Hybrid Quantum-Classical Neural Networks for Stock Price Forecasting

Nathaniel James Pacey

Terra Quantum

April 28, 2025

1. Introduction

Stock market data presents a challenging and realistic benchmark for evaluating machine learning models due to its inherent complexity and dependence on a multitude of socio-economic factors [1, 2]. Modeling and preprocessing strategies were adapted from my bachelor's thesis on RNN-based forecasting of American-style stock options.

2. Methodology

2.1 Data Source and Preprocessing

Stock prices are unbounded and influenced by factors such as macroeconomics, sentiment, volatility, and corporate events. Events like stock splits alter share prices without affecting value, potentially misleading models if not handled correctly [3, 2]. Understanding such structural behavior is essential for reliable forecasting. This work uses daily closing prices for **AAPL**, **MSFT**, and **GOOGL** from **January 1, 2015** to **December 31, 2024**, obtained via the `yfinance` API. Missing values were forward-filled to maintain continuity, and prices were normalized per ticker using **Min-Max scaling** to the range **[0, 1]** [4]. To structure the data for forecasting, a **sliding window** of **20** time steps was applied, with each sequence mapped to its next price value. This approach preserved short-term dependencies and momentum patterns [5]. Data was split into training and testing sets using a time-respecting **80/20** ratio, ensuring no future leakage [5]. Sequences from all tickers were then combined to improve diversity and model generalization across financial instruments.

2.3 Model Architecture

Given their strong performance in financial forecasting, recurrent neural networks (RNNs) were chosen as the core machine learning unit [1, 3]. To enable flexible experimentation and optimization, the model was designed as a fully modular framework inspired by Terra's QAI Hub. Users can, select activation functions, adjust skip connections, apply regularization methods such as dropout and layer normalization, change the learning rate and optionally insert a quantum layer. Both the model architecture and quantum circuit are automatically visualized from configuration settings to enable rapid testing and comparison, with the final hybrid and classical models shown in Appendix A.1.

The classical processing pipeline is implemented in `PyTorch`. It begins with a user-defined sequence of recurrent layers, which may include single or mixed combinations of LSTM, GRU, and RNN units. The first recurrent block uses a configurable number of layers, while subsequent blocks default to a single layer. These layers process inputs of shape `(batch_size, sequence_length, 1)`, extracting temporal features from which the final hidden state is retained. This hidden state is then projected via a dense layer to match the number of qubits, forming the input to the optional quantum layer [6, 7].

If the quantum layer is enabled, the projected features are passed through a variational quantum circuit (built using PennyLane's `QNode` interface, as shown in Figure 1 [8]), where each feature is encoded via an RY gate [9]. Across a configurable number of variational layers (`q_depth`), each qubit undergoes a parameterized rotation: either `RY` (1 parameter), `RX + RZ` (2 parameters), or `Rot` (3 parameters). After each layer, qubits are entangled via a ring of `CNOT` gates to create a circular topology [10]. The output is obtained by measuring the expectation value of `PauliZ` on each qubit, producing a real-valued vector.

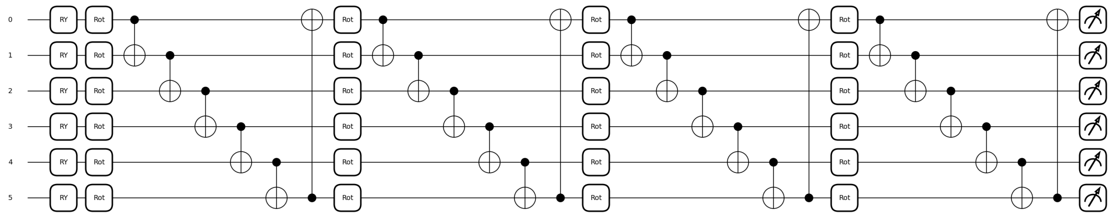


Figure 1: Optimal quantum circuit with `n_qubits=6`, `q_depth=4`, and `Rot` gates.

Post-quantum processing optionally applies non-linear activation functions (e.g., `Tanh`, `Sigmoid`) and regularization (dropout, layer normalization). The quantum output and pre-quantum classical features are merged either by concatenation or addition, depending on the `skip_connection` setting. The merged features are then passed through a final dense layer to generate the scalar prediction, optionally followed by an output activation function.

2.4 Training Strategy

The model was trained using the **mean squared error (MSE)** loss function and the **Adam** optimizer with a **learning rate of 0.001** [11]. To prevent overfitting, **early stopping** was applied with a patience window of 2–4 epochs depending on the experiment. During training, both **training loss** and **validation loss** were computed at every epoch. If no improvement in validation loss was observed within the specified patience, training was halted early. A training budget of **10 epochs** was used during experimentation, while **40 epochs** were allocated for final model evaluations. The best-performing model checkpoint (based on validation loss) was saved and later reloaded for testing. Model performance was evaluated using **MAE** (mean absolute error), **RMSE** (root mean squared error), and the **average percentage error** between predicted and true price values. Loss curves were also plotted to visualize convergence behavior and training dynamics [11].

2. Experimental Findings

Model design choices were guided by empirical results rather than assumptions. A series of structured experiments were conducted to evaluate different configurations of recurrent units, layer depths, dropout rates, activation functions, and quantum circuit parameters such as qubit count, depth, and rotation types.

2.1 Quantum Layer Optimization

A grid search was performed over key quantum circuit parameters, `n_qubits` from 2 to 6, `q_depth` from 1 to 7, and `n_rot_params` from 1 to 3, to evaluate their influence on forecasting performance. This included all combinations of gate complexity and circuit depth, with results assessed using MAE and average percentage error as shown in Figure 2.

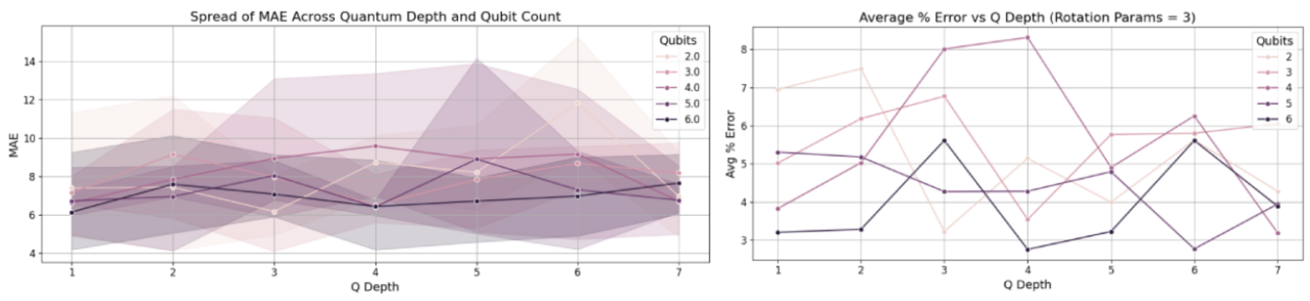


Figure 2: MAE and average error across qubit counts and depths for different gate rotation types.

To ensure that the quantum circuit could effectively capture the complexity of the input data, a more detailed analysis was conducted to identify when full rotation gates consistently outperformed simpler alternatives. This analysis revealed that the full rotation configuration becomes advantageous when using more than four qubits and circuit depths of at least four as seen in Appendix A.2. Based on these findings, the final quantum setup was selected as `n_qubits = 6`, `q_depth = 4`, and `n_rot_params = 3`.

2.5 Classical Architecture Optimization

Another challenge was optimizing the classical components of the hybrid model, as training with the quantum layer is time-consuming. To enable faster experimentation, classical components were first optimized using a classical-only model with limited epochs. Modular Python scripts were developed to perform parameter sweeps, varying one parameter at a time while holding others constant [1, 12, 13]. After each sweep, the best configuration was selected based on MAE, RMSE, and average percent error, and carried forward into subsequent sweeps. Coarse-grained sweeps were followed by fine-grained searches to further refine the parameter set. Parameters tested, sweep ranges, and optimal configurations are summarized in Table 1.

Parameter	Sweep Range	Best Result(s)
Hidden Size	{8, 16, 32, 64}	32
Dropout Rate	{0.0 - 0.7}	0.16 / None
Recurrent Layers	{1–8}	4
Recurrent Units	LSTM, GRU, RNN, combinations	GRU
Input / Output Activations	None, ReLU, Tanh, Sigmoid	Tanh / None
Skip Connection	concat, add	add

Table 1: Summary of classical architecture sweeps and selected best-performing configurations.

The final hybrid quantum and classical model configuration can be seen in Figure 6 and is: `GRU` with `4 layers`, `hidden size = 32`, `add skip connection`, `Tanh` post-quantum activation, and no output activation or dropout rate.

3. Results and Comparison

To assess model performance, both training and validation losses were tracked across epochs. The final hybrid models were trained for **40 epochs**, as shown in Figure 3. The best-performing model achieved a **training loss of 0.000083** and a **validation loss of 0.000346**.

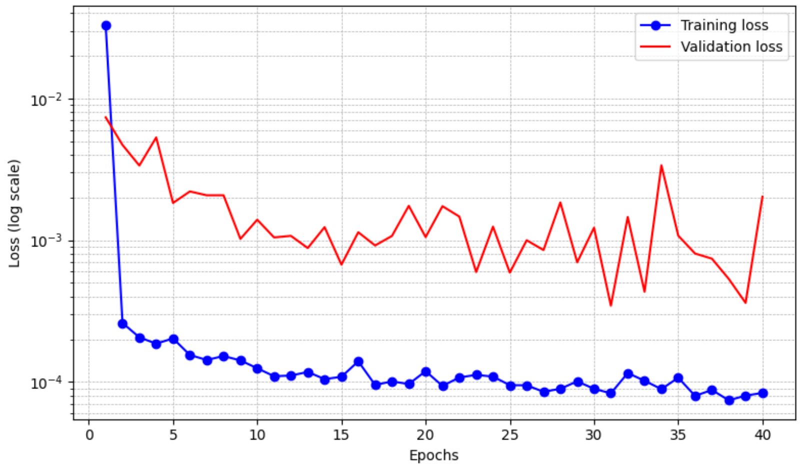


Figure 3: Training and validation losses over 40 epochs for the final Hybrid Quantum model.

While the training loss consistently decreased, the higher validation loss indicates overfitting, which is common when model capacity is high relative to data size. Reducing the number of recurrent layers or simplifying the quantum circuit may help improve generalization. However, the pattern may also reflect the *double descent* phenomenon, where validation loss initially worsens before improving as model complexity increases [14]. Further investigation is needed to explore both possibilities and identify the optimal model depth, circuit design, and training duration.

To further evaluate the hybrid model's predictive performance, the following metrics were computed on the test set: **Test MAE = 2.5660**, **Test RMSE = 3.1719**, and an **Average Absolute Percentage Error of 1.77%**. Additionally, the distribution of prediction errors was analyzed, as shown in Figure 4.

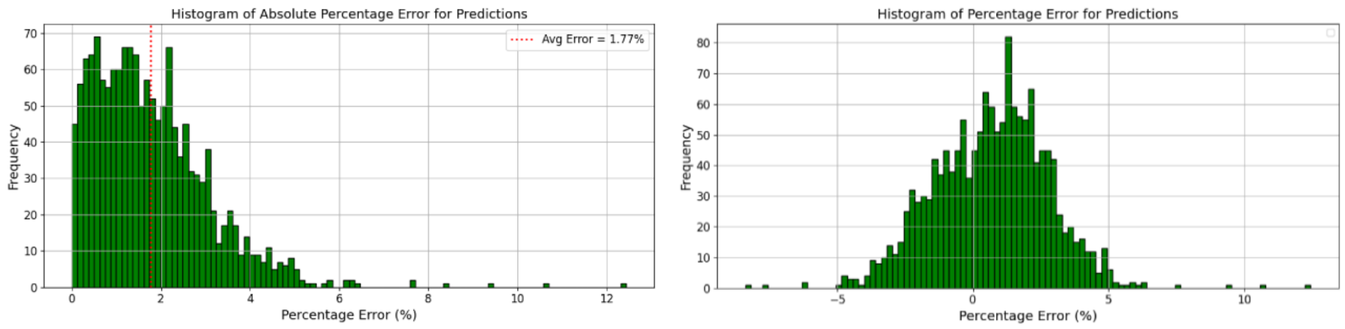


Figure 4: Distribution of prediction errors (percentage) for the hybrid model on test data.

The histogram reveals that most prediction errors fall within 0–2% absolute error, with the distribution centered near 0%. This near-Gaussian error distribution around zero suggests that the hybrid model achieved a high level of accuracy and produced unbiased predictions on the test data. The predicted results for a sample stock (**GOOGL**) are visualized against the true price values in Figure 5.

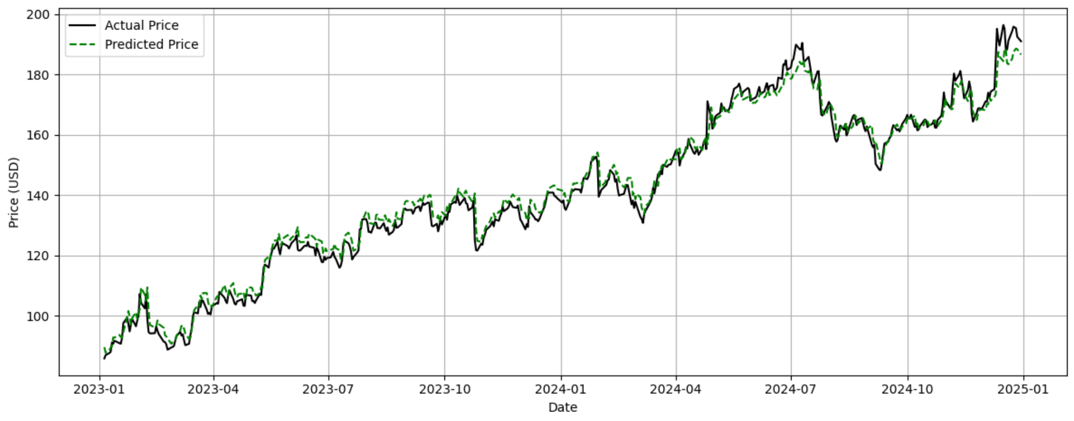


Figure 5: Predicted versus true prices for GOOGL using the hybrid quantum-classical model.

The model's predictions closely track the true stock price, often overestimating during upward trends and underestimating before price declines; patterns that could help identify potential buying and selling opportunities for traders. To assess the added value of the quantum layer, the same model was evaluated with the quantum layer disabled, which achieved a **Test MAE = 3.0522**, **Test RMSE = 3.9727**, and an **Average Absolute Percentage Error of 1.98%**. Histograms showing the distribution of prediction errors for the classical model can be found in Appendix A.3.

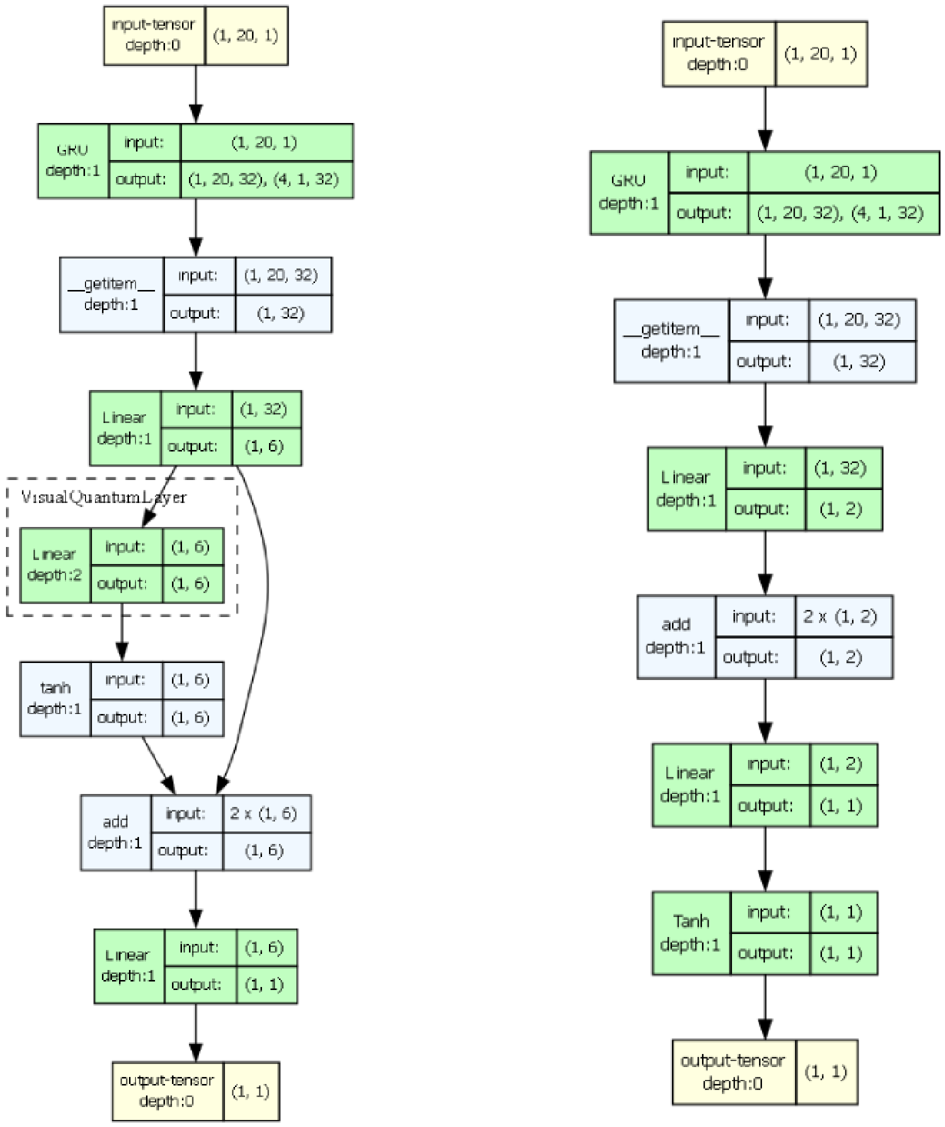
While the hybrid model shows a **slight improvement** over the classical baseline, the margin falls within typical predictive uncertainty ranges. As such, **no definitive advantage** of the quantum-enhanced model can be claimed based on this result alone. Further testing across additional datasets and scenarios would be required to conclusively demonstrate consistent performance gains from the hybrid approach.

4. Future Work

While the hybrid quantum-classical model demonstrated promising results, several avenues remain for future improvement. One major limitation was the long runtime required to train and evaluate quantum layers. Implementing parallel processing, utilizing GPU acceleration, or deploying models on cloud-based quantum platforms could significantly speed up experimentation and allow for more consistent and thorough parameter sweeps without manual cross-referencing using the classical model. Further optimization could target additional hyperparameters such as the learning rate and explore alternative quantum circuit designs. The dataset itself could also be expanded to include more features or additional stock tickers, increasing generalization and robustness. Additionally, testing the quantum circuit on different platforms such as Qiskit or running on real quantum hardware offered by various providers could offer insight into platform-specific advantages or limitations. Finally, evaluating the model on live financial data or unseen stocks would help assess its real-world usability and performance beyond the original dataset.

A Appendix

A.1 Model Architecture Visualization



input-tensor depth:0 (1, 20, 1)
GRU depth:1 input: (1, 20, 1) output: (1, 20, 32), (4, 1, 32)
__getitem__ depth:1 input: (1, 20, 32) output: (1, 32)
Linear depth:1 input: (1, 32) output: (1, 6)
VisualQuantumLayer
Linear depth:2 input: (1, 6) output: (1, 6)
tanh depth:1 input: (1, 6) output: (1, 6)
add depth:1 input: 2 x (1, 6) output: (1, 6)
Linear depth:1 input: (1, 6) output: (1, 1)
output-tensor depth:0 (1, 1)
input-tensor depth:0 (1, 20, 1)
GRU depth:1 input: (1, 20, 1) output: (1, 20, 32), (4, 1, 32)
__getitem__ depth:1 input: (1, 20, 32) output: (1, 32)
Linear depth:1 input: (1, 32) output: (1, 2)
add depth:1 input: 2 x (1, 2) output: (1, 2)
Linear depth:1 input: (1, 2) output: (1, 1)
Tanh depth:1 input: (1, 1) output: (1, 1)
output-tensor depth:0 (1, 1)

Figure 6: Model architecture overview. Left: Hybrid quantum-classical configuration. Right: Classical GRU-only baseline.

A.2 Rotational Parameter Sweep Results

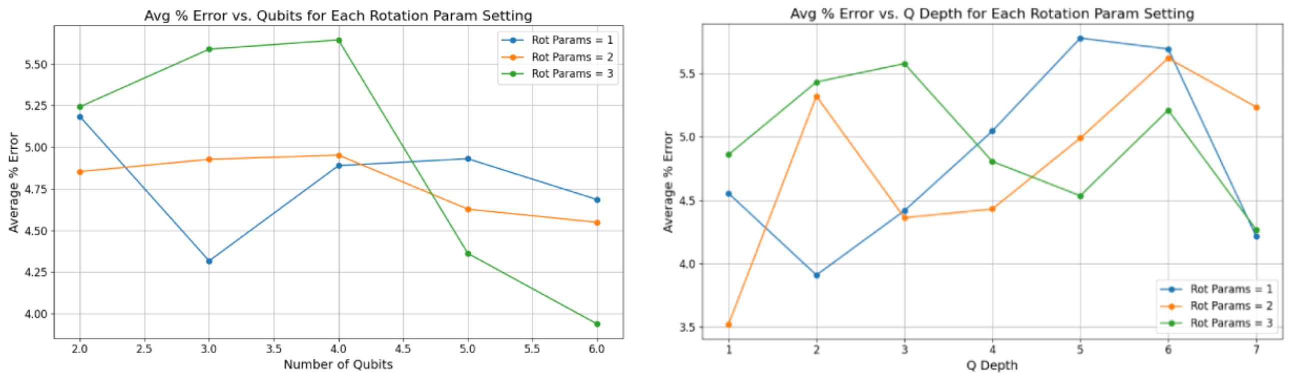


Figure 7: Average error segmented by rotation complexity across different qubit counts and depths.

A.3 Classical Model Prediction Error Distributions

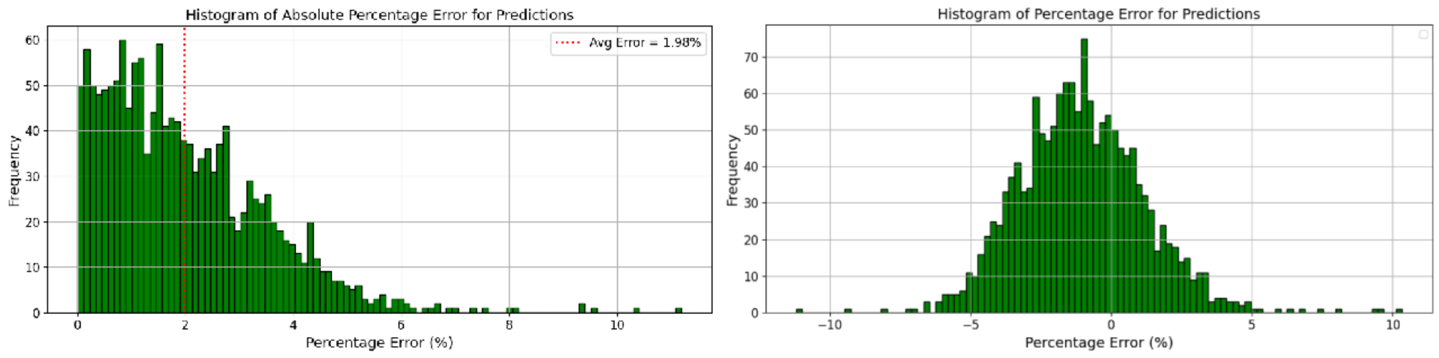


Figure 8: Distribution of prediction errors for the classical model.

References

[1] V. Madisson, "Stock market predictions with rnn using daily market variables," 2019. Accessed: 2023-10-11.

[2] J. Chen, "What are options? types, spreads, example, and risk metrics," 2023. Accessed: 2023-10-13.

[3] I. Jahan, *Stock Price Prediction Using Recurrent Neural Networks.* PhD thesis, North Dakota State University, 2018.

[4] M. Lesuisse, "What are the advantages of pricing american options using artificial neural networks?," Master's thesis, Université de Liège, 2022.

[5] T. Fischer and C. Krauss, "Deep learning with long short-term memory networks for financial market predictions," *European Journal of Operational Research*, vol. 270, no. 2, pp. 654–669, 2018.

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

[7] S. Kostadinov, "Understanding gru networks," 2017. Accessed: 2024-03-01.

[8] M. Schuld, V. Bergholm, C. Gogolin, J. Izaac, and N. Killoran, "Machine learning with pennylane," *Quantum*, vol. 5, p. 529, 2021.

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

[10] M. Benedetti, E. Lloyd, S. Sack, and M. Fiorentini, "Parameterized quantum circuits as machine learning models," *Quantum Science and Technology*, vol. 4, no. 4, p. 043001, 2019.

[11] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning.* MIT Press, 2016.

[12] D. Kalita, "A brief overview of recurrent neural networks (rnn)," 2023. Accessed: 2023-10-13.

[13] M. Saeed, "An introduction to recurrent neural networks and the math that powers them," 2023. Accessed: 2023-10-13.

[14] M. Belkin, D. Hsu, S. Ma, and S. Mandal, "Reconciling modern machine-learning practice and the classical bias–variance trade-off," *Proceedings of the National Academy of Sciences*, vol. 116, no. 32, pp. 15849–15854, 2019.