UNIVERSIDAD EAFIT

Predicción automática de contadores eléctricos: una comparación entre el aprendizaje automático cuántico y el aprendizaje automático clásico

Automatic Electrical Meter Forecasting: a Benchmarking Between Quantum Machine Learning and Classical Machine learning

Jonathan Javier Montes Castro

Trabajo de Grado Modalidad Investigación

Asesores

Juan Guillermo Lalinde Pulido

Daniel Sierra-Sosa

UNIVERSIDAD EAFIT

ESCUELA DE CIENCIAS APLICADAS E INGENIERÍA

MAESTRÍA EN CIENCIAS DE LOS DATOS Y LA ANALÍTICA

MEDELLÍN

2024

Automatic Electrical Meter Forecasting: a Benchmarking Between Quantum Machine Learning and Classical Machine learning

Jonathan J. Montes C.[1*†], Daniel Sierra-Sosa[2†] and Juan G. Lalinde-Pulido[3†]

[1*]Escuela de Ciencias Aplicadas e Ingeniería, Universidad EAFIT, Cra. 49 No. 7Sur-50, Medellín, 050022, Antioquia, Colombia.
[2]Department of Electrical Engineering and Computer Science, The Catholic University of America, 620 Michigan Ave., N.E., Washington, DC, 20064, Washington, DC, United States.
[3]Área de Ciencias Fundamentales, Universidad EAFIT, Cra. 49 No. 7Sur-50, Medellín, 050022, Antioquia, Colombia.

*Corresponding author(s). E-mail(s): jmontesc@eafit.edu.co;
Contributing authors: sierrasosa@cua.edu; jlalinde@eafit.edu.co;
†These authors contributed equally to this work.

Abstract

This work benchmarks Quantum Long Short-Term Memory (QLSTM) against classical LSTM networks using electrical meter data (KWh) from EPM, a public utility company, clients. The results show that QLSTM models learn in half the epochs compared to LSTM, as measured by the MSE cost function, while maintaining strong performance with respect to bias (Mean Absolute Percentage Error, MAPE) and variance ($\boldsymbol{R^2}$) metrics. QLSTM leverages variational quantum circuits (VQC) to replace traditional LSTM cell gates, demonstrating the potential of quantum-hybrid algorithms in forecasting tasks. This study highlights the efficiency and accuracy advantages of quantum machine learning applied to real-world data from EPM's electrical metering services.

Keywords: Quantum Machine Learning, Machine Learning, Quantum Computation, Variational Quantum Circuits, QLSTM, Amplitude Encoding.

1 Introduction

Forecasting the consumption of public services such as water, electricity, and gas is critical for decision-making in utility companies. In the electricity sector, the main goal is to ensure reliable and continuous service; however, predicting electrical loads has become increasingly complex due to various direct and indirect influencing factors. Traditional forecasting methods often struggle with this complexity, requiring the development of reliable and efficient predictive models. Accurate forecasting helps companies manage energy transmission, plan resources, and identify potential issues proactively, which is crucial given the rapid rise in energy consumption [1].

Load forecasting generally focuses on the hourly total electric load but can extend to predicting hourly, daily, weekly, and monthly system loads, including peak loads. Forecasts are classified by time horizon: short-term load forecasting (STLF) for up to one day; medium-term load forecasting (MTLF) for one day to one year; and long-term load forecasting (LTLF) for one to ten years. For larger systems like regional or national grids, these classification models achieve relatively high accuracy [2], [3]. This paper shows how the forecasting was addressed over a period of days, specifically targeting one to two week forecasts. In the existing literature, different investigations have shown that quantitative forecasting techniques such as moving averages [4], time series [5] and deep learning methods [6] are employed when the situation is steady and previous data is available, such as in the case of this work. A research suggests that traditional time series analysis techniques, such as regression models, ARIMA, GARCH, and hybrid models combining ARIMA and GARCH with wavelet transforms, are not suitable for short-term forecasting in high-dimensional or highly volatile data settings [7] as these methods often struggle with scalability and adaptability in such contexts. In contrast, artificial neural networks (ANNs) are more adept at handling the complexities of short-term forecasting. ANNs leverage their hidden layers and learning capabilities to identify and exploit hidden patterns in time series data, leading to more accurate forecasts. They are particularly valued for several key features:

1. Robustness: ANNs can generalize well even with incomplete or noisy data.
2. Non-parametric Nature: They do not require predefined assumptions about the data distribution, making them flexible in various applications. Generally speaking, in data science problems, this is an approximation that is often overlooked, and that is to assume a normal distribution in the data, which is not true for all cases.
3. Universal Approximation: ANNs can model any continuous function to a desired level of accuracy.

Recent advancements in short-term load forecasting have seen the integration of advanced machine learning techniques. In China, a hybrid model combining variation mode decomposition (VMD) with long short-term memory (LSTM) networks, optimized via Bayesian optimization (BOA), demonstrated superior performance, achieving a MAPE of 0.4186% and an R-squared of 0.9945 compared to other models like SVR, MLP regression, LR, RF, and EMD-LSTM [8]. Artificial neural networks (ANNs) have been effectively utilized for short-term load forecasting, particularly in handling non-linear data. For instance, in [9] the authors developed an ANN-based model to process vast and dynamic datasets, addressing non-linearity challenges in

historical load data, with validation performed on a real test facility. Other neural network architectures, including RNNs, CNNs, LSTMs, and deep networks, have also been employed to improve forecast accuracy [10]. Notably, LSTM networks have shown to outperform traditional statistical and machine learning approaches in reducing prediction errors [11], specifically, showing that LSTM models have been proven effective in residential short-term load forecasting, significantly outperforming models like ELM, BPNN, and k-nearest neighbor regression. In a similar research context (in which QLSTM is used), such as solar irradiance prediction [12], various recurrent neural network (RNN) models, including gated recurrent units (GRU) and long short-term memory (LSTM), have been utilized for forecasting one hour and one day ahead [13]. To enhance the accuracy of time-series forecasts, some researchers have combined LSTM with other models to better extract features from the data.

The primary objective of this work was to implement a predictive model and architecture for EPM, a public utility company in the electricity sector, to deliver measurements and forecasts with optimal accuracy and precision. For EPM, this model must be highly reliable and fast to proactively identify users who may experience issues, enabling efficient reviews and solutions, ideally before network failures occur. Accurate electricity usage forecasting is crucial for enhancing the intelligence of smart grids, as it allows utility providers to efficiently plan resources and take controlled actions to balance supply and demand. Additionally, metering solutions provide customers with insights into their energy consumption and future usage projections, aiding in better cost management. In this work, we propose a method for precise short-term electricity load forecasting for the next 24 hours and week, at the individual household level by social class, and the industry category. This need for rapid and precise predictions brings quantum machine learning into focus as a cutting-edge solution. But firstly, as a benchmark, the project began with classical LSTM models to evaluate their forecasting capabilities across different user groups, and to assess group dependencies and independencies in time series forecasting. This classical approach provides a baseline for comparing and guiding the development of quantum-enhanced LSTM (QLSTM) models.

Recent research and applications on QML have not shown, in some cases, superiority in terms of model accuracy in comparison with the classical counterpart. One example is the case of study of heart disease in which a quantum machine learning algorithm in SVM to classify artery disease was made. The results presented [14] show that QML and tensor network techniques can provide comparable accuracy to classical methods in noninvasive testing for coronary artery disease for 3 different dataset, in which the best QML model was for VQC MPS model showing for the best angle of rotation applied to the input feature space, but the classical counterpart showed better results (Naive Bayes and XGBoost), those in terms on the following metrics: accuracy, precision, recall and F1-score. Also, studies in [14] shows good results using VQC methods for classification problems, achieving more than 70% of accuracy in terms of classification metrics.

Like ML, QML paradigms are supervised learning or task based, unsupervised learning or data based, and reinforced learning or reward based. Due to its resilience to noise, good generalization properties, and in a broad sense, its potential to achieve a

quantum advantage, supervised learning has received special attention in recent years [15]. QML algorithms speed up quantum systems to improve ML regression or classification tasks, for example, through the Quantum Support Vector Machine, Quantum PCA, Variational Quantum Classifier, Quantum Boltzmann Machine, Quantum Neural Network (QNN), Quantum Convolutional Neural Network, and Quantum Deep Neural Network [16].

QNNs refer to the application in a data science problem of parameterized quantum circuits, which are a sequence of unitary gates acting on the quantum data states, some of which have free parameters that will be trained to solve a problem. This algorithm deserves special attention since it is used in all three QML paradigms. For instance, the goal of a QNN in a supervised classification task may be to map states in different classes to distinguishable regions of Hilbert space. In the unsupervised learning case, a clustering task can be mapped to a MAXCUT problem and solved by training a QNN to maximize the distance between classes; while in a reinforced learning task, a QNN can be used as a Q-function approximator to establish the best action of an agent given its current state [15].

In some cases, hybrid approaches are used with models that have classical and quantum neural networks, which seek to distribute representational capacity and computational complexity across classical and quantum computing. In other cases, researchers have proposed quantum versions of kernel methods, which map each input to a vector and learn a linear function in the reproducing kernel Hilbert space, which has a dimension that could be infinite and makes the kernel method potent in terms of its expressiveness [15].

The challenges associated with current quantum computers, particularly noisy intermediate scale quantum (NISQ) devices, are well-documented. These devices face difficulties in executing quantum circuits with large number of qubits or extensive circuit depths due to the lack of effective quantum error correction mechanisms [17]. Addressing this problem, recent work by Chinese researchers introduces a novel framework utilizing Variational Quantum Circuits (VQC) to implement Recurrent Neural Networks (RNNs), specifically Long Short-Term Memory (LSTM) networks. Their approach, termed Quantum LSTM (QLSTM), combines quantum and classical computing to leverage the expressive power of quantum entanglement while maintaining practical feasibility for NISQ devices. This hybrid quantum-classical framework shows promising results, including faster learning and more stable convergence compared to classical LSTMs in several physical applications[17] and in the prediction of solar irradiance [12]. Additionally, VQC has been demonstrated as a leading candidate for shallow quantum algorithms, successfully applying to classical classification, clustering, and even deep reinforcement learning [12]. Building on these advancements, this paper shows how the QLSTM perform better in terms of loss function, learning faster, approximately in half of the epochs for the classical LSTM, in the context of electrical meter forecasting.

This work is organized as follows: in section 2 is reviewed in detail the concept and context of QML, also each subsection in this chapter shows the theoretical background

of the LSTM, the importance of the bias-trade off explaining how a model in the classical forecasting can be defined as a good model in terms of loss functionals, also the workflow of a quantum machine learning algorithm is explained and the QLSTM is briefly explained. In this section the parameter shift-rule used in VQC as the optimization procedure is demonstrated, and a subsection named "a bit of physics" in which the concept of qubits, superposition and entanglement are reviewed as part of the explanation of the VQC architecture in each cell memory in the QLSTM. Section 3 contains elements used for this work, such as the embedding process explained in a deeper way; encoding, which is from utter importance in the quantum computing world, showing how the classical data is transformed to be used in the QLSTM. Also metrics performance used in this work are explained. Section 4 has an important part of this work, data description, preprocessing and feature engineering. This section describes the data provided by EPM, transformations and feature engineering of the data. Section 5 shows all the results in terms of clients, loss functions used, clients groups/categories independence and dependence tests, LSTM and QLSTM models performance for one client showing the best hyperparameters of each model. Finally there are two sections, 6 with conclusions of this work and 7 with the acknowledgments to the people and researchers who were part of this work.

2 Methodology

Since our classical machine learning problem is covered by artificial neural networks (ANNs), our comparison framework with QML has to be in the quantum version of ANNs, called Quantum Neural Network (QNN) [16]. It has been demonstrated that classical neural networks can be embedded into parameterized quantum circuits (PQC) [15] also known as Variational Quantum Circuit. These are a sequence of unitary gates acting on the quantum data states $|\psi_j\rangle$ some of which have free parameters θ that will be trained to solve an optimization problem [15]. Fig 1 shows one example of a VQC [16], we illustrate this later in subsection 2.5. In fig 2, taken from [15], we can see the schematic version of Feedforward Neural Network FNN in the quantum computation world in which each node corresponds to a qubit, while lines connecting qubits are unitary operations. Table 1, taken from [16], summarizes this correspondence between the quantum world and neural networks.

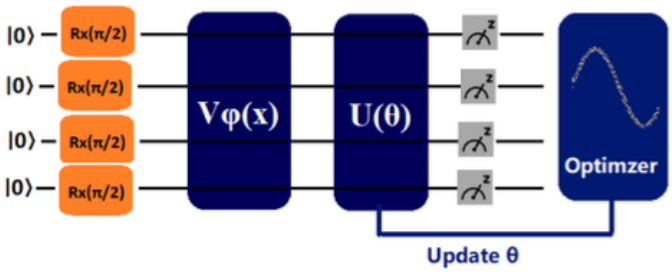


Fig. 1 Variational Quantum Circuit.[16].

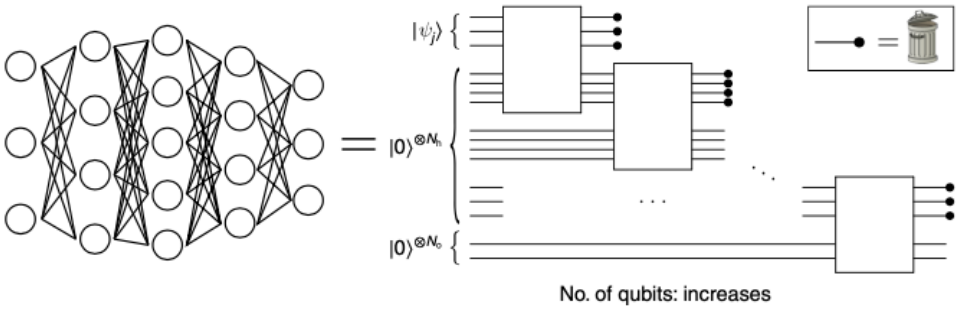


Fig. 2 Quantum circuit representation of a FNN. The circuit is initialized in a product state. [15].

Quantum Mechanics	Neural networks
Wave function	Neurons activation functions
Superposition (Coherence)	Interconnections (weights)
Measurement (De-coherence)	Evolution to attractor
Entanglement	Learning rule
Unitary transformation	Gain function

Table 1 Correspondence between the quantum mechanics elements in a VQC with a classical Neural network.

Both in the quantum world and in the classical one, the FNN uses the backpropagation algorithm that is based on the gradient descent [18, 19]. It was the aim of this research to use the back propagation algorithm in the LSTM and the parameter-shifted rule for QVC in the QLSTM to make a benchmark in terms of complexity and accuracy in the forecasting problem. In a variational quantum circuit (VQC), the parameter shift rule is used to compute the derivative of the expectation value with respect to a parameter θ of a unitary gate. This is achieved by evaluating the circuit at $\theta + \frac{\pi}{2}$ and $\theta - \frac{\pi}{2}$, and then combining these results to estimate the gradient [19]. This process allows for the optimization of the parameters in the VQC, analogue to how gradients are used in feedforward neural networks (FNNs) to optimize weights and biases through backpropagation, more precisely in our case of study backpropagation through time (BPTT) [20]. The circuit structure, in this case, is reused with slightly shifted parameters to find these derivatives, which play a role similar to the adjustment of parameters in an FNN during training. This demonstration is made on subsection 2.6 based on the article [19]. Several works, such as [21], use these analytic gradients on quantum hardware for hybrid quantum-classical algorithm. Given that quantum algorithms used to estimate the gradient of a model with D parameters acting on n-qubits require $O(\sqrt{D}/\epsilon)$ queries to estimate the gradient within error ϵ (in the infinity-norm), it implies that with high probability the gradient will require $O(\text{poly}(2^n))$ applications of the model to even learn the sign of any component of the gradient accurately [18].

Several researchers have proven the above mentioned, such as Amin et. al [22]. These authors used QNN to classify COVID desease with high accuracy and precision, implementing 3 dense layers, 500 neurons, ReLU activation, and 02 neurons with SoftMax for feature mapping with 4 qubits on Pennylane, using COVID 19 Pakistani and Chinese patient's lungs CT images [16]. Also E. El-Shafeiy et. al [23] used QNN model with 30 nodes getting 92.34% of accuracy in the prediction of severity of COVID-19 [15].

The following two subsections are intended to set the theoretical background and the methodology that will be used for each framework: classical forecasting defining the evaluation metrics of a machine learning model along with a review of FNN networks with LSTM, and quantum forecasting in which the hybrid version of LSTM with VQC (QLSTM) was used. It is important to define now what the differences are between both in terms of the construction steps of a classical machine learning model and a quantum machine learning model. But firstly, let's define what Quantum machine learning (QML) is. QML is the combination between machine learning and quantum computing. It is based on the next idea: if quantum processors are capable of producing statistical patterns that are computationally difficult for a classical computer to produce, then it is very likely that they can also recognize patterns on such information that are equally difficult to find classically [15]. The steps and differences in developing a machine learning model and a quantum machine learning model are detailed in Figure 3.

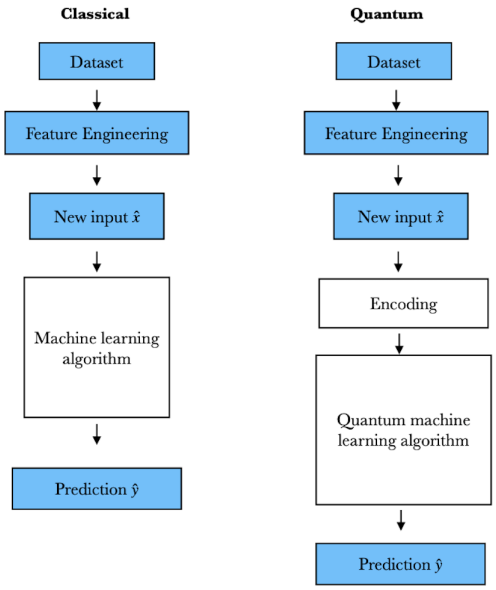


Fig. 3 Classical and Quantum machine learning models steps [24].

The main difference between classical and quantum machine learning models starts at how the information is processed by a quantum computer. After the feature engineering process, which is one of the most important steps in a machine learning model due to its susceptibility with respect to the type of data that is taking to train the model, *i.e*, using the most correlated variables/features and the standarization of the variables, these data and new inputs have to be encoded in the quantum world. Encoding techniques such as amplitude encoding and basis encoding, are procedures to transform data stored in a classical memory into a quantum state. For example, encoding a vector $\vec{x}$ into a qubit base vector which lives in a Hilbert space. After the encoding step, it is important to select the proper QML algorithm what will be applied depending on the machine learning problem [15, 16]. Then, encoded input passes through a unitary evolution or gates in the quantum computing framework and finally the measurement of the quantum system [16].

2.1 The bias/variance-tradeoff: the model evaluation in ML forecasting

In terms of classical machine learning, a supervised learning problem is considered where we have to infer from historical data the possible nonlinear dependency between the input, past embedding vector, and the output, future value. See figure 4.

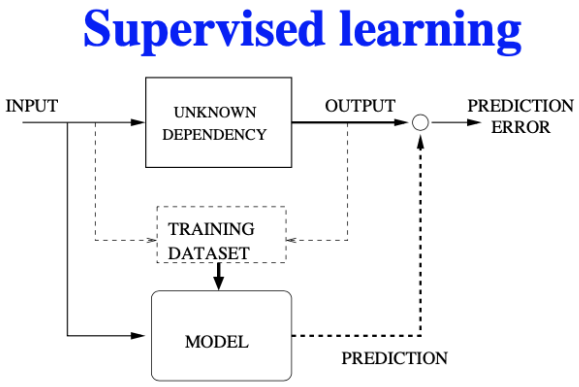


Fig. 4 Supervised classical machine learning scheme.[25].

For the aim of forecasting with this ML technique, it is important to choose a model complexity that satisfies a good bias/variance trade-off, which we can see in figure 5, which refers to a MISE, *i.e*, mean-integrated-squared-error, decomposed in three terms [20, 26]:

$$MISE = \sigma_w^2 + squared\ bias + variance,$$

Where σ represents the noise or random error of the regression model $y = f(x+w)$, and w is the input's error. The following task will be choosing the functional $\hat{f}(x, \alpha)$

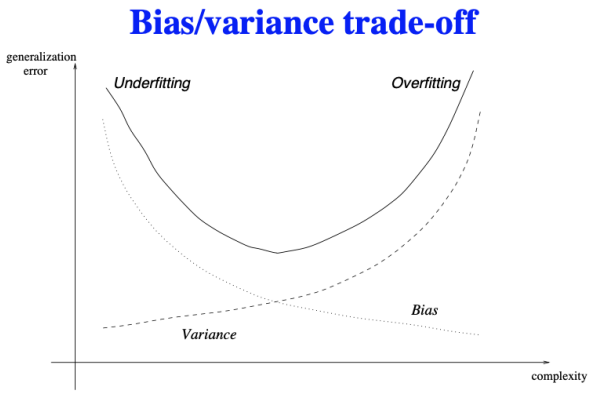


Fig. 5 Bias/variance trade-off.[25].

that depends on the degree of the functional (linear or nonlinear), and the hiperparameters. This decomposition explains how different factors contribute to prediction errors, particularly in non-parametric regression models. Here, the *variance* refers of how much the $\hat{f}(x)$ will move around its mean, and *squared bias* refers to the difference between the true model and the model being fitted, *i.e*, how well the model approximates the true underlying function.

In this work, the functional used to determine the minimal between $\hat{f}(x, \alpha)$ and y was the Mean Squared Error. This, similar to MISE is decomposed as [26]:

$$MSE = \sigma_w^2 + \text{squared bias} + \text{variance},$$

For the feature engineering of the data in this forecasting problem, this work was based on a backward step-wise selection, in which the starting point of the variable removing began with a correlation of the variables. This process, explained in the section 4, aims to get the best R^2 (*variance*) for the LSTM models in which the point that optimizes the *bias* is precisely in the optimization point to avoid underfitting and overfitting in terms of the training data set. This can be seen in figure 5.

For this purpose, we firstly have to make the data cleansing in terms of NaN variables and get the correlations between the variables in order to take into account endogenous or exogenous variables for the forecasting of the meters in terms of categories, locations, etc. R^2 and MAPE are going to be used as metrics of prediction, variance and bias metrics respectively. CRISP-DM methodology was applied for our classical scope, which is a process model that serves as the base for a data science process and is based on the following steps: business understanding, data understanding, data preparation, modeling, evaluation and deployment. The metrics mentioned previously are our benchmarking metrics so to speak. This will be explained in detail in section 4.

2.2 LSTM

Long Short-Term Memory (LSTM) networks were introduced by Sepp Hochreiter and Jürgen Schmidhuber to address the memory issues inherent in Recurrent Neural

Networks (RNNs) [27],[20]. RNNs have the ability to maintain a form of short-term memory, which allows them to process sequences of data. However, they struggle to retain information over long sequences due to the vanishing gradient problem, which leads to a loss of important information as it passes through multiple steps. This makes it challenging for RNNs to learn long-term dependencies in data sequences[20].

LSTMs were developed to alleviate these issues. An LSTM cell is similar to a standard RNN cell but introduces two separate vectors: $\mathbf{h}(t)$, the short-term state, and $\mathbf{c}(t)$, the long-term state. These two vectors work together to selectively retain and discard information throughout the sequence, enabling the network to learn dependencies over longer time spans. The structure of an LSTM cell includes three main components called gates: the Forget Gate, the Input Gate, and the Output Gate, as seen in Figure 6, where σ is the logistic or sigmoid function.

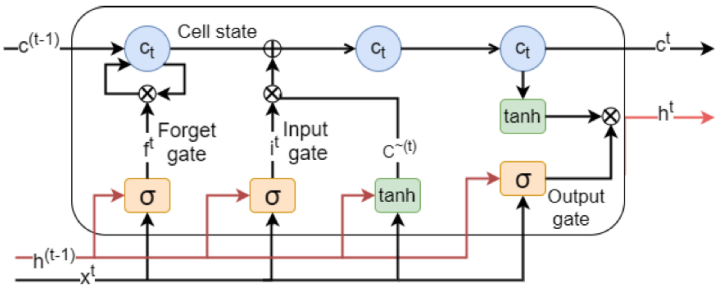


Fig. 6 Components of a Long Short Term Memory (LSTM) cell: forget gate, input gate and output gate[28].

These gates regulate the flow of information through the cell, deciding what to keep, update, or forget from the long-term state $C_{(t-1)}$ and the short-term state $h_{(t-1)}$. We can see that these are the inputs in the cells with $X_{(t)}$, where:

- $h_{(t-1)}$: Short-term state.
- $C_{(t-1)}$: Long-term state.
- $X_{(t)}$: Input data at time t.

C stands for cell and $C_{(t-1)}$ will store the most relevant information for a greater number of steps and $h_{(t-1)}$ will have in itself more immediate characteristics of the information contained in such step of the sequence, hence the names given to them. The vectors $h_{(t-1)}$ and $X_{(t)}$ will feed 4 fully connected layers, which will define in the context different gates, which will have the functions of learning what information to save, read and discard. These gates are the following:

2.2.1 Forget Gate

The Forget Gate determines which information from the previous short-term state $h_{(t-1)}$ and current input $X_{(t)}$ should be discarded. In this gate we have the calculation of the term $f_{(t)}$ as follows [20]:

$$f_{(t)} = \sigma(W_{xf}^T \cdot \mathbf{X}_{(t)} + W_{hf}^T \cdot \mathbf{h}_{(t-1)} + b_f) \tag{1}$$

Where we can see that the activation function is the logistic or sigmoid function, therefore we will have a restriction of the calculated values between 0 and 1. Thus, when we do the term-by-term multiplication with $C_{(t-1)}$ those values closer to zero will be erased and those closer to one will remain.

2.2.2 Input Gate

This is composed of a classic recurrent neuron where its output is given by:

$$g_{(t)} = \tanh(W_{xg}^T \cdot X_{(t)} + W_{hg}^T \cdot h_{(t-1)} + b_g) \tag{2}$$

This will process the information and obtain certain characteristics of the current stage of the sequence, which is contained between -1 and 1 thanks to the tanh function; in this way, it will allow us to have controlled values for the outputs. In addition to a layer whose output is [20]:

$$i_{(t)} = \sigma(W_{xi}^T \cdot X_{(t)} + W_{hi}^T \cdot h_{(t-1)} + b_i) \tag{3}$$

These will be joined by a term-by-term multiplication, which will allow, as in the forget gate, to filter the information according to its relevance. This gate will be in charge of controlling what information of the current sequence should be stored in the long-term memory term.

The Input Gate updates the long-term state with new information. It uses a combination of a tanh function and a sigmoid function to control what new information should be added to the long-term state. The updated information is then combined using element-wise multiplication to determine what should be added to the long-term state.

2.2.3 Output Gate

This will be in charge of defining what information will be saved for the short-term term for the next step in the sequence: $h_{(t-1)}$ and $X_{(t)}$ go through a logistic function, the result of which will be multiplied by the long-term term that is defined to exit this step of the process. We will thus have that the output term after the logistic function is given by:

$$o_{(t)} = \sigma(W_{xo}^T \cdot X_{(t)} + W_{ho}^T \cdot h_{(t-1)} + b_o) \tag{4}$$

Which will allow us to make the calculation that will give two of our outputs as follows:

$$h_{(t)} = o_{(t)} \otimes \tanh(C_{(t)}) \tag{5}$$

Where at the end of the sequence step we will have that the long-term term that carries our information for the longest time is defined as:

$$C_{(t)} = f_{(t)} \otimes C_{(t-1)} + i_{(t)} \otimes g_{(t)} \tag{6}$$

Where $\otimes$ denotes element-wise multiplication, and W_{xf}, W_{hf}, W_{xg}, W_{hg}, W_{xi}, W_{hi}, W_{xo}, and W_{ho} are the weight matrices.

LSTMs effectively tackle the vanishing gradient problem present in traditional RNNs by using their gating mechanisms to control the flow of information. They maintain a more stable gradient during training, enabling the network to learn long-term dependencies within the data making LSTMs particularly suited for tasks like time series forecasting, natural language processing, and other sequence-based tasks where understanding long-term dependencies is crucial.

By using the Forget Gate, the Input Gate, and the Output Gate, LSTMs selectively keep relevant information while discarding less important details, much like how the human brain simplifies and retains essential information while discarding less relevant details [27].

2.3 Quantum Machine Learning

Quantum Machine Learning (QML) is a research area and a computational paradigm that takes advantage of quantum Physics and Machine Learning (ML) through the use of quantum computers, with which it exploits the properties of superposition, quantum parallelism, entanglement and tunneling, to develop processes with high calculation, simplicity, fast application of algorithms, and lesser query complexity, in which performance and computational speed are significantly improved compared to classical computing [16], [15]. It is important to add that unlike the classical information that is stored in bits, quantum information processing uses superposition states of photons or atomic particles to process, store, and transmit data; this is summarized in figure 7. In this figure, quantum states of qubits are schematics drawn in a Bloch Sphere, which is represented by the basis states $|0\rangle$, $|1\rangle$ or a normalized complex linear superposition of the two [29]:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{7}$$

Here, α and β are complex numbers that satisfies $|\alpha|^2 + |\beta|^2 = 1$. This values, $|\alpha|^2$ and $|\beta|^2$ represents the probabilities of the system in the state $|0\rangle$ and $|1\rangle$, respectively. The qubit exists in a superposition of states $|0\rangle$ and $|1\rangle$ until it is observed [29]. In terms of the angles in the Bloch Sphere and using the normalization $|\alpha|^2 + |\beta|^2 = 1$, the equation 7 can be written as:

$$|\psi\rangle = \cos\frac{\theta}{2}\,|0\rangle + e^{i\phi}\sin\frac{\theta}{2}\,|1\rangle \tag{8}$$

The states contain information obtained from some physical process such as quantum sensing, quantum metrology [30], quantum networks [31], quantum control [32] or even quantum analog–digital transduction.

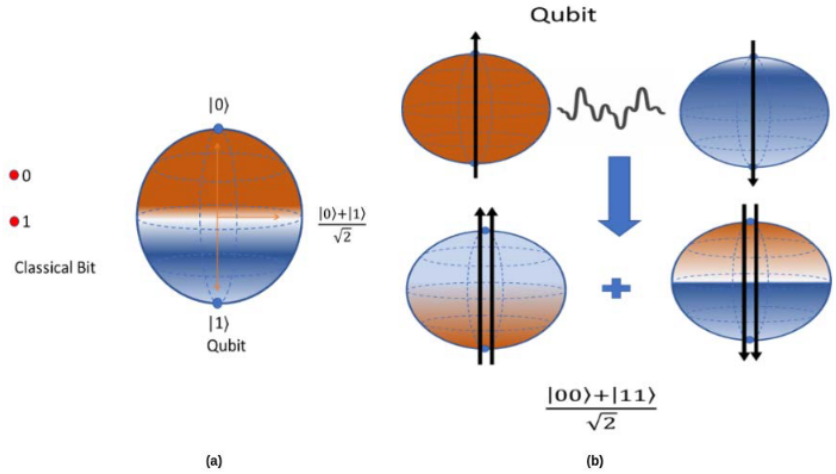


Fig. 7 (a) classical and a quantum state superposition(b) Entanglement state.[16].

Worth mentioning that classical bit-string can be easily encoded into n qubits, but the converse is not possible in a efficiently encoding way due to the fact that n-qubits system requieres $(2^n - 1)$ complex numbers to be specified. Figure 8 shows the different QML tasks depending on the kind of data and the algorithm to use. The QML forecasting used on this paper is based on classical data types using a quantum algorithm, aiming for one of the QML key applications: Classical data analysis, more precisely, optimization and learning enhancement of the learning rate of the algorithm used (specifically the hybrid classical-quantum algorithm QLSTM explained in subsection 2.5) in a supervised ML task [15]. QML models can be generalized into three components that are exposed in figure 9:

1. **Collection and preparation (encoding) of data - from source to quantum state:** The first step in this scope was the transformation of the classical data x (which represents the measurements of the meters in our time series for the forecasting) into qubits of information. This is done by encoding this classical data through an mapping embedding $x_j \rightarrow |\psi(x_j)\rangle$, with $|\psi(x_j)\rangle$ living in a Hilbert space $\mathcal{H}$. The encoding or mapping embedding used for this work was the amplitude encoding, which is explained in detail in section 3.
2. **The model, either quantum or hybrid quantum-classical**: In this work a variational quantum circuit is used. This is explained in detail in section 2.5.
3. **The optimization, activation and loss**: Recent works [12, 17, 33] have shown these steps performed in a classical device benefit the quantum components. This component is done by a variational algorithm. Variational algorithms represent a promising strategy in quantum computing by optimizing a set of classical parameters, θ, to minimize a given cost function, $C(\theta)$. In this approach, a classical computer interacts with a quantum machine to evaluate different values of θ, using a parameterized quantum circuit $U(\theta)$. This circuit prepares the quantum state $U(\theta)\,|0\rangle = |\psi(\theta)\rangle$, where measurements on the final state $|\psi(\theta)\rangle$

return estimates of expectation values or specific qubit states. These estimates are then used in the cost function $C(\theta)$, which defines the performance of the parameter set θ within the given problem. The objective is to find the optimal values of θ that minimize $C(\theta)$. One of the key advantages of this approach is that when the objective function $C(\theta)$ cannot be computed efficiently on classical computers—because simulating the quantum circuit $U(\theta)$ is computationally prohibitive—the variational algorithm can provide significant speedup, potentially exponential in some cases.

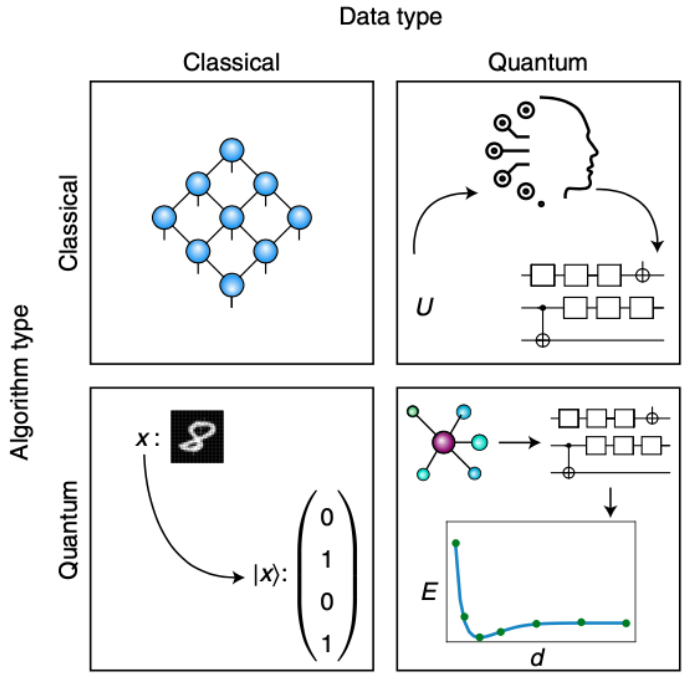


Fig. 8 QML tasks. This researching will be based on classical data type using a quantum algorithm type [15].

Several factors make variational circuits especially suitable for near-term quantum devices:

- **Adaptability**: Variational circuits can be designed with different implementations, tailored to specific problems. The depth of the circuit can be adjusted depending on the device's ability to handle noise.
- **Robustness**: The learning process in variational circuits can help mitigate systematic errors, enhancing the algorithm's robustness.

- **Noise resilience**: Due to the iterative nature of variational algorithms, noise does not pose as significant a challenge compared to other quantum algorithms. This makes them particularly promising for current, noisy intermediate-scale quantum (NISQ) devices.

In summary, variational algorithms leverage the strengths of quantum computing for optimization tasks, especially when classical methods are inefficient, making them a key candidate for exploiting the capabilities of near-term quantum technologies. In this work the QLSTM algorithm is used, an hybrid quantum-classical algorithm that uses VQC instead of the gates in a classical LSTM and also two activation functions. Section 2.5 explains this in detail.

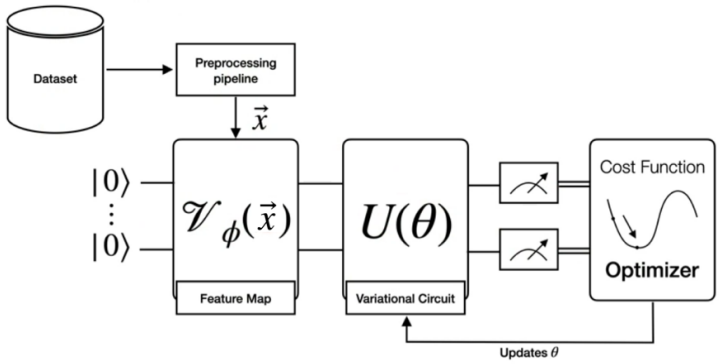


Fig. 9 QML models steps diagram [34].

According to [18], a second more ambitious definition of QML focuses on problems in which data is difficult to generate or store classically and refers to the use of a quantum device to classify or extract features from quantum states. The aim of QML, according to this definition, is to reduce the complexity in terms of either samples (sample complexity) or the number of operations needed in order to train the model and classify a test vector [18], something known as quantum supremacy.

2.4 A bit of physics and quantum computing: superposition, gates and entaglenment

In order to understand the architecture of the QLSTM used for the quantum-classical electrical meter forecasting we need the illustrate how the phenomenon of superposition described by equations 7 and 8 are getting in quantum computation, in which the quantum circuit contains wires and elementary quantum gates to carry around and manipulate quantum information [29]. In quantum circuits, these phenomena are manipulated using elementary quantum gates. Let's delve into some fundamental gates: the Hadamard gate, rotation gates R_y and R_z, and the CNOT gate, which is crucial for creating entanglement.

The Hadamard gate (H) is pivotal in quantum computing as it enables the creation of superposition states. When applied to a qubit in the computational basis states $|0\rangle$

or $|1\rangle$, the Hadamard gate transforms them into a superposition state (using Dirac's notation):

$$H|0\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}, \quad H|1\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}. \tag{9}$$

Mathematically, the Hadamard gate is represented by the following matrix:

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \tag{10}$$

Applying the Hadamard gate to a qubit in state $|0\rangle$:

$$H|0\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \frac{|0\rangle + |1\rangle}{\sqrt{2}}. \tag{11}$$

This transformation puts the qubit into an equal superposition of $|0\rangle$ and $|1\rangle$. The probability of measuring the qubit in either state is $\frac{1}{2}$.

Quantum gates can also rotate qubits around different axes on the Bloch sphere, crucial for more complex state manipulations. The R_y and R_z gates represent rotations around the Y and Z axes, respectively.

R_y Gate:

The R_y gate rotates the qubit around the Y-axis by an angle θ:

$$R_y(\theta) = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{bmatrix}. \tag{12}$$

When applied to a qubit $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, it modifies the probability amplitudes, influencing the qubit's superposition state.

R_z Gate:

The R_z gate rotates the qubit around the Z-axis by an angle ϕ:

$$R_z(\phi) = \begin{bmatrix} e^{-i\phi/2} & 0 \\ 0 & e^{i\phi/2} \end{bmatrix}. \tag{13}$$

This gate affects the phase of the qubit's state without altering the probabilities of measuring $|0\rangle$ or $|1\rangle$.

The CNOT (Controlled-NOT) gate is a two-qubit gate essential for creating entanglement. It flips the state of the second qubit (target) if the first qubit (control) is $|1\rangle$. Its matrix representation is:

$$\text{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{14}$$

Consider the initial state of two qubits, $|0\rangle \otimes |0\rangle$ (i.e., $|00\rangle$). Applying a Hadamard gate to the first qubit creates a superposition:

$$H|0\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}. \tag{15}$$

Now, the overall state is:

$$\frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes |0\rangle = \frac{|00\rangle + |10\rangle}{\sqrt{2}}. \tag{16}$$

Applying the CNOT gate on this state:

$$\text{CNOT}\left(\frac{|00\rangle + |10\rangle}{\sqrt{2}}\right) = \frac{|00\rangle + |11\rangle}{\sqrt{2}} = |\beta_{00}\rangle\,. \tag{17}$$

The resulting state $\frac{|00\rangle+|11\rangle}{\sqrt{2}}$ is an entangled state, known as the Bell state $|\beta_{00}\rangle$ [29], where the measurement of one qubit instantaneously determines the state of the other, no matter the distance between them. This phenomenon is at the heart of quantum computing's power and underlies many quantum algorithms. Table 2 contains a summary of these gates.

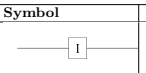
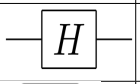
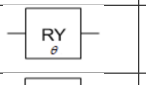
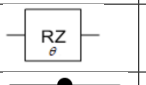
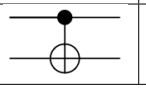
Name	Symbol	Matrix Representation	Diagonal Representation	Description
Identity (I)		$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	$\|0\rangle\langle 0\| + \|1\rangle\langle 1\|$	Leaves the qubit state unchanged.
Hadamard (H)		$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$	$\frac{1}{\sqrt{2}}(\|0\rangle\langle 0\| + \|0\rangle\langle 1\| + \|1\rangle\langle 0\| - \|1\rangle\langle 1\|)$	Creates a superposition state.
$R_y(\theta)$		$\begin{bmatrix} \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{bmatrix}$	Complex combination in terms of kets and bras	Rotates the qubit around the Y-axis by θ.
$R_z(\phi)$		$\begin{bmatrix} e^{-i\phi/2} & 0 \\ 0 & e^{i\phi/2} \end{bmatrix}$	$e^{-i\phi/2}\|0\rangle\langle 0\| + e^{i\phi/2}\|1\rangle\langle 1\|$	Rotates the qubit around the Z-axis by ϕ. Special case: Pauli-Z when $\phi = \pi$.
CNOT		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$	Complex multi-qubit state representation	Flips the target qubit if the control qubit is $\|1\rangle$. Creates entanglement.

Table 2 Quantum Gates: Summary of the gates used in the VQC architecture for the QLSTM.

2.5 QLSTM and VQC architecture

The quantum-hybrid model, that is the counterpart of the LSTM, used in this work is the Quantum Long Short Term Memory proposed by Samuel Yen-chi et. al in [17] but following the same architecture by Yunjun Yu et. al in [12]. The infrastructure is shown in the figure 10. This infrastructure is the same as exposed in figure 6 with the difference that each gate or classical recurrent neural network cell is replaced by a VQC (Variational Quantum Circuit).

VQC as a component of the variational algorithm used in this work, the QLSTM, is a quantum circuit that includes adjustable parameters subject to iterative optimizations [17]. The circuit itself evolves a quantum state, and the parameters typically

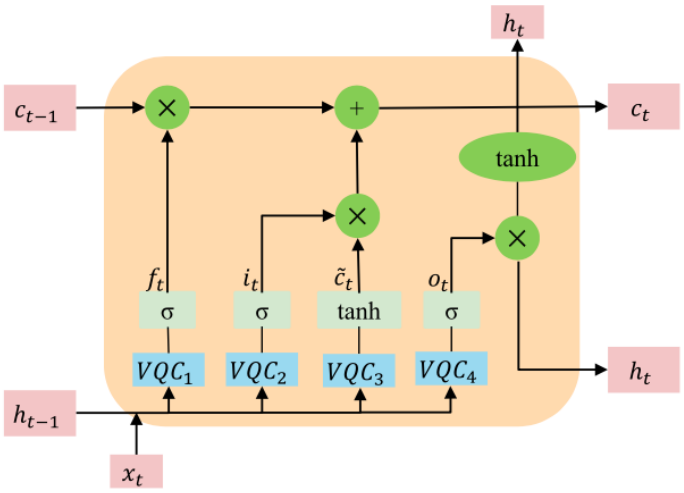


Fig. 10 QLSTM architecture. Similar to the architecture of a classical LSTM illustrated in 6 Functions σ and tanh represents the sigmoid and hyperbolic tangent activation function. x_t is the input at time t, h_t is the hidden state, and c_t stands for the cell state. Here the VQC circuits are shown for: VQC_1 the forget gate, VQC_2 the input gate, VQC_3 the update gate and VQC_4 the output gate. $\otimes$ and $\oplus$ represent element-wise multiplication and addition, respectively [12].

control aspects like gate angles or other tunable elements in the quantum circuit. As it is illustrated in figure 1, the VQC consists of three parts: data encoding layer R_x, R_y and/or R_z gates, variational layer $V(\theta)$ with *learneable* parameters θ that are optimized through gradient methods and a quantum measurement layer that could be done partially (a subset of qubits) or with all the qubits to retrieve a classical output. The VQC used for this work is described in figure 11 and was proposed in [17].

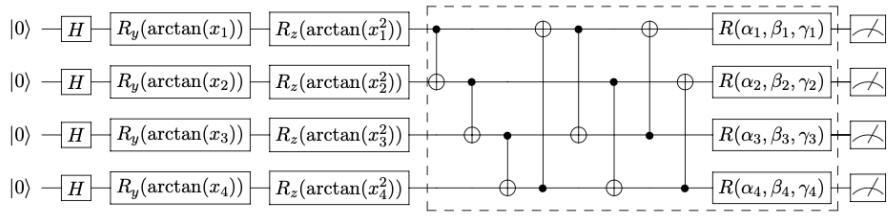


Fig. 11 VQC architecture proposed by [17] for 4 qubits ($N = 2^4$). This architecture consists of three key layers: the data encoding layer, which uses Hadamard (H), R_y, and R_z gates to encode classical data into a quantum state; the variational layer (denoted by a dashed box), where the tunable parameters are optimized to solve a specific problem; and the quantum measurement layer, which extracts relevant information from the quantum system. Both the number of qubits and the measurements can be adjusted according to the problem.

2.5.1 Data Encoding Layer

As it is explained in 3.1, a quantum embedding or encoding represents classical data as quantum states in a Hilbert space [12]. In this paper amplitude encoding was used. In this technique, classical data is encoded directly into the amplitudes of a quantum state. Specifically, a normalized N-dimensional data set x, $N = 2^n$ is represented by the amplitudes of an n-qubit quantum state $|\psi_x\rangle$, as $|\psi_x\rangle = \sum_{i=1}^{N} x_i |i\rangle$, where x_i is the ith element of x and $|i\rangle$ is the ith computational basis state.The quantum state then reflects the structure of the classical data. The classical input vector will be transformed into rotation angles to guide the single-qubit rotations [17]. The initial state $|0\rangle \otimes \cdots \otimes |0\rangle$ is transformed into the state:

$$\begin{aligned}(H|0\rangle)^{\otimes n} &= \frac{1}{\sqrt{2^n}}(|0\rangle + |1\rangle)^{\otimes n} \\ &= \frac{1}{\sqrt{2^n}}(|0\rangle \otimes \cdots \otimes |0\rangle + \cdots + |1\rangle \otimes \cdots \otimes |1\rangle) \\ &= \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} |i\rangle\,, \end{aligned} \tag{18}$$

$|i\rangle$ is the ith computational basis state and the running index i is the decimal number for the corresponding bit string. In this approach, 2N rotation angles are generated from the N-dimensional input vector $\vec{v} = (x_1, x_2, \ldots, x_N)$ using:

$$\theta_{i,1} = \arctan(x_i), \quad \theta_{i,2} = \arctan(x_i^2) \quad \text{for each } x_i. \tag{19}$$

The angle $\theta_{i,1}$ is applied using the $R_y(\theta_{i,1})$ gate, and $\theta_{i,2}$ is applied with the $R_z(\theta_{i,2})$ gate. Arctan function is chosen because the input values are from $\mathbb{R}$, not limited to $[-1, 1]$. The resulting quantum state, corresponding to the classical input vector $X_{(t)}$, is prepared for subsequent layers, with the $2N$ rotation angles serving solely for state preparation and not subject to iterative optimization [17].

2.5.2 Variational Layer

The encoded classical data, now represented as a quantum state, undergoes a series of unitary operations. These operations involve multiple CNOT gates and single-qubit rotation gates, as illustrated in the dashed box of 11. CNOT gates are applied to adjacent qubit pairs in a cyclic manner to create multi-qubit entanglement. The rotation angles $\{\alpha_i, \beta_i, \gamma_i\}$ associated with the single-qubit rotation gates $R_i = R(\alpha_i, \beta_i, \gamma_i)$ are not predetermined; instead, they are adjusted during the iterative optimization process using the gradient descent method. To enhance the layer's depth and increase the number of variational parameters, the dashed box can be repeated multiple times, though this study fixes the depth at 2 for all experiments [17].

2.5.3 Quantum Measurement Layer

At the conclusion of each variational quantum circuit (VQC) block, a quantum measurement layer is implemented, where the expectation values of each qubit are calculated using the Pauli matrix σ_Z, as shown in equation 20. This work was done using PennyLane, the quantum simulation software from Xanadu, in which these values were calculated numerically on a classical computer. In contrast, when using real quantum computers, the expectation values are estimated statistically through repeated measurements (shots), which should theoretically align with the simulated values in a zero-noise scenario. Following a series of quantum entanglement operations, the resulting output is a fixed-length vector that will be further processed on a classical computer. In the proposed QLSTM architecture, the measured values from each VQC will be integrated and processed within a QLSTM cell.

$$\sigma_Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{20}$$

In a QLSTM unit, there are four variational quantum circuits (VQCs), as illustrated in figure 10. The input to each VQC is the concatenation of x_t from the current timestep and h_{t-1} from the previous timestep. The output is a vector obtained from the quantum measurement layer, where the measured values are the Pauli Z expectation values of each qubit. These values then pass through nonlinear activation functions, such as sigmoid $\sigma(x)$ and $\tanh(x)$. All four VQCs are defined based on a basic VQC, as shown in figure 11. Each VQC includes a trainable fully connected layer that reduces the dimensionality of the information to match the circuit size. After each VQC, a quantum measurement of the expectation for each qubit is fed into a trainable fully connected expansion layer. This expansion layer increases the size back to the required dimensional space of the classical network, which is subsequently processed using the classical LSTM method explained in section 2.2.

The QLSTM architecture can be expressed with the following equations:

$$f_t = \sigma(VQC1(v_t)) \tag{4}$$

$$i_t = \sigma(VQC2(v_t)) \tag{5}$$

$$\tilde{c_t} = \tanh(VQC3(v_t)) \tag{6}$$

$$c_t = f_t \times c_{t-1} + i_t \times c_t \tag{7}$$

$$o_t = \sigma(VQC4(v_t)) \tag{8}$$

$$h_t = o_t \times \tanh(c_t) \tag{9}$$

This structured approach integrates quantum mechanics with classical LSTM methodologies, enhancing the capabilities of the model in terms of learning, as is explained in detail in section 5, in which with less epochs (precisely the half of the

epochs that the classical LSTM used to train the model) the model that worked with QLSTM learns faster in terms of the MSE functional in comparison with its classical counterpart, the classical LSTM, getting similar values in the MSE and MAPE metrics.

2.6 Optimization: proof of the parameter shift rule in VQC

Similar to the work done by Samuel Yen-chi *et. al* [17], for the optimization procedure, the parameter-shift method to compute the analytical gradient of quantum circuits was used. For instance, given the expectation value of an observable B, the function $f(x;\theta_i)$, where x is the input value and $U_0(x)$ is the state preparation routine that encodes x into the quantum state, is written as:

$$f(x;\theta_i) = \langle 0|U_0^\dagger(x)U_i^\dagger(\theta_i)\hat{B}U_i(\theta_i)U_0(x)|0\rangle = \langle x|U_i^\dagger(\theta_i)\hat{B}U_i(\theta_i)|x\rangle, \tag{21}$$

where i is the index of the circuit parameter for which the gradient is being calculated, and $U_i(\theta_i)$ represents the single-qubit rotation generated by the Pauli operators. The gradient of f with respect to the parameter θ_i is given by [19]:

$$\nabla_{\theta_i} f(x;\theta_i) = \frac{1}{2}\left[f\left(x;\theta_i + \frac{\pi}{2}\right) - f\left(x;\theta_i - \frac{\pi}{2}\right)\right]. \tag{22}$$

This method allows us to analytically compute the gradients of expectation values and apply gradient descent optimization, commonly used in classical machine learning, to variational quantum circuit-based models. It is important for this work to prove equation 22.

In order to evaluate the derivative of an expectation by measuring an overlap of two quantum state, authors as Mitariai *et. al* in [19] analized gates of the form $\mathcal{G} = e^{-i\mu\sigma}$, where σ is the tensor product of the Pauli operators. The derivative is computed by what we will call the "parameter shift rule". The first goal of the authors in [19], and the proof shown here, is to expand the parameter shift rule for any gate of the form $\mathcal{G} = e^{-i\mu G}$ where G is an Hermitian generator with at most two distinct eigenvalues. The gradients of expectation values of quantum measurements can be estimated using the same architecture that executes the original circuit.

As it is shown in the figure 12, in the context of hybrid optimization that we have, it is a quantum algorithm that consists of a gate $U(\theta)$ that depends on a set θ of m real gate parameters followed by a measurement of an observable $\hat{B}$. $U(\theta)$ generally consists of an architecture that is repeated K times, where K is a hyperparameter of the computation [19]. So, the expectation value of $\hat{B}$ is given by:

$$f(\theta) = \langle \hat{B} \rangle = \langle 0|\, U^\dagger(\theta)\hat{B}U(\theta)\,|0\rangle\,, \tag{23}$$

where the quantum device runs the quantum algorithm several times and averages measurement results to estimate $f(\theta)$, for that very reason is called expectation value. In order to get the expansion of the parameter shift rule for a hermitian operator we have to calculate the partial derivative $\partial_u f(\theta)$ where $\mu \in \theta$ is one of the gate parameters. The gradient ∇f is formed by the partial derivatives with respect to all the parameters.

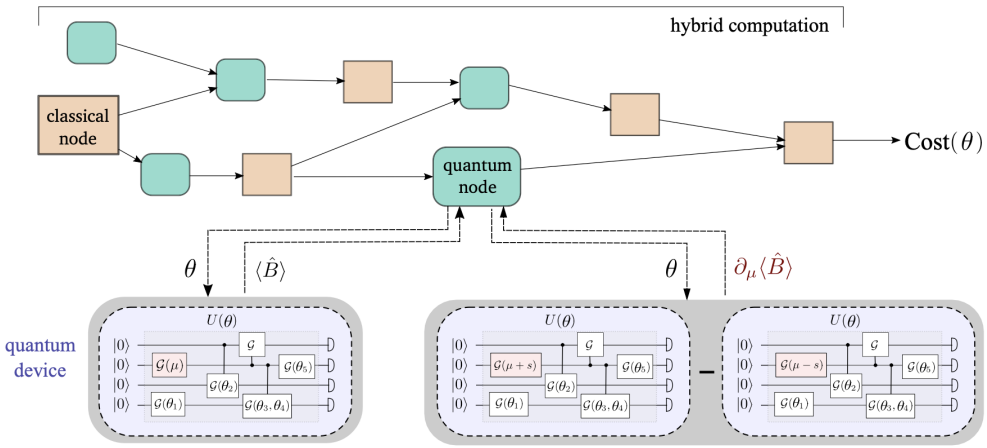


Fig. 12 Parameter shift rule in the scope of hybrid optimization. In the quantum node the variational quantum algorithm is executed and it computes the derivatives of its outputs with respect to the gate parameters by running the original circuit twice but with a shift in the parameter in question [19].

In relation with the computation of the gradients of quantum expectation values, the authors derive an equation for $\partial_u f(\theta)$, where the constituent parts can be evaluated on a quantum computer and then combined on a classical co-processor.

Now, taking into account that the parameter $\mu \in \theta$ only affects a single gate $\mathcal{G}(\mu)$ in the sequence, we have that $U(\theta) = V\mathcal{G}(\mu)W$ then we have that equation 23 turns out in:

$$\begin{aligned} f(\theta) = \langle \hat{B} \rangle &= \langle 0| U^\dagger(\theta)\hat{B}U(\theta) |0\rangle \;\; f(x) \\ &= \langle 0| W^\dagger \mathcal{G}^\dagger(\mu) V^\dagger \hat{B} V \mathcal{G}(\mu) W |0\rangle \\ &= \langle \Psi| \mathcal{G}^\dagger(\mu) V^\dagger \hat{B} V \mathcal{G}(\mu) |\Psi\rangle , \end{aligned}$$

with $Q = V^\dagger \hat{B} V$ we have:

$$f(\theta) = \langle \Psi| \mathcal{G}^\dagger(\mu) Q \mathcal{G}(\mu) |\Psi\rangle . \tag{24}$$

Now, calculating $\partial_u f(\theta)$ from eq. 24 we have that

$$\partial_u f(\theta) = \langle \Psi| \partial_u(\mathcal{G}^\dagger(\mu)) Q \mathcal{G}(\mu) |\Psi\rangle + \langle \Psi| \mathcal{G}^\dagger(\mu) Q \partial_u(\mathcal{G}(\mu)) |\Psi\rangle \tag{25}$$

$$= H.C + \langle \Psi| \mathcal{G}^\dagger(\mu) Q \partial_u(\mathcal{G}(\mu)) |\Psi\rangle , \tag{26}$$

in which H.C stands for Hermitian conjugate of $\mathcal{G}^\dagger(\mu) Q \partial_u(\mathcal{G}(\mu))$, as we can see that is the first term of equation 25. Now as $\mathcal{G}(\mu)$ can be generated by an hermitian

operator, we have that, $\mathcal{G}(\mu) = e^{-i\mu G}$ and $\partial_\mu \mathcal{G} = -iGe^{-i\mu G}$, then replacing in equation 26 we have:

$$\partial_\mu f = \langle\Psi|\, \mathcal{G}^\dagger(\mu) Q(-iGe^{-i\mu G})\, |\Psi\rangle + H.C \tag{27}$$
$$= \langle\Psi|\, \mathcal{G}^\dagger(\mu) Q(-iG\mathcal{G}(\mu))\, |\Psi\rangle + H.C \tag{28}$$
$$= \langle\Psi'|\, -iQG\, |\Psi'\rangle + H.C. \tag{29}$$

In the previous line it is used $|\Psi'\rangle = \mathcal{G}\,|\Psi\rangle$. Now, by $rr^{-1} = 1$, these being the eigenvalues of G and replacing this in equation 29, we have:

$$\partial_\mu f = \langle\Psi'|\, Q(-irr^{-1}G)\, |\Psi'\rangle + H.C \tag{30}$$
$$= r\,\langle\Psi'|\, Q(-ir^{-1}G)\, |\Psi'\rangle + H.C. \tag{31}$$

Now with $\hat{B} = \mathbb{1}$ and $\hat{C} = -ir^{-1}G$ and using equation 4 from [19], which is straightforward to proof this equation, we have that:

$$\partial_\mu f = \frac{r}{2}[\langle\Psi'|\, (\mathbb{1} - ir^{-1}G^\dagger) Q(\mathbb{1} - ir^{-1}G)\, |\Psi'\rangle - \langle\Psi'|\, (\mathbb{1} + ir^{-1}G^\dagger) Q(\mathbb{1} + ir^{-1}G)\, |\Psi'\rangle]. \tag{32}$$

Now, taking into account the Taylor expansion from [19], i.e., $\mathcal{G}(\frac{\pi}{4r})(\mathbb{1} - ir^{-1}G)$ and then $\mathcal{G}(-\frac{\pi}{4r})(\mathbb{1} + ir^{-1}G)$ because $\sin(r\mu)$ is an odd function. Then equation 32 turns out in:

$$\partial_\mu f = \frac{r}{\cancel{2}}[\cancel{2}\,\langle\Psi|\, \mathcal{G}^\dagger(\mu)\mathcal{G}^\dagger(\frac{\pi}{4r})\hat{Q}\mathcal{G}(\frac{\pi}{4r})\mathcal{G}(\mu)\, |\Psi\rangle - \langle\Psi|\, \mathcal{G}^\dagger(\mu)\mathcal{G}^\dagger(-\frac{\pi}{4r})\hat{Q}\mathcal{G}(-\frac{\pi}{4r})\mathcal{G}(\mu)\, |\Psi\rangle] \tag{33}$$

Now, taking into account that for unitary operators we have that $\mathcal{G}(a)\mathcal{G}(b) = \mathcal{G}(a+b)$, then we have in the equation 33 with $s = \frac{\pi}{4r}$:

$$\partial_\mu f = r[\langle\Psi|\, \mathcal{G}^\dagger(\mu + s)\hat{Q}\mathcal{G}(\mu + s)\, |\Psi\rangle - \langle\Psi|\, \mathcal{G}^\dagger(\mu - s)\hat{Q}\mathcal{G}(\mu - s)\, |\Psi\rangle], \tag{34}$$

and then finally we have that

$$\partial_\mu f = r[f(\mu + s) - f(\mu - s)]. \tag{35}$$

Last but not least, equation 35 means that the derivative is obtained using the product rule by shifting the parameter in each gate separately and summing the result up. For G being a one-qubit rotation generation then $r = \frac{1}{2}$, $s = \frac{\pi}{2}$ and $\mu_i = \theta_i$ finally getting the equation 22.

3 Elements

3.1 Feature selection: Encoding process.

The first step in our QML forecasting is the step of the encoding. Depending on the nature of the data, we have to decide wether to encode our classical data into quantum bits (Qubits) in the two following forms:

1. **Bit Encoding.** With $[v_i]$ representing the bit-string vector of the training vector this bit vector can be accessed on demand through a self inverse quantum oracle of the form [18]:

$$O_{\text{data}} |i\rangle |b\rangle = |i\rangle |b \oplus [v_i]\rangle ,$$

known as the coherent bit-oracle [18]. With this sort, a superposition of queries on the training data of the form

$$O_{\text{data}} \left(\sum_x a_x |x\rangle |0\rangle \right) = \sum_x a_x |x\rangle |[v_i]\rangle ,$$

which allows the achievement of quantum speedup over the size of the data [18]. However, being this one the sort of encoding the most similar manner as the data is provided to most of the classical machine learning algorithm, the number of qubits necessary to store the information can be prohibited, i.e, for MNIST data set, a quantum computer with a minimum of 784 qubits is necessary [18].

2. **Amplitude Encoding.** Amplitude encoding is a method used in quantum machine learning to efficiently represent classical data in a quantum state. In this scheme, the data, i.e., the vectors of the training set, are embedded as unit vectors in the quantum computer [35, 36] from feature vectors of dimension up to $N = 2^n$ which allow for the encoding of an exponentially large amount of information using the relatively small number of n qubits. With $\{v_i : i = 1, ..., N\}$ and $v_i = \sum_j v_{ij} |j\rangle$, the encoding is then $|v_i\rangle = \sum_j v_{ij}/|v_i| |j\rangle$. There are two main classes of amplitude encoding: coherent and incoherent encoding. In this paper, the incoherent version is used since it is the most commonly used in Quantum Neural Networks (QNN)[18]. This method is particularly useful in quantum machine learning as it enables the representation of high-dimensional data in a compact quantum form, leveraging quantum parallelism. Operations on these encoded states, such as quantum gates or variational quantum circuits, can process and manipulate the data more efficiently than classical methods, especially for tasks involving large datasets. However, preparing the quantum state with precise amplitudes (state preparation) is a non-trivial task and can be resource-intensive [18, 35].

3.2 Metrics: Model performance.

As it was exposed in section 2.1 there are two sorts of metrics: one is the variability metric, such as R^2 and the other the bias metric such as MAPE. This two metrics can be related with the accuracy and precision, in which accuracy relates to bias, which can be reduced by using only local information, but this can lead to underfitting. This

means that while the model may seem accurate with training data (low bias), it may not perform well with new test data, indicating high variance. Precision, on the other hand, is connected to the variance, and improving it requires gathering data from a larger sample space. The ideal scenario combines both accuracy and precision by using broad and representative data [26].

- **Mean Squared Error (MSE functional)**: Measures the average squared difference between the true values y_i and the predicted values $\hat{y}_i$:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{36}$$

- **R-squared (R^2)**: Indicates how well the predicted values explain the variability of the true values:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{37}$$

- **Mean Absolute Percentage Error (MAPE)**: Represents the percentage difference between the true and predicted values:

$$\text{MAPE} = \frac{100}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \tag{38}$$

4 Data description, preprocessing and feature engineering

Empresas Públicas de Medellín E.S.P., known by its acronym EPM, a Colombian public utility company provided us with a database of a total of 2,765,004 registers divided in 1,975 total clients across Antioquia department in 1,876 locations as it is shown in figure 13. These clients were divided in different categories such as *RESIDENCIAL* (residential), *COMERCIAL* (commercial), *INDUSTRIAL* (industries), *OFICIAL* (official), *ESPECIAL* (special) and *AUTOCONSUMOS EPM* (self-consumption EPM). Table 3 summarizes the features (columns) of the dataset. For each of the 1,975 clients the electrical meter records, in terms of the variable *ACTIVA_IMP* goes from June 06th, 2023 to July 31st, 2023. It was found that the 1.9% of the values for *ACTIVA_IMP* were NaN or missing values, for a total of 54,100 registers in this variable. This variable refers to the electrical energy consumed by a system or installation, measured in kilowatt-hours (kWh). It is the total energy received from the grid (imported) over a period of time. Active energy is the energy that actually performs useful work, such as running machinery, lighting, heating, etc. The term "imported" in contrast from "exported" active energy, refers to the energy generated by a user and sent back to the grid (such as from solar panels) [37]. Is the chosen variable for the forecasting due to the business importance. Figure 14 shows the hourly distribution of *ACTIVA_IMP* (KWh) for each category.

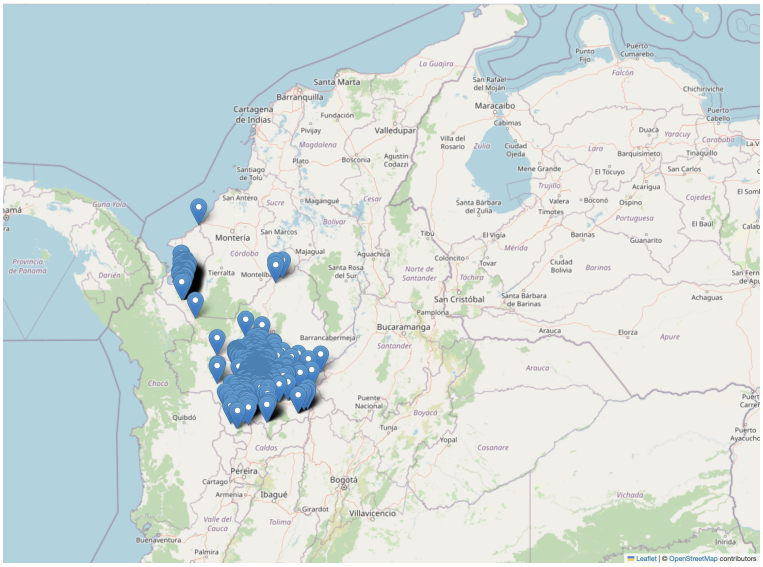


Fig. 13 EPM data clients distribution in Colombia. This data was taken from the dataset provided by EPM, specifically taking the unique values of X and Y of the feature/columns of the dataset, and located using *Folium* library in Python.

There are two forecasting approaches for LSTM and QLSTM models: the first only takes into account the $ACTIVA\_IMP$ (the forecasting variable y) in terms of the hours, and the second one takes the $ACTIVA\_IMP$ forecasting variable in terms of the most correlated variables. For the latter, it was important to choose the most correlated variables and also important variables based on the understanding of the electrical business that has influence in the forecasting variable. In this way, the most correlated variables were found by a correlation map described in figure 15 which finds the linear correlation between variables based on the Pearson correlation coefficient equation

$$r_{xy} = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \sum (y_i - \bar{y})^2}} \tag{39}$$

Where x_i and y_i are the individual data points of variables x and y, $\bar{x}$ and $\bar{y}$ are the means of x and y, the numerator represents the covariance of x and y and the denominator is the product of the standard deviations of x and y. It ranges from -1 to 1, where $r = 1$ indicates a perfect positive correlation, $r = -1$ a perfect negative correlation, and $r = 0$ no linear correlation.

From this correlation map we defined the most correlated variablesto be: $V1$, $V2$, $V3$, $I1$, $I2$, $I3$, $REACTIVA\_EXP$ and $HORA$. Of these, $HORA$ is considered an exogenous variable because it is an external factor that influences the electrical system but is not influenced by it. The remaining variables are considered endogenous, as they are part of the electrical system that generates $ACTIVA\_IMP$. These variables were used for the approach described in sections 5.5 and 5.6. The best hyperparameters for the LSTM and QLSTM models were selected using time series cross-validation with 3 folds, following multiple train-test splits: 60% train-20% test, 70% train-20% test and

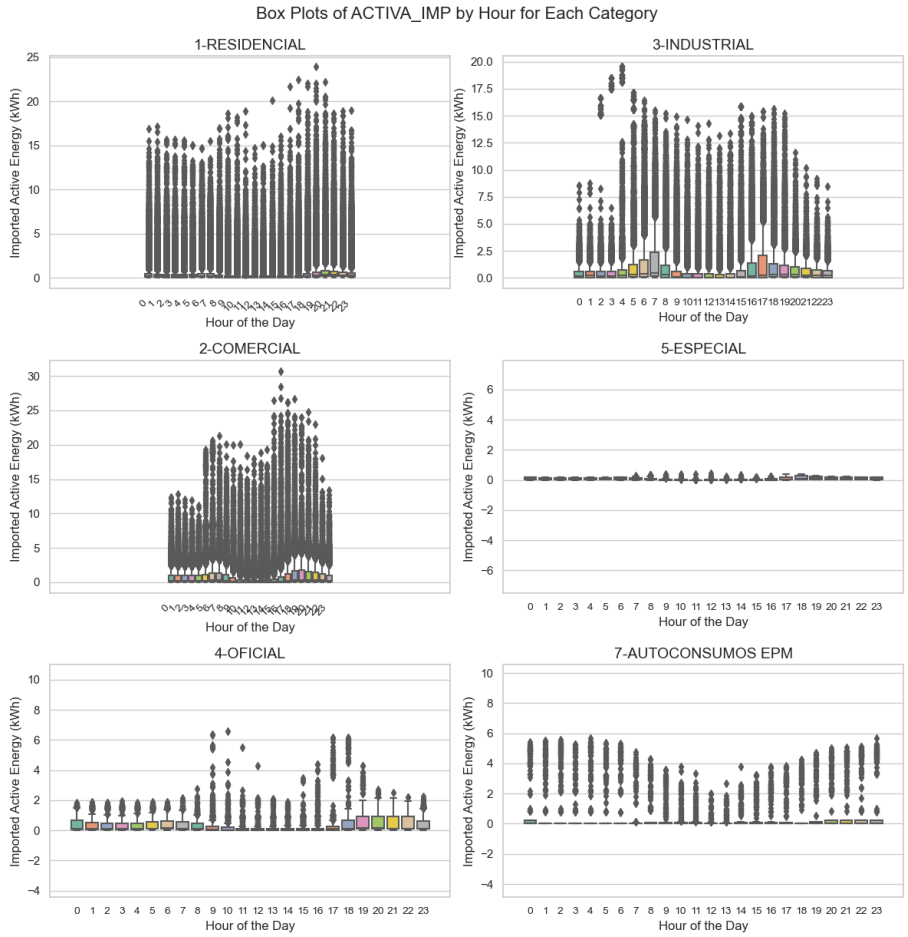


Fig. 14 Hourly distribution for *ACTIVA_IMP* in each category.

80% train-20% test. After this, a backstepwise selection process was implemented, but the best model performance was achieved using all the previously mentioned variables.

The dataframe was filtered taken the registers of the clients for the last 16 days or 384 hours. It was found that for the majority of clients, this action resulted in 408 hours registered for most of the 1,975 clients with 37 clients with counts less than 408 hours. So the dataframe was filtered to contain clients with 408 hours registered leaving us with 1,938 clients. This dataframe was filtered by column $SERIE$ and a new column *Consumption* that stores the values of the *ACTIVA_IMP* as a numpy array or tensor $(1938, 408, 1)$ this aiming to be used by client, category or subcategory.

For each approach that is described in section 5, each dataframe used (depending on the forecasting scope) had a preprocessing of the data in terms of the normalization using standard scaler, this for both predictions: LSTM and QLSTM forecasting. This preprocessing technique removes the mean and scales the variables to unit variance

Variable	Description	Values
SERIE	Unique identifier for the electrical meter or client.	Numeric (e.g., 18741849)
FECHA	Date of the measurement. It goes from 2023-06-01 to 2023-07-31	YYYY-MM-DD
HORA	Time of the measurement. It goes from 0 to 23 hour of the day	0,1, 2...23
SERVICIO_SUSCRITO	Subscribed service, identified by a numerical code (e.g., Residential, Industrial)	Numerical (e.g., 130820443 is for Categoria Industrial with subcategoría 11-220 Voltios)
ACTIVA_IMP	Imported active energy (kWh)	Numeric (e.g., 0.33)
REACTIVA_IMP	Imported reactive energy (kVARh)	Numeric (e.g., 0.04)
ACTIVA_EXP	Exported active energy (kWh)	Numeric (e.g., 0.53)
REACTIVA_EXP	Exported reactive energy (kVARh)	Numeric (e.g., 0.75)
V1, V2, V3	Voltage phase 1, 2 or 3 (V)	Numeric (e.g., 230.5)
I1, I2, I3	Current phase 1, 2 or 3 (A)	Numeric (e.g., 15.2)
ALARM_NAME	Alarm event triggered during the measurement	Alphanumeric (e.g., Overload)
CATEGORIA	Category of the measurement	1-RESIDENCIAL, 2-COMERCIAL, 3-INDUSTRIAL, 4-OFICIAL, 5-ESPECIAL, 7-AUTOCONSUMOS EPM
SUBCATEGORIA	Subcategory of the measurement	1-ESTRATO 1, 2-ESTRATO 2, 3-ESTRATO 3, 4-ESTRATO 4, 5-ESTRATO 5, 6-ESTRATO 6, 11-220 Voltios, 12-13200 Voltios
X, Y	Longitude and latitude coordinate respectively	Numeric (e.g., -74.00597, 40.71278)

Table 3 Description of Dataset Variables.

[25, 26]. Machine learning models are sensitive to the different scales of the variables, so it is important to put all the variables on the same scale. Aiming this, also the variables used for each model and scope, described in section 5, were scaled furthermore with Min-Max method and converted in the range of $[0, 1]$ using

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}. \tag{40}$$

5 Results and Discussions

This section contains the results obtained in this work. The first subsection shows an approach of time series clustering, as the first forecasting attempt, to show the viability of this approach and the periodicity and patterns of the clusters. The next subsections have the explanations of the two approaches developed for the LSTM and QLSTM showing the insights, results and conclusions of the models.

5.1 Clustering approach

Before showing the results of this work, an attempt to construct a forecasting was made using a clustering technique. To better understand the periodicity of the data, firstly the number of clusters was found using the elbow method and silhouette scores [26]. As it is shown in figure 16, the best fit of number of cluster are 2 or 3. Three clusters were chosen and time series clustering (specifically TimeSeriesKMeans) using the *tslearn* library with Soft Dynamic Time Warping (SoftDTW) as the distance metric was done. *PySpark* libraries were used for parallelization by the distributed computing power that this library offered. Also TimeSeriesKMeans from tslearn cluster scaled the time series data using all available CPU cores. The SoftDTW metric is particularly useful

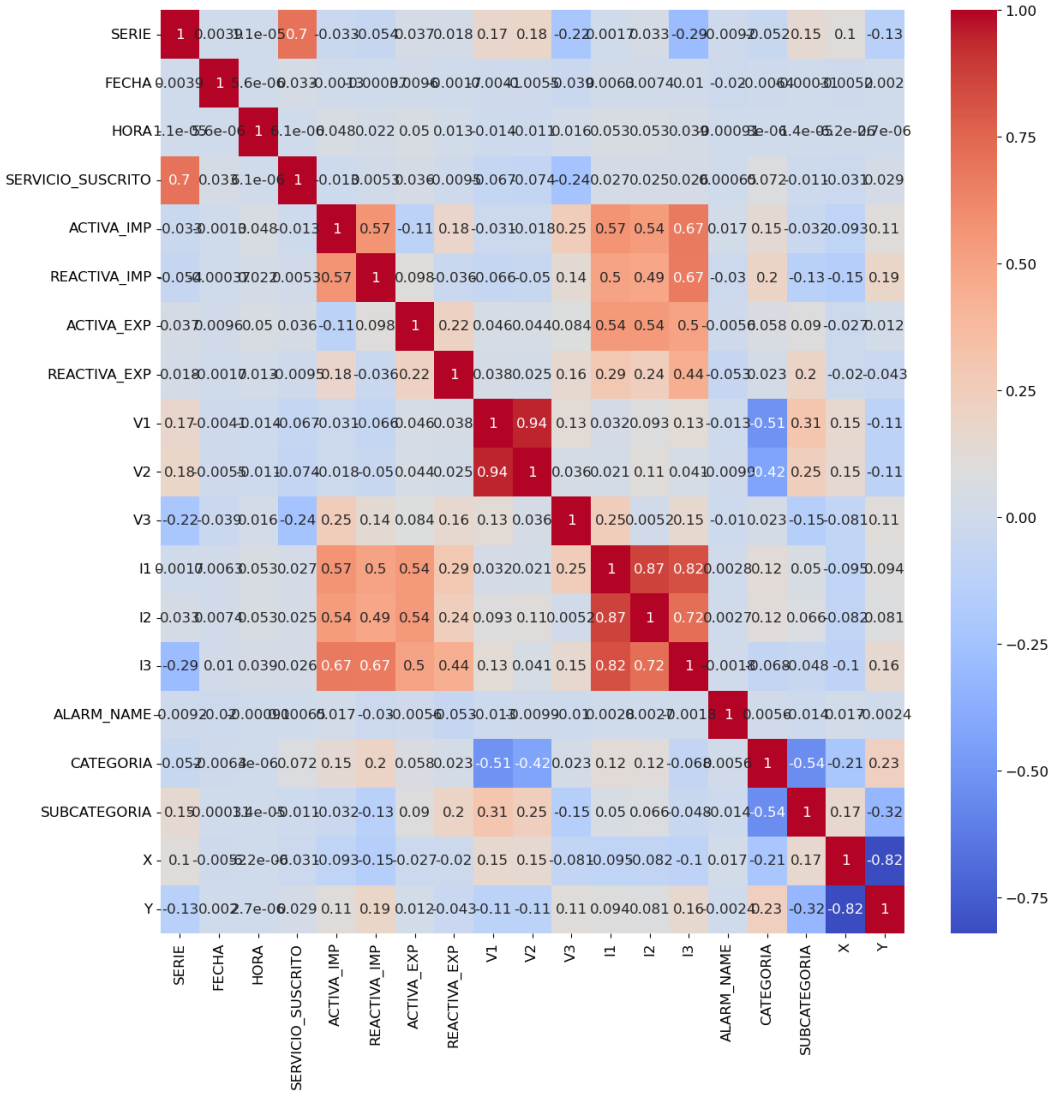


Fig. 15 Correlation between variables of the dataset provided by EPM.

for handling non-linear alignments in time series data [26]. The results in figure 17 show the prediction of the *ACTIVA_IMP* for the last 200 hours.

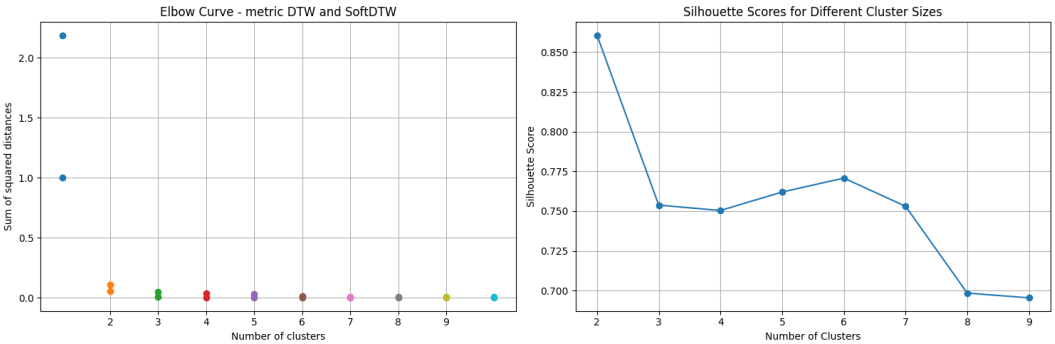


Fig. 16 Elbow method and silohouette scores showing that the best cluster numbers are 2 and 3.

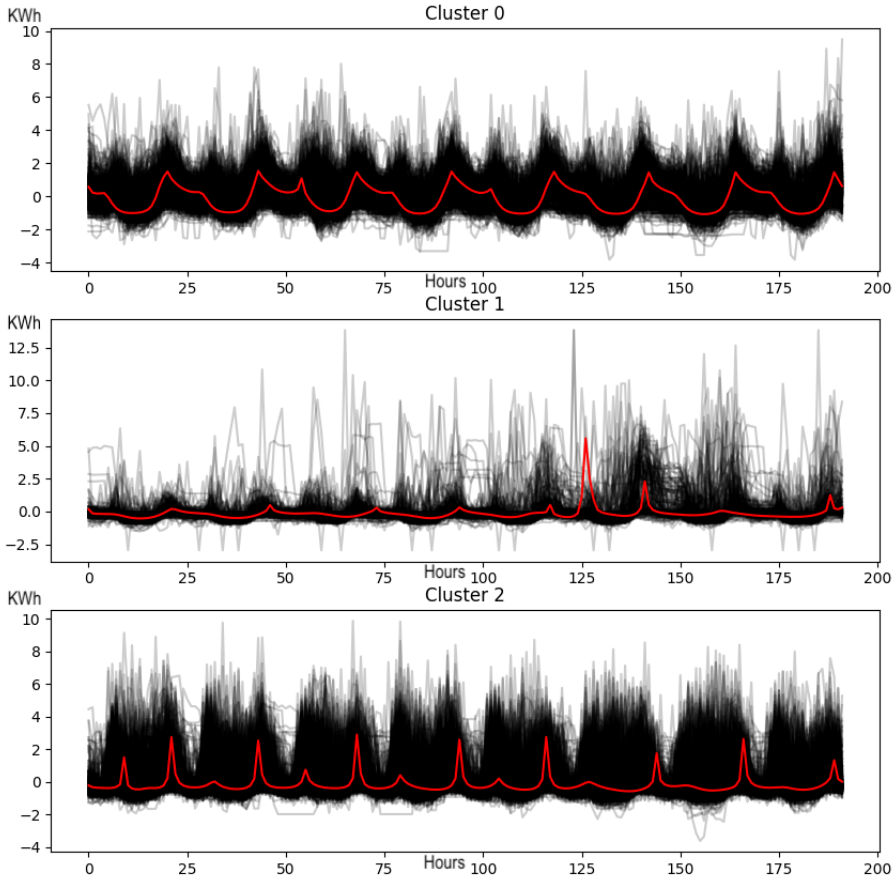


Fig. 17 Clusters using *tslearn* library with SoftDTW metric.

As it is shown in this figure 17, we can see a periodicity of the time series in the three clusters showing similar electrical patterns of consumption in the clusters. We then implemented an ensemble of clustering techniques, combining both DTW (Dynamic Time Warping) and SoftDTW (Soft Dynamic Time Warping) metrics to increase the robustness of time series clustering. DTW captures global changes, while SoftDTW focuses on local variations, making them complementary methods [26]. We combined their cluster labels using K-Means clustering, and further refined the clusters through majority voting and a custom condition to assign the most representative centroids. This ensemble approach enhances the prediction accuracy by balancing the strengths of each method, as reflected in the resulting clusters and their respective centroids. This approach can be seen in figure 18. The metrics of this approach are summarized in table 4, showing not the best bias metric result: 88% for cluster 0 and 35% for cluster 1. In the next subsection better metric performances on the classical LSTM and quantum-hybrid QLSTM forecasting models are shown.

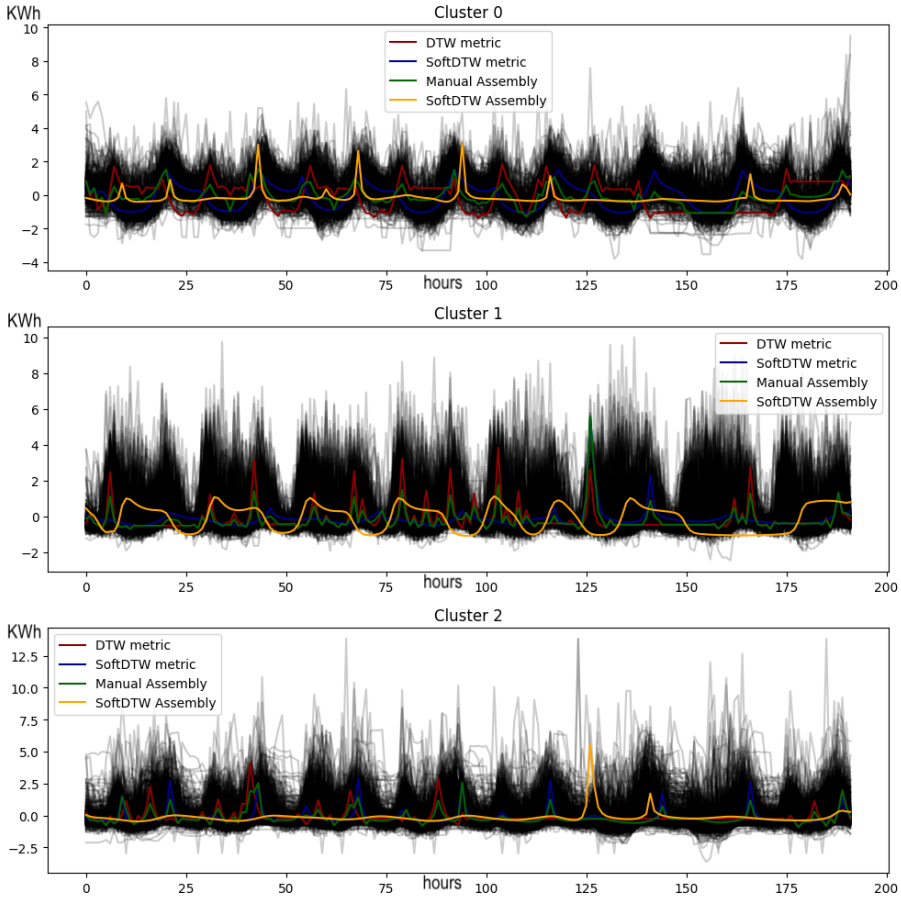


Fig. 18 Ensemble of clustering techniques combining both DTW (Dynamic Time Warping) and SoftDTW (Soft Dynamic Time Warping).

Cluster	Test MSE	Train MAE	Train MAPE
0	0.1517555432	0.3109731226	0.8811873999
1	0.0031502763	0.0384015272	0.3507619195

Table 4 Performance metrics for Clusters 0 and 1.

5.2 Tools and approaches used for LSTM and QLSTM

For this work, the classical models were constructed using python 3.11.5, Pytorch libraries for the LSTM and QLSTM. Pennylane software package was used for the construction of the VQC circuits and the QSLTM architecture based on the github repository from Pennylane [38] and the study done on the Stock price prediction using BERT and GAN developed in the GitHub repository [39]. There are two scopes used for the LSTM and QLSTM models:

1. **Approach 1**: The $ACTIVA\_IMP$ is the only variable taken into account for the training data $X_{training}$ and $Y_{training}$ and test data X_{test} and Y_{test} following the scope in [40] in which this variable is used for the LSTM model, this using the DataLoader object that handles batching and shuffling of the dataset. This approach was used only to show the independence between categories in subsection 5.4 and the demonstration of the best cost function in subsection 5.3
2. **Approach 2**: The $ACTIVA\_IMP$ is the forecasting variable y that has a dependence with the most correlated variables or features described in section 4, which are $V1$, $V2$, $V3$, $I1$, $I2$, $I3$, $REACTIVA\_EXP$ and $HORA$. This approach is based on [41] where the PyTorch Dataset and DataLoader objects are used in which for the SequenceDataset in each train and test dataset the features (in this study the most correlated variables) are set.

5.3 Best functional cost function: MSE *vs* MAE

In order to show what is the best cost function or error function, the functional described in subsection 2.1, the forecasting and model performance of one client (18741848) was made. Aiming at this, the following hyperparameters was chosen for the LSTM models:

Figure 19 shows that MSE cost function has a better performance as the training epochs pass. Different from MAE (Mean Absolute Error) cost function in which we can see from the figure that for both, training loss and test loss, have greater values (huge values) compared to MSE cost function. Also, we can see that for the first 20 epochs, the test loss is below of the training loss, thus showing underfitting, *i.e.*, the model may be too simple or not complex enough to capture the underlying patterns in the training data and it might not be learning effectively from the training set, resulting in a lower test loss but higher training loss.

In figure 20 we can observe that for this client the best fit for the actual values, KWh of the last 10 hours, is with the MSE cost function having a $R^2 = 0.9312$ and $MAPE = 0.075$ which means a 7.5 % for the latter, which is a pretty good error percentage. For MAE cost function we got a $R^2 = 0.68$ and $MAPE = 0.27$.

Hyperparameter	Value
Training/Test Split	80% training, 20% testing
Sequence Length (Training)	10
Sequence Length (Testing)	5
Input Size	1
Number of Layers	3
Hidden Size	64
Output Size	1
Batch Size	8
Number of Epochs	50
Forecast Steps	5
Learning Rate	0.0001
Number of Hidden Units	16

Table 5 Hyperparameters used for MSE and MAE LSTM models.

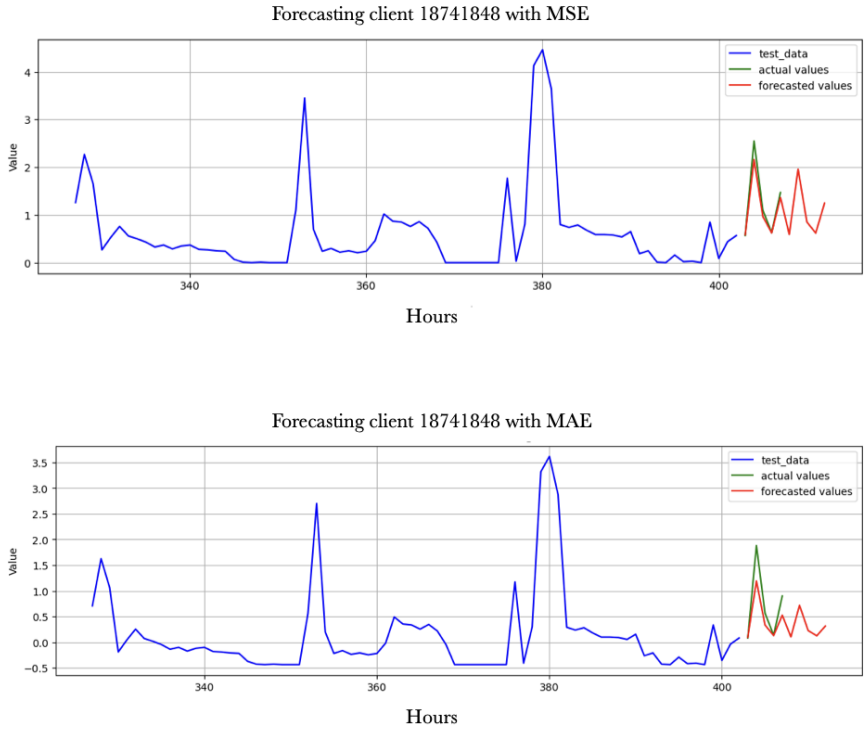


Fig. 20 Forecasting for client 18741848 using MSE and MAE cost functions.

5.4 Independence between categories using LSTM

Recalling that the aim of this work is to compare the performance of LSTM and QLSTM models. To achieve this, it is essential to develop our models based on predictions for individual clients (for business interests) as well as for groups of clients, as outlined in section 5.1. In this subsection, we demonstrate how certain categories

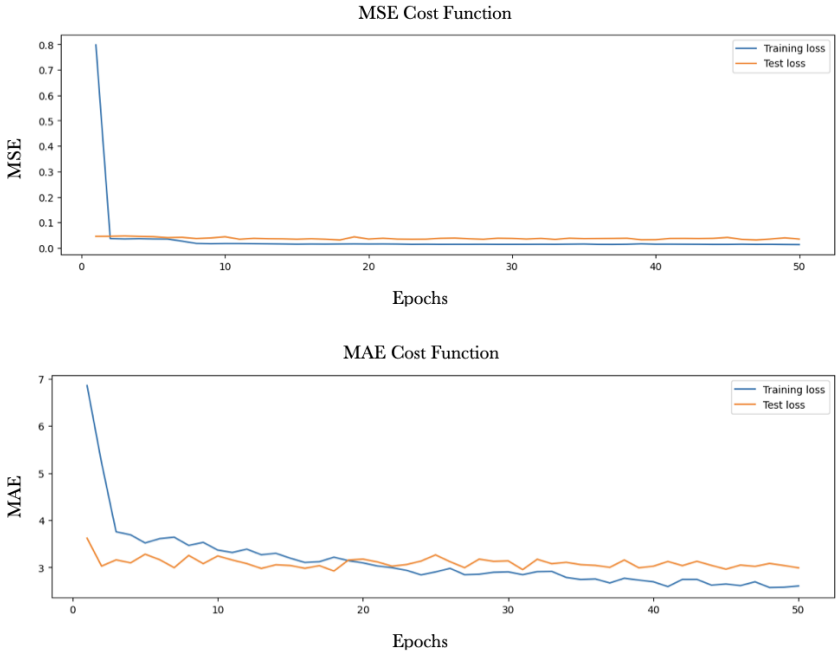


Fig. 19 LSTM forecasting model performance is evaluated using MSE and MAE cost functions, showing that training loss decreases over time, indicating optimization, and remains consistently below test loss. MSE demonstrates superior performance, with lower training and test loss values compared to MAE, which exhibits significantly higher losses throughout training.

exhibit low correlation by employing an analytical technique for time series analysis. This technique uses data from one category as the training dataset and data from another category as the test dataset. If both models for the categories or subcategories show strong performance in terms of the cost function (which measures error optimization) and the metrics used, such as R^2 and MAPE , one can conclude that there is a correlation between these categories or subcategories. For all the tests done for different subcategories we obtained good performance in terms of the MSE lost function. The first case was for both clients in the same category $(1 - RESIDENCIAL)$ and subcategory 6-ESTRATO 6. We can see in figure 21 a good fit for training and test dataset and the last 10 values of the $ACTIVA\_IMP$ for the last 10 hours. For this case we have R^2 : 0.8865 and $MAPE$: 0.3028472, which are good results. We can infer that clients in the same subcategory have similar electrical energy behaviour consumption.

However, taking two clients in different subcategory (6-ESTRATO 6 and 2-ESTRATO 2) bad results are obtained as can be seen in figure 22 with R^2 : -108.014921 and $MAPE$: 0.3261. With a negative R^2 we can conclude an inverse correlation between these two clients.

Figure 23 is the LSTM model prediction with the training data set for a client in the subcategory 6-ESTRATO 6 and a the test data set for a client that is the

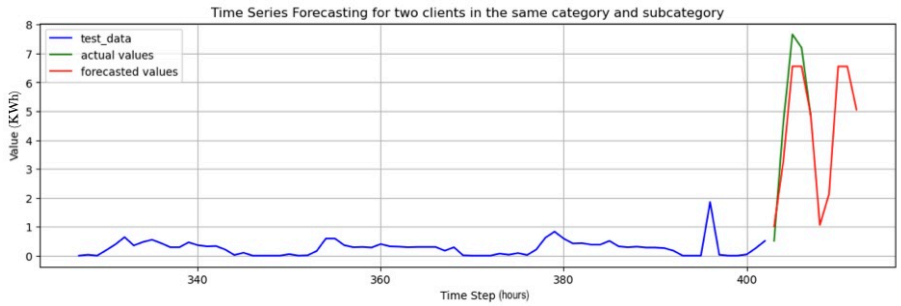


Fig. 21 LSTM forecasting using MSE cost function for two clients in the same category and subcategory.

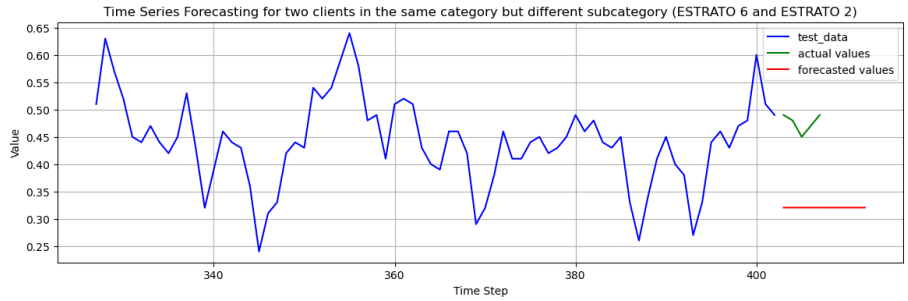


Fig. 22 LSTM forecasting using MSE cost function for two clients in the same category but in different subcategory. For training data set the *ACTIVA_IMP* values of a client in subcategory 6-ESTRATO 6 were taken and for test dataset the values of *ACTIVA_IMP* for a client in 2-ESTRATO 2 were taken.

$COMMERCIAL$ category. Bad correlation is observed with $R^2 = -2.16484$ and $MAPE = 0.2104$.

Fig. 23 LSTM forecasting using MSE cost function for two clients in the same category but in different subcategories.

Figure 24 is the LSTM model prediction with the training data set for a client in the subcategory $INDUSTRIAL$ and a the test data set for a client that is the

COMMERCIAL 11-220V category. Bad correlation is observed with $R^2 = -4.4936$ and $MAPE = 0.3454$

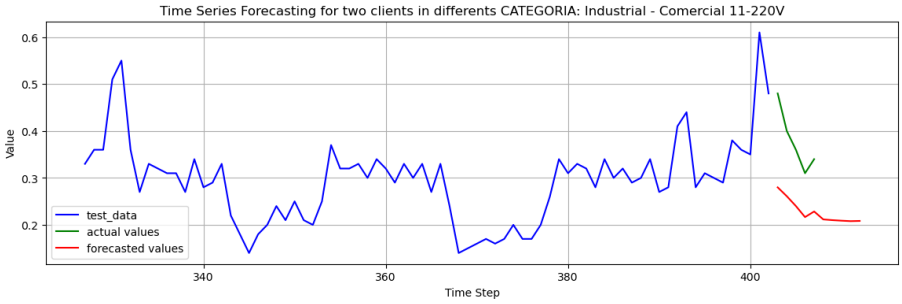


Fig. 24 LSTM forecasting using MSE cost function for two clients in the same category but in different subcategory. For training data set the $ACTIVA\_IMP$ values of a client in category $INDUSTRIAL$ were taken and for test dataset the values of $ACTIVA\_IMP$ for a client in COMMERCIAL 11-220V were taken.

5.5 One client forecasting: LSTM *vs* QLSTM

Our model that was evaluated for the LSTM and QLSTM was for one client. The client 18741848 was chosen using the approach 2 in which the most correlated variables were chosen. The two models were taken with the best hyperparameters described in table 6.

Hyperparameter	**Value**
Training/Test Split	67% training, 33% testing
Sequence Length (Training)	5
Sequence Length (Testing)	5
Input Size	1
Number of Layers	4
Hidden Size	52
Output Size	1
Batch Size	1
Number of Epochs	50
Forecast Steps	5
Learning Rate	0.0001
Number of Hidden Units	52

Table 6 Best hyperparameters for LSTM and QLSTM.

Furthermore, the best hyperparameters for the quantum layer (VQC hyperparameters) that were chosen are exposed in table 7.

As was explained in section 2.5 and in the encoding section 3 three angle rotations were chosen for the quantum embedding in the quantum layer encoding. Figure 25 shows the performance of the LSTM model in terms of the MSE cost function. The performance metrics for this model were $R^2 = 0.8580$ and $MAPE = 0.1796$.

Hyperparameter	Value
Number of Qubits	4
Angle Rotations	3
Quantum Layers	4

Table 7 Quantum Layer hyperparameters.

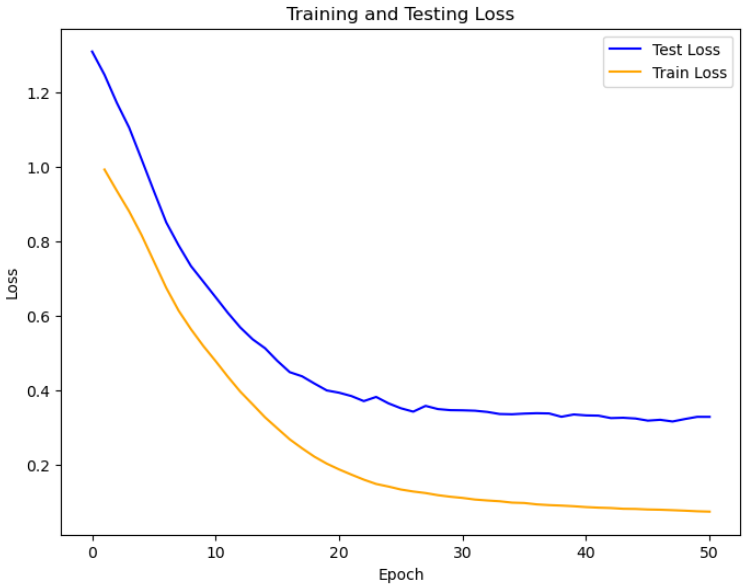


Fig. 25 LSTM forecasting model performance in terms of MSE cost function showing that the train loss as the epochs progress in time is getting lower values showing optimization (learning of the algorithm) and always being below of the test loss.

Figure 26 shows the performance of the QLSTM model in terms of the MSE cost function. The performance metrics for this model were $R^2 = 0.8585$ and $MAPE = 0.2217$. In terms of the variance metric both models QLSTM and LSTM for the best hyperparameters have the same results but the classical model wins with a 3% in the bias metric.

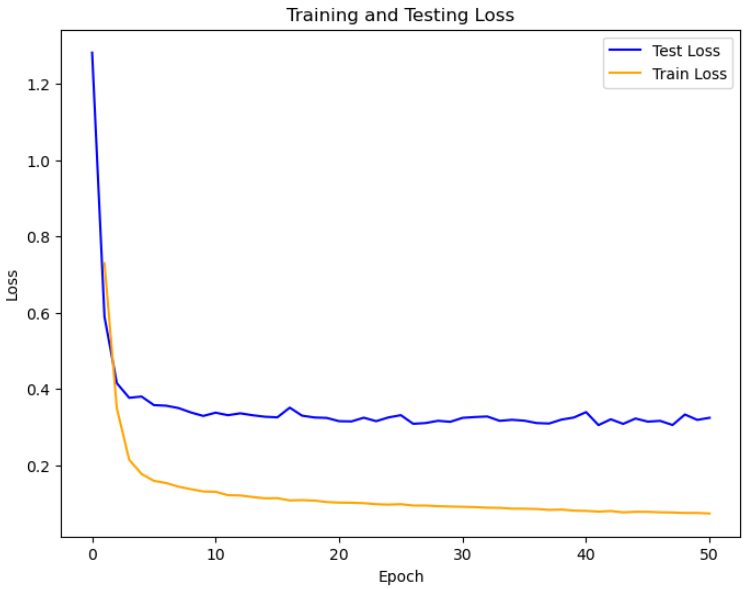


Fig. 26 QLSTM forecasting model performance in terms of MSE cost function showing that the train loss as the epochs progress in time is getting lower values showing optimization (learning of the algorithm) and always being below of the test loss.

Figures 27 and 28 show the comparison between LSTM and QLSTM in terms of the train and test loss respectively. From figure 27 is possible to observe how the QLSTM forecasting model learns faster than its classical counterpart, in the half of the epochs, approximately in the epoch 25, train loss has the same value of the train loss of the LSTM model in the epoch 50 demonstrating how QLSTM, when using the VQC shows a better optimization in terms of the angles tuning (deciding which parameters forget and which to update), that are later used in the activation functions of a classical LSTM infrastructure.

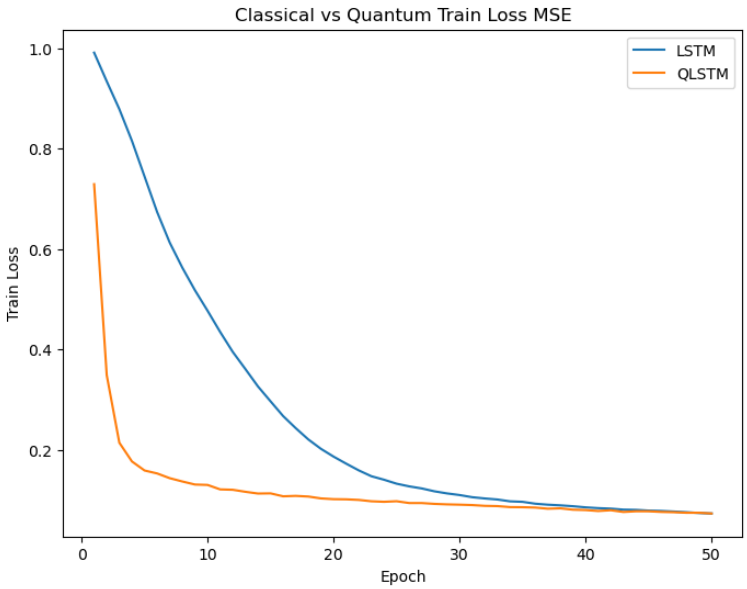


Fig. 27 LSTM vs QLSTM in terms of the train loss. In half of the epochs the QLSTM hybrid algorithm has the same value of the train loss or MSE cost function.

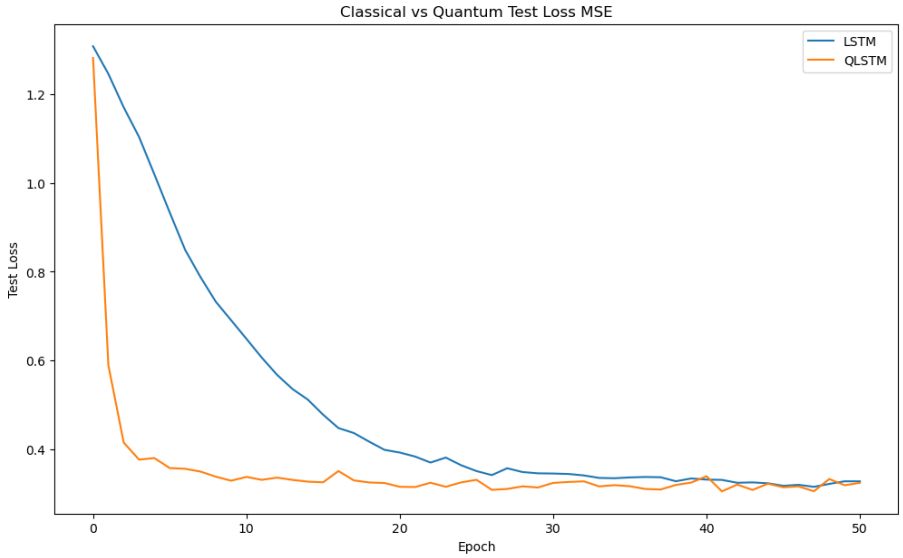


Fig. 28 LSTM vs QLSTM in terms of the train loss.

Finally, figure 29 shows the forecasting for the 408 hours of the electrical meter consumption (KHh) of the client 18741848 showing how both, classical LSTM model

and quantum-hybrid model have good fit with respect of the real data. Important to mention is that the number of parameters, weights of the LSTM, were 1553 and the parameters of QLSTM were 228, showing how the QLSTM is more efficient in terms of parameters to learn.

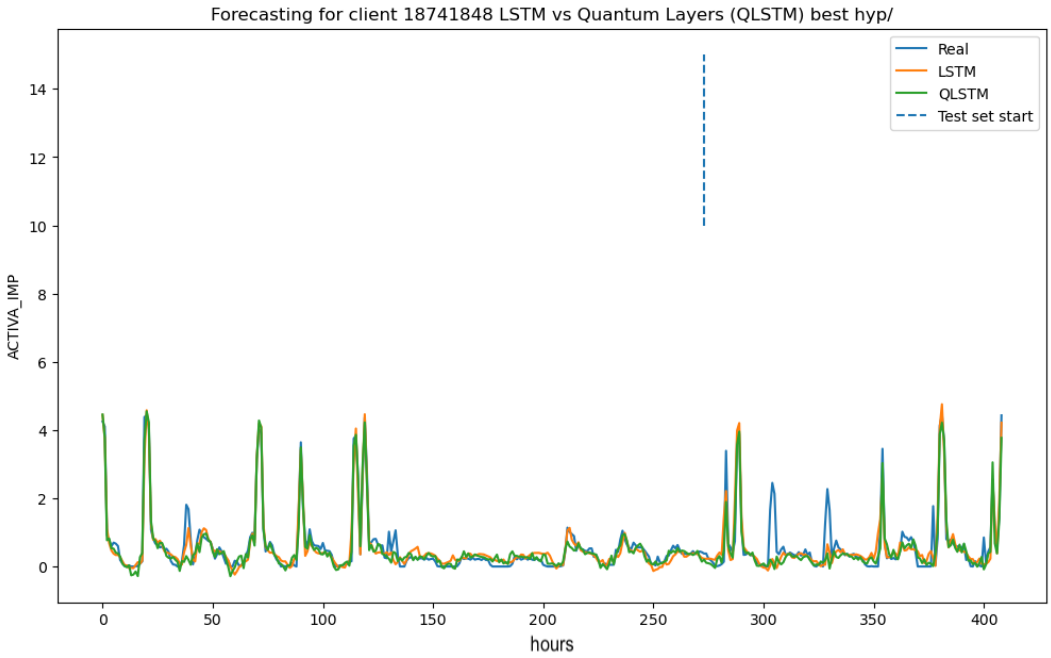


Fig. 29 Forecasting of *ACTIVA_IMP* for LSTM and QLSTM.

5.6 Subcategory: LSTM

Subcategory 6-ESTRATO 6 was chosen for this part of the work. In this category there were a total of 221 clients. Same hyperparameters from section 5.5 were chosen. For this classical LSTM the results for the $R^2 = 0.6614$ and $MAPE = 0.30$. These are good metrics of bias and variance taking into account that for this subcategory we have 221 clients in which many of them have different consumption habits. Even this value for the classical LSTM for this subgroup is better than the metrics found for the clustering work shown in section 5.1. Figure 30 shows the performance of the model in terms of the MSE cost function.

Figure 31 shows the LSTM model forecasting of this subcategory showing a good fit with the real data.

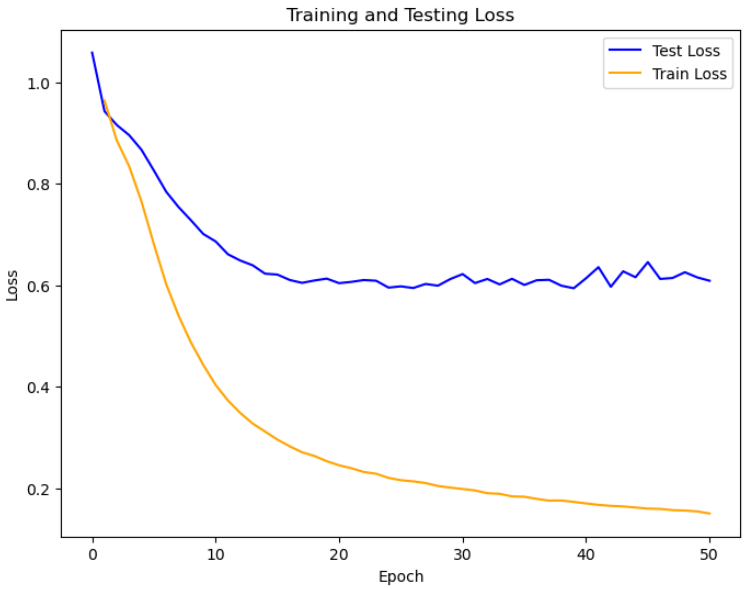


Fig. 30 LSTM forecasting model performance in terms of MSE cost function for subcategory 6-ESTRATO 6 showing that the train loss as the epochs progress in time is getting lower values showing optimization (learning of the algorithm) and always being below of the test loss.

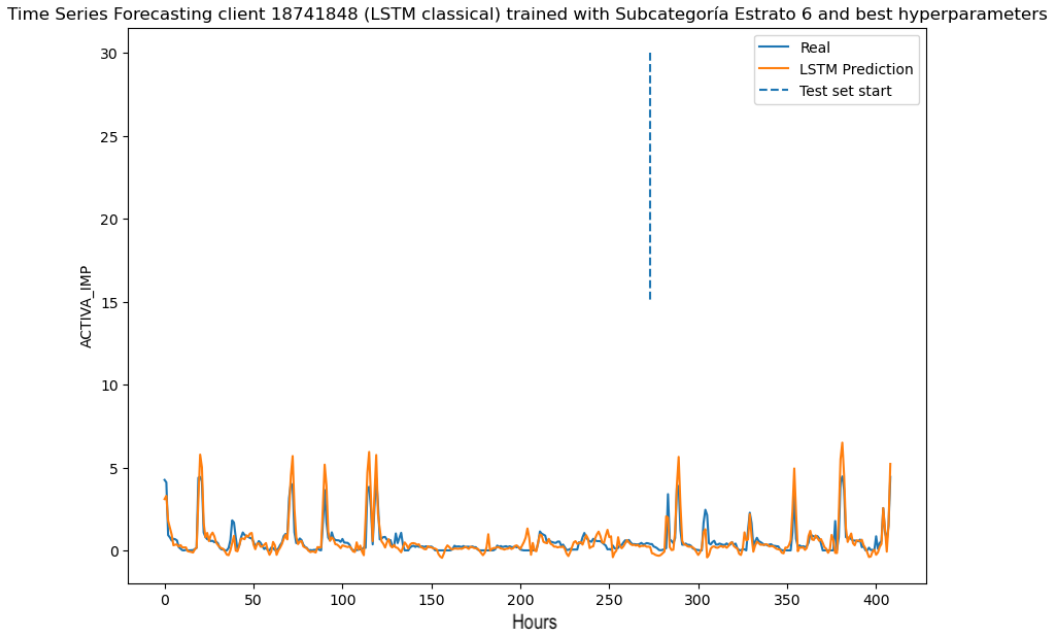


Fig. 31 Forecasting of *ACTIVA_IMP* for subcategory 6-ESTRATO 6 LSTM.

6 Conclussions and future work

Time series forecasting through clustering can be a viable option when combined with a distributed computing approach. However, as the bias and variance metrics suggest, this method may not be the most effective. Additionally, the lack of time correlation in certain series, as highlighted in section 5.4, underscores why this approach may be limited in its applicability.

In contrast, the application of both classical LSTM and quantum-hybrid QLSTM models for forecasting electrical metrics demonstrated strong performance, particularly in the realm of short-term load forecasting (STLF). These models excel in terms of learning efficiency during the optimization of tuning parameters in recurrent neural networks (RNNs). Notably, the QLSTM model, using the Pennylane simulator, exhibited a faster convergence compared to the classical LSTM. By approximately epoch 25, the QLSTM's training loss matched the classical LSTM's performance at epoch 50, demonstrating the QLSTM's superior parameter optimization, specifically in how it determines which parameters to retain or update. This optimization is enhanced by the variational quantum circuit (VQC) and directly impacts the activation functions within the LSTM framework.

This improvement is particularly relevant in the electricity sector, where accurate short-term load forecasting is crucial for predicting future consumption patterns. Additionally, the QLSTM's parameter optimization efficiency illustrates the benefits of quantum phenomena, such as superposition and entanglement, in hybrid quantum-classical models.

Future work will focus on applying QLSTM to various subcategories and categories within the electrical domain, leveraging high-performance computing. Furthermore, implementing these models on real quantum hardware will allow for a deeper evaluation of quantum computers' potential advantages in terms of computation time.

7 Aknowledgments

We would like to express our gratitude to EPM for providing the data and for their interest in exploring quantum computing as a solution to this forecasting problem. Special thanks to Jhon Dario Medina and Yeis Livis Taborda for their valuable insights into the electricity metering business and for Yeis' work on clustering that contributed to this research.

On a personal note, I would like to extend my deepest appreciation to Professor Juan Guillermo Lalinde for his trust in me as his master's student throughout the preparation of this work. I am also grateful to my co-advisor Daniel for his guidance and support. The teachings of both have greatly shaped my academic journey, and this work is a reflection of their mentorship.

References

[1] Liu, C.-L., Tseng, C.-J., Huang, T.-H., Yang, J.-S., Huang, K.-B.: A multi-task learning model for building electrical load prediction. Energy and Buildings **278**, 112601 (2023) https://doi.org/10.1016/j.enbuild.2022.112601

[2] Alfares, H.K., Nazeeruddin, M.: Electric load forecasting: Literature survey and classification of methods. International Journal of Systems Science **33**(1), 3–34 (2002)

[3] Khotanzad, A., Zhou, E., Elragal, H.: A neuro-fuzzy approach to short-term load forecasting in a price-sensitive environment. IEEE Transactions on Power Systems **17**, 1273–1282 (2006)

[4] Arora, S., Taylor, J.W.: Rule-based autoregressive moving average models for forecasting load on special days: A case study for france. European Journal of Operational Research **266**(1), 259–268 (2018) https://doi.org/10.1016/j.ejor.2017.08.056

[5] Maldonado, S., González, A., Crone, S.: Automatic time series analysis for electric load forecasting via support vector regression. Applied Soft Computing **83**, 105616 (2019) https://doi.org/10.1016/j.asoc.2019.105616

[6] Bendaoud, N.M.M., Farah, N.: Using deep learning for short-term load forecasting. Neural Computing and Applications **32**(18), 15029–15041 (2020) https://doi.org/10.1007/s00521-020-04856-0

[7] Javed, F., Arshad, N., Wallin, F., Vassileva, I., Dahlquist, E.: Forecasting for demand response in smart grids: An analysis on use of anthropologic and structural data and short term multiple loads forecasting. Applied Energy **69**, 15–160 (2012)

[8] He, F., Zhou, J., Feng, Z.-k., Liu, G., Yang, Y.: A hybrid short-term load forecasting model based on variational mode decomposition and long short-term memory networks considering relevant factors with bayesian optimization algorithm. Applied Energy **237**, 103–116 (2019) https://doi.org/10.1016/j.apenergy.2019.01.055

[9] Biswas, M.A.R., Robinson, M.D., Fumo, N.: Prediction of residential building energy consumption: A neural network approach. Energy **117**, 84–92 (2016) https://doi.org/10.1016/j.energy.2016.10.066

[10] Amarasinghe, K., Marino, D.L., Manic, M.: Deep neural networks for energy load forecasting. In: Proceedings of the 2017 IEEE 26th International Symposium on Industrial Electronics (ISIE), vol. 14. Edinburgh, UK, pp. 1483–1488 (2017)

[11] Kong, W., Dong, Z.Y., Jia, Y., Hill, D.J., Xu, Y., Zhang, Y.: Short-term residential load forecasting based on lstm recurrent neural network. IEEE Transactions on Smart Grid **10**(1), 841–851 (2019) https://doi.org/10.1109/TSG.2017.2753802

[12] Yu, Y., Hu, G., Liu, C., Xiong, J., Wu, Z.: Prediction of solar irradiance one

hour ahead based on quantum long short-term memory network. IEEE Transactions on Quantum Engineering **4**, 1–15 (2023) https://doi.org/10.1109/TQE.2023.3271362

[13] Wang, F., Xuan, Z., Zhen, Z., Li, K., Wang, T., Shi, M.: A day-ahead pv power forecasting method based on lstm-rnn model and time correlation modification under partial daily pattern prediction framework. Energy Conversion and Management **212**, 112766 (2020) https://doi.org/10.1016/j.enconman.2020.112766

[14] Maheshwari, D., Dierra-Dosa, D., Darcia-Zapirain, D.: Variational quantum classifier for binary classification: Real vs synthetic dataset. IEEE Access (2021) https://doi.org/10.1109/ACCESS.2021.3139323

[15] Cerezo, M., Verdon, G., Huang, H.Y., Cincio, L., Coles, P.J.: Challenges and opportunities in quantum machine learning. Nat Comput Sci **2**(9), 567–576 (2022) https://doi.org/10.1038/s43588-022-00311-3

[16] Maheshwari, D., Garcia-Zapirain, B., Sierra-Sosa, D.: Quantum machine learning applications in the biomedical domain: A systematic review. IEEE Access **10**, 80463–80484 (2022) https://doi.org/10.1109/ACCESS.2022.3195044

[17] Chen, S.Y.-C., Yoo, S., Fang, Y.-L.L.: Quantum long short-term memory. In: ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 8622–8626 (2022). https://doi.org/10.1109/ICASSP43922.2022.9747369

[18] Wiebe, N.: Key questions for the quantum machine learner to ask themselves. New Journal of Physics **22**(9) https://doi.org/10.1088/1367-2630/abac39

[19] Schuld, M., Bergholm, V., Gogolin, C., Izaac, J., Killoran, N.: Evaluating analytic gradients on quantum hardware. Physical Review A **99**, 032331 (2019) https://doi.org/10.1103/PhysRevA.99.032331

[20] Aggarwal, C.C.: Neural Networks and Deep Learning: A Textbook, 1st edn., p. 497. Springer, ??? (2018). https://doi.org/10.1007/978-3-319-94463-0 . Springer International Publishing AG, part of Springer Nature 2018. https://doi.org/10.1007/978-3-319-94463-0

[21] Kübler, J.M., Arrasmith, A., Cincio, L., Coles, P.J.: An adaptive optimizer for measurement-frugal variational algorithms. Quantum Physics **4**, 263 (2020) https://doi.org/10.22331/q-2020-05-11-263

[22] al., J.A.: Quantum machine learning architecture for covid-19 classification based on synthetic data generation using conditional adversarial neural network. Cognitive Computation (2021) https://doi.org/10.1007s12559-021-09926-6

[23] El-shafeiy, E., Hassanien, A.E., Sallam, K.M., Abohany, A.A.: Approach for training quantum neural network to predict severity of covid-19 in patients. Computers, Materials and Continua **66**(2), 1745–1755 (2021)

[24] Schuld, M., Petruccione, F.: Machine Learning with Quantum Computers, 2nd edn. Quantum Science and Technology, p. 312. Springer, ??? (2021). https://doi.org/10.1007/978-3-030-83098-4 . Originally published as "Supervised Learning with Quantum Computers"

[25] Weigend, A.S., Gershenfeld, N.A.: Time Series Prediction: Forecasting the Future and Understanding the Past. Addison Wesley, Harlow, UK (1994)

[26] Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning, 2nd edn. Springer Series in Statistics, p. 745. Springer, New York, NY (2009). https://doi.org/10.1007/978-0-387-84858-7 . https://doi.org/10.1007/978-0-387-84858-7

[27] Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press, ??? (2016). http://www.deeplearningbook.org

[28] Jenkins, I.R., Gee, L.O., Knauss, A., Yin, H., Schroeder, J.: Accident scenario generation with recurrent neural networks. In: 2018 21st International Conference on Intelligent Transportation Systems (ITSC), pp. 3340–3345 (2018). https://doi.org/10.1109/ITSC.2018.8569661

[29] Nielsen, M.A., Chuang, I.L.: Quantum Computation and Quantum Information. Cambridge University Press, ??? (2000)

[30] Sorjamaa, A., Lendasse, A.: Time series prediction using direct strategy. In: Verleysen, M. (ed.) ESANN06, European Symposium on Artificial Neural Networks, pp. 143–148. European Symposium on Artificial Neural Networks, Bruges, Belgium

[31] Giovannetti, V., Lloyd, S., Maccone, L.: Advances in quantum metrology. Nature Photonics **5**, 222–229 (2011)

[32] Chiribella, G., D'Ariano, G.M., Perinotti, P.: Theoretical framework for quantum networks. Physical Review A **80**, 022339 (2009)

[33] Alchieri, L., Badalotti, D., Bonardi, P., *et al.*: An introduction to quantum machine learning: from quantum logic to quantum deep learning. Quantum Machine Intelligence **3**, 28 (2021) https://doi.org/10.1007/s42484-021-00056-8

[34] Sierra-Sosa, D., Arcila-Moreno, J.D., Garcia-Zapirain, B., Elmaghraby, A.: Diabetes type 2: Poincaré data preprocessing for quantum machine learning. Computers, Materials & Continua, Tech Science Press (2021) https://doi.org/10.32604/cmc.2021.013196 . Received: 17 July 2020; Accepted: 07 December 2020

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

[36] Sierra-Sosa, D., Pal, S., Telahun, M.: Data rotation and its influence on quantum encoding. Quantum Information Processing **22**, 89 (2023) https://doi.org/10.1007/s11128-023

[37] Wadhwa, C.L.: Electrical Power Systems, 6th edn. New Age International, ??? (2005). https://www.newagepublishers.com/servlet/nagetbiblio?bno=000093

[38] Disipio, R.: QLSTM Implementation in PennyLane. https://github.com/rdisipio/qlstm/blob/main/qlstm_pennylane.py. Accessed: 2024-09-23 (2023)

[39] Dulal, D.: Data Collection Notebook for SoftServe QLSTM. https://github.com/DikshantDulal/SoftServe_QLSTM/blob/main/Data_collection.ipynb. Accessed: 2024-09-23 (2023)

[40] GeeksforGeeks: Time Series Forecasting Using PyTorch. https://www.geeksforgeeks.org/time-series-forecasting-using-pytorch/. Accessed: 2024-09-23 (2023)

[41] Kent, B.: How to use PyTorch LSTMs for time series regression. https://www.crosstab.io/articles/time-series-pytorch-lstm/. Accessed: 2024-09-23 (2021)