Modelling financial time-series data using a diffusion model.

dedges@oregonstate.edu**

Deep Learning's recent success in the domain of images and text has not yet translated in the financial domain. Empirical evidence has shown that financial data of indexes show "long" position on a large enough time scale but Brownian motion for smaller time ranges. That is why deep learning models built on historical data aren't quite reliable as the assumption of a long market does not contextualise current market conditions, whereas, models built with relatively recent data that is representative of current market conditions suffer from a lack of coverage from the dataset. Thus, modelling the distribution of financial time-series data is an important topic of research in quantitative finance. The ability to generate financial time-series data can robustify regression datasets as well as reveal stylized characteristics of an underlying asset to further use in problems like detection of market manipulation. In this project, a diffusion based approach namely TabDDPM is used to model and generate SPY stock and options time-series data from its distribution. The diffusion based approach is compared with QuantGAN, a generative adversarial method specifically designed for modelling sequential data. TabDDPM shows comparable results to the GAN based model and provides a flexible alternative to model financial time-series data.

1. Introduction

The financial market is a platform where humans with wants and needs along with their biases trade some assets with another human having their own wants and needs stemming from their own biases resulting in a complex intertwined system. As pointed out by Devi(2021)[3], this complex system of interacting biassed humans sometimes leads to chaotic behaviour resulting in wild swings for asset prices, driving it away from equilibrium and into the realm of perceived randomness. That is why modelling the distribution of an underlying asset is an important step in realising anomalies arising from market manipulation. Limitations like noisy and non-stationary information mentioned by Gupta et al.(2019)[4] lead to the motivation behind generating samples from financial time-series distribution to augment the capabilities for regression applications in finance.

After the formal introduction of denoising diffusion probabilistic models, a class of latent variable models, by Ho et al.(2020)[5] and the subsequent modifications by Nichol and Dhariwal(2021)[6], these models have garnered immense popularity in generative tasks. Diffusion models, for short, are state of the art for image(Nichol & Dhariwal, 2022)[7] and video(Ho et al., 2022)[8] generation. For text generation, Gong et al.(2022)[10] report comparable results to other autoregressive and non-autoregressive methods but outperforms all methods in sampling data points with high diversity. The Diffusion-Language Model by Li et al.(2022)[9] has better scores compared to other non-autoregressive methods while Strudel et al.(2022)[11] contributes towards formalising the scalability of these models. Although Tashiro et al.(2021)[12] adopt the diffusion approach for the imputation problem, there haven't been any attempts to model the whole financial time-series distribution using diffusion.

** https://github.com/sam-dedge/sim-stock-options

In this article, I will give a background about the relevant financial instruments and their terminologies used for modelling the distributions. Then, I will introduce the notion of neural networks leading to adversarial and diffusion models and the formal introduction to the network TabDDPM(Kotelnikov et al., 2022)[1], a diffusion methodology used to model structured data. The findings from TabDDPM will be compared with a baseline, QuantGAN(Wiese et al., 2020)[2]. My main contribution from this project is to investigate the diffusion approach on financial time-series data as a viable alternative to generative adversarial models. I compare the generated distributions using Wasserstein distance, Kolmogorov–Smirnov test and ACF score along with identifying the presence or absence of the stylized characteristics of assets in the financial data.

2. Background

There are different financial markets where different types of instruments are traded. From governmental bonds issued by the sovereign treasuries to global commodity markets; financial time-series data is available in multiple formats corresponding to multiple different instruments. In this project, I am going to focus on the capital markets relating to stocks and stock options of an underlying asset. Stocks are tokens of publicly traded companies which gives the holder part ownership of the organisation. As a reward for their investment, the common stockholder cannot get a slice directly from the revenue or profits of the company but have to sell their share of the ownership at market price to get a return from the increased valuation of the company. Thus, distribution of a stock is modelled after the relative return and not the price of the stock. The return is given by -

$$r_t = \frac{S_t - S_{t-1}}{S_{t-1}}$$

where r_t is the return at time t and S_t is the spot price of the stock at time t.
Toth and Jones(2019)[13] argue that modelling returns based on the Laplace distribution is a better approximation than the commonly used normal distribution. That is why I use probability density function of Laplace to fit the models given by the random variable X where -

$$f(x) = \frac{1}{2\lambda} e^{-\frac{|x-\mu|}{\lambda}},$$

μ is the centre of distribution and λ is the scale of the distribution.

As seen from Fig 1., Laplace distribution is able to fit the stylized fact of peaked and heavy tailed distribution of asset returns better than normal distribution. “Stylized facts” are characteristics of asset returns observed through historical empirical evidence. Wiese et al.(2020)[2] summarises some of the important stylized facts from Chakraborti et al.(2011)[14] and Cont(2001)[15] :

- heavier tails are observed from asset returns,
- the distribution of asset returns is more peaked than the gaussian/normal distribution,
- Volatility clustering is the observation that large changes in price tend to follow large changes and small changes tend to be followed by small changes. Volatility displays a positive autocorrelation over several days, which quantifies the fact that high-volatility or low-volatility events tend to cluster in time,
- the volatility of an asset is negatively correlated with the return of the asset, an effect named leverage effect,

- empirical asset returns are considered to be uncorrelated but not independent.

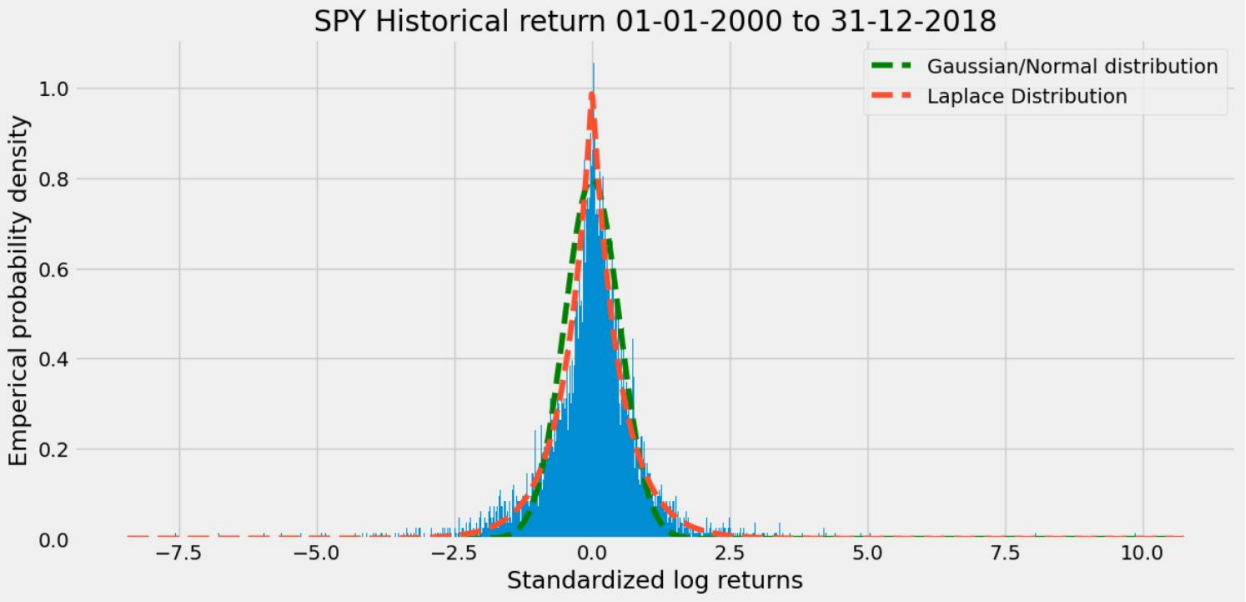


Fig 1. Historical returns' distribution of SPY ETF cements Toth and Jones'(2019)[13] findings that the returns follow Laplace distribution more closely than Gaussian distribution.

Options are financial instruments that allow 2 parties to be in a contract with each other where one party can buy/sell an underlying asset from/to the other party for an agreed upon fixed price on or before a fixed future date. The option contract gives the buyer, in exchange for paying the option premium, the right(but not the obligation) to buy/sell the underlying asset at a specific price on or before a specific date. The option contract gives the seller, in exchange for receiving the option premium, the potential obligation to sell/purchase the underlying asset at a specific price on or before a specific date. There are 2 types of options - Calls and Puts.

A call option gives the option holder/buyer the right to buy the underlying asset at the strike price(specified price) on or before the expiration date. Thus, if the spot price(present market price) of the underlying is more than the strike, then the buyer makes a profit, else the seller makes a profit by gaining the option premium. The return of calls where C_b is the return for a buyer and C_s is the return for a seller is -

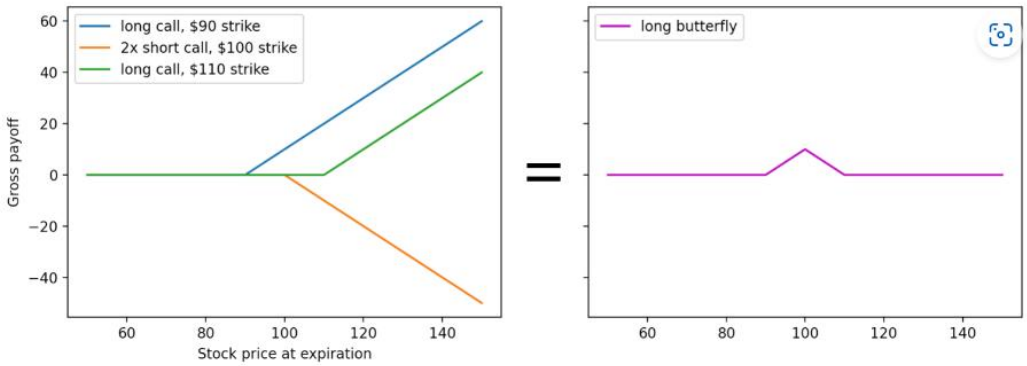
$$C_b = - \text{Premium} + \text{Max}(\text{Spot}-\text{Strike},0)$$
$$C_s = + \text{Premium} - \text{Max}(\text{Spot}-\text{Strike},0)$$

On the other hand, a put option gives the option holder/buyer the right to sell the underlying asset at the strike price(specified price) on or before the expiration date. Thus, if the spot price(present market price) of the underlying is less than the strike, then the buyer makes a profit, else the seller makes a profit by gaining the option premium. The return of puts where P_b is the return for a buyer and P_s is the return for a seller is -

$$P_b = - \text{Premium} + \text{Max}(\text{Strike}-\text{Spot},0)$$
$$P_s = + \text{Premium} - \text{Max}(\text{Strike}-\text{Spot},0)$$

The risk-neutral distribution of options is modelled after the expected payoff by creating a butterfly spread for adjacent strikes(Martinez, 2022)[28]. A butterfly spread(See Fig 2) is an

artificial exposure to the asset by buying a call of a low strike price, selling 2 calls of a high strike price and buying another call of an even higher strike price.



(Fig 2A) (Fig 2B)

Fig 2A. Shows the payoffs of the individual option contracts in order to create a butterfly spread.[30]
Fig 2B. Shows the cumulative payoff of the whole position for different spot prices of the stock at expiration.[30]

The cost of the option that the buyer paid for is its value and is given by its expected payoff. Therefore, the value of an option is given by $C_{(K,\tau)}$ where K is the strike and $\tau = T - t$ where t is now and T is the expiry of the option.

$$C_{(K,\tau)} = E(max[S_T - K], 0)$$

$$C_{(K,\tau)} = p(S_T)\Delta K$$

where $p(S_T)$ is the probability of the spot price at time T and ΔK is the maximum payoff for the butterfly spread. Thus, probability is given by -

$$p(S_T) = \frac{C_{(K,\tau)}}{\Delta K}$$

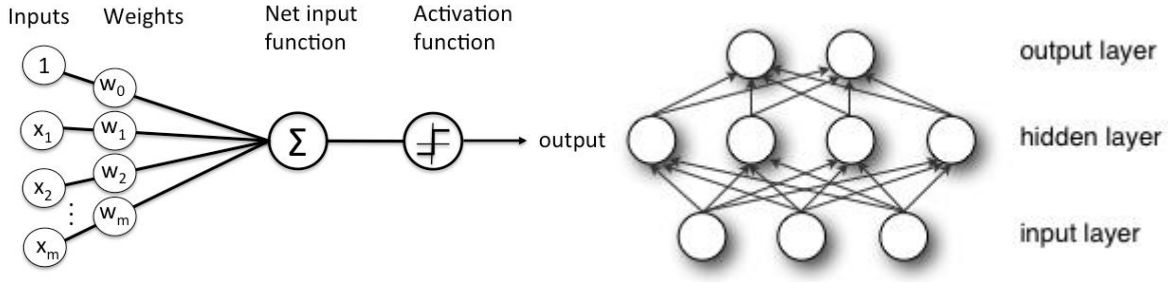
3. Neural Networks

Neural networks are computational models, inspired from the human brain, that can learn complex patterns from data by adjusting their weights based on the “wrongness” or loss of the model. They consist of nodes(See Fig 3A) arranged in layers that receive input from the previous layer, perform nonlinear transformations on their inputs and pass the output to the next layer(See Fig 3B). Neural networks can be trained for a wide variety of tasks, such as classification, regression, generation, and reinforcement learning.

Let N_0 & N_{L+1} be the dimension of input & output respectively where L is the number of hidden layers of the network whose dimensions are given by $N_{\{1, \dots, L\} \in N}$. Let Φ be a non linear activation function like a step or sigmoid function, Θ be a Euclidean vector space for any $l \in \{1, \dots, L\}$ and $a_l : R^{N_{l-1}} \to R^{N_l}$ be the affine function which takes the form $a_l : x \to W^{(l)} + b^{(l)}$, where weight matrix $W^{(l)} \in R^{N_l \times N_{l-1}}$ and $b^{(l)} \in R^{N_l}$.
Thus, if it is Lipschitz continuous, the network function $f: R^{N_0} \times \Theta \to R^{N_{L+1}}$ with parameter space Θ is given by -

$$f(x, \theta) = a_{L+1} \odot f_L \odot \dots \odot f_1(x) \qquad (1)$$

where $\odot$ is a composition operator and $f_l = \phi \odot a_l$, with the learnable parameters of the network given by $\theta := (W^{(1)}, \ldots, W^{(L+1)}, b^{(1)}, \ldots, b^{(L+1)}) \in \Theta$.



(Fig 3A) (Fig 3B)

Fig 3A. Generic representation of the computational unit called perceptron used in neural networks. [31]
Fig 3B. Layer structure of nodes forming a multilayer perceptron or neural network. [31]

One of the challenges of neural network training is to generate realistic and diverse samples from complex data distributions such as images, text and speech, as well as structured data. This is known as generative modelling which has many applications in computer vision, natural language processing, wave synthesis and data augmentation; data augmentation of financial time-series data being the primary goal of the project. Generative modelling can also help understand the underlying structure and latent factors of the data which is important when analysing financial assets.

Generative modelling paradigm is divided into 4 main classes of models - variational autoencoders(VAE), generative adversarial networks(GAN), diffusion models and transformers. According to Dogaria et al.(2022)[16], variational autoencoders seem to outperform GANs when producing synthetic financial time-series data but fail to not only provide code but also details about their architecture to replicate the results. Fu et al.(2022)[17] provides a transformer based GAN that beats QuantGAN but also does not provide any code to verify the findings. I adopt the idea of attention from Fu et al.(2022)[17] for the diffusion framework while modelling options and report the efficacy with and without the mechanism. In the following section, I will talk about GANs and diffusion models and the specific architectures used to model the data.

A. Generative Adversarial Networks

After their introduction by Goodfellow et al.(2020)[18], generative adversarial networks or GANs for short have been quite popular for generative modelling in various domains like image, text or speech synthesis. The framework of GANs consists of two networks, a generative model pitted against a discriminative model. The discriminator learns to determine whether a sample is from the real distribution or generated distribution whereas the generator learns to fool the discriminator by generating samples resembling true distribution. Goodfellow et al.(2020)[18] notes this adversarial framework of competition, that is analogues to police trying to catch counterfeit currency while the counterfeiters try to produce fake currency without being detected, should improve both their methodologies in practice.

The generator's distribution p_g over data x is learned by mapping the prior distribution over some noise $p_z(z)$ to data space $G(z; \theta_g)$ where G is a differentiable function represented by a

network with parameters θ_g. The network of discriminators is defined by $D(x; \theta_d)$ that outputs a single scalar where D(x) represents the probability that x came from the real data rather than p_g. The discriminator D is trained to maximise the probability of assigning the correct label to both real samples and generated samples from G. The generator G is simultaneously trained to minimise log(1-D(G(z)). Thus, the value function for the zero-sum game played by the discriminator and the generator called the GAN objective is given by -

$$min_G\ max_D\ V(D, G) = E_{x\sim p_{data}(x)}[logD(x)] + E_{z\sim p_z(z)}[log(1 - D(G(z)))] \quad (2)$$

For the particular case of financial time series data, Wiese et al.(2020)[2] propose a modification to the GAN architecture by replacing the multilayer perceptron formulated in (1) by temporal convolutional networks for modelling the sequence. Bai, Kolter and Koltun(2018)[19] give 2 principles for temporal convolutional networks (TCNs) that are: a) the network should produce an output that is the same length as the input. b) there can be no future leakage in the past sequence. To achieve the above principles, TCN uses a combination of a 1D fully convolutional network (FCN) and causal convolutions(See Fig. 4A) where an output at time t is only convolved with elements from time t and before of the previous layer. Causal convolutions by themselves suffer from needing extremely deep networks or very large filters to cover the effective past of the sequence which is not very feasible. van den Oord et al.(2016)[20] give dilated convolutions or convolutions with holes so as to overcome the problem of needing a large receptive field for longer sequences. Dilated causal convolutions(See Fig. 4B) capture a larger receptive field than simple causal convolutions making it an important tool in sequence modelling. TCNs made up of diluted causal convolutions are used by Wiese et al.(2020)[2] as their neural networks for the discriminator and generator and the modified GAN objective from (2) becomes -

$$min_G\ max_D\ V(D, G) = E_{x\sim p_{data}(x)}[logD(x_{1:T^{(d)}})] + E_{z\sim p_z(z)}[log(1 - D(G(z)_{1:T^{(d)}}))] \quad (3)$$

where $D(x_{1:T^{(d)}})$ and $D(G(z)_{1:T^{(d)}})$ are real and generated sequences, respectively.

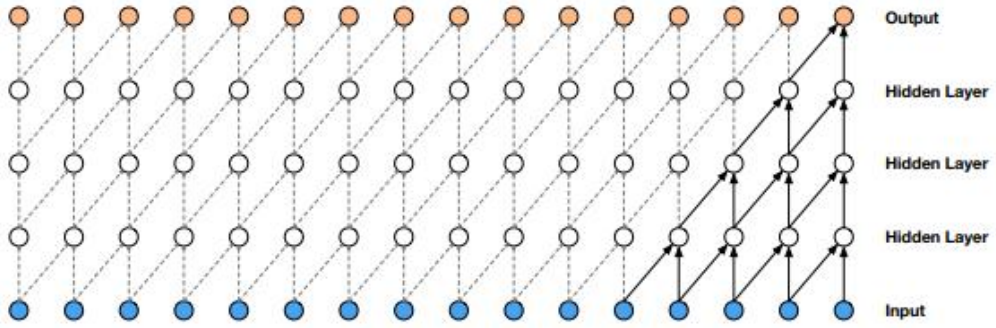


Fig 4A. Visualisation of a stack of causal convolutional layers with filter=2.

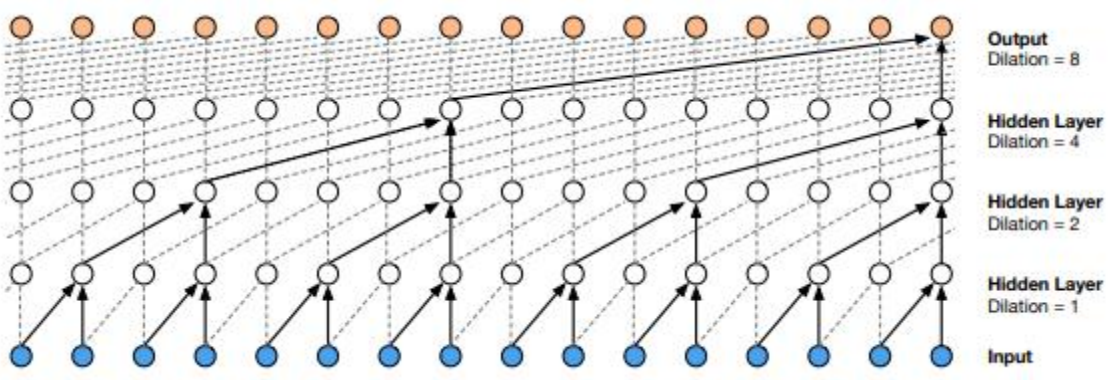


Fig 4B. Visualisation of a stack of dilated causal convolutional layers with filter=2.

The original architecture only models the returns by taking in the input with dimension 1 but I modify it to input multiple features so that other features' distributive comparison can also be done.

B. Denoising Diffusion Probabilistic Model

Inspired from the field of nonequilibrium thermodynamics, denoising diffusion probabilistic models or simply diffusion models are a class of latent variable generative models that are parameterised Markov chains trained using variational inference to produce samples matching data after a finite time (Ho et al., 2020)[5]. Yang et al.(2022)[21] describe the models as two Markov chains, a forward chain that destroys the data to noise and a reverse chain that converts noise back to data. The forward process is given by $q(x_i \mid x_{i-1})$ which uses data sample x_0 to gradually add Gaussian noise to produce latent noisy samples $x_1, \dots, x_T$ whereas the reverse process is given by $p_\theta(x_{i-1}|x_i)$ which gradually removes the noise to generate the target distribution(Hernandez & Dumas, 2022)[22].

Given a data distribution to be modelled $q(x_0)$, a forward process q is defined such that it adds noise to an initial sample x_0 to get a latent x_T that is from a nearly isotropic Gaussian distribution, where the noise sampled is from a gaussian distribution (Nichol & Dhariwal, 2021)[6] or from a predefined distribution like $q(x_t|x_{t-1})$ with variances $\{\beta_1, \dots, \beta_T\}$.

$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1}) \quad (3)$$

The reverse process p is defined such that it gradually denoises from a latent variable $x_T \sim q(x_T)$ that allows generating new data samples of $q(x_0)$. Because the distribution $q(x_0)$ is usually unknown, the reverse process is approximated using a neural network that is parameterised by θ.

$$p_\theta(x_{0:T}) = \prod_{t=1}^{T} p(x_{t-1}|x_t) \quad (4)$$

a. Gaussian diffusion models for numerical features

Forward and reverse process in models in continuous space ($x_t \in R^n$) are given as -

$$q(x_t|x_{t-1}) := N(x_t; \sqrt{1-\beta}x_{t-1}, \beta_t I) \quad (5)$$

$$q(x_T) := N(x_T; 0, I) \quad (6)$$

$$p_\theta(x_{t-1}|x_t) := N(x_{t-1}, \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)) \quad (7)$$

where Ho et al.(2020)[5] suggests using a diagonal $\Sigma_\theta(x_t,t)$ with a constant σ_t and $\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{\beta_t}{\sqrt{1-\underline{\alpha_t}}}\epsilon(x_t, t))$ where $\alpha_t = 1 - \beta_t$, $\underline{\alpha_t} = \prod_{i \le t} \alpha_i$ and $\epsilon_\theta(x_t, t)$ predicts "groundtruth" noise component ϵ for the noisy data sample x_t.

The diffusion training in a continuous space is done by optimising on the negative log likelihood:

$$log\ q(x_0) \geq E_{q(x_0)}[log\ p_\theta(x_0|x_1) - KL(q(x_T|x_0)|q(x_t)) - \sum_{t=2}^{T} KL(q(x_{t-1}|x_t, x_0)|p_\theta(x_{t-1}|x_t))] \quad (8)$$

Ho et al.(2020)[5] simplifies objective (8) and gives the new objective to be optimised as -

$$L_{simple}(\theta) := E_{t,x_0,\epsilon}[||\epsilon - \epsilon_\theta(\sqrt{\underline{\alpha_t}}x_0 + \sqrt{1-\underline{\alpha_t}}\epsilon, t||^2] \quad (9)$$

b. Multinomial diffusion model for categorical features

Multinomial diffusion models are a type of diffusion models introduced by Hoogeboom et al.(2021)[23] that are specifically designed to generate categorical data where $x_t \in \{0,1\}^K$ is a one-hot encoding of a categorical variable with K classes. Kotelnikov et al.(2022)[1] give a more understandable form of the multinomial forward diffusion process defined by $q(x_t|x_{t-1})$ -

$$q(x_t|x_{t-1}) := Cat(x_t; (1-\beta_t)x_{t-1} + \beta_t/K) \quad (10)$$

$$q(x_T) := Cat(x_t; 1/K) \quad (11)$$

$$q(x_t|x_0) := Cat(x_t; \underline{\alpha_t}x_0 + (1-\underline{\alpha_t})/K) \quad (12)$$

Thus, the posterior can be computed in a closed form from (11) and (12) -

$$q(x_{t-1}|x_t, x_0) := Cat(x_{t-1}; \pi/\sum_{k=1}^{K} \pi_k) \quad (13)$$

where $\pi = [\alpha_t x_t + (1-\alpha_t)/K] \odot [\underline{\alpha}_{t-1}x_0 + (1-\underline{\alpha}_{t-1})/K]$.

The reverse distribution $p_\theta(x_{t-1}|x_t)$ is parameterised as $q(x_{t-1}|x_t, \hat{x}_0(x_t, t))$ where $\hat{x}_0$ is the prediction by the neural network trained to maximise the variational lower bound in (8).

c. TabDDPM for structured data

The TabDDPM model by Kotelnikov et al.(2022)[1] combines the multinomial diffusion process for categorical and binary features and the Gaussian diffusion model for numerical ones. The model takes the input x_0 of dimensionality

$$(N_{num} + \sum_{i=1}^{x_{cat}} K_i), \quad (14)$$

were N_{num} consists of the numerical features $x_{num} \in R^{N_{num}}$ and K_i are the number of categories each from x_{cat} total categorical features. Numerical features are inputted after normalising while, categorical features are inputted as one hot encodings $(i.e\ x_{cat_i}^{ohe} \in \{0,1\}^{K_i})$ to the model resulting in the dimensionality of (14). Noise is sampled independently for each categorical feature for the forward diffusion process whereas the numerical features are transformed using quantile transformation from scikit-learn library (Pedregosa et al., 2011)[24]. The parameterised reverse diffusion process is learned by a fully connected neural network that has the output of the same dimensionality as x_0.

4. Evaluation

Generated financial time series data should be analysed quantitatively as well as qualitatively.

Qualitative analysis involves the evaluation of the stylized characteristics mentioned above whereas quantitative analysis involves statistical and numerical results to compare the performance of the generated distribution. Financial data generation is a relatively new idea where evaluation metrics have not yet been established, as is the case with other domains like image synthesis where Inception Score(IS) by Salimans et al.(2016)[25] and Frechet Inception Distance(FID) by Heusal et al.(2017)[26] are standard. Evaluation metrics designed for image synthesis cannot be used for the evaluation of financial times-series but signal processing metrics can be used. Thus, like Dogariu et al.(2022)[16], Wiese et al.(2020)[2] and Fu et al.(2022)[17], I perform quantitative analysis from metrics inspired from signal processing problems. The comparative analysis is performed between QuantGAN by Wiese et al.(2020)[2] and TabDDPM by Kotelnikov et al.(2022)[1]. Wiese et al.(2020)[2] architecture implemented by Sullivan(2021)[32] generates multiple return paths to recreate the results. For simplicity purposes, I adapt the code and randomly choose a return path from the generated paths for evaluation.

a. Quantitative Analysis

I adopt popular methods for evaluating distributions in line with other financial data generation papers.

i. Wasserstein-1 Distance : Levina and Bickel(2001)[27] gave a rigorous probabilistic interpretation of the Earth Mover's Distance(EMD) or the first Wasserstein distance which essentially gives a distance between two 1D distributions. Simply, it is the measure of "work" required to reconfigure or move the probability mass of one distribution to another. EMD between distribution u and distribution v is given by -

$$E(u, v) = inf_{\pi \in \Gamma(u,v)} \int_{RxR} |x - y| d\pi(x, y) \quad (16)$$

where $\Gamma(u, v)$is the set of probability distributions on RxR with marginals u and v.

ii. Kolmogorov-Smirnov Test (Massey Jr, 1951)[29]: K-S test is a test that quantifies the distance between the empirical distribution of the sample and the cumulative distribution of the reference distribution (one-sample K-S test), or the distance between empirical distributions of two samples (two-sample K-S test). The test is performed by randomly sampling a batch from real and generated data. A high 'pvalue' of the test indicates that the two observed instances of the distribution come from the same underlying distribution.

iii. Auto-Correlation Function score : The ACF score is a correlation score between values of a series and its lagged values at different time intervals used to analyse seasonality or stationarity of the series. Unlike the first two, ACF score is calculated between values of a series instead of comparing two distributions. The ACF proposed by Wiese et al.(2020)[2] for historical log return series is defined as a function of the time lag τ and return series $r_{1:T}$. Correlation of the lag with itself up until lag S ≤ T-1 is $C(\tau, r) = Corr(r_t, r_{t+\tau})$ whose output is given as C: R→$[0,1]^S$ and $r_{1:T} \rightarrow$ (C(1;r), … , C(S;r)). Thus, ACF score computed for a function $f: R \rightarrow R$ applied element-wise is -

$$ACF(f) : = \left|\left| C(f(r_{1:T})) - \frac{1}{M} \sum_{i=1}^{M} C\left(f(r_{1:T,\theta}^{(i)})\right) \right|\right|_2 \quad (17)$$

where S, T and M are constants.

iv. Leverage Effect Score : Leverage effect score is just a simple modification of the Auto-Correlation Function score. Leverage effect score compares the historical and generated time series by measuring the correlation between squared log returns of the lagged and log returns themselves given as $L(\tau, r) = Corr(r^2_{T+\tau}, r_t)$ for lag τ, similar to ACF score. Thus, similar to (17), the leverage effect score is given by -

$$LE(f): = \left|\left|L(f(r_{1:T})) - \frac{1}{M}\sum_{i=1}^{M} \quad C\left(f(r^{(i)}_{1:T,\theta})\right)\right|\right|_2 \quad (18)$$

Results from Table 1 show that TabDDPM performs better at the Earth Mover's Distance metric showcasing a better method for modelling distributions. The K-S test also backs that claim as the hypothesis that the generated coming from the real distribution is 96.38% for TabDDPM but only 42.455% for QuantGAN. If looked very carefully, this is also visible from A2 where the TabDDPM models the distribution slightly better. However, as the QuantGAN architecture is specifically designed for sequential data, it scores better at ACF scores. This is backed by the graph in Appendix A3 that shows a much more "peaky" and erratic behaviour of the generated distribution of TabDDPM than QuantGAN.

	QuantGAN(TCN)	TabDDPM	QuantGAN(TCN) - all_features	TabDDPM - all_features
EMD(1) ↓	8.033986e-04	**4.403211e-04**	5.904130e-06	**5.106787e-06**
EMD(5) ↓	2.058422e-03	**1.818781e-03**	**1.400402e-05**	1.559988e-05
EMD(20) ↓	1.072807e-02	**8.936049e-03**	8.636183e-05	**3.617040e-05**
EMD(100) ↓	1.269983e-01	**2.054403e-02**	6.638992e-04	**5.630576e-04**
K-S Test (p-value) ↑	42.455190%	**96.380698%**	52.053748%	**82.805256%**
ACF(·) ↓	1.211521e-01	**1.056656e-01**	1.219926e-01	**1.028182e-01**
ACF(| · |) ↓	**1.115744e-01**	1.326045e-01	**1.058974e-01**	1.103400e-01
ACF((·)^2) ↓	**1.027946e-01**	1.374898e-01	8.543895e-02	**7.982600e-02**
Leverage Effect ↓	1.310409e-01	**1.083866e-01**	**1.239125e-01**	1.275619e-01
ACF Score ↓	**2.342331e-01**	2.437218e-01	2.207953e-01	**2.130510e-01**

b. Qualitative Analysis

To evaluate the generated data with the real distribution, we randomly select a batch from both distributions during each epoch and observe behaviours of the distributions through correlation metrics and other transformations. Stylised characteristics observed in the generated data modelled from the real data are mentioned below.

i. Heavy Tail distribution - Appendix A1 shows that TabDDPM is able to model the heavy tail distribution of stock returns close to the real empirical finding.

ii. Peaked Distribution - Results from Appendix A1 also reveal that TabDDPM is not able to produce the peaked distribution of the real returns while QuantGAN is able to.

iii. Volatility Clustering - TabDDPM is able to replicate the volatility clustering effect of the real distribution much better than QuantGAN which can be seen from Appendix A3.

iv. Leverage effect - TabDDPM model fails to capture the leverage effect as seen from Appendix A. On the other hand, QuantGAN is able to capture the effect but overestimates the negative correlation of the real returns.

v. Uncorrelated but not independent returns - Small ACF scores in Appendix A5 show a low correlation between returns while the Volatility Clustering graph in Appendix A3 makes a case against the returns being independent.

5. Conclusion

In this project, the problem of time-series modelling and different methods to achieve it are discussed. Two different methodologies, TabDDPM and QuantGAN are compared for the modelling of the time-series task. Although TabDDPM models the overall distribution of the real data well as per EMD or K-S test, it falls short of the correlation metrics when compared to QuantGAN, which is designed for sequential data. On the other hand, while analysing stylized characteristics, it is observed that the effect of volatility clustering that is seen in neighbouring time-steps, is modelled by TabDDPM better than QuantGAN but that is not the case with the leverage effect. QuantGAN's architecture allows it to capture long range dependencies which is why the leverage effect can be modelled. As TabDDPM is a framework for tabular data, it does not have a way to encode sequential data. Improvements to the model may include some mechanism like attention to model the sequence which theoretically should help with the autocorrelation metrics. Further research also needs to be done to investigate why the GAN produces a more peaked distribution like empirical returns but the diffusion model is unable to. The heavy-tail distribution of empirical returns is modelled closely by TabDDPM but not by the GAN. The above mentioned points should be considered when using generative time-series models in other problems; like data augmentation for regression tasks. Thus, depending upon the particular application and the required properties desired in the generated sequence, the diffusion approach of TabDDPM is presented as an alternative to GANs for generating time series data.

6. References

[1] Kotelnikov, A., Baranchuk, D., Rubachev, I., & Babenko, A. (2022). TabDDPM: Modelling Tabular Data with Diffusion Models. arXiv preprint arXiv:2209.15421. doi:https://doi.org/10.48550/arXiv.2209.15421.

[2] Wiese, M., Knobloch, R., Korn, R., & Kretschmer, P. (2020). Quant GANs: deep generation of financial time series. Quantitative Finance, 20(9), 1419-1440. doi:https://doi.org/10.1080/14697688.2020.1730426.

[3] Devi, S. (2021). Asymmetric Tsallis distributions for modeling financial market dynamics. Physica A: Statistical Mechanics and its Applications, Volume 578, 126109. doi:https://doi.org/10.1016/j.physa.2021.126109.

[4] Gupta D., Pratama M., Ma Z., Li J. & Prasad M. (2019). Financial time series forecasting using twin support vector regression. PLOS ONE 14(3): e0211402. doi:https://doi.org/10.1371/journal.pone.0211402.

[5] Ho, J., Jain, A. & Abbeel, P. (2020). Denoising Diffusion Probabilistic Models. Advances in Neural Information Processing Systems, *33*, 6840-6851. doi:https://proceedings.neurips.cc/paper/2020/hash/4c5bcfec8584af0d967f1ab10179ca4b-Abstract.html

[6] Nichol, A.Q. & Dhariwal, P. (2021). Improved Denoising Diffusion Probabilistic Models. Proceedings of the 38th International Conference on Machine Learning, 139:8162-8171. doi:https://proceedings.mlr.press/v139/nichol21a.html.

[7] Dhariwal, P. & Nichol A., (2021). Diffusion Models Beat GANs on Image Synthesis. Advances in Neural Information Processing Systems, *34*, 8780-8794. doi:https://proceedings.neurips.cc/paper/2021/hash/49ad23d1ec9fa4bd8d77d02681df5cfa-Abstract.html

[8] Ho, J., Salimans, T., Gritsenko, A., Chan, W., Norouzi, M., & Fleet, D. J. (2022). Video diffusion models. arXiv preprint arXiv:2204.03458. doi:https://doi.org/10.48550/arXiv.2204.03458.

[9] Li, X., Thickstun, J., Gulrajani, I., Liang, P. S., & Hashimoto, T. B. (2022). Diffusion-lm improves controllable text generation. Advances in Neural Information Processing Systems, 35, 4328-4343. doi:https://proceedings.neurips.cc/paper_files/paper/2022/hash/1be5bc25d50895ee656b8c2d9eb89d6a-Abstract-Conference.html

[10] Gong, S., Li, M., Feng, J., Wu, Z., & Kong, L. (2022). Diffuseq: Sequence to sequence text generation with diffusion models. arXiv preprint arXiv:2210.08933. doi:https://doi.org/10.48550/arXiv.2210.08933.

[11] Strudel, R., Tallec, C., Altché, F., Du, Y., Ganin, Y., Mensch, A., Grathwohl, W., Savinov N. Dieleman, S., Sifre, L. & Leblond, R. (2022). Self-conditioned embedding diffusion for text generation. arXiv preprint arXiv:2211.04236. doi:https://doi.org/10.48550/arXiv.2211.04236.

[12] Tashiro, Y., Song, J., Song, Y., & Ermon, S. (2021). CSDI: Conditional score-based diffusion models for probabilistic time series imputation. Advances in Neural Information Processing Systems, 34, 24804-24816. doi:https://proceedings.neurips.cc/paper/2021/hash/cfe8504bda37b575c70ee1a8276f3486-Abstract.html

[13] Toth, D., & Jones, B. (2019). Against the Norm: Modeling daily stock returns with the Laplace distribution. arXiv preprint arXiv:1906.10325. doi:https://doi.org/10.48550/arXiv.1906.10325.

[14] Chakraborti, A., Toke, I. M., Patriarca, M., & Abergel, F. (2011). Econophysics review: I. Empirical facts. Quantitative Finance, 11(7), 991-1012. doi:https://doi.org/10.1080/14697688.2010.539248.

[15] Cont, R. (2001). Empirical properties of asset returns: stylized facts and statistical issues. Quantitative finance, 1(2), 223. doi:10.1088/1469-7688/1/2/304

[16] Dogariu, M., Ştefan, L. D., Boteanu, B. A., Lamba, C., Kim, B., & Ionescu, B. (2022). Generation of realistic synthetic financial time-series. ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM), 18(4), 1-27. doi:https://doi.org/10.1145/3501305.

[17] Fu, W., Hirsa, A., & Osterrieder, J. (2022). Simulating financial time series using attention. arXiv preprint arXiv:2207.00493. doi:https://doi.org/10.48550/arXiv.2207.00493

[18] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. & Bengio, Y. (2020). Generative adversarial networks. Communications of the ACM, 63(11), 139-144. doi:https://doi.org/10.1145/3422622

[19] Bai, S., Kolter, J. Z., & Koltun, V. (2018). An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arXiv preprint arXiv:1803.01271. doi:https://doi.org/10.48550/arXiv.1803.01271

[20] Oord, A. V. D., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., ... & Kavukcuoglu, K. (2016). Wavenet: A generative model for raw audio. arXiv preprint arXiv:1609.03499. doi:https://doi.org/10.48550/arXiv.1609.03499

[21] Yang, L., Zhang, Z., Song, Y., Hong, S., Xu, R., Zhao, Y., Zheng, W., Cui, B., & Yang, M. H. (2022). Diffusion models: A comprehensive survey of methods and applications. arXiv preprint arXiv:2209.00796. doi:https://doi.org/10.48550/arXiv.2209.00796

[22] Hernandez, E., & Dumas, J. (2022). Denoising diffusion probabilistic models for probabilistic energy forecasting. arXiv preprint arXiv:2212.02977. doi:https://doi.org/10.48550/arXiv.2212.02977

[23] Hoogeboom, E., Nielsen, D., Jaini, P., Forré, P., & Welling, M. (2021). Argmax flows and multinomial diffusion: Learning categorical distributions. Advances in Neural Information Processing Systems, 34, 12454-12465. doi:https://proceedings.neurips.cc/paper/2021/hash/67d96d458abdef21792e6d8e590244e7-Abstract.html

[24] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Müller, A., Nothman, J., Louppe, G., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. & Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12(85), 2825-2830. doi:https://www.jmlr.org/papers/v12/pedregosa11a.html

[25] Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., & Chen, X. (2016). Improved techniques for training gans. Advances in neural information processing systems, 29. doi:https://proceedings.neurips.cc/paper/2016/hash/8a3363abe792db2d8761d6403605aeb7-Abstract.html

[26] Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., & Hochreiter, S. (2017). Gans trained by a two time-scale update rule converge to a local nash equilibrium. Advances in neural information processing systems, 30. doi:https://proceedings.neurips.cc/paper_files/paper/2017/hash/8a1d694707eb0fefe65871369074926d-Abstract.html

[27] Levina, E., & Bickel, P. (2001, July). The earth mover's distance is the mallows distance: Some insights from statistics. In Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001 (Vol. 2, pp. 251-256). IEEE. doi:10.1109/ICCV.2001.937632

[28] Juan, M. (2022). Finding the Probability Distribution Implied in Option Prices. https://quantdare.com/finding-the-probability-distribution-implied-in-option-prices/

[29] Massey Jr, F. J. (1951). The Kolmogorov-Smirnov test for goodness of fit. Journal of the American statistical Association, 46(253), 68-78. doi:10.1080/01621459.1951.10500769

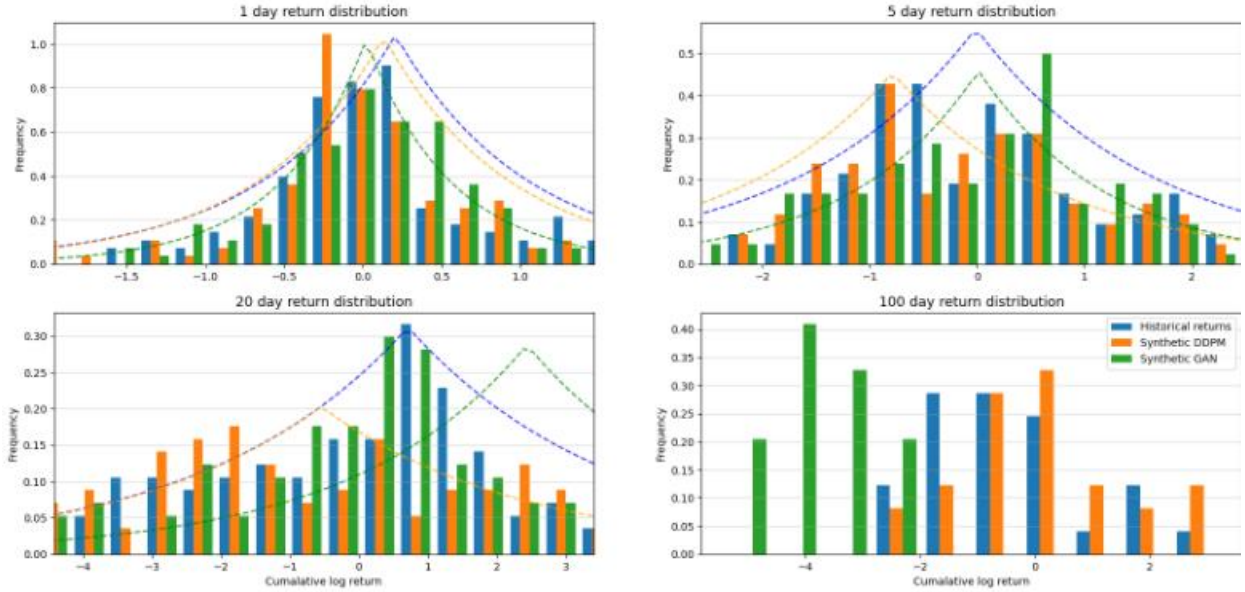
[30] Martin, R. A. (2020). Option-implied probability distributions. Reasonable Deviations Web. doi:https://reasonabledeviations.com/2020/10/01/option-implied-pdfs/

[31] Nicholson, C. V. (n.d.). A Beginner's Guide to Neural Networks and Deep Learning. Pathmind Web. doi:https://wiki.pathmind.com/neural-network

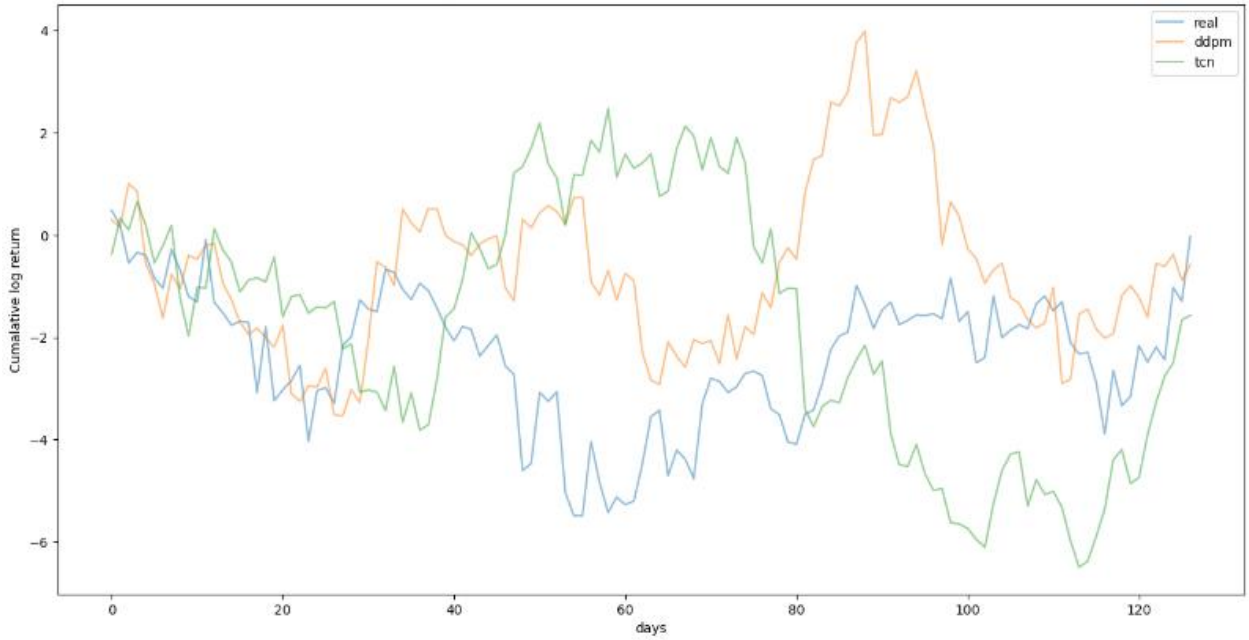
[32] Sullivan, J. (2021). temporalICN: Implementation of Quant GANs: Deep Generation of Financial Time Series, Github repository. doi:https://github.com/JamesSullivan/temporalCN

7. Appendix

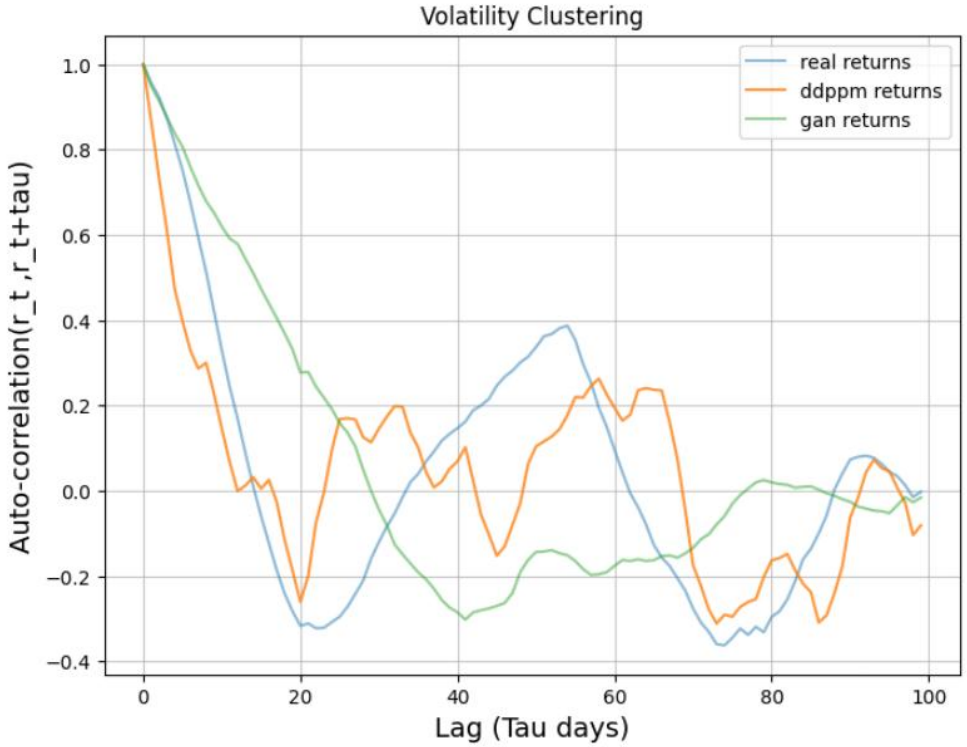
A1. Frequency Density Distribution of Cumulative Log returns



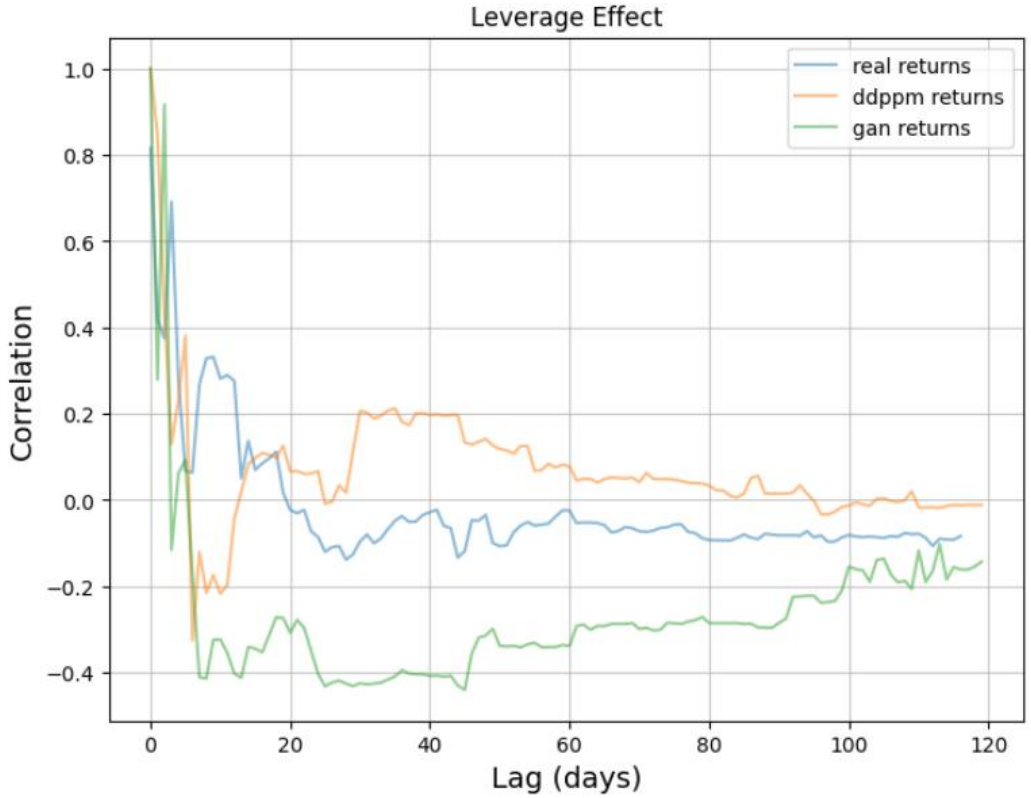
A2. Cumulative log return of the time sequences being compared.



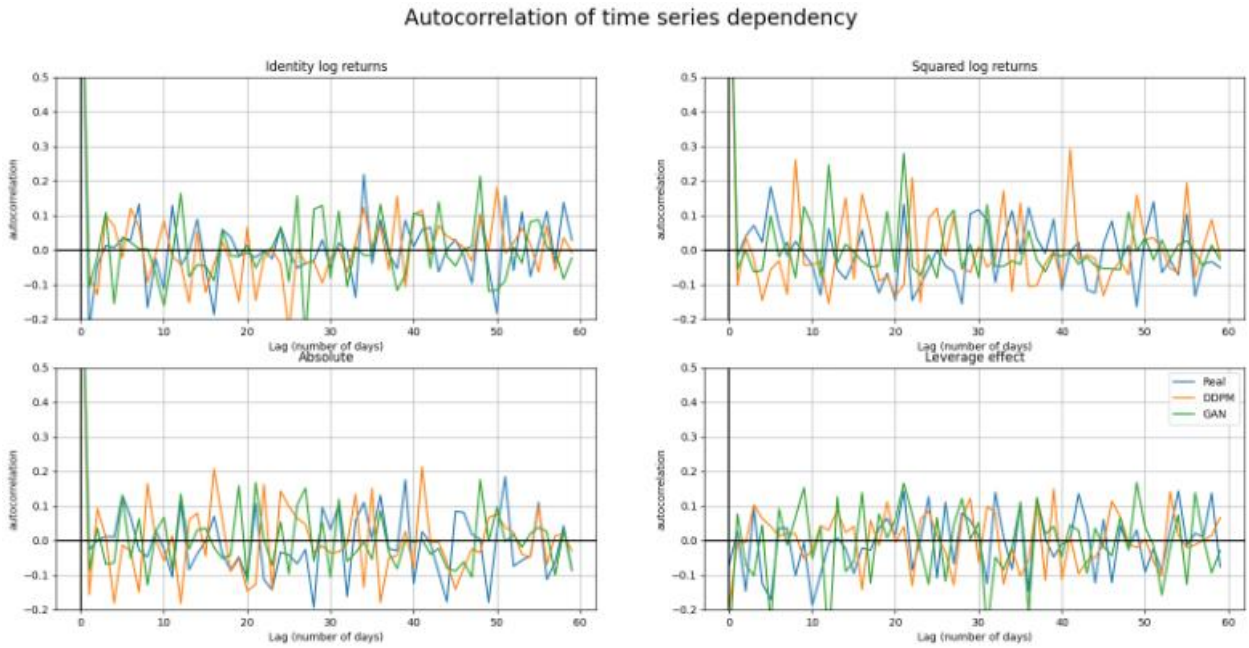
A3. Volatility Clustering Effect of the returns of real and synthetic distributions.



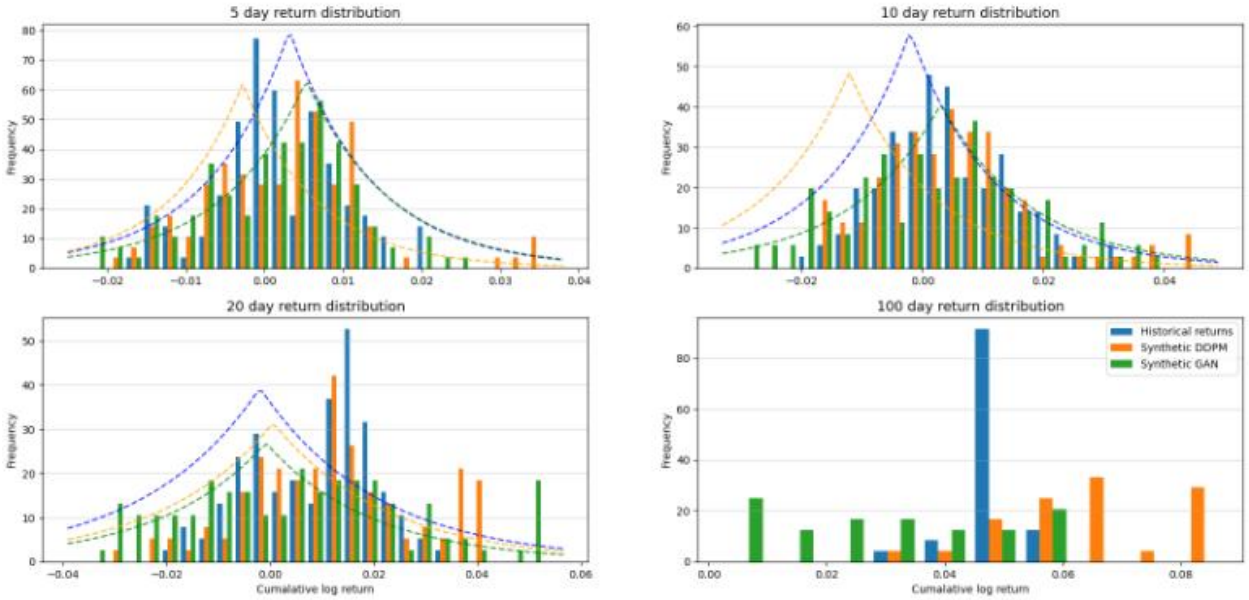
A4. Leverage Effect of the returns of real and synthetic distributions.



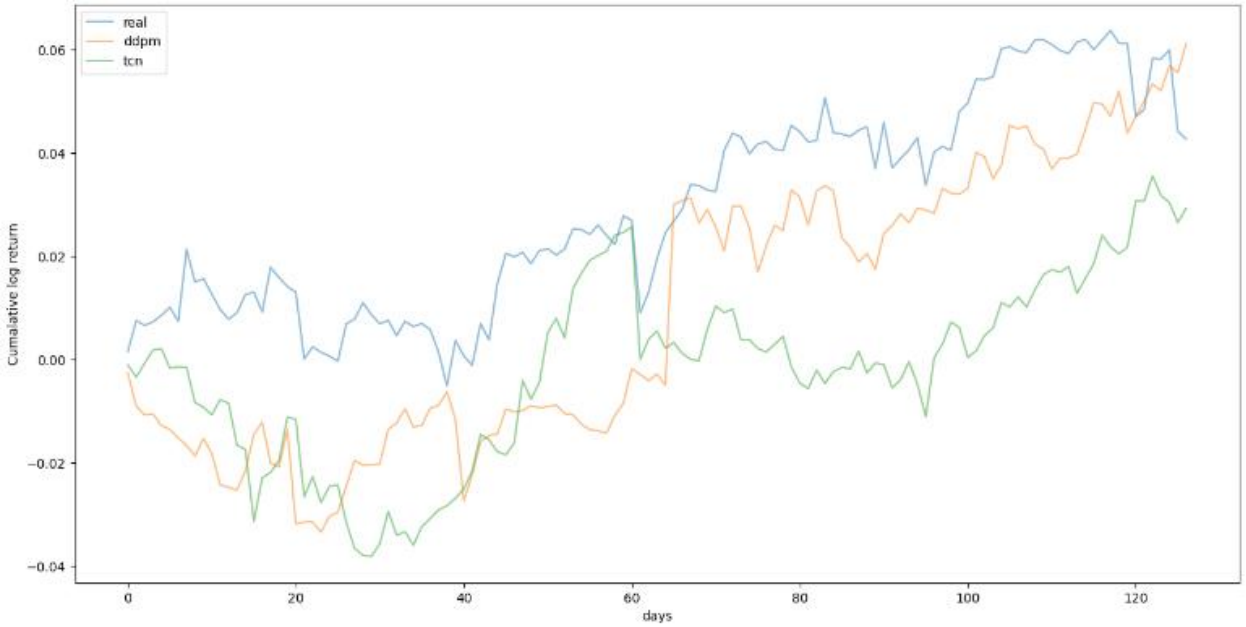
A5. Autocorrelation of the real and generated distributions.



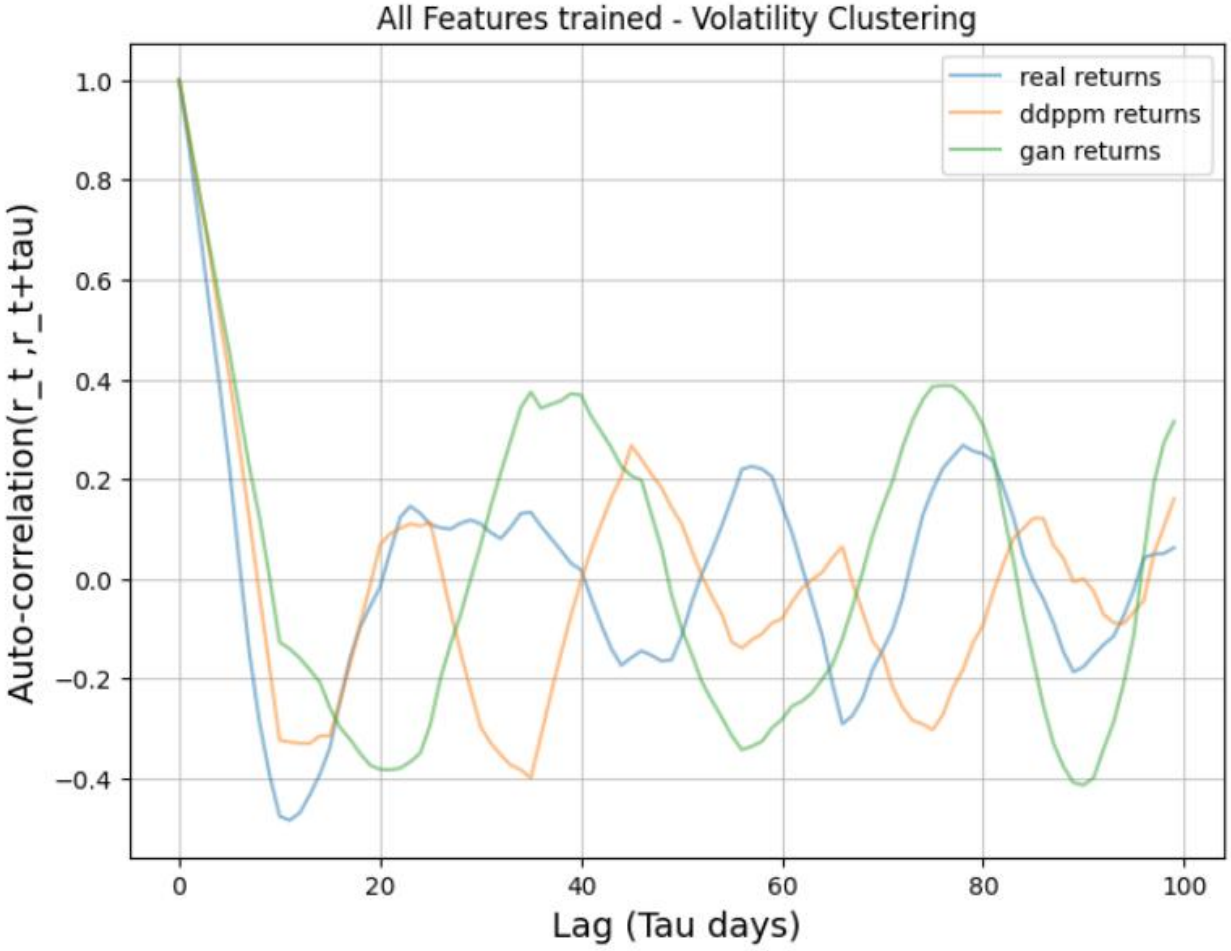
A6. All Features Trained - Frequency Density Distribution of Cumulative Log returns



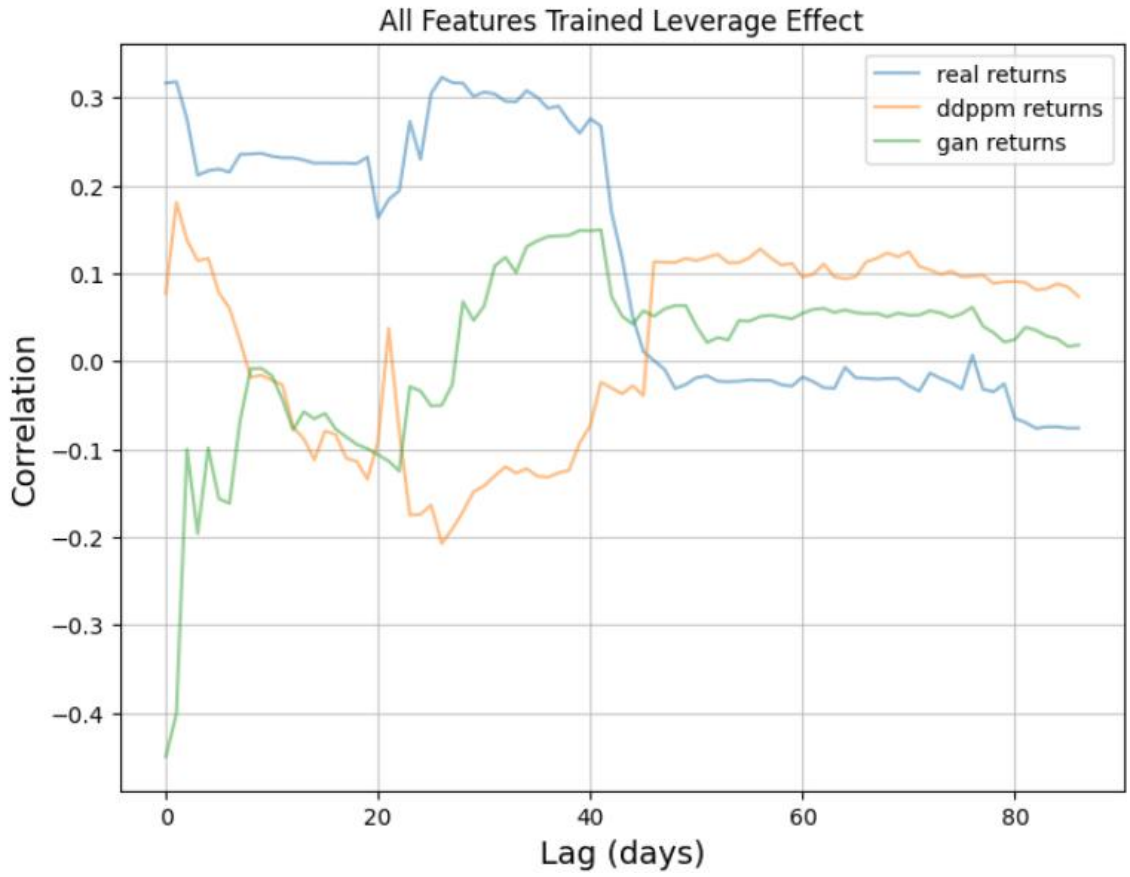
A7. All Features trained - Cumulative log return of the time sequences being compared.



A8. All Features trained - Volatility clustering



A8. All Features trained - Leverage Effect



A9. All Features Trained - Autocorrelation of the real and generated distributions.