Graphical Abstract

QSegRNN: Quantum Segment Recurrent Neural Network For Time Series Forecasting

Kyeong-Hwan Moon, Seon-Geun Jeong, Won-Joo Hwang

Highlights

QSegRNN: Quantum Segment Recurrent Neural Network For Time Series Forecasting

Kyeong-Hwan Moon, Seon-Geun Jeong, Won-Joo Hwang

- Quantum-RNN with amplitude embedding inherits more information than with angle embedding
- The quantum machine learning model allows the classical model to operate with fewer parameters
- QSegRNN with more quantum layers achieves higher performance compared to other quantum circuits

QSegRNN: Quantum Segment Recurrent Neural Network For Time Series Forecasting

Kyeong-Hwan Moon, Seon-Geun Jeong, Won-Joo Hwang

[a]*School of Computer Science and Engineering, Pusan National University, 2, Busandaehak-ro 63beon-gil, Geumjeong-gu, Busan, 46241, Gyeongsangnam-do, Republic of Korea*
[b]*Department of Information Convergence Engineering,Pusan National University, 2, Busandaehak-ro 63beon-gil, Geumjeong-gu, Busan, 46241, Gyeongsangnam-do, Republic of Korea*

Abstract

Recently a myriad of 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. However, 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 impossible 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 conventional AI 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 is overperformed to that of SegRNN and other forecasting models even though QSegRNN has only 85 percent of the parameters.

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

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

Moore's Law, proposed by Gordon E. Moore [1], predicted the doubling of transistors on microchips every two years, driving significant advancements in computing power. Since this law was passed in recent years, the demand for services and storage has also been satisfied. While this principle has shaped technological progress for decades, recent limitations in transistor scaling raise questions about its future relevance [2]. Additionally, it has been noted that global data center electricity usage could reach alarming levels, contributing significantly to environmental strain if not managed effectively [3]. Also, recent studies have shown that data centers contribute significantly to carbon emissions, with sustainable energy usage and power optimization becoming a primary concern.

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.

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 [4]. Several studies have attempted to address this issue using artificial intelligence models to forecast transformer temperatures [5, 6, 7, 8, 9, 10]. 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 model cannot be trained or deployed.

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 has fewer parameters than the similarly structured segment recurrent neural network [5], yet it produces higher results. 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 [11] 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 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.

- When adding a quantum circuit to the model, due to the limitation of the number of qubits in noisy intermediate-scale quantum (NISQ) devices, information loss occurs considerably depending on the embedding method. To alleviate this issue, we employed amplitude embedding which makes quantum circuits influence our model. This approach facilitated the transformation of the original SegRNN into QSegRNN, reducing the number of parameters while minimizing performance degradation.

- Our model offers 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 Section 2, we describe the related works of the QSegRNN. Section 3 introduces the structure of QSegRNN. The structure of QSegRNN is partitioned into the encoding and decoding phases. Simulation results are presented in Section 4. Finally, the conclusion is represented in Section 5.

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 [5, 6, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17, 18, 19, 20]. 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 [8, 12, 13, 19, 20]. M. Jin et al. [19] 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. [13] proposed a decoder-only forecasting model to enhance time series prediction performance. J. Liu et al. [12] 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. [20] proposed TimesNet, which enhanced time series forecasting performance by transforming 1D data into 2D. H. Wang et al. [8] enhanced forecasting performance using a time series decomposition method that leverages local features and global correlations.

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. [18] proposed Quantum Long Short Term Memory (QLSTM), a hybrid quantum-classical machine learning model, to adopt quantum computing into the classical LSTM [17]. Y. Cao et al. [16] 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. However, these models still sometimes suffer from barren plateaus due to the architecture of each model. Therefore, S.G. Jeong et al. [15] proposed Hybrid Quantum-Classical Gated Recurrent Unit (HQGRU) to address the limitations of traditional deep learning models like LSTM [17] and GRU [14], which require substantial computational resources due to their large number of parameters. Additionally, it overcomes the issue of barren plateaus encountered in QLSTM [18], 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 [21], a quantum convolution layer [22], and a strongly entangling layer [23]. 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 [16], and GRU, achieving similar performance with fewer parameters.

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) [22] with amplitude embedding [11] and a strongly entangling layer [23].

3. Preliminaries

3.1. Segment Recurrent Neural Network

Segment Recurrent Network (SegRNN) [5] is a time series forecasting model that addresses the gradient problem, which arises from performing many recurrent iterations in traditional models like Long Stort Term Memory (LSTM) [17] and Gated Recurrent Unit (GRU) [14]. 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

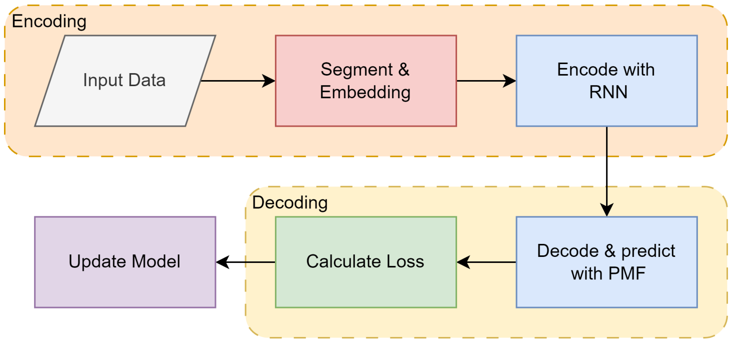


Figure 1: Training progress of SegRNN.

predicted sequences are compared with the original data to update the model. The training process of SegRNN is shown in Figure 1.

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. In this paper, we replace the Recurrent Neural Network (RNN) in the SegRNN model with a modified hybrid quantum gated recurrent unit (HQGRU) [15] 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 [21] and amplitude embedding [11] applied in the quantum GRU [15] cell.

Angle embedding is an embedding layer that encodes N features into the rotation of n qubits, where $N \leq n$ [21]. In traditional physics, a tensor can be represented like equation 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 used like equation 3.

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

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

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

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 [11].

Amplitude embedding is another embedding method that encodes 2^n features into the amplitudes of an n-qubit state [11]. This feature of amplitude embedding allows embedding many features to a few qubits compared to angle embedding. With this embedding algorithm, a 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) - also called parameterized quantum circuit (PQC) - is a type of quantum circuit that contains trainable parameters. These parameters work similarly to the weights in artificial neural networks. VQC-based algorithms have been proposed since Quantum Machine Learning (QML) emerged recently for resolving artifacts of artificial neural networks that utilize a lot of parameters in each model.

For example, the VQC was studied to avoid barren plateaus, where the loss suffers from decreasing because the loss landscape is flat [24]. VQC was also used to construct a hybrid quantum-based GRU model, replacing gates in the original GRU model [15].

The VQC allows the construction of flexible model depth and resistance 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 set of QML layers that perform a function similar to a convolutional neural network (CNN) in conventional machine learning [22]. Classical CNN consists of a convolution layer and a pooling layer [25]. 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 and pools 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. The quantum state of an i-th layer of QCNN can be expressed as follows [22]:

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

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 [11] and strongly entangling layer [23] in the quantum GRU [15] 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 (QSeg-RNN), a hybrid model that applies quantum machine learning (QML) to achieve performance similar to classical deep learning models with fewer parameters.

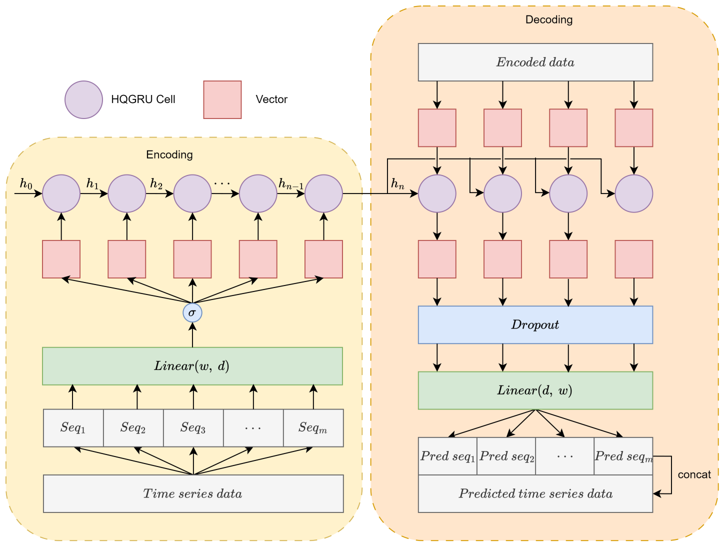


Figure 2: Overall architecture of QSegRNN.

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. The structure of QSegRNN is shown in Figure 2.

Our model replaces the RNN cell employed in the encoding and decoding phases of SegRNN [5], which requires many parameters, with the HQGRU-based [15] QML model. This substitution reduces the overall size of the model while maintaining the performance. This approach reduces the number of parameters from 1.6 million in SegRNN to 1.3 million in QSegRNN. 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 Equation 7, while n is decided by dividing the prediction

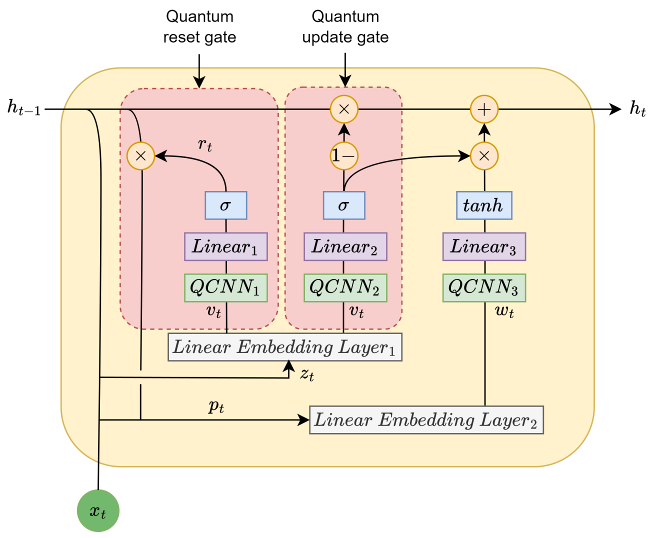


Figure 3: Architecture of RNN in QSegRNN.

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 RNN's hidden state. This layer not only allows the size of information to be forwarded in the RNN but also can expand the information dimension depending on the hidden state hyperparameter.

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

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

$$E_X = RNN(X_{embeded}) \tag{9}$$

At this stage, the RNN employs HQGRU [15], which its structure is shown in Figure 3. As shown in Figure 4, the QCNN layer uses the convolution circuit proposed by I. MacCormack et al. [22]. With this circuit, a quantum convolution layer in RNN can be constructed. In the structure of RNN's QCNN layer, Amplitude embedding [11] is employed to minimize information loss at each gate during the RNN forwarding process, and a strongly entangling layer [23] is added at the end to complete the RNN's VQC layer. This hybrid QML structure allows the RNN 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 RNN's formulation is as follows:

$$z_t = [h_{t-1}, x_t] \tag{10}$$

$$v_t = LE_1(z_t) \tag{11}$$

$$r_t = \sigma(L_3(QCNN_1(v_t))) \tag{12}$$

$$u_t = \sigma(L_4(QCNN_2(v_t))) \tag{13}$$

$$w_t = LE_2(r_t \otimes h_{t-1}) \tag{14}$$

$$h_t = tanh(L_3(QCNN_3(w_t))) \otimes u_t + (1 - u_t) \otimes h_{t-1} \tag{15}$$

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 RNN 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. 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 Figure 3. Each gate has a QCNN layer, linear layer, and sigmoid activation function. Like the structure of GRU, the output of quantum reset gate r_t is multiplied by 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 RNN of QSegRNN allow the layers to capture global features due to entanglement. The superposition of quantum allow the model to be constructed with a few parameters. Therefore our model can overperform the classical SegRNN model with fewer parameters.

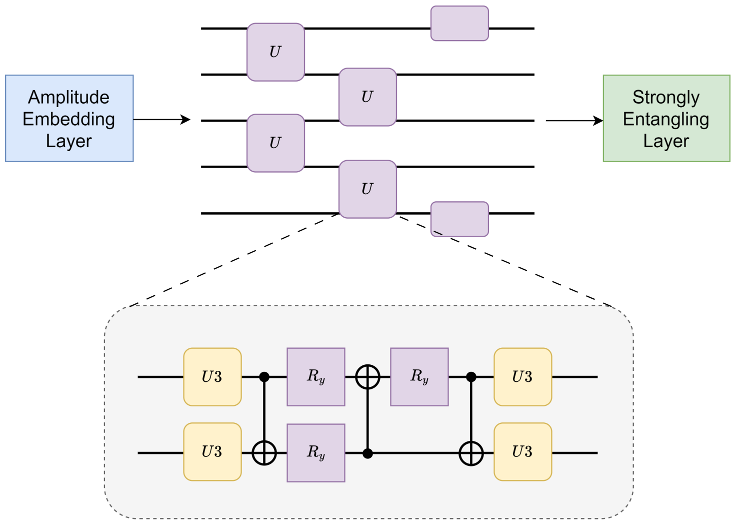


Figure 4: Quantum circuit used in QCNN layer in RNN.

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.

$$P = [p_e, c_e] \tag{16}$$

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

$$X_{union} = \bigcup_{i=1}^{m} E_X \tag{17}$$

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

$$D_X = RNN(P, X_{union}) \tag{18}$$

Finally, D_X is passed through a dropout mask r and a linear layer L_2 to obtain the prediction P_X.

$$P_X = L_2(r \otimes D_X) \tag{19}$$

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 over the prediction length using a loss function, and the model is updated accordingly. During the model's parameter update process, since the RNN is a QML model, the parameter-shift rule is employed to update the model [26, 27]. The pseudo-code for QSegRNN is shown in Algorithm 1.

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-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 design our experiments to answer the following questions:

- Does QSegRNN overcome the performance of classical models even if the model has fewer parameters?
- How does the VQC with amplitude embedding layer affect QSegRNN compared to the VQCs with angle embedding layer?

5.1. Experiment Setup

The experiments utilize the electricity transformer temperature (ETT) dataset provided by H. Zhou et al. [10]. The ETT dataset is recorded data of the temperature information of transformers in a time-series format. This study specifically uses the ETT1 dataset for the experiments. 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. 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 is 0.001. All RNN models consist of 512 hidden layers, a look-back of 720, a segment length of 48, a prediction length of 96, and a dropout rate of 0.5. The quantum RNN's numbers of qubits are set to 7 to match the shape of the

Algorithm 1 Training process of QSegRNN.

```
1: **Input**: Sequence data $X$.
2: **Output**: Predicted sequence data $P_X$.
3: for $epoch = 1, 2, \dots, 5$ do
4:     for $iteration = 1, 2, \dots, 7800$ do
5:         Partition input sequence data $X$:
               $$X_{partitioned} = \{X_1, X_2, \dots, X_{n-1}, X_n\}$$
6:         Embed partitioned data $X_{partitioned}$:
               $$X_{embeded} = ReLU(L_1(X_{partitioned}))$$
7:         Encode embedded data $X_{embeded}$:
               $$E_X = RNN(X_{embeded})$$
8:         Construct repeated data $X_{union}$:
               $$X_{union} = \bigcup_{i=1}^{m} E_X$$
9:         Decode unioned data $X_{union}$ and information embedding data $P$:
               $$D_X = RNN(P, X_{union})$$
10:        Predict output sequence $P_X$:
               $$P_X = L_2(r \otimes D_X)$$
11:    end for
12:    Update QSegRNN's parameters $\theta$.
13:    Update learning rate.
14: end for
```

data. Each experiment is repeated 5 times, and the final results are the average of these repetitions. The evaluation metrics are mean squared error (MSE) and mean absolute error (MAE). MSE and MAE are chosen since these metrics are widely used [5, 6, 7, 8, 9, 10]. 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 [11] 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 RNN.
- Finally, we analyze the prediction of classical SegRNN and QSegRNN about the ETT1 dataset.

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

5.2. Experiment Result

In this section, we compared SegRNN and QSegRNN models constructed with diverse RNNs. We experimented with diverse models usually experimented in time series forecasting. And we also experimented with QSegRNN with various embedding and VQCs. The experiment result, as shown in Table 2, shows that QSegRNN with amplitude embedding and strongly entangling layer overperforms MSE and MAE to 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 & basic entangling layer and angle embedding & strongly entangling layer, due to encoding larger information with amplitude embedding. Here, the RNN with angle embedding & basic entangling layer employs angle embedding and a basic entangling layer.

Table 2: 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	315	0.07M	0.443	0.446
QSegRNN	HQGRU	Amplitude Embedding	Strongly Entangling	840	1.38M	0.364	0.391

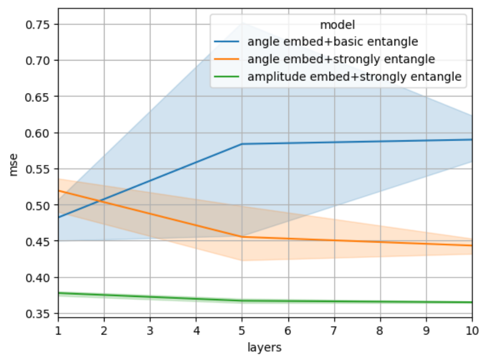
The model with angle embedding & strongly entangling layer employs angle embedding, QCNN, and a strongly entangling layer. Finally, amplitude embedding & strongly entangling layer employs amplitude embedding, QCNN, a strongly entangling layer.

Regarding quantum computing, the experiment result in Table 3 shows whether the quantum layer could affect the model. In QSegRNN with angle embedding & basic entangling layer, the number of quantum layers doesn't guarantee improvement of forecasting performance. QSegRNN with angle embedding & strongly entangling layer achieved an improvement of performance with the number of qubits but does not outperform classical models. However, this result illustrates that the QML model in QSegRNN does not fully affect the model. But the result of QSegRNN with amplitude embedding & strongly entangling layer shows the decrease of loss value when the number of layers increases. Table 3 and Figure 5 show that the QSegRNN with amplitude embedding reduces MSE and MAE when the quantum layer rises. Therefore, with amplitude embedding, our model allows the RNN cell to affect more on inheriting information from previous layers guaranteeing the model to achieve similar performance to SegRNN models.

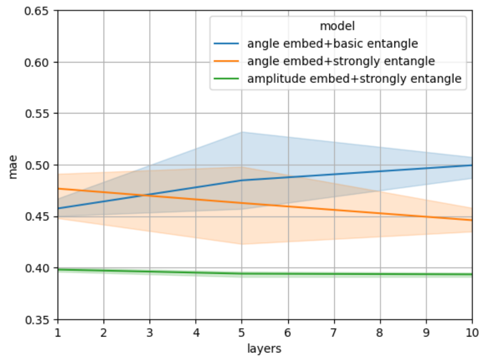
Finally, the forecasting result of our model in Figure 6 shows that our model's forecasting value is similar to SegRNN. In some cases, our model shows strength in forecasting the vertex of data. With this result, we can foresee the possibility

Table 3: 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.

that quantum machine learning could complement classical computing's problem that takes a lot of parameters.

6. 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

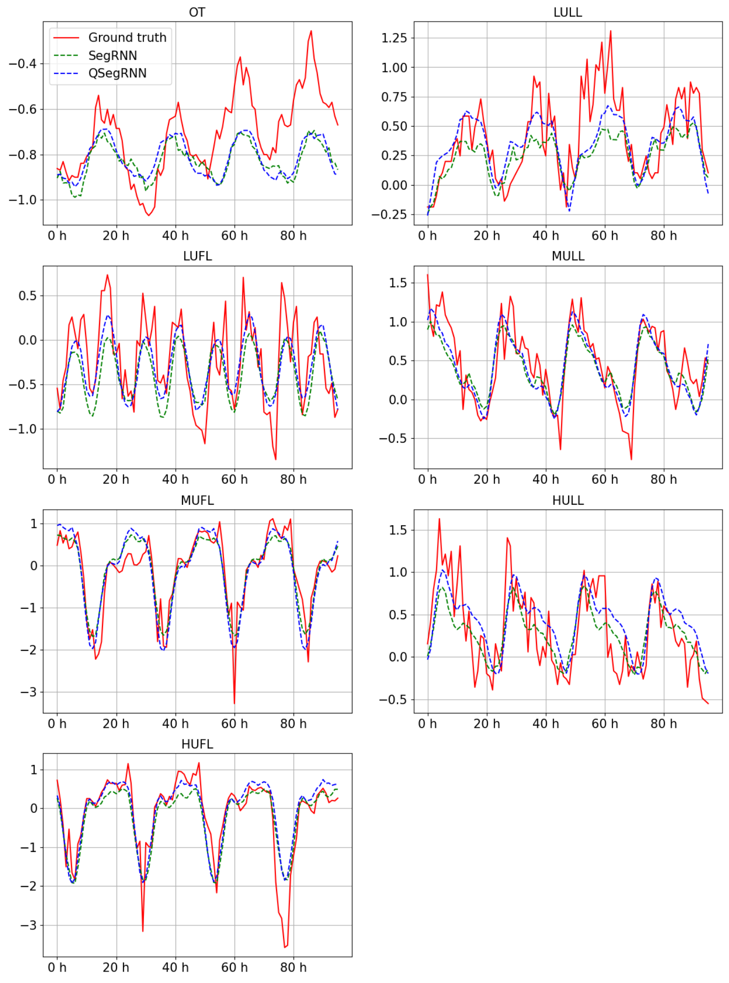


Figure 6: Normalized forecasting values of SegRNN and QSegRNN with ground truth.

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 RNN training. It has 300K fewer parameters than SegRNN, enabling training and execution on smaller computing devices. Furthermore, it overperforms 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.

References

[1] G. E. Moore, Cramming more components onto integrated circuits, Proceedings of the IEEE 86 (1998) 82–85.

[2] J. Shalf, The future of computing beyond moore's law, Philosophical Transactions of the Royal Society A 378 (2020) 20190061.

[3] N. Jones, et al., How to stop data centres from gobbling up the world's electricity, nature 561 (2018) 163–166.

[4] S. Sarkar, A. Naug, R. Luna, A. Guillen, V. Gundecha, S. Ghorbanpour, S. Mousavi, D. Markovikj, A. R. Babu, Carbon footprint reduction for sustainable data centers in real-time, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 38, 2024, pp. 22322–22330.

[5] S. Lin, W. Lin, W. Wu, F. Zhao, R. Mo, H. Zhang, Segrnn: Segment recurrent neural network for long-term time series forecasting, arXiv preprint arXiv:2308.11200 (2023).

[6] Y. Nie, N. H. Nguyen, P. Sinthong, J. Kalagnanam, A time series is worth 64 words: Long-term forecasting with transformers, arXiv preprint arXiv:2211.14730 (2022).

[7] A. Zeng, M. Chen, L. Zhang, Q. Xu, Are transformers effective for time series forecasting?, in: Proceedings of the AAAI conference on artificial intelligence, volume 37, 2023, pp. 11121–11128.

[8] H. Wang, J. Peng, F. Huang, J. Wang, J. Chen, Y. Xiao, Micn: Multi-scale local and global context modeling for long-term series forecasting, in: The eleventh international conference on learning representations, 2023.

[9] T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun, R. Jin, Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting, in: International conference on machine learning, PMLR, 2022, pp. 27268–27286.

[10] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, W. Zhang, Informer: Beyond efficient transformer for long sequence time-series forecasting, in: Proceedings of the AAAI conference on artificial intelligence, volume 35, 2021, pp. 11106–11115.

[11] V. Bergholm, J. Izaac, M. Schuld, C. Gogolin, S. Ahmed, V. Ajith, M. S. Alam, G. Alonso-Linaje, B. AkashNarayanan, A. Asadi, et al., Pennylane: Automatic differentiation of hybrid quantum-classical computations, arXiv preprint arXiv:1811.04968 (2018).

[12] J. Liu, S. Chen, Timesurl: Self-supervised contrastive learning for universal time series representation learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 38, 2024, pp. 13918–13926.

[13] A. Das, W. Kong, R. Sen, Y. Zhou, A decoder-only foundation model for time-series forecasting, arXiv preprint arXiv:2310.10688 (2023).

[14] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using rnn encoder-decoder for statistical machine translation, arXiv preprint arXiv:1406.1078 (2014).

[15] S.-G. Jeong, Q. V. Do, W.-J. Hwang, Short-term photovoltaic power forecasting based on hybrid quantum gated recurrent unit, ICT Express 10 (2024) 608–613.

[16] Y. Cao, X. Zhou, X. Fei, H. Zhao, W. Liu, J. Zhao, Linear-layer-enhanced quantum long short-term memory for carbon price forecasting, Quantum Machine Intelligence 5 (2023) 26.

[17] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural computation 9 (1997) 1735–1780.

[18] S. Y.-C. Chen, S. Yoo, Y.-L. L. Fang, Quantum long short-term memory, in: ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2022, pp. 8622–8626.

[19] M. Jin, S. Wang, L. Ma, Z. Chu, J. Y. Zhang, X. Shi, P.-Y. Chen, Y. Liang, Y.-F. Li, S. Pan, et al., Time-llm: Time series forecasting by reprogramming large language models, arXiv preprint arXiv:2310.01728 (2023).

[20] H. Wu, T. Hu, Y. Liu, H. Zhou, J. Wang, M. Long, Timesnet: Temporal 2d-variation modeling for general time series analysis, arXiv preprint arXiv:2210.02186 (2022).

[21] E. Stoudenmire, D. J. Schwab, Supervised learning with tensor networks, Advances in neural information processing systems 29 (2016).

[22] I. Cong, S. Choi, M. D. Lukin, Quantum convolutional neural networks, Nature Physics 15 (2019) 1273–1278.

[23] M. Schuld, A. Bocharov, K. M. Svore, N. Wiebe, Circuit-centric quantum classifiers, Physical Review A 101 (2020) 032308.

[24] Y. Liang, W. Peng, Z.-J. Zheng, O. Silvén, G. Zhao, A hybrid quantum–classical neural network with deep residual learning, Neural Networks 143 (2021) 133–147.

[25] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86 (1998) 2278–2324.

[26] K. Mitarai, M. Negoro, M. Kitagawa, K. Fujii, Quantum circuit learning, Physical Review A 98 (2018) 032309.

[27] M. Schuld, V. Bergholm, C. Gogolin, J. Izaac, N. Killoran, Evaluating analytic gradients on quantum hardware, Physical Review A 99 (2019) 032331.