VQC, the encoding layer can utilize various encoding methods, such as the most common angle encoding or amplitude encoding. In the BUQNN, we adopt angle encoding to encode feature vectors into the quantum circuit. However, can amplitude encoding be used? The nature of BUQNN (essentially a VQC using batch uploading) makes it difficult to use amplitude encoding for feature encoding; otherwise, the BUQNN would degrade into a traditional VQC. To compare

TABLE 7. Comparison of Our BUQLSTM Network With Amplitude-Encoded QLSTM Network on the BOOK-Reviews and YouTubeB-S Datasets

Model	BOOK-Reviews	YouTubeB-S
Amplitude-QLSTM	82.579	91.557
BUQLSTM	83.121	92.037

it with amplitude encoding, we used the BUQLSTM in Section IV-B and created a QLSTM using amplitude encoding. For 8-D input data, amplitude encoding requires four qubits to encode the features into the circuit. The model layers and experimental hyperparameters were kept consistent with Section IV-B.

Table 7 demonstrates the advantages of the proposed BUQLSTM network. To some extent, amplitude encoding theoretically offers higher information capacity. However, in practical applications, the state preparation required for amplitude encoding is expensive in terms of operations. In contrast, the encoding method of BUQLSTM is more direct and requires only a small number of qubits to embed feature vectors into the circuit.

REFERENCES

[1] J. J. Grefenstette, "Genetic algorithms and machine learning," in *Proc. 6th Annu. Conf. Comput. Learn. Theory*, 1993, pp. 3–4, doi: 10.1145/168304.168305.
[2] T. M. Mitchell, *Machine Learning*. New York, NY, USA: McGraw-Hill, 1997, pp. 1–19, doi: 10.5555/541177.
[3] Z. Zhou, *Machine Learning*. New York, NY, USA: Springer, 2021, doi: 10.1007/978-981-15-1967-3.
[4] P. M. Nadkarni, L. Ohno-Machado, and W. W. Chapman, "Natural language processing: An introduction," *J. Amer. Med. Inform. Assoc.*, vol. 18, no. 5, pp. 544–551, Sep. 2011, doi: 10.1136/amiajnl-2011-000464.
[5] D. W. Otter, J. R. Medina, and J. K. Kalita, "A survey of the usages of deep learning for natural language processing," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 2, pp. 604–624, Feb. 2021, doi: 10.1109/TNNLS.2020.2979670.
[6] W. Medhat, A. Hassan, and H. Korashy, "Sentiment analysis algorithms and applications: A survey," *Ain Shams Eng. J.*, vol. 5, no. 4, pp. 1093–1113, 2014, doi: 10.1016/j.asej.2014.04.011.
[7] M. Taboada, J. Brooke, M. Tofiloski, K. Voll, and M. Stede, "Lexicon-based methods for sentiment analysis," *Comput. Linguistics*, vol. 37, no. 2, pp. 267–307, 2011, doi: 10.1162/COLI_a_00049.
[8] A. Hasan, S. Moin, A. Karim, and S. Shamshirband, "Machine learning-based sentiment analysis for Twitter accounts," *Math. Comput. Appl.*, vol. 23, no. 1, pp. 11, Oct. 2018, doi: 10.3390/mca28050101.
[9] B. Agarwal, R. Nayak, N. Mittal, and S. Patnaik, *Deep Learning-Based Approaches for Sentiment Analysis*. New York, NY, USA: Springer, 2020.
[10] B. Zhang and W. Zhou, "Transformer-Encoder-GRU (TE-GRU) for Chinese sentiment analysis on Chinese comment text," *Neural Process. Lett.*, vol. 55, no. 2, pp. 1857–1867, 2022, doi: 10.1007/s11063-022-10966-8.
[11] X. Luo, "Efficient English text classification using selected machine learning techniques," *Alexandria Eng. J.*, vol. 60, no. 3, pp. 3401–3409, Feb. 2021, doi: 10.1016/j.aej.2021.02.009.
[12] B. Min et al., "Recent advances in natural language processing via large pre-trained language models: A survey," *ACM Comput. Surv.*, vol. 56, no. 30, pp. 1–40, Sep. 2023, doi: 10.1145/3605943.
[13] H. Wang, J. Li, H. Wu, E. Hovy, and Y. Sun, "Pre-trained language models and their applications," *Engineering*, vol. 25, pp. 51–65, Apr. 2022, doi: 10.1016/j.eng.2022.04.024.
[14] L. Alchieri, D. Badalotti, P. Bonardi, and S. Bianco, "An introduction to quantum machine learning: From quantum logic to quantum deep learning," *Quantum Mach. Intell.*, vol. 3, pp. 1–30, Oct. 2021, doi: 10.1007/s42484-021-00056-8.
[15] N. Wiebe, A. Kapoor, and K. M. Svore, "Quantum deep learning," *Quantum Inf. Comput.*, vol. 16, pp. 541–587, Dec. 2014, doi: 10.26421/QIC16.7-8-1.
[16] C. H. Bennett, E. Bernstein, G. Brassard, and U. Vazirani, "Strengths and weaknesses of quantum computing," *SIAM J. Comput.*, vol. 26, no. 5, pp. 1510–1523, Jan. 1997, doi: 10.1137/S0097539796300933.
[17] W. Lai, J. Shi, and Y. Chang, "Quantum-inspired fully complex-valued neutral network for sentiment analysis," *Axioms*, vol. 12, no. 3, Feb. 2023, Art. no. 308, doi: 10.3390/axioms12030308.
[18] S. Y. Chen, S. Yoo, and Y. L. Fang, "Quantum long short-term memory," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2022, pp. 8622–8626, doi: 10.1109/ICASSP43922.2022.9747369.
[19] J. Preskill, "Quantum computing in the NISQ era and beyond," *Quantum*, vol. 2, Aug. 2018, Art. no. 79, doi: 10.22331/q-2018-08-06-79.
[20] Y. Cao, X. Zhou, X. Fei, H. Zhao, W. Liu, and J. Zhao, "Linear-layer-enhanced quantum long short-term memory for carbon price forecasting," *Quantum Mach. Intell.*, vol. 5, no. 2, pp. 1–12, Jul. 2023, doi: 10.1007/s42484-023-00115-2.
[21] S. Sazzed, "Cross-lingual sentiment classification in low-resource Bengali language," in *Proc. 6th Workshop Noisy User-Generated Text*, 2020, pp. 50–60, doi: 10.18653/v1/2020.wnut-1.8.
[22] C. -H. H. Yang, J. Qi, S. Y. -C. Chen, Y. Tsao, and P. -Y. Chen, "When BERT meets quantum temporal convolution learning for text classification in heterogeneous computing," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2022, pp. 8602–8606, doi: 10.1109/ICASSP43922.2022.9746412.
[23] M. Al-Amin, M. S. Islam, and S. D. Uzzal, "Sentiment analysis of Bengali comments with Word2Vec and sentiment information of words," in *Proc. Int. Conf. Elect., Comput. Commun. Eng.*, 2017, pp. 186–190, doi: 10.1109/ECACE.2017.7912903.
[24] P. Chowdhury, E. M. Eumi, O. Sarkar, and M. F. Ahamed, "Bangla news classification using GloVe vectorization, LSTM, and CNN," in *Proc. Int. Conf. Big Data, IoT, Mach. Learn.*, 2017, pp. 723–731, doi: 10.1007/978-981-16-6636-0_54.
[25] M. R. Hossain, M. M. Hoque, and I. H. Sarker, "Text classification using convolution neural networks with fasttext embedding," in *Hybrid Intelligent Systems*, 2021, pp. 101–113, doi: 10.1007/978-3-030-73050-5_11.
[26] T. Pires, E. Schlinger, and D. Garrette, "How multilingual is multilingual BERT?," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*, 2019, pp. 4996–5001, doi: 10.18653/v1/P19-1493.
[27] S. S. Li et al., "PQLM—Multilingual decentralized portable quantum language model," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2023, pp. 1–5, doi: 10.1109/ICASSP49357.2023.10095215.
[28] R. Di Sipio, J.-H. Huang, S. Y.-C. Chen, S. Mangini, and M. Worring, "The dawn of quantum natural language processing," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2022, pp. 8612–8616, doi: 10.1109/ICASSP43922.2022.9747675.
[29] M. Cerezo et al., "Variational quantum algorithms," *Nature Rev. Phys.*, vol. 3, no. 9, pp. 625–644, Aug. 2021, doi: 10.1038/s42254-021-00348-9.
[30] R. Horodecki, P. Horodecki, M. Horodecki, and K. Horodecki, "Quantum entanglement," *Rev. Modern Phys.*, vol. 81, no. 2, 2009, Art. no. 865, doi: 10.1103/RevModPhys.81.865.
[31] R. LaRose and B. Coyle, "Robust data encodings for quantum classifiers," *Phys. Rev. A.*, vol. 103, no. 2, Aug. 2020, doi: 10.1103/PhysRevA.102.032420.
[32] I. Glendinning, "The bloch sphere," in *Proc. QIA Meeting*, 2005, pp. 3–18.
[33] M. Periyasamy, N. Meyer, C. Ufrecht, D. D. Scherer, A. Plinge, and C. Mutschler, "Incremental data-uploading for full-quantum classificationn," in *Proc. IEEE Int. Conf. Quantum Comput. Eng.*, 2022, pp. 31–3, doi: 10.1109/QCE53715.2022.00021.
[34] A. Perez-Salinas et al., "Data re-uploading for a universal quantum classifier," *Quantum*, vol. 4, Feb. 2020, Art. no. 226, doi: 10.22331/q-2020-02-06-226.
[35] Z. Hong, J. Wang, X. Qu, C. Zhao, W. Tao, and J. Xiao, "QSpeech: Low-qubit quantum speech application toolkit," in *Proc. Int. Joint Conf. Neural Netw.*, Padua, Italy, 2022, pp. 01–08, doi: 10.1109/IJCNN55064.2022.9892496.
[36] E. Hossain et al., "Sentiment polarity detection on Bengali book reviews using multinomial naive bayes," in *Progress Adv. Comput. Intell. Eng.*, 2021, pp. 281–289, doi: 10.1007/978-981-33-4299-6_23.
[37] B. Zhang and Q. Zhuang, "Fast decay of classification error in variational quantum circuits," *Quantum Sci. Technol.*, vol. 7, no. 3, 2022, Art. no. 035017, doi: 10.1088/2058-9565/ac70f5.