Imperial College London

Department of Earth Science and Engineering

MSc in Applied Computational Science and Engineering

Independent Research Project
Final Report

Multi-variate Regression For Time SeriesWaste-To-Energy Anaerobic Digestion DataOn Quantum Architecture

by

Lakshit Dabas

lakshit.dabas20@imperial.ac.uk
GitHub login: ldabas

Supervisors:

Dr. Po-Heng Lee
Dr. Lukas Mosser

August 2021

GitHub repository: https://github.com/ldabas/QCNNForecasting

Abstract

Forecasting biogas produced from anaerobic digestion offers waste-management facilities a unique opportunity to reduce waste-disposal costs and generate new revenue streams. Due to the complex nature of anaerobic digestion, it is immensely difficult to obtain a mathematical model for forecasting biogas production, and the utilisation of machine learning is therefore essential. By rigorous preprocessing of waste-management data, modified convolutional neural networks(CNN) with short-term memory were developed on both quantum and classical architectures. Data was made stationary through time-series analysis, and a first-of-its-kind quantum CNN with short-term memory was developed. Domain knowledge for forecasting biogas production was obtained by collaborating with waste-management facility stakeholders in Hong Kong. Trend and seasonality were removed from input features with exponential smoothing, and quantum encodings for representing classical data on quantum architectures were developed. The industry-leading Neuro-Fuzzy Inference System (ANFIS) model proposed by Ramachandran et al. (2019) offered RMSE and MAE values of 3.986, 0.673 for biogas production forecasts. These values were beaten by models on both architectures, with RMSE and MAE values of 0.55 and 0.431, respectively. Better accuracies were attributed to the application of robust preprocessing frameworks and the utilisation of CNN architectures for more excellent feature extraction.

Keywords– Time-Series Forecasting, Quantum Machine Learning, Convolutional Neural Networks

1 Introduction

Anaerobic digestion is utilized in waste management plants worldwide to produce methane by breaking down organic waste. In addition, due to the combustible nature of methane, it can be used to generate heat and electrical energy. Anaerobic digestion holds great potential for developing circular economies by utilising waste-management facilities to produce bioenergy. The rate of adoption of the utilization of anaerobic digestion to produce bio-energy is expected to grow steadily over time due to the worldwide consensus on the need for better waste management. Large economies, such as the USA, EU, and China, are transitioning towards circular economies to reduce waste disposal costs and generate new opportunities for increasing revenue(Labatut & Pronto 2018).

The rise of better machine learning algorithms and the growth of the quantum computing field provides a unique opportunity to deliver an accurate biogas forecasting model for waste management facilities(Phillipson 2020).

1.1 Background

1.1.1 Anaerobic Digestion

Anaerobic digestion is an incredibly complicated process at the heart of managing waste to produce energy. It is a series of complex biological operations such as acidogenesis, hydrolysis, and methanogenesis, in which organic matter is broken down by anaerobic micro-organisms(Wang et al. 2018, 2020). The presence of numerous convoluted processes within anaerobic digestion present a need to model the problem. However, since there are many variables with innumerable interlinking operations throughout several tanks, it is difficult to obtain a mathematical equation representing the process in large waste- management facilities. In order to run predictions, the utilization of machine learning is therefore essential(Lauwers et al. 2013).

1.1.2 Convolutional Neural Networks and Memory-Based Machine Learning

Convolutional Neural Networks(CNN) are a subset of Artificial Neural Networks. They are made up of a series of convolutional and pooling layers. Fully connected layers and a single non-linearity layer are present towards the end of a network to collect all weights and biases and make a prediction. CNNs perform convolutional and

downsampling operations to make predictions on a given set of data(O'Shea & Nash 2015). CNNs can be used to create highly accurate and robust models. They are also very computationally efficient since they do not require a vast amount of resources to train. Parameter sharing is performed by using convolution and pooling operations, thus allowing the training of CNNs on virtually any device, reducing the difficulty of adopting them to various system architectures(Albawi et al. 2017).

Traditionally, the input of a CNN is a 2D matrix. However, they can be used for regression by modifying them to accept 1-dimensional input(Xie et al. 2015, Chen & He 2018) which is a small window of the data that is related to its neighbours.

While Recursive Neural Networks(RNN) may be the more obvious choice in dealing with time-series data, RNNs frequently run into the issue of gradient vanishing during modeling. Long-Short Term Memory(LSTM) models may be used as an alternative, but CNNs generally perform as well as RNNs and LSTMs for Time-Series forecasting(Chen & He 2018) and require less training time due to the presence of fewer parameters.

1.1.3 Quantum Machine Learning

Since machine learning and artificial intelligence has helped find patterns in complex data of massive dimensional space(LeCun et al. 2015, Goodfellow et al. 2016, Wang & Raj 2017), attempts are now being made to combine machine learning algorithms with the speed-ups offered by quantum computers to solve problems with more considerable complexities(Biamonte et al. 2017, Schuld et al. 2015).

New and emerging quantum machine learning algorithms have made it possible to utilize computationally efficient training methods to make predictions on complex datasets. Quantum Neural Networks (QNNs) can be considered analogues for Artificial Neural Networks found in classical computation. They aim to unify quantum information theory with neural networks to generate vastly more accurate models in less training time.

QNNs show promise of accelerating growth in machine learning like the invention of the AlexNet model did for deep learning(Nielsen & Chuang 2010, Pattanayak & Pattanayak 2021). By utilizing the principles of entanglement and superposition (Nielsen & Chuang 2010), they can perform parallel-processed neural computing simultaneously(Pattanayak & Pattanayak 2021). The QNN research field shows promise in eventually decomposing big-data problems with the aid of quantum machine learning algorithms(Biamonte et al. 2017, Hansen et al. 2013, Tchagang & Valdés 2019, Wu et al. 2018).

Furthermore, with the development of Quantum Convolutional Neural Networks(QCNNs)(Cong et al. 2019) for image detection, it may now be possible to use them to forecast multi-variate time series data. This is due CNNs being able perform highly accurate feature extractions from multidimensional data. An image of resolution 5×5 is equivalent to a time-series input with five features and a five day memory, with the primary difference being the utlisiation of one dimensional convolutional and pooling layers for time series, as opposed to two dimensional operations being applied on an image.

1.2 Motivation

In order to forecast the production of biogas within extensive waste-management facilities, a model must be built that can utilise multiple features to forecast future trends in biogas prediction for waste-management facilities.

Models have been made in the past to predict the controls in waste management plants. Neural Networks, genetic algorithms, and fuzzy logic models have all been implemented(Ramachandran et al. 2019). Neuro-Fuzzy Inference System (ANFIS) and neural networks, among others, were evaluated by Kusiak & Wei (2014), and it was observed that ANFIS provided the most accurate forecasts(Ramachandran et al. 2019).

No attempts thus far have been made to model biogas production from anaerobic digestion with convolutional neural networks, RNNs, or LSTMs. Additionally, prior models predicting biogas production have not treated waste-management data as a time-series(Ramachandran et al. 2019). Since past models have not made the data stationary to remove the trend and seasonal bias present within time-series data, these models have suffered in analysing trends of biogas-production within waste-management facilities, leading to inaccurate forecasts.

While novel quantum RNN(QRNN) and quantum LSTM(QLSTM) models have been developed for forecasting uni-variate time series data (Bausch 2020, Chen et al. 2020), no QCNN models have been designed for forecasting either uni-variate or multi-variate time series data.

This project aims to design a model to forecast biogas production during anaerobic digestion in waste-management facilities by developing convolutional neural networks on classical and quantum architectures and investigate the performance and challenges of forecasting time-series data, particularly on quantum architectures.

This requires the successful completion of the following objectives:

- Rigorous preprocessing necessary for time series data and the data transformations necessary to provide input to Quantum CNN models.
- Development of classical and quantum CNNs for performing regression of discrete input features.
- Development of a modified CNN model that can simulate a short-term memory for input features.
- Development of a novel QCNN model that accepts a similar 'time-window' of data to simulate a short-term memory.

The structure of this project is as follow: Section 2 briefly details the code metadata for the software. Section 3 describes the methodology by detailing the structure of the software built, the nature of the dataset, evaluation metrics considered, the data preprocessing steps needed to model biogas production, and the architecture of the developed models. Section 4 presents the performance comparison of the models. Finally, Section 5 summarizes the conclusions from this work.

2 Code Metadata

The software for this project developed entirely in Python 3.8. The development was done in an `Ubuntu 18.04.5 LTS` subsystem on Google Colab. Due to the conflicting mathematical operations performed by Tensorflow and Tensorflow Quantum, two sanitised classical and quantum development environment were utilised. Tensorflow and Keras(TensorFlow 2021) were the primary libraries used for the development of classical models. Tensorflow Quantum and Cirq(TensorFlow 2021) were used for developing quantum machine learning models. Pandas(pandas development team 2020), Numpy(Harris et al. 2020), and Scikit-Learn(Pedregosa et al. 2011) were the primary libraries used for data preprocessing. Seaborn(Waskom 2021) and Matplotlib(Hunter 2007) were used for generating data visualisations.

The entire code base was developed on Google Colab. The execution and training of models does not require the installation of any packages and can be done entirely in Google Colab. Data files used can be found within `\data`, and the documentation for the code base can be found in `\docs`. All models developed during this investigatory project can be found within the `\models` subfolder while the performance metrics for all developed models can be found in the `\results` subfolder.

The entire codebase is contained within `.ipynb` notebooks detailing the data investigation, model development, and evaluation of models.

3 Methodology

3.1 Software Description

A combination of workflow pipeline and checkpoint design pattern was utilised for model development. Due to the delicate structure of QCNNs, resilience and fault tolerance were prioritised. Workflow pipeline design pattern was used to ensure that the models were reproducible. In this pattern, each step in the machine learning workflow, preprocessing, model training, and validation, was isolated and containerized into three distinct submodules. The checkpoint was used because of this pattern's unique ability to reduce overfitting by saving an instances of the model with the lowest validation loss during training. The preprocessing layer removed features with high sparsity and low correlation. Data imputation by replacing missing values with Random Forest regressors was applied to features with data density above the accepted tolerance of 60%. For models that utilised a short-term memory for forecasting, training data was made stationary with exponential smoothing methods by using the StatsModel API(Seabold & Perktold 2010). Finally, for models built on a quantum architecture, dense angle encoding, as proposed by LaRose & Coyle (2020), was applied with Cirq.

Both classical and quantum models were built using Tensorflow and Tensorflow Quantum in Python within the model training submodule. The checkpoint design pattern was utilised to prioritise fault tolerance and resilience for all models. After each epoch of training, the validation loss for the model was measured. If the validation loss was lower than that of the previous epoch, a snapshot of the model's state was saved to be able to resume training in the future.A so-called patience time, the time before the ML-training process is aborted due to no improvement in validation loss, was set to 10 epochs. This allows sufficient time for the algorithm to potentially escape local minima in the optimization process The data was continuously validated to maintain constant dimensional sizes for the dataset and separation among the training and testing data.

Finally, the validation submodule initialised the models saved after model training to obtain key metrics detailed in the Methodology section. These metrics, along with the forecasts generated, were saved within the `results` subfolder.

3.2 Data

Data describing the chemical compositions of waste management facilities in Hong Kong and the corresponding biogas produced was provided by stakeholders of Hong Kong's waste management facilities. The data spanned 28 months(851 days) and consisted of 27 features detailing the daily chemical compositions of tanks within waste-management facilities. The data was split into 21 months of training data, four months of validation data, and four months of test data. Due to the unstructured nature of the dataset, preprocessing steps were essential for modelling. By collaborating with the stakeholders of the waste management plant, nine key input features were identified from the dataset. These features were the Q_{in}, $DS\%$, and $VS\%$of Thickened Primary Sludge(TPS), Thickened Waste Activated Sludge(TWAS), and the Q_{out}, $DS\%$, $VS\%$ of Digested Sludge(DS).

3.3 Preprocessing

The dataset sparsity was found to be approximately 55%. Therefore, features with sparsity greater than 35% were removed to ensure that minimal noise was introduced into the input features during preprocessing. Since 6 of the key 9 input features identified with domain knowledge had greater than 60%data sparsity, they were dropped. Thus a black-box approach for model development was taken. Features with low correlation(< 0.02) to the label(biogas production) were also removed to mitigate the risk of overfitting the model. Removing features with low correlation and high data sparsity shrank the dimension space from 27 to 5 viable input features.

Features with low data sparsity could not be concatenated by removing missing rows due to the time-series nature of the dataset. In addition, removing days with missing values would compromise the seasonality and trend of the data, leading to larger errors. Therefore, features with data sparsity less than 60% required their missing data to be imputed.

The NIPALS algorithm and Random Forest Regression were explored as imputation methods. The NIPALS (Nonlinear Iterative Partial Least Squares) algorithm is a PCA method used to find the first few principal components with decomposition. Furthermore, the NIPALS algorithm can be modified to accommodate missing values by treating a dataset as an $N \times N$ matrix X and decomposing it into it's (pseudo-) singular-values (eig), set of components (t), and feature loadings (p)(Nelson et al. 1996). The matrix can then be reformed according to the following equation:

$$\hat{X} = t * diag(eig) * t(p) \tag{1}$$

The missing values from X were then approximated from this matrix. The NIPALS algorithm led to a percentage change in the mean of the dataset of $\delta = 11\%$.

As an alternative approach, Random Forest regressors were utilised to find missing values within the dataset. The regressors led to a percentage change in the mean of the dataset of $\delta = 6\%$, offering lower noise when imputing data and were thus utilised. Biogas Production was treated as the sole output of the model. $TPS - Q_1$, TWAS-Q_{in}, Digested Sludge Q_{out}, PS-Q_1, and the $DS\%$ of Digested Sludge after dewatering were the five viable features that were considered for model training.

3.3.1 Time Series Analysis

To identify the dimensional length of short-term memory time-window CNN models, the autocorrelation period of the input features was investigated with the Statsmodels API. Data was decomposed into its seasonality,

trend, and residuals. It was found that the features at time $= t$ influence the values of $t = t + 1$ and $t = t + 2$; therefore the autocorrelation period was found to be equal to 3. Thus data within a 'time-window' of 3 days was provided to forecasting models as a short-term memory for input features.

Due to the time-series nature of the dataset, the input data needed to be made stationary. A dataset is considered stationary if the mean and variance of the dataset are independent of time. Seasonality present within a time-series can lead to a variance that is a function of time, while trends can cause the mean of the dataset to fluctuate over successive time intervals. To investigate whether the input features were stationary, the Augmented Dicky-Fuller Test(ADF) was used as a statistical experiment. The ADF proposes a null hypothesis that the time-series under observation possesses a unit root. The unit root's presence makes the data a function of time and thus non-stationary, while rejection of the null hypothesis verifies that the data is indeed stationary. While the label and two input features were found to be stationary, the remaining three of the five input features were found to be non-stationary. Due to the contrasting nature of the input features, the data needed to be transformed to obtain a stationary dataset.

This was attempted via differencing, which transforms a time series dataset by subtracting the previous observation from the current observation. After each successive differencing step, the ADF test is performed to verify whether the time-series is stationary. This process is conducted iteratively until the dataset becomes stationary. Due to the the data being partially stationary, it was observed that repeated differencing of successive rows had a large impact on the output, leading to non-positive values for biogas emission. Since differencing was found to increase the risk of compromising the integrity of the partially stationary dataset, exponential smoothing transforms were used instead. Exponential smoothing assigns weights to observations. Older observations are weighted less than recent observations. Exponential smoothing is equivalent to a low-pass filter being applied to the dataset to remove high-frequency noise. The raw data sequence is represented by x_t at $t = 0$. The output, s_t of the smoothing algorithm, is an estimate of the next value of x when the sequence of observations begins at time $t = 0$.

$$s_t = \alpha x_t + (1 - \alpha) s_{t-1} \quad t > 0 \tag{2}$$

The smoothing parameter α determines the weight assigned to the most recent observation. For the training dataset, α was chosen as 0.1

3.3.2 Quantum Encoding

For QCNN models, input data needed to be further transformed by encoding it as a quantum circuit that can be measured to obtain an expectation value. A quantum encoding loads each feature $x \in X$ into a quantum state which in turn is added to a quantum circuit. This quantum circuit is then used as the input for neural networks built on quantum architectures. A set of input features X is encoded to n-qubit quantum states D_n where D_n is the dimensional space of the inputs. For the given data set, $D_n = 5$, any given feature vector $x = [x_1, ..., x_N]^T \in X^N$ can be encoded into a quantum circuit with 'dense angle encoding' as described by LaRose & Coyle (2020) with the following equation,

$$|x\rangle = \bigotimes_{i=1}^{N} cos(x_i) |0\rangle + sin(x_i) |1\rangle \tag{3}$$

By encoding the input features as rotations along the x and y-axis, n qubits with a quantum circuit of constant depth can be utilised for model training. The state preparation unitary operator $S_{x_j} = \bigotimes_{i=1}^{N} U_i$ is given by,

$$U_i := \begin{bmatrix} cos(x_j^{(i)}) & -sin(x_j^{(i)}) \\ sin(x_j^{(i)}) & cos(x_j^{(i)}) \end{bmatrix} \tag{4}$$

3.4 Validation

Objective performance measures were needed to compare the forecasting model performance on the test data fairly. Data was split into Training(80%), Validation(10%), and Test(10%) set. Mean Average Percentage Error(MAPE), Accuracy, Root Mean Square Error (RMSE), and Mean Average Error(MAE) are the primary metrics considered. Metrics were measured on both validation and training sets to obtain accurate in-sample and out-of-sample model performances. The number of epochs to achieve the minimum validation loss was

also measured to investigate the learning speeds of classical and quantum models. RMSE% and MAE% error metrics were also recorded to investigate accuracies between both architectures. This was done since quantum information encoding adds a transform unique to quantum architectures, thus changing the scale of raw RMSE and MAE values. RMSE was chosen due to the equal weightage it gives to predictions towards the beginning and end of the dataset. Using RMSE alongside MAPE prevented favouring models that lower the error artificially by underestimating it but lead to larger RMSE errors. Additional details about the calculation of the metrics can be found in Appendix A.

3.5 Model Development

Since no current QCNN models exist for forecasting time-series data, an iterative approach was best suited for model development. Therefore, four primary models were developed to forecast biogas emission and compare the performance to quantum convolutional neural networks. The first model(M1) designed was a classical deep convolutional neural network model. It was developed to investigate accuracies obtained by simple regression on individual data points without short-term memory being provided to the model. This was done in order to investigate whether the model's predictions were influenced by the addition of historical data to input features. The second model(M2) developed was a classical deep convolutional neural network model that accepted input features of length n where n is the auto-correlation period observed. By providing input features of length n, short-term memory was simulated for the CNN model. This is the primary classical model for forecasting biogas productions. The penultimate model developed was a quantum deep convolutional neural network model(QM1). It was developed primarily as a proof-of-work for designing a QCNN model(QM2) that accepted historical data as input. The development of model M1 also allowed for more accurate benchmarking for the discrete data QCNN model(M1Q). Finally, a multi-variate time-series forecasting QCNN model with the addition of time-window was developed(M2Q), and its performance analysed. These models are discussed in detail down below.

3.5.1 Classical Models

3.5.1.1 Discrete Data CNN Model

The first model developed was a CNN model that treated each datapoint within the dataset as discrete. No Short-Term Memory was provided to the model. Each row of data was input independently of any historical data. In the proposed architecture for this model, a deep CNN is applied to multi-variate time series regression. Joint feature learning is performed on the label time series. The output weights from the CNN are passed into a multi-layer perceptron(MLP) for biogas production estimation. Thus, five input features, each of length one serve as inputs to two convolution layers. After the final convolution layer, feature maps are concatenated into a flattened vector as the MLP input for biogas production estimation. A 'LeakyReLu' activation function is used for faster training times and to avoid errors associated with zero gradients during gradient descent. Model training is performed by estimating CNN parameters with back propagation using the Adam optimizer to minimise the model's mean square error.

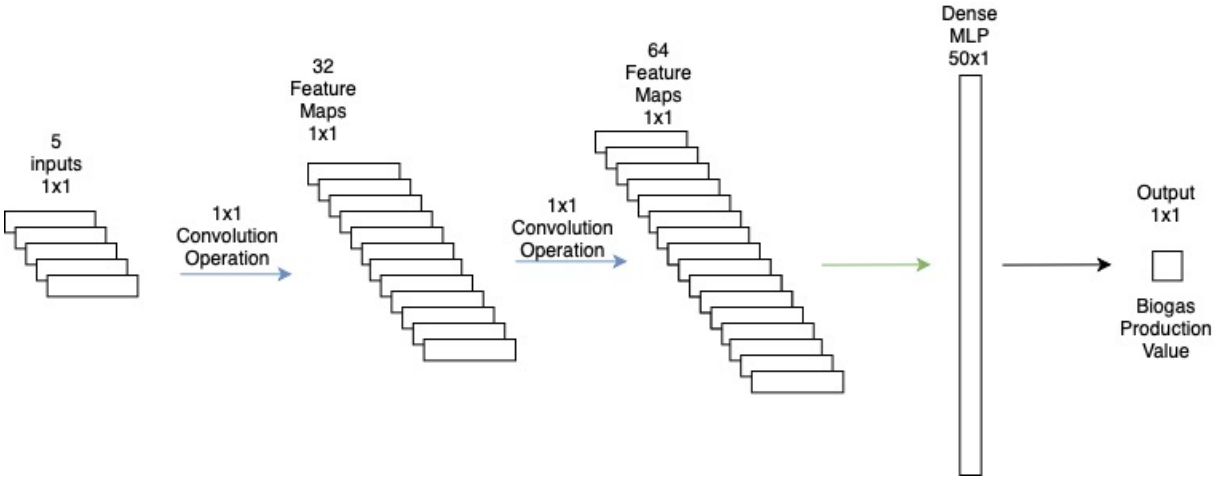

Figure 1: Model Architecture for Discrete-Data CNN Model.

Feature maps obtained as outputs from prior layers are convolved with convolution kernels. The output obtained from the convolution operators,the bias term, and the feature map for successive layers is evaluated with 'LeakyReLu' activation function. The evaluation of the output feature map for convolution layers is given by,

$$x_j^l = lrelu(z_j^l) \tag{5}$$

$$z_j^l = \sum_i x_i^{l-1} \odot k_{ij}^l + b_j^l \tag{6}$$

Where x_i^{l-1} and x_j^l are the convolution filter input and output, respectively. $\odot$ denotes the convolution operator, k represents the convolutional kernel, b represents the bias term, $lrelu()$ denotes the 'LeakyReLu' function and $z_j^l$is the input of the LeakyReLu.

3.5.1.2 Short-Term Memory CNN Model

Since the auto-correlation period of the dataset was found to be three days, a CNN model that took input features of length three days was developed. A sliding window of length equal to the auto-correlation period was then used to segment the time-series into smaller subsets of data. Data points fed into the CNN model were two-dimensional matrices with five rows representing distinct input features and three columns containing input feature values from the two previous days and the current day.

A similar structure as the discrete data CNN model was followed. A deep CNN was utilised for multi-variate time series forecasting. Joint feature learning is performed on the label time series, and the output weights from the CNN are passed into a multi-layer perceptron(MLP) for biogas production estimation. Thus, data

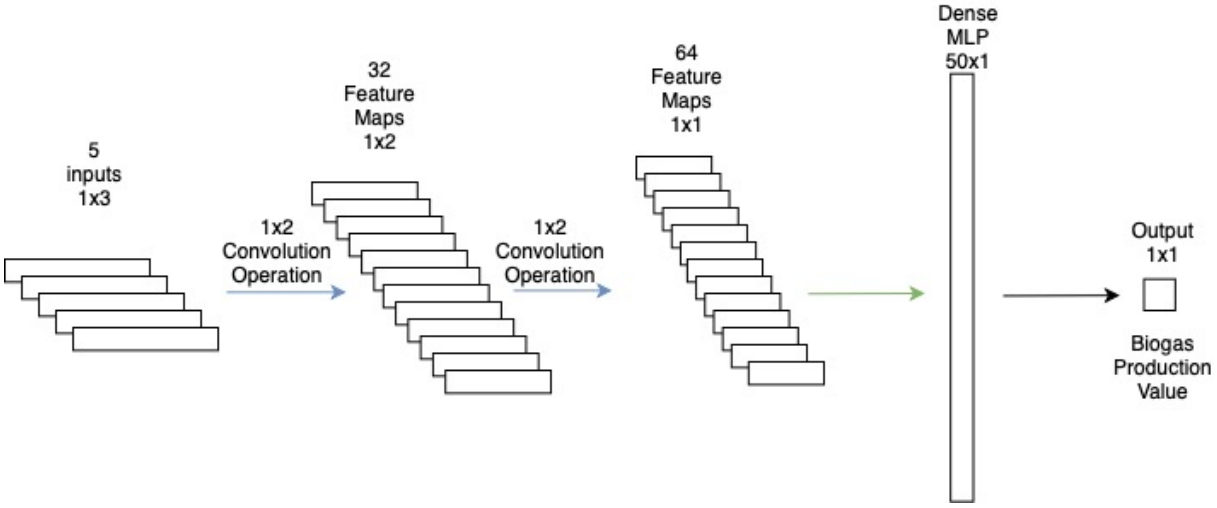

Figure 2: Model Architecture for Time-Window CNN Model.

with five rows, each representing an input feature was provided to the model. Each row had a lenght of 3 days, thus a short-term memory was proovided to the modified CNN model. This input is applied to a series of two convolutional layers. After the final convolution layer, feature maps are concatenated into a flattened vector as the MLP input for biogas production estimation. 'LeakyReLu' served as the activation function, and model training was performed by estimating CNN parameters with back propagation using the Adam optimizer to minimise the model's mean square error.

3.5.2 Quantum Models

A novel QCNN architecture was proposed by Cong et al. (2019). This architecture was utilised to develop quantum convolutional and quantum pooling layers for our quantum models. A CNN model's identifying properties are the translationally invariant convolution and pooling layers, each of which contains a fixed number of parameters and a sequential structure that continuously reduces the dimensional space of the data. Although the input of a given quantum circuit is a quantum state that the observer can not know, a QCNN

model can be built using the same principle as classical CNN models by passing a quantum circuit representing the input features as the input. The implementation of the Quantum Convolution and Pooling layers, as proposed by Cong et al. (2019) and implemented with TensorFlow (2021),can be found in Appendix A.

For the Quantum CNN model, convolution operations involved applying qubits gates between adjacent qubits while the pooling operations reduced the dimensions of the quantum model through measurement of qubits or by the applications of multiple-qubit gates like the CNOT gate. As Cong et al. (2019) propose, convolution and pooling layers were applied sequentially to reduce the dimensional space of the quantum system. Finally, a unitary, represented by a fully connected layer, was applied to the remaining qubits. Measurement was conducted on a singular output qubit to obtain the circuit output. Unitaries were learnt while keeping the number of convolution and pooling layers constant.

3.5.2.1 Discrete Data QCNN Model

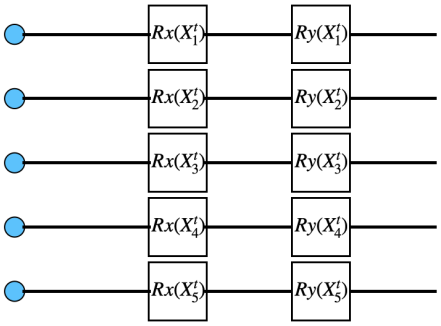

Figure 3: Dense Angle Encoded Quantum Circuit representing input datapoint for Discrete Data QCNN Model. Each wire accepts an input qubit and applies Rx and Ry transforms by rotating the quantum state of the input by the magnitude of the input feature in the X and Y-axis respectively.

Three pairs of quantum convolutional and quantum pooling layers were needed to accept quantum encoded data, represented as a single quantum circuit consisting of five qubits, with each qubit representing an input feature. A parameterized quantum circuit layer(PQC), serving as an MLP, was added after the QCNN layers to obtain the output. A Z-axis transform was applied to measure the quantum state of the qubit and obtain an expectation value representing the model's estimation of the biogas produced. An adjoint differentiator was used to specify how gradients were measured for the quantum circuits. The differentiator served as a quantum gradient descent function, performing forward and backward passes over the quantum circuit to differentiate it.

Model training was performed by estimating QCNN parameters with back propagation using the Adam optimizer to minimise the model's mean square error.

3.5.2.2 Short-Term Memory QCNN Model

After successfully developing a QCNN model for regression, a QCNN model with time-windowing was developed with time-windowing.

Since time-window size for an input feature n was three, the input expected for a time-window QCNN model was cluster of 3 circuits, each representing a discrete data point at $t = t-2, t = t-1, t = t$. These circuits were to be stacked as a series and fed as the input to the QCNN model. However, physically this is impossible since the input of a quantum circuit should always be a qubit. Thus, if a three circuit time-window were to be constructed, the inputs for future circuits would be the measured expectation values from prior circuits. By feeding in expectation values as inputs, any advantages gained by quantum machine learning would be lost.

In order to provide historical data to the model, a larger quantum circuit was built by concatenating circuits representing input at $t = t-2, t = t-1, t = t$ in series.

The structure of the time series QCNN beyond the shape of the input was similar to that of the QCNN regression model, with three pairs of quantum convolutional and quantum pooling layers. A parameterized

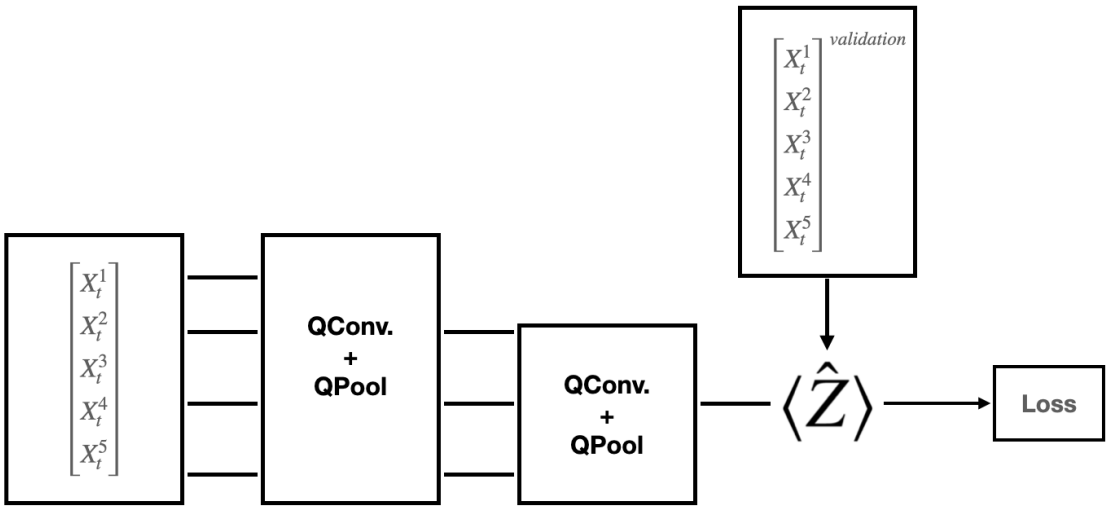

Figure 4: Model Architecture for Discrete Data QCNN Model.

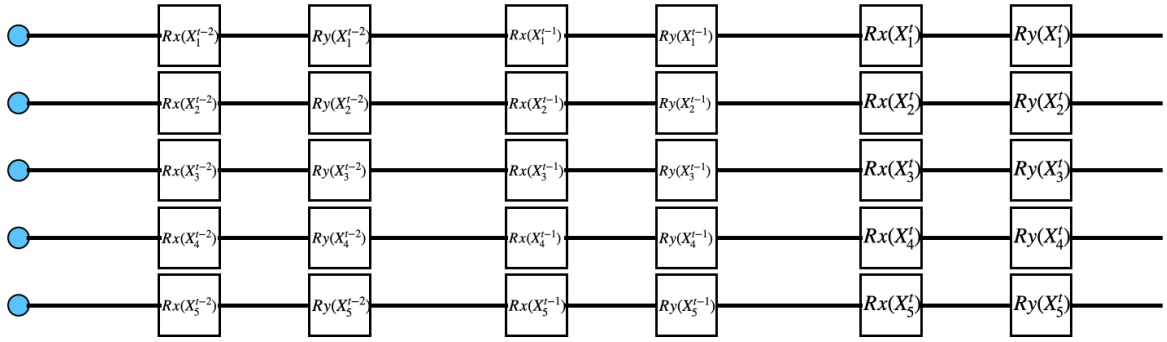

Figure 5: Dense Angle Encoded Quantum Circuit representing input datapoint for Time-Window QCNN Model. Each wire accepts an input qubit and applies a series Rx and Ry transforms by rotating the quantum state of the input by the magnitude of the input feature at $t-2, t-1$, and t in the X and Y-axis, respectively.

quantum circuit layer(PQC), serving as an MLP was concatenated to the end of the QCNN model to obtain the output, to which a Z-axis transform was applied in order to measure the quantum state of the output qubit and obtain an expectation value that represented the model's estimation of the biogas produced. An adjoint differentiator was used in order to specify how gradients were measured for the quantum circuits and model training was performed by estimating QCNN parameters with back propagation using the Adam optimizer to minimise the mean square error the model.

Three pairs of quantum convolutional and quantum pooling layers were needed to accept quantum encoded data, represented as a single quantum circuit consisting of five qubits, with each qubit representing an input feature. A PQC layer, served as an MLP, and was added after the QCNN layers to obtain the output. A Z-axis transform was applied to measure the quantum state of the qubit and obtain an expectation value representing the model's estimation of the biogas produced. Finally, an adjoint differentiator was used to specify how gradients were measured for the quantum circuits.

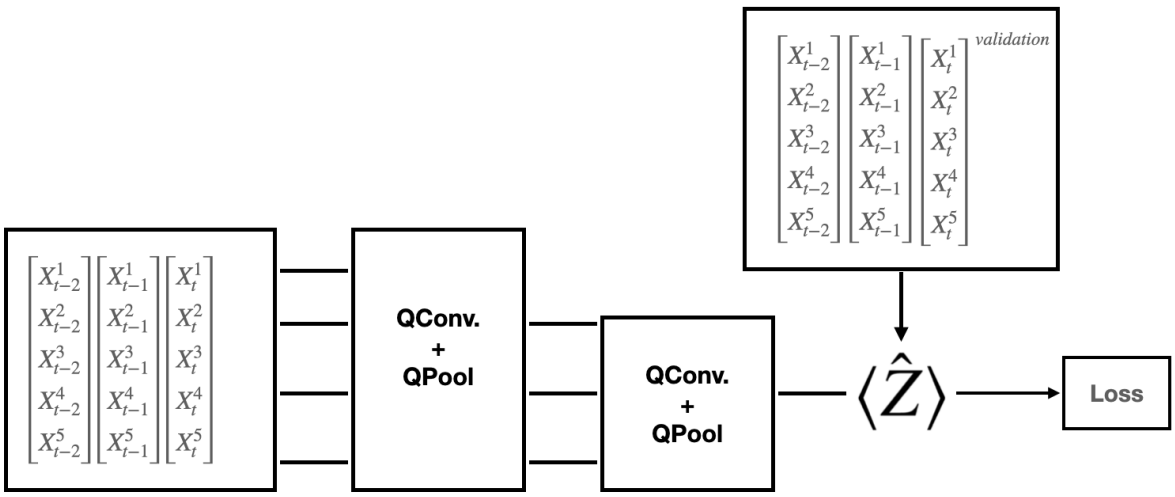

Figure 6: Model Architecture for Time-Window QCNN Model.

4 Experimental Results

		MAPE	Accuracy%	RMSE	RMSE%	MAE	MAE%
Discrete Data CNN	In-Sample	0.246	75.32	0.760	28.02	0.588	21.67
	Out-of Sample	0.388	61.21	1.066	39.05	0.948	34.77
Time Window CNN	In-Sample	0.182	81.78	0.418	15.44	0.298	11.01
	Out-of Sample	0.227	77.26	0.555	20.44	0.431	15.88
Discrete Data QCNN	In-Sample	0.148	85.16	0.099	17.91	0.080	14.49
	Out-of Sample	0.347	61.31	0.128	29.34	0.103	23.61
Time Window QCNN	In-Sample	0.306	69.36	0.142	33.03	0.117	27.37
	Out-of Sample	0.173	74.72	0.125	22.46	0.096	17.21

Figure 7: Results obtain for CNN models developed.

For the models built on classical architecture, time-window CNN model vastly outperforms the discrete data model. This is attributed to the lack of short-term memory within the discrete data model, making it unable to capture data trends. This is also evident in the forecasts generated by these models where the discrete data CNN model in Fig. 8a misses the trends observed in the beginning and the end of the out-of-sample data while the time-window CNN model is able to capture trends for the crests in biogas production observed in both the beginning and end of the data.

As demonstrated in Figure 7, classical CNN outperforms quantum models perform similarly for similar architectures. The time-window classical CNN model offers the highest accuracy at 77%. The quantum time-window CNN model offers similar accuracy at 74% but is outperformed by its classical counterpart in both RMSE% and MAE%. Raw RMSE and MAE were not considered when comparing classical models against quantum models since quantum-encoding for QCNN models changes the scale of output data; therefore RMSE%, MAE%, MAPE, and accuracy were used instead. By observing Fig. 8b and Fig. 8d, it is evident that although both short-term memory models perform similarly, the classical model is found to have the upperhand. The quantum time-window CNN model is able to track the trend of biogas production more accurately than its classical counterpart, it suffers from larger RMSE% and MAE% errors. The high variance in Fig. 8d is also attributed to the larger errors in the time-window QCNN model. The better overall performance of the classical time series model is attributed to the structure of the classical model's input data. Since the classical model does not require a transformation of inputs into quantum circuits, it suffers from less noise in its input features. Although the classical and quantum models trained to similar accuracies when similar parameters were provided, the classical neural network offers far less variance and lower errors than the quantum neural network for fundamentally classical data. Thus the classical time-window CNN model is proposed as the better model among all developed.

Interestingly, the discrete data QCNN model, built as a proof-of-work for the time-window QCNN model, offers marginally better accurqacies for forecasting but vastly outperforms its classical discrete data counterpart in error metrics, offering nearly 25% smaller error margins. This is attributed to the small dimensional size of the dataset. While the classical time-window CNN model performs better due to the lack of quantum encoding, the discrete data CNN is outperformed by its quantum counterpart. This is attributed to the relatively simple nature of the inputs provided to discrete data models. Given an input of $5x1$, the discrete data QCNN encodes the input data as a quantum circuit and considers input values in large dimensions by simulating a Hilbert-Space, thus providing more excellent feature extractions than the classical discrete data CNN model. For increasing dimensional complexity, the relative performance of a QCNN compared to a classical CNN model is expected to decrease, as is evidenced with the time-series CNN models, where quantum circuits developed for data of dimension $5x3$ lead to increased errors in forecasts.

It was also observed that both the quantum models required far fewer epochs to obtain accurate results.

(a) Discrete Data Classical CNN

(b) Time-Window Classical CNN

(c) Discrete Data Quantum CNN

(d) Time-Window Quantum CNN

Figure 8: CNN Model performances for out-of-sample data.

The current best forecasting model, ANFIS offers RMSE and MAE values of 3.986, 0.673 respectively, for out-of-sample data. These metrics are beaten by all models developed during this project. The better performance of the models is attributed to the rigorous preprocessing applied to classical and quantum models alike. Ensuring that the data was made stationary with exponential smoothing, imputation of missing data with Random Forest regressors, and the reduction of dimensional size by investigating feature-label correlations ensured that minimal noise is added to the developed models. It should be noted that the high data sparsity within the dataset led to the removal of crucial input features obtained by domain knowledge provided by the Hong-Kong waste management facility stakeholders. The addition of Q_{in}, $DS\%$, and $VS\%$of Thickened Primary Sludge(TPS), Thickened Waste Activated Sludge(TWAS), and the Q_{out}, $DS\%$, $VS\%$ of Digested Sludge(DS) is expected to influence the performance of the developed models positively, allowing for ever more accurate biogas production estimates which would be hugely beneficial for waste-management facilites. Since LSTM models are known to outperform models built on an ANFIS architecture(Do & Trang 2020), the better performance of the CNN models developed in this project is to be expected due to CNNs' comparable performance to LSTMs(Chen & He 2018).

5 Conclusions and Future Work

Biogas production was accurately forecasted with a modified CNN model using small subsets of data as a short-term memory for input features. A first of its kind quantum multi-variate time series forecasting model was built on a modified QCNN architecture by replicating a classical CNN model, where features were encoded into a quantum circuit. Short-Term memory was provided to the model by concatenating circuits from previous time intervals to the beginning of the input quantum circuit.

The classical time-window CNN model was the best performer among all models developed, offering the lowest error values, and the largest accuracies. Due to the high sparsity within the dataset, key features identified with domain knowledge obtained by collaborating with stakeholders in Hong Kong's waste management facility could not be used. Future work involves the use of denser datasets from the facility for forecasts with even

higher accuracy. All developed models were found to be better than the current best model, ANFIS, despite the high sparsity found within the data.

While classical and quantum neural networks with equal paramater sizes offer similar results, it is generally proposed that CNNs be preferred for classical data while QCNNs are preferred for quantum data. However, QCNNs were able to find more accurate models at a much faster rate than their classical counterparts. The development of Hybrid QCNNs is thus proposed to explore the utilisation of classical and quantum convolution and pooling layers to develop models that offer the benefits of faster training times from quantum layers and better accuracies from classical layers.

Future work also involves the development of hybrid quantum LSTM models and an input container to accept multiple circuits structured as a binomial tree to simulate short term-memory without the need to manually concatenate quantum circuits quantum architecture models.

References

Albawi, S., Mohammed, T. A. & Al-Zawi, S. (2017), Understanding of a convolutional neural network, *in* '2017 International Conference on Engineering and Technology (ICET)', pp. 1–6.

Bausch, J. (2020), 'Recurrent quantum neural networks'.

Biamonte, J., Wittek, P., Pancotti, N., Rebentrost, P., Wiebe, N. & Lloyd, S. (2017), 'Quantum machine learning', *Nature* **549**(7671), 195–202.

Chen, S. & He, H. (2018), Stock prediction using convolutional neural network, *in* 'IOP Conference series: materials science and engineering', Vol. 435, IOP Publishing, p. 012026.

Chen, S. Y.-C., Yoo, S. & Fang, Y.-L. L. (2020), 'Quantum long short-term memory'.

Cong, I., Choi, S. & Lukin, M. D. (2019), 'Quantum convolutional neural networks', *Nature Physics* **15**(12), 1273–1278.
URL: *http://dx.doi.org/10.1038/s41567-019-0648-8*

Do, Q. H. & Trang, T. (2020), 'Forecasting vietnamese stock index: A comparison of hierarchical anfis and lstm', *Decision Science Letters* **9**, 193–206.

Goodfellow, I., Bengio, Y. & Courville, A. (2016), *Deep Learning*, MIT Press. `http://www.deeplearningbook.org`.

Hansen, K., Montavon, G., Biegler, F., Fazli, S., Rupp, M., Scheffler, M., Von Lilienfeld, O. A., Tkatchenko, A. & Muller, K.-R. (2013), 'Assessment and validation of machine learning methods for predicting molecular atomization energies', *Journal of Chemical Theory and Computation* **9**(8), 3404–3419.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C. & Oliphant, T. E. (2020), 'Array programming with NumPy', *Nature* **585**(7825), 357–362.
URL: *https://doi.org/10.1038/s41586-020-2649-2*

Hunter, J. D. (2007), 'Matplotlib: A 2d graphics environment', *Computing in Science & Engineering* **9**(3), 90–95.

Kusiak, A. & Wei, X. (2014), 'Prediction of methane production in wastewater treatment facility: a data-mining approach', *Annals of Operations Research* **216**(1), 71–81.

Labatut, R. A. & Pronto, J. L. (2018), Chapter 4 - sustainable waste-to-energy technologies: Anaerobic digestion, *in* T. A. Trabold & C. W. Babbitt, eds, 'Sustainable Food Waste-To-energy Systems', Academic Press, pp. 47–67.
URL: *https://www.sciencedirect.com/science/article/pii/B9780128111574000048*

LaRose, R. & Coyle, B. (2020), 'Robust data encodings for quantum classifiers', *Physical Review A* **102**(3).
URL: *http://dx.doi.org/10.1103/PhysRevA.102.032420*

Lauwers, J., Appels, L., Thompson, I. P., Degrève, J., Van Impe, J. F. & Dewil, R. (2013), 'Mathematical modelling of anaerobic digestion of biomass and waste: Power and limitations', *Progress in Energy and Combustion Science* **39**(4), 383–402.
URL: *https://www.sciencedirect.com/science/article/pii/S0360128513000178*

LeCun, Y., Bengio, Y. & Hinton, G. (2015), 'Deep learning', *nature* **521**(7553), 436–444.

Nelson, P. R., Taylor, P. A. & MacGregor, J. F. (1996), 'Missing data methods in pca and pls: Score calculations with incomplete observations', *Chemometrics and Intelligent Laboratory Systems* **35**(1), 45–65.
URL: *https://www.sciencedirect.com/science/article/pii/S016974399600007X*

Nielsen, M. A. & Chuang, I. L. (2010), *Quantum Computation and Quantum Information: 10th Anniversary Edition*, Cambridge University Press.

O'Shea, K. & Nash, R. (2015), 'An introduction to convolutional neural networks', *arXiv preprint arXiv:1511.08458* .

pandas development team, T. (2020), 'pandas-dev/pandas: Pandas'.
URL: *https://doi.org/10.5281/zenodo.3509134*

Pattanayak, S. & Pattanayak, S. (2021), 'Quantum machine learning', *Quantum Machine Learning with Python: Using Cirq from Google Research and IBM Qiskit* pp. 221–279.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. & Duchesnay, E. (2011), 'Scikit-learn: Machine learning in Python', *Journal of Machine Learning Research* **12**, 2825–2830.

Phillipson, F. (2020), Quantum machine learning: Benefits and practical example.

Ramachandran, A., Rustum, R. & Adeloye, A. J. (2019), 'Review of anaerobic digestion modeling and optimization using nature-inspired techniques', *Processes* **7**(12), 953.

Schuld, M., Sinayskiy, I. & Petruccione, F. (2015), 'An introduction to quantum machine learning', *Contemporary Physics* **56**(2), 172–185.
URL: *https://doi.org/10.1080/00107514.2014.964942*

Seabold, S. & Perktold, J. (2010), statsmodels: Econometric and statistical modeling with python, *in* '9th Python in Science Conference'.

Tchagang, A. B. & Valdés, J. J. (2019), Prediction of the atomization energy of molecules using coulomb matrix and atomic composition in a bayesian regularized neural networks, *in* 'International Conference on Artificial Neural Networks', Springer, pp. 793–803.

TensorFlow (2021), 'Tensorflow'. Specific TensorFlow versions can be found in the "Versions" list on the right side of this page.¡br¿See the full list of authors ¡a href="htt ps://github.com/tensorflow/tensorflow/graphs/contr ibutors"¿on GitHub¡/a¿.
URL: *https://doi.org/10.5281/zenodo.5189249*

Tucci, R. (2005), 'An introduction to cartan's kak decomposition for qc programmers'.

Wang, H. & Raj, B. (2017), 'On the origin of deep learning'.

Wang, L., Long, F., Liao, W. & Liu, H. (2020), 'Prediction of anaerobic digestion performance and identification of critical operational parameters using machine learning algorithms', *Bioresource Technology* **298**, 122495.
URL: *https://www.sciencedirect.com/science/article/pii/S0960852419317250*

Wang, P., Wang, H., Qiu, Y., Ren, L. & Jiang, B. (2018), 'Microbial characteristics in anaerobic digestion process of food waste for methane production–a review', *Bioresource Technology* **248**, 29–36. Bioconversion of Food Wastes.
URL: *https://www.sciencedirect.com/science/article/pii/S0960852417310568*

Waskom, M. L. (2021), 'seaborn: statistical data visualization', *Journal of Open Source Software* **6**(60), 3021.
URL: *https://doi.org/10.21105/joss.03021*

Wu, Z., Ramsundar, B., Feinberg, E. N., Gomes, J., Geniesse, C., Pappu, A. S., Leswing, K. & Pande, V. (2018), 'Moleculenet: a benchmark for molecular machine learning', *Chemical science* **9**(2), 513–530.

Xie, Y., Xing, F., Kong, X., Su, H. & Yang, L. (2015), Beyond classification: structured regression for robust cell detection using convolutional neural network, *in* 'International conference on medical image computing and computer-assisted intervention', Springer, pp. 358–365.

A Appendix

A.1 Validation Metiricx

Where e_t, f_t, d_t are errors, predicted value, and expected value at time t, the metrics used for model validation are defined as given below:

$$e_t = f_t - d_t \tag{7}$$

$$MAPE = \frac{1}{n}\sum \frac{|e_t|}{d_t} \tag{8}$$

$$Accuracy\% = 100(1 - MAPE) \tag{9}$$

$$MAE = \frac{1}{n}\sum |e_t| \tag{10}$$

$$MAE\% = \frac{\sum |e_t|}{\sum d_t} \tag{11}$$

$$RMSE = \sqrt{\frac{1}{n}\sum e_t^2} \tag{12}$$

$$RMSE\% = \frac{\sqrt{\frac{1}{n}\sum e_t^2}}{\frac{\sum d_t}{n}} \tag{13}$$

A.2 QCNN Implementation

The design of quantum convolutiona and pooling layers was first proposed by Cong et al. (2019) built using unitary matrices proposed by Tucci (2005). This architecture was implemented by TensorFlow (2021) and has been used in developing the QCNN regression models investigated in this project. The implementation can be viewed down below.

```
def one_qubit_unitary(bit, symbols):
        return cirq.Circuit(
            cirq.X(bit)**symbols[0],
            cirq.Y(bit)**symbols[1],
            cirq.Z(bit)**symbols[2])

def two_qubit_unitary(bits, symbols):
    circuit = cirq.Circuit()
    circuit += one_qubit_unitary(bits[0], symbols[0:3])
    circuit += one_qubit_unitary(bits[1], symbols[3:6])
    circuit += [cirq.ZZ(*bits)**symbols[6]]
    circuit += [cirq.YY(*bits)**symbols[7]]
    circuit += [cirq.XX(*bits)**symbols[8]]
    circuit += one_qubit_unitary(bits[0], symbols[9:12])
    circuit += one_qubit_unitary(bits[1], symbols[12:])
    return circuit

def two_qubit_pool(source_qubit, sink_qubit, symbols):
    pool_circuit = cirq.Circuit()
    sink_basis_selector = one_qubit_unitary(sink_qubit, symbols[0:3])
    source_basis_selector = one_qubit_unitary(source_qubit, symbols[3:6])
    pool_circuit.append(sink_basis_selector)
    pool_circuit.append(source_basis_selector)
    pool_circuit.append(cirq.CNOT(control=source_qubit, target=sink_qubit))
    pool_circuit.append(sink_basis_selector**-1)
    return pool_circuit
```

```
def quantum_pool_circuit(source_bits, sink_bits, symbols):
    circuit = cirq.Circuit()
    for source, sink in zip(source_bits, sink_bits):
        circuit += two_qubit_pool(source, sink, symbols)
    return circuit

def quantum_conv_circuit(bits, symbols):
    circuit = cirq.Circuit()
    for first, second in zip(bits[0::2], bits[1::2]):
        circuit += two_qubit_unitary([first, second], symbols)
    for first, second in zip(bits[1::2], bits[2::2] + [bits[0]]):
        circuit += two_qubit_unitary([first, second], symbols)
    return circuit
```