

Quantum Modelling of Time Series

Expressivity vs. Trainability

Jacob L. Cybulski^{1,2}[0000–0002–9061–9389] and
Sebastian Zajac³[0000–0001–6616–8744]

¹ Enquanted, Melbourne, Australia
`jacob@enquanted.com`

² School of IT, Deakin University, Melbourne, Australia
`jacob.cybulski@deakin.edu.au`

³ Military University of Technology, Warsaw, Poland
`sebastian.zajac@wat.edu.pl`

Abstract. As time series feature unstructured data and unique processing requirements, they do not easily fit conventional approaches to quantum modeling. This paper therefore presents various quantum model architectures suitable for encoding and analysis of time series data. In particular, it investigates their architectural design aspects, such as methods of encoding and reuploading of temporal data, parameterizing quantum circuits, controlling the circuit qubit resources and entanglement, and dealing with issues of state evolution in the Hilbert space, as well as navigability of the classical parameter space by an optimizer. Each approach enhances or impedes model expressivity, i.e. its ability to effectively represent time series data in quantum space, as well as its trainability, i.e. its capacity to learn and generalize for predictive accuracy and efficiency in the process of model optimization. The paper evaluates different quantum circuit configurations for their impact on the expressivity and trainability of the model, and analyzes how these two design qualities influence model performance in training and testing.

Keywords: Quantum time series analysis · Quantum model architecture · Quantum model optimization · Quantum machine learning.

1 Introduction

Time series (TS) analysis and forecasting is a well-established field spanning mathematics, statistics, economics, and machine learning (ML). Recent advances in ML have incorporated concepts from quantum mechanics, giving rise to quantum machine learning (QML). While classical ML has achieved remarkable success in TS analysis, particularly through deep learning [7,13], progress in using quantum methods of time series analysis remains limited. Existing quantum applications include signal processing [5], time series classification [18], forecasting [1], and anomaly detection [11].

Unlike tabular data, TS requires dedicated methods for sequence representation and processing. Designing quantum models therefore involves selecting

suitable data encoding schemes, circuit architectures, and measurement strategies. In this work, we focus on the predictive capabilities of quantum neural networks for fitting and forecasting tasks. Specifically, we investigate the effects of circuit under- and overparameterization [10,6], compare encoding strategies with and without data re-uploading [15], and evaluate circuit designs capable of reusing qubits in the encoding pipeline, which we call “overloaded encoding”.

2 QTSA models

This work investigates gate-based quantum neural networks (QNNs) [1] optimized using variational quantum algorithms (VQAs) [16] for predictive quantum time-series analysis (QTSA), specifically curve-fitting and forecasting. While standard QNNs follow an input-ansatz-output block sequence, we explore architectural extensions designed to balance model *expressivity*—its ability to effectively represent and manipulate TS data in quantum space—and *trainability*—its capacity to learn and generalize efficiently during optimization.

Enhancing QNNs with auxiliary qubits increases parameter capacity and expressivity [9], but the resulting excess entanglement induces *nonlocal interactions* that impair gradient-based trainability [12]. Furthermore, high-dimensional parameter spaces can flatten the gradient geometry into barren plateaus [2]. In contrast, adding parameters beyond a critical threshold leads to circuit *overparameterization*, which can improve learning capacity and trainability [10]. To further enrich feature representation in high-dimensional Hilbert spaces without excessive circuit width, architectures can employ sequential data re-uploading across trainable layers [15]. Additionally, mid-circuit measurements and qubit reuse can reduce hardware resource demands while limiting entanglement [4].

We evaluate these architectural trade-offs across two predictive tasks:

- **Curve-fitting models** predict data values at specific timestamps by fitting a function to the sample data points. We evaluate three data re-uploading strategies in the following model architectures: single-qubit *serial* (Eq. 3), multi-qubit *parallel* (Eq. 6), and extended multi-qubit *xparallel* (Eq. 7).
- **Forecasting models** predict future TS values from a sliding window (SW) of their historical context. We evaluate two architectural variants: *xqnn*, which distributes the input window across dedicated qubits while extending circuit width for overparameterization (Eq. 11), and *ovload*, which overcomes circuit width constraints by splitting long temporal windows into fragments encoded sequentially between trainable layers via qubit reuse (Eq. 14). A classical neural network (*cnn*) that uses sliding-window inputs was also included to serve as the baseline benchmark.

The majority of the investigated models employ angle encoding [17,16] to efficiently represent continuous time-series windows without altering the circuit structures. Also, to isolate architectural effects for this investigation, all QNN variants apply the same measurement approach, i.e. monitoring only the state of the qubits involved in data encoding, and omitting the state of auxiliary qubits, within a noise-free state-vector simulation.

3 Formalization of QTSA architectures

Each QTSA model extends a generic QNN, which is defined as:

$$U(\mathbf{x}, \boldsymbol{\theta}) = U_{\text{ansatz}}(\boldsymbol{\theta}) \cdot \mathcal{U}_{\text{FM}}(\mathbf{x}), \quad (1)$$

where $\mathcal{U}_{\text{FM}}$ is a feature map encoding data and U_{ansatz} a trainable quantum component, taking the classical input data $\mathbf{x}$ and a set of trainable parameters $\boldsymbol{\theta} = \{\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_{n+1}\}$, where each $\boldsymbol{\theta}_k = (\theta_k^{(\gamma)}, \theta_k^{(\phi)}, \theta_k^{(\lambda)})$ provides angles γ , ϕ , and λ for the generic rotational operator $U_{\text{rot}}(\boldsymbol{\theta}_k) = Z(\gamma)Y(\phi)Z(\lambda)$, where Z, Y (and X) are standard Pauli spin operators [14, sect 2.1.3]. The predictions can be extracted by measuring the expectation value of an observable M :

$$y = \langle 0 |^{\otimes q} U^\dagger M U | 0 \rangle^{\otimes q}, \quad (2)$$

where q is the number of qubits, and unless specified otherwise, $M = \sum_{j=1}^q Z_j$.

3.1 Curve-Fitting Models

The QTSA models aim to approximate a continuous function $f : \mathbb{R} \rightarrow \mathbb{R}$ to predict individual values of the time series. These models are reuploading the same scalar input $x \in \mathbb{R}$ across the quantum circuit to capture non-linear temporal dependencies. As discussed above, we consider three architectural variants:

(a) *Serial Reuploading (serial)*. In the single-qubit case ($q = 1$), the circuit alternates between trainable unitary blocks and repeated data-encoding gates. Each data-encoding gate is a rotation about the X -axis, while each trainable gate is a general single-qubit unitary parameterized by three angles.

Given a scalar input x and the number of layers n , and the parameters set $\boldsymbol{\theta}$. The resulting state is:

$$|\psi(x, \boldsymbol{\theta})\rangle = U_{\text{rot}}(\boldsymbol{\theta}_{n+1}) \cdot \left(\prod_{j=1}^n [R_X(x) \cdot U_{\text{rot}}(\boldsymbol{\theta}_j)] \right) \cdot |0\rangle, \quad (3)$$

where $R_X(x) = \exp(-ixX/2)$ is the data encoding operation.

(b) *Parallel Reuploading (parallel)*. In the multi-qubits case ($q > 1$), this architecture involves reuploading data to a register of q qubits and encoding the same scalar input x identically in each:

$$R_X^q(x) = \bigotimes_{j=1}^q R_X^{(j)}(x). \quad (4)$$

Variational layers before and after the encoding consist of trainable entanglement blocks $U_{\text{ent}}(\boldsymbol{\theta}_j)$:

$$U_{\text{ent}}(\boldsymbol{\theta}_j) = \prod_{i=1}^q \text{CNOT}_{i, (i \bmod q)+1} \cdot U^q(\boldsymbol{\theta}_j), \text{ and } U^q(\boldsymbol{\theta}_j) = \bigotimes_{i=1}^q U_{\text{rot}}^{(i)}(\boldsymbol{\theta}_j^{(i)}), \quad (5)$$

where each $U_{rot}^{(i)}$ represents a trainable single-qubit unitary operator, and each θ_k consists of a block of q parameter vectors, resulting in a total of $3 \times q$ parameters for the k -th layer $U^q(\theta_k)$ spanning q qubits.

The full quantum state is prepared as:

$$|\psi(x, \theta)\rangle = \left(U^q(\theta_{n+1}) \prod_{j=m+2}^n [U_{\text{ent}}(\theta_j)] \right) R_X^q(x) \left(U^q(\theta_{m+1}) \prod_{k=1}^m [U_{\text{ent}}(\theta_k)] \right) |0\rangle^{\otimes q}, \quad (6)$$

where θ is the full parameter set and n, m are the numbers of layers after and before encoding, respectively.

(c) *Extended Parallel Reuploading (xparallel)*. This variant interleaves reuploading layers with entangling blocks multiple times, improving capacity to model long-term dependencies. The prepared state is:

$$|\psi(x, \theta)\rangle = \left(U^q(\theta_{m+1}) \prod_{k=L+1}^m U_{\text{ent}}(\theta_k) \right) \cdot \left(\prod_{l=1}^L [R_X^q(x) \cdot U_{\text{ent}}(\theta_l)] \right) |0\rangle^{\otimes q}, \quad (7)$$

where L is the number of reuploading layers.

3.2 Forecasting Models

Forecasting QTSA models aim to predict a sequence of future time-series values from a given input window. These models approximate mappings $f: \mathbb{R}^n \rightarrow \mathbb{R}^k$, where n is the window size and k is the prediction horizon.

(a) *Extended QNN (xqnn)* The architecture can be interpreted as an extension of the formulation given in Eq. (1), where the feature map $U_{FM}(\hat{\mathbf{x}})$ encodes the input vector $\hat{\mathbf{x}} = (x_1, \dots, x_n)$, consisting of n features, into a quantum state.

The employed **ZZ feature map** spans only n qubits (assuming $n \leq q$). We follow the approach introduced by Havlicek et al. [8], where the feature map on n qubits is defined as:

$$\mathcal{U}_{FM}(\mathbf{x}) = U_{\Phi(\mathbf{x})} \cdot H^{\otimes n}, \quad (8)$$

where H denotes the Hadamard gate and

$$U_{\Phi(\mathbf{x})} = \exp \left(i \sum_{S \subseteq [n]} \phi_S(\mathbf{x}) \prod_{i \in S} Z_i \right). \quad (9)$$

Only subsets S with cardinality $|S| \leq 2$ are considered. In particular, $\phi_{\{i\}}(\hat{\mathbf{x}}) = x_i$ and $\phi_{\{i,j\}}(\hat{\mathbf{x}}) = (\pi - x_i)(\pi - x_j)$. Each term $\exp(i\phi_{\{i,j\}}(\hat{\mathbf{x}})Z_iZ_j)$ can be implemented using a combination of CNOT and R_Z gates (see Fig. 1(c) in [8]).

The variational ansatz $U_{AN}(\theta)$ is a parameterized quantum circuit that acts on all q qubits. It consists of L_{AN} layers of trainable unitary blocks $U_{\text{ent}}(\theta_j)$ (see Eq. 5), where each θ_j denotes the parameter set for the j -th layer.

The total ansatz unitary is then:

$$U_{AN}(\boldsymbol{\theta}) = \prod_{k=1}^{L_{AN}} U_{\text{ent}}(\boldsymbol{\theta}_k). \quad (10)$$

The resulting quantum state is prepared by applying the feature map followed by the variational ansatz to the initial state:

$$|\psi(\mathbf{x}, \boldsymbol{\theta})\rangle = U_{AN}(\boldsymbol{\theta}) \cdot \mathcal{U}_{FM}(\mathbf{x}) |0\rangle^{\otimes q}. \quad (11)$$

In this model, we measure only those qubits that were used for data encoding. The final state of the remaining qubits (when $n < q$), which are used to increase the dimensionality of the parameter space, is not monitored.

(b) *Overloaded QNN (ovload)* When the number of input features exceeds the number of qubits, i.e., $n > q$, the input vector $\hat{\mathbf{x}} = (x_1, \dots, x_n)$ is cyclically assigned to the available qubits. Specifically, features are grouped into $\lceil \frac{n}{q} \rceil$ segments, and within each segment, the values are mapped to the corresponding qubits via repeated applications of the rotation gate R_X . The feature encoding is then given as:

$$\mathcal{R}_X^q(\hat{\mathbf{x}}) = \prod_{k=0}^{\lceil n/q \rceil - 1} \left(\bigotimes_{j=1}^q R_X^{(j)}(x_{kq+j}) \right), \quad (12)$$

where $x_m = 0$ for index $m > n$ to pad the input if n is not divisible by q . Each entanglement layer is formed by combining two unitary blocks:

$$W_{\text{ent}}^l(\hat{\boldsymbol{\theta}}) = U^q(\boldsymbol{\theta}_{l+1}) \cdot U_{\text{ent}}(\boldsymbol{\theta}_l), \quad (13)$$

where $\hat{\boldsymbol{\theta}}$ includes all sets of parameters.

The final quantum state is constructed by repeatedly applying input encoding and entangling blocks.

$$|\psi(\hat{\mathbf{x}}, \hat{\boldsymbol{\theta}})\rangle = W_{\text{ent}}^{L+1}(\hat{\boldsymbol{\theta}}) \cdot \left(\prod_{k=1}^L \left[\mathcal{R}_X^q(\hat{\mathbf{x}}) \cdot W_{\text{ent}}^k(\hat{\boldsymbol{\theta}}) \right] \right) |0\rangle^{\otimes q}. \quad (14)$$

This “overloaded” structure increases expressivity by coupling each qubit directly to an input dimension and adding more variational flexibility per layer.

4 Experimental setup

All models were implemented using Qiskit 1.3.2 (Machine Learning 0.8.2) and PyTorch 2.6.0, executing CPU-based Estimator primitives and state-vector simulations on Ubuntu Linux 22.04.5 LTS (64GB RAM, 32-core Intel Core i9).

As this study aimed to evaluate quantum model architectures based on their inherent design rather than real-world performance, consequently, a simple synthetic dataset was sufficient for their training.

To train all models, we used a small synthetic data set $\mathcal{Z}_{sins}$ (see Eq. 15), which combined trigonometric functions to generate an oscillation pattern of 70 data points, normalized to the value range of (0–1).

$$f_{2sins}(x) = \frac{\sin(5.0 * x) + 0.5 * \sin(8.0 * x)}{4} + 0.5 \quad (15)$$

In data preparation for working with curve-fitting models, the series was divided into 50 data points for training and 20 data points for testing.

Data for forecasting models were prepared differently, as the 70 $\mathcal{Z}_{sins}$ data points were first converted to sliding windows of 5 data points, the windows were offset by 1 step, with a horizon of 1, and all incomplete windows were rejected. The resulting series of 65 windows was subsequently split into 46 windows for training and 19 windows for testing (see Figures 1a and 1b). In both cases, data were divided into training and testing sets in an approximate ratio of 71/29.

Unlike tabular data, time series can present significant challenges for model training due to the close proximity of its neighboring values. In curve-fitting and forecasting models, the optimizer will have to deal with groups of very similar and inter-dependent values provided on input and expected on output. It is therefore crucial that the model has an architecture that maps these features effectively into a high-dimensional Hilbert space to support optimization tasks.

When developing QTSA models, many standard data preprocessing tasks were not required, either due to a small dataset size or the use of noise-free quantum simulators. These included: dimensionality reduction and time series compression, imputation of missing values, feature engineering, removal of noise, outliers and anomalies, dealing with data normality and homoskedasticity, and data shuffling (not applicable to time series).

5 QTSA models performance

The study focused strictly on the fundamental expressivity and trainability of the proposed quantum architectures. Expressivity is inherently a geometric property of the model’s ansatz rather than its execution medium, while trainability challenges—such as barren plateaus—are primarily governed by initialization, cost function design, and optimization landscapes. Because hardware noise introduces unresolvable confounding factors (e.g., state decoherence), physical device execution and error mitigation were explicitly scoped out to ensure a controlled baseline evaluation using an ideal statevector simulator.

The effectiveness of each model type was investigated by changing its meta-parameters, such as the number of qubits and layers, and their roles. This resulted in distinct model architectures. Each architecture was then instantiated 10 times, with each instance pseudo-randomly initialized in a continuous uniform distribution in $[0, 2\pi]$ (with seeds generated and saved), and then trained

to achieve a minimum cost (in our case MSE). At each optimization step, the instance parameters were saved for later scoring in training and testing data partitions. The scoring metrics included MSE and R^2 . The performance of a specific model architecture was then represented by the instance (among 10 instances) that resulted in a median test score (maximum R^2 of all its saved parameters).

All Qiskit QML models were implemented with *EstimatorQNN* primitives, designed to compute the expectation values of quantum observables. The models were trained using the *NeuralNetworkRegressor* performing regression tasks for Qiskit QNNs. The regressor relied on the limited-memory version of the Broyden-Fletcher-Goldfarb-Shanno (L_BFGS_B) algorithm, designed as an iterative method for solving unconstrained, large-scale nonlinear optimization problems. The optimization process was guided by the *L2Loss* (MSE cost).

As all quantum models had various number of parameters ($N \in [12, 240]$), they were optimized within $\text{maxiter} = 60$ epochs sufficient for all models to converge on a 10,000-shot statevector simulator. Classical baselines were excluded because they utilize backpropagation. For finite-difference gradients, total statevector computations scale as $N_{\text{eval}} = N_{\text{fev}} + 60N$. With line-search steps capped at $\text{maxls} = 20$, the strict upper bound is $1200 + 60N$ computations (1,920 to 15,600). Empirically averaging 2.5 function calls per iteration, the realistic expected cost was $\approx 150 + 60N$ computations (≈ 870 to $\approx 14,550$).

After training, performance measurement involved scoring all models with MSE and R^2 statistics on training and test datasets, and then comparing the models within and across curve-fitting and forecasting groups. It is important to note that, in general, models of both groups have unique objectives in this process. Curve-fitting models aim to minimize the error at each specific point in time. The residuals are directly linked to individual data points. Forecasting models, on the other hand, aim to predict future values based on a temporal context of sliding windows. The residuals are measured on the forecasted values, which is drastically different from the historical points the curve-fitting models operate on. In both cases, the notions of “data point” and “error” between the model groups are quite different. It is worth noting that these two notions are identical for all models in their respective groups, and hence the MSE statistics were useful for performance comparison within their groups. However, due to inconsistencies in the fundamental measurement concepts, MSE was not appropriate to compare the models and their performance between the curve fitting and forecasting groups, especially because the forecasting models relied on the overlapping window data, thus covering a significantly larger set of (repeated) data points (230) than those of the curve fitting models (70).

Hence, it was decided to compare performance between two groups of models based on some relative measure between prediction and ground truth, such as MAPE, sMAPE, NRMSE or R^2 statistic (coefficient of determination). R^2 was selected for its simplicity and widespread use. R^2 is a proportion of variance in the dependent variable that can be explained by independent variables. So, the R^2 statistic uses a very simple baseline - the mean of the dependent variable, it is unit-free and scale independent.

Table 1: Comparison of different QTSA architectures

Model	Type	Q#	P#	Training R2 (IQR)	Testing R2 (IQR)	Specs
serial	CF	1	66	0.9926 (0.0060)	<i>0.9122</i> (0.1617)	q1 l21
serial	CF	1	84	<i>0.9947</i> (0.0053)	0.8912 (0.1982)	q1 l27
parallel	CF	5	120	<i>0.8620</i> (0.0000)	0.6939 (0.0283)	q5 bl3 al3
parallel	CF	5	90	0.8620 (0.0001)	<i>0.7109</i> (0.0374)	q5 bl1 al3
xparallel	CF	3	81	<i>0.9996</i> (0.0002)	0.9812 (0.0254)	q3 bl7 al1
xqnn	FC	5	90	<i>0.9789</i> (0.0087)	0.5607 (0.2022)	q5 in5 fm1 anz5
xqnn	FC	7	84	0.9629 (0.0205)	<i>0.8643</i> (0.0844)	q7 in5 fm1 anz3
ovload	FC	4	167	<i>0.8888</i> (0.0013)	<i>0.8734</i> (0.0018)	q4 xl3 il2
cnn	FC	0	106	1.0000 (0.0021)	0.9684 (0.0420)	hl008 005 003
cnn	FC	0	375	1.0000 (0.0000)	0.9771 (0.0008)	hl010 015 010
cnn	FC	0	5950	1.0000 (0.0000)	<i>0.9796</i> (0.0010)	hl050 080 020

Note 1: Q# = Qubit Number, P# = Params Number, CF = Curve-Fitting, FC = Forecasting.

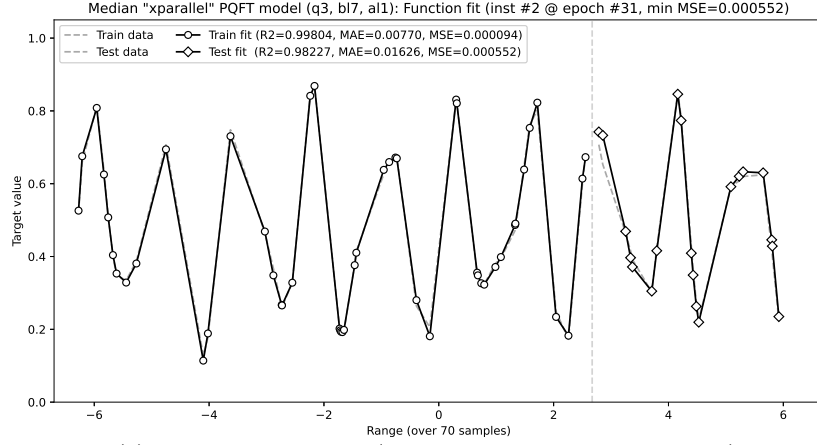
Note 2: Figures in **bold** = best overall, in *italic* = best for their model architecture.

In summary. To ensure rigorous and unbiased testing, all model variants were evaluated under a locked optimization protocol utilizing identical data partitions and an invariant optimization budget (L-BFGS-B, 60 epochs). Because no post-hoc hyperparameter tuning or selective model architecture adjustments were performed on the test set, information leakage during the selection phase was eliminated. Moreover, to mitigate initialization bias and the multiple comparisons problem, we reported the median values with their inter-quartile ranges across ten independent, randomly initialized instances for each variant. As all visualizations depict the median-performing instances, this ensured that reported predictive capabilities reflect typical behavior rather than outliers.

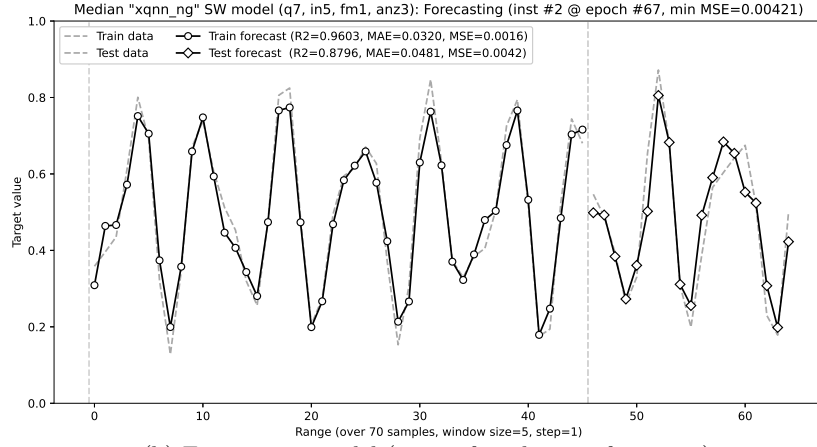
6 Summary of experimental results

Table 1 compares the experimental results of the best models. For each model architecture, the table reports the highest training and testing score R^2 achieved across (in bold) and within (in italic) their own architectures, where the baseline of each variant was determined by the median performance of its 10 instances.

This approach was preferred over mean-based selection, which was due to the sensitivity of quantum models to initialization, and the possibility of generating outliers (e.g., small *cnn* models) or skewing Gaussian distribution of instance results. In such circumstances, an effective calculation of their mean is impossible. The overall spread of the results obtained by all instances is given by their interquartile range (IQR). The additional advantage is that the median performance is attached to a specific model instance, which could be used to visualize its data fit (for even number of results, the first of the middle pair was used).



(a) Curve-fitting model (*xparallel* of median performance).



(b) Forecasting model (*xqnn* of median performance).

Fig. 1: Data fit of two selected models. Partitions: training - left, test - right.

Among curve-fitting quantum models (*serial*, *parallel* and *xparallel*), re-uploading data serially leads to models fitting the synthetic data better (training $R^2=0.9947$ and testing $R^2=0.9122$) than those reuploading data in a purely parallel fashion (training $R^2=0.8620$ and testing $R^2=0.7109$).

However, the best model in terms of training and test performance was *xparallel*, which incorporates serial and parallel re-uploading methods (training $R^2=0.9996$, testing $R^2=0.9812$). Although the optimum *xparallel* model relies on a smaller number of training parameters than the best *parallel* model (81 vs 90 parameters), its testing performance was slightly better than the performance of much larger classical neural networks (for example, at 5950 parameters, testing

$R^2=0.9796$). Figure 1a illustrates a nearly perfect fit of the data to the *x-parallel* model instance of median performance in training and testing.

Forecasting quantum models are represented by two variants of QNNs (*xqnn* and *ovload*). With their ability to overparameterize circuits, the *xqnn* models seemed to rapidly overtrain. While adding qubits allowed for expansion of the model learning capacity, this also resulted in patchy training performance (with the best $R^2=0.9789$), and yet, with systematic improvement of their test results (up to $R^2=0.8643$ at 84 parameters with 7 qubits and 3 layers). The *ovload* models, which are capable of overloading their qubits with parameters, do not have the same level of learning capacity (with training $R^2=0.8888$ only) as *xqnn* models. However, they exceeded their test performance (testing $R^2=0.8734$), and at a reduced number of qubits (only 4 vs 7 qubits), however, at the expense of doubling the number of model parameters (168 vs. 84). Figure 1b illustrates a good fit between the data and the median performing model instance of the *xqnn* model architecture in the training and testing data partitions.

Among curve-fitting models, the *xparallel* models not only achieved the best test performance, but were also the most stable across their instances (testing IQR=0.0254), especially against the second best performing *serial* model (testing IQR=0.1617). The same was observed in the highly stable model *ovload* (testing IQR=0.0018) vs. *xqnn* (testing IQR=0.0844). The *ovload* models were also the most predictable, as their training and test performance scores were very close (training $R^2=0.8888$ vs. testing $R^2=0.8734$) compared to *xqnn*, where training and test performance were widely divergent (training $R^2=0.9629$ vs. testing $R^2=0.8643$). Classical *cnn* models demonstrated great stability of results.

Overall, when we exclude the massive *cnn* model of the 20800 parameters, the model architecture that performed the best was *xparallel*.

7 Analysis and discussion

Critical to this analysis is understanding the relationships between different aspects of model design. These aspects include:

- *Hilbert space* is the quantum realm where the model state evolves (defined by the circuit) in response to unitary operations (defined by circuit gates);
- *data encoding* brings in classical data into the Hilbert space as unique and correlated quantum states during the model execution;
- *layers of unitary operations* (defined by circuit gates) determine the evolution of the quantum model’s initial state into its final state;
- *trainable parameter space* is a classical multi-dimensional space of trainable parameters (defined by circuit gates), which the optimizer navigates;
- *entanglements* (defined by CNOT gates) correlate non-separable qubit states, which alter the parameter space geometry, and thus the cost landscape;
- *measurement* of individual qubits projects their final quantum states onto classical outcomes, the probabilities of which reflect their correlation.

Based on earlier discussion (see Section 2), we will apply the following guiding principles: more qubits, higher circuit entanglement, data reuploading, and overparameterization *contribute to model expressivity* [9,10,15]; at the same time, poor data encoding, more layers and parameters, excessive entanglement, and nonlocal interactions can *reduce model trainability* [12,2].

Therefore, depending on the adopted model architecture, every aspect of its ansatz design may have a beneficial or detrimental effect on the expressivity and/or trainability of its models. The following analysis of experimental results will explicitly address these aspects. However, in order to isolate the effect of the ansatz design, we had to standardize circuit measurements, and consequently we have excluded measurement strategies from consideration—even though they can influence model trainability (e.g., local vs. global measurements, dimensionality, sparsity, and precision) [4].

Note that the complete and detailed multi-model comparison tables, featuring both R^2 and MSE median scores with IQR, are publicly archived and available at the authors' GitHub [3].

7.1 Experiments with curve-fitting models

Serial models. The curve-fitting *serial* models (see Eq. 3 and [3, table A]) were unique, since they used only a single qubit (q1), with several layers (l) of data reuploading and trainable blocks.

Because of repeated data reuploading interleaved with parametrized state rotations, the circuit implements a high-dimensional parameter-space mapping that maximizes the expressivity within the single-qubit state space.

As the Hilbert space dimension is strictly bounded at $2D$, this single-qubit architecture is inherently immune to standard qubit-dependent barren plateaus, contributing to its exceptionally fast and effective optimization, as observed in training $R^2=0.8892$ (IQR=0.0580) and testing $R^2=0.5120$ (IQR=0.0833) at 9 layers, with gradual improvement in training $R^2=0.9947$ (IQR=0.0053) at 27 layers and testing $R^2=0.9122$ (IQR=0.1617) at 21 layers. By increasing the number of layers, the *serial* models seem to approach the training and testing performance of the *cnn* models (classical neural networks).

However, when the parameter space becomes very large—due to heavy application of reuploading layers—single-qubit models do not completely avoid trainability problems. Instead, they are exposed to spectral dilution, in which the gradient signal is spread thinly across an increasingly intricate Fourier frequency spectrum. This effect may account for the unstable test performance (a decline in IQR) despite the models exhibiting strong results during training. Also, as the system is restricted to a single qubit, it inherently lacks multi-qubit entanglement and nonlocal effects, limiting its capacity to capture complex temporal data correlations. Furthermore, the two-dimensional nature of the Hilbert space ensures that the model is likely to be simulable on classical hardware, offering no asymptotic quantum computational advantage.

Parallel models. In experiments with curve-fitting *parallel* models (see Eq. 6 and [3, table B]), the circuits were varied by changing the number of qubits (q), layers of trainable pre-encoding blocks (bl) and post-encoding blocks (al).

The experimental results were structured into four groups, each aimed at varying the number of qubits as well as the number of pre- and post-encoding layers, and consequently the number of model parameters. The experiments demonstrated that, without the benefit of serial data reuploading, narrow circuits (less than 5 qubits) often produced worse than random performance (negative R^2 scores). Hence the subsequent tests focused on 5 qubit models, which had a higher training performance $R^2=0.8620$ (IQR=0.0000), but sub-optimal test performance reaching $R^2=0.7109$ (IQR=0.0066) at 1 pre- and 3 post-encoding layers. It is revealing that repeated layers of added parameters and entanglements, either pre- or post-encoding, had little (or zero) effect on the model expressivity and trainability, so a simple state preparation and ansatz at five qubits were the most beneficial for the performance.

While an IQR of 0.0000 typically indicates high training stability, in the context of the *parallel* models' suboptimal performance, it instead reveals a hard learning capacity ceiling. The model consistently converges to the same sub-par outcome regardless of parameterization or initialization, demonstrating that adding trainable layers without sequential data re-uploading fails to meaningfully increase expressivity.

Xparallel models. *Xparallel* curve-fitting models were designed to extend the *parallel* models by adding sequential data reuploading. Experiments with these multi-qubit models (see Eq. 7 and [3, table C]) focused on varying the number of qubits (q) and layers consisting of pairs: parallel reuploading blocks and parametrized training blocks (bl). The sequence of entangling and reuploading layers ended with optional parametrized entangling blocks (al, limited to 1). All models had a similar number of trainable parameters (69-84).

As evidenced from the experiments carried out, the *xparallel* model favored circuits that are narrow and deep, i.e. 3 qubits and 7 layers that achieved the best training $R^2=0.9996$ (IQR=0.0002) and testing $R^2=0.9812$ (IQR=0.0254), and the highest stability (low IQR). The models of this architecture had more data reuploading blocks when the number of qubits decreased, and fewer when their circuit increased its depth. With more data reuploading, the circuit created an expressive feature map, which compensated for the loss of Hilbert space dimensionality and expressivity within the ansatz.

A single qubit *xparallel* model was essentially *serial*, however, due to a different architecture (with more trainable parameters), it resulted in a slightly different performance (training $R^2=0.9963$, testing $R^2=0.9434$).

7.2 Experiments with forecasting models

Xqnn models. *Xqnn* models provide a small extension to the traditional QNNs to allow the addition of auxiliary qubits, consequently resulting in the widening of circuits, thus gaining more rotational parameters and entanglements (see Eq.

11 and [3, table D]). The model encoding adopted a traditional multilayered ZZ feature map (fm), embedding windows of 5 input values (in), followed by a post-encoding layer of several trainable blocks (anz), with optional auxiliary qubits (to a total of q qubits). For consistency with models featuring a fixed number of qubits, only input encoding qubits were measured. The experiments with *xqnn* models aimed to investigate their capacity to overparameterize, which also increased the model entanglement.

Three groups of experiments were conducted, each varying the circuit width (number of qubits), depth (number of post-encoding layers), and the complexity of state preparation (number of encoding layers).

The experiments show that *xqnn* models were among the least stable, with training $\text{IQR} \in [0.0087, 0.1596]$ and test $\text{IQR} \in [0.0581, 0.3260]$. This massive performance spread and erratic training trajectory highlight a severe optimization bottleneck. As time series windows are encoded in a highly entangling ZZ feature map, and the distinction between neighboring values is inherently small, this scrambles the closely related states, creating a rugged cost landscape where training convergence is choppy and heavily reliant on initialization luck.

Conversely, the *xqnn* models showed a definite, smooth performance pattern in testing, increasing with the number of layers up to an optimum and then decreasing. The peak training $R^2=0.9789$ ($\text{IQR}=0.0087$) was achieved at 5 qubits and 5 ansatz layers, and the best test $R^2=0.8643$ ($\text{IQR}=0.0844$) at 7 qubits and 3 ansatz layers. This indicates that while optimization luck drives training scores, the true expressivity of the circuit dictates generalizability.

Initially, with the addition of auxiliary qubits, the *xqnn* model’s Hilbert space expands exponentially, offering a richer feature space. However, once the architecture scales past its optimal capacity, excess entanglement completely homogenizes and randomizes the feature space. This makes the later state evolution ineffective, leading to a predictable degradation in test performance even when training appears successful. This homogenization penalty is distinctly visible in the model’s response to increased depth of its ZZ feature map: testing performance drops sharply from $R^2=0.8643$ ($\text{IQR}=0.0844$) at 1 ZZ layer, down to $R^2=0.2569$ ($\text{IQR}=0.1413$) at 3 ZZ layers.

Ovload models. *Ovload* models adopt the concept of multi-qubit data reuploading technique from the curve-fitting *xparallel* models. However, they tailor it to represent sliding TS windows, rather than individual TS values, to reduce the demand for qubit resources (see Eq. 14 and [3, table E]). This is accomplished by splitting longer windows into a number of smaller fragments (il) that fit the available qubits (q). The *ovload* layers (xl) are then formed by encoding all window fragments and interleaving them with trainable blocks. As a result, the circuit’s qubits are overloading the representational function for the windows sub-sequences, which relies on the continuity of reuploaded consecutive data points in a TS window (to replace their identity in the *xparallel* models). To deal with the threat of homogenization of TS features due to high encoding entanglement (the issue noted in *xqnn* model), the layers are repeated to reupload the window’s data, and hence increase the model expressivity.

The experiments vary the number of *ovload* model qubits (from 1 to 4 < 5, which is the window length) and the number of reuploading layers (from 1 to 3). The results indicate that in narrow circuits (small number of qubits), the discrepancy in the window values mapped against shared qubits prevents the model from forming a coherent quantum state, resulting in low training performance ($R^2=0.2321$). As the qubit numbers approach those of the QNN representation (4 qubits to represent windows of 5 values), the model reaches its performance peak in both training $R^2=0.8888$ (IQR=0.0013) and testing $R^2=0.8734$ (IQR=0.0018), the latter is similar to that of *xqnn*. At the same time, the model doubled the number of parameters, thus flattening the gradient space, resulting in less effective model optimization, simultaneously implying inadequate learning capacity (as evidenced by the suboptimal training $R^2=0.8888$).

Rather than viewing the limited training capacity of the *ovload* architecture as a weakness, its behavior highlights a compelling mechanism of structural regularization. By forcing the temporal data through a constrained Hilbert space—such as utilizing 4 qubits for a 5 step window—the architecture intentionally bottlenecks expressivity. While this restriction prevents the model from perfectly memorizing the training set, it effectively guards against the extreme overfitting observed in wider, highly entangled circuits. Consequently, *ovload* achieves the most stable and highest testing performance among the forecasting variants, demonstrating that controlled qubit reduction can act as a powerful regularizer to improve generalizability.

Comparing Quantum and Classical Baselines. While establishing definitive quantum advantage falls outside the scope of this architectural evaluation, the inclusion of classical CNN baselines provides a valuable reference point for current QML capabilities (see Table 1 and [3, table E]).

As expected, the classical models exhibited vastly superior trainability and stability, consistently achieving near-perfect training scores ($R^2 \approx 1$, $IQR \approx 0$) regardless of initialization.

However, the quantum models demonstrated a notable advantage in parameter efficiency. The optimal curve-fitting quantum architecture (*xparallel*) achieved a test performance of $R^2 = 0.9812$ utilizing only 81 parameters. This generalization capability proved highly competitive with, and in some cases marginally exceeded, the test performance of small-to-medium classical neural networks containing up to 5,950 parameters (testing $R^2 = 0.9796$).

Therefore, while current QTSA models face significant optimization bottlenecks, their ability to abstract temporal patterns and generalize efficiently within a highly constrained parameter space highlights a promising characteristic of quantum representations.

The analysis reviewed five types of QTSA models within the curve-fitting and forecasting model groups. It evaluated their architectural variants and their performance based on their design. Most importantly, it established the connection of model design to its expressivity and trainability. The following section provides a synthesis of the insights derived from this research.

8 Conclusions

Time series (TS) data challenges standard quantum machine learning methods due to high temporal correlation and feature homogenization. Drawing upon established theoretical principles from prior literature, this paper outlines a plausible architectural blueprint for predictive Quantum Time Series Analysis (QTSA), mapping how quantum circuit design influences the fundamental trade-off between model expressivity and trainability.

Our extensive empirical simulations—evaluating 500 model instances—provide strong indicative evidence that confirms general QML phenomena regarding excess entanglement, while also yielding insights specifically tailored to the unique demands of temporal data. Based on these evaluations, we identify three critical design paradigms for QTSA:

- **Combating Feature Homogenization (Novel Insight):** Unlike tabular data, encoding highly correlated temporal sequences into highly entangled circuits risks randomizing the quantum states. We demonstrate that strategic, repeated data re-uploading (as utilized in our *xparallel* and *ovload* architectures) appears essential to maintain state discernibility in the Hilbert space and prevent this homogenization.
- **Managing Entanglements (Confirmation and Extension):** Aligning with prior literature, we confirm that expanding ansatz depth and width to increase expressivity induces excess entanglements and non-local interactions. Our supplementary benchmarks [3] map this directly to QTSA, showing via widening Interquartile Ranges (IQR) that this flattens the cost landscape and degrades model stability, indicating that narrower, entanglement-controlled circuits often outperform wider variants in temporal tasks.
- **Overloading vs. Overparameterization (Novel Application):** To mitigate the trainability penalties of wide circuits without sacrificing learning capacity, we introduce the overloading of fewer qubits with sequential sliding-window data features. This approach regularizes the quantum models and smooths the cost landscape without requiring the excessive auxiliary resources that typically trigger barren plateaus.

Ultimately, this study translates theoretical principles into practical, evidence-based heuristics for constructing predictive QTSA models. While our supplementary results offer a robust empirical and indicative foundation, we are currently undertaking further work to validate these architectural paradigms against rigorous mathematical metrics, such as Krylov expressivity and Effective Dimension.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Cuéllar, M.P., Pegalajar, M.C., Ruiz, L.G.B., Cano, C.: Time Series Forecasting with Quantum Neural Networks. In: Rojas, I., Joya, G., Catala, A. (eds.) Adv. Comput. Intell. pp. 666–677. Springer Nature Switzerland (2023)

2. Cunningham, J., Zhuang, J.: Investigating and mitigating barren plateaus in variational quantum circuits: a survey. *Quantum Inf. Process.* **24**(2) (Jan 2025). <https://doi.org/10.1007/s11128-025-04665-1>
3. Cybulski, J.L.: Code, Results and Supplement for the QUANCOM'2026 paper on Quantum Modelling of Time Series - Expressivity vs. Trainability (Jul 2026), https://github.com/ironfrown/ts_expressivity_v1
4. DeCross, M., Chertkov, E., Kohagen, M., Foss-Feig, M.: Qubit-reuse compilation with mid-circuit measurement and reset (Oct 2022). <https://doi.org/10.48550/arXiv.2210.08039>, arXiv:2210.08039
5. Eldar, Y., Oppenheim, A.: Quantum signal processing. *IEEE Signal Process. Mag.* **19**(6), 12–32 (Nov 2002)
6. García-Martín, D., Larocca, M., Cerezo, M.: Effects of noise on the over-parametrization of quantum neural networks (Feb 2023), <http://arxiv.org/abs/2302.05059>, arXiv:2302.05059, version: 1
7. Goodfellow, I., Bengio, Y., Courville, A.: *Deep Learning*. The MIT Press (2016)
8. Havlíček, V., Córcoles, A.D., Temme, K., Harrow, A.W., Kandala, A., Chow, J.M., Gambetta, J.M.: Supervised learning with quantum enhanced feature spaces. *Nature* **567**(7747), 209–212 (Mar 2019). <https://doi.org/10.1038/s41586-019-0980-2>
9. Holmes, Z., Sharma, K., Cerezo, M., Coles, P.J.: Connecting Ansatz Expressibility to Gradient Magnitudes and Barren Plateaus. *PRX Quantum* **3**(1), 010313 (Jan 2022). <https://doi.org/10.1103/PRXQuantum.3.010313>
10. Larocca, M., Ju, N., García-Martín, D., Coles, P.J., Cerezo, M.: Theory of over-parametrization in quantum neural networks (Sep 2021), <http://arxiv.org/abs/2109.11676>, arXiv:2109.11676
11. Liu, N., Rebentrost, P.: Quantum machine learning for quantum anomaly detection. *Phys. Rev. A* **97**(4), 042315 (2018)
12. Marrero, C.O., Kieferová, M., Wiebe, N.: Entanglement induced barren plateaus. arXiv preprint arXiv:2010.15968 (2020)
13. Masini, R.P., Medeiros, M.C., Mendes, E.F.: Machine learning advances for time series forecasting. *J. Econ. Surv.* **37**(1), 76–111 (Feb 2023)
14. Nielsen, M., Chuang, I.: *Quantum Computation and Quantum Information: 10th Anniversary Ed.* Cambridge University, New York (2010). <https://doi.org/doi.org/10.1017/CB09780511976667>
15. Pérez-Salinas, A., Cervera-Lierta, A., Gil-Fuster, E., Latorre, J.I.: Data re-uploading for a universal quantum classifier. *Quantum* **4**, 226 (Feb 2020)
16. Schuld, M., Sweke, R., Meyer, J.J.: The effect of data encoding on the expressive power of variational quantum machine learning models. *Phys. Rev. A* **103**(3), 032430 (Mar 2021)
17. Weigold, M., Barzen, J., Leymann, F., Salm, M.: Encoding patterns for quantum algorithms. *IET Quantum Commun.* **2**(4), 141–152 (2021)
18. Yarkoni, S., Kleshchonok, A., Dzerin, Y., Neukart, F., Hilbert, M.: Semi-supervised time series classification method for quantum computing. *Quantum Mach. Intell.* **3**(1), 12 (Apr 2021)