Quantum Computing based Generative Adversarial Network for Time-Series forecasting

Aadittya Tiwari1

1 Lynbrook High School, 1280 Johnson Ave, San Jose, 95129, California

\*Corresponding author; E-mail: AadiTiwari1223@gmail.com

Abstract

Generative Adversarial Networks (GANs) are popular machine learning models that can generate synthetic data by training a generator to produce samples that a discriminator can't distinguish from real data. GANs have shown promise for time-series forecasting, where synthetic values can be used to predict future trends, such as stock prices. However, this can be taken one step further with the usage of Quantum Computers. In this project, we developed a novel Quantum GAN (QGAN) architecture that combined a Quantum variation of a Long Short Term Memory generator with a classical Convolutional Neural Network discriminator. The QGAN outperformed its classical counterpart in number of convergence epochs required and had a slightly higher prediction accuracy across a range of stock datasets, despite using a low qubit count. These results provide evidence of a quantum advantage in the domain of deep learning.

**Keywords:** Quantum Computing, Machine Learning, Time-series Forecasting

# Introduction

# Quantum Machine Learning (QML) is an emerging interdisciplinary field combining quantum computing with machine learning algorithms to enhance data processing. Quantum Computers utilize “qubits” (quantum-bits) which exist simultaneously in multiple states, unlike classical bits that can only exist as either a 0 or 1, called quantum superposition. Groups of Qubits in superposition can create complex, multi-dimensional computational spaces which can be then used to solve complex problems much more efficiently when compared to classical supercomputers [12]. By storing multiple states at once, machine learning algorithms can be optimized for Quantum Computers to process and analyze vast amounts of data faster and obtain more precise results. As QML develops as a field, new algorithms are found that demonstrate “Quantum Advantage,” cases where Quantum Computing algorithms have advantages over Classical Computing algorithms. QML has the potential to transform research fields such as disease detection, physical simulations, but for the context of this project, time-series forecasting, by providing more accurate and faster solutions for these complex problems. This paper describes a novel quantum machine learning model that does indeed demonstrate Quantum Advantage in the field of Generative Adversarial Networks for time-series forecasting. The paper is structured as follows: In section two we define Quantum Computing and Machine Learning. Section 3 outlines the architecture of the QGAN. Section 4 describes the implementation of the QGAN. Section 5 contains evaluation methods and results of the QGAN’s performance against a classical counterpart. Section 6 concludes the paper with a summary of the model and potential areas of improvement.

# Quantum Computing and Machine Learning

## Simple introduction to Quantum Computing

Quantum computing is an emerging field of computer science that promises to revolutionize how we solve complex problems. Unlike classical computers, which rely on bits that can only exist in one of two states, quantum computers use quantum bits, or qubits, which can exist in multiple states simultaneously. This allows quantum computers to perform certain calculations much faster than classical computers, making them particularly well-suited for tasks such as optimization, cryptography, and in our paper, machine learning.

One of the key features of quantum computing which makes it so powerful is the phenomenon of superposition, which allows a qubit to exist in a combination of both 0 and 1 at the same time. In pop culture it's often said that an n number of qubits correlate to the power of classical bits. While this isn't entirely true, it gives a rough idea of the computational advantage quantum computers provide. Another important feature is entanglement, where two or more qubits can be correlated in such a way that their states are linked together so that affecting one qubit will affect its entangled qubit as well. This paired with superposition is what provides such powerful exponential speed-ups in time complexities.

Quantum computing is still in its early stages, and many of the practical applications of this technology have yet to be realized. However, there has been significant progress in recent years, with the development of increasingly powerful quantum computers and the occasional demonstration and realization of advantages over classical computers for cherry-picked scenarios. However, Quantum Machine Learning as of late has been receiving a lot of attention in the last few years due to it being a conjunction of 2 of the most revolutionizing fields in Computer Science - Quantum Computing, and Machine Learning.

## Simple introduction to Machine Learning

Machine learning is a branch of artificial intelligence that allows machines to learn from data without being explicitly programmed. It involves training algorithms to identify patterns in data and make predictions or classifications based on those patterns. The process of machine learning consists of three main stages: data preparation, model training, and model evaluation. During the data preparation stage, data is collected, cleaned, and transformed into a format suitable for analysis. In the model training stage, machine learning algorithms are used to train models on the prepared data. Finally, in the model evaluation stage, the trained models are tested on new data to ensure their accuracy and effectiveness.

**2.2.2** **Generative Adversial Networks (GANs)**

Generative Adversarial Networks, or GANs, are a type of neural network machine learning architecture that can generate new data that is similar to the training dataset. They consist of two models - a Generator and a Discriminator - that compete to learn and generate complex data such as images, audio, and video files. The Generator creates fake data to train on the Discriminator, which in turn learns to identify real data from the fake data produced by the Generator. This adversarial game between the two models continues until the Generator produces data that is indistinguishable from the real data.

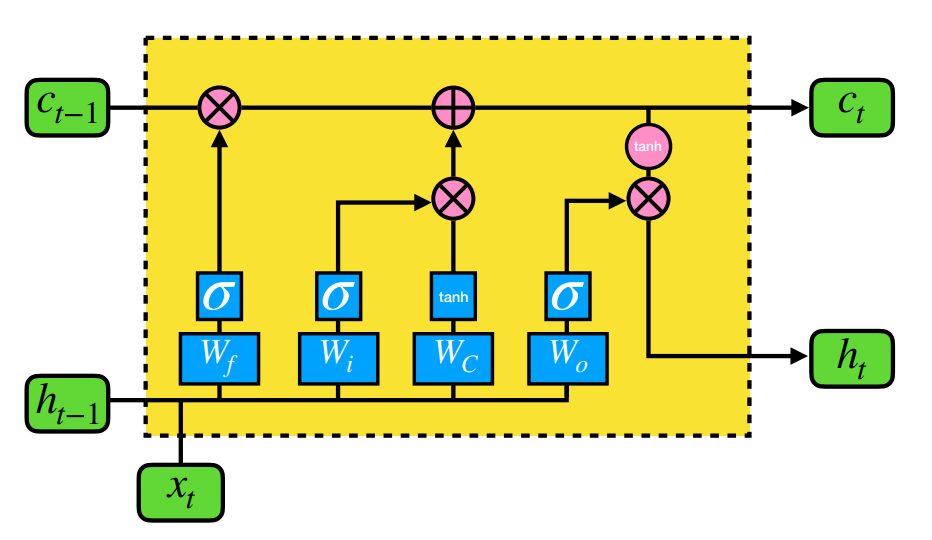
# Quantum GAN Generator and Discriminator

## Quantum Long Short-Term Memory Architecture

This paper proposes a Quantum GAN (Q-GAN), which is a hybrid machine learning model that combines quantum and classical computing. The generator for the Q-GAN is a Quantum Long Short-Term Memory (QLSTM) model developed by Chen et al. The QLSTM replaces a layer in a classical LSTM circuit with 6 Variational Quantum Circuits (VQCs) to form a QLSTM cell. VQCs are a type of quantum circuit used in quantum machine learning and optimization, which are parameterized by a set of randomly initialized parameters and optimized to minimize a cost function related to the task objective. VQCs have the advantage of being able to represent complex functions using a small number of qubits and gates, thus theoretically providing computational advantages over classical machine learning algorithms. This QLSTM generator uses 4 qubits from the Pennylane's default.qubit Quantum Computer simulator. The new quantum layer in the model was found to have quicker convergence and a more stable loss function graph than its classical counterpart. To recap the differences, the information flows from which data is processed between LSTMs and QLSTMs is shown below.

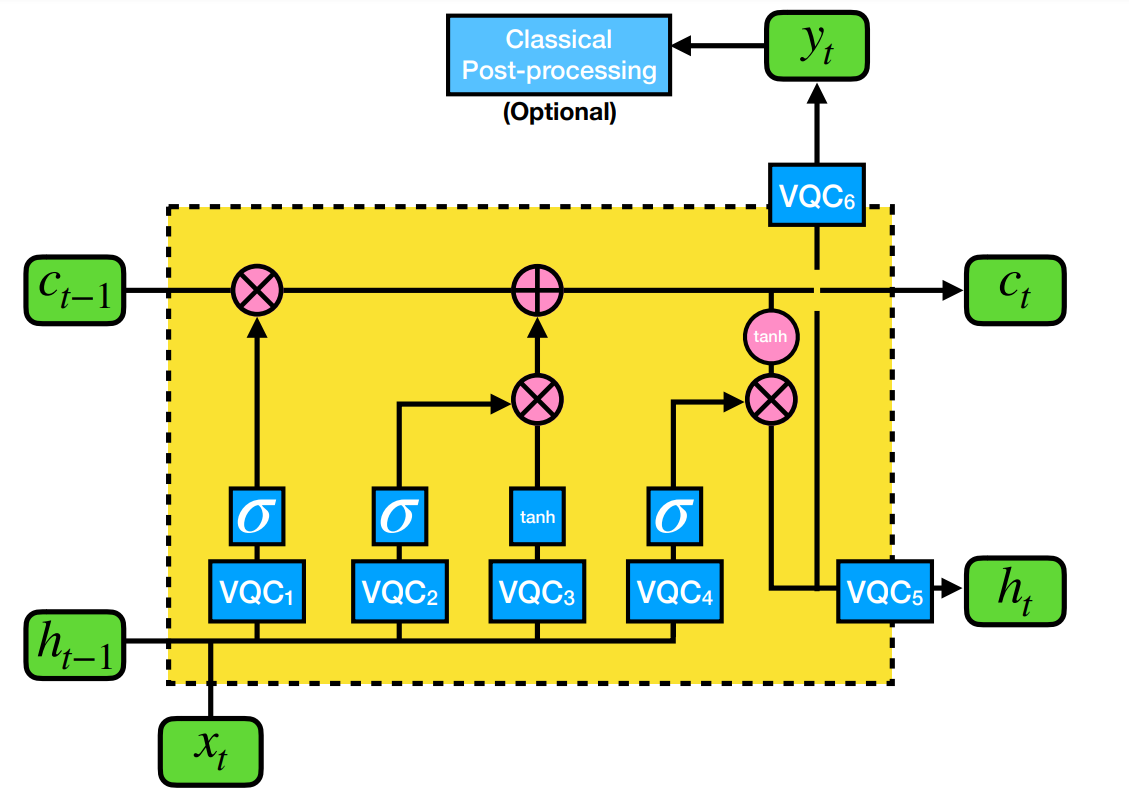
The information flow in an LSTM is as following:

where denotes the sigmoid function, are classical neural networks , where represents the forget block, represents the input block, represents a new state cell candidate, and represents the output block. The architecture of the LSTM can be visualized with the following image:

S

The information flow in a quantum LSTM is:

Where denotes a different Variational Quantum Circuit that will be used in the hybrid Quantum LSTM. The equations are relatively like each other, with the equation for being identical through both LSTMs. The difference, and supposed advantage comes from the usage of the VQCs in the Quantum LSTM. The architecture of the QLSTM can be visualized with the following image:

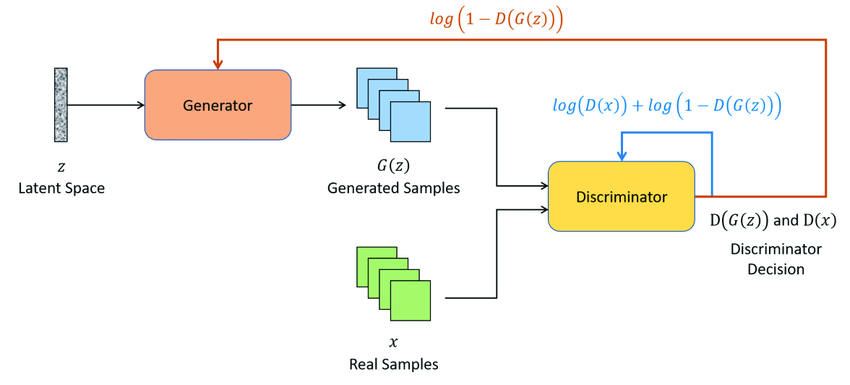
Each VQC box is of the form as detailed in Figure 4. The and tanh blocks represent the sigmoid and the hyperbolic tangent activation function, respectively. is the input at time t, is for the hidden state, is for the cell state, and is the output. ⊗ and ⊕ represents element-wise multiplication and addition, respectively. [1]

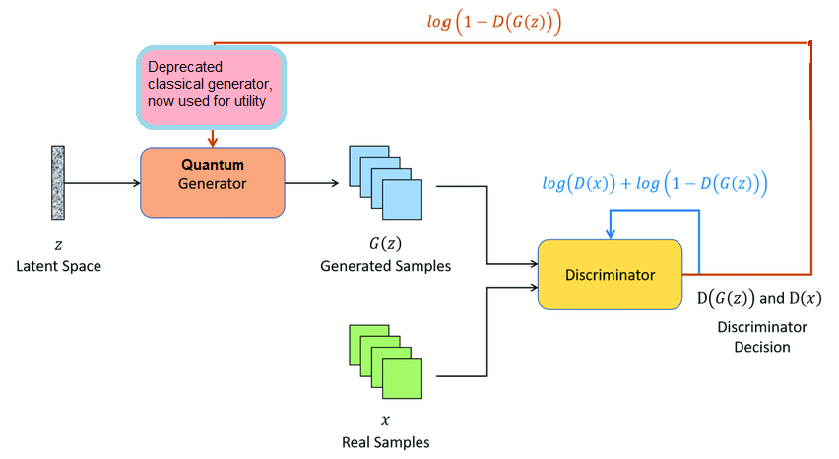
## Classical Convolutional Neural Network Architecture

This Discriminator architecture consists of several layers of convolutional and fully connected (linear) layers, with batch normalization and activation functions (LeakyReLU and ReLU). The input to the Discriminator is a tensor with dimensions (batch\_size, 4, *seq\_len*), where 4 is the number of channels and *seq\_len* is the length of the input sequence. The output of the last linear layer is passed through a sigmoid activation function, which outputs a probability between 0 and 1 indicating whether the input sample is real (1) or fake (0). The Convolutional Neural Network (CNN) architecture of this Discriminator comprises three convolutional layers with 32, 64, and 128 filters, respectively. The kernel size is set to 3, and the stride is 1. The padding is set to 'same', meaning the output feature map has the same size as the input feature map. The output of the convolutional layers is passed through two fully connected layers with 220 neurons each, with batch normalization applied to both. The output of the first fully connected layer is passed through a LeakyReLU activation function with a negative slope of 0.01, while the output of the second fully connected layer is passed through a ReLU activation function. The Discriminator network is trained to minimize the binary cross-entropy loss between the predicted probabilities and the true labels (1 for real samples and 0 for fake samples). The network is optimized using the Adam optimizer with a learning rate of 0.004. In the CNN, A one-dimensional convolution is applied to input signals. In the simplest case, the output value of the layer with input size and output can be described with the equation

[9].

# Q-GAN implementation

Moving on from the technical architecture, we begin to implement the Quantum LSTM into a time series generator. Essentially, all we do is replace the classical generator in the Classical GAN (CGAN) (which is currently a classical prebuilt PyTorch LSTM) with our custom Quantum LSTM. However, as our model is not prebuilt and the GAN relies on a prebuilt CNN model for the discriminator, we will need to still keep the code for the prebuilt generator and instead create what is basically a second instance of the generator and call that instead, while the classical generator remains but not for the actual purpose of generation, more as a utility function. An example of how this works is shown below, with the first image depicting a traditional GAN and the second depicting the QGAN. Classical GAN architecture.



Quantum GAN architecture. Again, all that changed in the overall GAN was giving the generation task to the Quantum LSTM while the prebuilt is just a utility function. It does no generation and only serves as a function to ensure the CNN works as intended.

## Variables

The variables for both models were set after lots of experimentation to find optimal values to allow both functions to perform at their best but keeping as many values alike as possible to allow objective testing. The data table as follows shows what each value was set to in the following categories for the generators and the discriminators:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Generator | Learning Rate | Epochs | Optimizer | Layers |
| QGAN | 0.000016 | 15 | Adam | 4 |
| CGAN | 0.004 | 100 | Adam | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Discriminator | Learning Rate | Epochs | Optimizer | Layers |
| QGAN | 0.00003 | 300 | Adam | 3 |
| GAN | 0.00003 | 300 | Adam | 3 |

The variables for the discriminator are identical since we are using the exact same Convolutional Neural Network (CNN) model for both GANs.

# Model evaluation methods (Experiments & Results)

## Evaluation Methods

To assess the effectiveness of our QGAN model for stock price prediction, we propose the following evaluation methods:

1. Selecting 3 stocks with differing trends to encourage robust models

To ensure that our model is robust, we will select three stocks with differing trends in the market. Specifically, we will select one stock with an upward trend, one with a downward trend, and one with a relatively stable trend. This will help us determine the model's ability to predict stocks across a range of market conditions. The data for these stocks will come from the ‘yfinance’ python package which gives us access to stock price and related info for the 3 stocks we will be choosing: AAPL (Apple), INTC (Intel), and SNPS (Synopsys). This data and package is public for anyone to use.

2. Using an existing classical GAN for time series forecasting

To establish a baseline for our QGAN model's performance, we will use an existing classical GAN model for time series forecasting. This will provide us with a benchmark for comparison and enable us to determine whether our QGAN model offers any improvements in prediction accuracy.

3. Developing a novel QGAN model that changes ONLY the generator of the classical GAN

We will develop a novel QGAN model for stock price prediction by changing only the generator of the classical GAN. This will allow us to compare the performance of our QGAN model to that of the classical GAN model, while keeping all other parameters constant.

Step 4: Evaluating the performance of both models on the 3 different stocks in the categories of prediction RMSE, epochs to converge, and parameters across all 6 predictions

To evaluate the performance of both models, we will use the following metrics:

* Prediction RMSE: This metric will measure the difference between the predicted stock prices and the actual stock prices for each of the three selected stocks. The RMSE values will be calculated for both the classical GAN and the QGAN models, and compared to determine which model performs better in terms of prediction accuracy.
* Epochs to converge: We will measure the number of epochs required for each model to converge on a stable prediction. This will help us determine which model requires less training time to achieve accurate predictions.
* Parameters across all 6 predictions: We will compare the number of parameters used in both models across all six predictions (two predictions per stock). This will enable us to determine which model is more efficient in terms of parameter usage.

By using these evaluation methods, we aim to determine the effectiveness of our QGAN model for stock price prediction and compare it to a classical GAN model for time series forecasting.

**5.1.1** Back testing, Unseen data, and avoiding overfitting

As mentioned above, we will be evaluating the Classical GAN and the Quantum GAN on 3 different stocks. This means that only one model will be created to predict 3 stocks. We will also be providing training and testing data forecast graphs for AAPL stock predictions and including them in the project GitHub. By doing these two things, we can back test, evaluate for unseen data, and avoid overfitting.

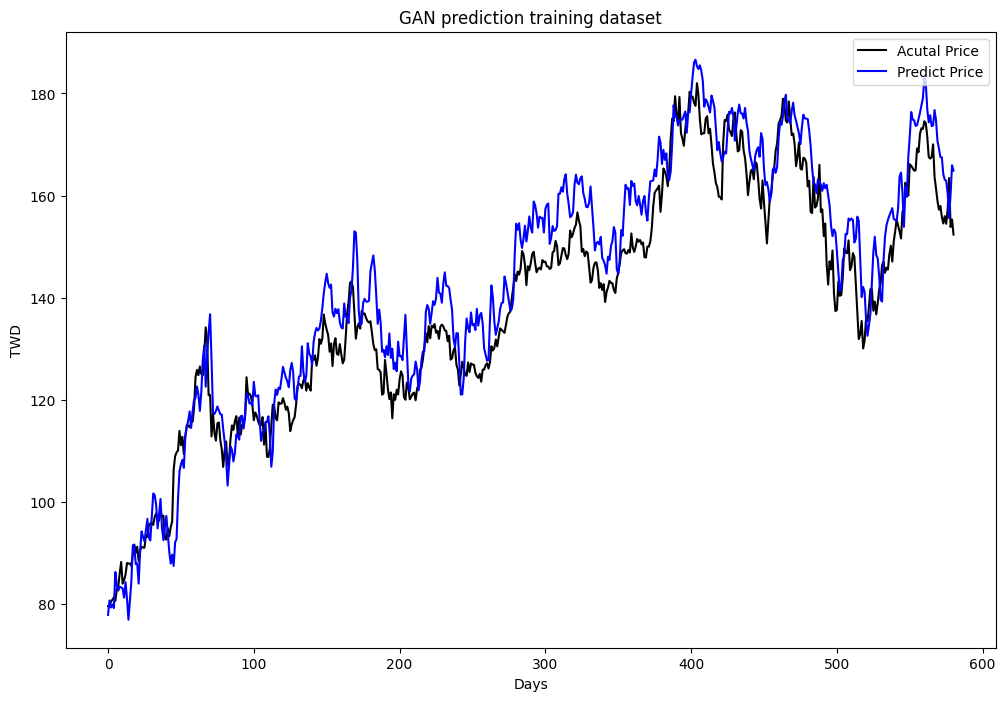
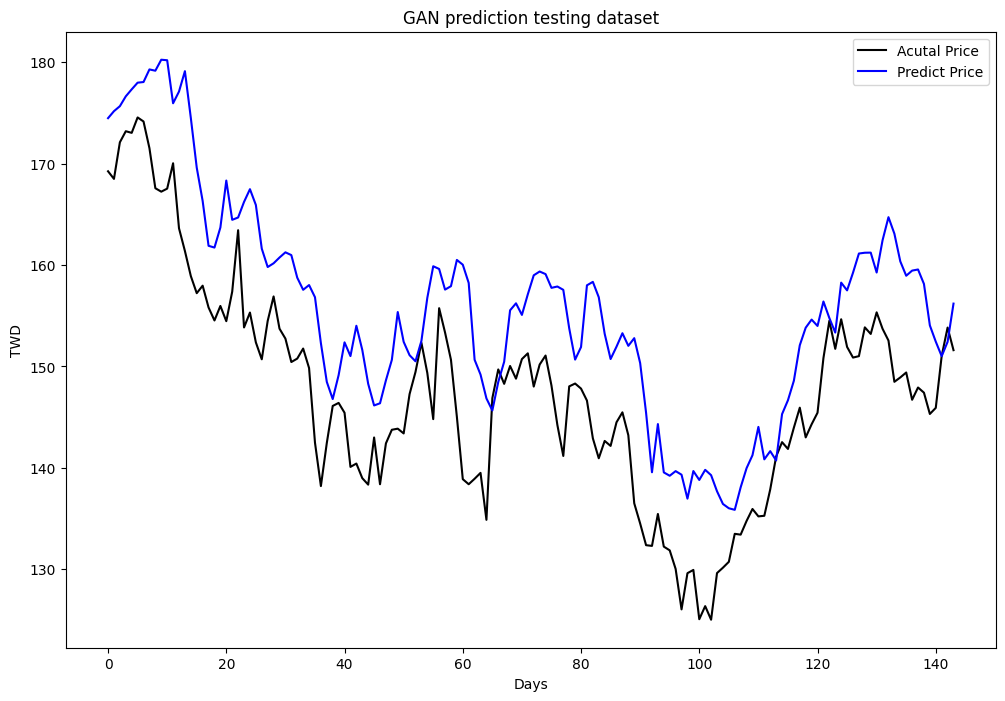
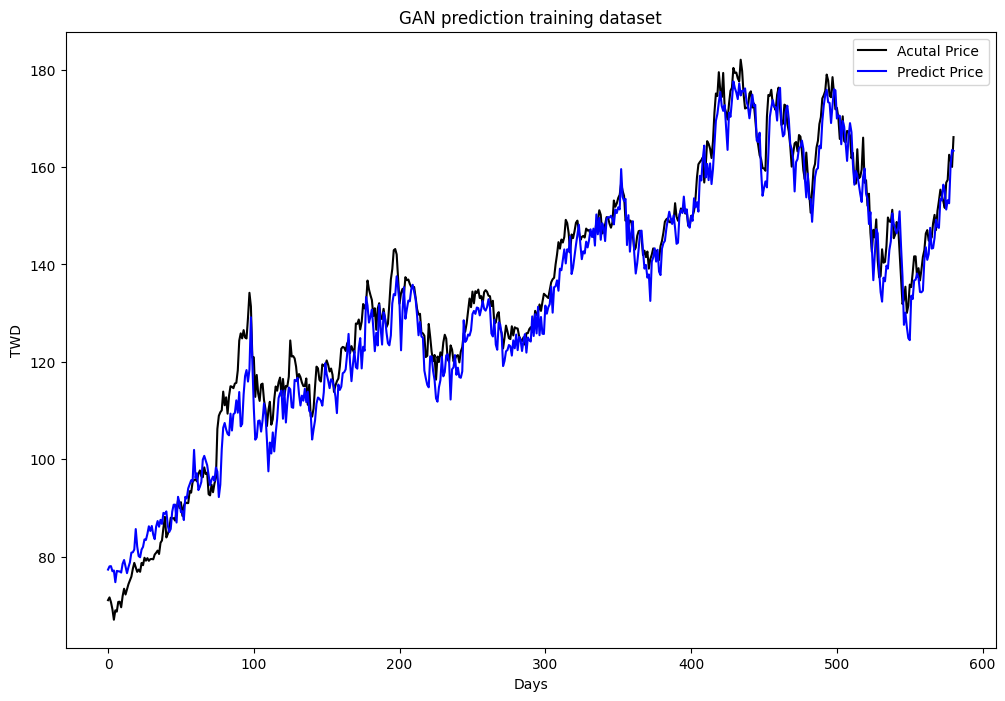
## Results

The following data tables show the performance of each model across the evaluation metrics defined above:

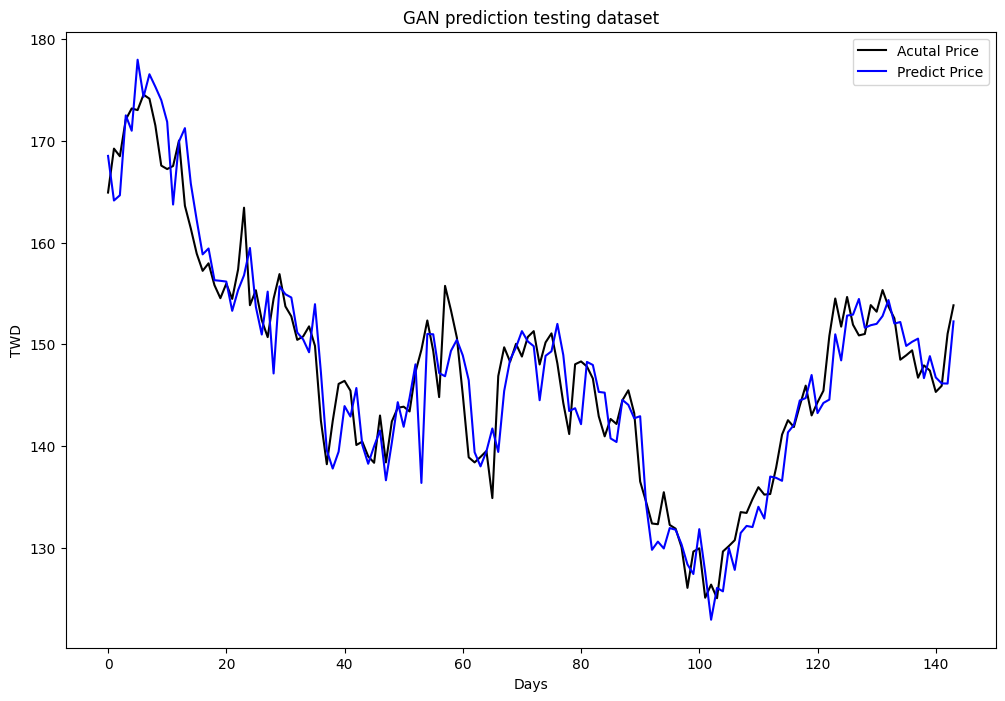
|  |  |  |
| --- | --- | --- |
| RMSE | CGAN | QGAN |
| Intel prediction | 7.11 | 13.10 |
| Apple prediction | 7.97 | 3.61 |
| Synopsys prediction | 4.77 | 8.50 |

|  |  |
| --- | --- |
| Epochs taken to reach convergence |  |
| QGAN | 6 out of 15 total epochs |
| CGAN | 82 out of 100 total epochs |

|  |  |
| --- | --- |
| Total parameters (Generator + Discriminator) |  |
| QGAN | 233+6199041 |
| CGAN | 109429+6199041 |

Graphically, the classical and Quantum GANs both performed best when predicting the stock prices for Apple, presumably due to its comparatively fewer volatile trends making for less calculations during the models’ attempts at predictions. The graphs are shown below along with the epochs taken by each model to achieve the result shown. The black lines represent the actual prices, and the blue line represents the respective model’s predictionClassical GAN’s training forecast for Apple stock prices after 100 epochs of trainingClassical GAN’s forecast for Apple stock prices after 100 epochs of training

Quantum GAN’s training forecast for Apple stock prices after 15 epochs of training

Quantum GAN’s forecast for Apple stock prices after 15 epochs of training

# Conclusion and Outlook

QML almost always returned higher accuracy for forecasts, albeit minimal. This higher accuracy is indeed a result of the QLSTM being used and having it perform calculations through a quantum computing simulator which then gave the QLSTM an advantage in converging quicker. QML was also able to reach comparable performance to CML in a fraction of the epochs. This is again related to the Qubits above - as Quantum Bits can exist in multiple states at once, it’s able to compute different possibilities at the same time. This means it trains more in one epoch than Classical ML does in one epoch because of these parallel computations. Another interesting result was that there were significantly less parameters in the QML models, again because of having qubits that make up for the lack of parameters. Finally, because of using a simulator (a classical computer coded to behave like a quantum computer) and not a real quantum computer, it will unfortunately take you more time to train your quantum models unless you use a real quantum computer. While classical epochs could be trained in under 10 seconds, epochs for quantum epochs took anywhere from one minute to six. Possible room for improvement would be to try out different QC simulators from different companies, or in the far future evaluate performance on a real quantum computer. As hardware scales with time, so will algorithms making QML models such as this one far more effective and pave the way for algorithms that aren’t constructed from existing classical machine learning algorithms and are instead unique with no classical counterparts. Overall, the QGAN has shown itself to be a competitive counterpart to classical GAN’s in terms of performance and further shows the future of Quantum Machine Learning as a viable alternative to Classical Machine Learning.

# Conflicts of Interest and author contributions

Being the only author, Aadittya Tiwari was responsible for the entirety of this paper and its corresponding code.

The author has no affiliation with any of the companies or products mentioned/used in this research paper.

# Code

The github repositiory containing all of the code in the form of jupyter notebooks can be found at this link: <https://github.com/AadiTiwar1/Quantum-GAN-for-Time-Series-Forecasting> .

# References

1. Chen, S. Y.-C., Yoo, S., & Fang, Y.-L. L. (2022). Quantum long short-term memory. *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. <https://doi.org/10.1109/icassp43922.2022.9747369>
2. Cerezo, M., & Arrasmith, A. (2021). Variational quantum algorithms. *Nature Reviews Physics*, *3*(9), 625–644. <https://doi.org/10.1038/s42254-021-00348-9>
3. Cong, I., Choi, S., & Lukin, M. D. (2019). Quantum Convolutional Neural Networks. *Nature Physics*, *15*(12), 1273–1278. <https://doi.org/10.1038/s41567-019-0648-8>
4. Goodfellow, I. J. (2023). Generative adversarial nets. *ArXiv*, 73–76. <https://doi.org/> <https://doi.org/10.48550/arXiv.1406.2661>
5. Yoon, J., Drumright, L. N., & van der Schaar, M. (2020). Anonymization through data synthesis using generative adversarial networks (ADS-gan). *IEEE Journal of Biomedical and Health Informatics*, *24*(8), 2378–2388. <https://doi.org/10.1109/jbhi.2020.2980262>
6. Lin, H. C., Chen, C., Huang, G. F., & Jafari, A. (2021). Stock price prediction using generative Adversarial Networks. *Journal of Computer Science*, *17*(3), 188–196. <https://doi.org/10.3844/jcssp.2021.188.196>
7. Dikshant Dulal, Yick Wei. (n.d.). *Dikshantdulal/softserve\_qlstm*. Retrieved January 3, 2023, from <https://github.com/DikshantDulal/SoftServe_QLSTM>
8. ChickenBenny. (n.d.). *Chickenbenny/stock-prediction-with-gan-and-WGAN: Stock prediction with Gan and WGAN*. GitHub. Retrieved April 11, 2023, from <https://github.com/ChickenBenny/Stock-prediction-with-GAN-and-WGAN>
9. *Training a classifier*. Training a Classifier - PyTorch Tutorials 2.0.0+cu117 documentation. (n.d.). Retrieved February 10, 2023, from <https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html>
10. *VQC*. VQC - Qiskit Machine Learning 0.6.0 documentation. (n.d.). Retrieved April 17, 2023, from <https://qiskit.org/ecosystem/machine-learning/stubs/qiskit_machine_learning.algorithms.VQC.html>
11. *Typical generative adversarial networks (GAN) architecture.* (n.d.). Retrieved April 17, 2023, from <https://www.researchgate.net/figure/Typical-Generative-Adversarial-Networks-GAN-architecture_fig2_349182009>
12. *What is quantum computing?* IBM. (n.d.). Retrieved January 31, 2023, from <https://www.ibm.com/topics/quantum-computing>