

Quantum Temporal Convolutional Networks and Their Application to Weather Forecasting

cyx team

Content

- 1 Background
- 2 Task Description
- 3 Technical Details
- 4 Experiment and Results
- 5 Summary and Future Work

Background

● What is weather forecasting?

To predict weather conditions (e.g. air pressure, temperature) within a certain period in the future by applying the law of atmospheric changes and considering the current and recent meteorological situations

● What are the motivations of weather forecasting?

- The weather become more and more important for national economy and people's livelihood
- There is also an increasing demand for information on weather changes in various industries (e.g. wind power generation)

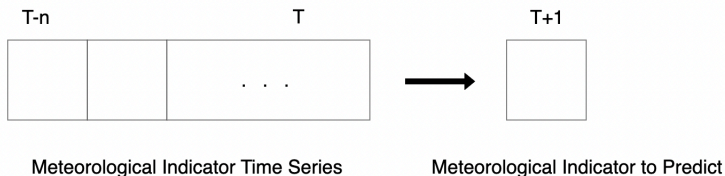
● What are the challenges for weather forecasting?

- The atmospheric system is very complex, and human beings have not fully understand its laws, leading to low prediction accuracies.
- Weather data is usually noisy.

Task Description

Scenario Introduction

The task is to predict five meteorological factors (i.e. atmospheric pressure, minimum temperature, maximum temperature, relative humidity, and wind speed) on the next day, using time series data for the past n days, which is a typical time series forecasting problem.



Task Illustration

Dataset

We focus on an internal desensitized dataset which is a collection of five meteorological indicators observed from 2018-01-01 to 2020-12-31. Each sample contains the following variables: Date, Atmosphere Pressure, Minimum Temperature, Maximum Temperature, Relative Humidity and Wind Speed. .

	Train Set	Test Set
Size	800	200
Granularity	daily	daily
Time Period	2018-01-01 ~ 2020-03-10	2020-06-15 ~ 2020-12-31

Summary of the dataset

Evaluation Metric

Meteorological indicator forecasting accuracy:

$$\text{Accuracy}^i = \left(1 - \sqrt{\frac{1}{N} \sum_{t=1}^N \left(\frac{|y_t^i - \hat{y}_t^i|}{y_t^i} \right)^2} \right)$$

- $i = 1, \dots, 5$ represents the index of meteorological indicators
- N is the number of predictions made for the test set and it is determined by the input size n of the prediction model, namely $N = 200 - n$.

Technical Details

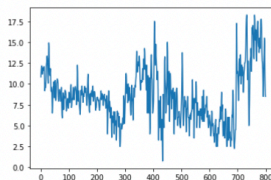
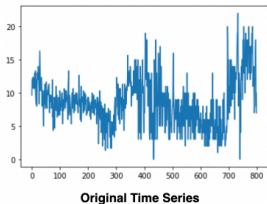
Data Pre-processing

Feature Selection

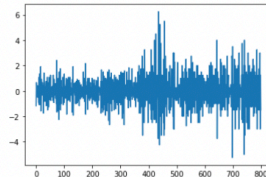
- ✗ **Correlated meteorological indicators**: Considering possible correlations among meteorological indicators, we can use other meteorological indicators as covariates when predicting one meteorological indicator.
- ✗ **Time-related features**: We can extract from existing time variables of our data some additional time-related features (e.g. day-of-the-week, day-of-the-month, month-of-year) which might help the model capture the trend and seasonality of the time series.
- ✓ **Singular spectrum analysis(SSA)**: SSA is an approach to extract different components (e.g. trend, periodicity and noise) from the original time series through time series decomposition and reconstruction. This method is non-parametric and does not introduce additional parameters into the model.

We finally choose SSA as the feature selection method as it performs the best in the experiments!

Singular Spectrum Analysis



ssa_0



ssa_1

Finally select input features as: **original time series + ssa_0 + ssa_1**

Data Normalization: when training machine learning models, especially gradient-based models, we often normalize the input features to the same scale (e.g. $[0,1]$), so that the model can converge faster.

✓ **MinMaxScaler:** to normalize features to the range in $[0, 1]$

$$X_{scaled} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

where x_{max} and x_{min} represent the maximum and minimum values of one feature respectively.

✗ **Quantum MinMaxScaler:** In QML, input features are encoded by quantum gates with period of 2π , so we need to normalized them to the range $[0, 2\pi]$ or $[-\pi, \pi]$ to avoid encoding very different values in the same way

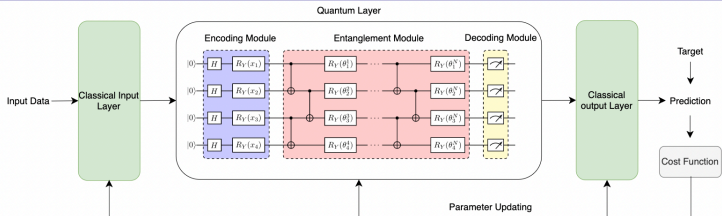
$$X_{scaled} = 2\pi \left(\frac{X - X_{min}}{X_{max} - X_{min}} \right)$$

or

$$X_{scaled} = 2\pi \left(\frac{X - X_{min}}{X_{max} - X_{min}} \right) - \pi$$

We finally found classical MinMaxScaler works better than its quantum counterpart. (It is fine to use classical MinMaxScaler as the normalized data range $[0, 1]$ is still within $[0, 2\pi]$)

Hybrid Quantum-classical Algorithm



● Encoding Module

- Encode classical information into quantum states using different encoding methods (e.g. qubit encoding, amplitude encoding)
- Hadamard gates are usually applied to transform the initial state into a uniform superposition state before the encoding operation.

● Entanglement Module

- Perform unitary evolution of the encoded quantum states using parametric quantum gates
- This module includes a set of single-qubit parametric gates (e.g. $R_Y(x)$) and two-qubit gates (e.g. $CNOT$)

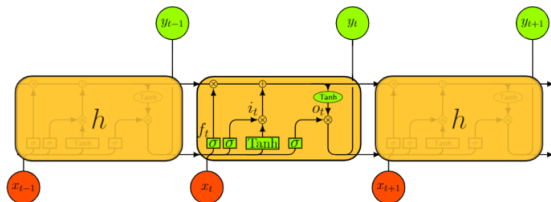
● Decoding Module

- Convert the final quantum states into classical information through measuring a set of observables (e.g. Pauli Z operator)

Classical Model Selection

Three mainstream deep learning models for time series forecasting:

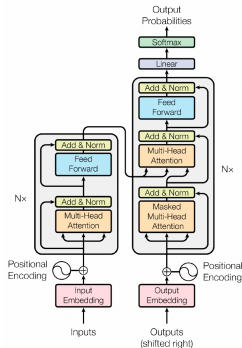
✗ Recurrent neural network-based models (e.g. LSTM)



LSTM Model Architecture [Harrane et al 2021]

- long training time due to recurrent mechanism
- gradient vanishing problem, especially for long time series
- more likely to have overfitting problem due to the large number of parameters

X Transformer-based models



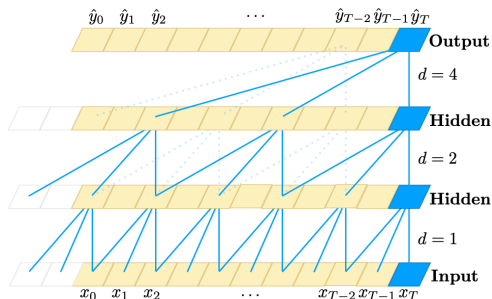
transformer architecture [Vaswani et al 2017]

- The model is lack of the ordering information. So additional time-related features are usually required, which increases the model complexity [Vaswani et al 2017]
- Effectiveness of the model for time series forecasting has been questioned [Zeng et al 2022]

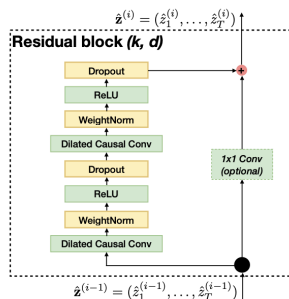
A. Vaswani, et al., "Attention is all you need," Advances in neural information processing systems, 2017

A. Zeng, et al., "Are transformers effective for time series forecasting?," arXiv preprint arXiv:2205.13504, 2022

✓ Temporal convolutional neural network (TCN) -based models



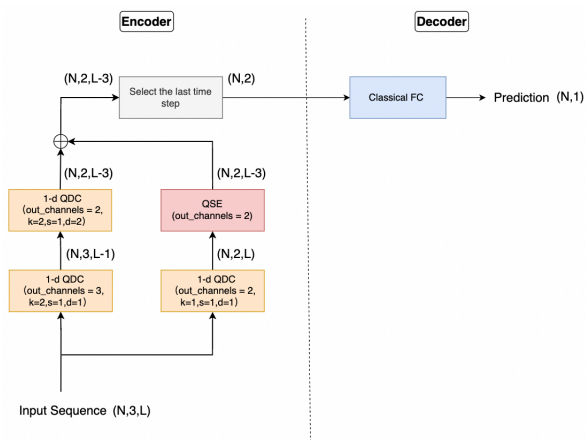
TCN architecture [Bai et al 2018]



- Convolutions enable parallel computation.
- Convolutions reduce the number of parameters due to parameter sharing mechanism.
- Dilated convolution can help increase the receptive field while decreasing the network depth, reducing the chance of vanishing gradients

S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," arXiv preprint arXiv:1803.01271, 2018

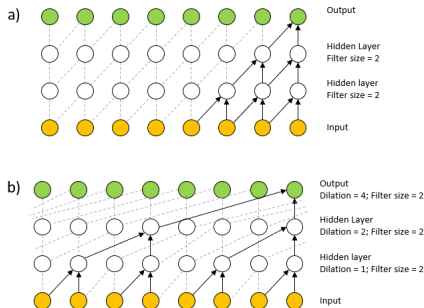
Quantum Temporal convolutional neural network (QTCN)



- Encoder-decoder architecture
- 1D quantum dilated convolution (QDC) + Quantum Squeeze-and-Excitation(QSE)
+ Residual module

1-dimensional Quantum Dilated Convolution

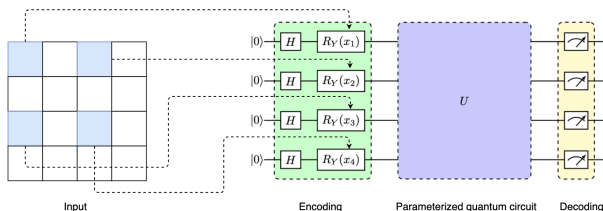
Dilated Convolution Versus Standard Convolution



a) 1D standard convolution b) 1D dilated convolution [Benson et al 2020]

- Dilated convolution provides a larger receptive field while keeping the same neural network depth

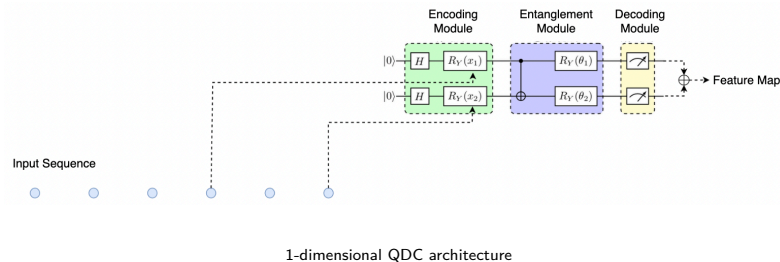
Quantum Dilated Convolution (QDC)



QDC architecture [Chen 2022]

- Combination of dilated convolution and variational quantum circuit.
- QDC is able to capture a larger receptive field without introducing more learnable parameters compared to standard quantum convolution with the same kernel size.
- QDC generally results in a feature map of a smaller size compared to standard quantum convolution for the same set of hyper-parameters (e.g. kernel size, stride, padding), reducing the number of quantum circuit executions.

1-dimensional QDC



● Encoding Module

- Qubit encoding is used, which requires n qubits to encode n input variables;
- Use the Hadamard gate to transform the initial state into a uniform superposition state;
- Encode the input features into $R_Y(x)$ gates, where x is the input feature

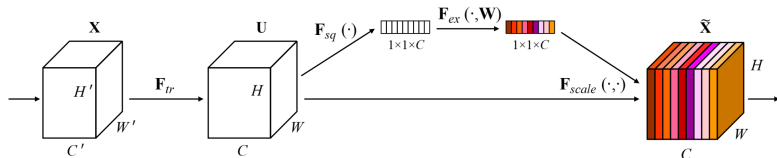
● Entanglement Module: CNOT gate + $R_Y(\theta)$ gate as the unit block

● Decoding Module

- Measure Pauli Z observables in the final quantum state and take the sum of measurement results as a output feature map

Quantum Squeeze-and-Excitation (QSE)

Classical Squeeze-and-Excitation (SE) Module

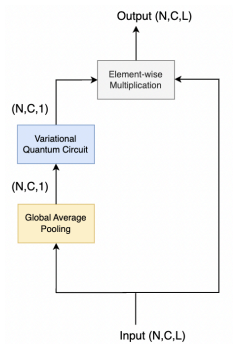


SE Module [Hu et al 2018]

- A channel-wise mechanism that adaptively adjusts the weights of different channels and helps the model focus on more important channels
- *Squeeze module* squeezes each channel to a single numeric value using global average pooling.
- *Excitation module* learns the weights of different channels through non-linear transformation.

J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2018

Quantum Squeeze-and-Excitation (QSE) Module

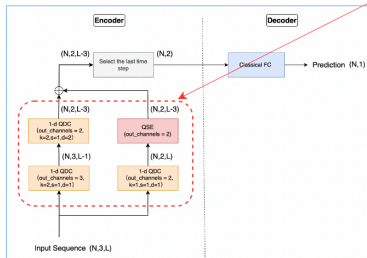
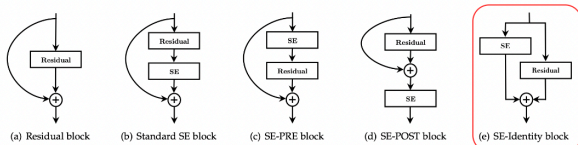


QSE Module

- The excitation module of the QSE block is implemented by a variational quantum circuit with the same input and output sizes.
- Activation functions (e.g. \tanh) are no longer required as the output values of the quantum circuit can range from -1 to 1.

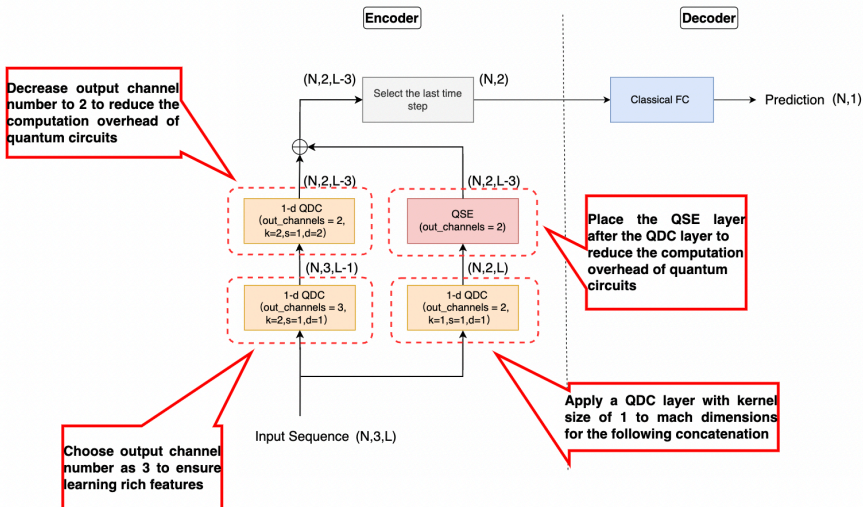
Position of QSE block

Standard SE block and its variants [Hu et al 2018]



J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2018

Detail of QTCN Model Design



Experiment and Result

Experiment Setting

- Experimental Environment
 - PennyLane and PyTorch
 - Local Mac laptop with 8-core CPU
- Model Parameters

Input Size (day)	Qubit Number (per circuit)	Circuit Depth	Quantum Parameters Number	Classical Parameters Number
5	2	1	38	3

- Training Setup

Batch Size	Optimizer	Learning Rate	Epochs
2	Adam	0.005	≤ 120

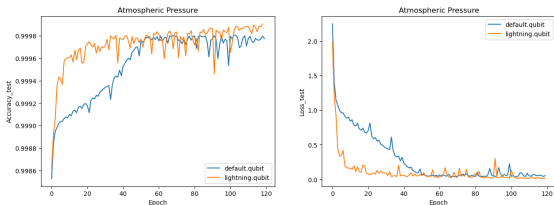
Experiment Result

Device	Atmospheric Pressure	Maximum Temperature	Minimum Temperature	Relative Humidity	Wind Speed
lightning.qubit	99.99%	98.92%	99.26%	98.15%	94.65%
default.qubit	99.98%	99.71%	99.53%	98.55%	92.92%

Test Accuracy

Device	Atmospheric Pressure	Maximum Temperature	Minimum Temperature	Relative Humidity	Wind Speed
lightning.qubit	0.0092	0.0884	0.0190	1.2376	0.1357
default.qubit	0.0390	0.0068	0.0084	0.8244	0.2331

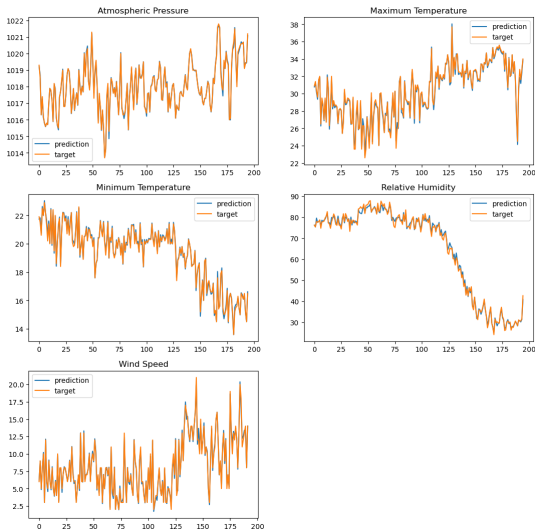
Test Loss



Comparison of Model Convergence

QTCN Model Performance Visualization

Model Performance Based on 'lightning.qubit'




Training Time Optimization

```

t_out = t - self.dilation
out = torch.zeros((bs, t_out))
# quantum circuit executions in for loops
for b in range(bs):
    for i in range(t_out):
        input = torch.Tensor([x[b,i],x[b,i+self.dilation]])
        q_results = torch.sum(self.circuit(input).float())
        out[b,i] = q_results

```

Old code with
for loops



```

t_out = t - self.dilation
batch_input = torch.zeros((bs, t_out, kernel_size)) # (bs, t_out, k)
for b in range(bs):
    for i in range(t_out):
        batch_input[b,i,:] = torch.Tensor([x[b,i],x[b,i+self.dilation]])
batch_input = torch.flatten(batch_input,0,1) # (bs*t_out,k)
batch_input = torch.transpose(batch_input,0,1) # (k,bs*t_out)
# quantum circuit executions with quantum parameter broadcasting
out = circuit(batch_input,self.q_para).float() # (bs*t_out,k)
out = torch.unflatten(out,0,(bs,t_out)) # (bs,t_out,k)
out = torch.sum(out,dim=2) # (bs,t_out)

```

New code with
quantum parameter
broadcasting

Device	Run Time Per Epoch (Old)	Run Time Per Epoch (New)
lightning.qubit	80 sec	41 sec
default.qubit	124 sec	96 sec

Summary and Future Work

- We proposed a novel QML model called quantum temporal convolution network (QTCN) which is a combination of the classical temporal convolution network (TCN) and variational quantum circuits. To our best knowledge, the proposed model is the first quantum version of TCN models.
- We proposed a channel-wise attention mechanism called quantum squeeze-and-excitation which helps the model focus on more important channels and can be applied to general quantum deep learning models.
- The proposed QTCN model achieved promising performances in the task of weather forecasting.
- We would like to extend this work to more time series problems (e.g. demand forecasting, anomaly detection, text classification) in the future.