# Investigating the Implementation of Quantum Algorithms in Weather Forecasting

## Abstract

This study investigates and aims to *implement a quantum-inspired regression algorithm* designed for accurate rainfall prediction, focusing on minimizing associated losses.
It endeavors to implement a quantum algorithm dedicated to the daily prediction of rainfall in millimeters (mm).
Accurate and timely rainfall predictions hold significant importance across various sectors, impacting agriculture, water resource management, and disaster preparedness. Traditional models often struggle to capture the nuanced patterns inherent in daily rainfall data due to their complexity and nonlinearity. Hence this project aims to implement a QLSTM algorithm to tackle such issues.
The primary objective of this study is to develop and implement a dedicated quantum algorithm optimized for precise daily rainfall prediction. This research investigates the feasibility and potential benefits of integrating quantum algorithms into weather forecasting models, aiming to improve accuracy and advance predictive capabilities for better preparedness against weather-related events.
Expected outcomes include a working demonstration of a quantum algorithm that aims to predict rainfall (in mm) and minimize associated losses with runtime.

## Problem Statement

To investigate the potential applications of implementing a predictive algorithm to forecast the weather; chances of rain in particular, using a quantum approach.

The project aims to implement an LSTM training model using quantum algorithms by implementing the LSTM gates using VQCs (Variational Quantum Circuits).

## Literature Review

Quantum computers utilize principles derived from quantum mechanics for computational tasks. In contrast to classical computers that rely on bits, representing information as 0 or 1, quantum computers employ quantum bits (qubits). Qubits can concurrently exist in multiple states, thanks to the concept of superposition. This unique feature enables quantum computers to process extensive amounts of data simultaneously.

### *Why use QLSTM as opposed to vanilla LSTM?*

Quantum computing operates based on principles such as superposition, entanglement, and quantum interference. The property of superposition allows qubits to simultaneously occupy multiple states, leading to a significant boost in computational power that scales exponentially with the increasing number of qubits. Entanglement allows qubits to be correlated with each other, even when separated by vast distances, leading to faster communication and more efficient computation. Quantum interference enables constructive or destructive interference between qubits, enhancing computational capabilities.

Entanglement also allows us to reduce the number of parameters in the quantum circuit. This greatly reduces the noise and generates high expressive power. Here, expressive power refers to the ability to represent certain functions or distributions with a limited number of parameters. QLSTM algorithms are also shown to have a faster learning curve and take less epochs. Stable convergence is achievable in comparison to classical LSTM, particularly, spikes in loss function, observed in classical LSTM is not observed in QLSTM.

Quantum systems could also be used to solve complex problems faster than classical ones by a significant margin, especially in areas like cryptography, optimization, drug discovery, and material science. However, building and maintaining stable qubits that can perform error-free computations, also known as quantum coherence, remains a major challenge in realizing practical, large-scale quantum computers. The quantum circuit model is a mathematical model that uses 'wires' for a quantum system. It results in a polynomial-time quantum computable function. The classical equivalent to a bit is a two-state quantum system called a qubit, or quantum bit. Mathematically, a qubit takes a value in the vector space. One of the fundamental principles of quantum mechanics is that the joint quantum state space of two systems is the tensor product of their individual quantum state spaces.[1]

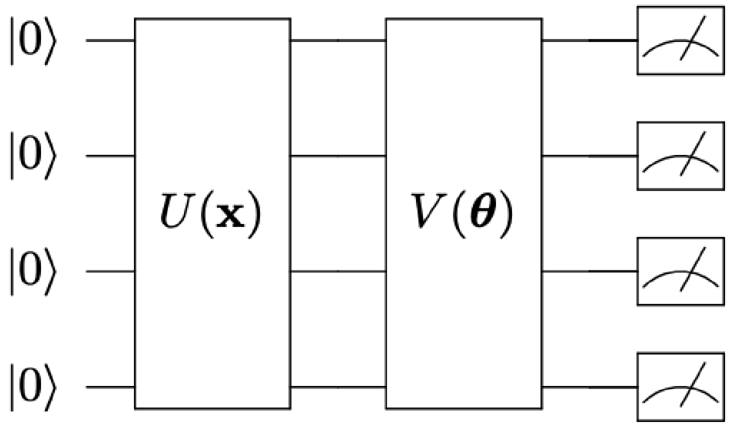
***Our study also dealt with the applications of LSTM and QLSTM in regression analysis.***
The Long Short-Term Memory (LSTM) model was designed to address the vanishing gradients problem faced by RNN. They key insight in the LSTM design was to incorporate nonlinear, data-dependent controls into the RNN cell, which can be trained to ensure that the gradient of the objective function with respect to the state signal (the quantity directly proportional to the parameter updates computed during training by Gradient Descent) does not vanish[2]
By utilizing LSTM networks within a quantum-enhanced framework, the aim is to significantly improve the precision and lead time of rainfall predictions.
Long Short-Term Memory (LSTM) networks, a type of recurrent neural network (RNN), offer promising avenues for enhancing rainfall prediction in conjunction with quantum computing.

LSTMs excel in capturing sequential patterns and long-range dependencies within data, making them suitable for time-series forecasting, such as predicting rainfall. By integrating LSTM models with quantum computing, there's an opportunity to optimize the analysis of historical weather data, enabling more accurate and dynamic rainfall predictions.
The advantage of LSTM lies in its ability to retain and utilize information over extended periods, mimicking the memory function of human cognition. When fused with quantum computing's processing prowess, LSTMs could handle larger datasets more efficiently and extract intricate patterns, crucial for forecasting rainfall variations over time.

***One of our findings was the significant advantage to using QLSTM models as opposed to traditional LSTM ones.*** To showcase the viability of implementing Recurrent Neural Networks (RNNs) using Variational Quantum Circuits (VQC) – a type of quantum circuit where gate parameters are optimized or trained classically – and to illustrate the potential quantum advantages within this framework.

Specifically, we apply VQCs to implement Long Short-Term Memory (LSTM), a well-known RNN variant capable of capturing long temporal dependencies. We term our Quantum Machine Learning (QML) architecture as Quantum LSTM, or QLSTM for brevity. In this proposed framework, we adopt a hybrid quantum-classical approach suitable for Noisy Intermediate-Scale Quantum (NISQ) devices. This approach involves iterative optimization, harnessing the enhanced expressive power facilitated by quantum entanglement. Through numerical simulations, we demonstrate that QLSTM exhibits faster learning, requiring fewer epochs compared to classical LSTM with a similar number of network parameters. In addition, the convergence of our QLSTM is more stable than its classical counterpart; specifically, no peculiar spikes that are typical in LSTM's loss functions is observed with QLSTM.[3]

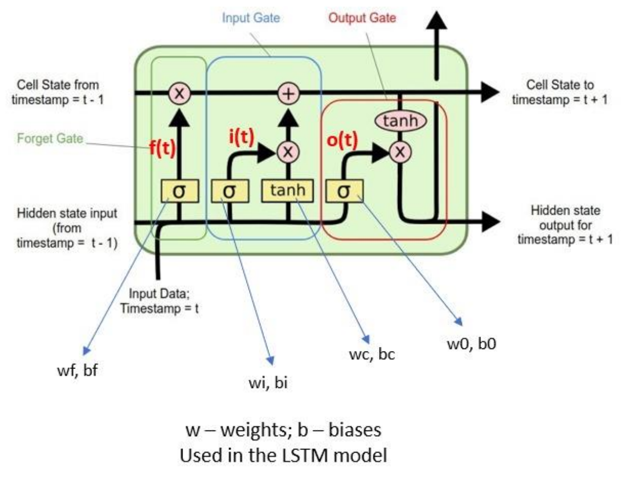
Additionally, we discovered the ***gravity of VQC's in our implementation.***
These circuits, composed of a sequence of quantum gates, serve as computational engines, where the manipulation of gate parameters, allows for the exploration of potential solutions to a given problem. The figure below illustrates the general VQC architecture.



In this context, the U(x) block serves as the component responsible for state preparation, translating classical data x into the quantum state of the circuit. Notably, this block is not subject to optimization. On the other hand, the V(θ) block represents the variational aspect, incorporating learnable parameters θ that will undergo optimization through gradient methods. Finally, we perform measurements on a subset (or all) of the qubits to obtain a (classical) bit string, such as 0100.

Prior findings have indicated the resilience of such circuits against quantum noise, rendering them suitable for Noisy Intermediate-Scale Quantum (NISQ) devices. Variational Quantum Circuits (VQCs) have demonstrated success in diverse applications, including function approximation, classification, generative modeling, deep reinforcement learning, and transfer learning.

Moreover, it has been highlighted that VQCs exhibit greater expressiveness compared to classical neural networks, potentially outperforming the latter. In this context, expressiveness refers to the capability to represent specific functions or distributions with a constrained number of parameters. Notably, artificial neural networks (ANN) are recognized as universal approximators, suggesting that even a neural network with a single hidden layer can theoretically approximate any computable function. As we will see below, using VQCs as the building blocks of quantum LSTM enables faster learning.[4] Modeling rainfall is challenging because of its large variability in space and time, and its highly skewed distribution. Numerical weather prediction (NWP) models must be simulated on discretized grids with finite resolution. Although important especially for the generation of rainfall, small-scale processes can therefore not be resolved explicitly and must be parameterized, that is, included as empirical functions of the resolved variables.[5]


Input Gate
Output Gate
Cell State from timestamp = t - 1
Cell State to timestamp = t + 1
Forget Gate
f(t)
i(t)
o(t)
tanh
σ
Hidden state input (from timestamp = t - 1)
Hidden state output for timestamp = t + 1
Input Data; Timestamp = t
wf, bf
wi, bi
wc, bc
w0, b0
w – weights; b – biases
Used in the LSTM model

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$$

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

$$o_t = \sigma\left(W_o \, [h_{t-1}, x_t] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

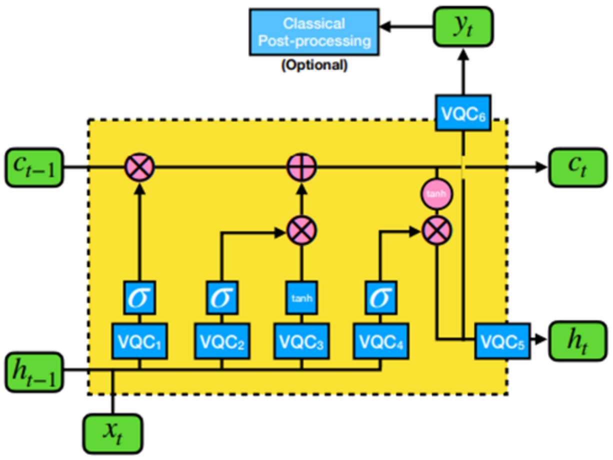
LSTM Model Working (L), Calculations (R)

***A part of our study includes observing the loss quantified by our implementation***. In our implementation we have employed the MSE loss function typically used in regression models.
A loss function is a crucial component in machine learning that quantifies the success in the prediction model in comparison with the verified results using the training and testdata. It measures the inconsistency between predicted values and actual values, providing a single scalar value that represents the model's performance.

***The goal during training is to minimize this loss function***, thereby improving the model's predictive accuracy. There are various types of loss functions, each suited for different types of machine learning tasks (classification, regression, etc.), and they play a vital role in guiding the model's learning process. For instance, in regression problems, mean squared error ***(MSE)*** is a commonly used loss function, while for classification tasks, cross-entropy loss is often employed.

## Methodology

We've deployed a ***quantum algorithm based on QLSTM*** to forecast rainfall volume, utilizing a dataset for training our model.
In addressing our rainfall prediction task, we've adopted a multi-faceted approach, utilizing ***regression*** as a primary method due to its aptitude in forecasting continuous numerical values like rainfall. This technique involves diverse factors such as temperature, humidity, and more to predict outcomes accurately, aligning well with our problem statement. Additionally, our exploration has extended to deploying ***VQC*** (Variational Quantum Circuits) and QLSTM (Quantum Long Short-Term Memory) algorithms to attain optimal outputs.

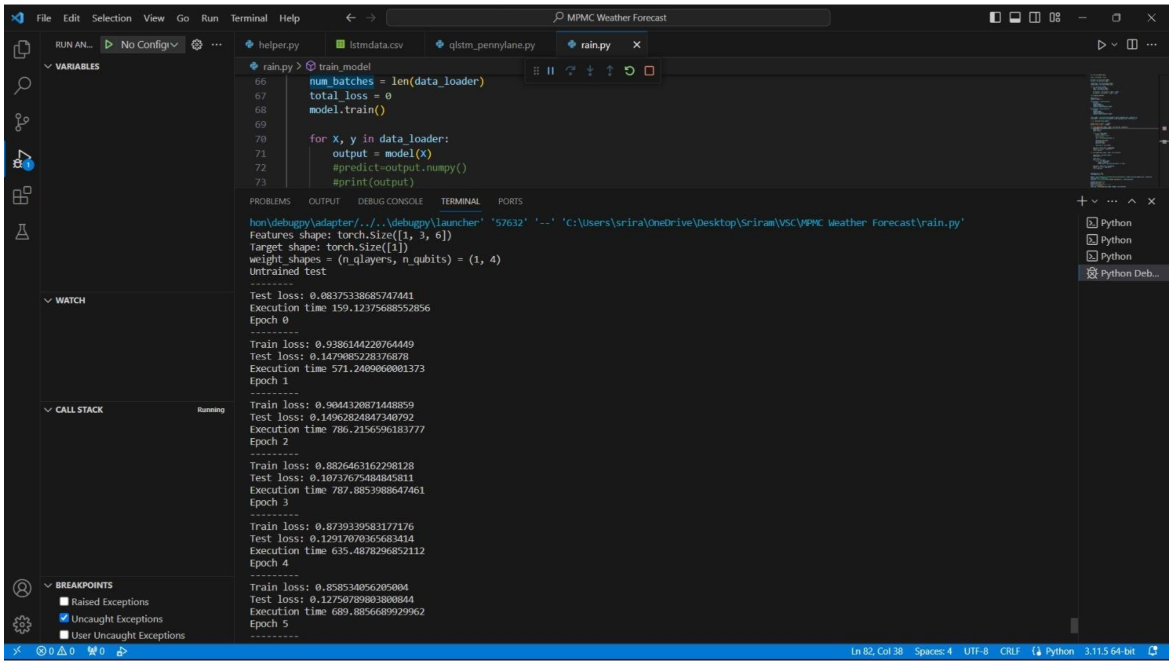


Implementation of VQCs to create QLSTM Model

Throughout our experimentation, the key focus has been on reducing the disparity between actual rainfall values and our model's predictions. This endeavor involves iterative training across numerous epochs, fine-tuning our algorithms to enhance predictive precision. However, a notable limitation emerges from our reliance on conventional computers to execute the code, as opposed to the potential advantages offered by quantum computation. This limitation somewhat hampers our ability to minimize errors effectively and observe substantial improvements.

Despite these constraints, our determination remains unwavering. We aspire to continue running and refining our code, exploring avenues for improvement within the confines of classical computing resources. Our study acts as a preliminary exploration into the application of various algorithms for rainfall prediction, acknowledging the potential enhancements that a shift to quantum computing could yield. While the full realization of error minimization might be impeded by current computational constraints, our commitment to advancing these methodologies in weather forecasting remains steadfast. Through continued investigation and potential integration of quantum computational resources, we aim to further propel the accuracy and reliability of rainfall predictions in the future.

## Results

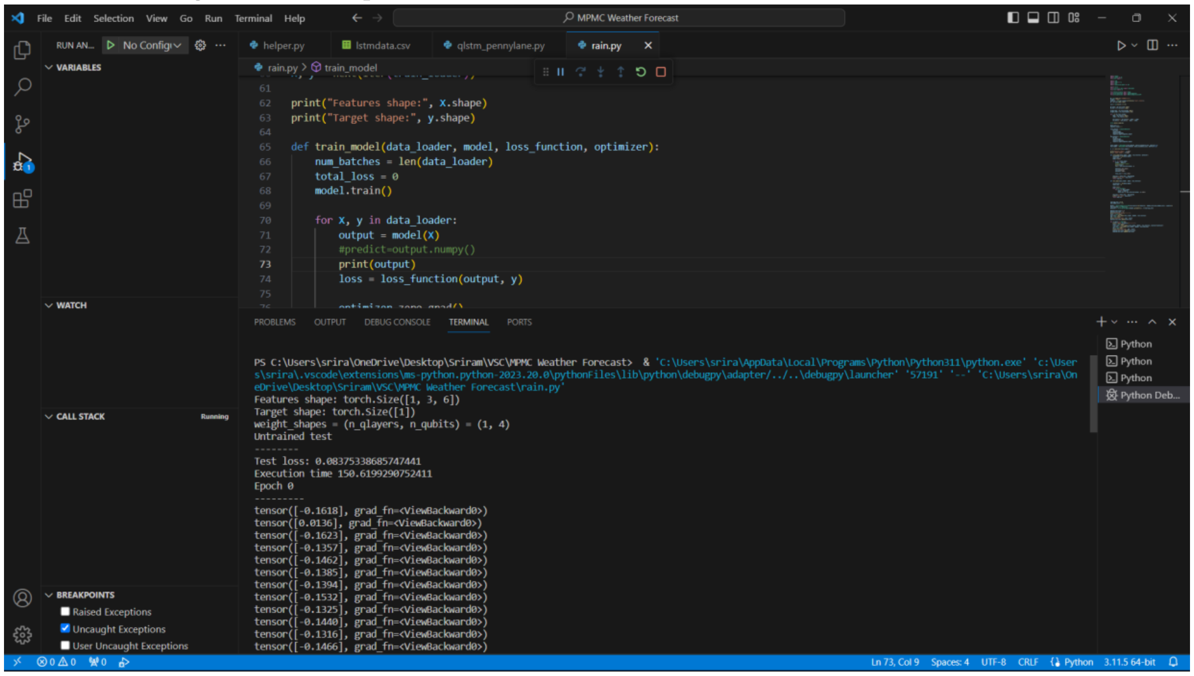


```python
    num_batches = len(data_loader)
    total_loss = 0
    model.train()

    for X, y in data_loader:
        output = model(X)
        #predict=output.numpy()
        #print(output)
```

```
hon\debugpy\adapter/../..\debugpy\launcher' '57632' '--' 'C:\Users\srira\OneDrive\Desktop\Sriram\VSC\MPMC Weather Forecast\rain.py'
Features shape: torch.Size([1, 3, 6])
Target shape: torch.Size([1])
weight_shapes = (n_qlayers, n_qubits) = (1, 4)
Untrained test
--------
Test loss: 0.08375338685747441
Execution time 159.12375688552856
Epoch 0
---------
Train loss: 0.9386144220764449
Test loss: 0.1479085228376878
Execution time 571.2409060001373
Epoch 1
---------
Train loss: 0.9044320871448859
Test loss: 0.14962824847340792
Execution time 786.2156596183777
Epoch 2
---------
Train loss: 0.8826463162298128
Test loss: 0.10737675484845811
Execution time 787.8853988647461
Epoch 3
---------
Train loss: 0.8739339583177176
Test loss: 0.12917070365683414
Execution time 635.4878296852112
Epoch 4
---------
Train loss: 0.858534056205004
Test loss: 0.12750789803800844
Execution time 689.8856689929962
Epoch 5
---------
```

Result with training tensors for Epoch 0:



```python

print("Features shape:", X.shape)
print("Target shape:", y.shape)

def train_model(data_loader, model, loss_function, optimizer):
    num_batches = len(data_loader)
    total_loss = 0
    model.train()

    for X, y in data_loader:
        output = model(X)
        #predict=output.numpy()
        print(output)
        loss = loss_function(output, y)

```

```
PS C:\Users\srira\OneDrive\Desktop\Sriram\VSC\MPMC Weather Forecast>  & 'C:\Users\srira\AppData\Local\Programs\Python\Python311\python.exe' 'c:\User
s\srira\.vscode\extensions\ms-python.python-2023.20.0\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '57191' '--' 'C:\Users\srira\On
eDrive\Desktop\Sriram\VSC\MPMC Weather Forecast\rain.py'
Features shape: torch.Size([1, 3, 6])
Target shape: torch.Size([1])
weight_shapes = (n_qlayers, n_qubits) = (1, 4)
Untrained test
--------
Test loss: 0.08375338685747441
Execution time 150.6199290752411
Epoch 0
---------
tensor([-0.1618], grad_fn=<ViewBackward0>)
tensor([0.0136], grad_fn=<ViewBackward0>)
tensor([-0.1623], grad_fn=<ViewBackward0>)
tensor([-0.1357], grad_fn=<ViewBackward0>)
tensor([-0.1462], grad_fn=<ViewBackward0>)
tensor([-0.1385], grad_fn=<ViewBackward0>)
tensor([-0.1394], grad_fn=<ViewBackward0>)
tensor([-0.1532], grad_fn=<ViewBackward0>)
tensor([-0.1325], grad_fn=<ViewBackward0>)
tensor([-0.1440], grad_fn=<ViewBackward0>)
tensor([-0.1316], grad_fn=<ViewBackward0>)
tensor([-0.1466], grad_fn=<ViewBackward0>)
```

## Code

The code files along with the dataset used to train the model have been attached in the GitHub repository referenced below.
https://github.com/Reflex-Angle/QLSTM-Prediction

## Bibliography and References

1.Shor, P. W. (1998). Quantum computing. *Documenta Mathematica*, *1*(1000), 1.

2.Sherstinsky, A. (2020). Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. *Physica D: Nonlinear Phenomena*, *404*, 132306.

3.Chen, S. Y. C., Yoo, S., & Fang, Y. L. L. (2022, May). Quantum long short-term memory. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 8622-8626). IEEE.

4.Liu, M., Dong, G., Felker, K. G., Otten, M., Balaprakash, P., Tang, W., & Alexeev, Y. (2022). *Exploration of quantum machine learning and ai accelerators for fusion science* (No. ANL/CPS-21/3). Argonne National Lab.(ANL), Argonne, IL (United States).

5.Hess, P., & Boers, N. (2022). Deep learning for improving numerical weather prediction of heavy rainfall. *Journal of Advances in Modeling Earth Systems*, *14*(3), e2021MS002765.

## Team Members

Sriram B Swami - 22BCE5103
AO Navin Kumar - 22BCE1020
Meghna Varma - 22BCE5085
Manasa Ganesh - 22BAI1262

## Additional Reference Materials

1.An Introduction to Long Short Term Memory: https://medium.com/analytics-vidhya/introduction-to-long-short-term-memory-lstm-a8052cd0d4cd

2. https://github.com/rdisipio/qlstm

3.VQC documentation:
https://qiskit.org/ecosystem/machine-learning/stubs/qiskit_machine_learning.algorithms.VQC.html

4.QMLdocumentation:
https://docs.pennylane.ai/en/stable/code/qml.html