

Quantum Temporal Fusion Transformer

Krishnakanta Barik*, Goutam Paul†

Cryptology and Security Research Unit, Indian Statistical Institute

August 7, 2025

Abstract

The Temporal Fusion Transformer (TFT), proposed by Lim et al. [*International Journal of Forecasting*, 2021], is a state-of-the-art attention-based deep neural network architecture specifically designed for multi-horizon time series forecasting. It has demonstrated significant performance improvements over existing benchmarks. In this work, we propose a Quantum Temporal Fusion Transformer (QTFT), a quantum-enhanced hybrid quantum-classical architecture that extends the capabilities of the classical TFT framework. Our results demonstrate that QTFT is successfully trained on the forecasting datasets and is capable of accurately predicting future values. In particular, our experimental results display that in certain test cases, the model outperforms its classical counterpart in terms of both training and test loss, while in the remaining cases, it achieves comparable performance. A key advantage of our approach lies in its foundation on a variational quantum algorithm, enabling implementation on current noisy intermediate-scale quantum (NISQ) devices without strict requirements on the number of qubits or circuit depth.

*krishnakanta_r@isical.ac.in

†goutam.paul@isical.ac.in

1 INTRODUCTION

Multi-horizon forecasting is a time series forecasting [1] technique in which a model predicts interesting variables for multiple future time steps. Unlike standard time series forecasting, which predicts variables one step ahead, multi-horizon forecasting predicts variables for several future time points, like predicting sales for the next week, not just the next day, i.e., it predicts across the entered future path. Multi-horizon forecasting has many effective applications in the real world, including healthcare [2–4], financial [5, 6], retail [7, 8]. Figure 1 provides an overview of the overall architecture of multi-horizon forecasting.

Multi-horizon forecasting relies on diverse data sources, including fixed time-independent features (e.g., the store’s location), known information about the future (e.g., an upcoming holiday), and comprehensive historical data (e.g., customer price trade). Without understanding the relationships between a variety of these data sources, multi-horizon time series forecasting is a challenging task. Several standard approaches have been proposed, and we discuss some of them below.

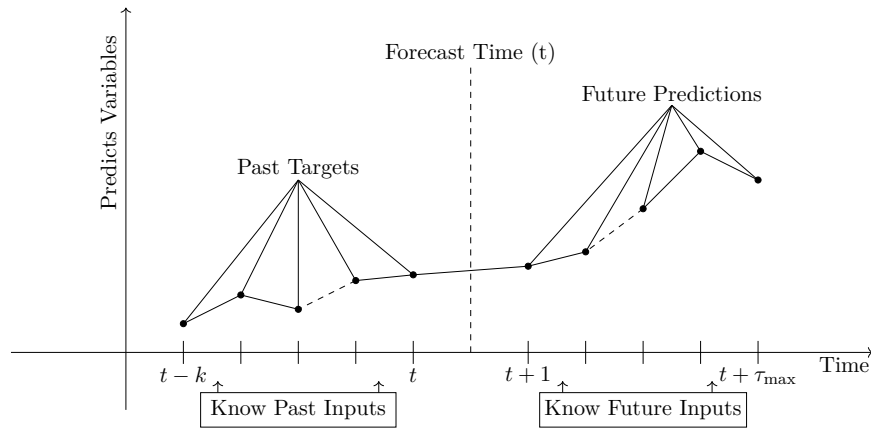


Figure 1: Illustration of multi-horizon forecasting. The X-axis represents the time steps (sliding window), while the Y-axis represents the target variables to be predicted. The forecast time point is denoted as t . The model uses historical data from $t - k$ to t to predict the selected variable over the future horizon, from t to $t + \tau_{\max}$.

There are various architectures based on Recurrent Neural Networks (RNNs) [9–11] that have been developed for performing multi-horizon time series forecasting. Deep Neural Networks (DNNs) have also been commonly used, and have demonstrated strong performance over traditional time series models [10, 12, 13]. Recently, transformer-based models [14] have been introduced for time series forecasting and have shown improved results. However, these models sometimes perform poorly or fail when dealing with different types of inputs commonly encountered in multi-horizon forecasting [9–11, 14]. In the paper [15], Lim et al. introduced a new model, the Temporal Fusion Transformer (TFT), a novel model for multi-horizon forecasting. Building upon Deep neural networks (DNNs) and attention mechanisms [16], the TFT model demonstrates superior performance compared to existing models.

Quantum computing is a rapidly advancing field in computer science that harnesses quantum bits (say qubits) to perform computations based on the principles of quantum mechanics, such as entanglement and superposition, thereby unlocking computational power beyond that of classical algorithms. Several leading technology companies, including Google [17], IBM [18], and D-Wave [19], have developed quantum computers that are accessible to the general public through cloud-based

services. These advancements mark significant progress in making quantum computing more practical and push research and innovation across various scientific and industrial domains. Quantum computing solves certain classes of problems exponentially faster than classical computing [20, 21]. However, this impressive speed-up highly depends on the standard of the underlying quantum computer system. Quantum circuits that involve many qubits or require deep circuit depths are not reliably executed on current Noisy Intermediate-Scale Quantum (NISQ) devices [22] due to the absence of quantum error and noise [23, 24]. Therefore, it is significant to design quantum frameworks for execution on NISQ devices, ensuring better outcomes despite hardware limitations.

Quantum Variational Algorithms (VQAs) [25–28] are one of the breakthrough innovations of quantum computing, offering a promising algorithm that is potentially applicable to NISQ devices. A VQA is essentially a suitable quantum circuit, where certain gate parameters are tunable and updated iteratively through a classical optimization process to solve a given problem. Since VQA is an iterative optimization method, the noise inherent in quantum devices can often be effectively mitigated through the tunable parameters of the quantum circuit. As a result, VQAs are particularly suitable for implementation on the currently available NISQ devices.

This work addresses the challenges of learning sequential data using Quantum Machine Learning techniques (QML) [29–32]. We propose a novel framework to explain the practically feasible implementation of attention-based deep neural networks with a variational quantum algorithm. Specifically, we propose the Temporal Fusion Transformer (TFT) - an attention-based DNN capable of learning from time series data and performing multi-horizon forecasting - using a variational quantum algorithm. Our quantum-classical hybrid model is designed to be efficiently implementable on current noise quantum hardware (NISQ devices), utilizing key quantum properties such as superposition and entanglement. In the numerical simulation part, we implement a simplified version of the quantum TFT model. In several test cases, the quantum model outperforms its classical counterpart in terms of both training and test loss, while in the remaining cases, it demonstrates comparable performance. This raises the question: why should we focus on this model? Our current implementation employs a highly simplified version due to the limitations of existing quantum hardware, which remains constrained by noise and error rates. However, in the future, as quantum computers overcome these constraints, they have the potential to deliver significantly better results. To the best of our knowledge, this is the first time successfully transformed a large-scale classical learning model into a quantum learning model.

Our contributions are summarized as follows

- We introduce, for the first time, quantum-enhanced Gated Residual Network [15] and Interpretable Multi-head Attention [15].
- We are the first to train and evaluate a quantum-enhanced Temporal Fusion Transformer (QTFT) model to perform multi-horizon time series forecasting.

The remainder of this paper is organized as follows. First, in Section 2, we present a brief review of the classical counterpart of the temporal fusion transformer, including a proper explanation of each component and details of the model architecture. In Section 3, we introduce the variational quantum algorithm, the building block of our model. We discuss our main proposal in Section 4. This section explains the tools used in QTFT, outlines model architecture, and describes the optimization procedure. In Section 5, we present the implementation of our model, followed by the conclusion in Section 6.

2 CLASSICAL TEMPORAL FUSION TRANSFORMER

Throughout this discussion, for the sake of explanation and understanding, we consider the dataset of stores in retail or patients in healthcare. We use the same notation as [15]. There are three main input components of the Temporal Fusion Transformer (TFT), including a set of static covariates $\mathbf{s} \in \mathbb{R}^{m_s}$, where m_s be the dimension of static variables, time-dependent inputs $\mathbf{x}_t \in \mathbb{R}^{m_x}$, and corresponding scalar targets output y_t at each time step t between 0 to T . Static covariates provide information that doesn't change over time, such as a store's size. The time-dependent input features are partitioned into two parts: observed inputs $\mathbf{z}_t \in \mathbb{R}^{m_z}$, which can only measure them after they happen (e.g., weather), and know inputs $\mathbf{x}_t \in \mathbb{R}^{m_x}$, that are known beforehand (e.g., holiday, voting day).

Another important concept in TFT is quantile forecasting, a technique that predicts an interval of the possible outputs rather than a single point output. Further details are given below. Quantile based multi-horizon forecast is represented as

$$\hat{y}(q, t, \tau) = f_q(\tau, y_{t-k:t}, \mathbf{z}_{t-k:t}, \mathbf{x}_{t-k:t+\tau}, \mathbf{s}),$$

where $y_{t-k:t} = \{y_{t-k}, y_{t-k+1}, \dots, y_t\}$ and similarly for \mathbf{z}, \mathbf{x} . Here $\hat{y}(q, t, \tau)$ is the predicted q -th sample quantile for the forecast τ time steps ahead at a time t , and $f_{(\cdot)}$ is the quantile-specific prediction model. The forecast horizon spans $\tau \in \{1, 2, \dots, \tau_{\max}\}$, and k defines the size of the past information window.

2.1 Components

The TFT model uses several component details below to learn successfully the time series data for multi-horizon forecasting.

A. Gated residual networks [15]

The relationship between multi-dimensional inputs and target outputs is typically unknown in advance, making it challenging to estimate which features are most relevant for prediction. Depending on the nature of the input data, non-linear and linear models may be necessary to accurately predict target values. The Gated Residual Networks (GRN) address this challenge by combining activation function and residual connection, serving as a core building block of Temporal Fusion Transformer (TFT). GRN receives two inputs: primary input \mathbf{a} and optional input \mathbf{c} .

First, the primary input \mathbf{a} and the optional input \mathbf{c} passed through a neural network with an Exponential Linear Unit (ELU) [33] activation function. The ELU would behave like an identity function when the input is positive and for negative input, the ELU would generate a constant output.

$$\boldsymbol{\eta}_1 = \text{ELU}(\mathbf{W}_1 \mathbf{a} + \mathbf{W}_2 \mathbf{c} + \mathbf{b}_{12}),$$

where $\mathbf{W}_{(\cdot)}$ and $\mathbf{b}_{(\cdot)}$ are denoted as the learnable weight matrices and bias vectors, respectively. Next, the output $\boldsymbol{\eta}_1$ from the previous layer is passed through another neural network without any activation function.

$$\boldsymbol{\eta}_2 = \mathbf{W}_3 \boldsymbol{\eta}_1 + \mathbf{b}_3.$$

Now $\boldsymbol{\eta}_2$ are fitted into gating layers based on Gated Linear Units (GLUs) [34] that are used to selectively deactivate parts of the model that are unnecessary for a specific dataset. The GLU is

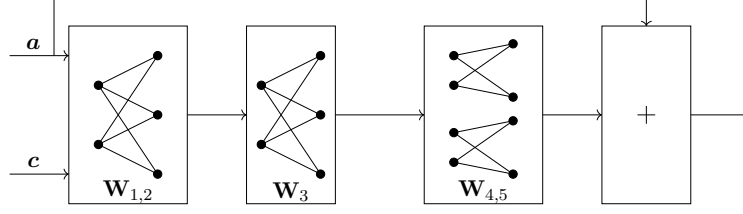


Figure 2: Generic architecture for Gated Residual Networks (GRNs). The input \mathbf{a} represents the primary input, and \mathbf{c} is an optional external context vector. $\mathbf{W}_{1,2}$ is a dense (neural network) layer followed by an ELU activation function. \mathbf{W}_3 is another dense layer without activation function. $\mathbf{W}_{4,5}$ represented the Gated Linear Unit (GLU) operation. Final block performance residual connection (add) and layer normalization.

defined as follows

$$\boldsymbol{\eta}_3 = \text{GLU}(\boldsymbol{\eta}_2) = \sigma(\mathbf{W}_4 \boldsymbol{\eta}_2 + \mathbf{b}_4) \odot (\mathbf{W}_5 \boldsymbol{\eta}_2 + \mathbf{b}_5),$$

where $\sigma(\cdot)$ denote as sigmoid activation function and \odot is the element-wise Hadamard product. Finally, the input \mathbf{a} is combined with $\boldsymbol{\eta}_3$ through a residual connection, and the result is refined through a layer normalization step [35] as below, ensuring stable and consistent activations,

$$\text{GRN}(\mathbf{a}, \mathbf{c}) = \text{LayerNorm}(\mathbf{a} + \boldsymbol{\eta}_3).$$

B. Variable selection networks [15]

In multi-horizontal forecasting, variables play an important role - some variables may be most significant for predicting problems, while other variables may introduce unnecessarily noisy input datasets and not impact performance. Therefore, identifying and distinguishing the most appropriate variables is a vital task for improving overall model effectiveness. To address the issues, variable selection networks, a learnable model, provide an effective solution for efficiently handling multiple variables in the dataset. For better mathematical representation, categorical variables are encoded using entity embedding [36] of dimension d_{model} while continuous variables are transformed linearly with the same dimension. Variational selection networks are applied separately to all three types of inputs - static, past, and future. Here present variational selection networks for past inputs, the same structure is applied to both static and future inputs.

Let the encoded past input of j -th variable at time t be denoted by $\boldsymbol{\xi}_t^{(j)} \in \mathbb{R}^{d_{\text{model}}}$. By concatenating encoded past inputs at time t , we set a representation as flattened vector with $\boldsymbol{\Xi}_t = \left[\boldsymbol{\xi}_t^{(1)T}, \dots, \boldsymbol{\xi}_t^{(m_\chi)T} \right]^T$. Both $\boldsymbol{\Xi}_t$ and an external context vector \mathbf{c}_s , obtained from a static covariate encoder (omitted for static variables), are passed through GRN, followed by a softmax layer [37]

$$\mathbf{v}_{\chi_t} = \text{Softmax}(\text{GRN}(\boldsymbol{\Xi}_t, \mathbf{c}_s)),$$

where the softmax function is defined as

$$\text{Softmax}(w_1, w_2, \dots, w_k) = \left(\frac{e^{w_1}}{\sum_{i=1}^k e^{w_i}}, \frac{e^{w_2}}{\sum_{i=1}^k e^{w_i}}, \dots, \frac{e^{w_k}}{\sum_{i=1}^k e^{w_i}} \right),$$

for any $(w_1, w_2, \dots, w_k) \in \mathbb{R}^k$ and \mathbf{v}_{χ_t} is an m_χ dimensional vector, called variable selection weights.

At each time step t , another GRN layer is applied to encoded input $\xi_t^{(j)}$, for all $j \in [0, m_\chi]$

$$\tilde{\xi}_t^{(j)} = \text{GRN} \left(\xi_t^{(j)} \right),$$

where $\tilde{\xi}_t^{(j)}$ is called processed feature vector. The final output of the variable selection network is a weighted sum of processed feature vectors, where the weights are given by the variable selection weights

$$\tilde{\xi}_t = \sum_{j=1}^{m_\chi} v_{\chi_t}^{(j)} \tilde{\xi}_t^{(j)},$$

where $v_{\chi_t}^{(j)}$ is j -th component of the vector \mathbf{v}_{χ_t} .

C. Static covariate encoders [15]

Static variables play a crucial role in time series forecasting, as different components of models utilize them in various forms. Specifically, there are three main parts of the TFT model where four distinct context vectors are required to improve predictive accuracy. This context vector $\mathbf{c}_s, \mathbf{c}_e, \mathbf{c}_c, \mathbf{c}_h$ are generated by static covariate encoder using separate GRN encoders (different by parameters). Each encoder takes the fixed input $\tilde{\xi}$, which is the output of the static variable selection network

$$\mathbf{c}_j = \text{GRN}(\tilde{\xi}),$$

for $j \in \{s, e, c, h\}$.

D. Interpretable Multi-head attention [15]

The attention mechanism [14, 16] is an important tool for capturing long-term relationships between different elements in the input data. We provide a general framework for applying the attention mechanism across different domains; in the context of the TFT, we specifically incorporate it within the temporal self-attention layer.

Let $\mathbf{S} \in \mathbb{R}^{N \times d}$ be the matrix representing the input vectors. Let $\mathbf{W}_q, \mathbf{W}_k \in \mathbb{R}^{d \times d_{\text{attn}}}$ and $\mathbf{W}_v \in \mathbb{R}^{d \times d_{\text{attn}}}$ be learnable parameter matrices used to project the input into query \mathbf{Q} , key \mathbf{K} , and value \mathbf{V} spaces, respectively, i.e., $\mathbf{Q} = \mathbf{S}\mathbf{W}_q$, $\mathbf{K} = \mathbf{S}\mathbf{W}_k$, $\mathbf{V} = \mathbf{S}\mathbf{W}_v$. The output of attention operation is defined as

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = A(\mathbf{Q}, \mathbf{K})\mathbf{V},$$

where $A(\mathbf{Q}, \mathbf{K}) = \text{softmax} \left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_{\text{attn}}}} \right)$. Multi-head attention, introduced in [16], improves the learning capacity of the model by enabling it to jointly attend (different heads) to information from different representation subspaces at various positions of the given input data. If the number of attention head is m_H , then output of multi-head attention mechanism is given by

$$\text{Multi-head}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \left[\text{Attention} \left(\mathbf{Q}^{(1)}, \mathbf{K}^{(1)}, \mathbf{V}^{(1)} \right), \dots, \text{Attention} \left(\mathbf{Q}^{(m_H)}, \mathbf{K}^{(m_H)}, \mathbf{V}^{(m_H)} \right) \right] \mathbf{W}_H,$$

where $\mathbf{Q}^{(h)}, \mathbf{K}^{(h)}, \mathbf{V}^{(h)}$ are weights for queries, key and value projections for the h -th attention head, and \mathbf{W}_H be the matrix used to combine the concatenated outputs of all attention head.

In a multi-head attention mechanism, the value vectors ($\mathbf{V}^{(\cdot)}$) play a crucial role in determining the importance of specific features. When different value vectors are used in each head, they may fail to prioritize certain features consistently. In contrast, sharing the same value vectors in each head and additive aggregation of all heads increases the model's capacity efficiently. This approach

is known as Interpretable Multi-head Attention [15]

$$\text{InterpretableMultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \tilde{\mathbf{H}}\mathbf{W}_{\tilde{\mathbf{H}}},$$

where

$$\tilde{\mathbf{H}} = \frac{1}{m_H} \sum_{h=1}^{m_H} \text{Attention}(\mathbf{Q}^{(h)}, \mathbf{K}^{(h)}, \mathbf{V}),$$

and $\mathbf{W}_{\tilde{\mathbf{H}}}$ is applied as a final linear projection.

2.2 Temporal Fusion Transformer

Figure 3 shows a high-label architecture of TFT, with individual layers explained in detail in the subsequent section.

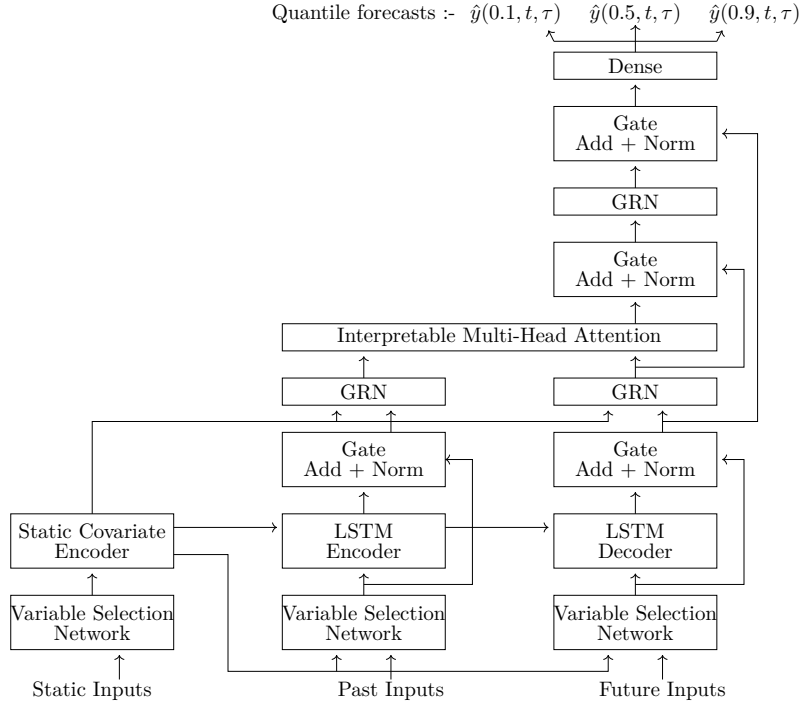


Figure 3: TFT architecture. TFT processes three types of inputs: static inputs, time-dependent past inputs, and prior known future inputs. The gated residual network facilitates the flexibility of information either through skip connections or via gated linear unit layers. The variable selection network dynamically identifies the most valuable features from the input data. LSTM layers capture local sequential dependencies, while interpretable multi-head attention enables the combining of information across all time steps.

A. Locality Enhancement with Sequence-to-Sequence Layer

In time series data, important events such as anomalies or cyclical patterns are best understood in the context of surrounding values. Locality enhancement refers to the extraction of local patterns using the same filter uniformly across all times. The following describes the process of locality enhancement for input time series data using a sequence-to-sequence layer.

For outputs $\tilde{\xi}_{t-k:t}$ from variable selection network, corresponding to past inputs, are passed

through an LSTM [38] encoder, while the outputs $\tilde{\xi}_{t+1:t+\tau_{\max}}$ from variable selection network, corresponding feature inputs, are passed through LSTM decoder. The cell state and hidden state of the first LSTM in the layer are initialized using the context vectors \mathbf{c}_c and \mathbf{c}_e respectively, which are obtained from static covariate encoders. The outputs from this layer are denoted as $\phi(t, -k), \dots, \phi(t, \tau_{\max})$. The final outputs of this layer are derived using Gated Linear Units (GLUs), applied through a residual connection followed by layer normalization

$$\tilde{\phi}(t, n) = \text{LayerNorm} \left(\tilde{\xi}_{t+n} + \text{GLU}(\phi(t, n)) \right), \quad \text{where } n \in [-k, \tau_{\max}].$$

B. Static Enrichment Layer

Temporal dynamics are significantly influenced by static metadata, and the static enrichment layer enhances these temporal features. Specifically, The static enrichment layer applies a GRN to the output locality enhancement $\tilde{\phi}(t, n)$, along with context vector \mathbf{c}_e from the static covariate encoder

$$\theta(t, n) = \text{GRN}(\tilde{\phi}(t, n), \mathbf{c}_e), \quad \text{where } n \in [-k, \tau_{\max}].$$

C. Temporal Self-Attention Layer

The long-range dependencies in the TFT model are efficiently captured by the self-attention layer. The layer operates as follows. Let $\Theta(t) = [\theta(t, -k), \dots, \theta(t, \tau_{\max})]^T$ denote the matrix formed by stacking the outputs of the static enrichment layer. Subsequently, an interpretable multi-head attention mechanism is applied to $\Theta(t)$

$$\mathbf{B}(t) = \text{InterpretableMultiHead}(\Theta(t), \Theta(t), \Theta(t)),$$

where $\mathbf{B}(t) = [\beta(t, -k), \dots, \beta(t, \tau_{\max})]$ represents the output of the interpretable multi-head attention mechanism. A gating layer (GLU) is also included as the final component of this Layer to improve training efficiency

$$\delta(t, n) = \text{LayerNorm}(\theta(t, n) + \text{GLU}(\beta(t, n))), \quad \text{where } n \in [-k, \tau_{\max}].$$

D. Position-wise Feed-forward Layer

In this layer, a non-linear module GRNs is applied to the outputs of the temporal self-attention layer

$$\psi(t, n) = \text{GRN}(\delta(t, n)).$$

Additionally, a gated (GLU) residual connection is included via a direct pathway to the sequence-to-sequence layer

$$\tilde{\psi}(t, n) = \text{LayerNorm} \left(\tilde{\phi}(t, n) + \text{GLU}(\psi(t, n)) \right), \quad \text{where } n \in [-k, \tau_{\max}].$$

E. Quantile Outputs

In many real-world cases, instead of predicting a single point estimate, providing prediction intervals is valuable for optimizing decision-making and managing risk, as it captures the likely best- and worst-case outcomes that the target variable can take. This is achieved through quantile forecasting. Quantile forecasts are generated by applying linear transformations to the output of

the position-wise feed-forward layer

$$\hat{y}(q, t, \tau) = \mathbf{W}_q \tilde{\psi}(t, \tau) + b_q,$$

where \mathbf{W}_q , b_q are the learnable coefficients corresponding to the specified quantile q , and $\tau \in [1, \tau_{\max}]$, since forecasts are only interested for future time steps.

3 VARIATIONAL QUANTUM ALGORITHM

Variational Quantum Algorithms (VQAs) are hybrid quantum-classical frameworks that leverage quantum properties such as superposition and entanglement to enhance the efficiency of solving optimization tasks. VQAs are considered parameterized quantum circuits or variational circuits, designed to train the circuit parameters iteratively according to the given optimization task. A VQA typically consists of four core components: an encoding layer $\mathbf{U}(\mathbf{x})$, a parameterized layer $\mathbf{V}(\boldsymbol{\theta})$, a cost function \mathbf{C} , and an optimizing procedure to update the parameters $\boldsymbol{\theta}$. Figure 4 illustrates the generic architecture of a Variational Quantum Algorithm (VQA).

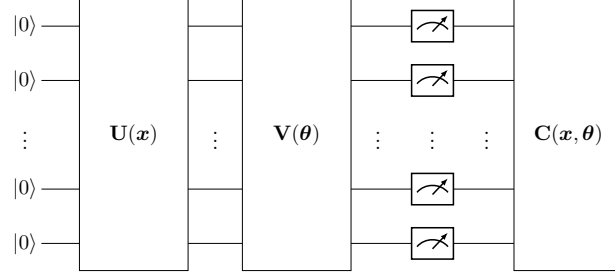


Figure 4: Generic architecture of Variational Quantum Algorithm (VQA). The block $\mathbf{U}(\mathbf{x})$ denotes the data encoding circuit, where \mathbf{x} is the input data. This is followed by the parameterized quantum circuits of variational circuit block $\mathbf{V}(\boldsymbol{\theta})$, which consists of trainable parameters $\boldsymbol{\theta}$. After, a quantum measurement operation is performed on all qubits. Finally, the cost function $\mathbf{C}(\mathbf{x}, \boldsymbol{\theta})$ is evaluated.

Classical information are first encoded into a quantum state via state preparation routine or feature map [39]. The choice of the feature map depends on the specified problem, as it significantly influences model performance and convergence speed. Notably, this feature map is neither trained nor optimized during training [40]. Here in Figure 5, we present two feature maps widely used in quantum machine learning: the AngleEmbedding [41] and ZZFeatureMap [42].

Once the classical data is encoded in the quantum device, a parametrized circuit [26, 43, 44] is applied to it. The parametrized circuit is the main component of VQAs, enabling them to learn and adapt during the optimization iteration. A parametrized circuit consists of the quantum gates - such as $\mathbf{R}_x, \mathbf{R}_y, \mathbf{R}_z$ - whose parameters are learnable during iterations. These gates, when combined with quantum phenomena like superposition, and entanglement between qubits, enable the circuit to capture model complex functions and optimize performance over successive iterations. Figure 6 and Figure 7 are two examples of quantum parametrized circuits - Basic Entangler layers [41] and N-local circuit [42] - commonly used in several variational quantum algorithms.

At this stage, classical information is extracted from the quantum circuit through a quantum measurement operation of a subset (or all) qubits of the circuit. Measurement is an important task of a quantum system, and Qiskit provides two primitives that can help to measure: Sampler, and Estimator [42].

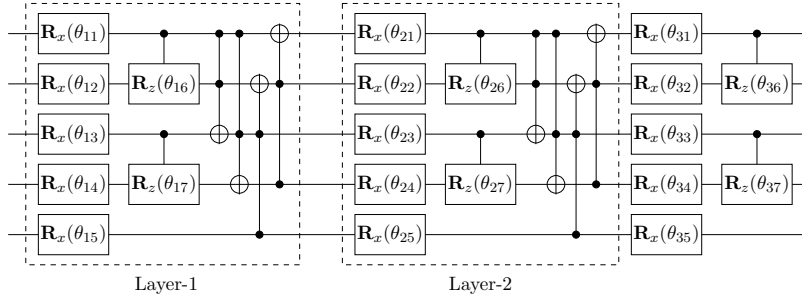
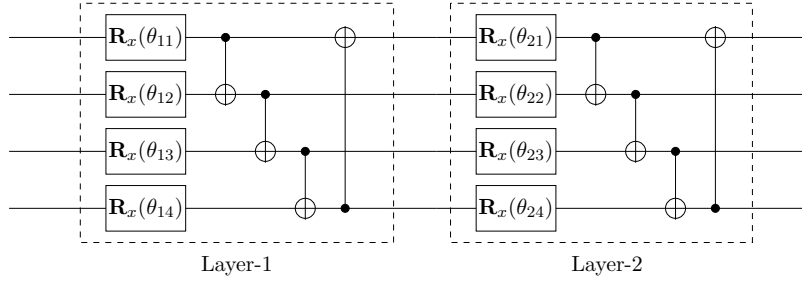
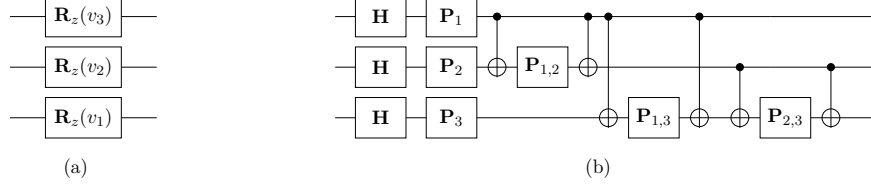


Figure 7: Diagram of N-local circuits. Each layer (dashed box) consists of Rotation blocks formed by \mathbf{R}_x and \mathbf{CR}_Z gates, followed by entanglement blocks formed by Toffoli gates.

The sampler primitives calculate the probability of each computational basis state $|k\rangle$, given a quantum circuit that prepares a quantum state $|\psi\rangle$. Specifically, It calculates

$$P_k = |\langle k|\psi\rangle|^2,$$

here P_k denotes the probability of measuring the quantum state $|k\rangle$.

The estimator primitives introduce a different notion called the observable $\tilde{\mathbf{H}}$, which is a Hermitian linear operator. Estimator primitives calculate the expectation value of $\tilde{\mathbf{H}}$ for the quantum state $|\psi\rangle$. Let $|\lambda\rangle$ be one of the eigenvector of the observable $\tilde{\mathbf{H}}$ with corresponding eigenvalue λ ,

then the observable probabilities are determined as: $P_\lambda = |\langle \lambda | \psi \rangle|^2$. The expectation value of the observable $\tilde{\mathbf{H}}$ with respect to a quantum state $|\psi\rangle$ is defined as the weighted sum of its eigenvalues λ , where each weight corresponds to the observable probability P_λ

$$\langle \tilde{\mathbf{H}} \rangle_\psi = \langle \psi | \tilde{\mathbf{H}} | \psi \rangle = \sum_\lambda P_\lambda \lambda.$$

The outcomes of this measurement are then fed into a cost function, defined by the optimization model. This cost function evaluates the performance of the parameterized quantum circuit and guides the update of its parameters during training. Based on the cost function, an optimization algorithm - either gradient-based or gradient-free is applied to minimize or maximize the objective. This process updates the parameters of the quantum circuit, which is then executed iteratively until convergence or for a fixed number of epochs. After completing the iterative steps, the quantum circuit is considered optimized for the given model and produces an approximate optimal solution.

One of the most important advantages of VQA is their robustness against quantum noise [45–47], making them suitable for implementation on today’s Noisy Intermediate-Scale Quantum (NISQ) devices. VQAs have been successfully applied across various domains in machine learning and artificial intelligence, including classification [43, 44, 48, 49], generative modeling [50], deep reinforcement learning [51], and transfer learning [52].

4 QUANTUM TEMPORAL FUSION TRANSFORMER

In this paper, we extend the classical Temporal Fusion Transformer (TFT) model into the Quantum Temporal Fusion Transformer (QTFT) model by replacing and appropriately modifying classical learning components within the TFT cell with VQCs.

There are three main components responsible for extracting the pattern from the datasets: Gated Residual Networks (GRNs), Long Short Term Memory (LSTM), and Interpretable Multi-head Attention Mechanism. In this section, we focus on two key components: Gated Residual Network and Interpretable Multi-head Attention Mechanisms, including all their associate sub-component. We are not focused on Long Short Term Memory in this work, as the Quantum Long Short Term Memory (QLSTM) has already been introduced [53].

4.1 Variational Quantum Circuit for QTFT

In this section, we present a fixed variational quantum circuit used within the learning components of Gated Residual Networks (GRNs) and Interpretable Multi-head Attention Mechanism. See Figure 8 for a schematic diagram of the Variational Quantum Circuit for QTFT.

There are various quantum simulator software platforms, such as PennyLane [41] and IBM Qiskit [42], that allow for calculating numerical evaluation of the quantum circuit on a classical computer. In contrast, real quantum computers estimate these values through statistical sampling obtained from iterative measurements.

A. Encoding Layer

Before performing any quantum computation within a quantum circuit, it is important to encode classical data into quantum states. This is achieved through an Encoding layer, the pre-defined technique or method to encode the classical data into the corresponding quantum state. Let n be

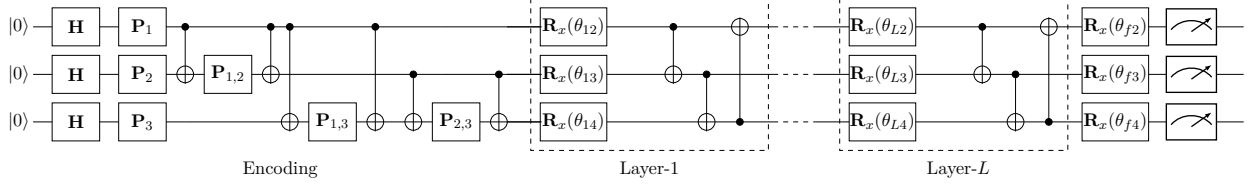


Figure 8: VQC architecture for QTFT model. It consists of three layers: the data encoding layer (\mathbf{R}_z), variational circuit layers (dashed boxes), and the quantum measurement layer (meter symbol). Now, the number of qubits and measurements depends on the problem of interest. Also, the variational circuit, the dashed boxes, can be adopted according to the accuracy of the result by increasing the number of layers of the circuit, enabling the mode to capture more complex patterns effectively.

the number of qubits in a quantum system. Then, any quantum state $|\phi\rangle$ can be expressed as

$$|\phi\rangle = \sum_{i=0}^{2^n-1} \alpha_i |i\rangle,$$

where $\alpha_i \in \mathbb{C}$ represents the complex amplitudes associated with the computational basis state $|i\rangle$, where the index i denotes the decimal representation of the bit-string. The square of the amplitude α_i is the probability of measuring the quantum state in the basis state $|i\rangle$. These amplitudes must satisfy the normalization condition

$$\sum_{i=0}^{2^n-1} |\alpha_i|^2 = 1.$$

Encoding layers implement a systematic method to embed a classical vector $\mathbf{v} = (v_1, v_2, \dots, v_n)$ into a quantum state by mapping its values v_j to the amplitudes α_i corresponding to a quantum state $|\phi\rangle$.

Here we use the ZZ Feature Map, an encoding scheme in which a classical input vector transforms into a quantum state. In the paper [48], the authors Havlíček et al. introduce the fundamental concept of ZZ Feature Map. The circuit corresponding to the encoding technique is defined by the following unitary operator

$$U(\mathbf{v}) = \exp \left(i \sum_{j=1}^n v_j Z_j + i \sum_{j < k} \psi(v_j, v_k) Z_j Z_k \right),$$

where ψ be an non-liner function and Pauli- Z_j denoted as Pauli-Z operator on the j -th qubit. The first term applies Z rotations encoding the features linearly as $\exp(iv_j Z_j)$, while the second term applies ZZ entangling rotations as $\exp(i\psi(v_j, v_k) Z_j Z_k)$. Below, we describe a specific variant of the ZZ Feature Map.

The first step is to create an equal superposition of all basis states from the initial state $|0\rangle^{\otimes n}$ using the Hadamard gate

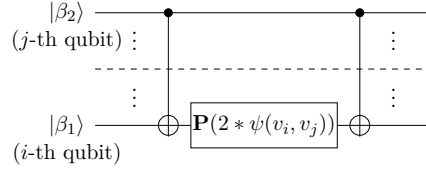
$$H(|0\rangle^{\otimes n}) = \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} |i\rangle.$$

There are two major components in the ZZ Feature Map : a phase gate \mathbf{P} , define as

$$\mathbf{P}(\lambda) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\lambda} \end{pmatrix},$$

where $\lambda \in \mathbb{R}$ called rotation angle, and a classical non-linear function ψ , which typically defaults to $\psi(x) = x$ for single-variable inputs and $\psi(x, y) = (\pi - x)(\pi - y)$ for pairwise interactions. Each qubit j , after the application of Hadamard gate, is transformed by phase gate with a rotational angle $2 * \psi(v_j)$, where v_j be the j -th component of the input vector \mathbf{v} .

We present a quantum routine that is repeatedly applied within the ZZ Feature Map. Let v_i and v_j denote the i -th and the j -th component of the inputs vector \mathbf{v} . For each such pair (v_i, v_j) , the routine applies the following sequence of quantum operation: two CNOT gates with target qubits j and control qubit i , and in between two CNOT gate applies a phase gate with an angle $2 * \psi(v_i, v_j)$ to the j -th qubit. Here is the Figure.



The outputs of phase gates are passed through the above quantum routine in the following sequential order: $(v_1, v_2), \dots, (v_1, v_n), (v_2, v_3), \dots, (v_2, v_n), \dots, (v_k, v_{k+1}), \dots, (v_k, v_n), \dots, (v_{n-1}, v_n)$.

But why do we use ZZ Feature Map as a data encoding scheme in variational circuits? As noted in reference [48, 54], this feature map offers several key advantages that leverage the computational power of the variational circuit, particularly in the context of machine learning tasks. The ZZ Feature Map provides nonlinear data encoding by mapping the data into a high-dimensional space. Its structure enables the exploration of a larger portion of the Hilbert space, allowing it to capture more complex relationships within the data. Sometimes it also provides a better starting point for the variational layer.

B. Variational Circuit Layer

The encoded data, in the form of the quantum state, is then passed through a series of quantum unitary operators. In this variational quantum algorithm setup, we employ N-local circuits [55] as the variational circuit or ansatz. A general structure for N-local circuits is described as follows. These quantum unitary operators consist of several single-qubit rotation gates and controlled-NOT (CNOT) gates. Single-qubit rotation \mathbf{R}_y gates are applied to each qubit with the rotational angle parameters $\theta_{(\cdot)}$. Rotational angles are not predetermined; instead, they are iteratively updated during the optimization process using the gradient-decent method. To generate multi-qubit entanglement, the outputs of rotation gates are passed through CNOT gates, which are implemented between two consecutive qubits in cycle order: $(1, 2, \dots, n-1, 1)$. A combination of rotation gates and CNOT gates is referred to as a layer denoted as a dashed box in Figure 8. The layers are elegantly formulated as

$$\bigotimes_{i=1}^n \mathbf{R}_y(\theta_i) \prod_{(i,j)} \text{CNOT}(i, j).$$

Depending on the problem's complexity, the layer may be repeated several times to increase the

parameter of the circuit, effectively capturing the more complex pattern of the dataset. At the end of all layers, a final rotation layer consisting of \mathbf{R}_y gates is appended.

However, repeating the layers of the variational circuit increases the depth of the quantum circuit which in turn affects the complexity and resource requirements of the quantum hardware. According to the problem and the limitations of current quantum hardware, it is important to optimize the depth of the circuit to produce the best possible result.

There are two key reasons for using N-local circuits as variation circuits: efficient implementation and the ability to capture important correlations. N-local circuits are composed of simple local gates that can be implemented on quantum hardware using a small number of qubits. Moreover, these circuits can capture the important correlation between the qubits in a quantum system, as the local gates can act on neighboring qubits and generate entanglement between them.

C. Measurement Layer

At the end of the VQC, a quantum measurement layer is added to extract quantum information for further post-processing on a classical computer. In our variational quantum setup, we use a fixed, hardware-efficient Pauli observable [56] - the Pauli-Z operator - as the measurement tool. The variational circuit is measured by applying the Pauli-Z observable independently to each qubit. Specifically, for i -th qubit (where $i = 1, 2, 3, \dots, n$), the observable is given by

$$Z_i = I^{\otimes(i-1)} \otimes Z \otimes I^{\otimes(n-i)}.$$

Let the quantum state after the variational circuit layer be denoted as $|\zeta\rangle$. We now demonstrate the calculation of measurement value by applying the Pauli-Z observable on the 0-th qubit, while the approach for calculating measurements on the remaining qubits follows the same. The quantum state $|\zeta\rangle$ can be expressed in computational basis as

$$\begin{aligned} |\zeta\rangle &= \sum_{i=0}^{2^n-1} \gamma_i |i\rangle \\ &= |0\rangle \left(\sum_{i=0}^{2^{(n-1)}-1} \gamma_i |i\rangle \right) + |1\rangle \left(\sum_{i=0}^{2^{(n-1)}-1} \gamma_{(2^{(n-1)}+i)} |i\rangle \right), \end{aligned}$$

where $\gamma_i \in \mathbb{C}$ and $\sum_{i=0}^{2^n-1} |\gamma_i|^2 = 1$. The eigenvalues and eigenvectors of the observable Pauli-Z are 1, -1 and $|0\rangle$, $|1\rangle$ respectively, i.e., $\lambda_1 = 1$, $\lambda_2 = -1$ and $|\lambda_1\rangle = |0\rangle$, $|\lambda_2\rangle = |1\rangle$. The probability, P_1 and P_{-1} of measuring the quantum state $|0\rangle$ and $|1\rangle$ are given by

$$\begin{aligned} P_1 &= |\langle 0|\zeta\rangle|^2 = \left| \sum_{i=0}^{2^{(n-1)}-1} \gamma_i |i\rangle \right|^2 = \sum_{i=0}^{2^{(n-1)}-1} |\gamma_i|^2, \\ P_{-1} &= |\langle 1|\zeta\rangle|^2 = \left| \sum_{i=0}^{2^{(n-1)}-1} \gamma_{(2^{(n-1)}+i)} |i\rangle \right|^2 = \sum_{i=0}^{2^{(n-1)}-1} |\gamma_{(2^{(n-1)}+i)}|^2. \end{aligned}$$

The expectation value of the Pauli-Z observable corresponding to the 0-th qubit is given as follows

$$\begin{aligned}\langle \zeta | Z_0 | \zeta \rangle &= P_{\lambda_1} \lambda_1 + P_{\lambda_2} \lambda_2 \\ &= \sum_{i=0}^{2^{(n-1)}-1} |\gamma_i|^2 - \sum_{i=0}^{2^{(n-1)}-1} |\gamma_{(2^{(n-1)}+i)}|^2.\end{aligned}$$

4.2 Components

The primary object of the QTFT model efficiently transform key subroutines of TFT into quantum counterparts that leverage quantum computational advantage. We discuss this transformation in detail below.

A. Quantum Gated Residual Network

In the classical part of Section 2, we have already discussed the significance of the Gated Residual Network (GRN) in detail. In this section, we will not revisit its structure; instead, we only focus on how this structure is adapted into a quantum form to improve the model's performance. In the previous section, we explored how classical neural network components (dense layers) can be replaced or alternated by quantum counterparts using Variation Quantum Algorithms (VQAs). Here we utilize VQAs as the foundational building block of a Quantum Gated Residual Network (QGRN).

Let \mathbf{a} and \mathbf{c} denote the primary input and optional context input, respectively (\mathbf{c} derived from quantum static covariate encoder). Fast, both the primary input \mathbf{a} and the optional input \mathbf{c} are plugged into the ZZ Feature map (denoted as ZZFeatureMap) to encode the classical data into quantum state $|\mathbf{a}\rangle$, $|\mathbf{c}\rangle$ respectively

$$\begin{aligned}|\mathbf{a}\rangle &= \text{ZZFeatureMap}(\mathbf{a}), \\ |\mathbf{c}\rangle &= \text{ZZFeatureMap}(\mathbf{c}).\end{aligned}$$

Next, two quantum states $|\mathbf{a}\rangle$ and $|\mathbf{c}\rangle$ are passed independently through two separate variational circuits known as N-local circuits (denoted as NLocal). These circuits consist of parametrized quantum gates, in which parameters (or weights) are trainable during learning iteration. The resulting quantum states are present as $|\mathbf{a}'\rangle$ and $|\mathbf{c}'\rangle$

$$\begin{aligned}|\mathbf{a}'\rangle &= \text{NLocal}_{\mathbf{a}'}(|\mathbf{a}\rangle), \\ |\mathbf{c}'\rangle &= \text{NLocal}_{\mathbf{c}'}(|\mathbf{c}\rangle),\end{aligned}$$

where the subscript \mathbf{a}' denotes the trainable parameters associated with this particular entangler layer. At the end of the quantum circuits, quantum measurement operations are performed on quantum states $|\mathbf{a}'\rangle$ and $|\mathbf{c}'\rangle$ to extract classical information. This is done by computing the expectation values concerning the Pauli-Z observable on each qubit

$$\begin{aligned}\mathbf{a}'' &= \langle \mathbf{a}' | \mathbf{Z} | \mathbf{a}' \rangle = \text{expval}(\text{PauliZ}(|\mathbf{a}'\rangle)), \\ \mathbf{c}'' &= \langle \mathbf{c}' | \mathbf{Z} | \mathbf{c}' \rangle = \text{expval}(\text{PauliZ}(|\mathbf{c}'\rangle)),\end{aligned}$$

where $\text{expval}(\text{PauliZ}(|\mathbf{k}\rangle))$ denote the expectation values concerning the Pauli-Z observable corresponding qubit $|\mathbf{k}\rangle$. These expectation values are classical outputs further processed in hybrid classical-quantum architecture. The classical two outputs obtained from quantum measurement,

\mathbf{a}'' and \mathbf{c}'' are first added, followed by the ELU activation function to introduce non-linearity as

$$\boldsymbol{\eta}_1 = \text{ELU}(\mathbf{a}'' + \mathbf{c}'').$$

This activated vector $\boldsymbol{\eta}_1$ is then encoded into the quantum state back using the ZZ Feature map for subsequence quantum processing

$$|\boldsymbol{\eta}_1\rangle = \text{ZZFeatureMap}(\boldsymbol{\eta}_1).$$

Another variational quantum circuit, N-local circuits, is applied to the quantum state $\boldsymbol{\eta}_1$ without performing any intermediate measurement operations

$$|\boldsymbol{\eta}_2\rangle = \text{NLocal}_{\boldsymbol{\eta}_2}(|\boldsymbol{\eta}_1\rangle).$$

Now we introduce Quantum Gated Linear Unit (QGLU), a quantum analog of the classical Gated Linear Unit (GLU). Let $\boldsymbol{\gamma}$ be the input of Quantum Gated Linear Unit (QGLU). If $\boldsymbol{\gamma}$ is classical vector, it is first encoded into a quantum state using the ZZ Feature Map. However, if the input is already in a quantum state, this encoding step is omitted. Next, the output of the feature map $|\boldsymbol{\gamma}\rangle$ passed through two distinct variational quantum circuits, both implemented using N-local circuits

$$\begin{aligned} |\boldsymbol{\gamma}\rangle &= \text{ZZFeatureMap}(\boldsymbol{\gamma}), \\ |\boldsymbol{\gamma}'\rangle &= \text{NLocal}_{\boldsymbol{\gamma}'}(|\boldsymbol{\gamma}\rangle), \\ |\boldsymbol{\gamma}''\rangle &= \text{NLocal}_{\boldsymbol{\gamma}''}(|\boldsymbol{\gamma}\rangle). \end{aligned}$$

Quantum measurement operations are applied to the quantum states $|\boldsymbol{\gamma}'\rangle$ and $|\boldsymbol{\gamma}''\rangle$

$$\begin{aligned} \boldsymbol{\gamma}' &= \langle \boldsymbol{\gamma}' | \mathbf{Z} | \boldsymbol{\gamma}' \rangle = \text{expval}(\text{PauliZ}(|\boldsymbol{\gamma}'\rangle)), \\ \boldsymbol{\gamma}'' &= \langle \boldsymbol{\gamma}'' | \mathbf{Z} | \boldsymbol{\gamma}'' \rangle = \text{expval}(\text{PauliZ}(|\boldsymbol{\gamma}''\rangle)). \end{aligned}$$

The final output of the Quantum Gated Linear Unit (QGLU) is computed using an element-wise multiplication between one of the sigmoid-activated outputs and another

$$\boldsymbol{\gamma}''' = \text{QGLU}(|\boldsymbol{\gamma}\rangle) = \sigma(\boldsymbol{\gamma}') \odot \boldsymbol{\gamma}'',$$

where $\sigma(\cdot)$ denote as sigmoid activation function and \odot is the Hadamard product.

In the final output of Quantum Gated Residual Network (QGRN), a residual connection is established between the output of the Quantum Gated Linear Unit (QGLU) and primary input, followed by layer normalization

$$\text{QGRN}(\mathbf{a}, \mathbf{c}) = \text{LayerNorm}(\mathbf{a} + \text{QGLU}(|\boldsymbol{\eta}_2\rangle)).$$

B. Quantum Variable Selection Network And Quantum Static Covariate Encoders

Now interesting fact that variable selection network and static covariate encoders are building upon GRN. Now we construct QGRN in the previous section, if we replace the GRN in the place of GRN in both variation selection network and static covariate encoders then we get corresponding quantum variation selection network and quantum static covariate encoders.

C. Quantum Interpretable Multi-head Attention

In Section 2, we have already discussed the attention mechanism and how its modified version, called interpretable multi-head attention, efficiently improves the performance of the model. In this section, we do not go through all the details; we focus only on building the architecture of interpretable multi-head attention within a quantum framework. The key components of the attention model are the learning parameters derived from three matrices: the query, key, and value metrics. A major problem in the classical model is efficiently learning and managing these large-scale parameters. The VQAs provide a quantum approach that can handle such parameters more effectively, potentially reducing computational overhead and improving learning efficiency. Below, we describe an approach for integrating VQAs into interpretable multi-head attention.

Let \mathbf{S} be the input of the attention mechanism, and let m_H represent the number of attention heads. If \mathbf{S} is provided as a matrix, its rows are processed one at a time. Each classical input row is first encoded into a quantum state by the ZZ Feature map

$$|\mathbf{S}\rangle = \text{ZZFeatureMap}(\mathbf{S}),$$

where we denote $|\mathbf{S}\rangle$ as the quantum states generated corresponding to all input rows. We implement quantum variation circuits using N-local circuits to construct the query, key, and value. From the input quantum state, we construct m_H number of distinct query, and key states, and value state as follows

$$\begin{aligned} |\mathbf{Q}^{(h)}\rangle &= \text{NLocal}_{Q^{(h)}}(|\mathbf{S}\rangle), \\ |\mathbf{K}^{(h)}\rangle &= \text{NLocal}_{K^{(h)}}(|\mathbf{S}\rangle), \\ |\mathbf{V}\rangle &= \text{NLocal}_V(|\mathbf{S}\rangle), \end{aligned}$$

for $h = 1, 2, \dots, m_H$. This is the key aspect of the attention mechanism, where the parameters - specifically, the rotational angles in the variational circuit - are configured and optimized during training. To extract classical information from the quantum states of queries, keys, and values, apply quantum measurement operations with respectively Pauli-Z observable

$$\begin{aligned} \mathbf{Q}^{(h)} &= \langle \mathbf{Q}^{(h)} | \mathbf{Z} | \mathbf{Q}^{(h)} \rangle = \text{expval}(\text{PauliZ}(|\mathbf{Q}^{(h)}\rangle)), \\ \mathbf{K}^{(h)} &= \langle \mathbf{K}^{(h)} | \mathbf{Z} | \mathbf{K}^{(h)} \rangle = \text{expval}(\text{PauliZ}(|\mathbf{K}^{(h)}\rangle)), \\ \mathbf{V} &= \langle \mathbf{V} | \mathbf{Z} | \mathbf{V} \rangle = \text{expval}(\text{PauliZ}(|\mathbf{V}\rangle)), \end{aligned}$$

for $h = 1, 2, \dots, m_H$. From this point onward, the Quantum Interpretable Multi-Head Attention mechanism operates analogously to its classical counterpart. It shares the same value \mathbf{V} across all heads while employing distinct query $\mathbf{Q}^{(\cdot)}$ and key $\mathbf{K}^{(\cdot)}$ projections for each head. The final output is obtained through additive aggregation of the attention outputs from all heads

$$\text{QuantumInterpretableMultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \frac{1}{m_H} \sum_{h=1}^{m_H} \text{Attention}(\mathbf{Q}^{(h)}, \mathbf{K}^{(h)}, \mathbf{V}).$$

The attention mechanism employed is identical to that used in the classical attention model. In classical models, a final linear projection is typically applied at the output. However, in our approach, we omit this projection since the variational circuit already contains enough number of learnable parameters to model the target function sufficiently.

4.3 QUANTUM MODEL ARCHITECTURE

This section explicitly discusses the Quantum Temporal Fusion Transformer (QTFT) architecture compared to its classical counterpart. As in the classical model, the Quantum Temporal Fusion Transformer (QTFT) also processes three kinds of input: static inputs, past inputs, and prior known future inputs.

First, the static input passes through Quantum Variable Selection Networks, followed by Quantum Static Covariate Encoder, which produces three context input vectors. Past inputs and prior known future inputs are also processed through the Quantum Variable Selection Networks, guided by one context vector, that derives from the Quantum Static Covariate Encoder.

The outputs of the Quantum Variable Selection Networks corresponding to the past inputs are passed through the LSTM Encoder, while those corresponding to the future inputs pass through the LSTM Decoder. The cell state and hidden state of the first LSTM in the layer are initialized using the context vector derived from the Quantum Static Covariate Encoder. Rather than using a classical LSTM, we replace it with a Quantum Long Short-Term (QLSTM) memory [53]. However, to ensure a fair comparison between our proposed subroutines - Quantum GRN and Quantum Interpretable Multi-Head Attention - and their classical counterparts, we retain the classical LSTM as the base architecture.

The final outputs of this layer are obtained using Quantum Gated Linear Units (QGLUs) and applied through a residual connection followed by layer normalization.

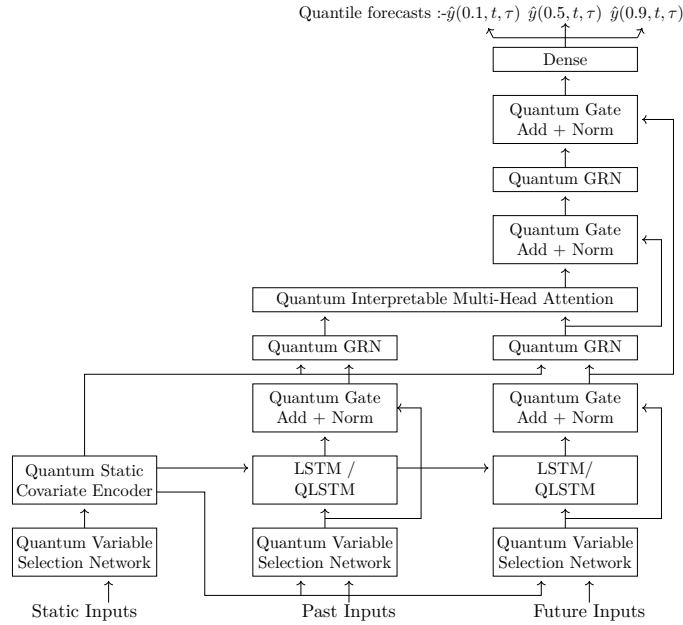


Figure 9: QTFT architecture. QTFT processes three types of inputs: static inputs, time-dependent past inputs, and prior known future inputs. In this architecture, all classical components - including the variable selection network, static covariate encoder, gating layer, gated residual network, and interpretable multi-head attention - are systematically and efficiently transformed into quantum subroutines.

Before applying Quantum Interpretable Multi-Head Attention, the output of Quantum Gated Layer Units is passed through a Quantum Gated Residual Network together with the last context

vector from the Quantum Static Covariate Encoder.

Both outputs of the Quantum Gated Residual Network, corresponding past inputs and future inputs are fed into the Quantum Interpretable Multi-Head Attention, followed by Quantum Gated Layer Units with residual connection and layer normalization.

The outputs of Quantum Gated Layer Units corresponding to future inputs are attached through a Quantum Gated Residual Network, similar to the previous layer, followed by Quantum Gated Layer Units with residual connection and layer normalization.

Finally, quantile forecasts are obtained by applying dense layers to the outputs of Quantum Gate Layers Units.

4.4 OPTIMIZATION PROCEDURE

The proposed architecture is a quantum circuit-based model, where each component is represented by a quantum circuit. This raises a question - how can we optimize these circuits to achieve the best possible result? As in the classical optimization process, we use the gradient-based method to optimize the quantum circuits. Specifically, we utilize the parameter-shift rule [41, 57], which enables the analytical computation of the gradient of the quantum circuits concerning their tunable parameters.

We are not going through the details of all the quantum circuit optimization procedures; instead, we illustrate a general quantum circuit optimization framework. The used quantum circuits in our architecture follow a similar structure, expecting primarily in their inputs, variational circuits, and measurement configurations.

Let \mathbf{x} denote the input data, $\mathbf{U}(\mathbf{x})$ represent the data encoding unitary, and $\mathbf{V}(\boldsymbol{\theta})$ be a variational circuit block, which consists of trainable parameters $\boldsymbol{\theta}$. Then the expectation value of an observable $\tilde{\mathbf{H}}$ is given by

$$\begin{aligned}\langle \tilde{\mathbf{H}} \rangle_{\mathbf{x}, \boldsymbol{\theta}} &= \langle \mathbf{U}^\dagger(\mathbf{x}) \mathbf{V}^\dagger(\boldsymbol{\theta}) | \tilde{\mathbf{H}} | \mathbf{V}(\boldsymbol{\theta}) \mathbf{U}(\mathbf{x}) \rangle \\ &= \langle 0 | \mathbf{U}^\dagger(\mathbf{x}) \mathbf{V}^\dagger(\boldsymbol{\theta}) \tilde{\mathbf{H}} \mathbf{V}(\boldsymbol{\theta}) \mathbf{U}(\mathbf{x}) | 0 \rangle.\end{aligned}$$

It can be shown [26] that gradient of the function $\tilde{\mathbf{H}}$ with respect to $\boldsymbol{\theta}$ is given by

$$\frac{\partial \langle \tilde{\mathbf{H}} \rangle_{\mathbf{x}, \boldsymbol{\theta}}}{\partial \boldsymbol{\theta}} = \frac{1}{2} \left[\langle \tilde{\mathbf{H}} \rangle_{\mathbf{x}, \boldsymbol{\theta} + \frac{\pi}{2}} - \langle \tilde{\mathbf{H}} \rangle_{\mathbf{x}, \boldsymbol{\theta} - \frac{\pi}{2}} \right].$$

Hence, it is proven that the gradient of the expectation values is evaluated analytically using the above equation. By combining this approach with classical machine learning gradient descent optimization, we obtained a quantum-based machine learning gradient descent optimization process and used it in our implementation.

5 EXPERIMENTS AND RESULTS

This section presents a comparative analysis of the QTFT and its classical counterpart, focusing on their respective capabilities and performance. We conduct numerical simulations of our proposed QTFT architecture under various scenarios. Specifically, we present experimental results of multi-horizontal time series forecasting across various time series datasets by using the QTFT model.

We implemented the classical TFT model using the PyTorch [58] framework. For the simulation of quantum circuits in the QTFT model, we use PennyLane [41], while the overall architecture is built using the same framework PyTorch as in the classical model. To ensure fair competition, we use the same structure for both the classical as well as quantum TFT models, including the cost function, fixed parameters, and the total number of trainable parameters.

Stock Market Prediction

We first investigate the capability of our QTFT’s in learning stock market prediction. In this section, we pick the stock market data of AXIS BANK and analyze it using the QTFT model to forecast an important feature. The AXIS BANK data was collected from India National Stock Exchange (NSE) of India. It represents a NIFTY-50 Stock Market Data record covering the years 2000 to 2021. The data set contains 5306 rows or feature vectors. The 15 columns (features) describe various aspects of the stock data: date, symbol, series, previous close, open, high, low, last, close, vwap, turnover, trades, deliverable volume, and deliverable percent.

Due to the limited number of available qubits and the inherent noise in current NISQ quantum devices, we do not use all the feature vectors and features for training and testing our QTFT model. Instance we select the first 10 feature vectors starting from the year 2000 and focus on 4 input features: Open, High, Low, and Last with the Close price as the target feature. We are not concerned with the data types - whether static, observed inputs or known inputs - because we are working on a small data set. If we further divide the dataset into these categories, it would result in feature vectors that lack sufficient value to extract the relationships between the input variables. The intention for using these selected feature vectors and features lies in their typically small numerical values, which facilitate more efficient QTFT model training and testing.

Similarly, due to the limitations of current quantum hardware, we do not use the original loss function used in the TFT paper [15]. Instead, we use a simplified yet similar kind of loss function that efficiently calculates the loss for further optimization produce. Specifically, we employ the quantile loss function, defined as is

$$\mathcal{L}_q(y, \hat{y}) = \frac{1}{m} \sum_{i=1}^m \max((q-1)(y_i - \hat{y}_i), q(y_i - \hat{y}_i)),$$

where y_i denotes the true value of the i -th data point, \hat{y}_i represents the corresponding predicted value, $q \in (0, 1)$ specifies the target quantile for estimation, and m indicates the total number of data points in the dataset.

The fixed hyperparameters used in both the use in classical and quantum TFT models are as follows: quantile $q = 0.5$, learning Rate = 0.1, number epochs = 100, input window: past steps = 10, forecast steps = 5, training data range = 0 to 19, test data range = 20 to 26.

In the classical temporal fusion transformer, the configuration includes an LSTM with a hidden layer of size 1, a hidden dimension of 2, and 190 trainable parameters. In contrast, for the quantum temporal fusion transform, we construct two models, one that employs classical LSTM, and another that incorporates a Quantum Long Short Term Memory (QLSTM). For input encoding, use Angle Embedding (easy implementation), for variational layers, use Basic Entangler Layers (which are equivalent to N-local circuit, but with removing the final rotation layer due to noise) with layers=2 and measure each qubit using Pauli-Z observable. The quantum TFT model without a QLSTM comprises 158 trainable parameters, whereas the version incorporating a QLSTM consists of 174 trainable parameters.

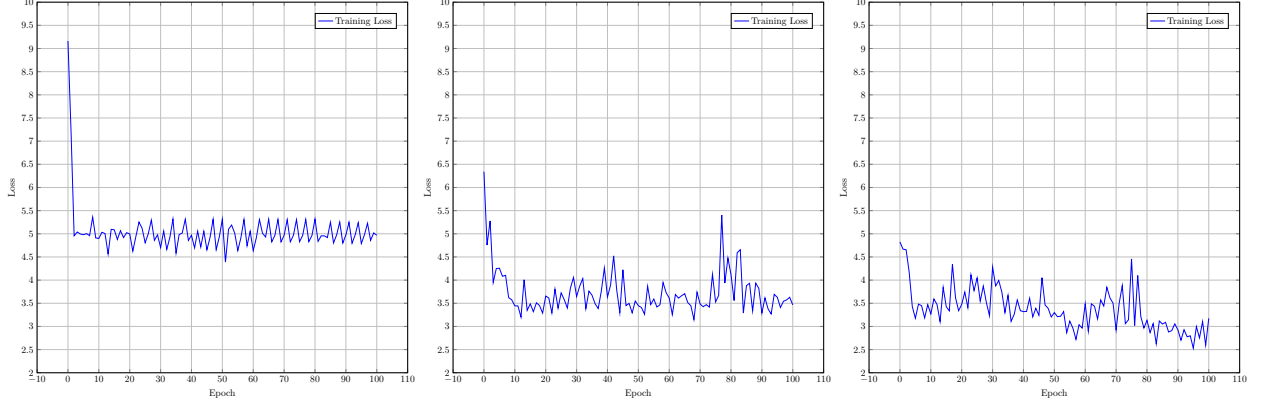


Figure 10: Graphical representation of Loss vs Epoch for training TFT model: the left-hand side graph depicts the classical TFT model, the middle graph illustrates the quantum TFT model without a quantum LSTM, and the right graph represents the quantum TFT model where the LSTM component is also quantum.

In Figure 10, we present the loss vs. epoch diagram. The left-hand side is the graph corresponding to quantum TFT model, while the right-hand side is for classical TFT. Although the quantum TFT model graph quantum graph exhibits more fluctuations compared to the classical TFT model, the overall result remains unaffected.

Figure 11 illustrates the training behaviors of both the quantum and the classical TFT models during epoch 0 to epoch 100. The top row shows the result for the quantum TFT model, while the bottom row corresponds to the classical one. In the diagram, the blue line indicates the true close value over time, while the dotted red line corresponds to the predicted close values. A close inspection reveals that the graph shows almost the same close values for two consecutive time steps, except at the start and end. This setup defines the configuration of our model, where we use two past input time steps to predict two future input time steps. Specifically, our model first takes feature vectors at the time of steps 1 and 2 as input to predict steps 3 and 4. Next, it takes steps 2 and 3 to predict steps 4 and 5, and so on. As a result of this overlapping window approach, we present almost the same predicted close values for two consecutive time steps, except for the starting and ending points.

	Training Loss	Testing Loss
TFT	0.2630	0.9856
QTFT (Without QLSTM)	0.2028	0.8381
QTFT (With QLSTM)	0.1711	0.8007

Table 1: Comparison of training and testing loss values for AXIS BANK data.

The table presents the training and testing loss values for AXIS BANK. The loss is computed here using the quantile loss function, as described earlier. Each value presents an average loss over a sliding window, the same setup described above.

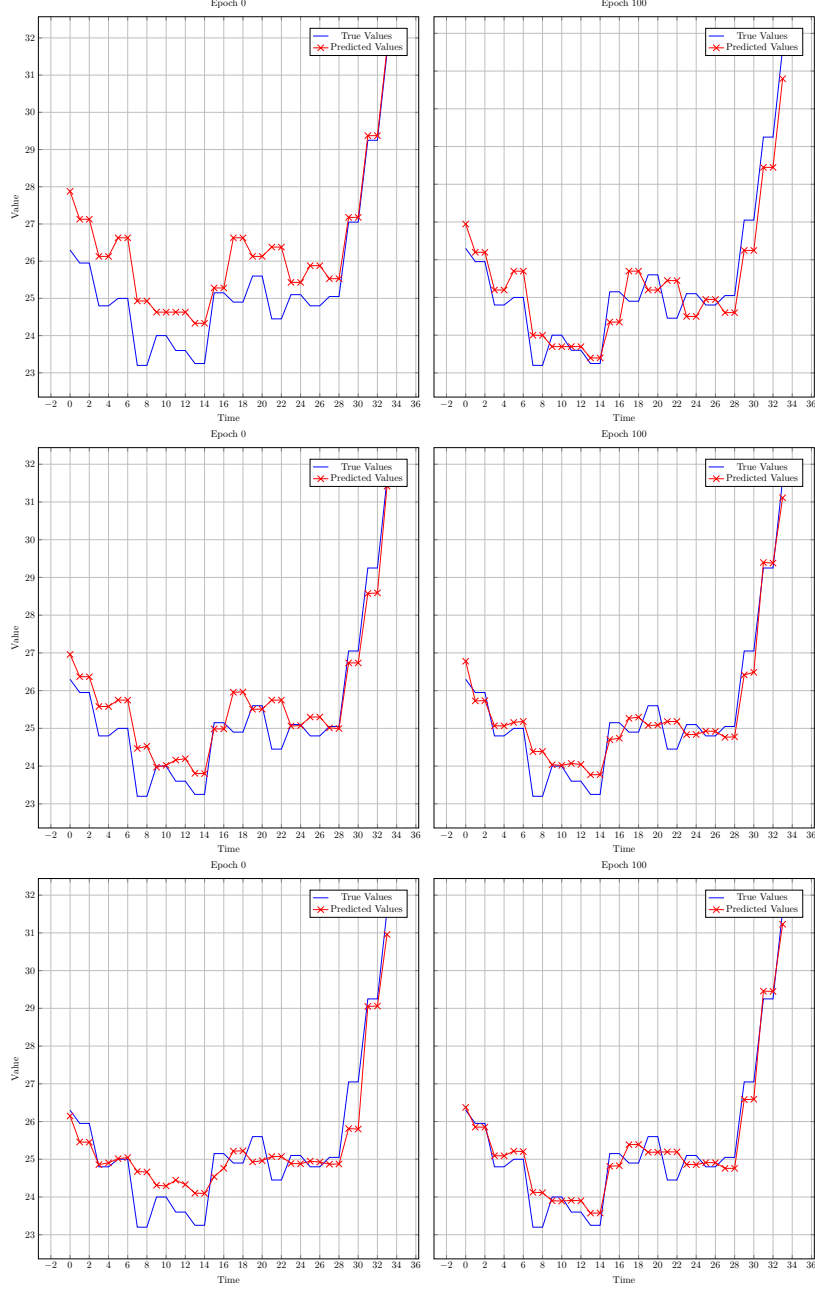


Figure 11: Learning from the AXIS BANK dataset: the top row depicts the Close Values vs. Time series graph for the classical TFT model, the middle row represents the quantum TFT without quantum LSTM, and the bottom row corresponds to the quantum TFT model with quantum LSTM components. At Epoch 0, the model calculates the loss and applies an optimizer for backpropagation as an initial step. Similarly, Epoch 100 indicates that the model computed the loss and applied the optimizer across 100 steps from the initial step.

6 CONCLUSION AND OUTLOOK

We provide a hybrid quantum-classical model architecture for quantum temporal fusion transformer (QTFT) which is able to perform multi-horizontal time series forecasting. In our experimental

setup, we demonstrate that the QTFT model effectively learns the model and generates forecasts accordingly. We show that under the constraint of a similar architectural structure and an approximately equal number of parameters, the QTFT model slightly performs better than the classical TFT.

While it is still impractical to run large-scale multi-horizontal time series forecasting due to the limitations of the current quantum simulator software, we emphasize that our architecture is general and scalable in principle. In general, the quantum variational circuits used in the QTFT model are given in a broad and flexible manner, incorporating a sufficient number of qubits, more different gate sequences, and a greater number of variational parameters - factors that potentially enhance the model’s learning capability and higher expressive power. However, it is possible that an appropriate modification in data encoding technique and structure variational circuit could give a better result.

Finally, if we assume the existence of a perfect quantum device with no noise, deployed with an unlimited number of qubits, exact control mechanisms, and full error-correction our model has the potential to yield highly efficient and insightful results.

In this work, we have explored how large-scale classical learning models are successfully trained and tested on quantum hardware. In the future, we are interested in investigating how modifications to the model’s architecture could enable its quantum version to achieve better performance than its classical counterpart. Specifically, we are interested in closely observing the quantum subcomponents of this model to improve each subcomponent individually, thereby improving the overall performance of the model.

7 Data Availability

The stock market dataset used in this work is publicly available from the National Stock Exchange of India (NSE) at <https://www.nseindia.com> (accessed on 2025-06-11).

References

- [1] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.
- [2] B. Lim, “Forecasting treatment responses over time using recurrent marginal structural networks,” *Advances in neural information processing systems*, vol. 31, 2018.
- [3] J. Zhang and K. Nawata, “Multi-step prediction for influenza outbreak by an adjusted long short-term memory,” *Epidemiology & Infection*, vol. 146, no. 7, pp. 809–816, 2018.
- [4] F. Piccialli, F. Giampaolo, E. Prezioso, D. Camacho, and G. Acampora, “Artificial intelligence and healthcare: Forecasting of medical bookings through multi-source time-series fusion,” *Information Fusion*, vol. 74, pp. 1–16, 2021.
- [5] D. Kroujiline, M. Gusev, D. Ushanov, S. V. Sharov, and B. Govorkov, “Forecasting stock market returns over multiple time horizons,” *Quantitative Finance*, vol. 16, no. 11, pp. 1695–1712, 2016.
- [6] C. Capistrán, C. Constandse, and M. Ramos-Francia, “Multi-horizon inflation forecasts using disaggregated data,” *Economic Modelling*, vol. 27, no. 3, pp. 666–677, 2010.

- [7] J.-H. Böse, V. Flunkert, J. Gasthaus, T. Januschowski, D. Lange, D. Salinas, S. Schelter, M. Seeger, and Y. Wang, “Probabilistic demand forecasting at scale,” *Proceedings of the VLDB Endowment*, vol. 10, no. 12, pp. 1694–1705, 2017.
- [8] P. Courty and H. Li, “Timing of seasonal sales,” *The Journal of Business*, vol. 72, no. 4, pp. 545–572, 1999.
- [9] D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski, “Deepar: Probabilistic forecasting with autoregressive recurrent networks,” *International journal of forecasting*, vol. 36, no. 3, pp. 1181–1191, 2020.
- [10] S. S. Rangapuram, M. W. Seeger, J. Gasthaus, L. Stella, Y. Wang, and T. Januschowski, “Deep state space models for time series forecasting,” *Advances in neural information processing systems*, vol. 31, 2018.
- [11] R. Wen, K. Torkkola, B. Narayanaswamy, and D. Madeka, “A multi-horizon quantile recurrent forecaster,” *arXiv preprint arXiv:1711.11053*, 2017.
- [12] A. M. Alaa and M. van der Schaar, “Attentive state-space modeling of disease progression,” *Advances in neural information processing systems*, vol. 32, 2019.
- [13] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, “The m4 competition: 100,000 time series and 61 forecasting methods,” *International Journal of Forecasting*, vol. 36, no. 1, pp. 54–74, 2020.
- [14] S. Li, X. Jin, Y. Xuan, X. Zhou, W. Chen, Y.-X. Wang, and X. Yan, “Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting,” *Advances in neural information processing systems*, vol. 32, 2019.
- [15] B. Lim, S. Ö. Arık, N. Loeff, and T. Pfister, “Temporal fusion transformers for interpretable multi-horizon time series forecasting,” *International Journal of Forecasting*, vol. 37, no. 4, pp. 1748–1764, 2021.
- [16] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” *Advances in neural information processing systems*, vol. 30, 2017.
- [17] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. Brandao, D. A. Buell *et al.*, “Quantum supremacy using a programmable superconducting processor,” *Nature*, vol. 574, no. 7779, pp. 505–510, 2019.
- [18] A. Cross, “The ibm q experience and qiskit open-source quantum computing software,” in *APS March meeting abstracts*, vol. 2018, 2018, pp. L58–003.
- [19] T. Lanting, A. J. Przybysz, A. Y. Smirnov, F. M. Spedalieri, M. H. Amin, A. J. Berkley, R. Harris, F. Altomare, S. Boixo, P. Bunyk *et al.*, “Entanglement in a quantum annealing processor,” *Physical Review X*, vol. 4, no. 2, p. 021041, 2014.
- [20] A. W. Harrow, A. Hassidim, and S. Lloyd, “Quantum algorithm for linear systems of equations,” *Phys. Rev. Lett.*, vol. 103, p. 150502, Oct 2009. [Online]. Available: <https://link.aps.org/doi/10.1103/PhysRevLett.103.150502>

- [21] A. J., A. Adedoyin, J. Ambrosiano, P. Anisimov, W. Casper, G. Chennupati, C. Coffrin, H. Djidjev, D. Gunter, S. Karra, N. Lemons, S. Lin, A. Malyzhenkov, D. Mascarenas, S. Mniszewski, B. Nadiga, D. O'malley, D. Oyen, S. Pakin, L. Prasad, R. Roberts, P. Romero, N. Santhi, N. Sinitsyn, P. J. Swart, J. G. Wendelberger, B. Yoon, R. Zamora, W. Zhu, S. Eidenbenz, A. Bärtschi, P. J. Coles, M. Vuffray, and A. Y. Lokhov, "Quantum algorithm implementations for beginners," *ACM Transactions on Quantum Computing*, vol. 3, no. 4, Jul. 2022. [Online]. Available: <https://doi.org/10.1145/3517340>
- [22] J. Preskill, "Quantum computing in the nisq era and beyond," *Quantum*, vol. 2, p. 79, 2018.
- [23] D. Gottesman, *Stabilizer codes and quantum error correction*. California Institute of Technology, 1997.
- [24] —, "Theory of fault-tolerant quantum computation," *Physical Review A*, vol. 57, no. 1, p. 127, 1998.
- [25] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio *et al.*, "Variational quantum algorithms," *Nature Reviews Physics*, vol. 3, no. 9, pp. 625–644, 2021.
- [26] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii, "Quantum circuit learning," *Physical Review A*, vol. 98, no. 3, p. 032309, 2018.
- [27] D. Wecker, M. B. Hastings, and M. Troyer, "Progress towards practical quantum variational algorithms," *Physical Review A*, vol. 92, no. 4, p. 042303, 2015.
- [28] O. Higgott, D. Wang, and S. Brierley, "Variational quantum computation of excited states," *Quantum*, vol. 3, p. 156, 2019.
- [29] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, "Quantum machine learning," *Nature*, vol. 549, no. 7671, pp. 195–202, 2017.
- [30] V. Dunjko and H. J. Briegel, "Machine learning & artificial intelligence in the quantum domain: a review of recent progress," *Reports on Progress in Physics*, vol. 81, no. 7, p. 074001, 2018.
- [31] M. Schuld, I. Sinayskiy, and F. Petruccione, "An introduction to quantum machine learning," *Contemporary Physics*, vol. 56, no. 2, pp. 172–185, 2015.
- [32] H.-Y. Huang, M. Broughton, M. Mohseni, R. Babbush, S. Boixo, H. Neven, and J. R. McClean, "Power of data in quantum machine learning," *Nature communications*, vol. 12, no. 1, p. 2631, 2021.
- [33] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *arXiv preprint arXiv:1511.07289*, 2015.
- [34] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, "Language modeling with gated convolutional networks," in *International conference on machine learning*. PMLR, 2017, pp. 933–941.
- [35] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.
- [36] Y. Gal and Z. Ghahramani, "A theoretically grounded application of dropout in recurrent neural networks," *Advances in neural information processing systems*, vol. 29, 2016.

- [37] M. Wang, S. Lu, D. Zhu, J. Lin, and Z. Wang, “A high-speed and low-complexity architecture for softmax function in deep learning,” in *2018 IEEE asia pacific conference on circuits and systems (APCCAS)*. IEEE, 2018, pp. 223–226.
- [38] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [39] M. Schuld, R. Sweke, and J. J. Meyer, “Effect of data encoding on the expressive power of variational quantum-machine-learning models,” *Physical Review A*, vol. 103, no. 3, p. 032430, 2021.
- [40] S. Lloyd, M. Schuld, A. Ijaz, J. Izaac, and N. Killoran, “Quantum embeddings for machine learning,” 2020. [Online]. Available: <https://arxiv.org/abs/2001.03622>
- [41] V. Bergholm, J. Izaac, M. Schuld, C. Gogolin, S. Ahmed, V. Ajith, M. S. Alam, G. Alonso-Linaje, B. AkashNarayanan, A. Asadi *et al.*, “PennyLane: Automatic differentiation of hybrid quantum-classical computations,” *arXiv preprint arXiv:1811.04968*, 2018.
- [42] M. Fingerhuth, T. Babej, and P. Wittek, “Open source software in quantum computing,” *PloS one*, vol. 13, no. 12, p. e0208561, 2018.
- [43] M. Benedetti, E. Lloyd, S. Sack, and M. Fiorentini, “Parameterized quantum circuits as machine learning models,” *Quantum science and technology*, vol. 4, no. 4, p. 043001, 2019.
- [44] M. Schuld, A. Bocharov, K. M. Svore, and N. Wiebe, “Circuit-centric quantum classifiers,” *Physical Review A*, vol. 101, no. 3, p. 032308, 2020.
- [45] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. M. Chow, and J. M. Gambetta, “Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets,” *nature*, vol. 549, no. 7671, pp. 242–246, 2017.
- [46] E. Farhi, J. Goldstone, and S. Gutmann, “A quantum approximate optimization algorithm,” *arXiv preprint arXiv:1411.4028*, 2014.
- [47] J. R. McClean, J. Romero, R. Babbush, and A. Aspuru-Guzik, “The theory of variational hybrid quantum-classical algorithms,” *New Journal of Physics*, vol. 18, no. 2, p. 023023, 2016.
- [48] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta, “Supervised learning with quantum-enhanced feature spaces,” *Nature*, vol. 567, no. 7747, pp. 209–212, 2019.
- [49] E. Farhi and H. Neven, “Classification with quantum neural networks on near term processors,” *arXiv preprint arXiv:1802.06002*, 2018.
- [50] P.-L. Dallaire-Demers and N. Killoran, “Quantum generative adversarial networks,” *Physical Review A*, vol. 98, no. 1, p. 012324, 2018.
- [51] S. Y.-C. Chen, C.-H. H. Yang, J. Qi, P.-Y. Chen, X. Ma, and H.-S. Goan, “Variational quantum circuits for deep reinforcement learning,” *IEEE access*, vol. 8, pp. 141 007–141 024, 2020.
- [52] A. Mari, T. R. Bromley, J. Izaac, M. Schuld, and N. Killoran, “Transfer learning in hybrid classical-quantum neural networks,” *Quantum*, vol. 4, p. 340, 2020.

- [53] S. Y.-C. Chen, S. Yoo, and Y.-L. L. Fang, “Quantum long short-term memory,” in *Icassp 2022-2022 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2022, pp. 8622–8626.
- [54] M. Schuld and N. Killoran, “Quantum machine learning in feature hilbert spaces,” *Physical review letters*, vol. 122, no. 4, p. 040504, 2019.
- [55] A. Javadi-Abhari, M. Treinish, K. Krsulich, C. J. Wood, J. Lishman, J. Gacon, S. Martiel, P. D. Nation, L. S. Bishop, A. W. Cross, B. R. Johnson, and J. M. Gambetta, “Quantum computing with Qiskit,” 2024.
- [56] G. Li, X. Zhao, and X. Wang, “Quantum self-attention neural networks for text classification,” *Science China Information Sciences*, vol. 67, no. 4, p. 142501, 2024.
- [57] M. Schuld, V. Bergholm, C. Gogolin, J. Izaac, and N. Killoran, “Evaluating analytic gradients on quantum hardware,” *Physical Review A*, vol. 99, no. 3, p. 032331, 2019.
- [58] A. Paszke, “Pytorch: An imperative style, high-performance deep learning library,” *arXiv preprint arXiv:1912.01703*, 2019.