Examining Lee et al's - Mutual Information Maximizing Quantum Generative Adversarial Network and Its Applications in Finance

Prepared by

Luca Cappellano
Jackson Fraser
Giordano Kogler Anele

CPEN 400Q — Final Project

The University of British Columbia

Vancouver, BC

April 12th, 2024

I. Introduction

The incorporation of machine learning with finance has yielded new insights into the field and has proven to be an essential tool for automated algorithms. The work by Lee et al [1] attempts to understand the application of these algorithms on quantum hardware. This paper will break down our attempts to replicate the progress made in "Mutual Information Maximizing Quantum Generative Adversarial Network and Its Applications in Finance" (2023) [1] where the major contribution is the development of InfoQGAN. The paper is divided into two main sections: theory and results. In this report, we reviewed the main theory and equations employed in GANs, InfoGANs and the author's novel InfoQGAN. To demonstrate a successful implementation of these the authors utilize 3 different data distributions which we have also studied and replicated the results. Finally, the authors implement an InfoQGAN and apply it to Modern Portfolio Theory.

II. Theory

A. Classical Generative Adversarial Networks

Although the paper does not cover the concept of GANs in depth this brief section is included as it provides clarity of relevant background for the preceding sections. GANs are designed to make two neural networks compete in tandem against each other ultimately improving the behaviour of both in an unsupervised manner. The first of these neural networks is the Generator which attempts to generate convincing replicas of the training data from input noise. The second neural network, the Discriminator, is trained in tandem with the Generator. The Discriminator learns to classify data as real (from the training dataset) or fake (generated by the Generator). The goal of the Generator is to "trick" the Discriminator, and the goal of the Discriminator is to correctly classify the data. In the ideal case, these competing objectives lead to a Generator which can create convincing replicas of input distributions, data, and images, and a discriminator that effectively differentiates real data from generated.

B. Mutual Information & MINE

Mutual information is used to quantize the interdependence of two variables (X and Y) and can be derived from the principles of Shannon Entropy. The specific formulation of the equation for mutual information is the Donsker-Varadhan [2] representation used by Lee et al. 2023 and is as follows:

$$I(X;Y) = H(X) - H(X|Y) = D_{KL}(P_{X,Y}||P_X \otimes P_y)$$

In other words, the mutual information between X and Y can be calculated from the divergence between the true probability density function of X and Y, and the probability density function of X times that of Y. This is because if the two variables are independent then:

$$P_{X,Y} = P_X \otimes P_Y \implies D_{KL}(P_{X,Y}||P_X \otimes P_y) = 0$$

But if they are dependent (i.e. they "share" mutual information) we have that:

$$P_{X,Y} \neq P_X \otimes P_Y \implies D_{KL}(P_{X,Y}||P_X \otimes P_y) > 0$$

Meaning the higher their interdependence is the higher the mutual information function is. Now taking into account the definition of the Shannon Entropy:

$$H(X) = -\Sigma_{x \in X} p(x) \log(p(x))$$

We can then infer the following lower bound for the mutual information where F denotes an arbitrary set of functions T in the parameter space between P and Q density distributions:

$$D_{KL}(\mathbb{P}||\mathbb{Q}) \geq \sup_{T \in \mathcal{F}} \mathbb{E}_{\mathbb{P}}[T] - \log(\mathbb{E}_{\mathbb{Q}}[e^T])$$

With this final expression, it is much easier for us to implement in code since we do not necessarily need to calculate the true probability densities function but instead use a random subset of these functions to get a lower bound. This means we will never attain the "true" mutual information value but if we utilize enough random subsets we will have a robust enough estimate that can be used. In sum, we can leverage MINE to calculate the intrinsic correlation between the generator output with various input parameters, also known as codes.

C. Quantum Generative Adversarial Networks

The behaviour of a QGAN [3] is exactly that of a classical GAN. The primary components follow the same philosophy, with the only major change being that the Generator network is replaced with a quantum Ansatz circuit. More specifically the QGAN takes advantage of states and operations to map an abstract random state onto a data space. The learning process of the algorithm can be modelled by the min-max function of D(x) which is the classifier's evaluation of real data and D(G(x)) which determines the authenticity of the generated data:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$

The QGAN's generator architecture will be further elaborated in section IV.

D. InfoQGAN

Described as a "novel hybrid machine learning model" (Lee et al 2023 [1]) an InfoQGAN [4] is the culmination of all the techniques mentioned above. Essentially extending the GAN to incorporate MINE [5] where a subset of the input noise is used as "codes" that undergo strategic manipulation producing a level of control to the unsupervised output of the generator.

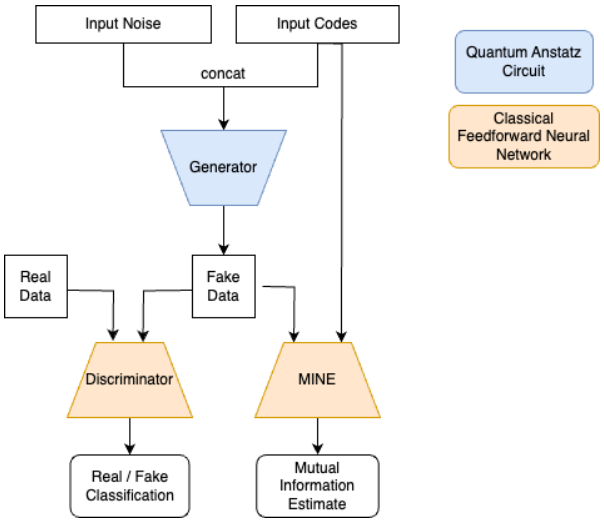


Figure 1. InfoQGAN Architecture

E. Modern Portfolio Theory

Modern Portfolio Theory (MPT) is a quantitative approach for portfolio creation. MPT gives the investor a quantitative measure of the risk versus return relationship between different assets. First, we obtain the returns R_A and volatility σ_A of a given asset A. Then set w_A, w_b to correspond to the weight of asset A and asset B respectively (these values are what the investor can change, ultimately guiding the portion of each asset). Note $w_a + w_b = 1$. We also calculate their covariance: $\rho_{AB} = \mathrm{Cov}(A, B)$. Finally, the volatility of the portfolio is described by:

$$\sigma_P = \sqrt{w_A^2\sigma_A^2 + w_B^2\sigma_B^2 + 2w_A w_B \rho_{AB}}$$

This means that a negative covariance yields a less risky option since if one asset decreases we expect the other to increase minimizing the losses, however, a high positive covariance would simply increase the variation in the portfolio since both assets seem to depend on the same factors. The expected returns for our portfolio can be written as

$$w_A R_A + (1 - w_A) R_B.$$

III. Framework

The paper is primarily a machine learning and data science project so the natural choice for programming language was Python due to its vast ecosystem of data science and machine learning libraries and support. We also decided to implement our classical machine learning models and training infrastructure with PyTorch and data science-related methods with Numpy due to group familiarity and their efficiency among other things. PennyLane was chosen for all quantum-related code due to its tight integrations with Numpy and PyTorch and overall ease of use.

IV. Results and Reproducibility

There are three primary experiments conducted by Lee et al (2023) which we replicated. The experiments all compare the performance of a QGAN and InfoQGAN at generating the following datasets:

(1) Offset circle (Figure 1 a.)

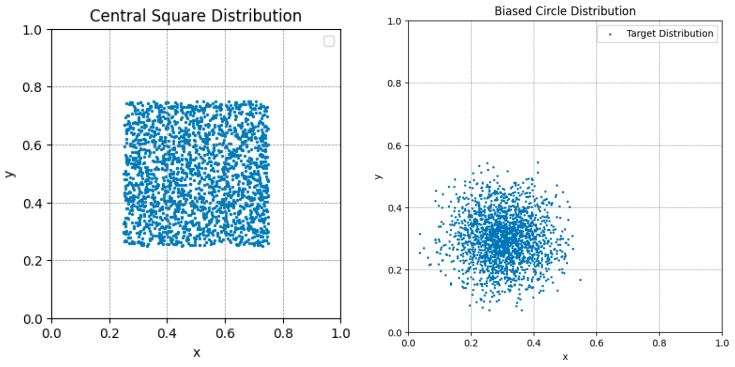
(2) Central square (Figure 1 b.)

(3) Portfolio returns distribution

A major note on the reproducibility of the results from the paper is that we found that the outputs of the neural nets were highly variable. In some runs, we got very similar results to the paper and in others, we did not, even though we used the same architecture. We believe that this is due to the stochastic nature of the input noise and the assumptions we had to make. The following discussion is on runs with outputs similar to the papers.

Preliminary Classical Experiments

Before attempting to implement a QGAN or InfoQGAN we ran experiments generating dataset (2) using a simplified classical GAN and classical InfoGAN. We will briefly outline the results of these experiments in the following section.



(a) (b)

Figure 2: Central square (a) and biased circle (b) distribution used for training the GAN, InfoGAN and InfoQGAN.

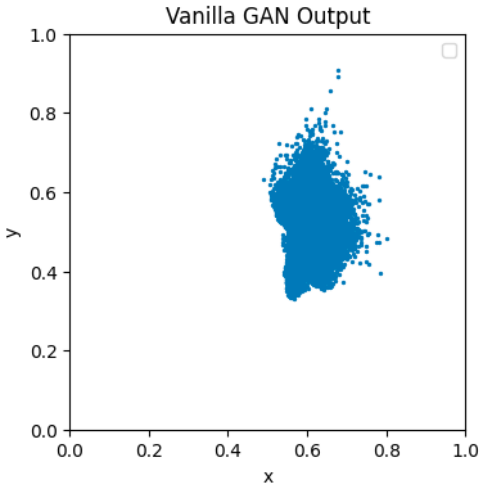


Figure 3: Classical GAN generator output (left) compared to InfoGAN generator output (right)

As is evident in Figure 3, the naive "vanilla," GAN does not have much success replicating the Central Square distribution (2) and we can see signs of mode collapse in Generator outputs. Varying the hyperparameters associated with both generator and discriminator did not lead to a significant change in results, the output of the generator was still not covering the whole range of the square but only a small subset of it thus we proceeded with MINE and InfoGAN.

Our classical InfoGAN showed promising results when generating the central square distribution. In Figure 4 we can see that InfoGAN does a visually good job of generating a central square, and more importantly, the clear separation of features by code values validated

our implementation of MINE which made us confident we could move on to replicating the real bulk of the paper; the InfoQGAN.

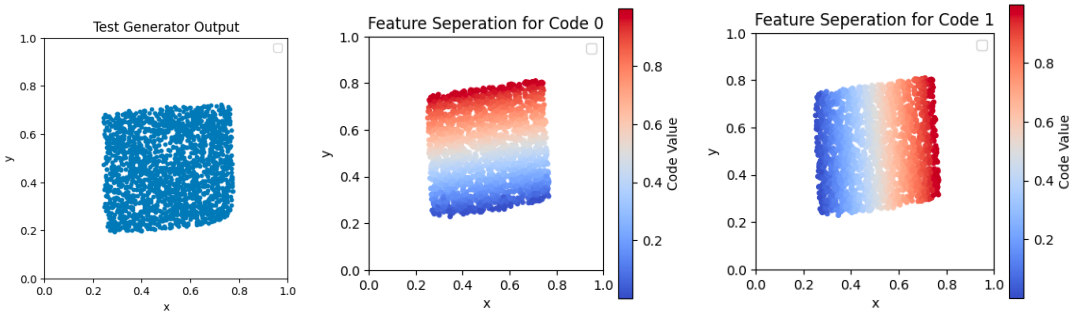


Figure 4: Classical InfoGAN generating the central square and demonstrating good code separation.

Central Square and Biased Circle Generation

Now that we validated our implementation of MINE and a general understanding of GANs, we moved on to replicating the InfoQGAN proposed by Lee et al 2023. The first two simple distributions Lee et al. used to demonstrate the efficacy of their InfoQGAN as opposed to a QGAN are the central square and biased circle distributions shown in Figure 2.

Lee et al. 2023 provided tables laying out hyperparameters used for training these networks, and they also describe the Generator Ansatz and its corresponding activation function. The only assumptions we were required to make here were primarily about the classical neural networks used for the discriminator and MINE estimator. This included the number of layers, layer sizes, and activation functions. We also had to make an educated guess about how exactly they generated the central square and biased circle distributions. We determined the first through several training runs and much trial and error, and we assumed the central square distribution was uniform for simplicity.

The results of our experiments can be seen in the following figures (Note: results of several further training runs in appendix). We also observed much better results from our InfoQGAN as opposed to our QGAN in the same way as Lee et al. 2023. In the case of the central square, the distribution is visually far more accurate and the codes display clear feature separation along orthogonal axes.

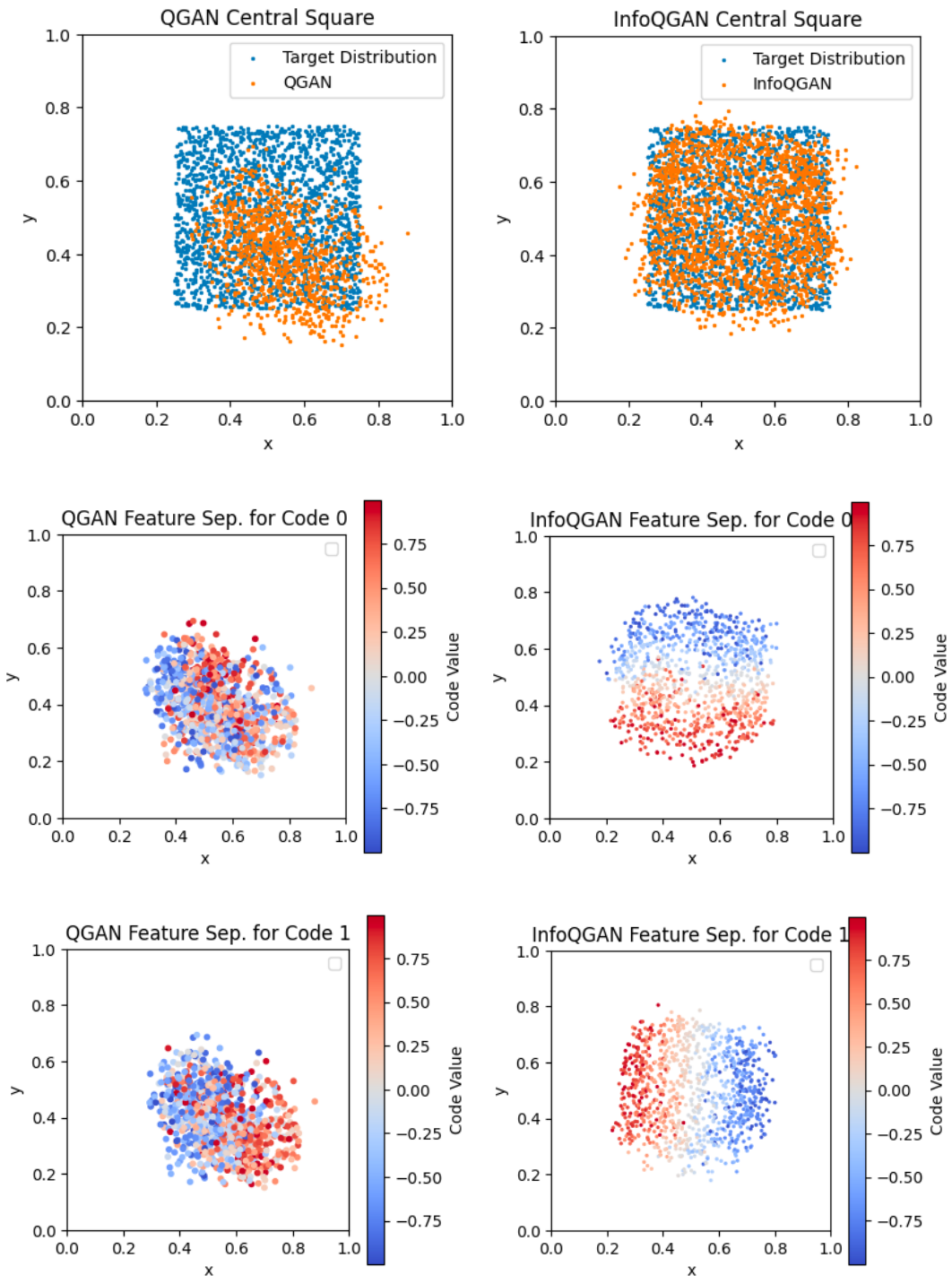


Figure 5: Scatter plot representing the distribution of the generator output using QGAN and the separation of features according to codes. There is no correlation between codes and output. (left)

Figure 6: Scatter plot representing the distribution of the generator output using InfoQGAN and the separation of features according to codes. There is a clear correlation between codes and output. (right)

The below plots also demonstrate how the correlation between codes and the positions on the x and y axes evolved as training progressed.

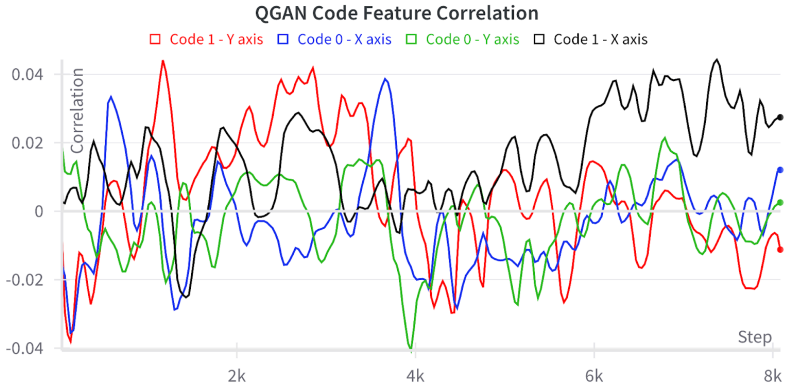


Figure 7: QGAN correlation between x,y-axis and codes 0 and 1

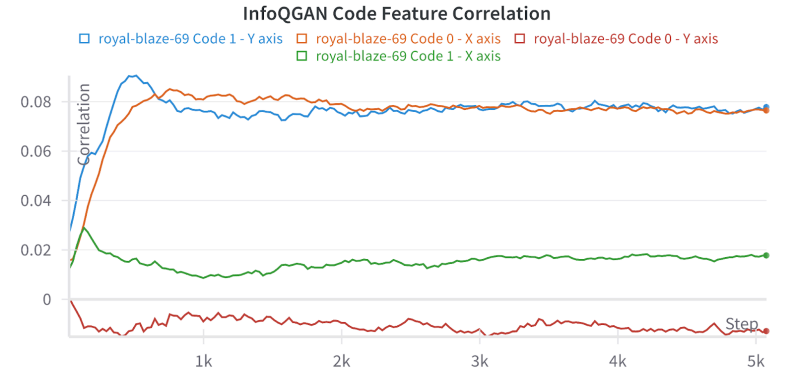


Figure 8: InfoQGAN correlation between x, y -axis and codes 0 and 1

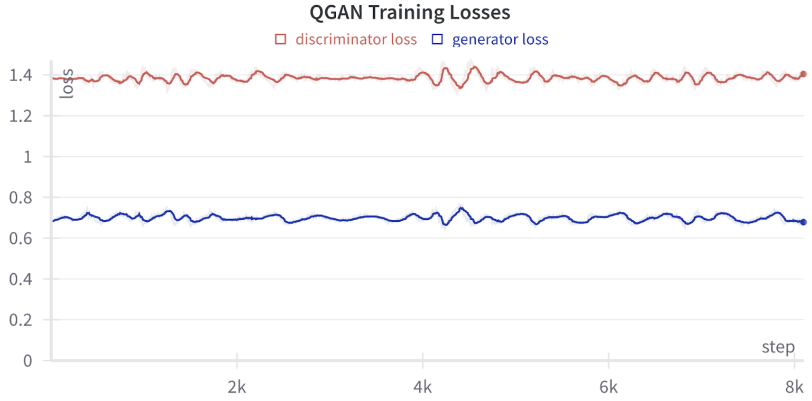


Figure 9: QGAN training loss for both discriminator and generator

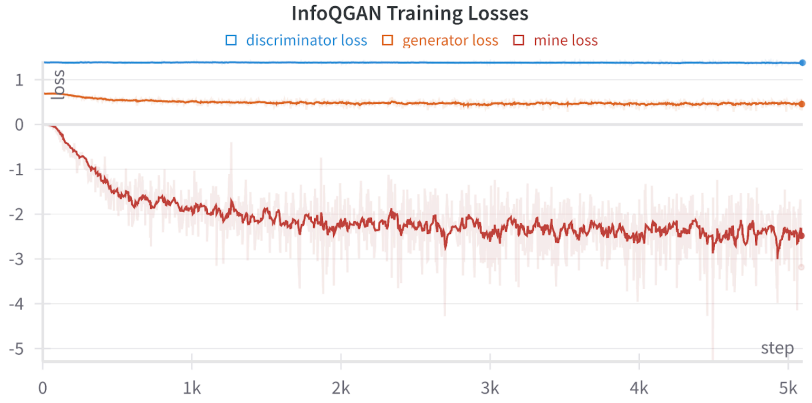


Figure 10: QGAN training loss for both discriminator and generator

To evaluate the accuracy of the generated distributions Lee et al. 2023 employed the two-dimensional Kolmogorov Smirnov Test [6] calculating p and KS values for the generated distributions. Lee et al. 2023 claimed to achieve a p-value of 0.3 and KS value of 0.06 using an InfoQGAN and we also managed to achieve comparable numbers on several training runs (see below figure).

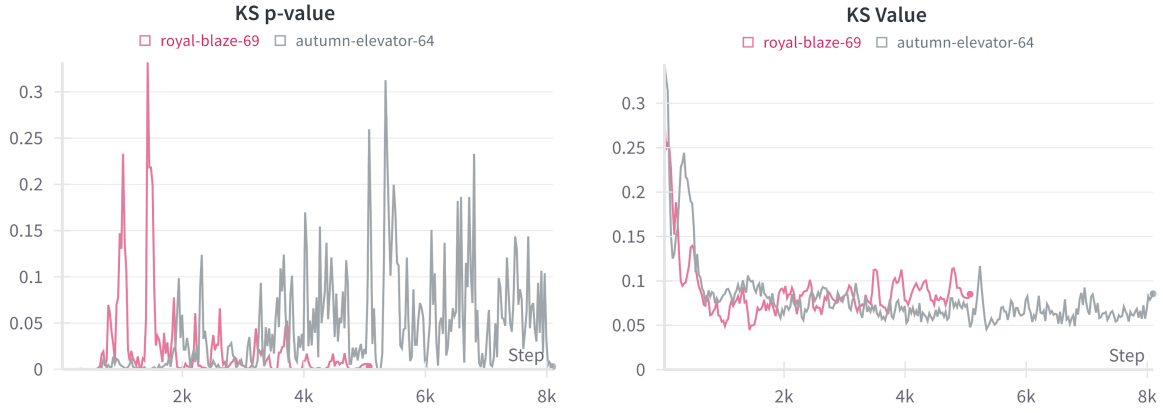


Figure 11: P-values (left) and KS values (right) from InfoQGAN

We also managed to replicate Lee et al (2023)'s generation of the biased circle dataset; plots shown below where we see clear feature separation via codes and an output which is visually convincing when compared to the original distribution.

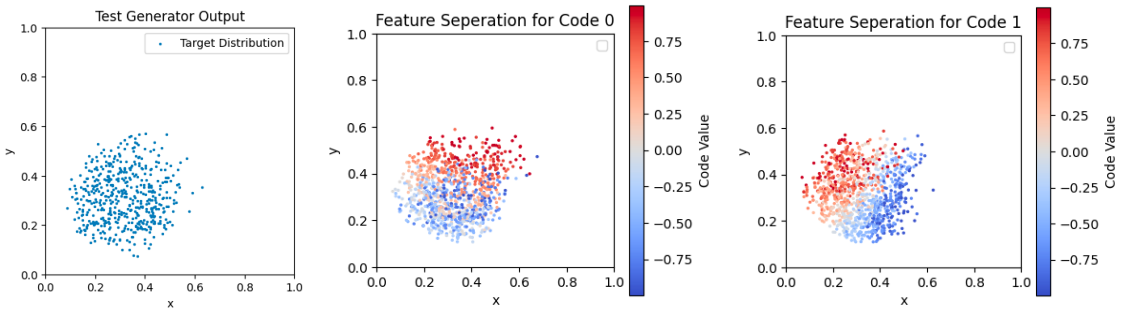


Figure 12: Biassed Circle outputs for InfoQGAN

Below we can also see plots of the QGAN's performance which visually performs worse than the InfoQGAN and does not display any feature separation via codes.

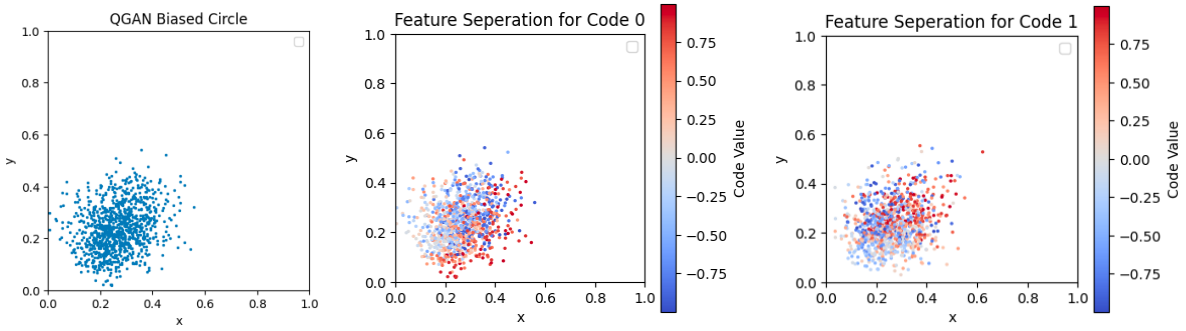


Figure 13: Biassed Circle outputs for QGAN

(3) Portfolio Return Distribution Generation

After assessing that our InfoQGan was functioning as expected through the implementation of the circle and square distribution, we were finally able to tackle the main goal of the paper of generating distributions of portfolio returns composed of different fractions of apple and tesla stock. We were able to use the same training framework as outlined previously for the generation of the square and circle distributions. Following what was done in the paper we changed our ansatz circuit from the one described above to the one shown in Figure 14.

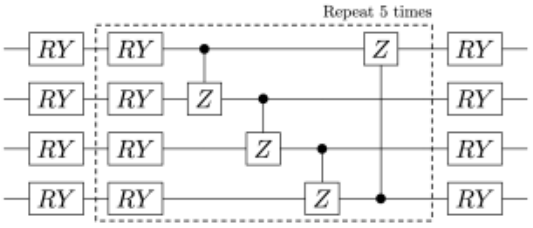


Figure 14: Quantum Ansatz from Lee et al. 2023

We believe that they changed the ansatz construction to simplify the circuit architecture and reduce training time. We also changed the output of the generator to the probability of all 4 wires which lead to 16 outputs. We then needed to generate the dataset to use for training. The paper focused on Tesla and Apple stocks from 2011 to 2022. We then had to generate a discrete distribution of the daily returns for each of the stocks, separating the daily returns into 16 bins in the interval [-0.1,0.1] to match the number of output probabilities from our quantum generator. When we plotted the data we were able to get the exact distribution which was shown in the paper. This result is shown in Figure 15.

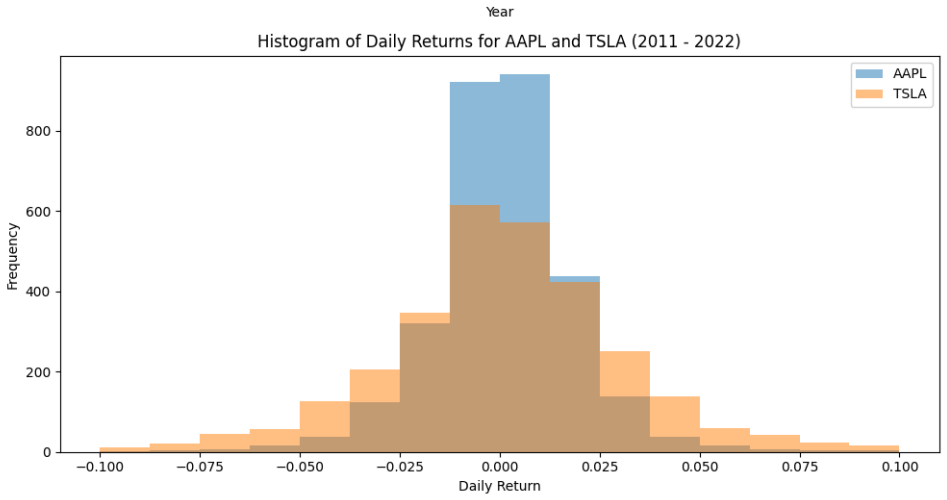


Figure 15: Histogram of APPL/TSLA Daily Returns from YFinance

Finally, we generated the training data set, by following the formula given in the paper for the distribution of assets between the two stocks. We changed the hyperparameters of each model based on what was given in the paper. These hyperparameters are shown in Figure 16:

Experimental Settings			QGAN	InfoQGAN
Assets			APPL, TSLA	
Generator (G)	Learning Rate		0.0004	
	Scheduler	Optimizer	Adam	
		Step Size	30	
		Gamma	0.7	
Discriminator (D)	Learning Rate		0.00004	
	Scheduler	Optimizer	Adam	
		Step Size	30	
		Gamma	0.7	
Mutual Information Neural Estimator (MINE)	Learning Rate		-	0.001
	Scheduler	Optimizer		Adam
		Step Size		30
		Gamma		0.7
Training Epoch			450	
Layers (Parameters)			5 (24)	
Qubits	Noise (ϵ)		4	3
	Code (c)		-	1
β in Loss Function (Equation (11))			-	0.15

Figure 16: Hyperparameters for MPT Experiment

Up to this point, we did not make any assumptions, all the parameters were taken from the paper. The first assumption we had to make was the size of the two classical models, the paper never mentioned the layer types, numbers or sizes so it varied until we got models which looked more similar to the papers. We also found that with the discriminator learning rate given by the paper, the generator loss would diverge and the model would never train properly. We ended up having to up the learning rate of the discriminator to be the same as that of the generator to get results which more closely aligned with the paper. We think this learning rate may even have been a typo, given how poorly it performed and the fact that we found other typos in the paper. After making this change and trying different layer sizes for the Discriminator and T models, we were able to get results which gave similar correlations as those found in the paper, but the generated probability distributions themselves were slightly different as shown in Figure 18. The difference between Figure 17 and 18 shows how our model converged to the same correlation after the same number of epochs used in the paper.

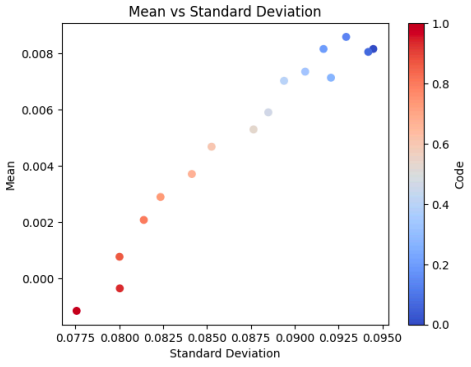


Figure 17: Correlation between mean and standard deviation after 150 epochs gives a different correlation than that found in Lee et al. 2023 after 450 epochs shown in Figure 18.

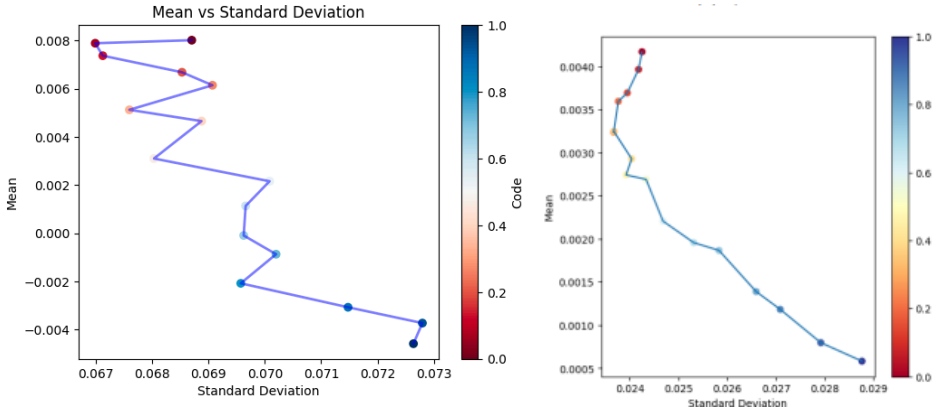


Figure 18: Correlation between mean and standard deviation after 450 epochs ours (left) vs Lee et al. 2023 (right)

As in Lee et al. 2023 we also generated a Correlation chart for the normal QGAN and found that the codes had no clear correlation. Proving their theory that an InfoQGAN leads to more control over the output.

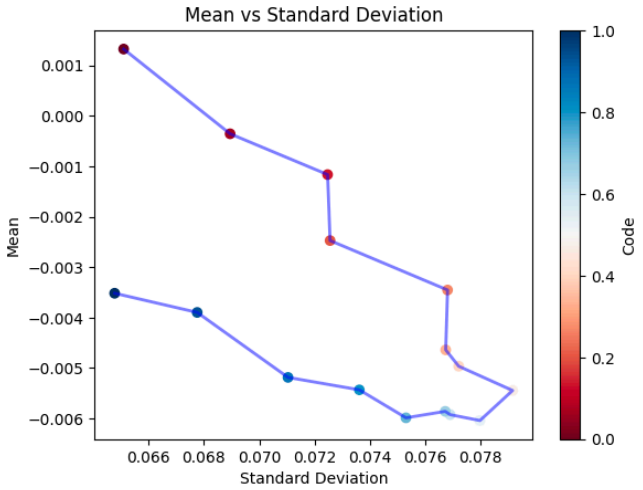


Figure 19: Correlation between mean and standard deviation for our QGAN model

A clear difference that can be observed in Figure 19 is that our model, although it had a similar correlation, had a wider spread over the values for the means and standard deviations. We think this most likely has to do with differences in the architecture of our classical models with respect to those used in the paper. Although these results are not the same they may be better. The paper describes that they wanted a wider spread in the InfoQGAN with respect to the QGan as it showed better resilience to mode collapse leading to more diverse exploration of the output space. This result is also apparent when comparing the generation of probability distributions with 16 code values evenly spaced between 0 and 1. We followed the generation of these plots as described in the paper averaging 512 generated distributions for each code value. The comparison is given in Figures 20 and 21. As we can see with our plots the difference in distributions as we change the code is much more apparent than those found in the paper. This seems to represent the model learning to create a more varied distribution of assets based on the codes.

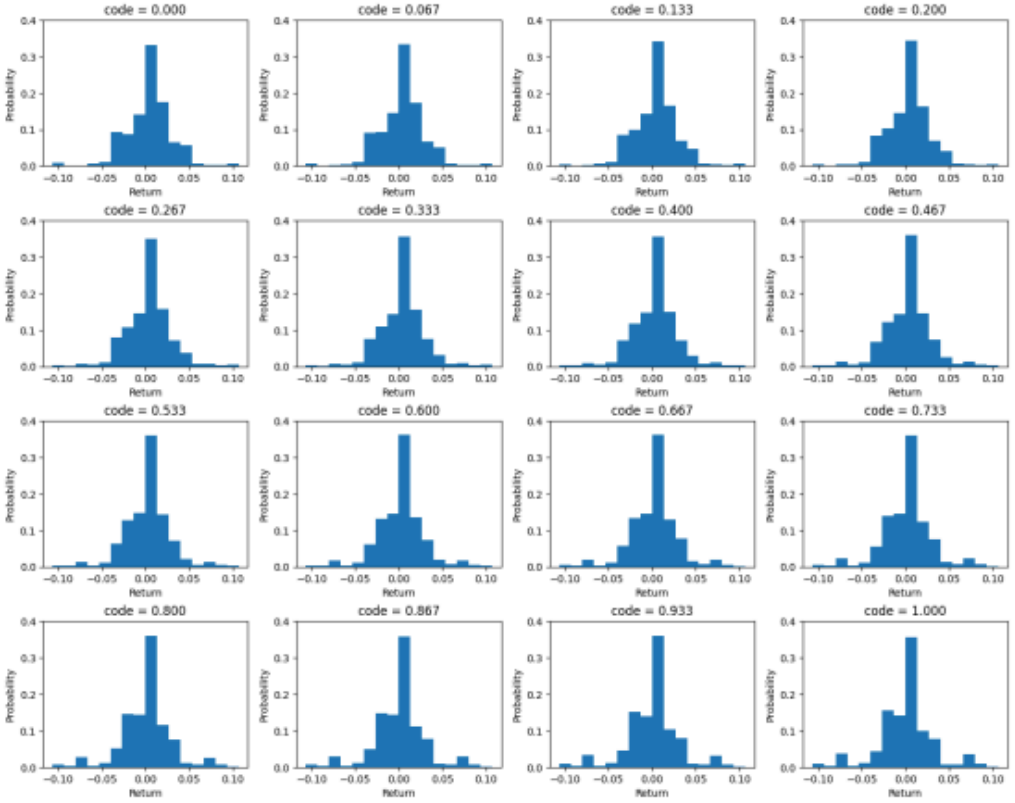


Figure 20: Lee et al. 2023 Plots

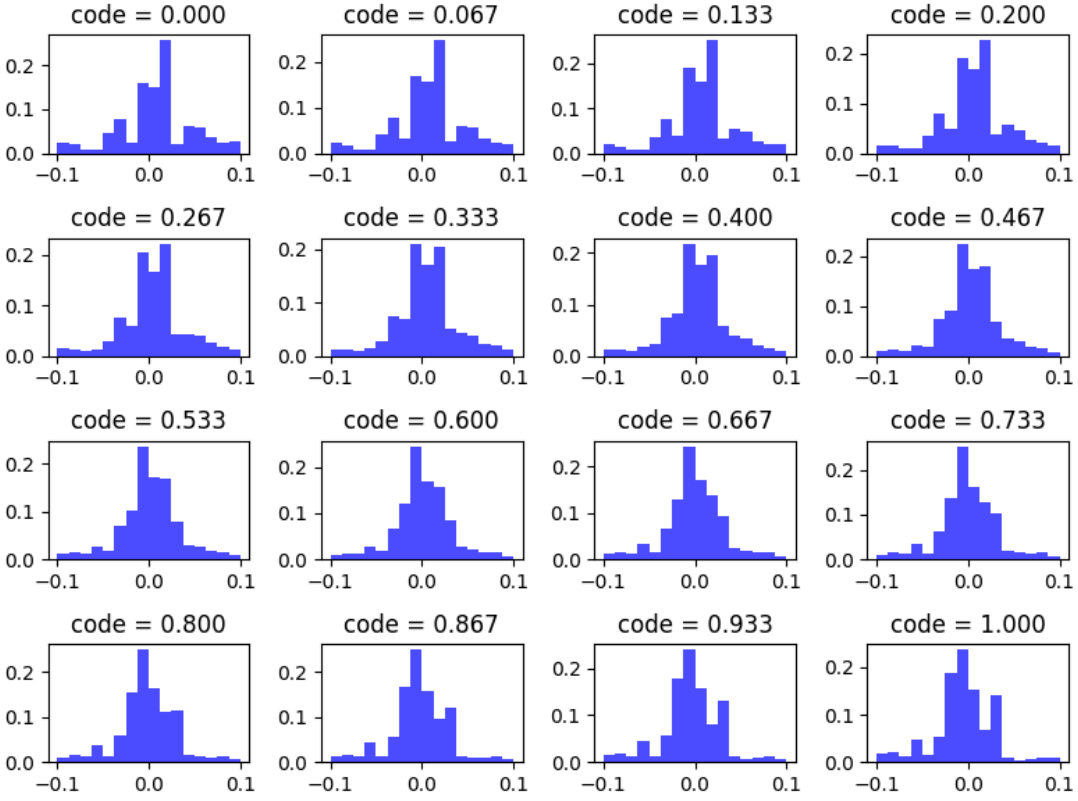


Figure 21: Plots from our neural net using 16 codes between 0 and 1.

V. Opinion and Interpretation

Overall the paper did a good job of laying out how they accomplished what they did and we were able to replicate, and in several cases improve upon their stated results. They were clear about what software components were done using quantum computing and didn't claim any level of quantum supremacy. This paper was a simple example of how quantum computing could be used in the field of machine learning. Although the generator was a quantum circuit they did not give any good reasoning for it being better than a classical system. We found that by replicating the paper using purely classical models we achieved much better results, with much faster training speeds than when using a quantum generator. The paper also doesn't leverage any quantum advantages when using an ansatz network instead of a classical neural net. They just replace the classical network with a quantum one, but don't restructure the rest of the code to take this into account. They discuss possible optimizations to be explored when using a quantum generator but don't apply any of them. They would have made a much larger contribution to the field if they could have shown how a quantum algorithm could have been applied along with the quantum ansatz to gain a measurable speed-up compared to the classical model.

Bibliography

1. Lee, M., Shin, M., Lee, J., Jeong, K. (2023). *Mutual Information Maximizing Quantum Generative Adversarial Network and Its Applications in Finance.* Arxiv. https://arxiv.org/pdf/2309.01363.pdf
2. Von Neumann, J. (2013) Mathematische grundlagen der quantenmechanik, vol. 38 Springer-Verlag.
3. Dallaire-Demers, P.L., Killoran N. (2018), Physical Review A 98, 012324.
4. Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., Abbeel, P., (2016). Advances in neural information processing systems 29.
5. Belghazi,M. I., Baratin, A., Rajeswar, S., Ozair, S., Bengio, Y., Courville, A., Hjelm, R. D. (2018). arXiv preprint arXiv:1801.04062

6. J. A. Peacock, Monthly Notices of the Royal Astronomical Society 202, 615 (1983)