scientific reports

Check for updates

OPEN

A quantum machine learning-based predictive analysis of CERN collision events

Sarvapriya Tripathi[1]✉, Himanshu Upadhyay[1] & Jayesh Soni[2]

With the advent of quantum computing, researchers have explored the applicability and any potential advantages of quantum algorithms. This study investigates the application of Quantum Machine Learning (QML) models for regression tasks. Utilizing two distinct CERN datasets (Dielectron events and Proton collision), we investigate prediction accuracy using two QML algorithms, namely Quantum Neural Network (QNN) and Quantum Long Short-Term Memory (QLSTM). We also discuss the comparative analysis and computational efficiency of QNN and QLSTM compared to classical regression methods. The results show that while the two QML models gave comparable accuracy scores to some of the classical models, the best scores were still achieved by some of the more advanced classical algorithms such as CatBoost. Further analysis of the QNN and QLSTM algorithms using multiple ansätz designs showed that increased circuit complexity did not yield substantial improvements in prediction accuracy. These findings suggest that QML models, especially QLSTM with simpler ansätz designs, offer a promising approach for modeling high-energy physics data, and highlight the importance of balancing circuit complexity with performance. The study also underscores the need for further evaluation of these algorithms on quantum hardware to better understand real-world applicability.

Keywords Dielectron events, Proton collision, Quantum long short-term memory, Quantum machine learning, Quantum neural network, Regression

Collision events, fundamental to high-energy physics, give essential information about the universe at its basic level. Processes such as Dielectron production in central Pb–Pb collisions[1] and pp collisions[2,3], serve as indicators of underlying particle interactions or decays of short-lived particles. Analysis of such collision events can help physicists explore fundamental forces and particle behaviors as formulated by the Standard Model of particle physics[4–6]. However, analysis of these events based on the data generated by the Large Hadron Collider (LHC) and other collider experiments remains computationally expensive. These data sets are usually large, having millions of particle collision events that are high dimensional and contain a lot of background noise. The conventional techniques used in physics data before the widespread adoption of machine learning relied heavily on developing explicit equations or algorithms for the physics involved[7,8].

The emergence of machine learning techniques has introduced new opportunities to address the challenges posed by the large datasets in HEP. Algorithms provided by classical Deep learning and Machine learning have shown the possibility of processing data and learning the patterns inherent in the data, which are difficult to achieve with statistical models[9]. However, these advanced models can also sometimes fail due to the need for considerably high computing powers and because of the fact that some of these models are physics-based. In such cases, machine learning also has to incorporate the basic physical processes behind the models and not just learn the patterns[10].

Quantum machine learning also presents a novel approach. Quantum computers are theoretically expected to offer computational advantages over classical systems[11,12]. However, the current generation of Noisy Intermediate-Scale Quantum (NISQ) computers remains limited in qubit count, coherence time, and error rates[13]. This restricts their ability to manage the data volume and complexity typically encountered in HEP applications. Despite these hardware constraints, research in developing efficient quantum algorithms remains essential to the future of quantum computing for scientific domains.

Wiebe et al.[14] investigated this question. Extending the work by Harrow, Hassidim, and Lloyd (HHL)[15] on "Quantum algorithm for linear systems of equations", they provided a positive answer in the form of a quantum algorithm for data fitting. Biamonte et al.[11] definitively summarized the foundational algorithms of quantum

[1]Department of Electrical and Computer Engineering, Florida International University, Miami, FL, USA. [2]Applied Research Center, Florida International University, Miami, FL, USA. ✉email: strip011@fiu.edu

machine learning (QML), while also defining the current software and hardware challenges to the full realization of QML.

The objective of this study is to explore the ability and efficiency of the QML algorithms to handle the unique challenges posed by High-Energy Physics. Here we investigate two well-known CERN datasets (Dielectron events[16] and Proton collision[17]). We then compare the results obtained from QML with those from classical machine learning strategies: simple statistical methods such as classical regression models to deep learning and quantum machine learning methods, which leads us to determine which model is the most suitable for the analysis of complex data related to particle physics when it comes to precision, computational efficiency, and scalability[2,10]. The comparison helps to reveal the prospects of quantum models in high-energy physics. It further helps to understand how quantum computing can be incorporated into different segments of scientific research where large-scale data analysis is consequential. This paper presents a critical analysis and evaluation of these models to provide computational improvement for the advancement of physics models and to identify future breakthroughs in the discipline[18,19]. It further lays the foundation for a comprehensive set of capabilities and use cases of advanced machine learning algorithms in high-energy physics, focusing on quantum machine learning.

The remainder of this paper is structured as follows: The "Literature review" section surveys existing work in the field of Quantum Machine Learning, highlighting key models, algorithms, and architectural choices explored in prior research. The "Methodology" section presents the datasets used, details the data preprocessing steps, and outlines the QML architectures implemented in this study. It further discusses data encoding strategies, variational circuit (ansatz) designs, considerations of computational complexity, and the incorporation of simulated noise in the experimental framework. The "Experimental setup" section describes the experimental configurations, including data partitioning strategies, training parameters, and the simulation and hardware environments used for both classical and quantum models. The "Results" section provides a detailed presentation and analysis of the empirical findings, focusing on the comparative performance of quantum and classical models across multiple regression metrics. The "Discussion" section interprets the observed results, outlines the implications of the findings, and situates them within the broader context of QML research. Finally, the "Conclusion, strengths, limitations, and future work" section summarizes the key outcomes of the study, reflects on its contributions and limitations, and proposes areas for further research in QML and its practical deployment.

Literature review

The development of QML represents a substantial shift in computational science[20] by combining quantum computing capabilities with the learning capabilities of neural networks. The QML algorithmic development has successfully extended several classical ML algorithms using quantum computing, including Quantum Neural Network (QNN)[21–24], Quantum Support Vector Machine (QSVM)[25–27], and Quantum K-nearest Neighbors (QKNN)[28–30]. The QML field has further seen algorithmic development in Quantum Convolutional Neural Network (QCNN)[31–33], and Quantum Generative Adversarial Networks (QGAN)[34–36].

The papers[37,38] introduce a new Quantum model of Long Short-Term Memory (QLSTM) as an advancement over classical LSTM models for learning temporal data. The researchers have emphasized that not only does QLSTM converge faster, it also offers better accuracy in terms of temporal structures, especially the ones that are more complicated.

To extend the Machine learning support in physics, the research by Arpaia et al.[39] introduced the ML approach applied to beam dynamics for better performance at CERN's LHC, marking one of the significant developments in accelerator physics. Methods like anomaly detection, data cleaning, and autoencoders helped the stability and increase in the accuracy of particle collision. Development of electron collision performed in the study by Nam et al.[40] analyzed the stability of the ML model in the classification and analysis of particle interactions, utilizing neural networks to enhance collision classification, experimental designs, and theoretical models. In their paper[41], Yalçin Kuzu demonstrates the use of AI and ML in high-energy physics experiments and in the analysis of dielectrons in collision data. Belis, Odagiu, and Aarrestad[42] provide an overview of state-of-the-art techniques for anomaly detection in particle physics using machine learning.

Liu, Arunachlam, and Temme[43] investigated QSVM for potential speedups. Muser et al.[44] expanded on that effort and identified a speedup utilizing Grover's algorithm in the kernel of a support vector machine. In their paper[45], Gyurik and Dunjko investigate conditions under which QML algorithms can achieve provable exponential speedups over classical methods. Huang et al.[46] present a framework for evaluating quantum advantage in machine learning by showing that, with access to data, classical models can often match or outperform quantum models on certain tasks. They also introduce a projected quantum model that demonstrates a significant prediction advantage over some classical models on engineered data sets. Bowles, Ahmed and Schuld[47] investigate benchmarking of QML models and find that overall, classical machine learning models outperform the quantum classifiers. Bowles, Wierichs, and Park[48] investigate backpropagation scaling in PQCs and find that for a class of novel structured circuits, some simple classification problems show competitive performance while reducing training cost.

Methodology

In this study, we investigated two different datasets from CERN collider experiments (dielectron and Multijet datasets). In large data generation experiments such as LHC, which can record millions of events per second, Quantum Machine Learning models provide a different, data-driven approach to finding complex patterns in the collision data. We investigate these two datasets with Quantum Neural Network (QNN) and Quantum Long Short-Term Memory (QLSTM) algorithms to perform regression analysis. We further compare the accuracy and

Name	Description	Usage
Run	Run number for the event	Not used
Event	Event number	Not used
E1 and E2	Total energy for electrons 1 & 2 in GeV	Feature
px1,py1,pz1,px2,py2,pz2	X,Y,and Z components of momentum for electrons 1 & 2 in GeV	Feature
pt1, pt2	Transverse momentum for electrons 1 & 2 in GeV	Feature
eta1, eta2	Pseudorapidity for electrons 1 & 2	Feature
phi1, phi2	Phi angle for electrons 1 & 2 in radians	Feature
Q1, Q2	Charge for the electrons 1 & 2	Feature
M	Invariant mass of the two electrons in GeV	Target

Table 1. Dielectron dataset features.

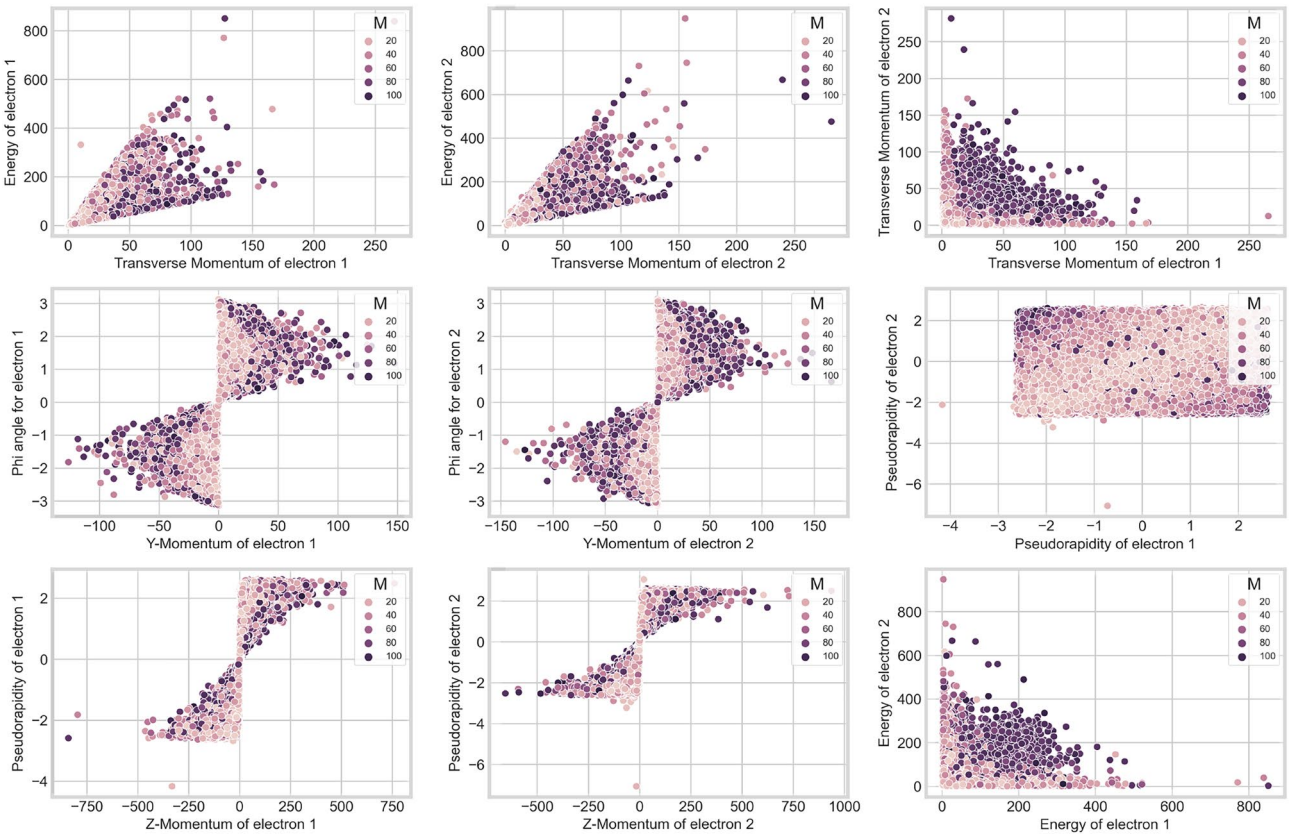


Fig. 1. Scatterplot visualizing correlations in dielectron data.

efficiency of the quantum algorithms with the corresponding classical algorithms, with a focus on determining whether quantum algorithms provide any significant improvements in accuracy and performance[49,50].

Dataset description

The Dielectron Dataset[16] used in this study includes dielectron events within an invariant mass range of 2 to 110 GeV, resulting in a dataset with 100,000 records and 19 attributes, which was further split in a 75:25 ratio to obtain Training and Validation datasets. This data is specifically selected for educational and outreach purposes, representing a subset of the total event information from the CMS experiment. Out of these, the Run and Event attributes were not used for analysis. The features and the target attribute are described in Table 1 and Fig. 1.

The Multijet Primary Dataset[17] is a part of the high-energy physics datasets from CERN, specifically designed for analyzing proton-proton collision events. Researchers use this dataset to search for anomalous jet behavior that could hint at new particles or interactions beyond the Standard Model. This dataset consists of 21,726 records with 17 attributes, which was further split in a 75:25 ratio to obtain Training and Validation datasets. The features and the target attribute for this dataset are described in Table 2 and Fig. 2.

Name	Description	Usage
Run	Run number for the event	Not used
Lumi	Lumi section for the event	Not used
Event	Event number	Not used
MR	First razor kinematic variable	Feature
Rsq	Second razor kinematic variable	Feature
E1,Px1,Py1,Pz1	Four-vector for leading megajet	Feature
E2,Px2,Py2,Pz2	Four-vector for sub-leading megajet	Feature
HT	Scalar sum of transverse momentum of the jets	Feature
MET	Magnitude of the vector sum of missing transverse energy	Target
nJets	Number of jets having transverse momentum above 40 GeV	Not used
nBJets	Number of b-tagged jets having transverse momentum above 40 GeV	Not used

Table 2. Multijet dataset features.

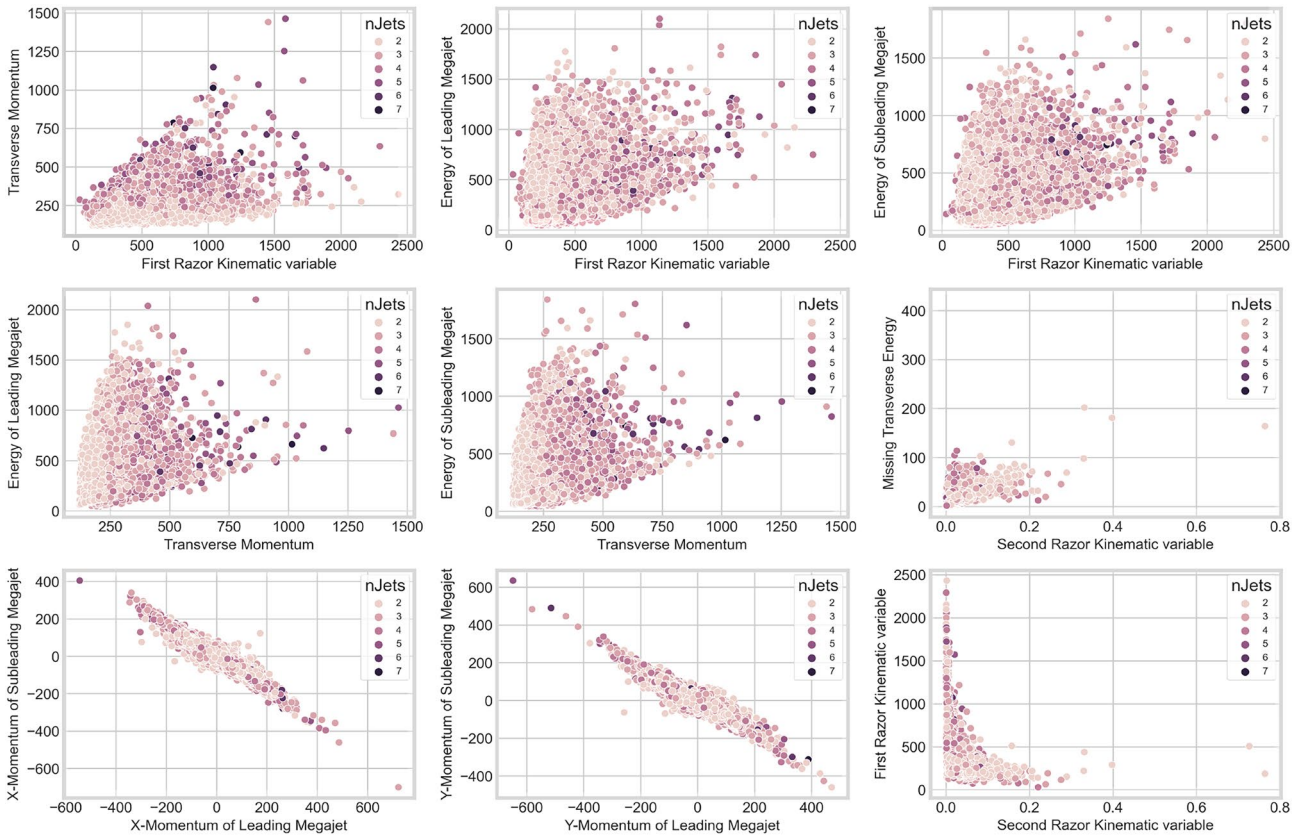


Fig. 2. Scatterplot visualizing correlations in multijet data.

Data preprocessing

The data preprocessing for the Dielectron event dataset included several steps to ensure that it was clean and ready for analysis. First, we perform exploratory data analysis (EDA) to inspect the data and summarize the statistics (Fig. 1). We also checked the correlation and covariance matrices and visualized them to understand the relationship between variables. Finally, we dropped the Run number and Event number attributes of the data from this analysis and kept the focus of the analysis on the core features in the datasets.

The data preprocessing for the Multijet dataset similarly included several steps to ensure that it was clean and ready for analysis. Of the available features, the Run, Luminosity, and Event attributes were not used for this analysis. Furthermore, during the feature engineering phase of our analysis on the dataset, we observed significant class imbalance in the distribution of the nJets and nBJets features. Specifically, over 93% of the data instances exhibited nJets values of 2 or 3 (20,206 out of 21,726 samples), while the remaining values (4–7) were sparsely represented. Similarly, for the nBJets feature, more than 94% of the samples (20,615 out of 21,726) held a value of 0, with minimal representation of other values. Such skewed distributions indicated limited

discriminative utility in the context of our learning task and raised concerns of potential bias or overfitting (Fig. 2). Consequently, these features were excluded from subsequent modeling efforts to evaluate the model's performance using the remaining, more informative features.

Quantum machine learning models

Quantum machine learning workflow

The QML training workflow typically involves the following steps (Fig. 3):

- Problem definition and data preprocessing: The first step involves studying the features of the data and evaluating whether any preprocessing is needed. Reduction may be done using feature selection, which ensures only the most significant subset of the relevant features are selected by using statistical methods or domain knowledge. Dimensionality reduction using techniques such as PCA is also typically employed to meet the restrictions placed by low qubit NISQ environments, some of which were discussed in Data Preprocessing section. The data also need to be normalized to ensure valid quantum states ($\sum |a_i|^2 = 1$).
- Data encoding and quantum circuit design: The selected data is then encoded so as to be applied to quantum circuits. There are various encoding techniques that are discussed in greater detail in the next section. The encoded data is then applied to the quantum circuits. There are several variations and considerations associated with quantum circuit design that are discussed in great detail in[51–55].
- Model training and evaluation: Using the complex algorithms designed with quantum circuits, a QML model is then trained with the encoded data (which forms the input parameters for the quantum layer). The output of the quantum circuit(s) is then measured and fed into a classical system that calculates the cost function and applies it to optimization. One such schematic of a quantum layer, known as a Parameterized Quantum Circuit (PQC) or Variational Quantum Circuit (VQC), is shown in Fig. 3.

Data encoding in quantum systems

The data encoding converts the classical data into a form that can function with quantum systems. The data encoding techniques use the principles of superposition and entanglement to transform the data into a form that can then be applied to quantum circuits. The data encoding could be processed by various methods like Basis Encoding, Amplitude Encoding, Angle Encoding, and Quantum Feature Maps[56–58].

- Basis encoding: This is the simplest type of encoding. In this system, the computational state of an n-qubit system is directly associated with a classical n-bit string. For a given binary input dataset S of features $x \in S$

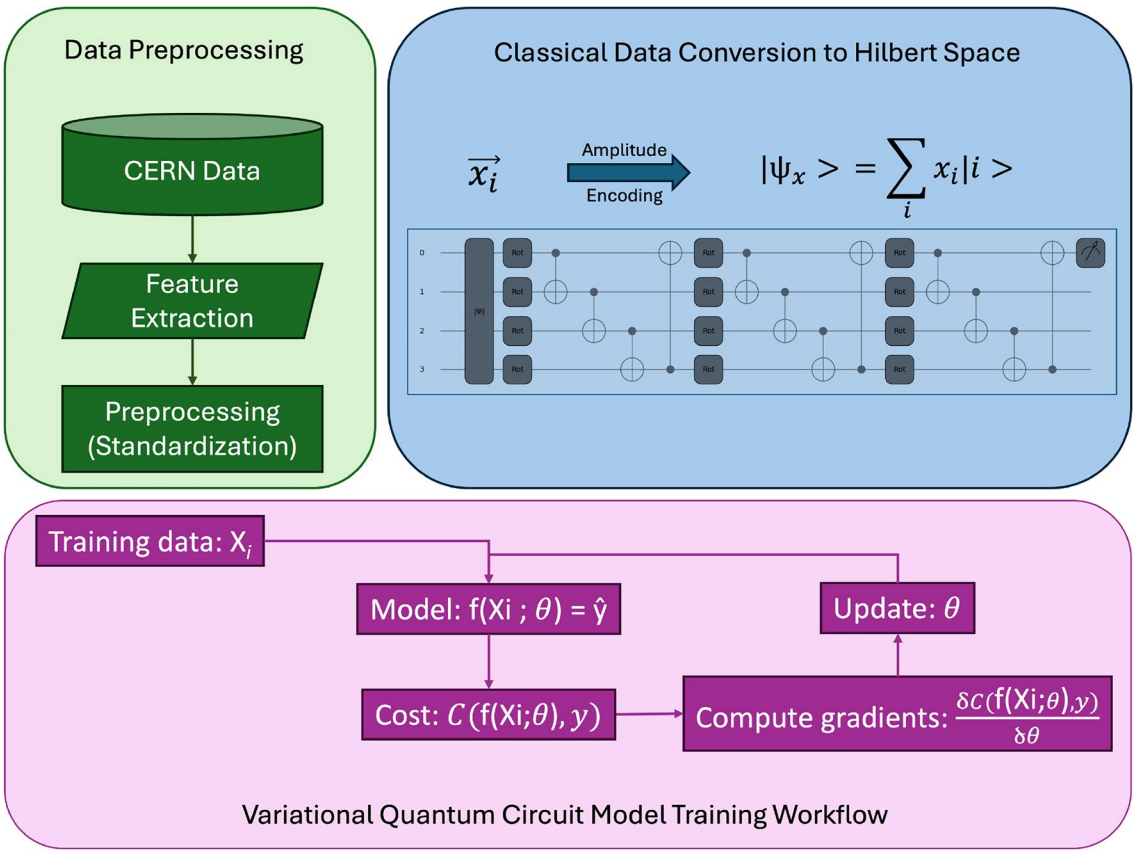


Fig. 3. The architecture and pipeline for QML model training.

where each feature $x^m = (b_1^m, ..., b_N^m)$ for $b_i^m \epsilon (0, 1)$ for $i <= N$, the superposition basis states $|x^m\rangle$ are given by the binary input pattern

$$|S\rangle = \frac{1}{\sqrt{M}} \sum_{n=1}^{\infty} |x^m\rangle \tag{1}$$

This encoding is inefficient for large datasets and is not suitable for continuous or high-dimensional data without further processing.

- Amplitude encoding: Here, the classical data are encoded as the amplitude of a quantum state. This encoding requires data to be normalized. A classical vector $x \epsilon \mathbb{C}^{2^n}$, where $\sum_k |x_k|^2 = 1$ may be amplitude-encoded to quantum state $|\psi\rangle \epsilon \mathscr{H}$ as

$$x = \begin{pmatrix} x_1 \\ . \\ . \\ . \\ x_m \end{pmatrix} \leftrightarrow |\psi_x\rangle = \sum_{i=1}^{2^n} x_i |i\rangle \tag{2}$$

- Angle encoding: In this encoding method, the classical data are encoded into the rotation angle of the qubits. In Angle Encoding, the rotations corresponding to features $x = (x_1, ..., x_N)$ for $x_i \epsilon \mathbb{R}$ and $i <= N$, are applied to N qubits such that

$$|\psi(x)\rangle = \bigotimes_{i=1}^{N} (\cos(\frac{x_i}{2}) |0\rangle + \sin(\frac{x_i}{2}) |1\rangle) . \tag{3}$$

This encoding typically requires N qubits for N features. There is a variation of this encoding, often referred to as Dense Encoding[58], that fits 2N features in N qubits by using both y-axis and z-axis rotations.

Ansatz designs

We experimented with three different ansatzes, with increasing complexity, to study the effect of the circuit design on the final accuracy of the models. These ansatzes are depicted in Fig. 4, and are described below. A deeper analysis of the computational complexity of QML models is done later in this section.

- Ansatz 1: This is the least complex of the three circuits, and consists of amplitude-encoded data undergoing X rotation and CNOT imprimitives as shown. This entanglement is equivalent to the BasicEntanglerLayers template in Pennylane[59].
- Ansatz 2: This circuit has an extra degree of complexity in that here the amplitude-encoded data undergoes X, Y, and Z rotations. The extra rotation operations allow for more fine-tuning of weights compared to what is available in Ansatz 1.
- Ansatz 3: This is the most complex of the three circuits, and consists of two layers of rotation and CNOT operations as shown. The extra operations make this circuit the slowest of the three. The extra layer of operations allows for more complex fine-tuning of weights. This design is equivalent to the StronglyEntanglingLayers template in Pennylane.

These three ansatzes were used in both QNN and QLSTM models (detailed in next section), and results were compared between the respective ansatzes.

Quantum neural network

It is well known that Artificial Neural Networks (ANNs) are the fundamental blocks of deep learning as well as other architectures such as Recurrent Neural Networks, Convolutional Neural Networks, and Transformers. The Quantum Neural Network (QNN) implements ANN in quantum systems. The illustration in Fig. 5 displays the general architecture of the QNN that mimics the ANN.

The experiments were conducted with QNN models designed for each of the three ansatzes described earlier. The underlying harness for the computation was created with PyTorch's torch.nn module. Our design of the QNN models has been optimized to handle regression tasks. The QNN Regression model's configuration, as depicted below in Algorithm 1, was designed based on the properties of the datasets and, therefore, was different for both Dielectron and Multijet datasets.

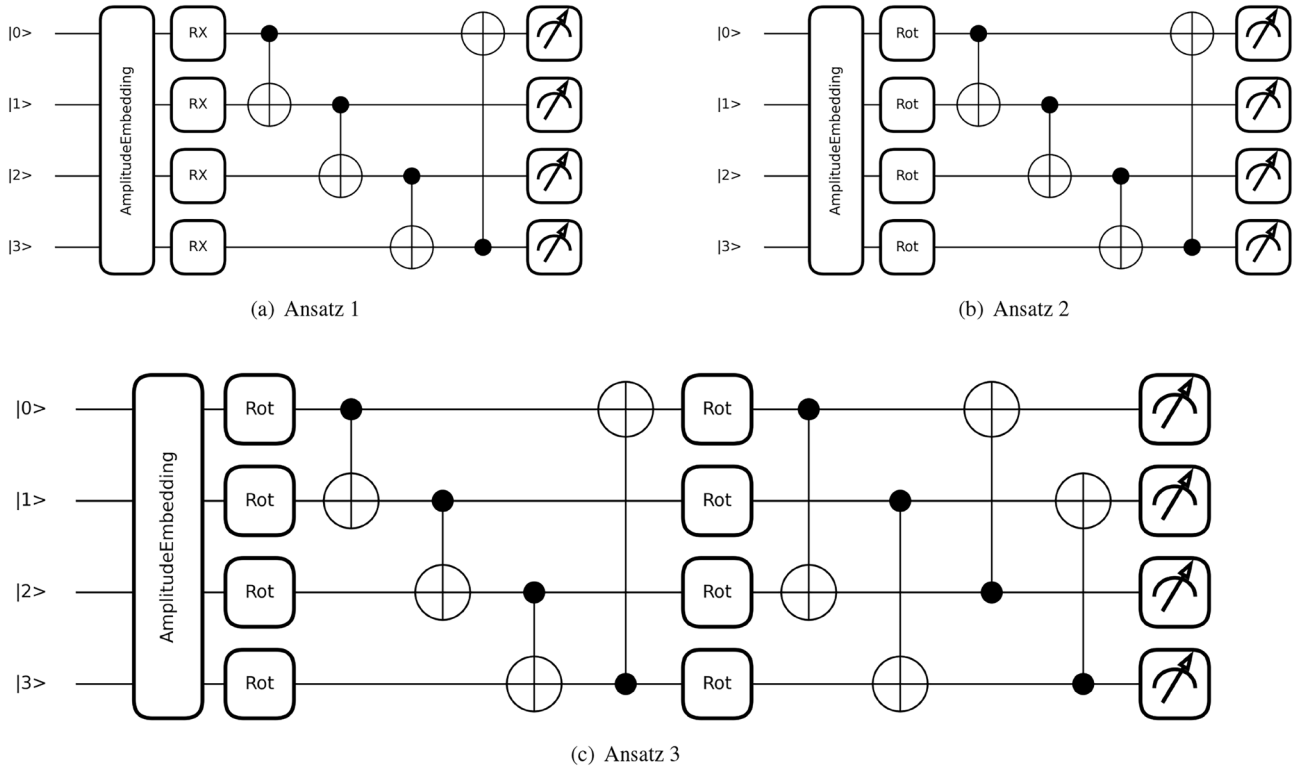


Fig. 4. Diagrams for three ansatzes used in this study.

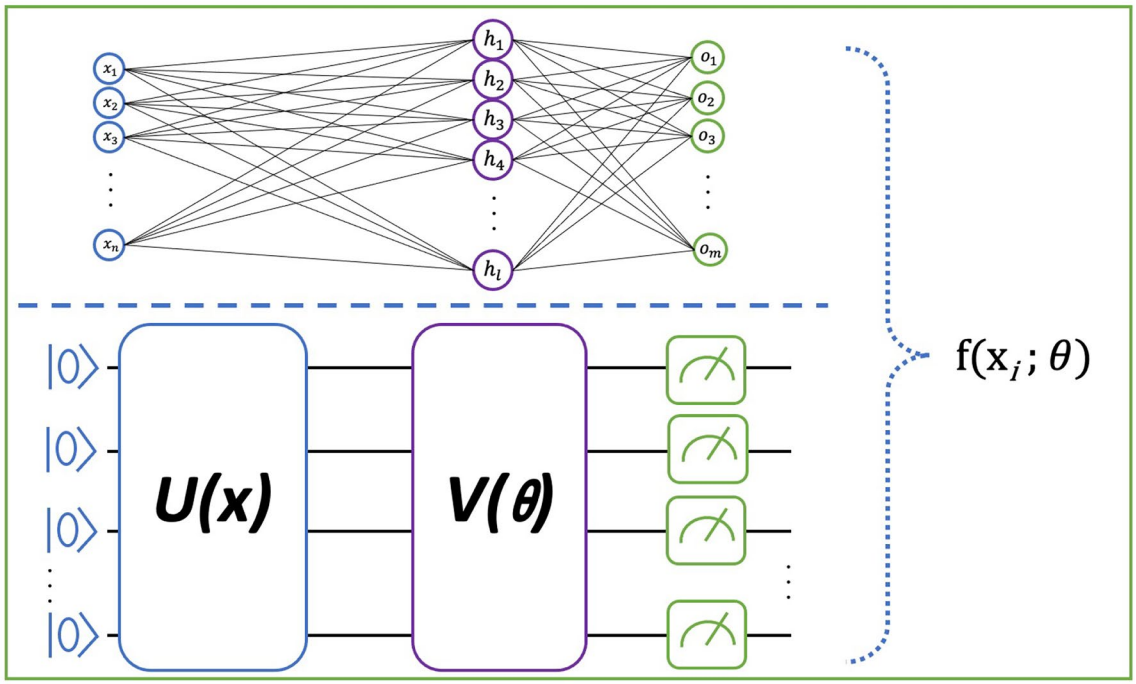


Fig. 5. Generic QNN Design. Here U is the encoding layer, and acts on the input data x. V is the quantum layer, and is tuned using tunable parameters θ.

Require: $featureSize = NumberOfInputDataFeatures$
Require: $nbrQubits = NumberOfQubits$

```
QNNModel(                /*Uses PyTorch nn.Module*/
  (clayer_in): Linear(in_features=featureSize, out_features=nbrQubits,
                      bias=True)
  (qlayer): <Quantum Torch Layer: func=qnode>
  (clayer_out): Linear(in_features=nbrQubits, out_features=1, bias=True)
  (linear): Linear(in_features=1, out_features=1, bias=True)
)
```

Algorithm 1. QNN model design

Quantum LSTM

The generic LSTM model was developed to handle the limitations of the Recurrent Neural Network (RNN), such as the vanishing gradient problem and the handling of the long-term dependencies. The primary deficiencies the RNN faced were the extreme deprecation (vanishing gradient) and the explosion of the gradients that resulted in imbalances in the weights of the model. Hochreiter and Schmidhuber[60] proposed to tackle the LSTM architecture issues by introducing the forget gate, input gate, and output gate in order to maintain the information in the cell state.

The quantum variant of the LSTM model follows the architecture (Fig. 6) of the LSTM model with few modifications, in order to facilitate quantum computations[37,38,61]. The original feature data undergoes encoding before being input to QLSTM cell to make it suitable for quantum operations. This encoding process has been explained in detail in the previous section. The quantum LSTM gates perform similarly to the generic LSTM-based gates. The critical information about each of the gates is described below and in Eq. (4):

- Forget gate: The gate controls the information in the cell state that needs to be discarded or kept.
- Input gate: The gate determines the information stored in the cell state.
- Output gate: The gate regulates the output from the current cell state.

$$f_t = \sigma(\mathrm{VQC}_f[h_{t-1}, x_t]) \tag{4a}$$

$$\tilde{C}_t = \tanh(\mathrm{VQC}_c[h_{t-1}, x_t]) \tag{4b}$$

$$i_t = \sigma(\mathrm{VQC}_i[h_{t-1}, x_t]) \tag{4c}$$

$$o_t = \sigma(\mathrm{VQC}_o[h_{t-1}, x_t]) \tag{4d}$$

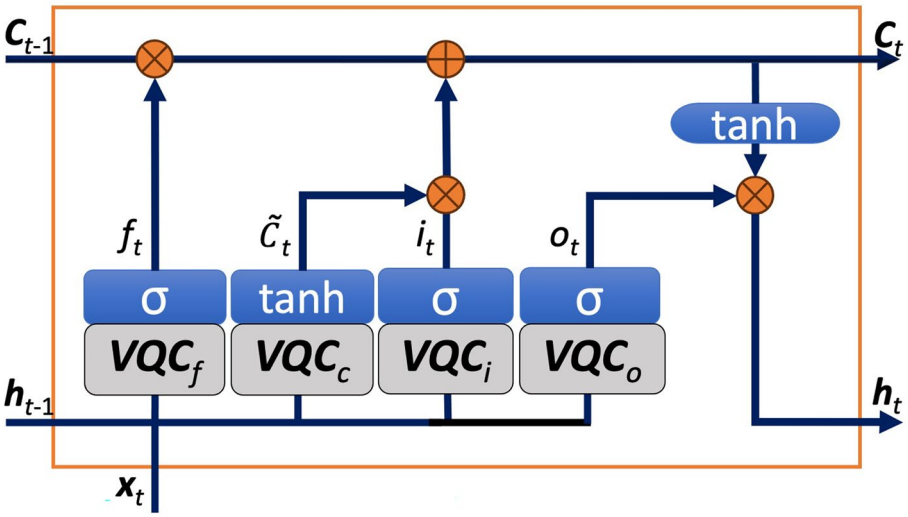


Fig. 6. Configuration of a quantum LSTM cell. Here X_t represents the input feature at timestep t, C_{t-1} and h_{t-1} represent the carryover cell state and hidden state from the previous timestep, and f_t, $\tilde{C}_t$, i_t, o_t represent *forget*, *candidate*, *input*, and *output* gates respectively.

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \tag{4e}$$

$$h_t = o_t \odot \tanh(C_t) \tag{4f}$$

The QLSTM implementation (Fig. 7) comprises a quantum circuit containing one layer and four qubits. It used MSELoss as the loss function, the default torch implementation for quantum processing as the device, and Adagrad as the optimizer. Pennylane was used as the quantum framework. Each gate reused the same functionalities of the generic LSTM for the quantum circuits. The circuit used amplitude embedding with strongly entangled layers and returned the expectation value for the results. The remaining implementation mimics the LSTM control flow.

The experiments were conducted with QLSTM models designed for each of the three ansatzes described earlier. Similar to QNN, the underlying harness for the computation was created with PyTorch's torch.nn module. The QLSTM Regression model's configuration, as depicted below in Algorithm 2, was designed based on the properties of the datasets and, therefore, was different for both Dielectron and Multijet datasets.

Require: $featureSize = NumberOfInputDataFeatures$
Require: $nbrQubits = NumberOfQubits$

```
QRegressionLSTM(              /*Uses PyTorch nn.Module*/
  (lstm): QLSTM(
    (clayer_in): Linear(in_features=featureSize, out_features=nbrQubits,
                        bias=True)
    (qlayer): <Quantum Torch Layer: func=qnode>
    (clayer_out): Linear(in_features=nbrQubits, out_features=2, bias=True)
  )
  (linear): Linear(in_features=2, out_features=1, bias=True)
)
```

Algorithm 2. QLSTM model design

Additional considerations on choice of Quantum LSTM for this study
Although LSTM models are traditionally employed to capture dependencies across multiple time steps in sequential data, our use of Quantum LSTM in this study was motivated by a different objective. The datasets used, specifically CERN's Dielectron and Multijet events, comprise high-dimensional feature vectors, each representing a single, self-contained physical event rather than a temporal sequence. Despite the lack of inter-sample temporal correlation, intra-sample dependencies among features (e.g., between momenta, pseudorapidity,

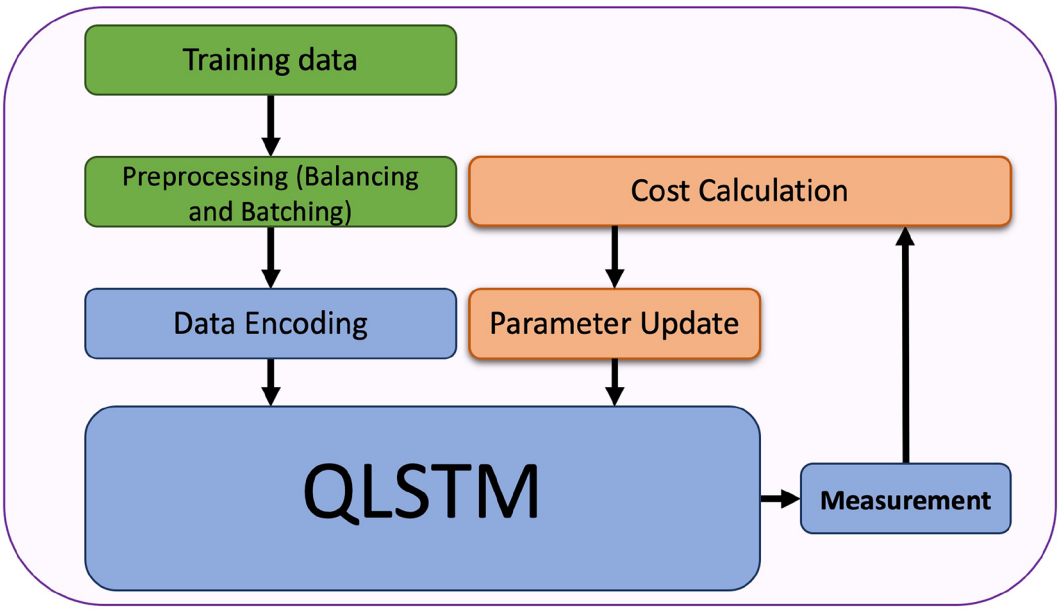


Fig. 7. Quantum LSTM algorithmic implementation.

and energy levels) can be complex and non-linear. In this context, we conceptualized each QLSTM gate as a feature-processing mechanism rather than a temporal memory unit. Even when configured with a sequence length of 1 (as is the case in this study), the QLSTM model retains its gating architecture and leverages variational quantum circuits to enable nonlinear transformations across entangled quantum states representing the input features. Thus, the internal gating structure serves to model latent dependencies within each high-energy event's feature space rather than traditional time-based memory.

This formulation permits the QLSTM architecture to act as a quantum-enhanced, dynamically weighted feature projector. The forget, input, and output gates, though typically interpreted through a temporal lens, are repurposed here to act as variational selectors and reweighing components that govern how feature representations evolve across layers of the model. The entanglement introduced by quantum operations in each gate enables exploration of complex, higher-order relationships among input features, relationships that may not be effectively captured by conventional feedforward networks. By maintaining architectural parity with LSTM-based baselines, the use of QLSTM in this setting also allows for a controlled comparison of quantum versus classical gated designs. This approach aligns with broader efforts in QML to explore how quantum-inspired gating mechanisms can be reinterpreted beyond sequential tasks to provide expressivity advantages in high-dimensional static datasets.

Computational complexity of QML models

The application of Variational Quantum Circuits (VQCs) is central to the architecture of QNN and QLSTM models, and comes with distinct computing resource requirements in emulation environments. The various components such as encoding, computation, and measurement, required to implement VQC have a computational overhead that needs to be planned carefully. Experimenting in emulation environments (e.g. Pennylane in our tests), the complexity of the calculations can be estimated as described in Table 3.

The computational complexity of the VQC overall is $\mathcal{O}(n_q^2.n_l)$, and that makes QNN and QLSTM in an emulation environment slower than corresponding classical algorithms (ANN and LSTM respectively). This was observed in our tests, where our QML test runs took longer to complete as compared to classical methods.

Effect of noise in QC

Sampling noise (Shot noise) is a fundamental source of uncertainty in quantum computing, where circuit outputs are estimated via repeated measurements. This noise stems from the quantum mechanical nature of the measurement process and the fundamental limits it imposes on the precision and reliability of quantum circuit outputs. Clerk[62] explains that quantum noise originates not only from Heisenberg constraints but also from Zero-point motion and Frequency asymmetry. In their work, Barron et al.[63] discuss and establish methods to formally quantify the sampling overhead. Nico-Katz, Keenan, and Goold[64] develop a scalable, device non-specific protocol for quantifying idle information loss. In addition, quantum circuits can experience unintended interactions or fluctuations that may alter the behavior of a quantum system from its ideal evolution. This noise can arise in several operations and stages in quantum circuits, such as bit flip, phase flip, amplitude damping, and phase damping, etc[65]. We briefly touch upon the impact of noise in our experiments by comparing the effect on QNN model training by introducing simulated amplitude damping noise. The results and the analysis of those experiments are presented in the Results section.

Training and evaluation

Training

The datasets used for training and testing various models were split with 80% of the data used for training and 20% of the data reserved for testing. This proportion of data split ensures that the models are trained on large volumes of data, where they learn data patterns and help the model predict the target variable using unseen testing datasets.

Evaluation metrics

The performance measurements of the trained models were done using the following evaluation metrics:

- Mean squared error (MSE): MSE measures the average of the square of the difference between predicted and actual values. If the MSE value is lower, the model performs better.

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{5}$$

Task	Complexity
Data encoding	$\mathcal{O}(n_f)$. Here n_f represents the number of features. The complexity of this step stems from the need to convert the inputs to the appropriate parameters for quantum circuits (such as angles for angle-encoding)
Variational computation	$\mathcal{O}(n_q^2.n_l)$. Here n_q represents the number of qubits and n_l represents the number of layers used in the circuit. The complexity of this step encompasses the entanglement operations and the rotation operations that are applied to qubits using learnable parameters
Quantum measurement	$\mathcal{O}(n_q)$. Here n_q is the number of qubits. The complexity of this step comes from the operations needed to measure each qubit

Table 3. Computational complexity of quantum operations.

- Root mean squared error (RMSE): RMSE is the square root of Mean Squared Error, and indicates the average magnitude of the error. Like MSE, lower values of RMSE indicate better performance.

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{6}$$

- Mean absolute error (MAE): MAE measures the average of the absolute errors. It provides a measure of the average magnitude of the errors in a set of predictions, without considering their direction. The smaller the MAE values, the better the model's performance.

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{7}$$

- Mean absolute percentage error (MAPE): MAPE measures the size of the error in percentage terms. It is determined by calculating the mean of the absolute percentage errors of the predictions. Essentially, a lower MAPE value tells us the model is performing better.

$$\mathrm{MAPE} = \frac{100}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \tag{8}$$

These metrics help us to evaluate and compare the performance of different models, which allows us to choose the best and most reliable model for predicting the invariant mass of dielectron events.

Experimental setup

This study used various quantum and classical regression models to predict the invariant mass (M) in the Dielectron dataset, and the missing transverse energy (MET) in the Multijet dataset. We focused on Quantum Neural Network (QNN) and Quantum Long Short-Term Memory (QLSTM) and compared those with the corresponding classical models of Artificial Neural Networks (ANN), and Long Short-Term Memory (LSTM). We analyzed the models based on their performance, including how fast they converged to a solution. The evaluation metrics used to measure the performance of the models are MSE, RMSE, MAE, and MAPE.

The experiments were done in two parts. In the first part, the quantum models were tested against classical machine learning and deep learning models. The quantum models were created using Ansatz 1 (Fig. 4), and the quantum and classical models were all trained for 20 epochs to keep parity between various algorithms. These results are discussed in the Results section.

In the second part, the quantum models were extended to consider two more ansatzes in increasing order of complexity, as detailed in the methodology section, and the performance of the quantum models (loss, accuracy, and execution times) were compared against each other. This was a rigorous test, where 6 quantum models, one model for each of the three ansatzes for both QNN and QLSTM, were prepared and tested for both of the datasets. Each model was trained for 20 epochs, and this training process was repeated 20 times each. The execution statistics were then averaged, and are discussed in detail in results section.

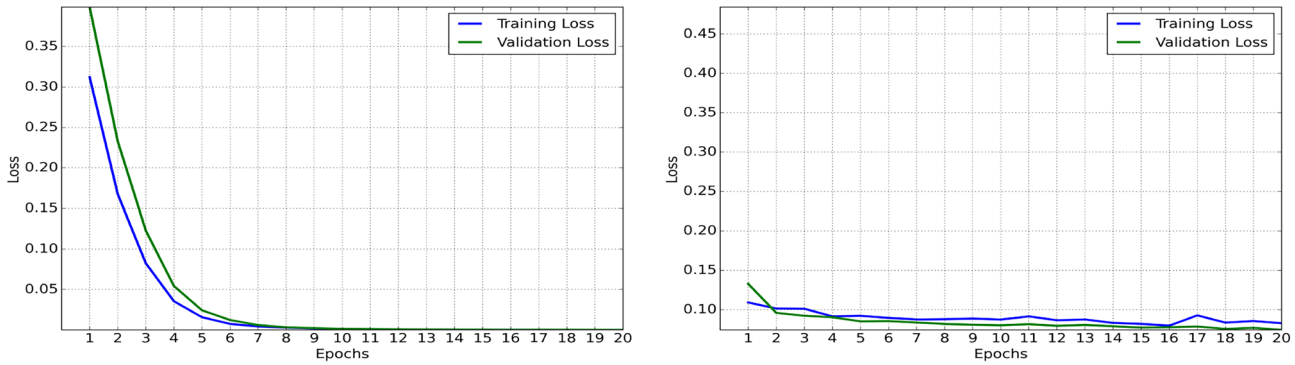
For quantum models in these experiments, we used the PennyLane framework[59] with PyTorch. PennyLane is an open-source library for QML and integrates well with other common Machine Learning packages. This integration provides for using PyTorch's capabilities while constructing and training QNN and QLSTM models. The number of qubits was determined based on the number of features (16 for the Dielectron dataset and 11 for the Multijet dataset) after data pre-processing, which involved the removal of some features. The flow starts with all the qubits in superposition, and the circuit comprises multiple layers of quantum operations. In each layer, qubits undergo rotation, and each qubit is entangled with its adjacent qubit in a rotational process. The final layer allows the circuit to be measured to produce the output for the record. This has been discussed in greater detail in the methodology section. The execution physical environment consisted of a Windows Server with Intel Xeon CPUs, 128 GB RAM, and an NVidia A100 graphics processor as described in Table 4.

Component	Specification
Operating system	Linux 6.1.0-25-amd64-x86_64 with glibc 2.36
CPU	Intel(R) Xeon(R) Gold 5218 CPU @ 2.30GHz (12 physical cores)
RAM	128 GB
GPU	NVIDIA A100
NVIDIA Driver Version	560.35.03
CUDA Version	12.6

Table 4. System specifications for experimental setup.

parameters	QLSTM	QNN	LSTM	ANN
Learning rate	0.01	0.01	0.001	0.001
Batch size	32	32	32	32
Epochs	20	20	20	20
qubits	4	4	–	–
Optimizer	Adagrad	SGD	Adam	Adam
Activation	Linear	Linear	Relu	Relu

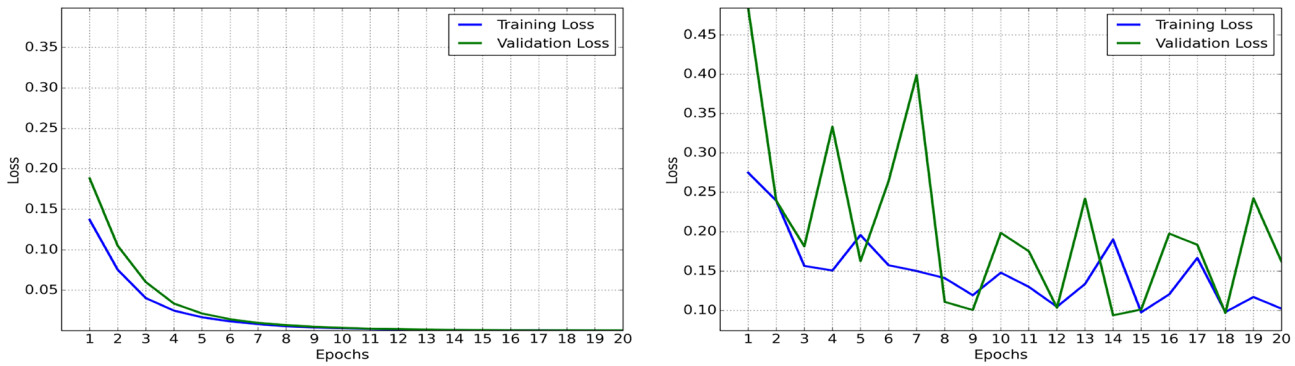
Table 5. Hyper-parameters for quantum and classical models.



8.1 - Quantum Neural Network

8.2 - Quantum Long Short-Term Memory

Fig. 8. Dielectron dataset—model performance: training vs testing loss.



9.1 - Quantum Neural Network

9.2 - Quantum Long Short-Term Memory

Fig. 9. Multijet dataset—model performance: training vs testing loss.

Hyperparameters

The hyperparameters used in the experiments are given in the Table 5. The selection of hyperparameters for both quantum and classical models in this study was guided by a structured empirical process. For quantum machine learning models, amplitude encoding was employed to efficiently represent input data within the quantum circuit, with circuit depths tailored to the dimensionality of the feature space. Parameters such as batch size, number of epochs, optimizer type, and learning rate were optimized through iterative trial-and-error procedures. Also noteworthy is the fact that QLSTM model was trained with a sequence length of 1, the reasoning for which has been discussed before. These exploratory runs were conducted to identify stable parameter ranges that enabled model convergence while mitigating risks of overfitting and excessive computational cost. A comparable strategy was adopted for the classical architectures, including LSTM and ANN, as well as advanced models like CatBoost. In each case, hyperparameters were selected based on preliminary experimental runs and aligned with established practices reported in prior literature.

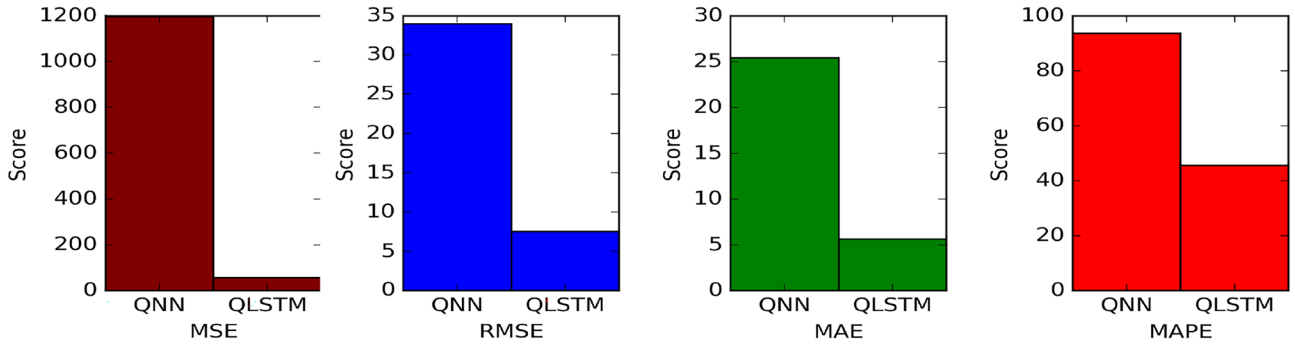


Fig. 10. Dielectron dataset—QML models performance metrics.

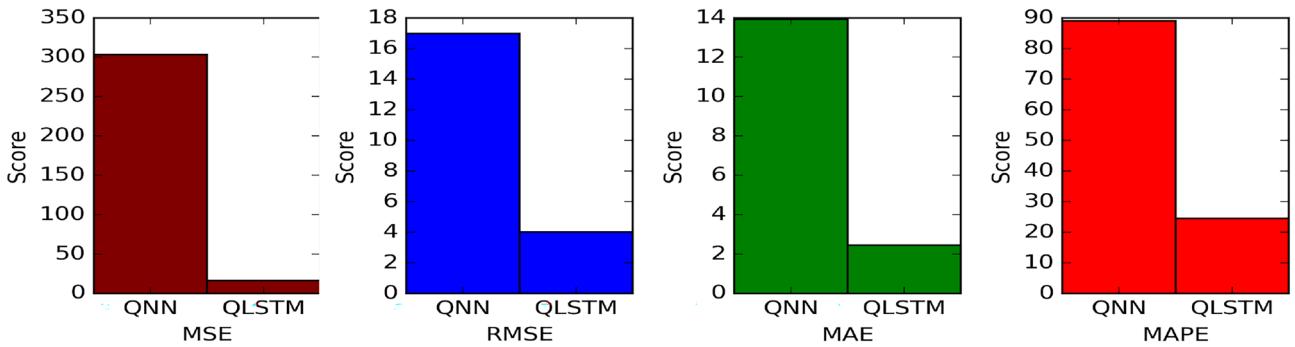


Fig. 11. Multijet dataset—QML models performance metrics.

Results

Quantum model performance

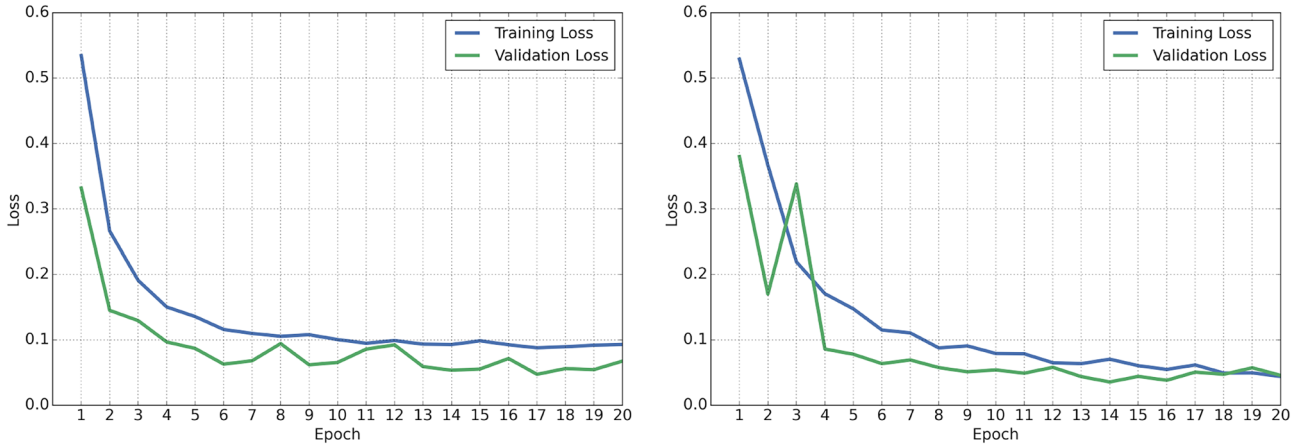
The training and validation loss trends of QNN and QLSTM models for predicting the invariant mass (M) of the Dielectron dataset are shown in Fig. 8 and for predicting Missing Transverse Energy (MET) in the Multijet dataset are shown in Fig. 9. As the results show, the QNN model for both the datasets converged to a solution (9 epochs for Dielectron and 17 epochs for Multijet). QLSTM for these datasets showed a similar characteristic, where the model quickly converged to a solution for the both datasets. For Dielectron dataset the validation loss for QLSTM model went below 0.1 from epoch 4, whereas for the Multijet dataset, the QLSTM model validation loss went below 0.1 in 14 epochs. Owing to the overall complexity of QML and overheads of the simulation environment, both the QNN and QLSTM took longer to train compared to the classical models. Overall, both the QNN and QLSTM models converged to a solution.

The performance of the quantum models, as illustrated in Figs. 10 and 11, indicates a consistent superiority of the QLSTM model as compared to the QNN model, thus highlighting the QLSTM model's capabilities. Overall, the QLSTM model's performance was comparable or better than classical LSTM performance with reduced variance of the predictions. For the Dielectron dataset, the MSE score of the QLSTM model was 52 which was 22% better compared to that of the classical LSTM model which stood at 67. Similarly, for the Multijet dataset, the MSE for the QLSTM model was 11, which was 69% better than the MSE of 37 for the classical LSTM model. These metrics support the suitability of the QLSTM model for this problem domain.

Classical models performance

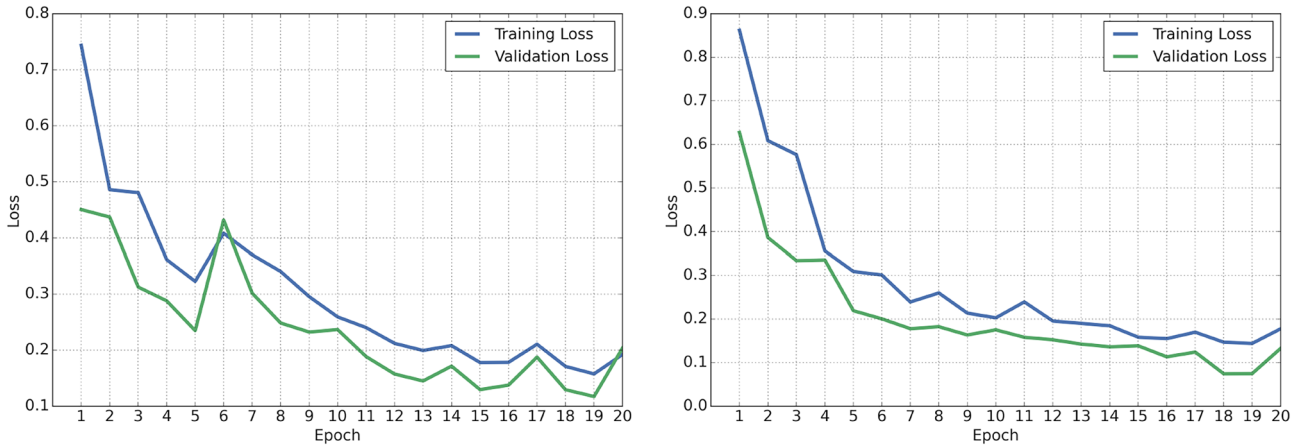
The training and validation loss trends of ANN and LSTM models for predicting the invariant mass (M) of the Dielectron dataset are shown in Fig. 12 and for predicting Missing Transverse Energy (MET) in the Multijet dataset are shown in Fig. 13. These plots show that for both of these datasets, the classical models converge to a solution quickly, although ANN and LSTM have a relatively higher loss for the Multijet dataset as compared to the Dielectron dataset when trained for an identical number of epochs (20 in our experiments). Both ANN and LSTM models show a decreasing loss trend over epochs, indicating effective learning and convergence during the training process. The ANN model shows a gradual decrease in training and validation loss, indicating strong generalization ability. The validation loss of the LSTM model stays a bit steady, while the training loss decreases rapidly at first. Then, the depreciation slows down, representing that the model is well-fitted.

Figures 14 and 15 show the evaluation metrics for various classical machine and deep learning models. Linear regression models, such as Linear Regressor, Ridge Regressor, Lasso Regressor, and ElasticNet Regressor, we see a moderate fit, suggesting a limited ability to capture the variation in the data. The Random Forest Regressor performs significantly better, showing its strength in handling complicated data patterns, than the Decision Tree



12.1 - Artificial Neural Network (ANN) 12.2 - Long Short Term Memory (LSTM)

Fig. 12. Dielectron dataset—model performance: training vs validation loss.



13.1 - Artificial Neural Network (ANN) 13.2 - Long Short Term Memory (LSTM)

Fig. 13. Multijet dataset—model performance: training vs validation loss.

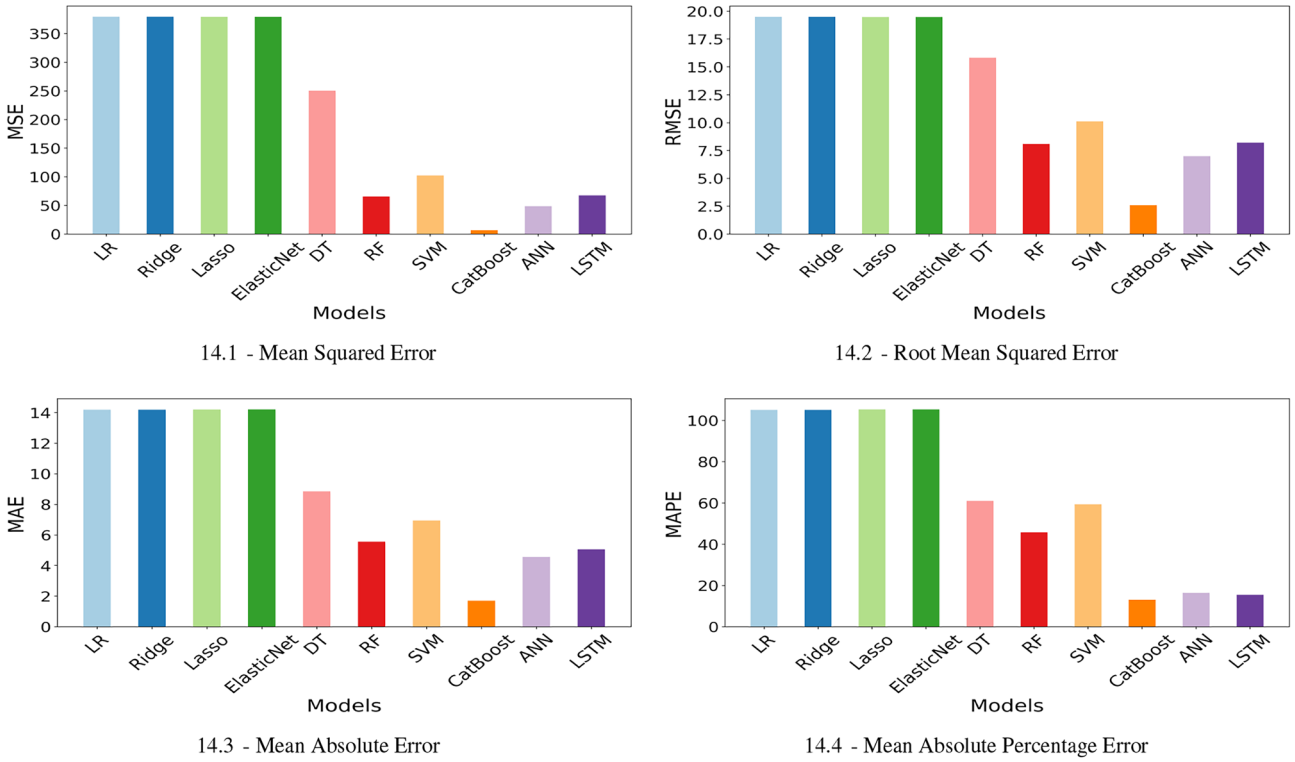
Regressor. The Support Vector Machine (SVM) Regressor also performs well, but the CatBoost Regressor shows the strongest performance. The ANN and LSTM models demonstrate their strength too. The overall analysis shows that deep learning models such as ANN and LSTM, ensemble algorithms such as Random Forest, and boosting algorithms such as CatBoost better understand the data complexity and predict the invariant mass for the Dielectron dataset.

As we can see in the comparative performance trends of machine learning (ML) and deep learning (DL) models, advanced models like Random Forest, CatBoost, ANN, and LSTM predict better than decision tree regressor and linear regression models like LR, Ridge, Lasso, and ElasticNet. The evaluation of MSE, RMSE, MAE, and MAPE indicates how well the CatBoost, ANN, and LSTM models perform. These models perform better by generating lower error metrics. Linear regression models give higher error values across all metrics, indicating their inadequacy in minimizing prediction errors. Finally, in terms of performance, decision tree and SVM models lie between linear models and advanced ensemble or deep learning models.

Quantum models performance for various ansatzes

In the second part of the experiment, 6 models (one each for three ansatzes for QNN, and same for QLSTM) were developed for trained for both the Dielectron and Multijet datasets. Each model was trained for 20 epochs, and this was repeated 20 times to account for any computing environment fluctuations. All of the training times were collected and averaged to obtain training time per epoch in seconds as shown in Table 6.

In line with the complexity of the ansatzes used, the training times of the models increased consistently and significantly going from Ansatz 1 to Ansatz 3. Both types of algorithms, QNN and QLSTM, exhibited this pattern. Furthermore, compared to the corresponding classical models, the training of the quantum models was slower by factors ranging from 3 to 29.



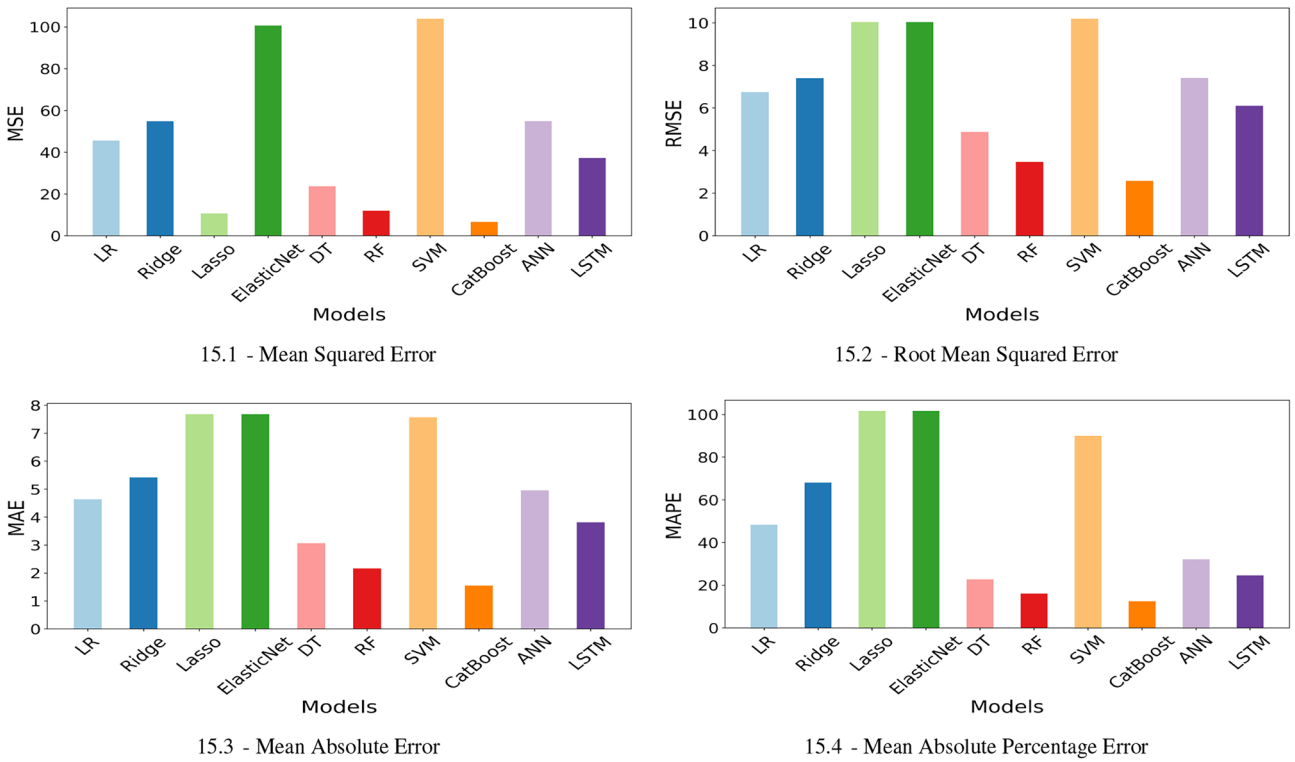
14.1 - Mean Squared Error

14.2 - Root Mean Squared Error

14.3 - Mean Absolute Error

14.4 - Mean Absolute Percentage Error

Fig. 14. Dielectron dataset—evaluation metrics across various machine learning and deep learning models.



15.1 - Mean Squared Error

15.2 - Root Mean Squared Error

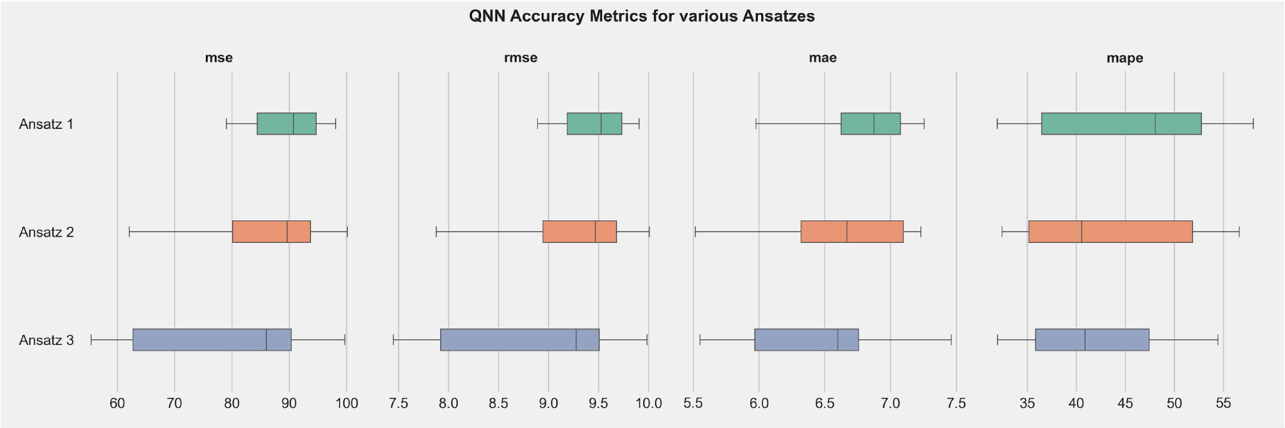
15.3 - Mean Absolute Error

15.4 - Mean Absolute Percentage Error

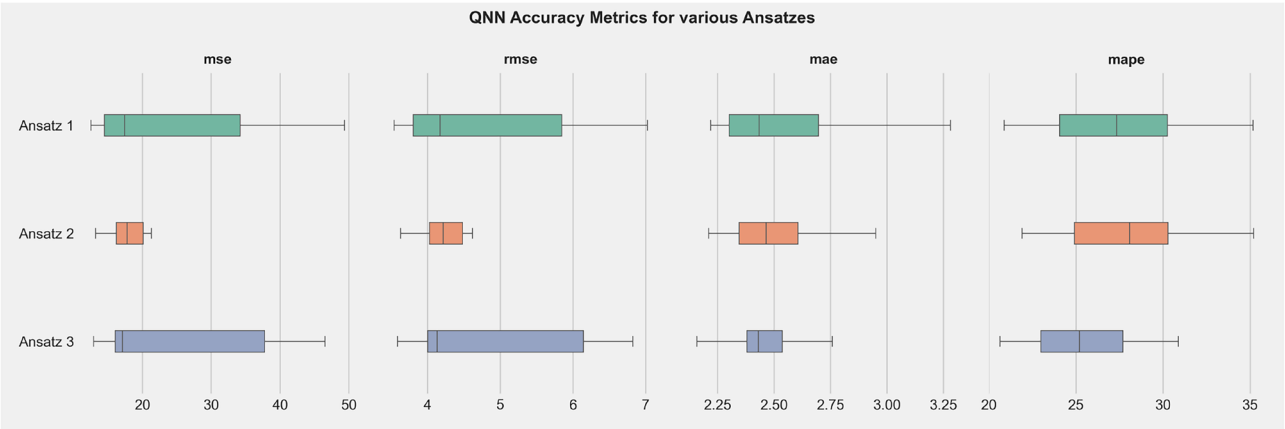
Fig. 15. Multijet dataset—evaluation metrics across various machine learning and deep learning models.

Ansatz	QLSTM			LSTM	QNN			ANN
	1	2	3		1	2	3	
Dielectron	48	74	123	17	43	51	72	14
Multijet	42	71	116	4	36	41	55	3

Table 6. Execution times for quantum and classical models, in seconds per epoch.



(a) Performance Metrics for ansatzes for Dielectron Dataset



(b) Performance Metrics for ansatzes for Multijet Dataset

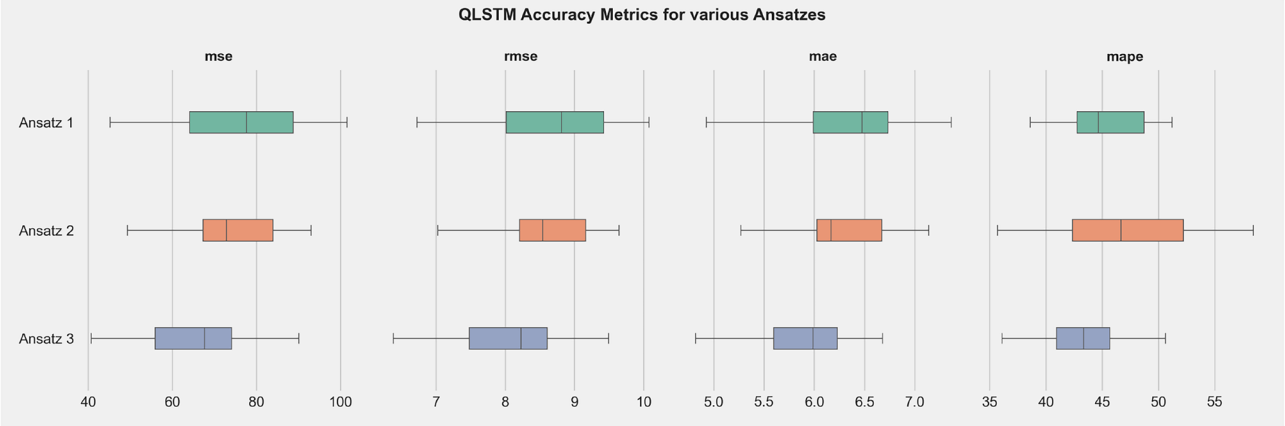
Fig. 16. QNN models performance metrics. The box plots show the median, IQR, minimum, and maximum for the respective metrics.

To evaluate the impact of ansatz choice on model accuracy, we also captured the accuracy metrics (mse, rmse, mae, and mape as discussed in the methodology section) during these runs. The results were then averaged, and the metrics were compared by ansatzes. The results are summarized for both QLSTM and QNN models using boxplots, showing the median, Inter-Quartile Range (IQR) and minimum and maximum for each statistic, and are shown in Figs. 16 and 17.

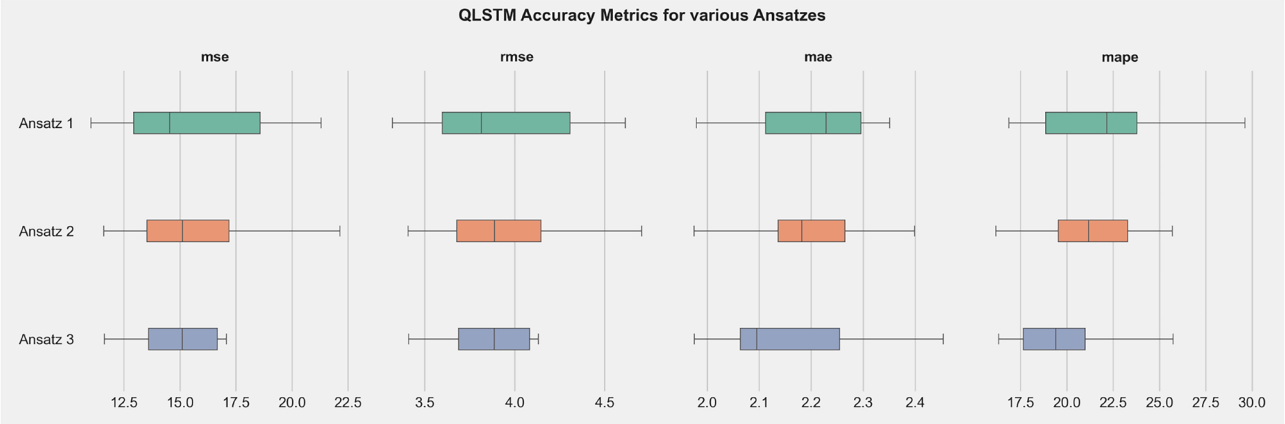
As observed in the mse accuracy metrics above, while Ansatz 3 shows the most favorable performance across all four metrics with the lowest median values, It was also the most complex of the three ansatze. However, the very modest improvement in the metrics for ansatz 3 may not justify the added complexity. This was true for both the quantum algorithms (QNN and QLSTM) and for both the datasets. Since models based on Ansatz 2 and 3 were significantly more computationally expensive to train, the small increase in accuracy may not be justified due to the extra computational cost.

Quantum models performance with simulated noisy circuits

To investigate the effect of noise on the training dynamics and generalization performance of quantum neural networks, we evaluated QNN models across a range of simulated noise levels (γ = 0.00 to 0.50) by adding PennyLane's Amplitude Damping noise channel to the end of the circuit. As illustrated in the training and testing loss curves (Fig. 18), the overall convergence behavior demonstrates that the QNN can learn effectively under



(a) Performance Metrics for ansatzes for Dielectron Dataset



(b) Performance Metrics for ansatzes for Multijet Dataset

Fig. 17. QLSTM models performance metrics. The box plots show the median, IQR, minimum, and maximum for the respective metrics.

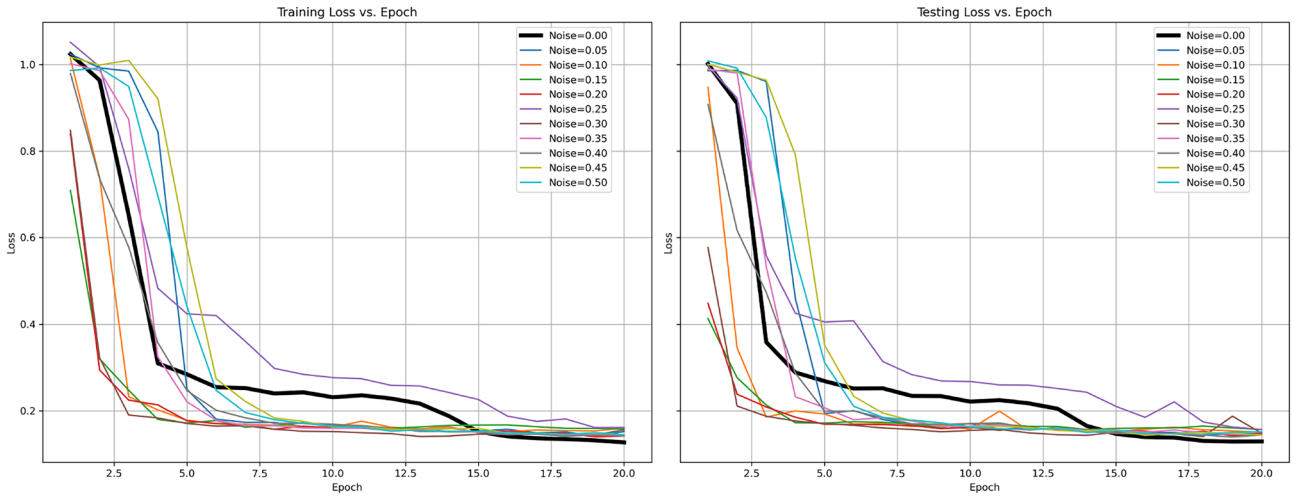


Fig. 18. Training and testing loss of QNN models for various noise levels.

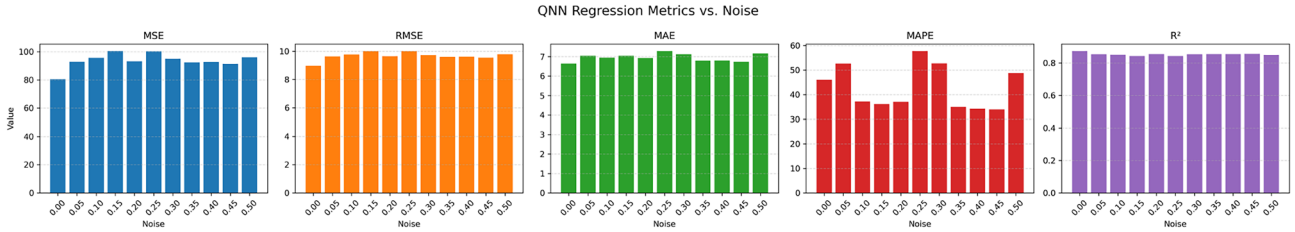


Fig. 19. Performance metrics for QNN models for various noise levels.

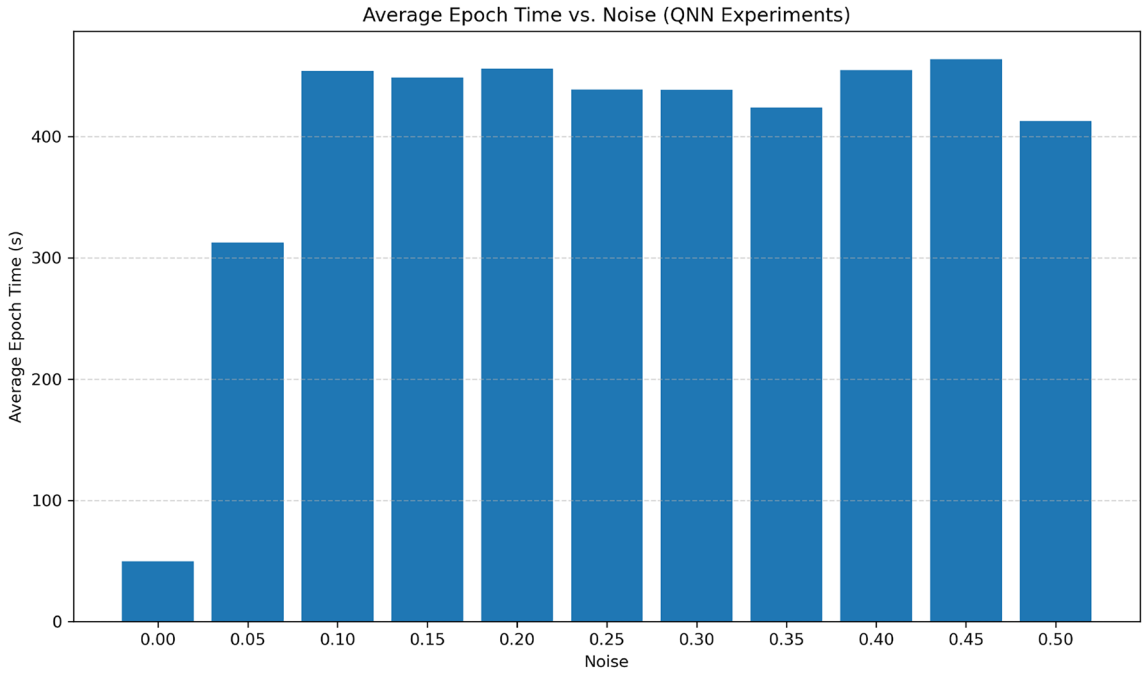


Fig. 20. Average epoch time for QNN models for various noise levels.

moderate noise. Looking at the five standard metrics to evaluate the impact of noise on QNN performance (Fig. 19), we see the adverse effects on absolute accuracy metrics, especially under moderate noise levels. To assess the computational implications of noise in QNNs, we measured the average epoch duration across the range of noise values (Fig. 20). The results indicate a pronounced increase in training time with the introduction of even minimal noise. This increase can be attributed to the higher number of quantum circuit evaluations required to mitigate unreliable results out of noisy circuits. While the training time remains relatively stable across the noise range 0.10–0.50, the abrupt jump from 0.00 to 0.05 underscores a key performance trade-off, that even low levels of noise can substantially increase computational overhead. These findings suggest that the practical deployment of QNNs on NISQ hardware remains computationally challenging for large datasets.

Discussion

In this study, we provided a comprehensive study of Quantum Machine Learning models. We detailed the experimental setup, including data preprocessing, model architecture, and training procedures. Our results underscore the strengths and limitations of the approaches and offer insights into their practical implications for the problem domain. By examining key performance metrics, we gained a nuanced understanding of the potential benefits of quantum machine learning models.

The performance of classical and quantum models varied, with some performing better than others. Among the classical models, Catboost, ANN, and LSTM were the top performers. The complexity of the tasks, interdependencies among the features, and the sequential nature of the collision events are possible reasons for these models to perform well. As this work primarily focused on QML algorithms, the classical models were not studied in detail.

QLSTM's functioning was the best among the QML models, with the nature of the data lending itself very well to the model. Despite being faster to train, the QNN model had a higher range of error occurrences as compared to QLSTM.

Abbreviation	Definition
AI	Artificial intelligence
CPU	Central processing unit
GPU	Graphics processing unit
LSTM	Long short-term memory
QLSTM	Quantum long short-term memory
QNN	Quantum neural network
RAM	Random access memory
A100 GPU	NVIDIA A100 graphics processing unit

Table 7. List of abbreviations used in this paper.

Comparing among the three ansatzes studied, we could find a direct correlation of complexity of the ansatz to the time takes during training. However, we could not find a direct proof that the added complexity of the Ansatz 2 and Ansatz 3 made them outperform Ansatz 1 in a significant manner.

The Quantum models present a new area of research in machine learning. However, the state of the NISQ devices does not allow QML to be a competitor to classical ML yet, and long-term research, development, and optimization of QML algorithms needs to happen establish quantum computing as a practical solution.

In our future work, the authors are planning to enhance this study by studying these models on real quantum hardware. We are currently in discussion with IBM on setting up access to IBM Quantum, and we will be presenting those results in our follow-up research. The authors also plan to investigate and benchmark performance comparisons between emulation and real quantum environments. Further, the authors plan to investigate the scaling of quantum models to see the effects while handling large datasets. Lastly, the authors also plan to do detailed benchmarking of this work against the existing body of work using advanced quantum and classical machine learning in high-energy physics.

Conclusion, strengths, limitations, and future work

This study investigated the feasibility and performance of QML models for regression tasks over large-scale, high-energy physics datasets, such as those generated by CERN. The research focused on multiple aspects critical to QML design, including data encoding strategies, the choice of ansatz, and the computational implications of circuit complexity. Feature selection was conducted based on domain knowledge for this specific dataset; however, more generalized approaches, such as automated feature selection or dimensionality reduction techniques, may be necessary when applying these methods to datasets from other domains.

Our empirical results demonstrate that classical models such as CatBoost, Artificial Neural Networks (ANN), and Long Short-Term Memory (LSTM) networks delivered strong baseline performance. Among the quantum models, the Quantum LSTM (QLSTM) consistently outperformed the Quantum Neural Network (QNN). It is important to note, however, that all quantum models in this study were evaluated in a simulated environment. Future research should prioritize benchmarking these models on real quantum hardware to understand the effects of hardware-specific noise, decoherence, gate fidelity, and readout errors.

These experiments study the potential viability of QML approaches in domains requiring complex regression tasks, especially as quantum computing hardware continues to evolve. As such, this work contributes to the growing body of literature that bridges quantum computing and machine learning, emphasizing the importance of continued algorithmic refinement and hardware experimentation.

In addition, the findings highlight critical considerations for practical QML deployment. Simulated environments, while indispensable for initial validation, cannot fully replicate the operational variability of physical quantum systems. As such, empirical evaluation using real hardware is essential for understanding trade-offs in performance, scalability, and noise resilience, and for guiding the design of standardized QML workflows.

Moreover, ansatz design emerges as a important consideration for model effectiveness. Our observations indicate that increasing circuit complexity does not necessarily translate to improved accuracy or robustness. Therefore, future work should explore hardware-aware and task-adaptive ansätze to see if they offer pathways to more efficient, noise-tolerant architectures. Integrating these methods with environmental modeling and energy profiling will be critical to advancing both the performance and sustainability of QML algorithms.

List of abbreviations

The abbreviations used in this study are summarized in Table 7, providing definitions of commonly referred technical terms.

Data availability

This study used CERN open-source datasets: (1) Dielectron dataset (DOI:10.7483/OPENDATA.CMS.PCSW.AHVG). (2) Multijet dataset (DOI:10.7483/OPENDATA.CMS.GACK.GEJA). The code for these experiments can be found at the following repositories: (1) QNN (https://github.com/SarvaTripathi/QNN_HEP.git) (2) QLS TM (https://github.com/SarvaTripathi/QLSTM_HEP.git).

Received: 17 August 2025; Accepted: 24 November 2025

Published online: 12 December 2025

References

1. Collaboration, A. Dielectron production in central pb—pb collisions at $\sqrt{s_{\mathrm{NN}}}$ = 5.02 tev (2023). arXiv:2308.16704.
2. Larkoski, A. J., Moult, I. & Nachman, B. Jet substructure at the large hadron collider: A review of recent advances in theory and machine learning. *Phys. Rep.* **841**, 1–63. https://doi.org/10.1016/j.physrep.2019.11.001 (2020).
3. Andersen, J. R. & Smillie, J. M. Multiple jets at the LHC with high energy jets. *J. High Energy Phys.* **2011**, 10. https://doi.org/10.1007/JHEP06(2011)010 (2011).
4. Schwartz, M. D. *Quantum Field Theory and the Standard Model* (Cambridge University Press, 2014).
5. Martin, B. R. *Nuclear and Particle Physics An Introduction by Brian Martin*, 2nd edn (Wiley, 2008).
6. Griffiths, D. *Introduction to Elementary Particles* 2nd edn (Wiley-VCH, 2008).
7. Ellis, J. Outstanding questions: physics beyond the Standard Model. *Philos. Trans. R. Soc. A Math. Phys. Eng. Sci.* **370**, 818–830. https://doi.org/10.1098/rsta.2011.0452 (2012).
8. Betancourt, M. A Conceptual Introduction to Hamiltonian Monte Carlo (2018). ArXiv:1701.02434 [stat].
9. Carleo, G. et al. Machine learning and the physical sciences. *Rev. Mod. Phys.* **91**, 045002. https://doi.org/10.1103/RevModPhys.91.045002 (2019).
10. Baldi, P., Sadowski, P. & Whiteson, D. Searching for exotic particles in high-energy physics with deep learning. *Nat. Commun.* **5**, 4308. https://doi.org/10.1038/ncomms5308 (2014).
11. Biamonte, J. et al. Quantum machine learning. *Nature.* **549**, 195–202. https://doi.org/10.1038/nature23474 (2017). ArXiv:1611.09347.
12. Preskill, J. Quantum computing in the NISQ era and beyond. *Quantum.* **2**, https://doi.org/10.22331/q-2018-08-06-79 (2018).
13. AbuGhanem, M. & Eleuch, H. NISQ computers: A path to quantum supremacy. *IEEE Access* **12**, 102941–102961. https://doi.org/10.1109/ACCESS.2024.3432330 (2024).
14. Wiebe, N., Braun, D. & Lloyd, S. Quantum algorithm for data fitting. *Phys. Rev. Lett.* **109**, 050505. https://doi.org/10.1103/PhysRevLett.109.050505 (2012).
15. Harrow, A. W., Hassidim, A. & Lloyd, S. Quantum algorithm for linear systems of equations. *Phys. Rev. Lett.* **103**, 150502. https://doi.org/10.1103/PhysRevLett.103.150502 (2009).
16. McCauley, T. Events with two electrons from 2010. https://doi.org/10.7483/OPENDATA.CMS.PCSW.AHVG (2014).
17. Duarte, J. Example CSV output file for SUSYBSMAnalysis-RazorFilter. https://doi.org/10.7483/OPENDATA.CMS.GACK.GEJA (2015).
18. Aaij, R., Abdelmotteleb, A., Beteta, C. A. & Abudinén, F. Observation of the decay. *Phys. Rev. Lett.* **128**, 191803. https://doi.org/10.1103/PhysRevLett.128.191803 (2022).
19. Nachman, B., Provasoli, D., De Jong, W. A. & Bauer, C. W. Quantum algorithm for high energy physics simulations. *Phys. Rev. Lett.* **126**, 062001. https://doi.org/10.1103/PhysRevLett.126.062001 (2021).
20. Kwak, Y., Yun, W. J., Jung, S. & Kim, J. Quantum neural networks: concepts, applications, and challenges. In *2021 Twelfth International Conference on Ubiquitous and Future Networks (ICUFN)*, 413–416. https://doi.org/10.1109/ICUFN49451.2021.9528698 (IEEE, 2021).
21. Schuld, M., Sinayskiy, I. & Petruccione, F. The quest for a Quantum Neural Network. *Quantum Inf. Process.***13**, 2567–2586, https://doi.org/10.1007/s11128-014-0809-8 (2014).
22. Killoran, N. et al. Continuous-variable quantum neural networks. *Phys. Rev. Res.* **1**, 033063. https://doi.org/10.1103/PhysRevResearch.1.033063 (2019).
23. Niu, X. F. & Ma, W. P. A novel quantum neural network based on multi-level activation function. *Laser Phys. Lett.* **18**. https://doi.org/10.1088/1612-202X/abd23c (2021).
24. Gyongyosi, L. & Imre, S. Training optimization for gate-model quantum neural networks. *Sci. Rep.* **9**, https://doi.org/10.1038/s41598-019-48892-w (2019).
25. Rebentrost, P., Mohseni, M. & Lloyd, S. Quantum support vector machine for big data classification. *arXiv* https://doi.org/10.1103/PhysRevLett.113.130503 (2013). ArXiv: 1307.0471.
26. Schuld, M. & Killoran, N. Quantum machine learning in feature hilbert spaces. *Phys. Rev. Lett.* **122**, https://doi.org/10.1103/PhysRevLett.122.040504 (2019). ArXiv: 1803.07128 Publisher: American Physical Society.
27. Gentinetta, G., Sutter, D., Zoufal, C., Fuller, B. & Woerner, S. Quantum kernel alignment with stochastic gradient descent. In *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, 256–262. https://doi.org/10.1109/QCE57702.2023.00036 (IEEE, 2023).
28. Zhou, N.-R., Liu, X.-X., Chen, Y.-L. & Du, N.-S. Quantum K-nearest-neighbor image classification algorithm based on K–L transform. *Int. J. Theor. Phys.* **60**, 1209–1224. https://doi.org/10.1007/s10773-021-04747-7 (2021).
29. Li, J., Lin, S., Yu, K. & Guo, G. Quantum K-nearest neighbor classification algorithm based on Hamming distance. *Quantum Inf. Process.* **21**, 18. https://doi.org/10.1007/s11128-021-03361-0 (2022).
30. Gong, L., Ding, W., Li, Z., Wang, Y. & Zhou, N. Quantum K-nearest neighbor classification algorithm via a divide and conquer strategy. *Adv. Quantum Technol.* **7**, 2300221. https://doi.org/10.1002/qute.202300221 (2024).
31. Hur, T., Kim, L. & Park, D. K. Quantum convolutional neural network for classical data classification. *Quantum Mach. Intell.* **4**, https://doi.org/10.1007/s42484-021-00061-x (2022).
32. Wei, S., Chen, Y., Zhou, Z. & Long, G. A quantum convolutional neural network on NISQ devices. *AAPPS Bull.* **32**, https://doi.org/10.1007/s43673-021-00030-3 (2022).
33. Gong, L.-H., Pei, J.-J., Zhang, T.-F. & Zhou, N.-R. Quantum convolutional neural network based on variational quantum circuits. *Opt. Commun.* **550**, 129993. https://doi.org/10.1016/j.optcom.2023.129993 (2024).
34. Zoufal, C., Lucchi, A. & Woerner, S. Quantum generative adversarial networks for learning and loading random distributions. *NPJ Quantum Inf.* **5**, 103. https://doi.org/10.1038/s41534-019-0223-2 (2019).
35. Lu, S., Duan, L.-M. & Deng, D.-L. Quantum adversarial machine learning. *Phys. Rev. Res.* **2**, 033212. https://doi.org/10.1103/PhysRevResearch.2.033212 (2020).
36. Tsang, S. L., West, M. T., Erfani, S. M. & Usman, M. Hybrid quantum-classical generative adversarial network for high-resolution image generation. *IEEE Trans. Quantum Eng.* **4**, 1–19. https://doi.org/10.1109/TQE.2023.3319319 (2023).
37. Chen, S. Y.-C., Yoo, S. & Fang, Y.-L. L. Quantum long short-term memory. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 8622–8626. https://doi.org/10.1109/ICASSP43922.2022.9747369 (IEEE, 2022).
38. Sipio, R. D., Huang, J.-H., Chen, S. Y.-C., Mangini, S. & Worring, M. The dawn of quantum natural language processing. *IEEE* (2022).
39. Arpaia, P. et al. Machine learning for beam dynamics studies at the CERN Large Hadron Collider. *Nucl. Instrum. Methods Phys. Res. Sect. A* **985**, 164652. https://doi.org/10.1016/j.nima.2020.164652 (2021).
40. Nam, J., Yong, H., Hwang, J. & Choi, J. Training an artificial neural network for recognizing electron collision patterns. *Phys. Lett. A* **387**, 127005. https://doi.org/10.1016/j.physleta.2020.127005 (2021).
41. Yalçin Kuzu, S. Artificial intelligence based machine learning approach in high energy physics. *Int. J. Innov. Eng. Appl.* **5**, 176–180. https://doi.org/10.46460/ijiea.929292 (2021).

42. Belis, V., Odagiu, P. & Aarrestad, T. K. Machine learning for anomaly detection in particle physics. *Rev. Phys.* **12**, 100091. https://doi.org/10.1016/j.revip.2024.100091 (2024).
43. Liu, Y., Arunachalam, S. & Temme, K. A rigorous and robust quantum speed-up in supervised machine learning. *Nat. Phys.* **17**, 1013–1017. https://doi.org/10.1038/s41567-021-01287-z (2021).
44. Muser, T., Zapusek, E., Belis, V. & Reiter, F. Provable advantages of kernel-based quantum learners and quantum preprocessing based on Grover's algorithm. *Phys. Rev. A* **110**, 032434. https://doi.org/10.1103/PhysRevA.110.032434 (2024).
45. Gyurik, C. & Dunjko, V. Exponential separations between classical and quantum learners. https://doi.org/10.48550/arXiv.2306.16028 (2024).
46. Huang, H.-Y. et al. Power of data in quantum machine learning. *Nat. Commun.* **12**, 2631. https://doi.org/10.1038/s41467-021-22539-9 (2021).
47. Bowles, J., Ahmed, S. & Schuld, M. Better than classical? The subtle art of benchmarking quantum machine learning models. https://doi.org/10.48550/arXiv.2403.07059 (2024).
48. Bowles, J., Wierichs, D. & Park, C.-Y. Backpropagation scaling in parameterised quantum circuits. https://doi.org/10.48550/arXiv.2306.14962 (2024).
49. Wang, Y. & Liu, H. Quantum computing in a statistical context. *Annu. Rev. Stat. Appl.* **9**, 479–504. https://doi.org/10.1146/annurev-statistics-042720-024040 (2022).
50. Leporini, R. & Pastorello, D. An efficient geometric approach to quantum-inspired classifications. *Sci. Rep.* **12**, https://doi.org/10.1038/s41598-022-12392-1 (2022).
51. Clader, B. D., Jacobs, B. C. & Sprouse, C. R. Preconditioned quantum linear system algorithm. *Phys. Rev. Lett.* **110**, 250504. https://doi.org/10.1103/PhysRevLett.110.250504 (2013).
52. Huang, H.-Y., Bharti, K. & Rebentrost, P. Near-term quantum algorithms for linear systems of equations (2019). ArXiv:1909.07344 [quant-ph].
53. Subasi, Y., Somma, R. D. & Orsucci, D. Quantum algorithms for systems of linear equations inspired by adiabatic quantum computing. *Phys. Rev. Lett.* **122**, 060504. https://doi.org/10.1103/PhysRevLett.122.060504 (2019). ArXiv:1805.10549 [quant-ph].
54. Bravo-Prieto, C. et al. Variational quantum linear solver (2020). ArXiv:1909.05820 [quant-ph].
55. Lee, Y., Joo, J. & Lee, S. Hybrid quantum linear equation algorithm and its experimental test on IBM Quantum Experience. *Sci. Rep.* **9**, 4778. https://doi.org/10.1038/s41598-019-41324-9 (2019). ArXiv:1807.10651 [quant-ph].
56. Schuld, M. & Petruccione, F. *Machine Learning with Quantum Computers*, 2 edn. Quantum Science and Technology (Springer, 2021). https://doi.org/10.1007/978-3-030-83098-4.
57. Cerezo, M., Verdon, G., Huang, H.-Y., Cincio, L. & Coles, P. J. Challenges and opportunities in quantum machine learning. *Nat. Comput. Sci.* **2**, 567–576. https://doi.org/10.1038/s43588-022-00311-3 (2022).
58. LaRose, R. & Coyle, B. Robust data encodings for quantum classifiers. *Phys. Rev. A* **102**, 032420. https://doi.org/10.1103/PhysRevA.102.032420 (2020).
59. Pennylane. Pennylane (https://pennylane.ai/).
60. Hochreiter, S. & Urgen Schmidhuber, J. Long short-term memory. *Neural Comput.* **9**, 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735 (1997).
61. Bausch, J. Recurrent quantum neural networks. In *34th Conference on Neural Information Processing Systems* (2020).
62. Clerk, A. Quantum noise and quantum measurement. In *Quantum Machines: Measurement and Control of Engineered Quantum Systems*, 1 edn. (eds. Devoret, M., Huard, B., Schoelkopf, R. & Cugliandolo, L. F.) 61–112. https://doi.org/10.1093/acprof:oso/9780199681181.003.0002 (Oxford University Press, 2014).
63. Barron, S. V. et al. Provable bounds for noise-free expectation values computed from noisy samples. *Nat. Comput. Sci.* **4**, 865–875. https://doi.org/10.1038/s43588-024-00709-1 (2024).
64. Nico-Katz, A., Keenan, N. & Goold, J. Can quantum computers do nothing? *NPJ Quantum Inf.* **10**, 124. https://doi.org/10.1038/s41534-024-00918-6 (2024).
65. Ahmed, T., Kashif, M., Marchisio, A. & Shafique, M. A comparative analysis and noise robustness evaluation in quantum neural networks. *Sci. Rep.* **15**, 33654. https://doi.org/10.1038/s41598-025-17769-6 (2025).

Acknowledgements

This research received no specific grant from any funding agency for the submitted work.

Author contributions

S.T., H.U., and J.S. conceived the experiment, S.T. conducted the experiment, and S.T. and J.S. analyzed the results. All authors reviewed the manuscript.

Declarations

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to S.T.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.