Toward Practical Quantum Machine Learning: A Novel Hybrid Quantum LSTM for Fraud Detection

Rushikesh Ubale
Qkrishi Quantum
India
rushikesh.ubale@qkrishi.com

Sujan K. K.
Theoretical Sciences Unit
Jawaharlal Nehru Centre For
Advanced Scientific Research, India
sujanbharadwaj97@gmail.com

Sangram Deshpande
Dept. of ECE
NC State University
Raleigh, NC, USA
ssdesh24@ncsu.edu

Gregory T. Byrd
Dept. of ECE
NC State University
Raleigh, NC, USA
gbyrd@ncsu.edu

Abstract—We present a novel hybrid quantum–classical neural network architecture for fraud detection that integrates a classical Long Short-Term Memory (LSTM) network with a variational quantum circuit. By leveraging quantum phenomena such as superposition and entanglement, our model enhances the feature representation of sequential transaction data, capturing complex non-linear patterns that are challenging for purely classical models. A comprehensive data preprocessing pipeline is employed to clean, encode, balance, and normalize a credit card fraud dataset, ensuring a fair comparison with baseline models. Notably, our hybrid approach achieves per-epoch training times in the range of 45–65 seconds, which is significantly faster than similar architectures reported in the literature, where training typically requires several minutes per epoch. Both classical and quantum gradients are jointly optimized via a unified back-propagation procedure employing the parameter-shift rule for the quantum parameters. Experimental evaluations demonstrate competitive improvements in accuracy, precision, recall, and F1 score relative to a conventional LSTM baseline. These results underscore the promise of hybrid quantum-classical techniques in advancing the efficiency and performance of fraud detection systems.

Index Terms—Hybrid Quantum-Classical Neural Networks, Fraud Detection, Long Short-Term Memory (LSTM), Variational Quantum Circuit, Parameter-Shift Rule, Financial Risk Analysis

I. INTRODUCTION

Access to the internet is growing globally, resulting in an increased number of online transactions. Along with the rise in total online transactions, online frauds are also increasing worldwide (provide supporting statistics). There are many categories of fraud, such as credit card fraud, online payment fraud, phishing fraud, and e-commerce fraud. Frauds can also be investment-related, healthcare-related, business-related, job-offer-related, and more. Machine Learning (ML) and Artificial Intelligence (AI) approaches have been employed extensively to tackle these issues. ML, in particular, has found significant applications in addressing fraud in sectors such as healthcare, e-commerce, insurance, credit card fraud, email-related fraud, and beyond. In this article, we focus on developing a model to address credit card fraud using Machine Learning techniques in conjunction with Quantum Computing.

Numerous models have been developed for credit card fraud detection (FD). Many of these are purely classical ML models, such as XGBoost, Random Forest (RF), Support Vector Machines (SVM), and deep learning models. Additionally, hybrid quantum-classical models, such as Quantum Support Vector Machines (QSVM), Quantum Convolutional Neural Networks (Quantum CNN), and Variational Quantum Classifiers (VQCs), have been explored. The integration of quantum principles into classical models offers unique advantages, such as leveraging entanglement, superposition, and access to complex spaces not achievable by classical models. Quantum models also demonstrate high expressivity, quantum parallelism, and the ability to represent data in higher-dimensional spaces via quantum feature maps, making them particularly effective. However, quantum computing faces challenges, including the interpretability of complex representations and the barren plateau problem. In this article, we leverage quantum principles in combination with classical LSTM models to address credit card fraud detection.

When addressing online fraudulent transactions, it is crucial to minimize false positives (FP), which lead to customer dissatisfaction, and false negatives (FN), which erode customers' trust in the banking system. Therefore, any ML model developed for this purpose must focus on reducing FP and FN (or related derived metrics). An effective model should identify important behavioral patterns and repeated sequences in the data to improve performance. Although there is no strict rule for feature selection, handling imbalanced datasets remains a significant challenge. The ratio of fraudulent to normal transactions is often very low, making it difficult to train accurate classification models. Techniques like SMOTE (Synthetic Minority Oversampling Technique) are commonly used to address this issue. Additionally, model robustness should be evaluated by testing on different datasets.

II. LITERATURE REVIEW

In recent years, fraud detection has attracted significant attention in both industry and academia due to its critical impact on financial systems. Researchers have explored a variety of approaches ranging from classical deep learning methods to emerging hybrid quantum-classical models. This section reviews the most relevant works, highlighting the motivation

for our proposed model and emphasizing its architectural novelty.

A. Classical Models

Classical deep learning methods, particularly recurrent neural networks (RNNs) and their variants, have been widely applied to fraud detection. For example, Heryadi *et al.* [1] compared Convolutional Neural Networks (CNNs), Stacked LSTMs, and CNN-LSTM hybrid models under varying normal-to-fraud ratios. Their study demonstrated that while Stacked LSTMs achieved higher training accuracy, CNNs provided better discrimination as measured by the area under the ROC curve (AUC), highlighting the trade-off between accuracy and discrimination capability.

Similarly, Guo *et al.* [2] proposed the Historical Attention-based and Interactive LSTM (HAInt-LSTM), which introduces modifications to the LSTM forget gate to account for variable time intervals. By incorporating a self-historical attention mechanism, HAInt-LSTM better captures long-term behavioral patterns, thereby distinguishing normal from fraudulent activities more effectively than plain LSTM, GRU, phased-LSTM, or Update Gate RNN (UGRNN).

Other works, such as Cheng *et al.* [3], have integrated spatial and temporal features using models like the Spatial-Temporal Attention-based Neural Network (STAN). STAN combines 3D convolutions with spatio-temporal attention mechanisms to consider geographical and transactional timing aspects, outperforming classical models like Logistic Regression, Gradient Boosted Trees, and Deep Neural Networks.

Additionally, Alghofaili *et al.* [4] enhanced LSTM architectures by optimizing training time and accuracy. However, their reported dataset exhibited significant class imbalance, which limits the generalizability of their model's performance.

Recent enhancements have also involved integrating attention mechanisms with LSTMs. For instance, Benchaji *et al.* [5] introduced an attention layer in conjunction with LSTM, leading to improvements in precision, recall, and overall accuracy compared to traditional models such as k-Nearest Neighbors (KNN), Artificial Neural Networks (ANN), and Support Vector Machines (SVM).

While these classical models have demonstrated success, they are inherently limited by the computational constraints of classical processing and their inability to efficiently handle high-dimensional feature spaces. This motivates the exploration of hybrid quantum-classical approaches to **overcome these limitations**.

B. Hybrid Quantum-Classical Models

Hybrid models that combine classical and quantum computing paradigms have recently emerged as promising alternatives, offering the potential to enhance the representational power of classical models through quantum encoding. Grossi *et al.* [6] employed a hybrid quantum-classical model based on a Quantum Support Vector Machine (QSVM) and demonstrated improved performance by exploiting quantum-enhanced feature spaces. Their work illustrated that quantum kernels could provide superior classification accuracy compared to classical SVMs in high-dimensional spaces.

Similarly, Kyriienko *et al.* [7] introduced an unsupervised quantum machine learning model that utilized quantum kernels to map classical features into an enlarged latent space, achieving higher expressivity compared to classical Radial Basis Function (RBF) kernels.

In the domain of fraud detection, Innan *et al.* [8] applied Quantum Graph Neural Networks (Quantum GNN) to financial fraud detection, showing that quantum approaches can outperform classical counterparts in both accuracy and efficiency. Furthermore, Wang *et al.* [9] enhanced QSVMs with quantum annealing solvers, yielding improvements in both speed and predictive performance.

Notably, Innan *et al.* [10] further pushed the boundaries by integrating federated learning with quantum neural networks, addressing data privacy concerns while achieving a remarkable accuracy of 0.95. This highlights the potential of hybrid models to balance performance with privacy preservation.

C. Performance Comparison: Quantum vs Classical Models

A particularly relevant study by Saad Zafar Khan *et al.* [11] compared classical and quantum LSTM models for solar power forecasting. Their findings indicated that the quantum model achieved better performance metrics. However, the authors reported an average training time per epoch of approximately 1 hour and 26 minutes.

In our work, we achieve a significantly reduced epoch time of 45 to 65 seconds on a dataset with 10,000 data points while using PennyLane's `default.qubit` simulator, which runs on the CPU without GPU acceleration. This is noteworthy because the `default.qubit` simulator is a general-purpose state vector simulator, which is not specifically optimized for speed. In contrast, simulators such as `lightning.qubit` and `lightning.gpu` from PennyLane leverage advanced linear algebra libraries and GPU acceleration, enabling faster execution.

Despite using the less optimized `default.qubit` simulator, our model achieves a considerably faster training time, indicating its efficiency in terms of computational overhead. This demonstrates the practical advantage of our hybrid quantum-classical architecture, as it offers both competitive accuracy and faster execution.

D. Architectural Novelty and Design Choices

Our proposed model introduces several architectural innovations that distinguish it from prior works, addressing their limitations in terms of expressivity and computational efficiency:

- **Hybrid Architecture with Variational Quantum Circuits:** Unlike previous models that rely on either classical deep learning or shallow quantum models, we integrate a **variational quantum circuit (VQC)** with a classical LSTM. This hybrid design combines the temporal sequence modeling capabilities of LSTMs with

the enhanced feature representation of quantum circuits, improving both accuracy and efficiency.

- **Efficient Quantum Encoding:** We use **AngleEmbedding** to efficiently map classical features into the quantum Hilbert space. This encoding technique allows the model to represent complex patterns using quantum rotations, increasing the **expressivity** of the model while maintaining low overhead.

- **Strongly Entangling Layers (SEL):** To enhance the feature representation power, we incorporate **Strongly Entangling Layers** in the VQC. These layers introduce controlled quantum entanglement between qubits, enabling the model to capture complex, non-linear dependencies in the data. Prior works have demonstrated that entanglement improves the expressive power of quantum models, making them more effective for high-dimensional problems [12].

- **Efficient Training with CPU-Based Simulation:** Despite using the `default.qubit` simulator (CPU-based) instead of the more optimized `lightning.qubit` or `lightning.gpu` simulators, our model achieves **training times of 45–65 seconds per epoch** on 10,000 data points. This demonstrates that our architecture achieves high efficiency even without hardware acceleration, making it practical for real-world deployment.

E. Motivation and Scope

While prior studies have shown the effectiveness of hybrid quantum-classical models in improving accuracy, they often suffer from **computational inefficiencies** and **prolonged training times**. In contrast, our work addresses this limitation by demonstrating that a **carefully designed hybrid model** can achieve both **high accuracy** and **reduced training time**.

Furthermore, our implementation uses the `default.qubit` simulator, which, despite being CPU-based, achieves efficient training times. This highlights the feasibility of deploying hybrid quantum-classical models on standard hardware without the immediate need for GPU acceleration, making our approach more accessible and practical for real-world applications. However, future studies could introduce GPU acceleration to further improve training speed and scale the model to larger datasets. Additionally, by training on a higher number of data samples, we can effectively mitigate overfitting, enhancing the model's generalizability and reliability for real-world deployment.

III. DATA PREPROCESSING AND FEATURE ENGINEERING

Our dataset consists of credit card transaction records that include various attributes such as transaction timestamps, geospatial coordinates, and customer demographics. To prepare the data for robust fraud detection, we performed extensive preprocessing and feature engineering. These steps transform raw data into informative features while addressing issues such as class imbalance.

A. Temporal and Geospatial Feature Extraction

First, the raw transaction timestamp (`trans_date_trans_time`) was converted into a standardized `datetime` format. From this, we extracted several time-based features (hour, day, weekday, month, and year) to capture temporal patterns that may be indicative of fraudulent behavior. In addition, we derived a key geospatial feature: the distance between customer and merchant locations. To calculate this, we used the well-known haversine formula, which computes the great-circle distance between two points on a sphere. This is especially useful for detecting anomalies when transactions occur over unexpectedly large distances. The formula is given by:

$$d = 2R \cdot \arctan 2\left(\sqrt{a}, \sqrt{1-a}\right),$$

with

$$a = \sin^2\left(\frac{\Delta\phi}{2}\right) + \cos(\phi_1)\cos(\phi_2)\sin^2\left(\frac{\Delta\lambda}{2}\right).$$

Here, R is the Earth's radius (6371 km), ϕ_1 and ϕ_2 are the latitudes (in radians) of the customer and merchant respectively, and $\Delta\lambda$ is the difference in their longitudes (in radians). Including this feature is necessary to quantify the spatial relationship between transaction parties, which can be critical for detecting geographically inconsistent or suspicious transactions.

B. Categorical Encoding and Demographic Feature Engineering

Categorical variables such as `category`, `gender`, `state`, and `job` were transformed into numerical values using label encoding. This conversion is essential for incorporating non-numeric information into machine learning algorithms. Additionally, we computed the customer age from the date of birth, which serves as a valuable demographic indicator for assessing fraud risk.

C. Balancing the Dataset

Fraud detection datasets often suffer from severe class imbalance, where non-fraudulent transactions greatly outnumber fraudulent ones. To mitigate this issue, we applied resampling techniques: oversampling the minority (fraud) class and downsampling the majority (non-fraud) class to achieve a balanced dataset of 50,000 samples per class. This balance is crucial for preventing the predictive model from being biased toward the majority class and for ensuring that both classes are adequately represented during model training.

Overall, these preprocessing and feature engineering steps were necessary to convert heterogeneous raw data into a structured, refined dataset. By capturing temporal, spatial, and categorical nuances, we enhance the interpretability and performance of our fraud detection models.

IV. METHODOLOGY

In this work, we propose a hybrid quantum-classical neural network architecture for fraud detection, integrating a Long Short-Term Memory (LSTM) network for temporal feature extraction with a quantum layer for enhanced representation learning. The methodology consists of several stages: data preprocessing, classical feature extraction, quantum embedding, and final classification. First, raw transaction data undergoes preprocessing, including normalization and encoding, to ensure compatibility with the neural network input. The LSTM processes sequential features, capturing temporal dependencies, and its output is mapped to a lower-dimensional space using a fully connected (FC) layer. This reduced feature set is then passed to a variational quantum circuit (VQC), where quantum embeddings and entangling layers transform the data. The quantum circuit's measurement outcomes are subsequently processed by an additional FC layer, producing the final classification output. The entire pipeline is trained using backpropagation, with classical gradients propagating through the quantum circuit using the parameter-shift rule. The following sections detail each stage, including data preparation, model architecture, training setup, and evaluation metrics.

A. Data Acquisition and Preprocessing

The dataset used in this study is a synthetically generated credit card fraud dataset designed to mimic real-world transaction data. It contains 27 features, categorized into different types, as shown in Table I. These features include card details, merchant information, transaction amount, personal identifiers, geographical data, transaction metadata, and additional encoded categorical variables.

As described in Section 2, the raw data was initially loaded, and basic cleaning was performed. In this section, we detail the subsequent preprocessing steps applied prior to model training.

1) Data Acquisition and Preprocessing Pipeline: The data acquisition and preprocessing pipeline consists of the following steps to prepare the dataset for model training and evaluation:

1) **Data Cleaning:** To reduce irrelevant information, personally identifiable columns such as `first`, `last`, `street`, and `trans_num` are removed. This step eliminates unnecessary attributes that do not contribute to fraud detection while complying with data protection regulations.

2) **Categorical Encoding:** Since machine learning models require numerical inputs, categorical features such as `merchant` and `city` are transformed using *label encoding*. Each unique category is assigned a numerical value, allowing models to process these features effectively.

3) **Dataset Subsetting:** Due to computational constraints, we select a balanced subset of 10,000 transactions, consisting of 5,000 fraudulent and 5,000 non-fraudulent transactions. This ensures that our dataset remains manageable while maintaining class balance, preventing model bias.

4) **Feature Normalization:** To standardize continuous numerical features, we apply *z-score normalization*. Each feature x is normalized as follows:

$$x_{\text{norm}} = \frac{x - \mu}{\sigma},$$

where μ and σ represent the mean and standard deviation of the feature, respectively. This transformation ensures that all numerical attributes are on a comparable scale, improving model convergence and performance.

5) **Data Partitioning:** The preprocessed dataset is split into three subsets: training (70%), validation (15%), and test (15%). *Stratified sampling* is used to maintain an equal proportion of fraudulent and non-fraudulent transactions in each subset, ensuring that the model learns effectively from both classes.

6) **Tensor Conversion:** To enable efficient computation in deep learning models, the processed data is converted from NumPy arrays to PyTorch tensors. This step facilitates seamless integration with PyTorch-based architectures, optimizing performance during training and inference.

B. Model Architecture

Our hybrid quantum–classical neural network is composed of three major components: (1) a classical Long Short-Term Memory (LSTM) module for extracting temporal features, (2) a dimensionality-reduction layer mapping the LSTM state into a dimension suitable for quantum processing, and (3) a variational quantum circuit (VQC) followed by a final fully connected (FC) layer for classification. Figure **??** illustrates the quantum portion of the architecture, highlighting how the classical data is embedded, transformed through strongly entangling layers, and measured.

1) Classical LSTM Component: The LSTM [13] processes sequential transaction data to capture long-term dependencies, which is crucial for fraud detection. Let x_t be the input at time t, and let h_t and C_t denote the hidden and cell states, respectively. The LSTM computes:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i), \tag{1}$$
$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f), \tag{2}$$
$$\tilde{C}_t = \tanh(W_C x_t + U_C h_{t-1} + b_C), \tag{3}$$
$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t, \tag{4}$$
$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o), \tag{5}$$
$$h_t = o_t \odot \tanh(C_t), \tag{6}$$

where $\sigma(\cdot)$ is the sigmoid activation function, $\tanh(\cdot)$ is the hyperbolic tangent, W and U are weight matrices, b is the bias term, and $\odot$ denotes element-wise multiplication. Equations (1)–(6) describe how the LSTM selectively remembers or forgets information at each time step [14], [15].

2) Handling of Time and Batch Dimensions: In our implementation, the LSTM is configured with `batch_first=True` and expects input tensors of shape

$$(\text{batch\_size}, \text{sequence\_length}, \text{input\_size}).$$

TABLE I: Feature Description of the Credit Card Fraud Dataset

Feature Category	Feature Name	Description
Card Details	`cc_num`	Credit card number (hashed)
	`merchant`	Merchant information for transaction
Transaction Amount	`amt`	Transaction amount in USD
Personal Identifiers	`first, last`	First and last name of cardholder
	`street`	Street address of cardholder
	`gender_encoded`	Encoded gender information
	`customer_age`	Estimated age of the customer
Geographical Data	`city, zip`	City and ZIP code of the transaction
	`lat, long`	Latitude and longitude coordinates
	`city_pop`	Population of the city
	`state_encoded`	Encoded state information
Transaction Metadata	`unix_time`	Timestamp of the transaction
	`transaction_hour`	Hour of the transaction
	`transaction_day`	Day of the transaction
	`transaction_weekday`	Day of the week
	`transaction_month, transaction_year`	Month and year of transaction
Additional Features	`category_encoded`	Encoded transaction category
	`job_encoded`	Encoded job category of the cardholder
	`customer-merchant distance`	Calculated distance between customer and merchant

Since each transaction is represented by a static feature vector (i.e., with shape (batch_size, input_size)), we insert a dummy time dimension using `unsqueeze(1)`. This reshapes the input to

$$(\text{batch\_size}, 1, \text{input\_size}),$$

effectively treating each sample as a sequence of length 1. The LSTM processes this input and outputs the hidden state corresponding to this single time step, which is then used for subsequent dimensionality reduction and quantum embedding.
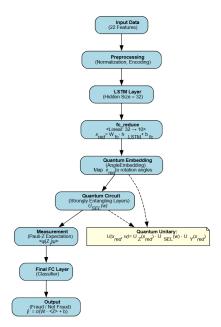

Fig. 1: Model Architecture

3) Dimensionality Reduction via FC Layer: After processing all time steps, the final hidden state h_T is extracted. Often, h_T has a higher dimensionality than the number of qubits in our quantum circuit. We therefore apply a fully connected (FC) layer that projects h_T onto $\mathbb{R}^{n_{\text{qubits}}}$. This ensures that

the classical feature vector is appropriately sized for the subsequent quantum embedding stage.

4) Quantum Layer: The primary innovation lies in the quantum layer, implemented as a variational quantum circuit (VQC). This layer exploits quantum effects—such as superposition and entanglement—to enhance feature representation [16], [17].

a) Angle Embedding and Strongly Entangling Layers.: Figure **??** depicts 10 qubit wires labeled from 0 to 9. Initially, the classical feature vector $\vec{x} = [x_1, x_2, \ldots, x_{n_{\text{qubits}}}]$ is encoded onto these qubits via *AngleEmbedding*:

$$|\psi(\vec{x})\rangle = \bigotimes_{i=1}^{n_{\text{qubits}}} R_Y(x_i) |0\rangle, \qquad (7)$$

where $R_Y(x_i)$ is a rotation about the Y axis by angle x_i, mapping the classical data into the quantum Hilbert space [18], [17]. Next, *StronglyEntanglingLayers* apply repeated blocks of entangling gates (e.g., CNOT) interspersed with single-qubit parameterized rotations:

$$U(\theta) = \prod_{\ell=1}^{L} U_{\text{entangle}}^{(\ell)}(\theta^{(\ell)}), \qquad (8)$$

where $\theta = \{\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(L)}\}$ are trainable parameters. These layers introduce correlations among qubits, enabling the circuit to learn highly non-linear mappings of the input features [19], [20].

b) Measurement.: After the final entangling layer, the circuit measures each qubit in the Pauli-Z basis:

$$q_i = \langle \psi(\vec{x}, \theta) | Z_i | \psi(\vec{x}, \theta) \rangle, \quad i = 1, \ldots, n_{\text{qubits}}. \qquad (9)$$

This yields a real-valued vector $\vec{q} = [q_1, q_2, \ldots, q_{n_{\text{qubits}}}]$ that captures quantum-enhanced features of the data. By leveraging entanglement and superposition, the circuit can represent transformations that may be difficult for purely classical architectures [21].
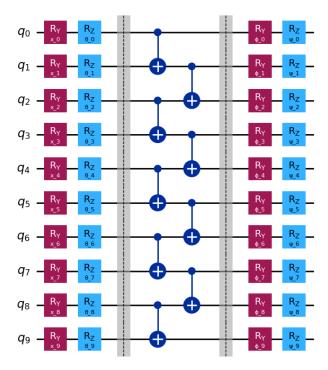
Fig. 2: Decomposed quantum circuit used in our QLSTM-based fraud detection model. Each qubit is initialized in the state $|0\rangle$ and undergoes a data-encoding R_y rotation with parameter x_i, followed by a trainable $R_z(\theta_i)$. The circuit includes barriers to demarcate the encoding, entanglement, and final transformation stages for clarity. The entanglement layer is implemented using staggered CNOT gates between adjacent qubits to ensure local entanglement. This is followed by a second set of trainable rotations $R_y(\phi_i)$ and $R_z(\psi_i)$. All qubits are measured in the computational (Z) basis. This diagram is a simplified decomposition of the PennyLane `StronglyEntanglingLayers` template used in the model.

5) Final Classification Layer: The quantum output $\vec{q}$ is passed into a final fully connected layer:

$$\hat{y} = \sigma\Big(W_{\text{out}}\,\vec{q} + b_{\text{out}}\Big), \tag{10}$$

where $\sigma(\cdot)$ is the sigmoid function, W_{out} is the weight matrix, and b_{out} is the bias term. The resulting scalar $\hat{y}$ represents the predicted probability of fraud, with values closer to 1 indicating a higher likelihood of fraudulent activity.

6) Note on Parameter Optimization:

a) Integration of Classical and Quantum Gradients.: Both the classical (LSTM and FC layers) and quantum circuit parameters are trained end-to-end in a unified backpropagation loop [14], [20]. Concretely, the partial derivatives of the

---

loss function with respect to the classical parameters (e.g., W_i, U_i, b_i in the LSTM) are computed via standard automatic differentiation. However, gradients for the quantum parameters θ require the *parameter-shift rule*:

1) **Forward Pass:** The model processes the input sequence through the LSTM and FC layer, then encodes the data into the quantum circuit. After quantum operations, a measurement vector $\vec{q}$ is produced, which is fed into the final FC layer to obtain the loss $\mathcal{L}$.

2) **Quantum Parameter-Shift:** For each parameter θ_j, the circuit is evaluated twice: once at $\theta_j + \Delta$ and once at $\theta_j - \Delta$, where Δ is a small shift (often $\Delta = \pi/2$). The partial derivative is approximated by:

$$\frac{\partial \mathcal{L}}{\partial \theta_j} \approx \frac{\mathcal{L}(\theta_j + \Delta) - \mathcal{L}(\theta_j - \Delta)}{2 \sin(\Delta)}.$$

3) **Combine Gradients:** The gradient contributions from the quantum circuit parameters are then merged with those from the classical LSTM and FC layers. Because all components are part of the same computational graph, the chain rule naturally propagates errors back through both classical and quantum modules.

4) **Optimizer Step:** A single optimizer (e.g., Adam) applies the combined gradient updates to all parameters in one pass, ensuring consistent learning dynamics across the entire model.

This unified approach allows classical layers to learn temporal features while the quantum circuit simultaneously learns an expressive mapping in Hilbert space, resulting in a more powerful end-to-end system [19], [21].

b) Practical Implementation.: We implement this architecture using the PennyLane library [20] for quantum operations and PyTorch for classical layers. PennyLane seamlessly integrates with PyTorch, allowing us to compute classical gradients using automatic differentiation and quantum gradients via the parameter-shift rule in a single backpropagation loop.

Overall Pipeline:

1) **Sequential Modeling:** The LSTM extracts temporal features from transaction sequences.

2) **Dimension Matching:** An FC layer reduces h_T to size n_{qubits}.

3) **Quantum Circuit:** The quantum layer embeds and entangles these features, outputting a measurement vector $\vec{q}$.

4) **Classification:** The final FC layer outputs a probability of fraud, with all parameters updated end-to-end in one training loop.

This design exploits both classical time-series modeling and quantum feature representation to enhance fraud detection performance [16], [18], [17], [21].

V. TRAINING AND EVALUATION

After defining our hybrid quantum-classical model architecture, we proceed with the training procedure, hyperparameter

configuration, and evaluation protocol. The following subsections describe how the model is optimized, validated, and tested, as well as the metrics used to assess performance.

Flow Diagram of the Training Process: Figure 3 summarizes the training pipeline, outlining the sequence of steps from data loading to optimizer updates.
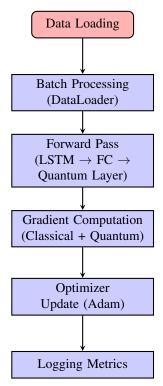

Fig. 3: Flow diagram summarizing the training process: data loading, forward pass through the hybrid model, gradient evaluation (integrating classical gradients and quantum gradients via the parameter-shift rule as detailed in Methodology Section), and optimizer update followed by metric logging.

A. Training Procedure

In this section, we describe the complete training process for our hybrid quantum–classical model. We detail the configuration of data loaders and batch processing, the choice of loss function and optimizer, the integrated gradient computation for both classical and quantum parameters, and our logging strategy over the training epochs.

1) Data Loaders and Batch Processing: To ensure efficient and stable training, we employ PyTorch's `DataLoader` for batch processing of the dataset. The data is split into training, validation, and test sets with stratified sampling to maintain class balance. For training, we set the batch size to 32 and enable shuffling:

- **Batch Size:** A batch size of 32 is chosen to balance between memory efficiency and convergence speed. Smaller batches can introduce noisy gradient estimates, while larger batches may require more memory and lead to slower updates.

- **Shuffling:** Shuffling the data at each epoch ensures that the model does not learn spurious order-related patterns and improves the stability of the training process.

These steps help mitigate overfitting and improve the overall convergence of the model.

2) Loss Function and Optimizer: For binary classification tasks such as fraud detection, we use the `BCEWithLogitsLoss` function provided by PyTorch. This loss function combines the sigmoid activation and the binary cross entropy loss into one stable computation:

$$\mathcal{L}(y, \hat{y}) = -\frac{1}{N} \sum_{i=1}^{N} \Big[y_i \log\big(\sigma(\hat{y}_i)\big) + (1 - y_i) \log\big(1 - \sigma(\hat{y}_i)\big) \Big],$$

where $\sigma(\cdot)$ denotes the sigmoid function and N is the number of samples in the batch [14].

We use the Adam optimizer for its adaptive learning rate properties, which is especially useful when training deep networks with both classical and quantum components. The hyperparameters are set as follows:

- **Learning Rate:** $\eta = 0.001$. This rate offers a good balance between convergence speed and the risk of overshooting minima.

- **Weight Decay:** 1×10^{-4} to regularize the model and prevent overfitting.

The hyperparameters for our model were chosen through a combination of insights from prior literature and empirical tuning on a validation set. A batch size of 32 was selected as it provided a good trade-off between gradient noise reduction and computational efficiency while fitting within available hardware memory. Similarly, the learning rate of 0.001 was chosen based on recommendations from previous studies [22] and further refined via grid search to ensure stable convergence. A weight decay of 1×10^{-4} was incorporated to mitigate overfitting, in line with standard regularization practices in deep learning [14]. These choices have been validated across multiple runs and are reported here to facilitate reproducibility of our approach.

3) Gradient Computation (Classical + Quantum): Our model integrates both classical layers (LSTM, FC layers) and a quantum layer. For classical layers, gradients are computed automatically via PyTorch's autograd engine. However, gradients for the quantum circuit parameters require a different approach known as the *parameter-shift rule* [19], [20]:

1) **Forward Pass:** The model processes the input data through the LSTM and FC layers, encodes it into the quantum circuit, and obtains the measurement vector $\vec{q}$. This is then passed to the final FC layer to compute the loss $\mathcal{L}$.

2) **Quantum Gradient Evaluation:** Our training process computes gradients for both the classical layers (LSTM and FC layers) and the quantum circuit in a unified backpropagation loop. For the classical components, PyTorch's automatic differentiation efficiently computes the gradients. In contrast, the quantum circuit parameters are updated using the parameter-shift rule, as described

in Methodology Section. Briefly, this rule estimates the gradient for each quantum parameter by perturbing it by a small shift (typically $\Delta = \pi/2$) and measuring the resulting change in the loss. By integrating these quantum gradients with the classical ones via the chain rule, we obtain a complete gradient vector for all parameters. Additionally, we employ standard techniques such as gradient clipping (to prevent exploding gradients in the LSTM) and learning rate scheduling to enhance training stability.

3) **Gradient Integration:** The gradients from the quantum circuit are then merged with those from the classical layers using the chain rule. This unified gradient is used to update all parameters simultaneously in a single optimizer step.

Additionally, we implement gradient clipping for the LSTM layers to avoid exploding gradients, and we may use learning rate scheduling to adjust the learning rate dynamically during training.

4) Epochs and Logging: We train our model for a total of 80 epochs. At each epoch, we track the following metrics:

- **Training Loss:** Computed over all batches in the training set.
- **Validation Loss and Accuracy:** Evaluated on the validation set at the end of each epoch to monitor overfitting and convergence.
- **Epoch Time:** The duration of each epoch is recorded to assess training efficiency.

We log these metrics and print summary statistics after each epoch to monitor training progress. Early-stopping criteria are not enforced in the present experiments, but checkpoint saving is implemented to allow recovery of the best-performing model based on validation loss. This logging strategy ensures that we can analyze the training dynamics and adjust hyperparameters if needed.

VI. BENCHMARKING MODELS

To assess the effectiveness of the quantum-enhanced model, we benchmark its performance against a well-established classical deep learning model: the **Classical LSTM**. This ensures that improvements (if any) are attributable to quantum advantages rather than architectural disparities.

A. Classical LSTM Model

Long Short-Term Memory (LSTM) networks [13] are a type of recurrent neural network (RNN) designed to capture long-range dependencies in sequential data. The classical LSTM model in this benchmark closely mirrors the quantum model's data processing pipeline, ensuring a fair comparison.

1) Data Preprocessing and Setup: The dataset used for training both the Classical LSTM and Quantum LSTM models originates from a balanced credit card fraud detection dataset. The preprocessing steps are identical in both cases:

- Encoding categorical variables using `LabelEncoder`.
- Standardizing numerical features with `StandardScaler`.

- Stratified train-validation-test split to maintain class balance.

The processed data is then wrapped into PyTorch `Dataset` and `DataLoader` objects for efficient mini-batch training.

2) Classical LSTM Architecture: The classical LSTM model follows a two-layer LSTM configuration with a dropout layer for regularization. The architecture consists of:

- An **LSTM** module with `hidden_size` of 32 and 2 layers.
- A fully connected (`fc`) layer mapping the final LSTM output to a single scalar prediction.
- A dropout rate of 0.3 to prevent overfitting.

The model is trained using the **Binary Cross-Entropy Loss with Logits** (`BCEWithLogitsLoss`) since fraud detection is a binary classification problem. The optimizer chosen is **Adam** with a learning rate of 0.001 and weight decay of 10^{-4}.

3) Training and Evaluation: The training follows an iterative mini-batch gradient descent approach:

- Each batch is forward-passed through the LSTM, and loss is computed.
- Gradients are backpropagated, and weights are updated via Adam optimization.
- Training is conducted for **80 epochs**.

Validation is performed at each epoch to monitor generalization. The final test evaluation computes:

- **Accuracy**
- **Precision, Recall, and F1-score**
- **Confusion Matrix**
- **Inference Time**

B. Comparison with Quantum LSTM

Both the Classical and Quantum LSTM models share the following key characteristics:

- **Data Processing:** Identical feature engineering and normalization steps.
- **Batch-based Training:** Both models utilize PyTorch's DataLoader for efficient gradient computation.
- **LSTM Core:** The Classical LSTM uses standard recurrent gates, whereas the Quantum LSTM replaces some transformations with variational quantum circuits.
- **Loss Function and Optimization:** Both use `BCEWithLogitsLoss` and Adam optimizer.

However, the primary distinction lies in the representation of sequential dependencies:

- **Classical LSTM:** Uses standard matrix multiplications and nonlinear activations.
- **Quantum LSTM:** Incorporates quantum-parameterized transformations to enhance expressivity.
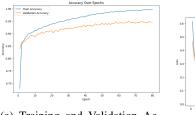
This direct comparison ensures that any observed improvements in the Quantum LSTM are due to quantum advantages rather than differences in data handling, model structure, or training strategy.
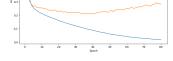
VII. RESULTS

In this section, we present the experimental outcomes of our hybrid quantum-classical model for fraud detection. We begin by analyzing the training and validation dynamics, then evaluate the final performance on the test set. We also discuss the computational efficiency of our approach by examining the per-epoch training time.

A. Quantum Hybrid Model Results

1) Training and Validation:

2) Loss and Accuracy Curves: Figure 4a illustrates the *training and validation accuracy* throughout the same training period. The training accuracy rises quickly from about 70% to over 95%, eventually approaching 99% by epoch 80. The validation accuracy follows a similar pattern, reaching around 95%. This convergence trend indicates that the model successfully captures the temporal and feature-level correlations in the fraud dataset. Figure 4b shows the *training and validation loss* over 80 epochs. Initially, the training loss starts around 0.6, dropping significantly during the first 10 epochs and then steadily declining until the end of training. The validation loss also decreases, indicating that the model is learning effectively without severe overfitting.

(a) Training and Validation Accuracy

(b) Training and Validation Loss

Fig. 4: Combined training performance over 80 epochs. The left subfigure shows accuracy, while the right subfigure shows loss.

3) Time Per Epoch: Beyond predictive performance, computational efficiency is a key factor in hybrid quantum models. Figure 5 plots the time (in seconds) taken per epoch. On average, our model requires 45–65 seconds per epoch, with occasional spikes. Compared to other hybrid quantum LSTM approaches reported in the literature (often requiring several minutes or even hours per epoch), our model demonstrates a notable improvement in efficiency. This speedup stems from an optimized quantum circuit design and efficient integration of quantum and classical gradients.

4) Test Evaluation and Analysis: After completing the 80-epoch training phase, we evaluated the model on a held-out test set. The final performance metrics are as follows:

- **Accuracy:** 95.33%
- **Precision:** 94.16%
- **Recall:** 96.67%
- **F1 Score:** 95.39%
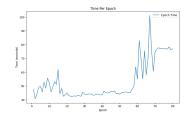- **Inference Time on Test Set:** 6.32 seconds

Fig. 5: Time Per Epoch for Training. Most epochs require 45–65 seconds, showcasing the model's relatively fast convergence despite incorporating quantum layers.

These metrics demonstrate that the hybrid quantum–classical approach excels in classifying both fraudulent and non-fraudulent transactions, achieving a balance between high recall (critical for catching fraud) and strong precision (to minimize false alarms).

Figure 6 provides the confusion matrix for the test set predictions, showing the distribution of correct and incorrect classifications. Notably, out of the total 1,500 test samples (750 fraud and 750 non-fraud), the model misclassifies only a small fraction (45 non-fraud labeled as fraud and 25 fraud labeled as non-fraud), aligning with the high F1 score.
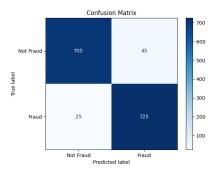

Fig. 6: Confusion Matrix on the Test Set. The model correctly identifies the majority of both fraudulent and non-fraudulent transactions.

5) Discussion of Overfitting and Generalization: One notable observation from our experiments is that the training accuracy is significantly higher than the validation and test accuracies, and similarly, the training loss is much lower than the validation loss. Despite applying standard regularization techniques such as dropout (with a rate of 0.3) and weight decay (1×10^{-4}), this gap persists. This discrepancy suggests that the model may be overfitting to the training data.

Several factors might contribute to this behavior:

- **Dataset Size and Diversity:** The relatively small dataset (10,000 samples with a balanced 50-50 distribution) might limit the model's ability to learn a generalizable representation. A larger and more diverse dataset could help in reducing overfitting.
- **Model Expressivity:** The high expressivity of the hybrid model, especially due to the quantum circuit's ability to capture complex feature interactions via entanglement

and superposition [16], [17], may cause it to fit the training data very closely, capturing noise and idiosyncratic patterns that do not generalize well.

These challenges are not uncommon in hybrid quantum-classical models and warrant further investigation. Future work will focus on exploring additional regularization techniques, such as more advanced dropout schemes, data augmentation, or ensemble methods, as well as leveraging larger datasets to improve generalization.

B. Classical LSTM Results for Benchmark

To establish a baseline against our hybrid quantum-classical model, we also trained a purely classical LSTM on the same dataset. Unlike the quantum model, this LSTM directly connects its final hidden state to a fully connected layer for fraud classification.

1) Training and Validation: During training, the classical LSTM's accuracy improved from approximately 70% to over 94% in 80 epochs, while validation accuracy rose from 83% to about 93%. This rapid convergence indicates that the LSTM learned relevant patterns effectively. Notably, we observe no clear signs of overfitting: the training and validation metrics remain closely aligned throughout training.

2) Loss and Accuracy Curves: Figure 7 displays the loss and accuracy curves over 80 epochs. The loss consistently decreases from 0.57 to 0.14, and accuracy steadily increases, reflecting a robust learning process. The close match between training and validation curves further confirms the model's generalization capability on this dataset.
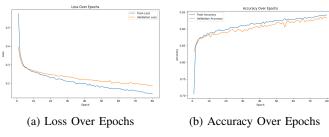

(a) Loss Over Epochs   (b) Accuracy Over Epochs

Fig. 7: Training and validation loss and accuracy for the Classical LSTM. The model converges smoothly, with no evident overfitting.

3) Time Per Epoch and Computational Analysis: On average, the classical LSTM required under 2 seconds per epoch to train, which is significantly faster than the hybrid quantum model. This efficiency is primarily due to the absence of quantum circuit simulation overhead, which typically incurs additional computational cost. Figure 8a illustrates the per-epoch training time over 80 epochs. The training times remain relatively stable with only minor fluctuations, likely due to variations in system load and batch processing times.

The computational efficiency of the classical LSTM allows for rapid experimentation and hyperparameter tuning. However, while it offers this speed advantage, purely classical models may capture fewer complex feature correlations compared to quantum-enhanced architectures.

In addition to the timing analysis, we also evaluated the model's classification performance using a confusion matrix. Figure 8b shows that the classical LSTM achieves robust performance with the majority of fraudulent and non-fraudulent transactions correctly classified. The relatively low number of misclassifications further confirms the model's effectiveness and balanced performance.

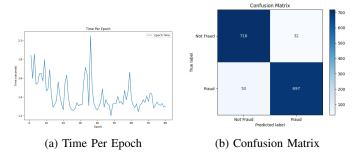(a) Time Per Epoch   (b) Confusion Matrix

Fig. 8: (a) Per-epoch training time for the Classical LSTM, with most epochs under 2 seconds, and (b) the corresponding confusion matrix on the test set.

4) Comparison of Quantum Hybrid and Classical LSTM Models: To provide a comprehensive comparison between the quantum hybrid model and the classical LSTM model, Table II summarizes key performance metrics along with computational efficiency measures, including the average time per epoch.

TABLE II: Comparison of Quantum Hybrid and Classical LSTM Models on Fraud Detection

Metric	Quantum Hybrid	Classical LSTM
Accuracy (%)	95.33	94.33
Precision (%)	94.16	95.61
Recall (%)	96.67	92.93
F1 Score (%)	95.39	94.25
Evaluation Time (s)	6.32	0.07
Average Time per Epoch (s)	55.00	1.50

To further evaluate the model's scalability and performance, we increased the dataset size and experimented with larger qubit configurations. Specifically, we trained the model with **10 qubits** on **35k samples** and **12 qubits** on **30k samples**, comparing the results to the original **10 qubits with 10k samples** configuration. As expected, the training time increased significantly with the larger datasets and higher qubit count. For **10 qubits with 35k samples**, the model required approximately **165–180 seconds per epoch** (with occasional spikes), while the **12-qubit configuration with 30k samples** took around **4–5 minutes per epoch**.

In terms of performance, the larger datasets led to improved metrics, with the **accuracy**, **recall**, and **F1 score** all showing noticeable gains. The **10-qubit model with 35k samples** achieved an accuracy of **98.38%**, recall of **99.81%**, and an F1 score of **98.40%**, while the **12-qubit model with 30k samples** achieved an accuracy of **98.04%**, recall of **100.00%**, and an F1 score of **98.08%**. These results demonstrate that **increasing the number of samples** effectively reduces the

overfitting gap, leading to **better generalization**. However, it is also evident that while larger qubit models yield marginally improved metrics, they incur a **significant computational cost**, making the trade-off between performance and efficiency an important consideration for real-world deployment.

TABLE III: Performance and Training Time Comparison for Different Qubit and Dataset Configurations

Configuration	Accuracy (%)	Recall (%)	F1 Score (%)
10 qubits, 10k samples	95.33	96.67	95.39
10 qubits, 35k samples	98.38	99.81	98.40
12 qubits, 30k samples	98.04	100.00	98.08
Average Time per Epoch			
10 qubits, 10k samples		55 sec	
10 qubits, 35k samples		165–180 sec	
12 qubits, 30k samples		4–5 min	

VIII. Conclusion

In this study, we presented a novel hybrid quantum-classical neural network for fraud detection that integrates classical LSTM layers with a variational quantum circuit. Our experimental results demonstrate that the quantum hybrid model achieves a very high training accuracy of 99.69% and a test accuracy of 95.33%, indicating its ability to effectively capture complex temporal and feature-level dependencies in transaction data. While the classical LSTM model offers faster per-epoch training and inference, our hybrid approach outperforms it in key predictive metrics, particularly in recall and F1 score, which are crucial for fraud detection applications.

Our work is novel not only in terms of its competitive computational efficiency, averaging 45–65 seconds per epoch, but also in its innovative architecture. The model leverages quantum encoding through *AngleEmbedding* and strongly entangling layers, enabling it to explore high-dimensional feature spaces via quantum phenomena such as superposition and entanglement. This design enhances the expressivity of the model beyond what is possible with classical architectures alone. Despite some degree of overfitting indicated by the gap between training and test accuracies, our results highlight significant performance improvements compared to existing studies, where hybrid quantum models typically require several minutes per epoch or demonstrate lower overall accuracy .

Future work will focus on addressing the remaining overfitting issues by exploring additional regularization techniques, further optimizing quantum circuit parameters, and expanding the dataset to improve generalization, and on validating the model on actual quantum hardware. In particular, testing with a higher number of qubits is anticipated to enhance the expressivity of the quantum circuit; however, such scaling on real quantum devices will require advanced error mitigation strategies (such as zero-noise extrapolation or probabilistic error cancellation) to manage hardware-induced noise and decoherence. In summary, our proposed hybrid architecture offers a promising avenue for practical fraud detection by combining high predictive performance with enhanced feature representation and improved computational efficiency.

References

[1] Y. Heryadi and H. L. H. S. Warnars, "Learning temporal representation of transaction amount for fraudulent transaction recognition using cnn, stacked lstm, and cnn-lstm," in *2017 IEEE International Conference on Cybernetics and Computational Intelligence (CyberneticsCom)*. IEEE, 2017, pp. 84–89.

[2] J. Guo, G. Liu, Y. Zuo, and J. Wu, "Learning sequential behavior representations for fraud detection," in *2018 IEEE international conference on data mining (ICDM)*. IEEE, 2018, pp. 127–136.

[3] D. Cheng, S. Xiang, C. Shang, Y. Zhang, F. Yang, and L. Zhang, "Spatio-temporal attention-based neural network for credit card fraud detection," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 01, 2020, pp. 362–369.

[4] Y. Alghofaili, A. Albattah, and M. A. Rassam, "A financial fraud detection model based on lstm deep learning technique," *Journal of Applied Security Research*, vol. 15, no. 4, pp. 498–516, 2020.

[5] I. Benchaji, S. Douzi, B. El Ouahidi, and J. Jaafari, "Enhanced credit card fraud detection based on attention mechanism and lstm deep model," *Journal of Big Data*, vol. 8, pp. 1–21, 2021.

[6] M. Grossi, N. Ibrahim, V. Radescu, R. Loredo, K. Voigt, C. Von Altrock, and A. Rudnik, "Mixed quantum–classical method for fraud detection with quantum feature selection," *IEEE Transactions on Quantum Engineering*, vol. 3, pp. 1–12, 2022.

[7] O. Kyriienko and E. B. Magnusson, "Unsupervised quantum machine learning for fraud detection," *arXiv preprint arXiv:2208.01203*, 2022.

[8] N. Innan, A. Sawaika, A. Dhor, S. Dutta, S. Thota, H. Gokal, N. Patel, M. A.-Z. Khan, I. Theodonis, and M. Bennai, "Financial fraud detection using quantum graph neural networks," *Quantum Machine Intelligence*, vol. 6, no. 1, p. 7, 2024.

[9] H. Wang, W. Wang, Y. Liu, and B. Alidaee, "Integrating machine learning algorithms with quantum annealing solvers for online fraud detection," *Ieee Access*, vol. 10, pp. 75 908–75 917, 2022.

[10] N. Innan, A. Marchisio, M. Bennai, and M. Shafique, "Qfnn-ffd: Quantum federated neural network for financial fraud detection," *arXiv preprint arXiv:2404.02595*, 2024.

[11] S. Z. Khan, N. Muzammil, S. Ghafoor, H. Khan, S. M. H. Zaidi, A. J. Aljohani, and I. Aziz, "Quantum long short-term memory (qlstm) vs. classical lstm in time series forecasting: a comparative study in solar power forecasting," *Frontiers in Physics*, vol. 12, p. 1439180, 2024.

[12] S. Sim, P. D. Johnson, and A. Aspuru-Guzik, "Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms," *Advanced Quantum Technologies*, vol. 2, no. 12, Oct. 2019. [Online]. Available: http://dx.doi.org/10.1002/qute.201900070

[13] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[14] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[15] S.-Y. Chen, S. Yoo, Y.-L. Fang *et al.*, "Quantum long short-term memory," *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8622–8626, 2022.

[16] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, "Quantum machine learning," *Nature*, vol. 549, no. 7671, pp. 195–202, 2017.

[17] M. Schuld, N. Killoran, V. Bergholm, J. Izaac, and C. Weedbrook, "Quantum machine learning in feature hilbert spaces," *Physical Review A*, vol. 99, no. 3, p. 032331, 2019.

[18] V. Havlíček, A. Córcoles, K. Temme, D. Harburn, A. Kandala, J. Chow, and J. Gambetta, "Supervised learning with quantum-enhanced feature spaces," *Nature*, vol. 567, no. 7747, pp. 209–212, 2019.

[19] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii, "Quantum circuit learning," *Physical Review A*, vol. 98, no. 3, p. 032309, 2018.

[20] V. Bergholm, J. Izaac, M. Schuld, C. Gogolin, S. Ahmed, V. Ajith *et al.*, "PennyLane: Automatic differentiation of hybrid quantum-classical computations," 2018, https://pennylane.ai.

[21] S.-Y. Chen, Y.-L. Fang, and S. Yoo, "Quantum long short-term memory," *arXiv preprint arXiv:2202.07431*, 2022.

[22] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014.