Moon et al. *EPJ Quantum Technology* (2025) 12:32
https://doi.org/10.1140/epjqt/s40507-025-00333-6

EPJ Quantum Technology
a SpringerOpen Journal

RESEARCH Open Access

QSegRNN: quantum segment recurrent neural network for time series forecasting

Kyeong-Hwan Moon[1†], Seon-Geun Jeong[2†] and Won-Joo Hwang[1,2*]

*Correspondence:
wjhwang@pusan.ac.kr
[1]School of Computer Science and Engineering, Pusan National University, Busandaehak-ro 63beon-gil, Geumjeong-street, Busan, 46241 Gyeongsangnam-do, Republic of Korea
[2]Department of Information Convergence Engineering, Pusan National University, Busandaehak-ro 63beon-gil, Geumjeong-street, Busan, 46241 Gyeongsangnam-do, Republic of Korea
[†]Equal contributors

Abstract

Recently many data centers have been constructed for artificial intelligence (AI) research. The important condition of the data center is to supply sufficient electricity, resulting in many electricity transformers being installed. Especially, these electricity transformers have led to significant heat generation in many data centers. Therefore, managing the temperature of electricity transformers has emerged as an important task. Notably, numerous studies are being conducted to manage and forecast the temperature of electricity transformers using artificial intelligence models. However, as the size of predictive models increases and computational demands grow, substantial computing resources are required. Consequently, there are instances where the lack of computing resources makes these models difficult to operate. To address these challenges, we propose a quantum segment recurrent neural network (QSegRNN), a time series forecasting model utilizing quantum computing. QSegRNN leverages quantum computing to achieve comparable performance with fewer parameters than classical counterpart models under similar conditions. QSegRNN inspired by a classical SegRNN uses the quantum cell instead of the classical cell in the model. The advantage of this structure is that it can be designed with fewer parameters under similar architecture. To construct the quantum cell, we benchmark the quantum convolutional circuit with amplitude embedding as the variational quantum circuit, minimizing information loss while considering the limit of noisy intermediate-scale quantum (NISQ) devices. The experiment result illustrates that the forecasting performance of QSegRNN achieves better performance than SegRNN and other forecasting models even though QSegRNN has only 85 percent of the parameters.

Keywords: Quantum–classical neural networks; Quantum encoding; Quantum machine learning; Time series forecasting; Variational quantum circuits

1 Introduction

Recent advancements in artificial intelligence (AI) and many data-driven applications request more computational resource services(especially cloud computing-based services) and higher storage demands. These features have led to the establishment of many data centers. Data centers are notorious for consuming large amounts of energy, with electric-

ity usage continuously increasing due to the growing demand for cloud services and AI workloads.

Moreover, it has been noted that global data center electricity usage could reach alarming levels, contributing significantly to environmental strain if not managed effectively [1]. Also, recent studies have shown that data centers contribute significantly to carbon emissions, with sustainable energy usage and power optimization becoming a primary concern.

Consequently, the important factor of data centers is to supply electricity that satisfies the data center's devices to be operated. However, with the advent of the big data era, the need to store vast amounts of data has increased, leading to substantial power consumption in data centers. Data centers use transformers to supply electricity, but the extensive use of electricity causes transformer temperatures to rise, which could create a significant problem such as malfunction or blaze. Therefore managing electricity consumption and electricity transformer temperatures has come to the surface.

Challenges. Several studies have attempted to address this issue using artificial intelligence models to forecast transformer temperatures [2–7]. However, as the number of parameters in AI models increases, the resources required for training and deployment also rise, potentially leading to negative impacts on temperature and performance. Furthermore, insufficient computing resources and long training time can lead to situations where the models are difficult to train or deploy.

Contribution. To address this issue, we propose a hybrid quantum-classical time series forecasting model incorporating quantum computing, called quantum segment recurrent neural network (QSegRNN). QSegRNN employs fewer parameters than the recurrent neural network (RNN) proposed in SegRNN [2] while also demonstrating superior performance. Our model consists of encoding and decoding phases, employing the hybrid quantum-classical model, which allows inheriting more information from the previous layer with an amplitude embedding layer [8] operating only a few qubits for these phases. The experiment proceeds by comparing our model with classical time series forecasting models and comparing the quantum layer hyperparameter, experimenting with the effect of the quantum parameter on the forecasting results. The experiment results showed our model's performance improvement at mean squared and mean average errors compared with classical SegRNN, 0.006 and 0.004. The main contributions of our model are as follows:

- QSegRNN addresses the drawback of SegRNN, which requires substantial computing resources for training due to its large number of parameters, by applying quantum computing to maintain similar performance with fewer parameters.
- Our model employed amplitude embedding which encodes 2^n classical features into the amplitudes of an n-qubit state to address the information loss that occurs considerably depending on the embedding method due to the limitation of the number of qubits in noisy intermediate-scale quantum (NISQ) devices. The experiment result illustrates that our model with amplitude embedding overperformes compared with the model with angle embedding and amplitude embedding.
- Comprehensive simulation results offer an opportunity for the electricity transformer temperature model to be applied in real industry data centers since it has fewer parameters compared to classical SegRNN and takes advantage of quantum computing.

The remainder of this paper is organized as follows. In Sect. 2, we describe the related works of QSegRNN. Section 3 explains the preliminary. Section 4 introduces the structure of QSegRNN. The simulation results are presented in Sect. 5. Section 6 introduces the discussion. Finally, the conclusion is represented in Sect. 7.

2 Related works

2.1 Time series forecasting based on classical model

Time series forecasting is utilized in various fields such as temperature, environment, and machinery [2–7, 9–18]. With the rise of environmental and financial technology interest, the interest in time series forecasting has also increased.

Recently, a growing body of research has been conducted to improve time series forecasting performance. Some have developed large-scale forecasting models to enhance nonlinearity or incorporated additional training techniques to boost predictive performance [5, 9, 10, 16, 17]. M. Jin et al. [16] developed Time-LLM, a time-series forecasting model, based on a pre-trained large language model, enhancing predictive performance. Time-LLM uses patch reprogramming to align time-series data with natural language. A. Das et al. [10] proposed a decoder-only forecasting model to enhance time series prediction performance. J. Liu et al. [9] proposed a time series forecasting model training strategy that utilizes negative sample-based contrastive learning to learn high-dimensional latent representations, enhancing forecasting accuracy and generalization performance on time series data. Meanwhile, research is being conducted on diversifying the use of data information through preprocessing and model architecture transformation. H. Wu et al. [17] proposed TimesNet, which enhanced time series forecasting performance by transforming 1D data into 2D. H. Wang et al. [5] enhanced forecasting performance using a time series decomposition method that leverages local features and global correlations. Meanwhile, Sarkar et al. proposed a real-time multi-agent reinforcement learning framework to reduce carbon emissions by optimizing cooling systems and energy consumption in dynamic environments, achieving a 14 percent reduction in energy usage [19].

However, data transformation requires continuous and additional work, and large models can be difficult to use on certain devices due to memory constraints. Therefore, in this paper, we address the issue by leveraging the quantum machine learning technique to reduce the number of model parameters achieving the quantum advantage such as computational efficiency.

2.2 Time series forecasting based on quantum model

Even though the classical time series forecasting models show remarkable performance, these models still face a problem which are constructed with many parameters. Therefore, various quantum time series forecasting models were proposed to address the issue of classical time series forecasting models.

S.Y.C. Chen et al. [15] proposed Quantum Long Short Term Memory (QLSTM), a hybrid quantum-classical machine learning model, to adopt quantum computing into the classical LSTM [14]. Y. Cao et al. [13] proposed Linear-layer-enhanced Quantum Long Short Term Memory (L-QLSTM) to address barren plateaus encountered while training QLSTM. L-QLSTM enhances feature extraction ability with shared linear layers. P. Singh [20] proposed Fuzzy-Quantum Time Series Forecasting Model (FQTSFM), a hybrid model that

integrates fuzzy time series approaches with a Fast Forward Quantum Optimization Algorithm (FFQOA) to enhance prediction accuracy and optimize fuzzy membership functions and the universe of discourse selection. However, these models still sometimes suffer from barren plateaus due to the architecture of each model. Therefore, H.H.S. Chittoor et al. [21] proposed Quantum Long-Term Time Series Forecasting (QuLTSF), a hybrid quantum-classical machine learning model, combining amplitude embedding and variational quantum circuits (VQCs) with classical linear layers to improve multivariate long-term time series forecasting. S.G. Jeong et al. [12] proposed Hybrid Quantum-Classical Gated Recurrent Unit (HQGRU) to address the limitations of traditional deep learning models such as LSTM [14] and GRU [11], which require substantial computational resources due to their large number of parameters. Additionally, it overcomes the issue of barren plateaus encountered in QLSTM [15], a model that adapts LSTM for quantum computing, by employing a Variational Quantum Circuit (VQC) based algorithm. Here, the VQC consists of the angle embedding layer [22], a quantum convolution layer [23], and a strongly entangling layer [24]. HQGRU addresses the shortcomings of traditional time-series models by using hybrid gates based on VQC for each gate in GRU such as reset gate and update gate. HQGRU outperforms LSTM, L-QLSTM [13], and GRU, achieving similar performance with fewer parameters. In this paper, we employee HQGRU to our model's RNN part.

However, using angle embedding in VQC can lead to significant information loss during the training process of the QSegRNN model when projecting from higher to lower dimensions, especially when the previous layer's output has a large vector of features. In this paper, we minimize information loss by employing a VQC that combines the convolution layer of the Quantum Convolutional Neural Network (QCNN) [23] with amplitude embedding [8] and a strongly entangling layer [24].

3 Preliminaries

3.1 Segment recurrent neural network

Segment recurrent network (SegRNN) [2] is a time series forecasting model that addresses the gradient problem, which arises from performing many recurrent iterations in traditional RNN models such as long stort term memory (LSTM) [14] and gated recurrent unit (GRU) [11]. It mitigates this issue by reducing the recurrent iteration count through encoding and decoding techniques. SegRNN reduces the recurrent iteration count by replacing point-wise iterations with segment-wise iterations and proposing parallel multi-step forecasting that includes parallel encoding and decoding.

The training process of SegRNN involves splitting the input sequence data and performing embedding and encoding using RNN. Subsequently, the encoded data is decoded and predicted with parallel multi-step forecasting (PMF). Finally, the predicted sequences are compared with the original data to update the model.

In the encoding phase, to replace the original time point-wise iterations with sequence segment-wise iterations, the model segments and embeds input sequence data and forwards it into the RNN cell. Therefore, SegRNN reduces the number of training iterations. In the decoding phase, SegRNN utilizes the PMF strategy to reduce the number of iterations. In the PMF strategy, the positional embedding data are concatenated with channel embedding data. With this method, the model can enhance the ability to capture relationships between variables. Finally, the concatenated data is repeated and forwarded into the RNN cell to reconstruct output data.

However, with 1.6 million parameters and 3.8M flops, the SegRNN model may be difficult to train or operate if hardware support is insufficient [2]. In this paper, we replace the RNN in the SegRNN model with a modified hybrid quantum gated recurrent unit (HQGRU) [12] to reduce the number of parameters used for training, guaranteeing performance similar to the original SegRNN.

3.2 Angle embedding and amplitude embedding

The embeddings in quantum computing represent classical data as quantum states in a Hilbert space. It transforms N feature classical data x into a set of parameters in a quantum circuit via a quantum features map. In this paper, we remark angle embedding [22] and amplitude embedding [8] applied in the quantum GRU [12] cell.

Angle embedding is an embedding layer that encodes N features into the rotation of n qubits, where $N \leq n$ [22]. In traditional physics, a tensor can be represented as in Eq. (2). Where, $\Phi^{s_j}(x_j)$ denotes local feature map and x_j denotes the component of N-dimension vector x. The function $\Phi^{s_j}(x_j)$ can be defined as in Eq. (3).

$$x = [x_1, x_2, \ldots, x_N] \in \mathbb{R}^N \tag{1}$$

$$\Phi(x) = \Phi^{s_1\ s_2\ \cdots\ s_N}(x) = \phi^{s_1}(x_1) \otimes \phi^{s_2}(x_2) \otimes \cdots \otimes \phi^{s_N}(x_N) \tag{2}$$

$$\Phi^{s_j}(x_j) = [cos(\frac{\pi}{2}x_j), sin(\frac{\pi}{2}x_j)] \tag{3}$$

Therefore, angle embedding can represent the input data x in a quantum feature map. The rotation of angle embedding can be applied with the Pauli-X, Pauli-Y, and Pauli-Z gate, where each gate contributes to rotation through π radians around the x-axis, y-axis, and z-axis in the Bloch Sphere [8].

Amplitude embedding is another embedding method that encodes 2^n features into the amplitudes of an n-qubit state [8]. It allows for encoding a larger number of features into fewer qubits compared to angle embedding. With this embedding algorithm, normalized classical data $\hat{x} \in \mathbb{R}^N$ can be represented to the n-qubit quantum state $|\psi_x\rangle$:

$$|\psi_x\rangle = \frac{1}{||\hat{x}||} \sum_{i=1}^{N} \hat{x}_i |i\rangle \tag{4}$$

Where $|i\rangle$ is the ith computational basis state. The number of qubits needed to apply amplitude embedding can be calculated with the following equation:

$$n = \lceil log_2(N) \rceil \tag{5}$$

In this paper, we employed amplitude embedding to fully reflect the information of previous classical layers.

3.3 Variational quantum circuits

Variational quantum circuit (VQC) is a type of quantum circuit that contains free trainable parameters. These parameters work similarly to the weights in artificial neural networks. VQC can be trained using a classical optimization algorithm interacting with quantum devices through queries [8]. This hybrid quantum-classical approach is central to many

quantum machine learning (QML) models, as it enables efficient training while utilizing the unique advantages of quantum computation.

For example, the VQC was studied to avoid barren plateaus, where the loss suffers from decreasing because the loss landscape is flat [25]. VQC was also used to construct a hybrid quantum-based GRU model, replacing gates in the original GRU model [12].

The VQC allows AI models to be constructed with decreased model depth and robustness to noise. Therefore, VQC is expected to be suitable for QML models operated in Noisy Intermediate-Scale Quantum (NISQ) devices. However, training or operating QML models with many qubits via classical computers is hard. Because n qubits can take 2^n of states, resulting in mal-operation in limited resources. In this paper, we only employ reasonable size of QML models to satisfy the model to be operated in NISQ devices.

3.4 Quantum convolutional neural networks

A quantum convolutional neural network (QCNN) is a collection of quantum circuits that is inspired by a convolutional neural network (CNN). Therefore, QCNN performs the same as a CNN in conventional machine learning [23]. Classical CNN consists of a convolution layer and a pooling layer [26]. The convolution layer extracts features of an input image with filters. After extracting features from an image, the pooling layer chooses the significant feature value based on pooling algorithms. Since CNN extracts features in an image, CNN can achieve translation invariance which takes the same output value even if the input image's target has a different position in the image. The QCNN adopts the architectures of CNN, which can also achieve translation invariance. Also, QCNN can overcome the barren plateau, one of the problems that could occur in VQC [27]. The quantum state of an i-th layer of QCNN can be expressed as follows [23]:

$$|\psi_i(\theta_i)\rangle\langle\psi_i(\theta_i)| = Tr_{B_i}(U_i(\theta_i)|\psi_{i-1}\rangle\langle\psi_{i-1}|U_i(\theta_i)^\dagger) \tag{6}$$

Here, Tr_{B_i} is the trace of subsystem $B_i \in \mathbb{C}^{2^{n/2^i}}$, U_i is a unitary gate that could be trained via input and output of the networks. The unitary gate includes the convolution and pooling layer of QCNN. The state $|\psi_0\rangle = |0\rangle^{\otimes n}$ is also included in U_i. In our research, the QCNN is employed with the amplitude embedding [8] and strongly entangling layer [24] in the quantum GRU [12] model to overcome the forecasting performance of classical RNN models with lower parameters.

4 Methodology

In this paper, we introduce quantum segment recurrent neural network (QSegRNN), a hybrid model that applies quantum machine learning (QML) to achieve performance similar to classical deep learning models with fewer parameters. QSegRNN consists of an encoding phase and a decoding phase. In the encoding phase, the first step is segmenting the data for encoding. After segmentation, recursive encoding is performed, which transforms the data to use in the decoding phase. In the decoding phase, the recursive decoding and prediction of the encoded data is conducted. Additionally, instance normalization is applied to mitigate the problem of distribution shift.

Our model replaces the RNN cell employed in the encoding and decoding phases of SegRNN [2] with the HQGRU-based [12] QML model. Throughout this study, the QML model will be referred to as HQRNN. The structure of QSegRNN is illustrated in Fig. 1.

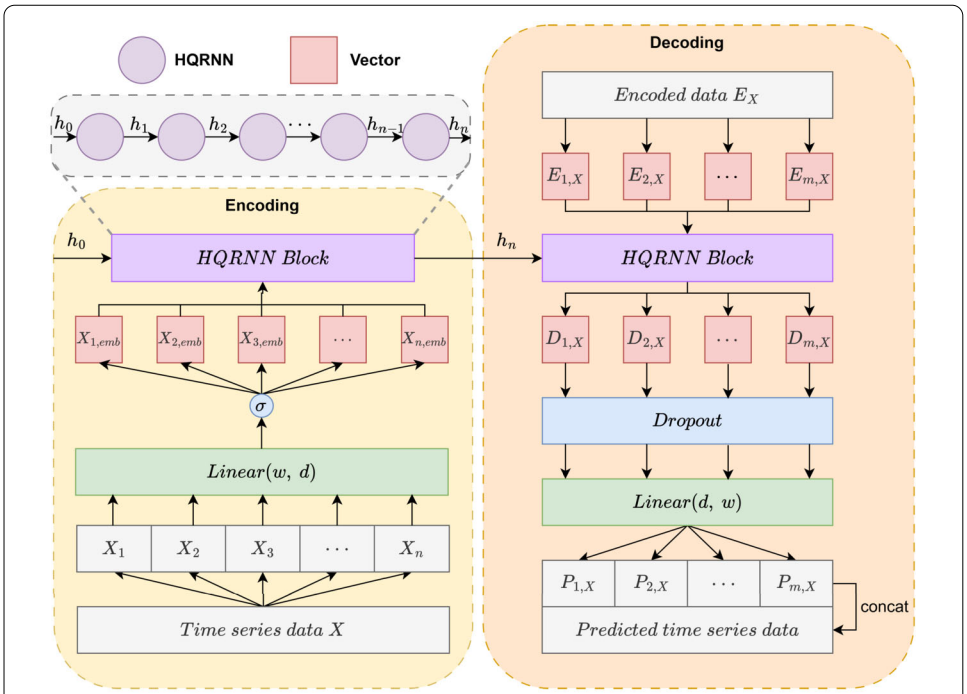


Figure 1 Overall architecture of QSegRNN

The advantage of this study is to reduce the size of the classical machine learning model while maintaining performance. Our model achieved improvement in forecasting performance even though the model only contains 0.3M fewer parameters. Additionally, applying amplitude embedding to the VQC of the HQGRU in both phases minimizes information loss compared to the angle embedding. This feature enables QSegRNN to perform similarly to SegRNN despite having fewer parameters.

4.1 Encoding phase

In the encoding phase, the input data is restructured and encoded to be usable in the decoding phase. The output of the encoding phase contains encoded information of input data sequence by sequence. First, the input sequence data X is partitioned into n segments. The n partitioned segmented data $X_{partitioned}$ is defined as shown in Eq. (7), while n is decided by dividing the prediction sequence length by the input sequence length.

$$X_{partitioned} = \{X_1, X_2, \ldots, X_{n-1}, X_n\} \tag{7}$$

Subsequently, $X_{partitioned}$ passes via a value embedding layer that is constructed with linear layer L_1 and ReLU activation function to be restructured into data the same size as the HQRNN's hidden state. This layer not only allows the size of information to be forwarded in the HQRNN but also can expand the information dimension depending on the hidden state hyperparameter.

$$X_{emb} = ReLU(L_1(X_{partitioned})) \tag{8}$$

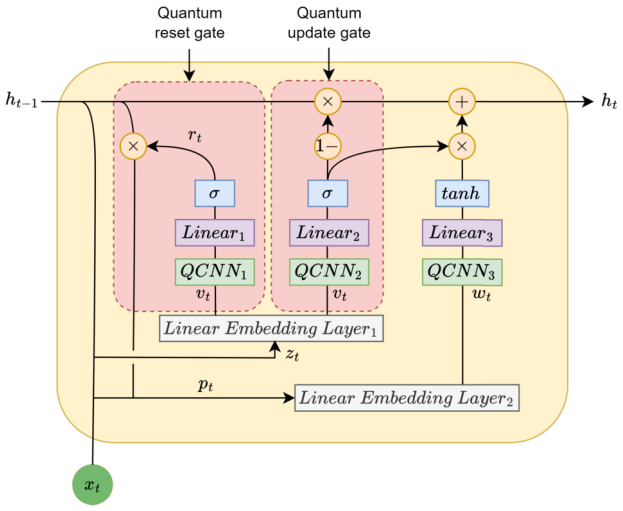


Figure 2 Architecture of HQRNN in QSegRNN

Here, X_{emb} is the set of embedded data. It can be represented as shown below.

$$X_{emb} = \{X_{1,emb}, X_{2,emb}, \cdot, X_{n,emb}\} \tag{9}$$

The embedded data X_{emb} passed through the value embedding layer becomes the encoded data E_X after passing through the HQRNN.

$$E_X = HQRNN(X_{emb}) \tag{10}$$

At this stage, the HQRNN employs HQGRU [12], which its structure is shown in Fig. 2. As shown in Fig. 3, the QCNN layer uses the convolution circuit proposed by I. MacCormack et al. [23]. With this circuit, a quantum convolution layer in HQRNN can be constructed. In the structure of HQRNN's QCNN layer, amplitude embedding [8] is employed to minimize information loss at each gate during the QRNN forwarding process, and a strongly entangling layer [24] is added at the end to complete the HQRNN's VQC layer. This hybrid QML structure allows the HQRNN to embed information from the previous layer with a dimension of 2^n and get the output of the encoding phase with fewer parameters than classical machine learning models. The HQRNN's formulation is as follows:

$$z_t = [h_{t-1}, x_t] \tag{11}$$

$$v_t = LE_1(z_t) \tag{12}$$

$$r_t = \sigma(L_3(QCNN_1(v_t))) \tag{13}$$

$$u_t = \sigma(L_4(QCNN_2(v_t))) \tag{14}$$

$$w_t = LE_2(r_t \otimes h_{t-1}) \tag{15}$$

$$h_t = tanh(L_3(QCNN_3(w_t))) \otimes u_t + (1 - u_t) \otimes h_{t-1} \tag{16}$$

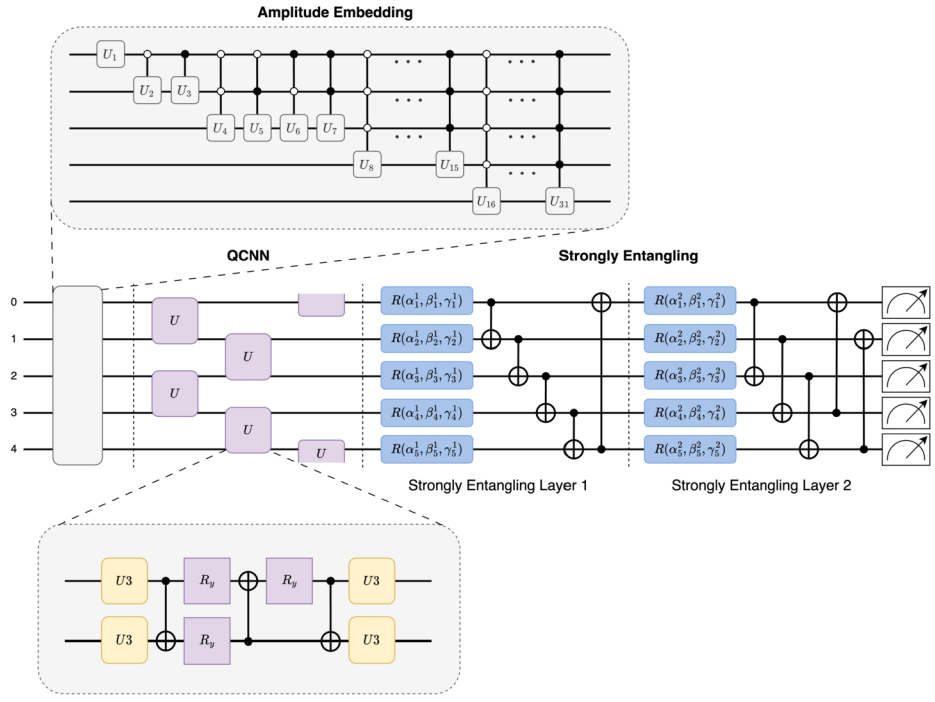


Figure 3 Quantum circuit used in QCNN layer in HQRNN

where LE_1 and LE_2 denote linear embedding layer, L_3 and L_4 denote linear layer and σ denotes sigmoid activation function.

The HQRNN consists of a reset gate and an update gate. The input v_t for both gates concatenates the previous state h_{t-1} and the input data x_t as defined to z_t. The input state z_t is forwarded to a linear embedding layer LE_1 to get the input of the quantum reset gate and update gate in Fig. 2. Each gate has a QCNN layer, a linear layer, and a sigmoid activation function. Like the structure of GRU, the output of the quantum reset gate r_t is multiplied by the previous state h_{t-1} to produce p_t, which is then forwarded through the linear embedding layer LE_2. The output of the quantum update gate is multiplied by the previous state h_{t-1} after subtracting 1. The output of linear layer 2, w_t, is forwarded into a QCNN layer, a linear layer, and a hyperbolic tangent activation function. Finally, the output of the branch starts from LE_2. It passes through a QCNN layer, a linear layer, and a hyperbolic tangent activation function, multiplied by the update gate's auxiliary value to get the current state h_t.

Since quantum computing is applied to our model, our model can take advantage of the entanglement and superposition. The QCNN layers in HQRNN of QSegRNN allow the layers to capture global features due to entanglement [23]. The superposition of quantum allows the model to be constructed with a few parameters. Therefore, our model can achieve better performance than SegRNN with fewer parameters.

4.2 Decoding phase

The decoding phase comprises the decoding of encoded data and the prediction stage. First, the position embedding data p_e is concatenated with the channel embedding data c_e

to enhance the ability to capture relationships between variables.

$$P = [p_e, c_e] \tag{17}$$

Next, The embedded data E_X is repeated m times to form the unified data X_{union}.

$$X_{union} = \bigcup_{i=1}^{m} E_{X_i} \tag{18}$$

Subsequently, P and X_{union} are forwarded through the HQRNN to obtain the set of output vectors D_X. The employed HQRNN is the same HQRNN used in the encoding phase.

$$D_X = HQRNN(P, X_{union}) \tag{19}$$

Finally, D_X is passed through a dropout mask r and a linear layer L_2 to obtain the set of predicted sequences P_X.

$$P_X = L_2(r \otimes D_X) \tag{20}$$

The predicted data P_X is a set of partitioned sequence data, which undergoes a concatenation process to form a single sequence. The set of predicted data P_X is compared with the original data T over the prediction length using a loss function, and the model is updated accordingly. During the model's parameter update process, since the HQRNN is a QML model, the parameter-shift rule is employed to update the model [28, 29]. The pseudo-code for QSegRNN is shown in Algorithm 1.

4.3 Training

The QSegRNN training process is shown in Fig. 4. In the encoding phase, our model segments input sequence data X with a fixed segment length. After segmenting the input data, the segmented data $X_{partitioned}$ are passed into a linear embedding layer L_1 to embed the data and train the linear embedding layer with segmented vector information. The embedded data X_{emb} are passed into an HQRNN block to encode and let the network learn time series features. Here, X_{emb} are forwarded into a quantum embedding layer and HQRNN.

In the decoding phase, we concatenate the encoded data with positional embeddings P to preserve temporal order while eliminating recurrent dependencies, and the concatenated data E_{X_i} are forwarded into a quantum embedding layer and HQRNN. In the Parallel Multi-step Forecasting (PMF) step, same as SegRNN [2], instead of sequentially predicting each future step, our model predicts all future time steps simultaneously in a single forward pass. This is achieved by leveraging a Transformer-style decoder [30], which takes the segment representations and learns to map them to multiple future values at once. Finally, the model is trained by optimizing multi-step loss functions with predicted data P_X and ground truth data T to ensure accuracy across all predicted time steps.

5 Experiment

Electricity Transformer Temperature (ETT) forecasting is a long-term time series forecasting (LTSF) task. LTFS forecasts longer horizons, while some tasks focus on short-

Algorithm 1 Training process of QSegRNN

Input: Sequence data X.
Output: Predicted sequence data P_X.
for *epoch* $= 1, 2, \ldots, 5$ **do**
  for *iteration* $= 1, 2, \ldots, 7800$ **do**
    Partition input sequence data X:

$$X_{partitioned} = \{X_1, X_2, \ldots, X_{n-1}, X_n\}$$

    Embed partitioned data $X_{partitioned}$:

$$X_{emb} = ReLU(L_1(X_{partitioned}))$$

    Encode embedded data X_{emb}:

$$E_X = HQRNN(X_{emb})$$

    Construct repeated data X_{union}:

$$X_{union} = \bigcup_{i=1}^{m} E_{X_i}$$

    Decode unioned data X_{union} and information embedding data P:

$$D_X = HQRNN(P, X_{union})$$

    Predict output sequence P_X:

$$P_X = L_2(r \otimes D_X)$$

  end for
  Update QSegRNN's parameters θ.
  Update learning rate.
end for

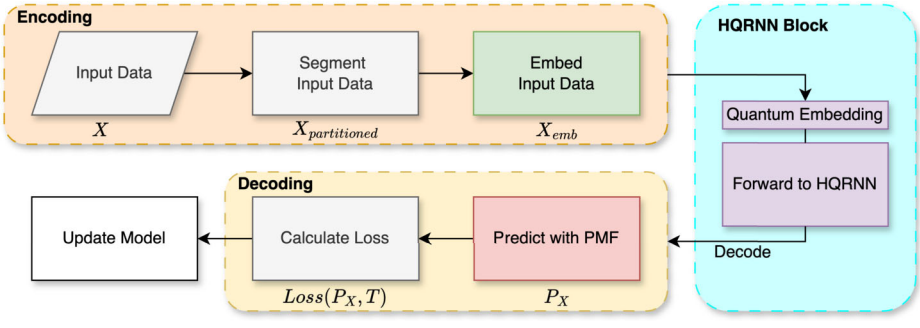


Figure 4 Training progress of QSegRNN

Table 1 Dataframe with 7 features of ETT1 dataset

Date	HUFL	HULL	MUFL	MULL	LUFL	LULL	OT
2016-07-01 00:00:00	5.827	2.009	1.599	0.462	4.203	1.340	30.531000
2016-07-01 01:00:00	5.693	2.076	1.492	0.426	4.142	1.371	27.787001
2016-07-01 02:00:00	5.157	1.741	1.279	0.355	3.777	1.218	27.787001
2016-07-01 03:00:00	5.090	1.942	1.279	0.391	3.807	1.279	25.044001
2016-07-01 04:00:00	5.358	1.942	1.492	0.462	3.868	1.279	27.948000

horizon forecasting. In the experiment, we focus on predicting the long horizon of transformer temperature with classical and quantum models. First, we explore comparing the results of classical SegRNN and QSegRNN to clarify whether our model's performance could overcome the performance of classical SegRNN with fewer parameters. Next, we focus on the experiment on how the quantum hyperparameters affect the QSegRNN. We analyze our experiments as follows:

- analysis the effect of diverse VQC structures
- analysis of QSegRNN performance compared to classical models
- analysis of the noise effects

5.1 Experiment setup

1) Data description: The electricity transformer temperature (ETT) dataset [7] includes recorded data of the temperature information of transformers in a time-series format. In this study, we specifically uses the ETT1 dataset for the experiments since it is widely used in time-series forecasting benchmarks [2–7]. The dataset has 7 columns that are recorded every hour. These columns are high use frequency load (HUFL), high use low load (HULL), medium use frequency load (MUFL), medium use low load (MULL), low use frequency load (LUFL), low use low load (LULL), and oil temperature (OT). The structure of ETT1 is shown in Table 1. We split the dataset with a ratio of 0.6, 0.2, and 0.2 for training, validation, and testing.

2) Training model settings: In our experiment, the training consists of 5 epochs, with each epoch comprising 7800 iterations. This training period was determined by experimentally verifying that the model converges. The batch size is set to 1 due to the limitation of random access memory (RAM). And the learning rate was set to 0.001 to facilitate stable convergence while preventing both divergence at higher values and excessively slow optimization at lower values. The hidden layer size was set to 512, as this configuration provided sufficient capacity to capture temporal dependencies while avoiding the excessive computational costs and parameters associated with larger hidden layer sizes. A window size of 720 was chosen to incorporate long-term dependencies in transformer temperature data. The window size enables the model to train using past data from the previous step. A shorter window size resulted in the loss of critical temporal information, leading to a decline in performance. However, 48 segment length, 96 prediction length, and rate of 0.5 were choosen to match the hyperparameters and compare with the classical model [2]. The HQRNN's numbers of qubits are set to 7 to match the shape of the data. Each experiment is repeated 5 times, and the final results are the average of these repetitions. The evaluation metrics used are mean squared error (MSE) and mean absolute error (MAE). These metrics were selected as they are widely adopted in previous studies [2–7].

All experiments were performed on a device with an AMD Ryzen 5 7600G 6-core Processor CPU and 31 GB of RAM. The experiments were implemented using PyTorch 2.3.0

with PennyLane [8] in a virtual environment with Python version 3.9. The summary of our experiments is as follows:

- We experiment with comparing classical time series forecasting models with QSegRNN comprised of various embedding and VQC.
- We compare the QSegRNN's performance with the diverse embedding layer, VQC, and the entangling layer in the model's HQRNN.
- We compare the QSegRNN's performance with diverse noise rates.
- Finally, we analyze the performance of classical SegRNN and QSegRNN regarding the ETT1 dataset.

5.2 Experiment result

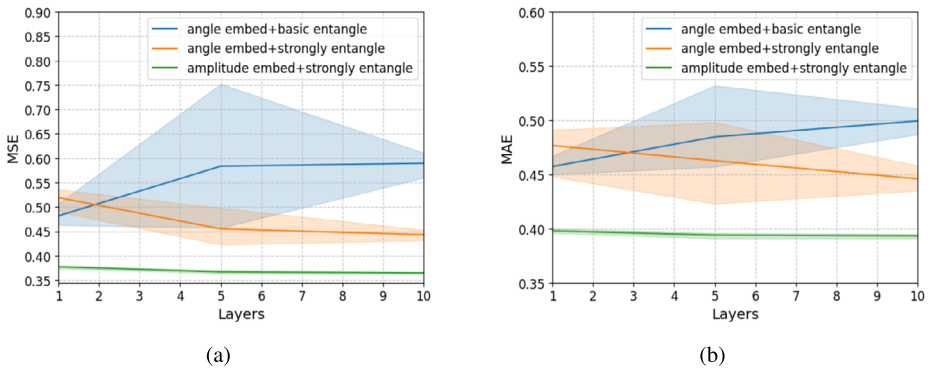
In this section, we compare QSegRNN with diverse method of embedding, entangling, and VQCs. Here, the QSegRNN with angle embedding and basic entangling layer employs angle embedding and a basic entangling layer, where the basic entangling layer is composed of RX and RY gates for each wire. The model with angle embedding and strongly entangling layer employs angle embedding, QCNN, and a strongly entangling layer. The amplitude embedding and strongly entangling layer employs amplitude embedding, QCNN, a strongly entangling layer. The number of quantum parameters increases linearly as the number of layers increases. Furthermore, we compare QSegRNN with SegRNN and diverse time series forecasting models.

The experimental results in Table 2 provide insights into the impact of quantum layers on model performance. In QSegRNN with angle embedding and basic entangling layer, increasing the number of quantum layers does not necessarily enhance forecasting accuracy, as both MSE and MAE degrade despite the rise in quantum parameters from 63 to 630. This suggests that additional quantum layers, when paired with a basic entangling structure, may fail to capture meaningful correlations, thereby impeding effective training. In contrast, QSegRNN with angle embedding and strongly entangling layer shows moderate improvements, with MAE decreasing from 0.477 to 0.446. While the strongly entangling layer facilitates better feature representation, the model still lacks robust convergence properties and remains inferior to classical counterparts, indicating that entanglement alone is insufficient for significant performance gains.

On the other hand, QSegRNN with amplitude embedding and strongly entangling layer demonstrates a consistent reduction in loss as circuit depth increases, with MAE decreasing from 0.396 to 0.391. Unlike angle embedding, amplitude embedding effectively utilizes deeper quantum architectures by encoding richer representations in Hilbert space. Additionally, the expansion of the linear embedding layer output dimension to match the

Table 2 Experiment result of QSegRNN with a diverse number of quantum circuit layers

RNN	Quantum embedding layer	Quantum entangling layer	Layers	Quantum parameters	Classical parameters	MSE	MAE
HQGRU	Angle Embedding	Basic Entangling	1	63	0.06M	0.482	0.457
			5	315	0.06M	0.584	0.485
			10	630	0.06M	0.590	0.499
	Angle Embedding	Strongly Entangling	1	63	0.07M	0.519	0.477
			5	315	0.07M	0.455	0.463
			10	630	0.07M	0.443	0.446
	Amplitude Embedding	Strongly Entangling	1	84	1.38M	0.374	0.396
			5	420	1.38M	0.367	0.395
			10	840	1.38M	0.364	0.391



(a) (b)

Figure 5 (a) Mean Squared Error and (b) Mean Absolute Error result of QSegRNN by layers

Table 3 Experiment result of SegRNN and QSegRNN with diverse RNNs

Model	RNN	Quantum embedding layer	Quantum entangling layer	Quantum parameters	Classical parameters	MSE	MAE
SegRNN	LSTM	None	None	0	2.12M	0.368	0.396
SegRNN	GRU	None	None	0	1.63M	0.370	0.395
GRU	None	None	None	0	0.08M	1.126	0.831
PatchTST	None	None	None	0	10.78M	0.370	0.400
FEDformer	None	None	None	0	20.68M	0.376	0.415
Informer	None	None	None	0	11.33M	0.941	0.769
Dlinear	None	None	None	0	0.14M	0.375	0.399
MICN	None	None	None	0	21.24M	0.421	0.431
QSegRNN	HQGRU	Angle Embedding	Basic Entangling	63	0.06M	0.482	0.457
QSegRNN	HQGRU	Angle Embedding	Strongly Entangling	630	0.07M	0.443	0.446
QSegRNN	HQGRU	Amplitude Embedding	Strongly Entangling	840	1.38M	0.364	0.391

amplitude embedding input size, which increases classical parameters, allows the model to leverage a larger parameter space during training. As illustrated in Table 2 and Fig. 5, QSegRNN with amplitude embedding exhibits a reduction in MSE and MAE, emphasizing the advantages of hybrid architecture. Given its superior performance trends with increasing quantum depth, we conclude that this structure is well-suited for QSegRNN.

Next, we compare QSegRNN with diverse models usually experimented in the time series forecasting tasks. The experiment result of QSegRNN, with each method, we selected the model in Table 2 that illustrates the best performance. The experiment result, as shown in Table 3, shows that QSegRNN with amplitude embedding and strongly entangling layer achieves better performance on MSE and MAE than the classical forecasting models and SegRNN constructed with GRU and LSTM, even though our model has only about 85 and 64 percent of the parameters compared to SegRNN. Especially, our model achieved higher performance than the QSegRNN model with angle embedding and basic entangling layer and angle embedding and strongly entangling layer, due to the ability of encoding larger information with amplitude embedding.

In the noise experiment, we experimented with the effect of noise on QSegRNN with depolarization noise [31]. In the depolarizing channel model, each Pauli error X, Y, and Z occurs with an equal probability of p. We experimented with noise rates of 0.1, 0.01, and 0.001 since these rates are commonly used in noise experiments [32–34]. As shown in Table 4, the MAE of QSegRNN with angle embedding and strongly entangling layer

Table 4 Result of noise experiment

Model	Quantum embedding layer	Quantum entangling layer	Error rate	MAE
QSegRNN	Angle Embedding	Basic Entangling	0	0.457
			0.001	0.457
			0.01	0.459
			0.1	0.472
	Angle Embedding	Strongly Entangling	0	0.446
			0.001	0.447
			0.01	0.449
			0.1	0.458
	Amplitude Embedding	Strongly Entangling	0	0.391
			0.001	0.391
			0.01	0.392
			0.1	0.401

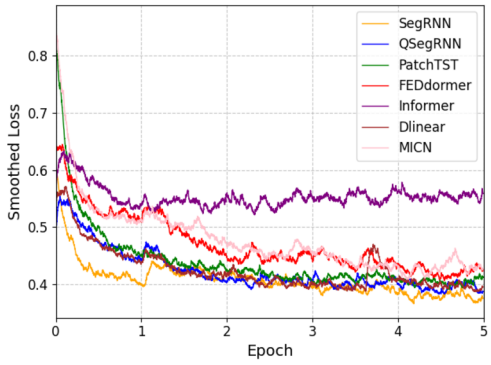


Figure 6 Convergence performance of forecasting models

decreased by 0.01 when the error rate was 0.1, while the other error rate shows almost no difference from the model. Compared to other methods, our model demonstrates robustness to noise across all error rates. This result illustrates that the hybrid model architecture enhances noise robustness by incorporating a classical layer alongside quantum circuits.

Comparison of the aspect of model convergence, as shown in Fig. 6, Informer [7] converges early but does not guarantee higher performance compared to other models. However, compared to other classical time series forecasting models, since our model is constructed with a hybrid structure, QSegRNN does not take advantage of convergence speed. Note that these results do not mean that the quantum model can not achieve training efficiency. It is noteworthy that QSegRNN takes advantage of lower parameter size, leading to a decrease in training cost. Finally, the forecasting result of our model in Fig. 7 shows that our model's forecasting value is similar to SegRNN. In addition, for the model efficiency, we discuss in Sect. 6.

6 Discussion

In this section, we will discuss the efficiency and the impact of noise on our model. For the aspect of model efficiency, we compare QSegRNN with the classical counterpart, SegRNN. Our model follows the SegRNN architecture, but instead of a classical GRU, we employ HQGRU, distinguishing it from the original design. Therefore, we discuss the efficiency by comparing the RNN unit for each model.

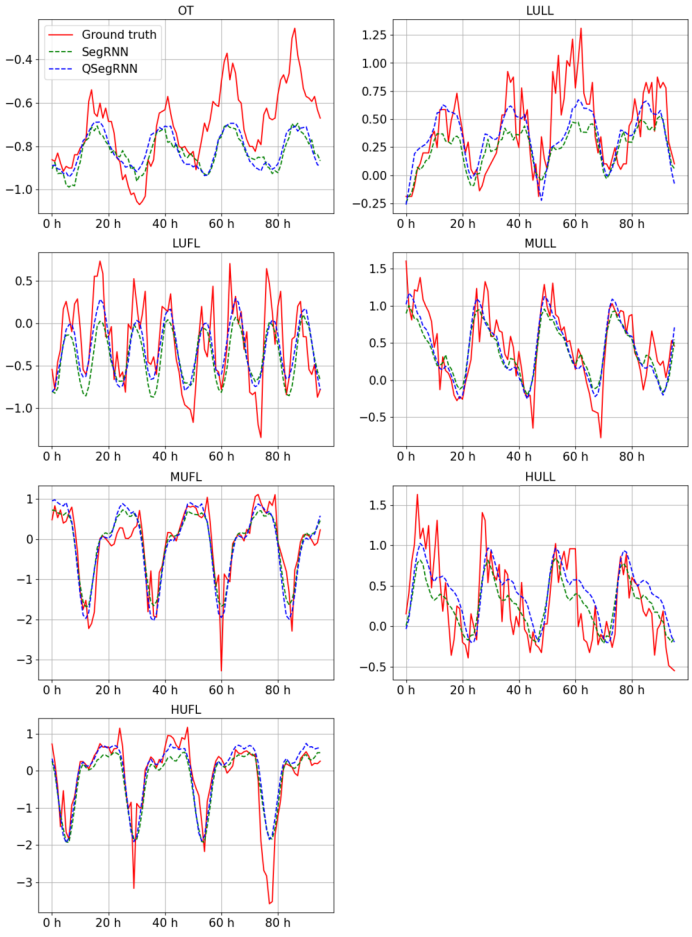


Figure 7 Normalized forecasting values of SegRNN and QSegRNN with ground truth

1) GRU: SegRNN introduces the traditional GRU [11], which consists of an update gate, a reset gate, and an output gate. The architecture of the GRU is illustrated in Fig. 8. For each gate, given the d-dimensional input in each linear layer, the h-dimensional hidden state, the parameter can be calculated as $d \times h$. The parameter of hidden state matrix size can be represented as h^2. The parameter at the bias of GRU is h. Therefore, the total parameter of GRU is calculated as $3dh + 3h^2 + 3h$, since the parameter of bias is small, the parameter can also be represented as $3(d + h)h$.

2) HQGRU: The HQGRU in QSegRNN is also constructed with the same gates as the counterpart, GRU. However, HQGRU employs 2 linear embedding layers to match the quantum circuit's input shape in Fig. 2. For each gate, consider d-dimesional input, the parameter can be calculated as $(d + h) \times q$, where q denotes the number of qubits. Also, the parameter of hidden state matrix size can be represented as $h \times q$. The number of parameters of QCNN depends on the depth of quantum layer number L and qubit q. Finally, the total number of quantum parameters in HQGRU can be represented as $3qL$ due to the three gates. Therefore, the overall parameter of HQGRU can be calculated as follows: $2(d + \frac{5}{2}h)q + 3qL$. The circuit depth of HQGRU with amplitude embedding can be represented as $O(poly(q)) + (18 + q)L$ and the HQGRU with angle embedding's circuit depth can be represented as $O(q) + (18 + q)L$. Here, the *poly* denotes a polynomial.

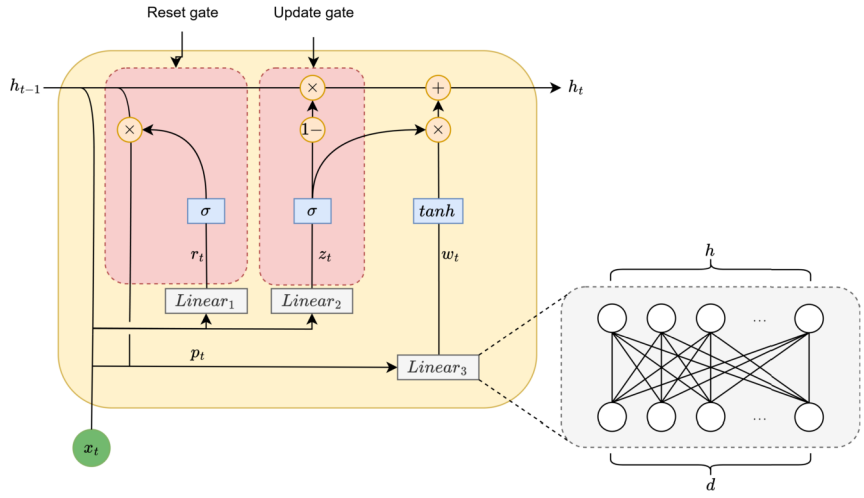


Figure 8 Architecture of classical GRU

Table 5 Model analysis of QSegRNN compared to SegRNN

Model	# Cell	# Classical parameters	# Quantum parameters	Circuit depth	# Gates	
					U(2)	U(4)
HQGRU-Amplitude	1	$2(d+\frac{5}{2}h)q$	$3qL$	$O(\text{poly}(q))+(18+q)L$	$24qL$	$4qL$
HQGRU-Angle	1	$2(d+\frac{5}{2}h)q$	$3qL$	$O(q)+(18+q)L$	$24qL$	$4qL$
GRU	1	$3(d+h)h$	–	–	–	–

Comparison: In terms of the number of parameters, the HQGRU can achieve the more efficiency than the GRU. The detailed information is listed in Table 5, where the number of gate include only with respect to the strongly entangling layer's gates. However, the circuit depth represents the required circuit depth for each embedding method. To demonstrate the efficiency of QSegRNN compared to SegRNN, we calculate the number of parameters of HQGRU and classical GRU under the same input dimension. We prove the efficiency of the HQGRU in the following. After, we compare the angle embedding with amplitude embedding in the HQGRU circuit to evaluate the feasibility of the physical implementation on NISQ devices.

Proof To compare HQGRU with classical GRU, assuming that the parameter of GRU is the same as the parameter of HQGRU, the equation can be calculated as follows:

$$3(d+h)h = 2(d+\frac{5}{2}h)q + 3qL \tag{21}$$

$$3(d+h)h = q(2d+5h+3qL) \tag{22}$$

$$q = \frac{3(d+h)h}{2d+5h+3L} \tag{23}$$

Here, since the L is a small value, we can ignore this value.

$$q = 3h \times \frac{d+h}{2(d+h)+3h} \tag{24}$$

In Eq. (24), as both of the d and h values become larger, the q value becomes smaller. Therefore, we can inspect that except in cases where d and h are small, such as d is 2 and h is 2, the q value is determined to be smaller than both d and h values. Therefore, the total parameters of HQGRU can also be calculated as smaller than classical GRU for the same input dimension. Consequently, QSegRNN offers greater cost efficiency compared to SegRNN. □

We compare the angle embedding with amplitude embedding. Here, we compare the embedding methods with the two points of view. First, we assume the fixed qubit number, resulting fixed parameter number. In this case, HQGRU with amplitude embedding shows $O(poly(q)) + (18 + q)L$ circuit depth, while HQGRU with angle embedding shows $O(q) + (18 + q)L$. $O(q)$ in circuit depth of angle embedding compared to $O(poly(q))$ in circuit depth of the amplitude embedding illustrates that HQGRU with amplitude embedding leads to a more complex design since $poly(q)$ is more complex than q. Also, since the HQGRU with amplitude embedding requires longer coherence time, it might be difficult to implement the amplitude embedding for high-dimensional input data in practice [35]. Note that current NISQ devices can implement up to 512 quantum volume (QV) [35], where QV gives a guideline to users of the quantum devices as to what circuit sizes and depths appear reasonable to run on the device.

Secondly, we assume the fixed number of the QCNN qubits q. In this case, we set the output dimension in the linear embedding layer as q and 2^q for angle embedding and amplitude embedding, respectively. Since 2^q becomes greater than q when q is over 0, the HQGRU with amplitude embedding can preserve more information from the linear embedding layer. However, due to the increase in linear embedding layer output shape, the training cost and the number of parameters also increase. To address this challenge, approximate amplitude encoding [36] or a transformer-decoder-based amplitude encoder [37] can be utilized to reduce circuit depth and lower training costs.

For the aspect of noise, to inspect the experiment results in Table 4, we observed the performance decreased as the noise rate increased. These results mean that the noise affects the gradient calculation. We prove the effect of the noise on the gradient in the following.

Proof We demonstrate the effect of noise on the model. Initially, the noise experiment in a quantum circuit can be represented as follows:

$$\langle \hat{B} \rangle(\theta) = Tr(\hat{B}U(\theta)\rho U^{\dagger}(\theta)) \tag{25}$$

Here, ρ denotes an initial quantum state and $\hat{B}$ denotes measurement observation. While optimizing the quantum circuit, the parameter shift rule [28, 29] is commonly applied.

$$\nabla_{\theta}\langle \hat{B} \rangle(\theta) = \frac{1}{2}[\langle \hat{B} \rangle(\theta + \frac{\pi}{2}) - \langle \hat{B} \rangle(\theta - \frac{\pi}{2})] \tag{26}$$

In this step, the noisy channel Λ is applied to the end of gates.

$$\langle \hat{B} \rangle(\theta) = Tr(\hat{B}\Lambda[U(\theta)\rho U^{\dagger}(\theta)]) \tag{27}$$

The observable $\hat{B}$ can be transformed to $\hat{B} = \Lambda^{\dagger}[\hat{B}] = \hat{B}'$ with Heisenberg picture [38].

$$\langle \hat{B} \rangle(\theta) = Tr(\hat{B}'U(\theta)\rho U^{\dagger}(\theta)) = \langle \hat{B}' \rangle(\theta) \tag{28}$$

Therefore, Eq. (26) and Eq. (28) can be transformed as follows:

$$\nabla_\theta \langle \hat{B} \rangle(\theta) = \frac{1}{2}[\langle \hat{B}' \rangle(\theta + \frac{\pi}{2}) - \langle \hat{B}' \rangle(\theta - \frac{\pi}{2})] \tag{29}$$

In Eq. (29), we can observe that noise affects the gradient calculation, resulting in a decrease in performance. □

To address this issue, a mitigation strategy could be applied, such as measurement error mitigation [39] or zero-noise extrapolation [40].

In summary, QSegRNN adopts a hybrid quantum-classical structure, enabling it to be built with fewer parameters than the classical SegRNN. QSegRNN consists of amplitude embedding, leading to the reduction of qubits. Experimental results illustrate the performance is improved as the layer increases. Also, our model shows robustness at noise, except when the error rate is 0.1. These findings imply that QSegRNN still needs to be optimized with mitigation strategies since it is not entirely free from noise. Our model requires polynomial circuit depth due to amplitude embedding, requiring long coherence time.

7 Conclusion

Recent advancements in various AI technologies have led to managing and analyzing vast data, resulting in significant heat generation. Among the infrastructure of data centers, the temperature monitoring and management of electricity transformers, through forecasting the electricity transformer temperature, can greatly impact transformer performance. Consequently, a large body of research has been conducted on this topic. Even though research on AI-based forecasting has been actively conducted, the model's size can sometimes surpass the limits of available computing resources. Therefore, this paper proposes QSegRNN, a hybrid time series forecasting model incorporating quantum computing.

QSegRNN employs a QCNN-based quantum circuit with amplitude embedding and a strongly entangling layer to minimize information loss during HQRNN training. It has 300K fewer parameters than SegRNN, enabling training and execution on smaller computing devices. Furthermore, it achieves better performance than SegRNN's performance despite having fewer parameters. This result demonstrates that quantum computing can achieve similar performance with fewer parameters than conventional AI layers, suggesting the potential for more efficient AI training in the future.

Our model can be applied to the temperature control of transformers and various time series forecasting tasks. As large AI models are being developed, conventional AI models often face memory limitations. The QSegRNN model, with its smaller size, offers the possibility of deployment on a variety of devices, even when memory constraints prevent the use of larger AI models.

However, our model still contains many parameters, making it unsuitable for deployment on some resource-constrained devices. Additionally, since our model uses amplitude embedding, there is challenge: amplitude embedding requires deep circuit depth, where current NISQ devices may not cover this issue.

For future studies, we'll focus on reducing model complexity to enhance scalability and deployment feasibility while maintaining prediction accuracy. We'll also explore more efficient quantum algorithms and hybrid optimization techniques to mitigate computational

costs and improve stability. Additionally, we'll develop and experiment with diverse strategies to adapt the model to various quantum hardware platforms, which could enhance its applicability in real-world scenarios.

Acknowledgements
Not applicable.

Author contributions
K.H.M. contributed to methodology, implementation, experiment, and writing the original draft. S.G.J. was contributed to conceptualization, validation, and writing – review and editing. W.J.H. provided supervision. All authors reviewed the manuscript.

Funding
This work was supported in part by Quantum Computing based on Quantum Advantage challenge research(RS-2024-00408613) through the National Research Foundation of Korea(NRF) funded by the Korean government (Ministry of Science and ICT(MSIT)); in part by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT)(RS-2024-00336962); in part by the Korea Agency for Infrastructure Technology Advancement(KAIA) grant funded by the Ministry of Land Infrastructure and Transport (No. RS-2023-00256816); in part by Institute of Information & communications Technology Planning & Evaluation (IITP) under the Artificial Intelligence Convergence Innovation Human Resources Development (IITP-2025-RS-2023-00254177) grant funded by the Korea government(MSIT); in part by the IITP(Institute of Information & Coummunications Technology Planning & Evaluation)-ITRC(Information Technology Research Center) grant funded by the Korea government(Ministry of Science and ICT)(IITP-2025-RS-2023-00260098); and in part by 2023 BK21 FOUR Program of Pusan National University.

Data Availability
No datasets were generated or analysed during the current study.

Declarations

Competing interests
The authors declare no competing interests.

Received: 9 January 2025 Accepted: 24 February 2025 Published online: 03 March 2025

References

1. Jones N, et al. How to stop data centres from gobbling up the world's electricity. Nature. 2018;561(7722):163–6.
2. Lin S, Lin W, Wu W, Zhao F, Mo R, Zhang H. Segrnn: Segment recurrent neural network for long-term time series forecasting. 2023. arXiv preprint. arXiv:2308.11200.
3. Nie Y, Nguyen NH, Sinthong P, Kalagnanam J. A time series is worth 64 words: Long-term forecasting with transformers. 2022. arXiv preprint. arXiv:2211.14730.
4. Zeng A, Chen M, Zhang L, Xu Q. Are transformers effective for time series forecasting? In: Proceedings of the AAAI conference on artificial intelligence. vol. 37. 2023. p. 11121–8.
5. Wang H, Peng J, Huang F, Wang J, Chen J, Xiao Y. Micn: multi-scale local and global context modeling for long-term series forecasting. In: The eleventh international conference on learning representations. 2023.
6. Zhou T, Ma Z, Wen Q, Wang X, Sun L, Jin R. Fedformer: frequency enhanced decomposed transformer for long-term series forecasting. In: International conference on machine learning. PMLR; 2022. p. 27268–86.
7. Zhou H, Zhang S, Peng J, Zhang S, Li J, Xiong H, Zhang W. Informer: beyond efficient transformer for long sequence time-series forecasting. In: Proceedings of the AAAI conference on artificial intelligence. vol. 35. 2021. p. 11106–15.
8. Bergholm V, Izaac J, Schuld M, Gogolin C, Ahmed S, Ajith V, Alam MS, Alonso-Linaje G, AkashNarayanan B, Asadi A, et al. Pennylane: Automatic differentiation of hybrid quantum-classical computations. 2018. arXiv preprint. arXiv:1811.04968.
9. Liu J, Chen S. Timesurl: self-supervised contrastive learning for universal time series representation learning. In: Proceedings of the AAAI conference on artificial intelligence. vol. 38. 2024. p. 13918–26.
10. Das A, Kong W, Sen R, Zhou Y. A decoder-only foundation model for time-series forecasting. 2023. arXiv preprint. arXiv:2310.10688.
11. Cho K, Van Merriënboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, Bengio Y. Learning phrase representations using rnn encoder-decoder for statistical machine translation. 2014. arXiv preprint. arXiv:1406.1078.
12. Jeong S-G, Do QV, Hwang W-J. Short-term photovoltaic power forecasting based on hybrid quantum gated recurrent unit. ICT Express. 2024;10(3):608–13.
13. Cao Y, Zhou X, Fei X, Zhao H, Liu W, Zhao J. Linear-layer-enhanced quantum long short-term memory for carbon price forecasting. Quantum Mach Intell. 2023;5(2):26.
14. Hochreiter S, Schmidhuber J. Long short-term memory. Neural Comput. 1997;9(8):1735–80.
15. Chen SY-C, Yoo S, Fang Y-LL. Quantum long short-term memory. In: ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE; 2022. p. 8622–6.
16. Jin M, Wang S, Ma L, Chu Z, Zhang JY, Shi X, Chen P-Y, Liang Y, Li Y-F, Pan S, et al. Time-llm: Time series forecasting by reprogramming large language models. 2023. arXiv preprint. arXiv:2310.01728.
17. Wu H, Hu T, Liu Y, Zhou H, Wang J, Long M. Timesnet: Temporal 2d-variation modeling for general time series analysis. 2022. arXiv preprint. arXiv:2210.02186.

18. Song Z, Zhou X, Xu J, Ding X, Shan Z. Recurrent quantum embedding neural network and its application in vulnerability detection. Sci Rep. 2024;14(1):13642.
19. Sarkar S, Naug A, Luna R, Guillen A, Gundecha V, Ghorbanpour S, Mousavi S, Markovikj D, Babu AR. Carbon footprint reduction for sustainable data centers in real-time. In: Proceedings of the AAAI conference on artificial intelligence. vol. 38. 2024. p. 22322–30.
20. Singh P. Fqtsfm: a fuzzy-quantum time series forecasting model. Inf Sci. 2021;566:57–79.
21. Chittoor HHS, Griffin PR, Neufeld A, Thompson J, Gu M. Qultsf: Long-term time series forecasting with quantum machine learning. 2024. arXiv preprint. arXiv:2412.13769.
22. Stoudenmire E, Schwab DJ. Supervised learning with tensor networks. Adv Neural Inf Process Syst. 2016;29.
23. Cong I, Choi S, Lukin MD. Quantum convolutional neural networks. Nat Phys. 2019;15(12):1273–8.
24. Schuld M, Bocharov A, Svore KM, Wiebe N. Circuit-centric quantum classifiers. Phys Rev A. 2020;101(3):032308.
25. Liang Y, Peng W, Zheng Z-J, Silvén O, Zhao G. A hybrid quantum–classical neural network with deep residual learning. Neural Netw. 2021;143:133–47.
26. LeCun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. Proc IEEE. 1998;86(11):2278–324.
27. Pesah A, Cerezo M, Wang S, Volkoff T, Sornborger AT, Coles PJ. Absence of barren plateaus in quantum convolutional neural networks. Phys Rev X. 2021;11(4):041011.
28. Mitarai K, Negoro M, Kitagawa M, Fujii K. Quantum circuit learning. Phys Rev A. 2018;98(3):032309.
29. Schuld M, Bergholm V, Gogolin C, Izaac J, Killoran N. Evaluating analytic gradients on quantum hardware. Phys Rev A. 2019;99(3):032331.
30. Vaswani A. Attention is all you need. Adv Neural Inf Process Syst. 2017.
31. Urbanek M, Nachman B, Pascuzzi VR, He A, Bauer CW, Jong WA. Mitigating depolarizing noise on quantum computers with noise-estimation circuits. Phys Rev Lett. 2021;127(27):270502.
32. Liang Z, Wang Z, Yang J, Yang L, Shi Y, Jiang W. Can noise on qubits be learned in quantum neural network? A case study on quantumflow. In: 2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD). IEEE; 2021. p. 1–7.
33. Skolik A, Mangini S, Bäck T, Macchiavello C, Dunjko V. Robustness of quantum reinforcement learning under hardware errors. EPJ Quantum Technol. 2023;10(1):8.
34. Somogyi W, Pankovets E, Kuzmin V, Melnikov A. Method for noise-induced regularization in quantum neural networks. 2024. arXiv preprint. arXiv:2410.19921.
35. Pelofske E, Bärtschi A, Eidenbenz S. Quantum volume in practice: what users can expect from nisq devices. IEEE Trans Quantum Eng. 2022;3:1–19.
36. Nakaji K, Uno S, Suzuki Y, Raymond R, Onodera T, Tanaka T, Tezuka H, Mitsuda N, Yamamoto N. Approximate amplitude encoding in shallow parameterized quantum circuits and its application to financial market indicators. Phys Rev Res. 2022;4(2):023136.
37. Daimon S, Matsushita Y-i. Quantum circuit generation for amplitude encoding using a transformer decoder. Phys Rev Appl. 2024;22(4):041001.
38. Heisenberg W. Quantum-theoretical re-interpretation of kinematic and mechanical relations. Z Phys. 1925;33:879–93.
39. Funcke L, Hartung T, Jansen K, Kühn S, Stornati P, Wang X. Measurement error mitigation in quantum computers through classical bit-flip correction. Phys Rev A. 2022;105(6):062404.
40. He A, Nachman B, Jong WA, Bauer CW. Zero-noise extrapolation for quantum-gate error mitigation with identity insertions. Phys Rev A. 2020;102(1):012426.

Publisher's Note