Generation Of Synthetic Data To Improve Financial Prediction Models

Stefan Xuereb

Supervisor: Dr. Vince Vella

July, 2023

Submitted in partial fulfilment of the requirements for the degree of M.Sc. in Artificial Intelligence.

L-Università ta' Malta
Faculty of Information & Communication Technology

Acknowledgements

The word wonder has two related meanings. The first is the emotion that comes out of inquiry. The second, that is an even more potent emotion, is what is felt when having an aesthetic experience that was previously unknown to you. The time spent studying this subject was for me, a source of constant wonder. I am truly grateful to the people that have made the time I have spent on this course such an enriching experience, namely my supervisor, Dr. Vince Vella, all the tutors of the various courses and my partner, Althea, in her unwavering support.

Abstract

The main issue at the heart of this dissertation is the improvement of ML financial prediction systems through the use of augmented training data. An overview and assessment of various financial data synthesis methods has been carried out. To be able to accomplish this assessment, a review of evaluation methods had to be undertaken keeping in mind that asset price time sequences have characteristics that will be missed if only the typical statistical measures are used. This is because asset time sequences have particular patterns that play out over the time dimension in addition to the magnitude dimension.

Augmented asset price time sequences may be useful in a number of scenarios. High fidelity synthetic data may be needed to substitute real data to preserve confidentiality. The temporal portions of the data that are of interest may be short as is the case for financial bubbles or short lived changes in the behavior of market participants which require the training data to exhibit these particular characteristics repeatedly. In addition to this, ML systems improve in their ability to deal with unseen data if they are trained on larger datasets. The objective of this work is to choose the best performing (in terms of fidelity) financial data synthesis method and to verify that the data thus generated actually improves the performance of an asset price prediction system.

We undertook to use a number of qualitative visual evaluation metrics that helped confirm the quantitative assessments carried out. This resulted in SigCWGAN being selected. The asset time series generated using this GAN were then used to train an ARIMA/RNN asset price prediction system. The data obtained using SigCWGAN did, in fact result in an improvement in the performance of the predictive system we used in terms of MAE, MSE and MAPE metrics.

Contents

List of Figures

List of Tables

List of Abbreviations

ANFIS neuro-fuzzy inference system

ANN Artificial Neural Networks

AR-FNN autoregressive feedforward neural network

AR Autoregression

ARIMA Autoregressive Integrated Moving Average

CNN Convolutional Neural Networks

CV Computer Vision

DBN Deep Belief Network

DL Deep Learning

DSS Decision Support System

EMH Efficient Market Hypothesis

GA Genetic Algorithms

GARCH Generalized autoregressive conditional heteroskedasticity

GMM Gaussian Mixture Model

GRU gated recurrent unit

HMM Hidden Markov Model

HSI Hong Kong Hang Seng Index

I.I.D Independent and identically distributed

I Integrated

ILSVRC ImageNet Large Scale Visual Recognition Challenge

IXIC NASDQ Index

KOSPI Korea Composite Stock Price Index 20 0

MA Moving average

MAE Mean Absolute Error

MAPE Mean Absolute Percentage Error

ML Machine Learning

MLP	Multi-Layer Perceptrons
MSD	mean squared deviation
MSE	Mean Squared Error
N225	Nikkei index
NB	Naive Bayes
NN	Neural network
PCA	Principle component analysis
PSO	Particle Swarm Optimisation
RBM	Restricted Boltzmann Machine
ResNet	Residual Networks
RF	Random Forest
RNN	Recurrent Neural Networks
SSEC	Shanghai Composite Index
SVR	Support Vector Regression
t-SNE	t-Distributed Stochastic Neighbour Embedding
TA	Technical Analysis
UCR	University of California Riverside
VAE	Variational Autoencoder
VGG	Visual Geometry Group

1 Introduction

Synthetic data is defined by Nikolenko (2021) as "data produced artificially with the purpose of training machine learning models". Akyash et al. (2021) state that "Data Augmentation which refers to generating new training samples to generalize the model" and this gives the impression that augmentation is synonymous with synthesis. We have found the word augmentation used in a similar way in a number of papers (Iwana and Uchida, 2021a; Oh et al., 2020; Shorten and Khoshgoftaar, 2019). The word augmentation means "to enlarge" and in computer vision an augmented data set is one were the original real data set has new synthetic images added to it. In this application augmented data simply means new synthetic data that is added to the original data.

In the case of time sequences there is a distinction that must be made. We adopt the Nikolenko (2021) definition for the word "synthetic" but we will here use "augmented" as meaning a time series that has portions of synthetic data added to real data.

The term financial prediction also needs to be put in the context of the work presented here. Financial prediction encompasses a number of financial areas where forecasting or prediction are performed. Definitions are often fairly elastic and generally depend on the area of discourse and the particular author. The terms forecasting and prediction are often used interchangeably though forecasting is usually used in qualitative terms as often happens when the management of a company discusses changes in strategic direction. We, here, adhere to the quantitative meaning of prediction where financial models of some mathematical or computational form are used to estimate some statistic. Businesses monitor and attempt to forecast a number of internal statistics like projected cash flow, future sales, costs of inputs and others. The financial quantities that are at the focus of this work are asset or index prices as they are issued by the various stock exchanges around the world.

The issue that will be dealt with in this dissertation is whether artificially generated asset price time series will result in the improvement in the capability of a stock price prediction system in terms of a number of standard accuracy metrics.

1.1 Motivation

A number of related rationales have influenced individuals and organisations involved in finance to investigate data synthesis and augmentation. At the root of this interest is the ongoing advance in the capabilities of machine learning (ML). The use of ML in finance is clearly motivated by the fact that its application has resulted in substantial improvement in profitability, risk management and the reduction in errors of human origin (Van Liebergen et al., 2017).

The performance of a system is judged on the quality of its output given a range of inputs in its operational phase (i.e. not during training). ML systems fall under one of two basic categories: Regression or Classification. Regression operates like a continuous mathematical function, i.e. given an input X from a given domain, the system will give an output Y usually in a continuous range. Classifiers, on the other hand, will produce output that falls in discrete classes. In finance both of these types of systems are employed. Thus, the quality of a financial ML system is measured on how close to the correct answer, be it a class or a value, the system produces. After multiple repetitions during the testing phase, various statistics relating to the quality of the output of the system are produced.

Quality of the system output can be measured along two dimensions; bias and variance. To understand the meaning of these terms one should imagine the configuration of darts landing on a target. The diffusion of the darts is equivalent to the variance; thus low variance implies that the darts have landed close together. Bias, on the other hand, measures how far off-target the cluster of darts is.

Two fundamental factors in the making of an ML system are the algorithm and the data on which the system is trained. The quality of the system depends closely on both these factors. A high level overview of ML systems, including academic papers and text books will concentrate primarily on algorithms; their comparison and relative performance. This literature will deal with data in terms of its sourcing and pre-processing such that it is suitable as input to the given ML system.

An aspect pertaining to the issue of data that has been drawing increased attention in the literature is the question of the sufficiency of the volume of data available to ML systems (Liu et al., 2022). Particularly in the case of neural networks, the more nodes these have, the more data is needed to stabilise the weight values of the system's nodes during the training phase. Thus, the issue of the volume of training data becomes predominant. It is easy to see that even though an advanced and complex algorithm may be available it will be rendered useless unless a suitable amount of data is available for training. This lack of data may lead to poor system performance in that it will either under-fit or over-fit its output. Under-fitting results in a lack of output accuracy. Over-fitting happens when the system is accurate when given data it has been trained with but performs badly with unseen data. There are numerous cases where this happens in Financial applications of ML. For example, ML is often used in the prediction of future stock values. Oftentimes the only data available for these prediction systems are the daily opening and closing prices. Thus, a year's worth of data will have 240 data points where the amount of data to properly train the ML system may be several thousand.

Another type of problem arising from insufficient data is analogous to class imbalance in tabular data (time sequences are predominant in finance, particularly in stock forecasts). This problem manifests itself in at least two ways. First is the behaviour (i.e. adopted strategies) of financial market traders, where successful strategies will only be used for a finite period of time, after which the strategy no longer yields a profit (Assefa et al., 2020). These changes in behaviour will affect the stochastic quality of the stock's time series. This means that if traders have recently adopted a new strategy and the ML system is capable of identifying this trend, it will be unable to do so if it is not trained on a sufficient amount of data containing the hidden new trend. This new trend may exist in only the most recent data (possibly a couple of months). This makes the problem of a sufficient volume of data considerably worse.

The second problematic area is the trends that are only temporary but may re-occur in the future like the rise of financial bubbles and their bursting (Zhou et al., 2019). These have occurred very rarely and are another case where sufficient data could lead to the detection of a very important trend in a time series.

The two cases mentioned above are simply two special cases where additional relevant data will improve the performance of financial ML systems. The value of data synthesis and augmentation must not be underestimated in their ability to improve the performance ML systems in general.

Since the need for suitable data is an integral part of the implementation of any ML system, the scope of application of augmentation technology is similarly very broad. The choice made here to focus on financial prediction is an attempt to balance general applicability with a specific use case that is extensively employed in the finance industry.

Over the years since their introduction by Goodfellow et al. (2014) there has been a growing interest in the effectiveness of Generative Adversarial Networks (GANs) in generating artificial data. Surveys like that performed by Pan et al. (2019) have pointed to an implicit shift in preference towards the use of GANs in artificial data generation.

That said, literature proposes a number of different approaches. Generative solutions involving natural language have used Naive Bayes and Latent Dirichlet Allocation (Karim and Rahman, 2013; Ma et al., 2023). Among the various generative applications of Gaussian Mixture Models one should mention the generation of bag-of-visual words as used by

Fernando et al. (2012). Restricted Boltzmann Machines and their extension, Deep Belief Networks have been used in generative applications of music and facial expressions (Lattner et al., 2018; Susskind et al., 2008). Variational Autoencoders have found application in the generation of graphs (Ma et al., 2018; Simonovsky and Komodakis, 2018).

1.2 Aim and Objectives

The aim of this dissertation is to identify generative machine learning approaches that generate emulated data sets with satisfactory stochastic and temporal characteristics, closely resembling real financial data "stylized facts." The generated simulated data will be validated to ensure that it performs at least as well, if not better, than real data when used to train a financial prediction system.

The following objectives have been identified as the means to achieve this aim. Our first objective was to identify and validate whether GANs can be used to effectively generate synthetic financial data. We split our first objective into two sub-objectives:

- Objective 1a: We wish to analyse and compare different generative approaches suggested by literature in the generation of financial synthetic data.
- Objective 1b: As a second part, We wish to analyse the performance of GANs in relation to other methods proposed in literature.
- Objective 2: To identify whether the combination of GANs with a financial prediction model can improve stock prices prediction performance.

1.3 Contributions

This dissertation focuses on enhancing financial prediction systems in machine learning (ML) using augmented training data through the application of GANs. Various methods for synthesizing financial data were comprehensively evaluated, considering the unique temporal patterns present in asset price time sequences that cannot be captured by traditional statistical measures alone.

The chosen method, SigCWGAN (Signature Conditioned Wasserstein Generative Adversarial Network), was validated using both qualitative and quantitative evaluations. The generated asset time series were employed to train an ARIMA/RNN asset price prediction system, leading to improved performance based on metrics like MAE, MSE, and MAPE.

As a result of this work, two contributions result from this undertaking:

1. We show that GANs, in particular SigCWGAN can be used to effectively generate synthetic financial data.
2. In addition we show that data augmentation of real data with synthetic GAN generated data can improve the performance of financial prediction systems.

1.4 Document Structure

This document has been structured as follows: Chapter 2 provides the background information that will help clarify the concepts discussed in this work. Chapter 3 describes the methodology used and the structure of the solution presented here. Chapter 4 presents the results of the experiments carried out to achieve the objectives described above together with an evaluation of these results. The final chapter serves as a conclusion were an overview of what has been achieved is presented, as well as a discussion of future work.

2 Background and Literature Review

This dissertation is a study of ways to improve financial prediction. The history of financial prediction is instructive in understanding how this technology has arrived to its' current state, which is the focus of this work. Algorithmic trading is intimately concerned with the ability of machines to predict stock market movements and thus, be able to benefit from these predictions by executing the appropriate trades.

The technologies available to perform this financial prediction are overviewed in a synopsis given below. Although the prediction technologies themselves are discussed here, the main issue of concern is how these technologies can be improved by having them operate on input that has been enhanced with the addition of synthetic data.

What follows is, thus, an overview of machine learning as applied to finance, trading algorithms and data augmentation. Before addressing these issues, the question that was most famously asked by Eugene Fama, i.e. is financial prediction at all possible, will be dealt with.

2.1 Historical Issues Regarding Financial Prediction

In spite of Fama's Efficient Market Hypothesis, researchers in the field of finance come up with a range of methods to forecast the stock market (Cavalcante et al., 2016). One school of thought follows what is called fundamental analysis, where statistics that are used to evaluate the performance of companies are used to forecast future performance. On the other hand there is technical analysis (TA), where predictions are based on the assumption that observed trends in prices and volumes will reoccur in the future.

2.1.1 Fundamental Analysis

Fundamental analysis considers factors like the company's balance sheet, the market and economy it operates in. The amount of literature dealing with fundamental analysis is far smaller than what is available regarding technical analysis (Cavalcante et al., 2016) since it is much more challenging to build models that embody this knowledge and give acceptable performance. Time series for gross domestic product, interest rates, customer price indices, currency exchange rates and other macroeconomic indicators are used when doing fundamental analysis (Boyacioglu and Avci, 2010) . Unstructured data like financial news and blogs are also used but are difficult to deal with because of their varying formats and irregular availability. One approach to handle these difficulties is text mining. Other techniques used for stock price forecasting that is gaining traction are sentiment analysis and social network analysis (Bollen et al., 2011) .

2.1.2 Technical Analysis

Technical analysis takes a completely different approach that mostly focuses on the observation of prices and volumes. The essential idea here is that price movements trace out recognisable patterns that can be identified. The timely identification of these patterns is used by traders to forecast what an asset price will do in the short term future as it is assumed that the identified pattern will evolve in a predictable manner.

Technical analysis embraces a set of techniques that are essentially based on the belief that history will repeat itself, that is, that past trends have a tendency to reoccur. The origins of technical analysis may be traced back to Joseph de la Vega, a merchant from Amsterdam with his accounts of the Dutch financial markets in the 17th century. Some attribute technical analysis to Munehisa Homma, (1724-1803), also known as Sokyu Homm who was a wealthy rice merchant from Sakata, Japan. He is believed to have originally developed technical analysis because he invented candlestick charting, which is a fundamental technique of technical analysis even today.

The development of technical analysis in the US began later, in the late 19th century. Scholars usually identify the writings of Charles Dow, who was the founder and editor of " Customer's Afternoon Letter" that was the precursor of the Dow Jones Industrial Average. Dow is also remembered as the originator of the Dow Theory that has evolved and is seen as the basis of today's technical analysis.

Classical technical analysis (i.e. that performed by human traders) primarily deals with what are known as technical indicators. Traders use Indicators as overlays on the chart they are studying and this gives them additional information through suitable transformations of price and volume.

An overview of the literature reveals that technical analysis is the approach that has been studied most (Aguilar-Rivera et al., 2015; Atsalakis and Valavanis, 2009; Cavalcante et al., 2016). One of the underlying beliefs in technical analysis is that all new information coming from news sources and published statistics, is already accounted for in stock prices. Thus the analysis of price patterns should be enough to predict future price movements. Nazário et al. (2017) have carried out a survey of technical analysis studies over 55 years. This paper also presents an extensive study of technical indicators that are used by technical analysts as their sources of buy and sell signals. There are, however, studies like the one carried out by Park and Irwin (2007) that find that strategies that use technical indicators have limited success. Notwithstanding a widespread scepticism toward TA, Park and Irwin (2004) have reviewed 58 positive studies from a total of 92.

There is ongoing controversy between the community of technical analysts operators and the academics who study the subject. Flanegin and Rudd (2005) highlight this divergence in a survey they carried out amongst academics and practitioners dealing with technical analysis. It showed that TA operators ranked the books on TA in the given list far higher than the academics. Strong (1987) had already noted that 60% of PhD's believed TA could not be used to improve investment selection and effectiveness. Malkiel (2019) in his famous book "A Random Walk Down Street" asserts that TA "shares a pedestal with alchemy". Lo et al. (2000) amongst others state that the effectiveness of technical analysis is inferior to that of Fundamental analysis.

The main objection these authors attest is that asset price time sequences are like random walks; their future behaviour cannot be surmised from their past; the sequences have no "memory". This statement goes back to one of the founding fathers of Financial Analysis as an academic subject; Louis Bachelier. In his 1900 PhD thesis Bachelier (1900) comments "the mathematical expectation of the speculator is zero.". In Cowles 3rd and Jones (1937) assert that stock market prices were fundamentally random. Cootner (1964) made similar assertions. Eugene Fama (1965) wrote the seminal article in which he expressed his belief that the stock market is essentially random.

2.2 Efficient Market hypothesis

There are two fundamental questions that need to be tackled with regard to the possibility of prediction in financial market prices. The first is whether the market allows profits to be made if it is, in fact, "efficient" as stated by Fama (1965) in his thesis.

Some authors believe the EMH is a steady state theory rather than it holding true at all times. Grossman and Stiglitz (1980) argue that since information is costly nobody would seek it if it gave no advantage as stated by the EMH. Here we can also cite Samuelson's Dictum that states:

"Modern markets show considerable micro efficiency (for the reason that the minority who spot aberrations from micro efficiency can make money from those occurrences and, in doing so, they tend to wipe out any persistent inefficiencies). In no contradiction to the previous sentence, I had hypothesized considerable macro inefficiency, in the sense of long waves in the time series of aggregate indexes of security prices below and above various definitions of fundamental values." quoted in Shiller (2015).

The second question is whether it is possible to predict asset prices or trends simply by analysing the historical behaviour of the price of assets. If the "Efficient Market Hypothesis" (EMH) were to be strictly correct then no profit could be made from technical analysis (TA) or any analysis (including fundamental analysis). A definition of the EMH states that "asset prices reflect all available information". This means that there should be only one market price for any given asset and deviations from that price are guaranteed to produce gains or losses once the buyers pay less or more for the said asset. A good analogy to understand why the market price may be temporarily "out of kilter" is the concept of steady state in economics. This is said to happen when there are no changes in independent economic variables and thus dependent economic variables do not change either. Thus, one could hypothetically calculate the value of dependent variables from the appropriate theoretical laws. Economists do not believe that economies are ever in "steady state" and, in fact, their subject is mostly about predicting economic outcomes when independent variables change. In the same manner, the EMH should be seen as the market's hypothetical steady state.

The cost of information has been discussed by Cross et al. (2008). Rode et al. (1995) of the Wharton school states that “substantial constraints on the information processing time allowed,” where “there is also a continual abundance of new information made available,” and that “this flow of information easily exceeds investor's abilities to process it completely.” Lo and MacKinlay (1988) in the article "The Adaptive Markets Hypothesis: Market Efficiency from an Evolutionary Perspective" states that even if one accepts the EMH, risk aversion varies and depends on market behaviour making it fluctuate with time.

The concept of "Information" should be looked into in the EMH definition. The EMH assumes that "perfect information" is known to the market. What this means is that the market as a whole is aware of the information necessary to value an asset but not that any one participant is aware of that "perfect information". In fact if it were the case that the parties in a trade had perfect information, then the trade would most likely not happen since no gain would accrue to any participant. This leads to understanding why trades do in fact happen for two fundamental reasons.

First, the market and the market price for one particular asset do deviate from a "perfect market price" as events, especially trades themselves cause perturbances that take time to settle back to the "steady state". An analogy to this is a guitar string. The steady state is when the string is stationary and that would be the EMH price of the asset. Events are analogous to the string being struck and the oscillation of the string represents the asset price being over or under it’s "real price". These oscillations can be the source of profits or losses for traders.

The second reason why trades do in fact occur is that participants in the market hardly ever possess "perfect information". The trade occurs, even if the market is in fact pricing the asset correctly (which, as seen above hardly ever happens), because a deficit of information leads one party in the trade to value the asset more than the market and vice-versa.

A phenomenon that challenges the "purely random" point of view, as affirmed by the EMH, is that of "fat tails". Much of the theory in academic financial analysis is based on the normal distribution. This does not mean that there do not exist other statistical distributions, but only that incorporating these more exotic distributions into the broader theory is both hard and exceptional. This has lead to the fairly widespread view that extraordinary market events like "Black Monday", the 2008 financial crises and indeed the recent Covid-19 pandemic can "statistically" be ignored. Jackwerth and Rubinstein (1996) state that:

"if the life of the universe had been repeated one billion times and the stock market were open every day, a crash of that magnitude would still have been unlikely."

In his well known book "Why Stock Markets Crash" Didier Sornette (2009) states a crash as large as the one that occurred on "Black Friday" should only happen once every 520 million years. This sort of faith in traditional statistical methods should surely justify some level of scepticism.

A test of this scepticism was carried out by Lo and MacKinlay (2015) as they report the results of the "Proportional Amplitude" test they carried out and found that stock market price sequences fail this test. Larson (1960); Alexander (1961); Osborne (1962); Cootner (1964); Steiger in Cootner (1964); Niederhoffer and Osborne (1966); and Schwartz and Whitcomb (1977) also had previously made similar assertions.

There is also the question of the rationality of market participants who, according to the EMH should behave rationally. Some irrationality has been introduced into the theory by Black (1986) who allows for some noise trading that is not countered by "informed" market participants.

The following is a list of behaviours that has been documented in the literature:

• Over-confidence based on little information (Barber and Odean, 2001; Fischhoff, 1980; Gervais and Odean, 2001).

• “Herding” (Huberman and Regev, 2001).

• Regret (Bell, 1982); Clarke et al. (1994).

• Miscalculation of probabilities (Lichtenstein et al., 1982).

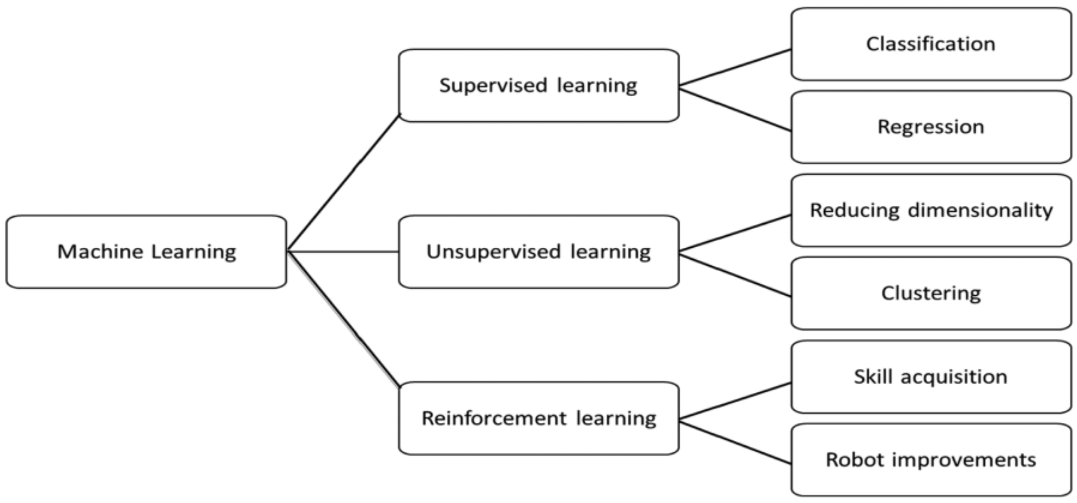


Figure 2.1: Machine Learning Taxonomy - from Milana and Ashta (2021)

- Overreaction (De Bondt, 1993).
- Hyperbolic discounting (Laibson, 1997)

2.3 Machine Learning in Finance

In the last ten or fifteen years, artificial intelligence has seen a significant escalation in its adoption in the field of economics (Agrawal et al., 2019; Brynjolfsson et al., 2021; Furman and Seamans, 2019). Taddy (2018) states that AI is a technology with a very wide scope of applicability. and has had an effect, directly or indirectly, in many academic, business and industrial areas. The financial industry is one where AI has been more extensively used (Biallas and O'Neill, 2020; Bredt, 2019). The range of ways in which AI has influenced finance have been investigated academic literature for nearly forty years (Pau et al., 1986; Pau and Tan, 1996; Shap, 1987).

Large Asset management companies like Black Rock, use Machine Learning (ML) techniques like algorithmic trading and portfolio management. When dealing with portfolios ML is used for rebalancing based on customer specifications (Tokic, 2018). ML is used to moderate the biases that the human traders may have thus reducing potential risk. However Bhatia et al. (2020) claim that the technology is not mature enough to carry out risk analysis for retail investors.

Mathematics, probability theory, and statistics are the theoretical foundations on which ML is based. The main categories in ML are supervised learning, unsupervised learning, and reinforcement learning as shown in figure 2.1 from Milana and Ashta (2021)

Supervised learning works by using feedback and labeled data (i.e. data that is unambiguously identified with a name or tag) to determine a correct outcome. On the other hand, unsupervised learning determines structure or patterns from unlabeled data. Thus, in unsupervised learning the system teaches itself autonomously. In Reinforcement learning the system must learn from the given data by credit assignment that is based on positive and negative messages (Boden, 1996; Li et al., 2018) . Reinforcement learning is used in portfolio management where there is a need for continuous rebalancing (Luo et al., 2019).

Algorithmic Trading systems are computational frameworks that are designed to replace or augment the work done by fund managers in selecting and trading financial instruments. This is done by adopting strategies that follow the same principles that human fund managers use like portfolio optimisation and trend prediction but with the consideration of a larger number of assets, at higher levels of accuracy and also at greater speed. Fund managers' goals are better profitability, lower draw downs and lower risk. It has been reported (Burgess, 2021) that algorithmic trading improves on these criteria when compared to the performance of human counterparts.

A problem that is continuously faced in the operation of algorithmic trading systems is the dearth of available training data especially when considering advanced machine learning models like Deep Learning (DL) Neural networks. One cause of this phenomenon is the increase in complexity of the DL networks used in these systems that has been facilitated by the continuous increase in component density that is available in the GPU's that drive these networks as evidenced in van Rhijn (2020).

Another factor that justifies a greater demand for good quality training data is the fact that some algorithmic trading strategies require specific types of data, like data arising in high volatility periods or data that is typical of financial bubbles. It is a well known fact that financial time series are non-stationary and the underlying distribution is ever changing depending on the current market situation (Bureš, 2021; Eckerli, 2021). This makes the data availability a more difficult problem since data for specific conditions can be very sparse. Le Guennec et al. (2016) and Lim and Zohren (2020) amongst others have stated that one of the most significant drawbacks to using deep learning methods in algorithmic trading is the shortage of usable training data. The number of parameters used in DL neural networks is in the order of milions (Le Guennec et al., 2016). The number of data points (closing prices for example) that are available for a particular asset in one year is around 250.

Another effect that insufficient data has on neural network performance is that it leads to overfitting. This is the result of the trade-off between bias and variance. A high bias means that the model predictions are off target. High variance indicates the model predictions are too diffuse. Implicit in this discussion are the concepts of over or under fitting. Overfitting happens when the accuracy of the model is high when using training data and low when used on unseen (test) data. The greater the complexity (i.e. layers and neurons) the more data is necessary to avoid overfitting.

The work that we have engaged in is the investigation of suitable methods by which the amount of available, good quality data may be increased. The issue of data augmentation has been investigated by a number of researchers in various areas with some positive results being reported (Fons et al., 2020; Fu et al., 2020; Lim and Zohren, 2020).

Data augmentation methods must produce good quality data in that it must:

- be suitable to the application area
- accurately reproduce the statistical features of the original (real) data
- preserve any time-wise conditionality

Various researchers in financial trading systems (Eckerli, 2021; Fu et al., 2020; Iwana and Uchida, 2021b; Shorten and Khoshgoftaar, 2019) have attempted to address these problems by looking into data augmentation methods. The current literature shows that there is a preference for the use of GANs over other methods although a variety of GAN architectures have been attempted and a critical review of these needs to be done. GANs have been used in algorithmic trading by Efimov et al. (2020), Zhang and Khoreva (2019), Takahashi et al. (2019), Snow (2020) and Wiese et al. (2020).

A significant problem that arises from the survey of GAN augmentation is the lack of an agreement on a common evaluation criterion, thus special attention to this has been given in this study. This also emerges from the comprehensive overview of GANs used in finance that Eckerli (2021) has compiled. Thus the key motivation of this work is the search for an optimal data augmentation technique that will translate into better algorithmic trading performance that gives rise to improved profitability.

2.4 Neural Networks

We will now turn to the core technology underlying ML; Neural Networks. Artificial Neural Networks (ANN) are strongly influenced by the structure and operation of biological brains after the work of Rosenblatt (1957) . In 1992 Siegelmann (1993) showed that ANNs can function as Turing machines. Because of their simplicity, early adoption in numerous fields and their well understood operation, they have been extensively adopted in the forecast of asset prices in the stock market. Bustos and Pomares-Quimbaya (2020) state that the Multilayer Perceptron (MLP) is the most often used ANN technique to forecast stocks.

Neural networks (NN) technology is one of the foundational techniques in the field of AI. Neural networks comprise of collections of connected neurons. Neural Networks have a variable number of layers excluding the outer ones. Deep Learning refers to Neural Networks that have multiple inner layers. Some network topologies allow connections to be eliminated and new ones to be put in (Boden, 1996).

In Neural Networks outputs are connected to inputs via what are usually referred to as hidden layers of neurons (Li et al., 2018). Each neuron in the layer is connected to any number of other neurons. Each of these neurons performs only one calculation. The neurons in these layers trigger according to non linear functions at their inputs and outputs. This non linearity is essential to the operation of NNs to avoid what is called the "vanishing gradient" problem.

Once the neuron fires, its signal is amplified or attenuated by a weight factor before the signal reaches the neurons in the next neural layer. Each neuron will produce an output depending on the reaction an activation function has to an input. The activation function may be as simple as a threshold or may be considerably more complex as those used in recent technologies like LSTMs and Transformers.

The connections are assigned weights that usually are modified by some algorithm. The most common approach to establishing the connection weights is by back propagation. The algorithm works from the final output layer and moves backwards through the intermediate layers until it gets to the input. This involves calculating the derivative of the activation function of the outputting neuron and passing a value from the receiving neuron back to the outputting neuron. Thus, the connection weights are sequentially modified from the output layer back to the initial input layer. This is known as loss function minimisation.

When NNs were first used in finance they were considered to be an evolution of regression models that were habitually used in the industry (Bodt et al., 1995). Eventually it was seen that NNs (or artificial NNs) have superior generalization power when compared to the statistical tools that were in use up to that time (Quah and Srinivasan, 1999). Other advantageous characteristics NNs exhibited was the ability to organize themselves from a random initialisation and the fact that they did not require complete data (Boden, 1996). Li et al. (2018) state that in DL there is no need to specify the logic of the system being modelled. Boden (1996) asserts that the salient characteristics of NN's are pattern recognition, the association between patterns, tolerance of messy and incomplete inputs, and robustness.

Li and Mei (2020) state that neural networks "allow intuitive learning" and are often more effective than the techniques normally used by technical analysis operators. One acknowledged drawback is that it is difficult to explain the results obtained.

In a study of the Chinese financial markets, the prediction of asset returns using NNs with two layers improved over what was attained using statistical techniques. The study by Li and Mei (2020) also showed that predictions with one layer or three layers were not as good as those obtained using two layers. More recent studies of NNs in finance are Coelho et al. (2019) , Hu et al. (2018) Qiu and Song (2016) , Mingyue et al. (2016) and Zhong and Enke (2017). These all used MLP models using the backpropagation technique.

2.5 GANs

Generative modelling, which is the main data augmentation technique investigated here, is considered to be in the class of unsupervised learning in ML. It essentially amounts to automatically acquiring patterns that occur in input data into some sort of digital structure that can then be used to generate new instances that seem realistic and plausible when compared to the original input data.

2.5.1 Early GAN History

GANs have been developed in the field of Computer Vision (CV). Their most acclaimed contribution has been the ability to generate images of human faces that are completely artificial and practically indistinguishable from images of real human beings. The technology that preceded GANs and was a precursor to their development was Autoencoders. The origin of autoencoders is variously attributed to LeCun et al. (1998); Bourlard and Kamp (1988) ; Hinton and Zemel (1993) and as early as Rumelhart et al. (1986).

Autoencoders consist of two neural networks back to back. The first takes an image as input and this is passed to smaller and smaller neural layers. This is done to extract the essential features of the input image and produce a "compression" that encapsulates the fundamental characteristics of the input image. This compressed version of the input data is stored in what is called the "latent space". This configuration is shown in figure 2.2 below. The output of this stage is then passed to a decoder stage that reconstitutes the original image. Autoencoders have proven to be useful in CV applications such as "denoising" of images.

GANs have a similar structure to autoencoders in that they consist of two networks back to back. The first network performs the generation stage which attempts to produce the digital structure that is required, for example the image of a human face. This digital structure is then passed onto the second stage that functions as a discriminator. The job of the discriminator is to decide if the data at its input is real or artificial. Feedback loops provide the loss functions to both networks. The generator assesses its performance as it

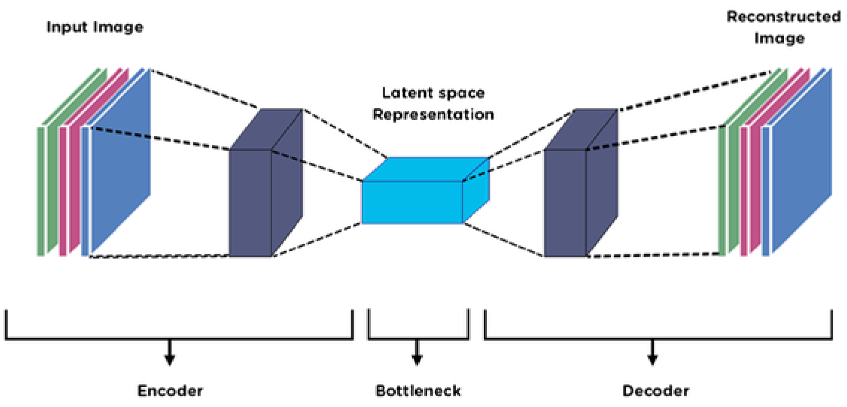


Figure 2.2: Autoencoder - from Grietzer (2017)

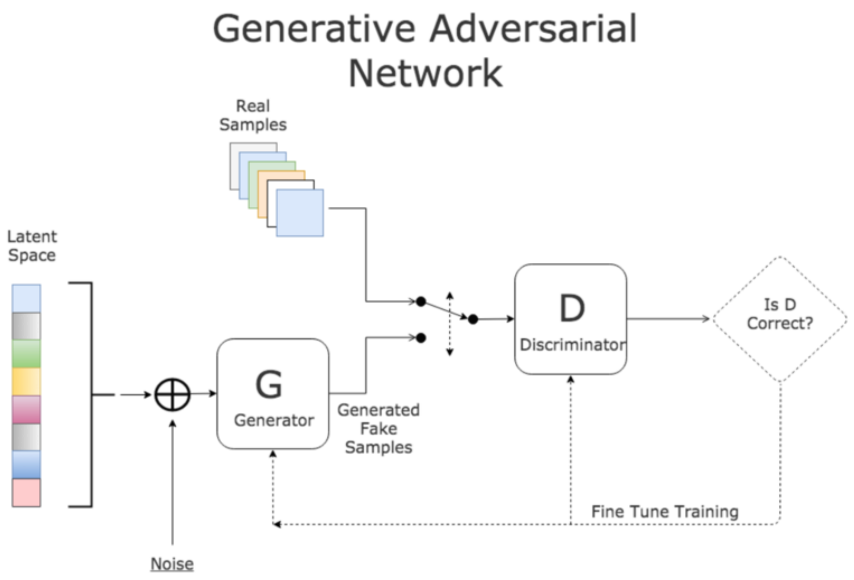


Figure 2.3: GAN Architecture - from Chaudhari et al. (2020)

can tell if the discriminator has been "fooled" or not. The generator assessment is passed onto the discriminator which determines if its weights need to be changed or not. This is shown in figure 2.3.

This process is a zero-sum game, meaning that if the discriminator is successful the generator is penalised (and must thus perform a further training iteration) or vice versa. Ideally the process may terminate once the discriminator is "fooled" half the time; i.e it considers all its' input sequences (both real and fake) as being real.

The generator seeks to minimise the cross-entropy loss given by Equation 2.1

$$\underset{G}{min}\, V(G) = \mathbb{E}_{x \sim p_{data(x)}}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))] \tag{2.1}$$

The discriminator seeks to maximise the cross-entropy loss given by Equation 2.2

$$\underset{D}{max}\, V(D) = \mathbb{E}_{x \sim p_{data(x)}}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))] \tag{2.2}$$

The formal equation representing the overall GAN minimax Loss function is given in Equation 2.3.

$$\underset{G}{min}\,\underset{D}{max}\, V(D, G) = \mathbb{E}_{x \sim p_{data(x)}}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))] \tag{2.3}$$

The symbols used are as follows:

D = Discriminator
G = Generator
$Pz(z)$ = Input noise distribution
$Pdata(x)$ = Original data distribution
$Pg(x)$ = Generated distribution
$\mathbb{E}$ = Expectation

The left hand side of the equation means that the discriminator loss function is maximised whereas the generator loss function is minimised.

The generator stage of a GAN is seeded by a fixed length vector that comprises a series of normally distributed random numbers. This vector functions in a similar fashion to the latent space in autoencoders and goes by the same name. Once the training phase has completed the various portions of the latent space will generate specific parts of the synthetic object. Thus, seeding the latent space with a different random vector will generate a new object that preserves the stochastic behaviour of the objects the GAN was trained on. The latent space may therefore be seen as a "compression" of the high level characteristics found in the training data.

The generator is the final product of the GAN system since it alone can be used to produce new synthetic data. This means that after training the discriminator may be discarded.

We will now give a brief overview of supervised and unsupervised learning as they are typically understood in ML literature.

2.5.2 Supervised vs unsupervised learning

The discriminator and generator of a GAN belong to two different classes of ML models. The discriminator is an example of a predictive model, namely a classifier. Typically classifiers are trained on labeled data and this labeling is a process that requires human action and so classifiers are categorised as supervised models. Classifiers usually have to distinguish between objects that fall under a number of different classes but in the case of GAN discriminators they perform a simpler binary operation; i.e. is the input data real or fake?

The use of labelled or tagged data is the fundamental characteristic that distinguishes supervised learning in machine learning. This data is used to train or “supervise” algorithms which classify data or make predictions. The resulting outputs are then evaluated using any number of metrics that will be discussed further down.

During training in supervised learning, the algorithm “learns” from the training dataset by making predictions iteratively on the input data and adjusting progressively towards the correct answer. Supervised learning models are typically more accurate than unsupervised learning models (Boden, 1996). Since human intervention is required to tag the data, this sometimes causes misclassification errors.

Unsupervised learning models, work autonomously to discover the structure and layout of the input data. Some human intervention may still be needed for validation of the output, but the process usually gives a good starting point for human operators to elaborate. These clustering and classification systems are used to suggest to the customers of online retailers, like Amazon, other products they might be interested in. Unsupervised learning models are used principally in: clustering, association and dimensionality reduction.

A subclass of unsupervised learning is that where the model’s function is to summarise the statistical distribution of the input data so that a data set with analogous statistical characteristics may be generated. These models are known as generative models. A very simple generative model that operates on Gaussian distributions will simply measure the mean and standard deviation of the input and produce a random number sequence fitting those characteristics. This is the general principle behind generative models. Generative models may not only produce plausible data sets but have produced data sets that have proven to be indistinguishable from the real object (be it images or time sequences like sound and financial data)

2.5.3 Generator technologies

Apart from GANs there are a number of other techniques that have been used as generative models. These generator models may be seen to fall under the two main technology classes that underly machine learning: statistical methods and neural networks. We will review three statistical methods; Naive Bayes, Latent Dirichlet Allocation and Gaussian mixture models.

Before looking at Naive Bayes we will first describe an even simpler statistical method that arises from the use of the normal distribution. If we are to model a system that exhibits one random variable that follows the normal distribution, then we could model that system by establishing the mean and standard deviation of the system. A generative model that simulates the given system is simply a random number generator whose output are data points that fit in a Gaussian with the given mean and standard deviation. Thus, this generator's data will be a table consisting of one column.

A Naive Bayes generator model caters for multivariate data points, i.e. datasets whose elements have a number of different characteristics, like the vehicle population in a city. Vehicles will have a number of characteristic features; Type (sedan, truck, motorbike, etc), make, colour, Engine capacity, seating capacity and possibly other features. Data may be collected in a multicolumn database. The Naive Bayes generator is to generate a similar table whose element distribution follows that of the real database.

A Naive Bayes classifier gives an estimate of the identity of a vehicle whose characteristics were not in the training data. It does this by first calculating the probability of each different element type, for example the probability of a vehicle being a sedan, truck or motorbike from the actual data. The Naive Bayes algorithm is simply the multiplication of the of the characteristics being considered, for example, the probability for bhp multiplied by the probability for top speed if those are the only characteristics in the classification query. Thus the classifier will be able to tell that a car is a Ferrari given the engine power and top speed because the probability for 400bhp and 300 km/h is far higher for a Ferrari than any other car type. The Naive Bayes generator ensures that the population of elements in the generated table follows the probability distributions of the original data.

Latent Dirichlet Allocation that was developed by Blei et al. (2003) is typically used to classify texts. In the case of texts the allocation works over three levels; words make up books and books belong under topics. The distribution of words will differ depending on the topic the book is dealing with. Evidently, medical books will contain medical terms and law books will contain legal terms. Thus, once books have been classified under certain topics in the training phase, the respective word distributions will be established. Given these statistics an unclassified text may be given different probability scores that it be-

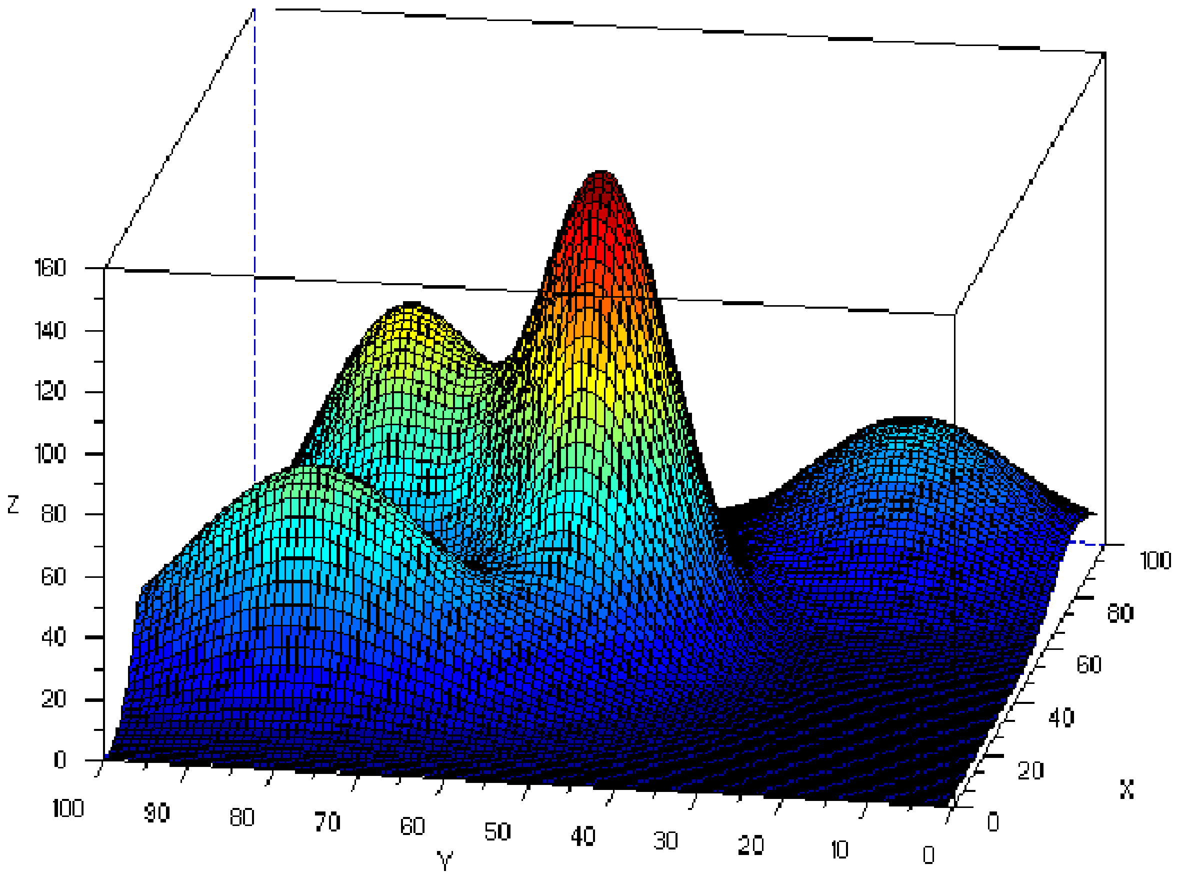


Figure 2.4: Gaussian Mixture Model taken from Takagi and Pallez (2009)

longs to a series of topics, resulting in an allocation with the highest probability. Similarly to the data generation using Naive Bayes one may generate a text corpus that simulates certain topics using LDA by ensuring that the word distributions in the generated texts follows those in real texts.

Gaussian Mixture Models (GMM) are fundamentally a clustering technique. The pre-eminent clustering technique is k-means clustering were the algorithm will identify groups of elements that have similar characteristics. The shortcoming of the k-means method is that it will be unable to identify superimposed clusters. GMMs overcome this problem. The algorithm is similar to k-means in that it starts by randomly superimposing a number (k) n-dimensional Gaussians over the target dataset. Let us say that each Gaussian is given an identifying colour, red and blue, for example. Each point on the dataset is attributed the probability of belonging to the superimposed Gaussian thus having a red component and a blue component. As a second step, new red and blue Gaussians are superimposed on the data elements where this time the red Gaussian covers the red "spot" distribution and the blue Gaussian covers blue spots. This second step causes the distributions to shift. The underlying data points are again given red and blue weights as in the first step. The process is iterated until the overlying Gaussians stop moving. This will result in superimposed Gaussians as seen in figure 2.4. Here again, once the GMM process is performed on the given data, the generation process to simulate the data will simply produce data that follows the Gaussian distributions as ascertained by the GMM process.

Now we move to Neural Network Generators. Hinton and Sejnowski (1983) developed the Restricted Boltzmann Machines (RBM) that like VAEs have the ability to learn the statistical characteristics of the input dataset. RBMs are based on the computationally intensive Boltzmann Machine but operate with a reduced number of connections. Restriced Boltzman Machines (RBM) may be used as data generators but are the constituent element of the more complex Deep Belief Network (DBN). The RBM is a two layer network were the first layer nodes are the visible nodes and the second layer nodes are called the hidden nodes. Each visible neuron is connected to all the hidden neurons but not to the other visible neurons. The hidden neurons are also not connected among themselves. An RBM may be seen as someone looking at himself in a reflecting surface whose image is constantly improving as the light photons bounce back and forth from the subject to the image. Thus, an RBM will transmit the inputs at the visible layer onto the hidden layer through weighted connections. These hidden neurons will simply transmit the activation they receive back to the visible neurons over the weighted connections. The loss function to be minimised is thus the difference at the visible neurons of the input signal and the signal received from the hidden neurons. Once this settles to a minimum, (i.e. the signal coming from the hidden neurons is very similar to the input) the RBM will act as a generator that has the characteristics of the original input. Liang et al. (2017) used this technique to predict the movement of the SSE Composite and the FTSE100 indices by generating data with new features. Random Forest, SVM and Logistic Regression were used as classifiers trained both on real and generated data achieving an accuracy of 61%. Li et al. (2017) used a similar system trained on the S&P500.

DBNs were also developed by Hinton et al. (2006) and are simply stacked RBMs were the hidden layer of the first RBM is attached to the visible layer of the second RBM. This stacking can be repeated to give an N layer DBN. Each layer is trained independently, so when the first layer is trained it is frozen and the second layer RBM is then trained. The process carries on until all the RBM layers are trained. The fully trained DBN is then used as a generator.

Autoencoders have been overviewed in section 2.5.1. Another architecture derived from Autoencoders is the Variational Autoencoder (VAE). Kingma and Ba (2014) proposed this solution where the encoding stage is regularised explicitly to follow the distribution of the input samples. This is achieved by adopting a loss function that is a development over the loss function in the original autoencoder that minimised the loss between the output and input distributions. One shortcoming of the standard autoencoder was the way the elements in the latent space were set with no particular scheme resulting in an essentially random structure. Thus two adjacent latent space elements would result in two completely different output morphologies. The variational autoencoder adopts a scheme whereby the latent space is given a coherent structure. One way to understand this is to imagine a 2 dimensional colour chart representing the latent space of a variational encoder where adjacent pixels would have similar colours and moving around on the colour chart produces smooth changes. This is what is meant by having an enforced regularisation of the latent space. On the other hand the latent space of the standard autoencoder would have the colour pixels being randomly distributed with no structure. The regularisation used in the variational autoencoder is achieved by having gaussian distributions being encoded into the latent space rather than single values. The VAE loss function still minimises the difference between the input and output data but also minimises the difference between the encoded gaussians by using the Kulback-Liebler divergence.

Gu et al. (2019) used a VAE to generate data with specific market and macroeconomic characteristics. This data was used to train an S&P500 index prediction system based on the LSTM architecture. These predictions were then fed into a trading algorithm. Gu et al. (2019) reported a prediction accuracy close to 83%.

2.5.4 Problems manifested by these early GANs

Particularly at the initial stages of development the operation of GANs was known to present a number of difficulties. The information below has been drawn from Wang and Yan (2021) and Jabbar et al. (2021):

- Nash-equilibrium that leads to non-convergence: the model parameters oscillate and never converge. Nash-equilibrium is structural to GANs in that the discriminator and generator should be seen as playing a zero sum game where one is trying to gain advantage over the other. This process is intrinsically unstable and is at the heart of the difficulty in using GANs.
- Mode collapse: the generator produces a limited variety of samples. This happens when the generator and discriminator cycle through a limited set of solutions because of local minima in their respective loss functions.
- Vanishing gradient: the discriminator outperforms the generator and leads to vanishing gradient.
- Unbalance between the generator and discriminator causing overfitting.

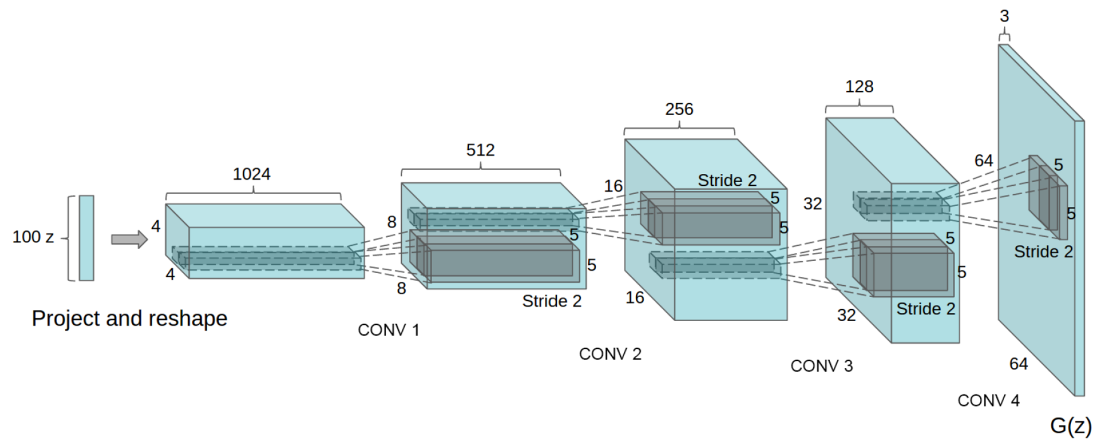


Figure 2.5: DCGAN Generator - from Liu et al. (2018a)

• High sensitivity to the hyperparameter selections: During development the operator may cause instabilities by selecting hyperparameters aggressively.

• Internal covariate shift: This is caused by inputs having a different statistical distribution to the ones in prior training that leads to a slowdown in learning rate.

• Lack of proper evaluation metrics.

2.5.5 Solutions adopted to overcome early problems

The DCGAN is included in this overview because it is considered to be the baseline GAN architecture. The diagram in figure 2.5 shows the structure of the generator and it is taken from the original DCGAN paper by Radford et al. (2015). As can be seen it is a canonical 5 layer CNN.

Various authors have explored a number of solutions many of which have been incorporated in the more recent implementations of GANs that are listed in Table 2.1

2.6 Prediction Algorithms

The concept of algorithms is integral to the understanding and function of ML in finance. Algorithms may be seen to be a set of rules that are applied sequentially. They find application in mathematics and computer science and are usually associated with the solution of a specific problem. Simply put, they take an input, and produce an output. Once a

GAN Technique	Paper(s)
Normalization techniques	Salimans and Kingma (2016); Salimans et al. (2016); Miyato et al. (2018)
Mini-batch discrimination	Salimans et al. (2016)
Label smoothing	Szegedy et al. (2016)
Alternative loss functions	Mao et al. (2017)
Adding noise to inputs	Arjovsky and Bottou (2017); Sønderby et al. (2016)
Hybrid model	Brock et al. (2016)
Feature matching	Salimans et al. (2016)
Historical averaging	Salimans et al. (2016)
Proper optimizer	Kingma and Ba (2014); Tieleman and Hinton (2012)
Unrolled GAN	Metz et al. (2016)
Two time-scale update rule	Heusel et al. (2017)
Using labels	Antipov et al. (2017); Kingma and Welling (2013); Liu et al. (2018b)
Gradient penalty	Gulrajani et al. (2017)
Cycle-consistency loss	Zhu et al. (2017)
Self-attention GAN	Zhang et al. (2019a)
Relativistic GAN	Jolicoeur-Martineau (2018)
Sampling GAN	White (2016)

Table 2.1: Solutions to improve GAN performance

problem is identified, and a set of steps are seen as a potential solution, the procedure is tested using data that covers as many potential scenarios as possible. The range of algorithms is extensive as are the various taxonomies used to classify them. In both academic and business realms involved in finance and stock markets, the range of algorithms used is considerable (Kim et al., 2010).

There are two main classes of algorithms used in ML specifically in deep learning. The first is characterised by the use of back propagation. As mentioned above this uses gradient descent were an error component arising from the training stage of an NN is gradually reduced until it stabilises to a minimum value. The most prevalent area of application of this class of algorithms is market forecasting and prediction. A second class of algorithms are those that promote self organisation through competitive learning that is used for

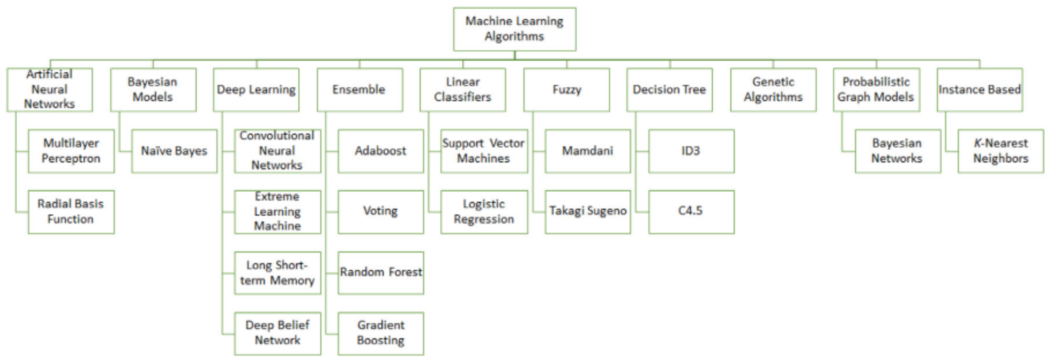


Figure 2.6: Prediction Algorithm Taxonomy - Taken from Bustos and Pomares-Quimbaya (2020)

clustering and segmentation of financial markets into areas of similar behaviour. (Tiwari et al., 2020).

2.6.1 Prediction of asset prices or trends

A number of authors have studied the difficulty that financial prediction entails. Araujo and Ferreira (2013) state that its not only prices that are the issue in financial prediction but other characteristics like volatility, noise and changes in trends are also critical. Tay and Cao (2001a); Zhang et al. (2017) state that one of the problems lies with the fact that financial data is non stationary. Bezerra and Albuquerque (2017); Göçken et al. (2016); Kumar and Thenmozhi (2014) identify the dynamic, chaotic and noisy nature of asset prices as being the main issues causing most trouble. This behavior is the result of the underlying economy, government behavior and the erratic nature of individual investors (Chen et al., 2017; Zhong and Enke, 2017). Technical analysis (section 2.1.2) and fundamental analysis (section 2.1.1) are two approaches that human analysts have adopted in trying to predict asset prices. The non ML techniques used by these analysts include autoregressive models, discriminating analyses and correlations (Kumar and Thenmozhi, 2014; Wang et al., 2012)

Figure 2.6 shows a taxonomy by Bustos and Pomares-Quimbaya (2020) of the scope of market prediction algorithms. A brief overview of the more prevalent models is being given here.

White (1988), implemented the earliest market prediction model, according to Yoo et al. (2005), using a Feed Forward neural network. This attempt set out to search for recurring patterns in asset prices that would have been missed by human analysts, thus laying the foundations for what would become a fundamental tool in the finance industry.

Statistical techniques: Prior to the uptake of machine learning methods, prediction was accomplished using statistical methods that were used to produce correlations between economic phenomena like inflation, interest rates and the stock market itself. Maddala (1992) in his introductory ecomomics text book states that these correlations were established using univariate and multivariate regression. Multivariate may be seen as univariate regressions that are able to discern more than one causal factor (Van Eyden, 1996). One of the initial and frequently used techniques is that by Box and Jenkins (1976). They outlined the ARMA method that is the precursor to the autoregressive integrated moving average that is described in section 2.6.3. Lawrence (1997) described some issues regarding the Box-Jenkins method where he states that it necessitates the use of large data sets and is best suited for short term prediction. Generalized autoregressive conditional heteroskedasticity (GARCH) by Bollerslev (1986) is another well researched non-ML technique.

Genetic algorithms: Genetic Algorithms (GA), in their present form were invented by John Holland in the 1960's and were refined by him and his students in the 1960's and 70's (Holland, 1975) at the University of Michigan. Holland's original goal was not to have GA's solve specific problems, he was more interested in developing a formal theory based on adaptation as seen in nature that could be integrated into future computer systems. In his 1975 book "Adaptation in Natural and Artificial Systems" Holland (1975) sees GA's as an abstraction of natural evolution and gives a theoretical framework for GA's. GA's, as conceived by Holland, are a method for traversing from one population of "chromosomes" (strings of ones and zeros, or "bits ") to a previously unseen population by using "natural selection" methods together with the operations of crossover, mutation , and inversion as seen in natural genetics.

In 2000 Phua et al. (2000) used a combination of NNs and GA to forecast the Singapore stock exchange and reported 81% accuracy. In the same year Kim and Han (2000) used an architecture similar to phua predicting weekly price rises and falls of of the Korea Stock price Index 200 resulting in a stated 82% accuracy. Goldberg (1989) also reports using GAs. GAs have been used to improve the performance of NNs in the selection of the appropriate architecture, feature set optimisation and the determination of the number of hidden layers and their width (Yoo et al., 2005). Other researchers that have used GAs together with NNs in financial prediction are Kimoto et al. (1990) and Nikolopoulos and Fellrath (1994)

A technology that is somewhat similar to GAs is Particle Swarm Optimisation (PSO) (Majhi et al., 2008).

Fuzzy logic: Fuzzy Logic tries to imitate the way humans reason. Fuzzy Logic consists in the formulation of statements that resemble if-then rules, where the premises use category sets instead of unitary values. Fuzzy Logic is often used to distil rules arising from human expertise. Adaptive neuro-fuzzy inference system (ANFIS) is an algorithm that usually used in the construction of such (expert) systems.

Fuzzy logic has been used by Appadoo (2006); Hassan (2009); Thavaneswaran et al. (2007) in their pricing models. Carlsson and Fullér (2001) and Thavaneswaran et al. (2009) have employed fuzzy techniques to estimate initial stock prices (Zhou et al., 2018)

Rough sets: Rough set theory is a mathematical technique to estimate uncertain data that is similar to fuzzy sets. Kim et al. (2017) used Rough Sets to forecast the Korea Composite Stock Price Index 20 0 (KOSPI). Their system also made use of other techniques, like genetic algorithms and various trading strategies. When compared to other trading strategies they used as benchmarks (e.g. buy and hold) Rough Sets performed better than the benchmarks.

Support Vector Machines: SVMs have been developed in the 1970s by Vapnik (1999). The algorithm will find a hyperplane (i.e. a boundary in higher dimensions) that renders the distance between two sets of input data as large as possible using what is known as the kernel trick (Bustos and Pomares-Quimbaya, 2020). SVMs use structural risk minimisation that explicitly reduces generalisation error resulting in better generalisation performance than other techniques like NNs. The SVM technique is the mostly used as a linear separation algorithm because it is essentially devoid of parameters. SVMs have been shown to have similar or better performance than other more complex algorithms (Yoo et al., 2005).

SVM have drawn a great deal of attention because of their ability in classification and regression tasks, particularly in time series like asset prices (Yang et al., 2002). Support Vector Machines are used in both classification and regression systems. SVMs have been noted for their ability to avoid overfitting, have good generalisation ability and are able to work away from local minima that habitually cause problems for NN loss functions. Kim (2003); Tay and Cao (2001a,b) have reported a number of instances where SVMs have been used successfully. Kim (2003) compared SVMs to NNs and Case based Reasoning and stated that SVMs gave the best results.

Neural networks: As seen above, NNs have been shown to perform better that traditional statistical techniques in non-linear systems like financial markets (Lawrence, 1997). NNs have a number of characteristics that make them suitable for financial prediction. They are able to extract relationships between features inherent in the data to be analysed without these needing to be formally identified. Their accuracy is primarily dependent on the amount of data they are trained on. They exhibit a level of tolerance to noisy and incomplete data. They also manage to extract non-linear relationships in data and are intrinsically non-parametric (Yoo et al., 2005). All these characteristics make them particularly suited to financial prediction since data coming from these sources have a high level of complexity that makes financial data otherwise hard to model. As Lawrence (1997) states NNs will not make explicit the relationships that are extracted from the input data (black box problem) hiding the significance of the variables on which the relationships depend. Thus traders and analysts will not be able to use these relationships in their own (non ML) trading strategies (Lawrence, 1997).

(Garliauskas, 1999; Schumann and Lohrbach, 1993; Yoon and Swales, 1991) have looked into the performance of NNs when compared to the more conventional statistical methods employed up to then. A comparison of NNs with discriminant analysis was carried out by Yoon et al. (1993) and found that NNs performed better. Garliauskas (1999) used NNs in combination with the 'kernel trick' and recursive prediction. This study also established that NNs perform better than classical statistical methods. The findings of Kim et al. (1993); Patuwo et al. (1993); Subramanian et al. (1993); Yoon and Swales (1991) came to similar conclusions.

A number of researchers have established that NN's can perform better than multivariate regression models (Garliauskas, 1999; Schumann and Lohrbach, 1993). Lawrence (1997) used a GA/NN combination to forecast the Johannesburg Stock Exchange achieving an impressive 92% accuracy. This was in contrast with the Box-Jenkins that achieved 60%. Refenes et al. (1995) and Steiner and Wittkemper (1995) have also reported similar results showing the superiority of NNs to multiple linear regression. Another study asserting similar results is that carried out by Yoon et al. (1993) where NN's achieved 91% accuracy against 74% using multiple discriminant analysis.

NNs have been used in conjunction with other ML techniques. Hiemstra (1995) used NN's with fuzzy expert systems using fuzzy logic that, again, avoid the explicit identification of the inherent variable relationships. NNs together with rule based technique was used by Tsaih et al. (1998) to forecast daily direction change in the S&P500 index futures.

Deep learning: Deep learning models are a subset of neural networks, they were intentionally classified in this way to differentiate from traditional neural networks that do require feature engineering (Bustos and Pomares-Quimbaya, 2020).

Code Based Reasoning: Code Based Reasoning is often used to build expert systems. Expert systems attempt to convert the knowledge held by human experts into what is essentially a cascading list of if/then statements. The principal advantage of CBRs over NNs is their ability to provide an explanation into how the results they produce were accomplished. This is a characteristic the NNs are often criticised for. The tendency for NNs' loss functions to get caught in local minima and their dependence on their stochastic behaviour (that gives different results on separate runs) causes CBR solutions to be sometimes preferred (Trippi and Turban, 1992)

A system that combined GAs with a CBR was implemented by Kim (2004). Motivations for this solution were the optimisation of feature weighting and subset selection to be used in the CBR system. The paper states that a simultaneous optimisation that uses both feature weighting optimisation and subset optimisation performs better than systems that adopt these optimisations separately or the unoptimised CBR.

Boosting, Bagging, and Stacking: Boosting is a meta-ensemble technique where models are trained by varying the distribution of training data. Boosting will increase the weight of training samples that failed to classify properly. McCluskey and Liu (2017) carried out a study using Boosted Trees. The AdaBoost algorithm was used by Huang et al. (2018) to model the U.S. stock market to improve a Naive Bayesian classifier.

Bagging or Bootstrap Aggregation techniques do not make changes to the data distribution but use subsets of the training data selected randomly. One of the most often used Bagging algorithms is the Random Forest (RF). This algorithm entails the training of a specific number of decision trees, that use diferent features randomly. There is a large number of studies on stock market prediction that show that Random forests, namely; Kamble (2017); Labiad et al. (2016); Patel et al. (2015); Napate et al. (2020); Zhang et al. (2018). In particular Patel et al. (2015) showed that the Random Forests performs better than SVMs with regard to stock price prediction.

Other ML based financial prediction technologies include Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Decision Support System (DSS), Hidden Markov Model (HMM), Naive Bayes (NB), Support Vector Regression (SVR).

2.6.2 Technical Analysis Indicators

The benchmark paper (YU and Li, 2018) used in this dissertation makes use of the moving average that is an indicator that is very much used in the field of financial trading. Volatility is also a factor that is used in the benchmark model. Below is a short description of the prevalent classes of indicators:

Trend indicators: Trend indicators will help identify which direction the market is moving in. They are also known as oscillators, since they move between high and low values in a wavelike fashion.

Momentum indicators: Momentum indicators give an indication as to the strength of the trend in question. These indicators can also help identify whether a reversal is likely to happen as well as price peaks and troughs.

Volume indicators: Volume is an important quantity that is the subject of analysis using indicators as we have seen with prices. Volume indicators are used to asses price changes since the volume can give a good indication of how significant the change is, where high volumes show there is "bullish" sentiment on the given stock in that period.

Volatility indicators: Volatility is a very important characteristic in finance as it shows instability and increased activity. Most profit making opportunities happen when there is increased volatility as price movements are limited when volatility is low.

A good list of technical indicators is given by Achelis (2001).

The two prediction models used in the benchmark paper (ARIMA and RNN) will now be described.

2.6.3 Arima

The first model is the autoregressive integrated moving average (ARIMA).

The development of time series analysis and machine learning techniques has lead to their application in the prediction of asset prices and trends on stock markets. Work has been ongoing in the investigation of time series models such as ARIMA (Autoregressive Integrated Moving Average Model) that has been proposed by Box and Jenkins (1976).

The ARIMA technique gives good results when the data series is linear or at least approximately linear, but may not be appropriate for forecasting future fluctuations if the time series is nonlinear (Schmidhuber, 2015) as is often the case with stock market prices. The fact that linear models like ARIMA may not be applicable for the analysis of nonlinear time series that are often encountered, has lead researchers to look into nonlinear models such as SVM (support vector machine) and neural networks.

It is best to explain the ARIMA model by describing each of its components:

Autoregression (AR): An AR model is one that focuses on a variable that correlates with its own lagged values.

Integrated (I): Time series data, particularly asset prices, will exhibit ramping up or ramping down of the data points over time. A time series with no ramping is known as being "stationary". The "integrated" (I) term in ARIMA indicates that the time series is differenced such that it becomes stationary.

Moving average (MA): The moving average for a particular data point in a time series is the average calculated on a arbitrary number of data points prior to the point of interest. The observation of the asset price plot with the moving average plot superimposed is one of the most often used tools by quantitative analysts.

The ARIMA model (pdarima) is used in this paper to give a low frequency (i.e. low volatility) baseline This algorithm finds the best fit for the ARIMA parameters p, q and d. The differencing order used by the algorithm is ascertained using Phillips–Perron, Kwiatkowski–Phillips–Schmidt–Shin, or Augmented Dickey-Fuller. The result of the ARIMA algorithm is then superimposed (added) to an RNN generated high frequency component resulting in the final predicted time sequence.

2.6.4 RNN

Essentially RNNs are a set of daisy chained neural networks where the output of one network will feed into the input of the next. This arrangement allows sequences of input data to be processed rather than static data sets. Thus systems that are time varying may be processed to extract time dependent patterns.

2.6.5 The Benchmark paper prediction algorithm

The first thing the algorithm does is to find the best Moving average (MA) period by selecting the one where the resulting time sequence has kurtosis closest to 3. This is done so that the MA sequence is as close to normal as possible. Once the optimal period is found a low volatility time sequence is generated. This is simply the moving average of the input sequence.

A high volatility time sequence is generated by taking the difference of the input sequence and the low volatility sequence mentioned above. Low volatility and high volatility time sequences can be seen as the input sequence being passed through a low frequency filter (thus giving a smoothed curve) and its high frequency residual.

The prediction error is calculated by adding two components; the ARIMA prediction error and the RNN prediction error as described below.

The ARIMA pass is done by generating an ARIMA model on the low volatility curve (i.e. the moving average). This is done using the pmdarima python package. The MSE, MAPE and MAE metrics between the input sequence and the prediction are calculated using the sklearn package. The ARIMA prediction is done by using the model to predict the next day on a 250 day sequence of input (i.e. real) test data that is increased by one day for each successive prediction. This process requires a time series of closing prices.

The RNN consists of a 4 neuron input layer followed by a 2 neuron intermediate layer feeding an dense single neuron output. The Keras library is used to implement the neural network and its' early stopping function was used to optimise speed performance and reduced the risk of over-fiiting. The high frequency (volatility) time series is pre-processed to bring its amplitude to between 0 and 1.

The RNN error metrics use, at their fundamental level, 4 day sequences as the inputs to the RNN. The RNN in this algorithm does not work on the original time sequence but on a delta of the original minus the low volatility sequence. Thus, implicitly it requires the result that was obtained using the original time sequence.

The MSE, MAPE and MAE metrics obtained from the two passes are then added to get a final metric. Twenty instances of the code were run asynchronously using a shell script as this was the number of simultaneous runs the hardware could sustain without significant performance degradation of any of the instances.

2.7 Evaluation

The issue of evaluation, particularly in the appraisal of GANs has been discussed by several authors (Barua, 2019; Borji, 2019; Ducoffe et al., 2019; Eckerli, 2021; Jabbar et al., 2021; Koochali et al., 2020; Simonetto, 2018). These authors all concur that GAN evaluation is a challenge. As will be seen in the discussion of the selected GAN architectures (section 2.10), the signature transform does provide a metric that measures how close the synthetic output of a GAN matches the characteristics of the real training data. This notwithstanding, all other surveyed GAN architectures will exhibit a flat loss function after a minimal number of training cycles. This means that the discriminators used in these GANs struggle to distinguish the generated data set from the original data set.

The difficulty in finding a suitable discriminator algorithm implies that a suitable numeric evaluation method is a major concern. This is true for GANs in their primary original application, i.e. image creation, and in other utilisations like finance. A number of authors have thus resorted to qualitative evaluations, including visualisations.

Before embarking on a description of evaluation techniques that are suitable for financial time sequences, it would be pertinent to give a brief overview of the work done on the evaluation methods developed to appraise GANs used in computer vision as this is the matrix from which Financial time sequence GANs originated. For this purpose we have included a list of CV evaluation methods taken from Borji (2019) in Appendix C.

2.7.1 Evaluation Methods Survey

Quoting Wang et al. (2021): "Previous work has proposed various evaluation metrics for GANs (Barratt and Sharma, 2018; Borji, 2019; Gulrajani et al., 2017; Hartmann et al., 2018; Heusel et al., 2017; Theis et al., 2015; Wang et al., 2020a,b; Xu et al., 2018) and it is an active area of research."

	Time Series	Finance
Borovykh et al. (2017)	x	
Eckerli (2021)		x
Smith and Smith (2020)	x	
Leznik et al. (2021)	x	
Efimov et al. (2020)		x
Sun et al. (2020)	x	x
Hogenboom (2020)		x
Zhang and Khoreva (2019)	x	
Franco-Pedroso et al. (2019)		x
Wiese et al. (2020)		x
Ni et al. (2020)	x	x
Takahashi et al. (2019)		x
Yoon et al. (2019)	x	

Table 2.2: Time Series Evaluation Methods

Tables 2.2 and 2.3 represent the distribution of the different evaluation methods used in papers surveyed of the authors shown in the first column. The column "Stats" in Table 2.3 indicates the adoption of various statistical metrics in the respective papers. Table 2.2 shows whether the paper mainly dealt with general "Time Series" or time series in "Finance".

Table 2.4 illustrates the range of the various statistical metrics (first column) that were used in the surveyed papers and the range of visualisations (second column) that have

	Predict	Wasser.	Stats	Fid	Entropy	ACF	Bespoke	Visual
Borovykh et al. (2017)	x							x
Eckerli (2021)			x			x		x
Smith and Smith (2020)				x				x
Leznik et al. (2021)			x		x			x
Efimov et al. (2020)			x				x	x
Sun et al. (2020)		x						x
Hogenboom (2020)			x					
Zhang and Khoreva (2019)				x				
Franco-Pedroso et al. (2019)			x			x		x
Wiese et al. (2020)		x				x		x
Ni et al. (2020)			x			x		x
Takahashi et al. (2019)			x					x
Yoon et al. (2019)	x							x

Table 2.3: General Time Series Evaluation Methods

been used by the authors of the papers surveyed.

"Bespoke" in Table 2.3 refers to the "DataQC" application developed by American Express that uses a mix of the metrics illustrated above specifically to assess the quality of generated data.

Numeric	Visual
Mean	Q-Q plot
Std-Dev	Histogram
Entropy	ACF
Asymmetry	PCA
Approx entropy	t-SNE
Granger	Heavy Tails
Johansen	Volatility Clustering
P-values	Gain/ Loss
chi-squared	
Skewness	
Kurtosis	
Extreme Values	

Table 2.4: Numeric and Visual Evaluation Methods

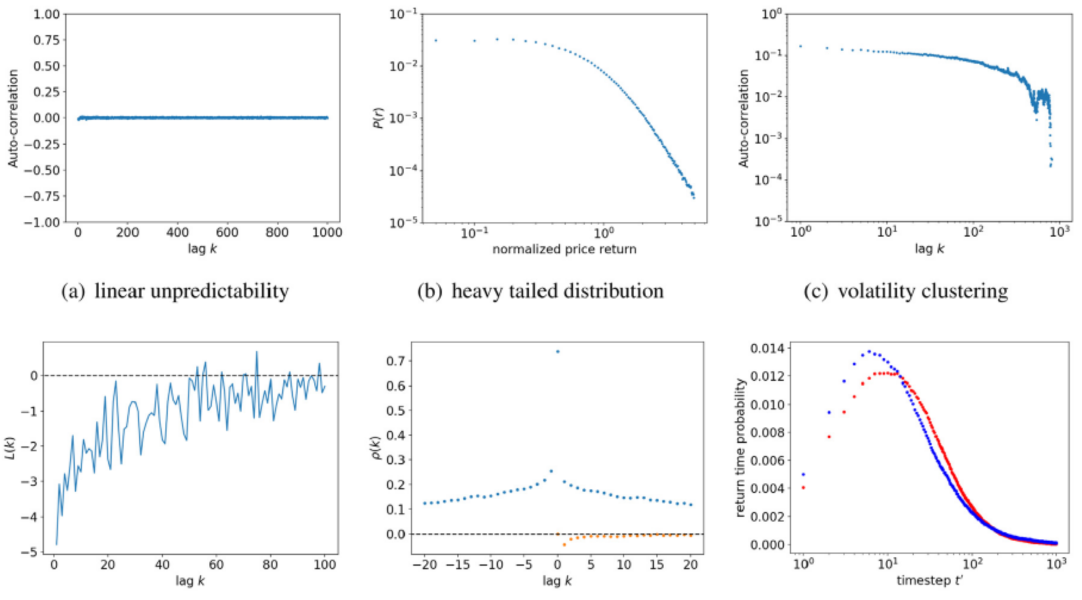
2.7.2 Visual Evaluations

What follows is a technical overview of stylised facts with their plots. The techniques described in the papers by Takahashi et al. (2019) and Eckerli (2021) formed the basis of the work presented here. Takahashi in particular, bases his analysis on "stylised facts" rather than standard statistical measures as these will capture the characteristic behaviour that is exhibited in financial price sequences and are absent from random walks based on the normal distribution. Takahashi et al. (2019) bases his visual evaluation metrics on the paper by Chakraborti et al. (2011). The metrics described below give a good indication on the quality of generated financial time series and parties wishing to use synthetic time series should be accustomed to these metrics.

Here a representation of the relevant stylised facts in financial time series that appears in the Takahashi paper (Takahashi et al., 2019) is given.

Linear unpredictability: This is an important characteristic of financial time series. It is expressed mathematically as:

$$\frac{E[(r_t - \mu)(t_{t+k} - \mu)]}{\sigma^2} = Corr(r_t, r_{t+k}) \approx 0, for\, k \geq 1 \tag{2.4}$$



(a) linear unpredictability (b) heavy tailed distribution (c) volatility clustering

(d) leverage effect (e) coarse-fine volatility correlation (f) gain/loss asymmetry

Figure 2.7: stylised facts - from Takahashi et al. (2019)

Here μ and σ are the mean and the standard deviation of the daily (or periodic) asset return. The equation represents the auto-correlation function of return. Figure 2.7 (a) shows this behaviour against an increasing daily lag. The flatness of this plot indicates that the S&P500 is quite efficient in the EMH sense (Chakraborti et al., 2011). This flatness is also exhibited by the GAN synthesised time series (section 3.2.6).

Fat-tailed distribution: The plot in figure 2.7 (b) shows Fat tailed distribution. (Chakraborti et al., 2011; Liu et al., 1999) state that the probability distribution of the asset returns P(r) has been thoroughly analysed in various asset domains using various time scales. It has been observed that the probability distribution P(r) exhibits a power-law decay in the tails:

$$P(r) \propto r^{-\alpha} \tag{2.5}$$

The exponent has been observed to have a range of $3 \leq \alpha$.

Volatility clustering: Even though there is very little, if any, auto-correlation of the price returns (2.7 (a)), volatility clustering is still visible as figure 2.7 (c) shows. This phenomenon can also be observed in figure 4.7, that shows returns, where it is evident that price instability occurs in clusters of several days rather than separate individual days. By observing these plots one can see that of periods of high volatility and low volatility are often bundled together, and this is what renders the stylised fact of volatility clustering so relevant. As can be seen from the plots of returns generated by the selected GANs (section 3.2.6) this phenomenon is also visible and is similar to what is observed in the real S&P500 returns plot in figure. 4.7.

Quantitatively, volatility clustering may be defined by the auto-correlation of the price returns that follow power-law decay:

$$Corr(|r_t|, |r_{t+k}|) \propto k^{-\beta} \tag{2.6}$$

It has been observed that in S&P500 data the exponent β ranges between 0.1 and 0.5.

Leverage effect: Leverage effect (2.7 (d)) refers to a tendency that has been observed in asset price behaviour where the asset volatility is correlated with its returns, i.e. the price rises when volatility diminishes and v.v. (Bouchaud et al., 2001). Bouchaud et al. (2001); Qiu et al. (2006) state that this statistical property can be expressed mathematically as the lead–lag correlation function in:

$$L(k) = \frac{E[rt(t)|r(t+k)^2| - r(t)|r(t)|^2]}{E[|r(t)|^2]^2} \tag{2.7}$$

Qiu states that leverage effect is dependent on the observed market where the German DAX clearly exhibits this behaviour and Chinese markets tend to show the reverse i.e. the anti-leverage effect.

Coarse-fine volatility correlation: The coarse volatility and the fine volatility (2.7 (e) has been defined by Müller et al. (1997) as:

$$v_c^{\tau} = \left| \sum_{i=1}^{\tau} r_{t-1} \right| \tag{2.8}$$

The coarse volatility is the absolute value of the price movement in τ days whereas the fine volatility is the sum of the absolute price return in τ days. The coarse-fine volatility correlation has to be a multi-time-scale analysis of volatility since the effect is hardly discernable with a single time-series at the daily scale.

Gain/loss asymmetry: Gain/loss asymmetry (2.7 (f)) indicates the fact that usually asset prices drop faster than price increases. Jensen et al. (2003) defined this behaviour as:

$$T^t_{wait}(\theta) = \begin{cases} inf\{t' | \log p_{t+t'} - \log p_t <= \theta, t' > 0\} & (\theta > 0) \\ inf\{t' | \log p_{t+t'} - \log p_t <= \theta, t' > 0\} & (\theta < 0) \end{cases} \quad (2.9)$$

where $T^t_{wait}(\theta)$ is the number of time-step t' required to reach the price return θ from the time t

Jensen et al. (2003) state that "This statistical property is also relatively noisy as the clear functional form cannot be seen in a single stock data"

We now turn to the analysis of stylised facts as described by Eckerli (2021).

Analysis of returns: Eckerli (2021) dwells on the statistical behaviour of real and generated time sequences. These characteristics are certainly important in that the generated sequences are inadequate for the job at hand if they do not emulate these factors correctly. Nevertheless, this behaviour is only necessary but not sufficient to satisfy the requirement of synthesised financial time series. The following metrics will still produce the same figures after the time sequence data set has been shuffled. Shuffling will destroy time dependence but preserves the statistical qualities.

2.7.2.1 Probability distribution, Kurtosis and Skewness:

Skewness: Skewness measures the asymmetry of the distribution that is centred around its mean. It measures the probability density that occurs off centre. Since asset prices exhibit gain/loss asymmetry in their log returns a real asset should exhibit a negative skewness because there are usually more large downward shifts in prices than large upward ones.

Kurtosis: Kurtosis measures the aggregate weight of a distribution's tails as compared to the centre of the distribution. In other words it measures how much a given distribution's tails differ when compared to the tails of a gaussian distribution. So it indicates how "fat" tails are; an important characteristic in financial data where these fat tails occur regularly. This happens because these low probability events have a significant effect on financial distributions as opposed to normal distributions that lack "fat tails"

Autocorrelation: Autocorrelation measures the similarity of sequential data to a time shifted version of itself. In view of the linear unpredictability characteristic described above, ACF should be very low.

The fundamental issue that impinges on the applicability of these plots is that they depend both on the objectivity and acuity of the observer. This implies that a clear comparison of the performance of a given synthesis technique has no guarantee of reliability and reproducibility. Thus, it was necessary to search for a quantitative metric that was a reliable predictor of how the synthesised time sequence would perform when compared to the real time sequence.

2.7.3 PCA and t-SNE

What follows is a theoretical overview of two visual evaluation techniques used here, namely, PCA and t-SNE. These two visualisation methods have been adopted by several authors (Arnout et al., 2021; Debnath et al., 2021; Efimov et al., 2020; Fawaz, 2020; Grilli and Santoro, 2020; Lakshminarayanan et al., 2021; Santoro and Grilli, 2022; Snow, 2020). The wide use of these visualisations is because they give a very good indication of the similarity between time series. This is achieved by observing the overlap of the data sets being analysed. This behaviour is essential to the ability to assess whether a generated time series has characteristics that are found in the original data which is being emulated.

2.7.3.1 PCA: Principle Component Analysis (PCA)

Principle component analysis (PCA) is considered to be one of the mainstays of multivariate analysis based on the use of projections. The procedure was first mentioned by Cauchy and given a mathematical treatment by Hotelling (1933).

The main goal of PCA is to obtain a projection of a body of data onto a space of fewer dimensions. A photo of a three dimensional scene is a projection from three dimensions to two that preserves a lot of the information present in the three dimensional space. PCA seeks to do this by projecting a multidimensional (i.e. multivariate) space onto a space composed of a minimum of orthogonal dimensions.

The projection in a given direction must have a maximum variance (i.e. the information in the projection must be maximally spread out along that dimension). One way to visualise this is to imagine taking a video of a billboard as one is driving along a road adjacent to the billboard. There will be a point where the information on the billboard will produce the largest projection on the camera's sensor; that is the point where the variance of the information in the projection is at its maximum.

The PCA process will rank the maximal projections in various directions and the projection with the greatest amount of information will be the first principal component followed, in order, by other projections. This process may be done mathematically using eigenvalues and eigenvectors of the covariance matrix of the data. Projection matrices obtained using eigenvectors derived from an ordered set of eigenvalues will generate a set of principal components. The observation of the resulting PC's facilitates the selection of the more relevant components and the exclusion of the components that have a minimal impact and should thus not be fed into the ML system. This is done because training with data that has no bearing (i.e. data that will not aid the regression) will degrade the performance of the system.

2.7.3.2 t-SNE

Another technique that is used for visualisation of high dimensional data is t-Distributed Stochastic Neighbour Embedding (t-SNE). This algorithm has been developed far more recently than PCA by Van der Maaten and Hinton (2008). It is based on the SNE algorithm developed by Hinton and Roweis (2002) but using the Cauchy t-distribution instead the Gaussian distribution used in the original SNE algorithm. It addresses some of the deficiencies of the PCA method. There are a number of characteristics that distinguish t-SNE from PCA;

- t-SNE is stochastic and accepts non-linear relationships in the data whereas PCA is deterministic (though the probabilistic version of PCA used in this study is stochastic).
- t-SNE preserves the distance of data that is close whereas PCA essentially preserves large pairwise distances.
- t-SNE requires more computing power but will give visualistaions that show areas of similarity more clearly as can be seen from the diagram 2.8 of the two techniques being applied to the MNIST digits classification dataset.

The difference in quality will also be evident in the results section (4) where PCA and t-SNE plots are used to compare the performance of the various GAN architectures that have been investigated.

The t-SNE algorithm uses a similarity metric between pairs of data points both in high dimensional space and in low dimensional space. It then uses a cost function (usually Kullback-Liebler divergence) to optimise the two measures.

t-SNE uses the Gaussian and the t-distribution to transform the Euclidean distances between all data points onto these distributions. This means that the Euclidean distance between two data points is transformed to a measure of proximity rather than separation, i.e. points that are close get a higher "distance" value than those that are further apart. Let's consider the Gaussian distribution of the heights of people in a population. The people whose height is closest to the mean are indicated on the Gaussian curve as having

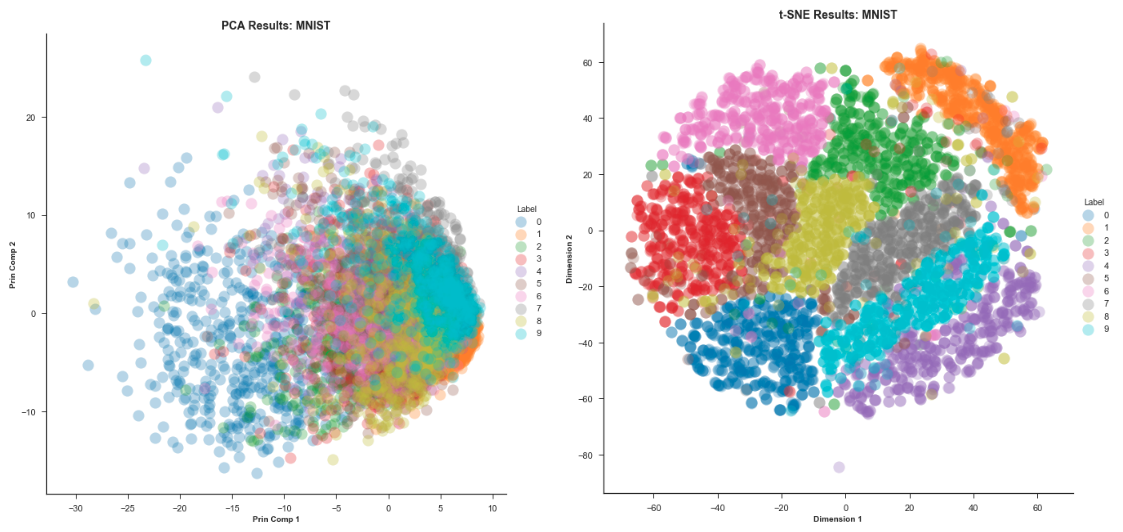


Figure 2.8: MNIST PCA and t-sne plots - from Thakur (2020)

a high ordinate value whereas those that are either exceptionally tall or short will sit on the tails of the distribution and have a low ordinate value. So this transformation will measure the "closeness" of any point with respect to the point of interest. This procedure is performed for each point in the high dimension domain resulting in a "closeness" matrix of all points with respect to all others.

A simple projection of the high dimension data onto two or three dimensions is then performed. The "closeness" transformation is now carried out on this low dimensional projection using the t-distribution instead of the Gaussian. Again this results in a closeness matrix as before. This new matrix will be disordered as the said projection does not preserve the clustering information.

The next step involves sequential pairwise swapping in the matrix until the KL metric between the matrix of the low dimensional projection and the high dimensional matrix is a minimum. This results in clustering in the low dimensional representation to reflect that of the high dimensional case.

At this point it should be noted that there is a connection between the dimension reduction that PCA and t-SNE attempt to do and the concept of "embedding". The process of embedding consists of translating high-dimensional data to low-dimensional data into a vector space such that the two are semantically related. Generally, “embedding” refers to the extraction of a portion from something of interest. Embeddings have been used to improve the efficiency of ML models and are mentioned here because of their relevance to the SigCWGAN model discussed below.

2.7.4 Quantitative Evaluation

Since the publication of the Yoon et al. (2019) paper a number of authors have converged on the evaluation methods illustrated in that paper. (Arnout et al., 2021; Debnath et al., 2021; Jabbar et al., 2021; Lakshminarayanan et al., 2021; Remlinger et al., 2022; Santoro and Grilli, 2022) are papers written in or after 2021 that adopt the Yoon evaluation methodology. In 2018, Simonetto (2018) uses an SVM as opposed to an LSTM to give a discriminative metric.

Arnout et al. (2021) state that an evaluation process should focus primarily on fidelity and usefulness. Borji (2019) defines fidelity as "the ability to distinguish generated samples from real ones i.e. discriminability". Since the synthetic data is primarily used in predictive applications, "Usefulness" is interpreted by these authors (Arnout et al., 2021; Jabbar et al., 2021; Lakshminarayanan et al., 2021; Remlinger et al., 2022; Santoro and Grilli, 2022), as predictive ability.

Discriminative score: This metric measures the error of a standard RNN or LSTM classifier in distinguishing between the real and generated sequences

Predictive score: This metric measures the predictive ability of synthetic data. Here, again standard RNN or LSTM networks are used. The Mean Absolute Error (MAE) obtained by using these networks is given as the predictive score.

2.8 Time Series Augmentation

Time series constitute an important subset of the types of data that are the object of ML systems. The analysis of time series is at the heart of a number of areas like biometrics, sound, trajectories, the recognition of signals and various other sequences. The most challenging characteristic of time series is the fact that time series store information not only in the constituent data points, but more significantly, in their order. In a significant portion of time series data, the time feature does not need to represent actual time but is used to preserve the data point order.

Distance-based methods have in the past been used to analyse time series (Box and Jenkins, 1976) but the recent surge in artificial neural networks has brought about an number of significant advances (Schmidhuber, 2015; Wang et al., 2017). Numerous studies have shown that Recurrent Neural Networks (RNN) (Rumelhart et al., 1986) have been effective in the use of time series in biosignals Kim and Pyun (2020); Xu et al. (2020), handwriting (Carbune et al., 2020; Sun et al., 2016). and gait recognition (Chen et al., 2019;

Kluwak and Niżyński, 2020). Other ML techniques like Multi-Layer Perceptrons (MLP) and temporal Convolutional Neural Networks (CNN) (Bai et al., 2018; LeCun et al., 1998) have achieved good results for time series recognition (Iwana and Uchida, 2020; Wang et al., 2017). Much of this success in this application of ML is due to the considerable growth in the amount of data that is amenable to these systems (Schmidhuber, 2015). The availability of this data also helps improve the performance and the ability to generalise of the models (Banko and Brill, 2001; Torralba et al., 2008).

A quick look at the Kaggle data repository will confirm that the number of data sources is indeed large and constantly increasing but this wealth of data masks an underlying problem; i.e. the data sets often do not have a usable number of data points. This may be confirmed by looking at one of the most often used time series datasets: the 2018 University of California Riverside (UCR) Time Series Archive (Dau et al., 2019) where only 12 datasets out of 128 have more than a thousand data points.

A solution to the problem of having an insufficient volume of data is to generate it synthetically which is the aim of data augmentation. Data augmentation is a process that is independent of the model being used. Good quality data augmentation will increase the ability of the ML models to perform successfully with unseen data by decreasing overfitting and broadening their decision boundary (Shorten and Khoshgoftaar, 2019). This performance may suffer if there is a lack of data or the data is not balanced (i.e. some classes of data are heavily overrepresented in the training data) (Blagus and Lusa, 2013), Hasibi et al. (2019).

Computer Vision (CV), particularly image recognition, has been the area where data augmentation was developed early and where it finds continued application. In Convolutional Neural Networks (CNN) (LeCun et al., 1998), a technology that continues to be the mainstay of CV, a number of successful applications have been trained using augmented data. AlexNet by Krizhevsky et al. (2009) made its mark in 2012 by winning ImageNet Large Scale Visual Recognition Challenge (ILSVRC) (Russakovsky et al., 2015). The system used cropping, mirroring, and colour augmentation of the images in the competition dataset. Other examples include:

- Visual Geometry Group (VGG) network (Simonyan and Zisserman, 2014) using scaling and cropping,
- Residual Networks (ResNet) (He et al., 2016) using scaling, cropping, and colour augmentation,
- DenseNet (Huang et al., 2017) which used translation and mirroring, and Inception networks (Szegedy et al., 2015) using cropping and mirroring.

Even though augmentation is well established in image recognition using neural networks, it has yet to gain wide acceptance in time series analysis (Wen et al., 2020). By using some techniques that were developed for CV augmentation, a number of data augmentation techniques for time series use random transformations of the original input data.

Augmentation has been applied to a number of application areas as image recognition and object localization, recognition of human action, signals, sequences, sound, trajectories, biometrics, and vital signs analysis. A good overview of the available data augmentation methods and its classification and taxonomy is given by Iwana and Uchida (2021a). The taxonomy in figure 2.9 is from that paper. As can be seen from Iwana taxonomy, there are four categories of data augmentation techniques; random transformation, pattern mixing, generative techniques and decomposition.

Fons et al. (2020) presents a survey where the methods described here are applied to Finance. Javeri et al. (2021) studies a wide variety of data augmentation techniques that are applicable to finance.

One of the simplest procedures is to add random noise that has been used to improve the performance of GANs before other more explicit augmentation techniques started being used (Fields et al., 2019). Other techniques include slicing, cropping (Le Guennec et al., 2016), scaling (Um et al., 2017), random warping in the time dimension (Fields et al., 2019; Um et al., 2017), and frequency warping (Jaitly and Hinton, 2013).

Figure 2.10, taken from (Iwana and Uchida, 2020) graphically illustrates the effects of random transformations.

The problem with random transformation-based data augmentation is that there are a diverse amount of time series each having different properties, and, thus, not every transformation is applicable to every dataset. One issue that needs to be addressed is the applicability of the particular augmentation techniques in different use cases since time series arising from different types of sources will have different mathematical properties. Whereas jittering (random noise) may be added to audio or ECG data that are inherently noisy, it gives poor results with financial time series that have specific stochastic and, more importantly, temporal properties.

An improvement on random transformations is the synthesis of time series in a manner that preserves the information content (stochastic and temporal) of the original series. This has been achieved with some success using the following techniques:

• **Random transformation:** Random transformation (Iwana and Uchida, 2020) involves perturbation, warping or slice transformations in the time, frequency or magnitude domains.

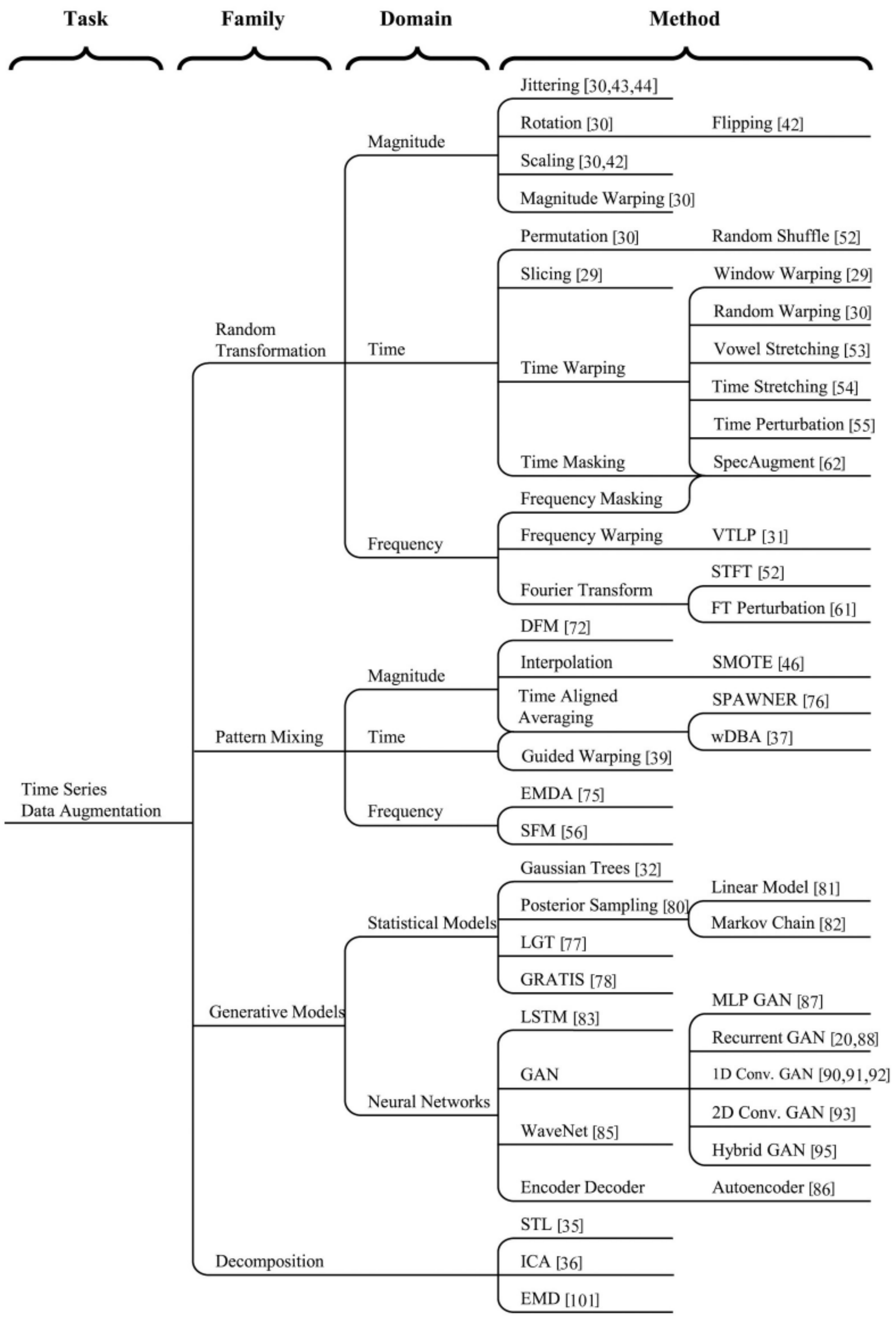


Figure 2.9: Time Series Augmentation Methods - taken from Iwana and Uchida (2021b)

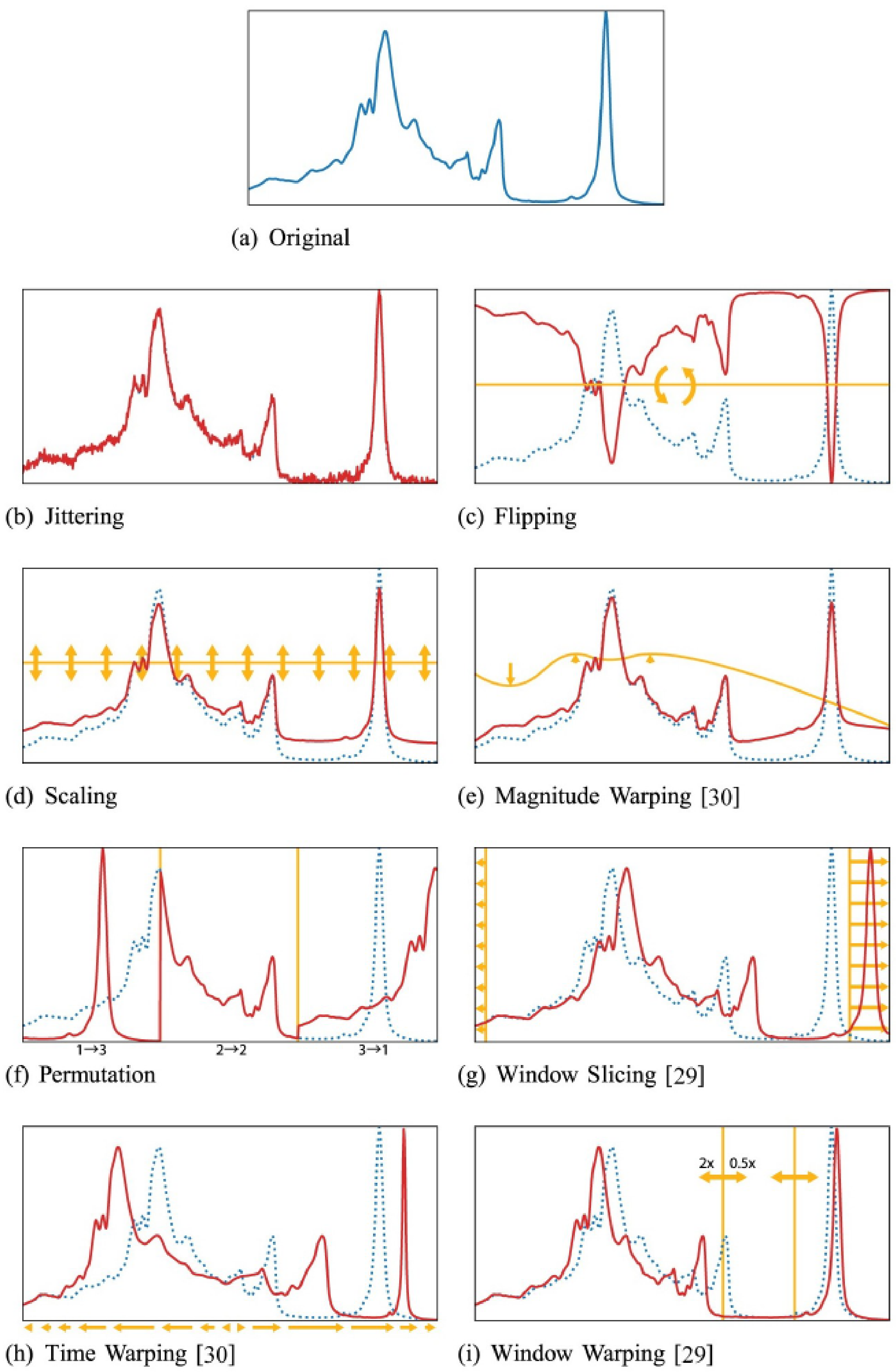


Figure 2.10: Non-GAN Augmentation - taken from Iwana and Uchida (2020)

• **Pattern mixing:** Pattern mixing (Iwana and Uchida, 2020) combines two or more real time series to give a new series that has the characteristics of the original input series. Pattern mixing may still be carried out in the time, frequency and magnitude domains but entails more sophisticated techniques that take into account the sequences' local characteristics.

• **Generative models:** Generative models attempt to reproduce the mathematical characteristic of the target series, i.e. the statistical characterisation of the given sequence. One such technique uses Gaussian trees as a statistical model that is then used as a generative template (Cao et al., 2014). Generative Adversarial Networks (GANs) (Goodfellow et al., 2020) fall under this category and are the subject of active research as illustrated here.

• **Pattern decomposition:** Pattern Decomposition involves the extraction of features and trends from the dataset (Bergmeir et al., 2016; Eltoft, 2002) to generate new patterns that have similar features. The advantage of these techniques is that the distribution of time series is preserved (Forestier et al., 2017), whereas other cruder methods may alter the data distribution. Decomposition methods usually entail the breaking down of the time series into different frequency domains like seasonal, monthly or daily granularity so that each domain's characteristics are extracted and used in the generation process.

2.9 Synthesis and Augmentation Methods

The main objective of this dissertation is the improvement of the performance of ML financial prediction models in terms of accuracy as compared to classic (non ML) methods already attempted in the literature. The manner in which we have determined to achieve this is by using data augmentation to produce more suitable data to train the selected financial prediction system.

There are two categories of data synthesis methods; GAN synthesis and a number of other methods that do not use GANs; the non-GAN methods. After an overview of non-GAN time series augmentation methods we will give a synopsis of methods specifically targeting the synthesis and augmentation of financial time series.

2.9.1 Non-GAN Augmentation Methods

Non-GAN methods fall under two sub categories; the ones that handle the magnitude of the data and the ones that deal with the time aspect.

2.9.1.1 Magnitude domain transformations.

Magnitude domain transformations operate by modifying the scalar value of each data point in a time series and do not affect the sequence of the data in the time dimension.

Jittering: Jittering is one of the simplest and effective magnitude domain transformations and consists of adding noise to the time series. It is defined as:

$$\mathbf{x}' = x + \epsilon_1, ..., x_t + \epsilon_t, ..., x_T + \epsilon_T, \tag{2.10}$$

where ϵ is typically Gaussian noise added to each time step t. One hyperparameter that needs to be determined is the standard deviation σ of the noise that is being added. The addition of noise is an accepted method that is used to increase the generalisation of neural networks (An, 1996; Bishop, 1995). It is assumed that the new time series will differ from the test or production time series by some amount of noise. It has also been shown that jittering diminishes time series drift in some neural network models (Fields et al., 2019). This drift happens when the distribution of the data shifts as new data is added.

Rotation: Rotation is defined as:

$$\mathbf{x}' = Rx_1, ..., Rx_t, ..., Rx_T, \tag{2.11}$$

where R is an element-wise random rotation matrix for angle θ. Rotation has been found to be effective in the image domain but it is recognised to be less effective in the time series since it effects the time dependence of the data elements in the sequence (Fawaz et al., 2018). This has been found to be so in neural networks when applied to time series classification (Iwana and Uchida, 2020; Ohashi et al., 2017; Rashid and Louis, 2019; Um et al., 2017) found that it could be used when supplementing other augmentation methods since an improvement in accuracy was reported.

Scaling: Scaling is done by changing the magnitude of the whole series by a given random factor as given below:

$$\mathbf{x}' = \alpha x_1, ..., \alpha x_t, ..., \alpha x_T, \tag{2.12}$$

Here the scaling factor α is taken from a normal distribution as this factor multiplies all data points with the standard deviation σ as a hyperparameter (Um et al., 2017). Alternatively σ is taken from a predefined set (Rashid and Louis, 2019). Scaling time series is different to scaling images since a this process only increases the magnitude of each data point and not the number of points in a time series.

Magnitude warping: Magnitude warping (Um et al., 2017) is a technique that applies specifically to time series. Here a smoothed curve deforms (warps) the given time series as in:

$$\mathbf{x}' = \alpha_1 x_1, ..., \alpha_t x_t, ..., \alpha_T x_T, \tag{2.13}$$

where the sequence α_1, ..., α_t, ..., α_T is generated by by interpolating a cubic spline S(u) with knots u = u_1, ..., u_i, ..., u_I where the ui values are taken from a normal distribution as is the number of knots. The standard deviation of this normal distribution is a modifiable hyperparameter. The rational behind this method is to increase the complexity of scaling augmentation by modifying the scaling factor according to a smooth function. As in other transformations dependent on normally generated randomness, one drawback this method has is that it may distort any time dependence the original time sequence may have. Another issue is that this method depends on two hyperparameters (Knot number and standard deviation) rather than one, making it more demanding to use than other transformation based methods.

2.9.1.2 Time domain transformations.

The techniques described above with regard to magnitude may be modified to transform the time domain. This entails displacing data points in the time series by different amounts in the time domain.

Window Slicing: Slicing calls for the removal of chunks of the given time series from its beginning and end. This is conceptually similar to cropping in the image domain. The process is given by:

$$\mathbf{x}' = x_{\varphi}, ..., x_t, ..., x_{W+\varphi}, \tag{2.14}$$

where W is the size of a window and φ is a random integer such that $1 \leq \varphi \leq$ T − W. Slicing is also known as Window Slicing (WS) (Le Guennec et al., 2016) because it crops to a window size W

Permutation: Um et al. (2017) first suggested permutation where parts of the time sequence are shuffled so that a new sequence is produced. This method implicitly negatively affects any time dependence that exists between the data points in the time sequence.

There are two versions of the tranformation where the segments are either equal or variable in size (Pan et al., 2020). The equal segment version divides the sequence in N segments of the same size and then proceeds to shuffle them. Similarly, the variable segment version partitions the series in N variable length segments that are also shuffled.

Random shuffling: Random shuffling is the same process but instead of segments, the individual data points are permuted. Steven Eyobu and Han (2018) used this transformation in addition to local averaging both before and after shuffling.

Time warping: Time warping is similar to magnitude warping but affecting the time domain. It also uses a smooth warping path (Um et al., 2017). Here the time series becomes:

$$\mathbf{x}' = x_{\tau(1)}, ..., x_{\tau(t)}, ..., x_{\tau(T)}, \tag{2.15}$$

where $\tau()$ is a warping function that warps the time steps based on a smooth curve and this curve is a cubic spline S(u) with knots u = u_1, ..., u_i, ..., u_I. The knot heights are taken from a normal distribution. This transformation will change the input time series by expanding and contacting different parts of the series.

Window warping: A second method of time warping involves window warping as suggested by Le Guennec et al. (2016). Here a widow of the time series is either expanded to twice the size or contracted to half its original length. These authors suggest that the factor of 2 they used may be a variable hyperparameter.

Random Guided Warping and Discriminative Guided Warping: Time domain mixing Guided warping (Iwana and Uchida, 2020) involves using Dynamic Time Warping (DTW) (Sakoe and Chiba, 1978). DTW is an algorithm that originated in dynamic programming where it incorporates elastic element matching. The technique is used to compare to time sequences by expanding or contracting segments of the second time sequence until it maximally matches the first. Thus the target sequence is warped so that it gets to resemble a ’teacher’ sequence using DTW.

Random Guided Warping: Random Guided Warping (RGW) uses a random intra-class teacher sequence. Discriminative Guided Warping (RGW) on the other hand uses a directed discriminative teacher (Iwana and Uchida, 2020).

SPAWNER: Suboptimal element alignment averaging. Kamycki et al. (2019) developed suboptimal warped time series generator (SPAWNER) . This algorithm generates sequences using suboptimal time warping. SPAWNER is an adaptation of DTW that constrains the warping to operate at various random points. SPAWNER may be seen as a process that averages any number of aligned time sequences.

2.9.1.3 Selection

We have chosen to use Window Slice and Window Warping by Le Guennec et al. (2016) as the non-GAN methods to be used in experiment 1b (section 3.3) since these were the two methods that Iwana and Uchida (2021a) reported to be the best performers. A quick perusal of the plots in appendix B showing a preliminary overview of non-GAN methods also confirms that Window Slice and Window Warp behave in a satisfactory manner.

2.9.2 Synthesis and Augmentation of Financial Time Series

We now focus on the efforts made in data augmentation specifically targeting financial data. Financial data has a number of application areas like high frequency trading, laundering detection and its use in Monte Carlo methodologies (Athey et al., 2021; Jullum et al., 2020). Operators in this field have reported a number of challenges regarding the quality of available financial data like missing or inconsistent data and restricted access due to privacy or ownership concerns (Liu, 2020). Even though data may be available it may also be noisy or out of date. There is also the issue that some established augmentation methods, like oversampling, will perform less than satisfactorily since important characteristics like temporal information will be degraded.

It is pertinent here to mention a number of non-ML methods of artificial data generation that are mentioned in most introductions to the subject of financial data generation; ARIMA by Box and Jenkins (1976) and GARCH by Bollerslev (1986) that have been described in section 2.6.1 as statistical prediction techniques. These have, nevertheless, been used as generative models in their own right using the maximum likelihood technique.

A number of authors have suggested ways to address the challenges mentioned above. ML models have been proposed by Liu et al. (2021) together with other techniques. Duan et al. (2022) reports the use of new features like macroeconomic measures and various market signals to train ML models. Transfer learning has been suggested by Jørgensen and Igel (2021) where models trained on the data from particular companies is used to help analyse companies were data is scarce. Lahmiri (2020) and Braun et al. (2019) have investigated wavelet transformation as a data augmentation method.

Reinforcement learning has been used by Tizziano (2018) on data that was generated by a recurrent GAN. Reinforcement learning has been mentioned briefly in section 2.3. It works by having the system maximise a reward that in this case was the profit made from trading assets. Tizziano (2018) states that the trading system performed better with augmented data since it was able generalise over a wider spread of data.

There has been a surge, in the last few years, of studies that relate to GANs used in financial data augmentation. de Meer Pardo (2019) uses the Wasserstein GAN (WGAN) to generate data that simulates the S&P500. The evaluations used to validate the generated data were autocorrelation and the kurtosis figure in addition to loss and accuracy plots. The author then trains a ResNet model using real and synthetic data. The paper states that a 70% accuracy was achieved using this methodology.

Another GAN that has been mentioned numerous time in the literature is TimeGAN by Yoon et al. (2019). Liu et al. (2022) state that " TimeGAN is the most suitable GAN-based generative model for augmenting our training dataset with realistic synthetic data". Our preliminary test using TimeGAN found that because of the difficulty in generating time series that are longer than the ones generated in Yoon's paper (usually 24 data points) it was considered unsuitable for this dissertation as the sequences used spanned years (i.e. at least 250 data points) that made TimeGAN computationally intractable.

Other GANs used for the generation of financial time sequences are:

- Conditional GAN (RCGAN) (Esteban et al., 2017)]
- CGAN [fu2019time] C-RNN-GAN [Mogren (2016)]
- WaveGAN [Donahue et al. (2018)]
- Conditional Generative Adversarial Networks (cGANs) [Jordon et al. (2018)]
- QuantGAN [wiese2020quant] and StockGAN [Li et al. (2020)]

The papers of Efimov et al. (2020), Zhang and Khoreva (2019), Takahashi et al. (2019), Snow (2020) and Wiese et al. (2020) describe a series of GAN architectures specifically targeting GAN data augmentation in finance.

An augmentation model that is inspired by the GAN architecture is the Generative Teaching Network (GTN) developed by Such et al. (2020). GTNs also have two primary components; the generator and the learner. The generator is not constrained by the fact that it has to produce data that is similar to real data but is allowed to produce a wider range of data. The burden of constraint is shifted to the learner unit were the validity of the generated data is determined by how accurately the learner is able to perform the required regression or classification job. Thus, when the generator produces data that is not very realistic but still allows the learner to perform better, the data is considered satisfactory.

2.10 GANs in this study

An initial overview of the literature covering GANs and in particular GANs used in finance has lead us to select two GANs to be investigated here; BigGAN and SigCWGAN.

Brophy et al. (2021); de Meer Pardo et al. (2022); Eckerli (2021); Labiad et al. (2021) all report that SigCWGAN gave the best results from the investigation they carried out.

Before dealing with the GAN architectures themselves we will first overview two fundamental design solutions; the wasserstein distance and the signature transform.

2.10.1 Wasserstein distance (WD)

The Wasserstein distance may be seen to be the minimum cost of transporting mass as data distribution q is converted to the to the data distribution p. The Wasserstein distance for the real data distribution Pr and the generated data distribution Pg is mathematically defined as the greatest lower bound (infimum) for any transport plan that is the cost for the cheapest plan. The term "Distance" is somewhat of a misnomer here as the WD is actually the minimum sum of a product (just like work is force x distance) of the weight of what is being moved multiplied by the distance moved. The "Plans" mentioned in the definition given above are the possible paths through which the "weights" (i.e. chunks of the distributions) will be moved. The WD algorithm will calculate all the "distances" of all plans and chose the one with smallest cost. One can now see why the WD metric is also known as the "Earth Mover" metric.

The use of various loss functions have been attempted in different GAN architectures over time. Since the purpose of a loss function is to measure the difference between the network's prediction and the actual data, various algorithms have been used, two of the most popular being the Kullback–Leibler (KL) divergence and the Jensen–Shannon (JS) divergence. In the figure 2.11 are their characteristc plots.

The figure 2.11 illustrates one the most consequential issues with the choice of loss function in GANs and that is the "vanishing gradient" problem. The training phase of any neural network involves the "gradient descent" mechanism were the network is supposed to look for an optimal configuration of the weights of its neurons by descending a slope in the error lanscape. This will obviously prove to be impossible if the gradient is very close

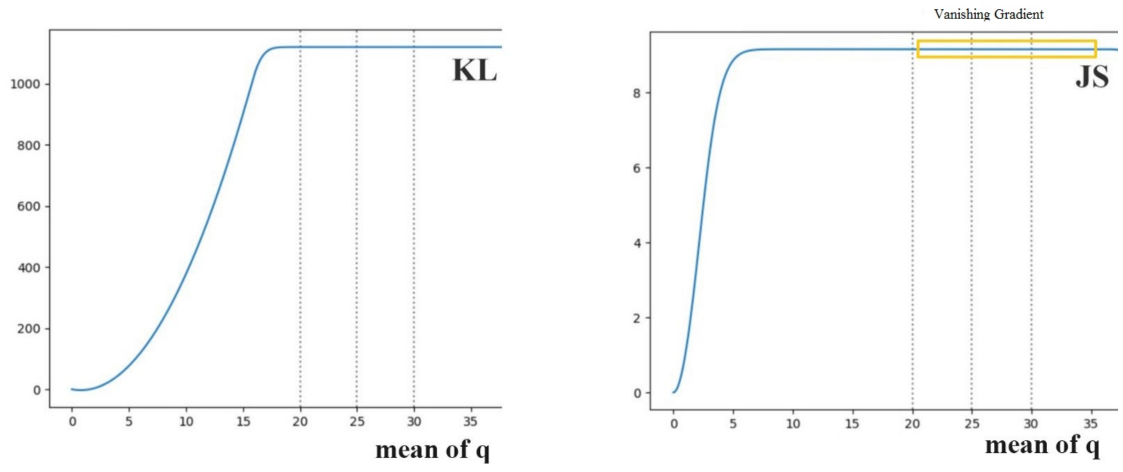


Figure 2.11: Vanishing Gradient - from Violante (2018)

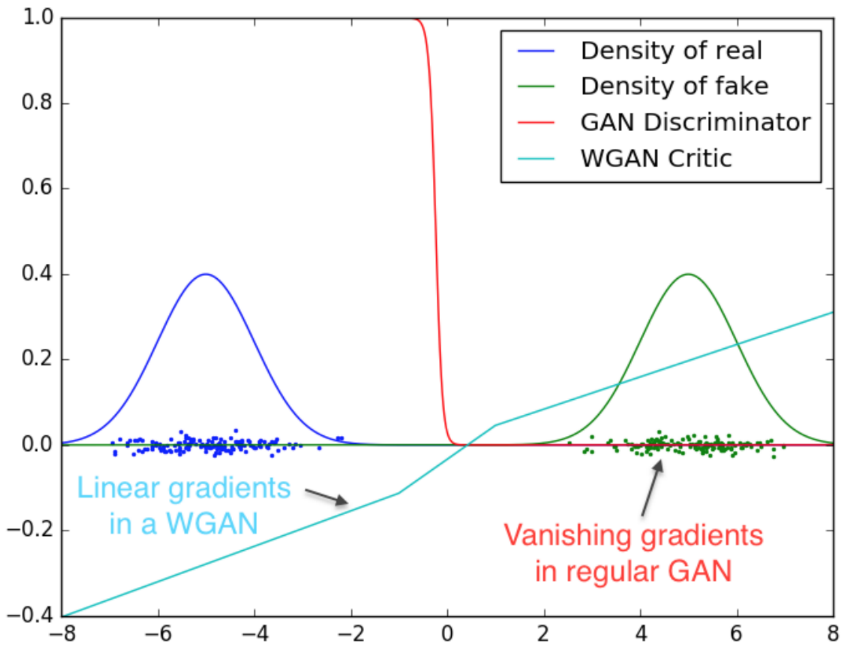


Figure 2.12: WGAN Gradient - from Arjovsky et al. (2017)

to zero. This is one of the main reasons why the generators in previous GAN configurations failed.

Using the Wasserstein distance as a loss function gives better results as seen in the figure 2.12.

2.10.2 Signature Transform

One way to understand the function of the signature transform is to compare it to the Fourier transform where the target of these transforms are sequences of data like financial time series. Whereas the Fourier transform captures frequency information, the signature transform condenses order and area information. In another difference to the Fourier transform, order and area signify the range of nonlinear effects in the data. This means the signature transform is referred to as a "universal nonlinearity". This means that all continuous functions of the input stream will be approximated (up to an arbitrary limit) by a linear function of its signature.

Chevyrev and Kormilitzin (2016) provide a comprehensive overview of the mathematics of the signature transform. One path to grasp the essential elements of the transform is to work through a series of concepts that should be understood before exploring the signature transform:

- The Taylor series as an expansion of a function into a polynomial.
- The Fourier expansion that breaks a function down into component frequencies.
- The Picard algorithm that analyses ordinary differential equations into a series of iterated integrals.

It must be stated that what is being referred to as a function above is actually a path in a topological space. The non linear features of financial time series may be handled as the analysis allows for linear piecewise paths.

The signature transform has a number of properties that make it very useful in ML, namely:

- It is well behaved when data is missing or sampled irregularly.
- It may be translation invariant.
- It may be sampling invariant.

Thus the signature transform is used to provide an embedding that accurately represents the statistical characteristics and time dependencies of the target time sequence. This embedding is then used to ascertain the proximity of the distribution of the input time sequence with the generated time sequences using the Wasserstein distance covered in section 2.10.1 above.

For these reasons it is used in the SigCWGAN as a replacement to the discriminator stage found in the standard GAN configuration.

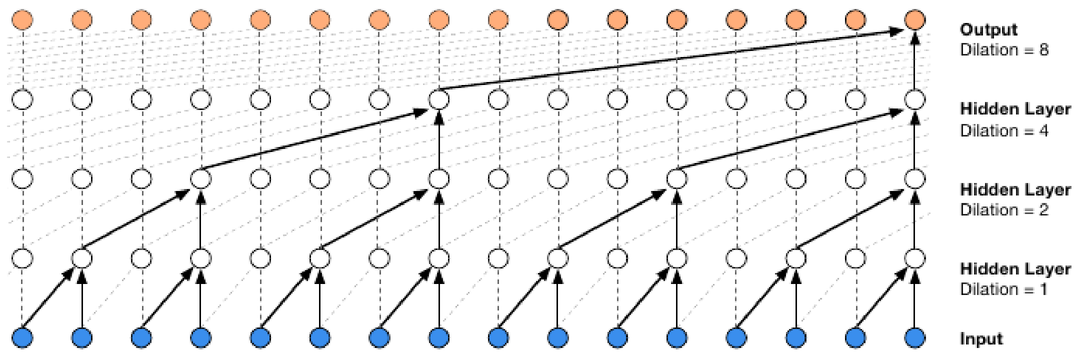


Figure 2.13: Wavenet - from Oord et al. (2016)

2.10.3 SigCWGAN

The SigCWGAN by Ni et al. (2020) also nominally follows the standard GAN architecture having a discriminator and a generator as in figure 2.3. The generator is a autoregressive feedforward neural network (AR-FNN) that has an architecture inspired by WaveNet (Oord et al., 2016) in figure 2.13. The main idea in an autoregressive neural network is that a number of previous data points (i.e. temporal predecessor data points) are fed as inputs to the neural network which then produces an output based on that (historical) data.

The signature transform is the primary architectural component in the operation of SigCWGAN. Its' purpose is to produce a highly condensed mathematical characterisation of an input data sequence. Using this signature on the real and generated time sequences, the discriminator can produce a loss statistic using the Wasserstein distance. The basic idea behind the Wasserstein distance or Earth-Mover distance is for the algorithm to measure how much "content" (i.e. earth) must be moved from the target distribution to the source distribution such that they are the same. Kulback-Liebler divergence is based on information theory. Any communication (i.e. information) may be encoded using a minimal encoding scheme, that is, an encoding scheme that will reduce to a minimum the number of bits necessary for that information to be transmitted. KL divergence measures how much more information (bits) must be used to transmit a target message using encoding optimised for a source message. Entropy is related to KL Divergence in that KL divergence measures the average number of bits required whereas entropy measures the total number of bits. Thus, SigCWGAN operates like a standard GAN in minimising this loss function.

Ni et al. (2020), who is the author of the original SigCWGAN paper compares SigCWGAN with TimeGAN (Yoon et al., 2019), RCGAN (Hyland et al., 2017) and GMMN (Li et al., 2015). His evaluations show that SigWGAN globally performs better than the other generators. This result is particularly significant as it shows that SigCWGAN performs better than TimeGAN that is considered by many authors as an outstanding benchmark. Labiad et al. (2021) state "The results of experiences conducted on the S&P 500 index (SPX) and Dow Jones index (DJI) data show that SigCWGAN achieves superior or comparable performance to the other baselines." This is significant because the experiments carried out show SigCWGAN's suitability in generating synthetic financial data. Li et al. (2022) compares SigCWGAN with RCGAN (Hyland et al., 2017) and TimeGAN (Yoon et al., 2019) and also shows that SigCWGAN performs as well or better than the other benchmarks.

2.10.4 BigGAN

A GAN architecture that has been very successful in addressing the difficulties mentioned in section 2.5.4 is BigGAN by Brock et al. (2018). BigGAN has performed very well in the evaluations carried out here and therefore the techniques it implements are overviewed here.

Ding et al. (2022); Jabbar et al. (2021); Makhmudkhujaev and Park (2021); Tseng et al. (2021); Wang et al. (2021); Wiatrak et al. (2019); Zhang et al. (2019b) all have found that BigGAN performed as well or better than any other GAN in the Computer Vision applications they investigated. Ding et al. (2022); Tseng et al. (2021); Wang et al. (2021); Wiatrak et al. (2019); Zhang et al. (2019b) have cited BigGAN as being "State of the art". Google scholar states that the original BigGAN paper by Brock et al. (2018) was refered to by over 3500 other publications, far more than any architecture we have surveyed. It is for these reasons that we chose BigGAN as one of the architectures to be investigated in this study.

The extent of the design characteristics listed here may be appreciated by perusing the BigGAN discriminator and generator summaries reproduced in appendix D

Self-Attention Module and Hinge Loss: The Self-Attention Module and Hinge Loss techniques are adopted from the paper byZhang et al. (2019a). As can be seen from the diagram below, this involves the introduction of a self attention mechanism to a DCGAN. A problem with simpler GAN architectures is that the target of the convolutional mechanism may be too mall and miss the importance of larger scale features which may result in well generated noses being placed in the wrong position on the face. The self-attention mechanism provides feedback at larger scales. This solution is applied both to the discriminator and to the generator.

Class Conditional Information: This technique is derived from two papers:

1. A Learned Representation For Artistic Style by Dumoulin et al. (2016)
2. Spectral Normalization for Generative Adversarial Networks by Miyato et al. (2018) In essence the benefits of Conditional GAN's (CGAN) are employed using an embedding mechanism that increases fidelity and reduces training speed.

Spectral Normalization: The Spectral Normalisation technique was also first escribed in the Miyato et al. (2018) cited above. The highly effective batch normalisation procedure is modified to operate on stochastic gradient descent.

Update Discriminator More Than Generator: This is a technique that has been widely adopted by GAN developers and has been reported to be effective in several papers and literature. In BigGAN the discriminator is updated twice for every time the generator is updated.

Moving Average of Model Weights: This development is by Karras et al. (2017) Here, the authors have suggested that the weights of the generator are moderated by the value of a moving average across prior iterations.

Orthogonal Weight Initialization: In the paper Saxe et al. (2013) it is suggested that model weights are initialised to a random orthogonal matrix that satisfies:

$$W^T \cdot W = I \tag{2.16}$$

Larger Batch Size: The BigGan developers tested the use of very large batch sizes all the way up to 2048 images. The experiments conducted indicated that there were improvements by doing this and the best results (46% improvement) were obtained by using a batch size of 2048. The authors posit that this happens for a reason similar to why Self-attention works i.e. the network's convolutions have a broader view of the distributions of the processed images.

More Model Parameters: The quest for increased size also targeted the quantity of model parameters, whereby the number of feature maps in each layer was doubled. This lead to an IS score improvement of 21%.

Skip-z Connections: An approach that was then developed comprehensively in Style-GAN is the adoption of skip-z connections. This involves the connection of the latent space not only to the input of the generator but also to deeper layers. The authors state that this led to a training speed improvement of 18%

Orthogonal Regularization: Here this technique was introduced by Brock et al. (2016). This approach takes the idea used in "Orthogonal weight initialisation" (described previously) and applies it to regularisation by adding a term in the cost function that promotes this behaviour.

Truncation Trick: In the training and inference phases of the GAN process, the generator is fed two different distributions; a normal Guassian in the training phase and a truncated Guassian in the inference (generation) phase. This allows the generated images to be either more varied with a wider gaussian or more accurate (higher fidelity) with a narrower sampling range.

2.11 Conclusion

The outcome that has been sought in this study is the improvement of financial prediction by increasing the amount of data that is used to train the prediction system. Financial prediction is by no means a given and this is evident from the amount of discordant literature dealing with this subject. A consistent number of very authoritative opinions that uphold the Efficient Market Hypothesis exclude the possibility of financial prediction. The position that has been subscribed to here is that the EMH should be seen as a steady state view and that this, then, would allow the possibility of prediction during the transient periods between market events.

Machine Learning financial prediction systems depend on two significant antecedents; human financial analysis and Machine Learning technologies. Again, with regard to financial analysis, an explanation must be sought to justify why ML financial prediction has followed on the techniques pursued in Technical Analysis rather than, the more academically reputable Financial Analysis. Apart from the arguments debated in the literature, the main argument in favour of TA is a pragmatic one; there is substantial evidence that ML prediction (adopting TA techniques) works.

An overview of ML prediction techniques has been given here (section 2.6). The objective in this work was to establish whether data augmentation would enhance the performance of a financial prediction system. The choice of an optimal financial prediction system was beyond the scope of this dissertation. The review of the literature carried out here showed a predominance of Neural Networks being used in financial prediction systems (section 2.3) and were, thus, used here. This reasoning extended to the choice of the benchmark paper that uses both a statistical (ARIMA) prediction technique as well as an ML technique (RNN).

There were a number of issues that had to be dealt with regarding financial data synthesis and augmentation. Even though there was a predominance of the use of Generative Adversarial Networks (GANs) in papers (section 2.8) that dealt with data augmentation in time sequences (including financial time sequences) it was decided to include other augmentation methods in this study. Following on the assessment by Iwana and Uchida (2021a), the two methods developed by Le Guennec et al. (2016), namely Window slice and Window Warping were chosen as the alternative (non-GAN) augmentation methods to be evaluated here.

An issue that was prevalent in the literature dealing with all forms of data synthesis and augmentation was the evaluation of the suitability of the generated sequences (section 2.7). There were clearly two methodologies that emerged from the literature; the Visual/Qualitative and the Quantitative evaluations. Although the visual/qualitative evaluations were dependent on observation and were thus somewhat subjective, they were still used in our appraisal to confirm the quantitative assessments. On the quantitative front, a consensus emerged from the perusal of the papers reviewed in favour of discriminative/predictive metrics as described by Yoon et al. (2019).

The problems (jabbar2021survey) faced early in the development of GANs lead to a wide range of solutions being explored as can be construed from a number of survey papers (Brophy et al., 2021; Iwana and Uchida, 2021a; Jabbar et al., 2021; Shorten and Khoshgoftaar, 2019; Wang et al., 2021; Wen et al., 2020; Wiatrak et al., 2019). BigGAN was selected since it was deemed by many authors to represent the state of the art and a benchmark against which other GANs have been compared.

SigCWGAN by Ni et al. (2020) is a recent development and was chosen for the elegance of the solution adopted and the fact that it was reported to perform well. The literature we have perused has shown that the application of SigCWGAN to generate augmented data for the purpose of training a financial prediction system has not yet been attempted.

3 Methodology

3.1 Experiments Overview

As described in the introduction (section 1) the motivation for this study is to explore data augmentation methods in view of enhancing ML financial prediction models by making it possible to provide more training data leading to an improvement in out of sample performance. A number of candidate data augmentation methods were tested against a selection of evaluation methods by Yoon et al. (2019), Eckerli (2021) and Takahashi et al. (2019). Thus, the work here involved finding the data synthesis method that gave the best results according to the Yoon et al. (2019) quantitative evaluations. This permitted a selection of data augmentation methods. Visual/qualitative evaluation methods were used to corroborate the quantitative results. Augmented data generated by the best performing synthesis method was then used to train the prediction algorithm described in the YU and Li (2018) benchmark paper. Following the methodology in Teng et al. (2020) the prediction algorithm was first trained with real financial time series to provide a baseline for comparison. Subsequently the prediction algorithm was trained with synthesised data using the generative method chosen after the testing mentioned above. The error metrics as per the benchmark paper (YU and Li, 2018) were used to asses the performance of the augmentation system being investigated in this study.

3.2 Experiment 1a - GAN Evaluation

3.2.1 Motivation

The first experiment is intended to fulfil objective 1a as stated in section 1.2. The aim here is to evaluate the performance of the GAN architectures selected in section 2.10 so that it may produce the financial time sequences required in experiment 2.

Experiment No	Objective	Experiment
1a	A survey of current literature dealing with Financial Time Series Augmentation indicates that Generative Adversarial Networks (GANs) are the preeminent technology in this field. The first objective in this study is to establish which, out of a selection of GAN architectures, performs best when these are compared using a suitable evaluation metric. Thus, the objective here is to produce a coherent tabulation ranking of the GANs according to a specific metric.	section 3.2
1b	Even though there is an apparent consensus that GANs are the technology of choice with regard to time series augmentation, it would be, nonetheless, prudent to investigate alternative technologies. Thus, objective 1b is to survey these non-GAN technologies and compare their performance using the same evaluation metric that was used in the fulfilment of objective 1a.	section 3.3
2	The second objective to be dealt with is to establish whether the augmentation method selected after performing the experiments to fulfil the objectives 1a and 1b result in the improved performance of a financial prediction system as measured by standard error metrics. This will entail the comparison of the error metrics obtained using real financial data against the results obtained when using augmented data in the training phase of the selected prediction system.	section 3.4

Table 3.1: Experiments corresponding to Objectives

3.2.2 Actions undertaken

1. We use the visual evaluation tools developed by Eckerli (2021) and Takahashi et al. (2019) to perform a preliminary selection of GAN candidate architectures.
2. We use the evaluation methods developed by Yoon et al. (2019) to give a performance metric of the GANs chosen from the previous task.
3. We verify that the evaluation obtained in task 2 is reflected in the quality of the visualisations mentioned in task 1

3.2.3 Data

3.2.3.1 Data Sources

The replication of the benchmark paper selected (YU and Li, 2018) required the procurement of the OHLC (open, high, low, close) for the S&P500, SSEC, N225, HSI and IXIC indices. The S&P500 index, in particular was used as the ground truth data for for experiments 1a and 1b. As suggested by Ni et al. (2020) (and also used in his code) the Oxford-Man Institute's "realised library" was used as a source for the indices' OHLC data. The downloaded data was loaded into comma delimited CSV files named for each specific asset. These files included a date column.

3.2.3.2 Preparation

Below are a number of routines that are part of the standard processing that is carried out on input data before it is fed to algorithms as carried out by YU and Li (2018) and Eckerli (2021) in their code.

MinMaxScaling and StandardScaling: The purpose of scaling is to transform data that has differences in scale and units to quantities that are comparable (Lezmi et al., 2020). For example one set of data may be measured in kilometres whereas another is expressed in light years. The scaling algorithms will ignore these differences and reduce the data to some desired range, usually from zero to one. In the case of financial data this phenomenon is evident in the wide range of current asset prices that range from fractions of a dollar to at around $414,000 for Berkshire Hathaway class A shares, although the price variations in the indices used in this study are less extreme. Normalisation reduces the range of the input data to between zero and one using the formula:

$$y = (x - min)/(max - min) \tag{3.1}$$

This is performed in the code by the scikit-learn MinMaxScaler function. Data standardisation will rescale the distribution of the input data such that its mean is zero and its standard deviation is one. This assumes that the data is normally distributed. The transformation is

$$y = (x - mean)/standard\_deviation \tag{3.2}$$

This is performed in the code by the scikit-learn StandardScaler function.

Date ranges, Price and log returns: Since the stochastic behaviour of most assets will change over time, ensuring that the date range of the required data is consistent is important.

Following Eckerli (2021) and Ni et al. (2020) the principle format that the GANs used as input and also their output was in terms of price. The process of generating log returns by taking the difference in price on successive days transforms the price time series into a series that is stationery. This facilitates the operation of statistical functions used in various AI models. Log returns are scale free thus allowing a better comparison of the performance of various assets and the synthesised data that simulates these assets.

A Log_Returns column was added to the generated files. This is because the generated files had no intrinsic date or price information and only represented the relative movement of the asset from one data point to the next. Thus, for there to be a comparison with the original data, these movements had to be anchored to a starting price corresponding to the asset price at a given date. To enable this process log returns were used. Code was implemented to convert sequences of log returns to asset prices and vice versa.

Generating the training and test data sets: The data that was used as input to the various algorithms was selected by the appropriate date range and then split into train and test data in a 3:1 ratio as per the procedure in YU and Li (2018).

Sliding window slices and shuffling: Following Eckerli (2021), the GANs that have been investigated have been trained on a number of short slices (24 data points) generated from the price time series of the asset in question. Thus if the asset CSV contains 1000 sequential data points from n=0 to n=999, the first slice will contain data points n=0 to n=23, the second n=1 to n=24 and the kth slice n=k-1 to n=k+22. These slices are added to a python list and this list is then shuffled so that the data set is independent and identically distributed (I.I.D). This is the format that is fed into the GANs at the training stage.

Stationarity: The GAN code following Eckerli (2021) and Ni et al. (2020) transformed daily closing prices into log returns. This is done to ensure the stationarity of the GAN training data.

Stationarity may be understood as the condition when the statistical properties of a random process that produces a time series are invariant over time. This does not imply that the time series does not change over time, only that it preserves its statistical attributes. An analogy in simple algebraic terms is a linear function rather than a flat horizontal line. In this analogy the mathematical feature that is preserved is slope and not simply a constant y value.

Stationarity is a characteristic that is often required at the outset of most data analysis processes since a large majority of the tools used for analysis require that the data is stationary. Simply considering the most abundant type of data that is dealt with in stock market trading; asset price time series, will lead to the realisation that not all stochastic data is stationary since, like in the case of asset prices, these usually have a variable mean; i.e. they ramp upwards or downwards. The process of differencing the time series one or more times is often what is needed to render the data stream stationary. The first difference is another quantity that underlies several analytic processes, namely the periodic (e.g. daily) return of an asset. An implication resulting from this is that since a stationary series has some statistical characteristic that is constant, then predictions may be inferred from the data; the extrapolation of the stationary characteristic may be used to make simple predictions. More complex models, also use and are dependent on data being stationary. Trend estimation, forecasting, causal inference and the GANs investigated here use the stationarity of data.

3.2.4 Experiment Design

As stated in above an initial appraisal of the BigGAN hyperparameters was carried out resulting in the retention of the values established in the paper (Brock et al., 2018). Eckerli (2021) states he has adopted a "black box" approach in evaluating the performance of the GAN models. This means that the various GANs were implemented as per the relative papers. The motivation behind this approach was to assess how the GANs perform "out of the box" since Eckerli (2021) expects that these will be used as one of a number of building blocks underlying a more complex applications. Another reason for keeping changes to the original code to a minimum is that a comparison can be done on their "baseline" configurations, limiting the possibility of changes drastically improving the performance of some of the architectures and not others. Eckerli (2021) adapted the code from that in the original paper where the latter was designed to deal with images rather than time

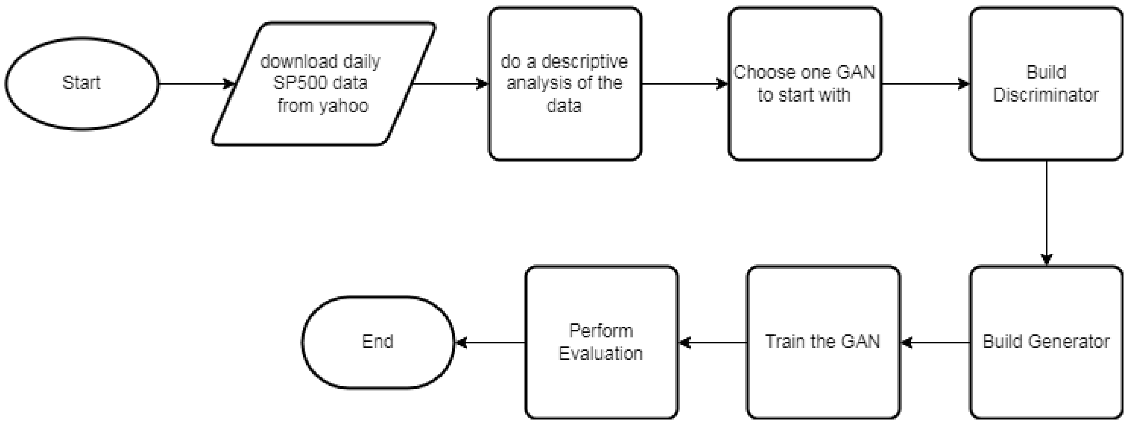

Start
download daily SP500 data from yahoo
do a descriptive analysis of the data
Choose one GAN to start with
Build Discriminator
Build Generator
Train the GAN
Perform Evaluation
End

Figure 3.1: Experiment 1 Workflow

series. This required the modification of the dimensions of the input layers. The workflow adopted in this experiment is shown in the flowchart in diagram 3.1.

3.2.5 GANs

In this study, following the methodology by Eckerli (2021), 12 GANs where given an initial appraisal, namely: DCGAN, LSGAN, SAGAN, WGAN, WGAN-gp, DRAGAN, RAGAN, RALSGAN, YLGAN, BigGAN, BigGAN_deep, TransGAN and SigCWGAN using the Visual/Qualitative evaluation methods described in section 2.7.2.

BigGAN and SigCWGAN were selected from the wider list above after a visual evaluation following the methods described by Eckerli (2021) and Takahashi et al. (2019) (Task 1). Another criteria for this choice of GAN architectures was the fact that they used intrinsically different loss functions. As stated in the Literature Review (section 2.10.4) BigGAN was also selected since it is very often used as a benchmark in papers dealing with GAN performance.

SigCWGAN was selected for three reasons. First, the authors that have investigated it (Li et al., 2022; Ni et al., 2020) have found it to perform better than TimeGAN, a favourite with authors working in financial time series augmentation. Secondly, it appears to solve a problem that has afflicted most other GAN architectures that came before; i.e. the quick stabilisation of the generator loss function (the vanishing gradient problem). Thirdly, it substitutes a discriminator that requires training and behaves stochastically with a deterministic function (the Signature Transform) and this reduces the computational burden and the variablility of the result.

The implementation of BigGAN is adapted from Eckerli (2021). SigCWGAN was implemented on code based on that provided by Ni et al. (2021).

3.2.5.1 BigGAN:

The BigGAN architecture is the synthesis of a substantial number of solutions to issues that had been identified in the development of early GAN architectures. These solutions were addressed by different authors in their corresponding papers. This makes BigGAN one of the preeminent architectures available. The layers used in the BigGAN generator and discriminator are given in tables D.1 and D.2.

3.2.5.2 SigCWGAN

The background regarding the signature function that underlies the function of the SigCW-GAN architecture has been given in section 2.10.2 of the literature review.

SigCWGan still has the basic architecture that characterises GANs, i.e it has a generator and a disccriminator. The Generator architecture chosen is autoregressive feedforward neural network (AR-FNN).

The signature function is used to characterise both the original time sequence and the one that is produced by the AR-FNN generator. The loss function is then simply the difference of the two signatures.

3.2.6 Evaluation Criteria

GAN's have originally been used in Computer Vision (CV). The evaluation of GANs has always posed a challenge to the operators in the various AI fields that use this technology. An overview of the literature has indicated that the methods described below have attracted some consensus in the field (Borovykh et al., 2017; Eckerli, 2021; Efimov et al., 2020; Franco-Pedroso et al., 2019; Leznik et al., 2021; Smith and Smith, 2020; Sun et al., 2020; Wiese et al., 2020).

The function that is the most essential in achieving objectives 1a and 1b (section 1.2) is the evaluation of the fidelity of the generated time series with respect to the original time series, both with regard to statistical properties and crucially to time dependence.

Two approaches have been adopted in order to achieve this; a series of visual assessments (Eckerli, 2021), Takahashi et al. (2019), and the predictive and discriminative metrics as devised by Yoon et al. (2019). As described in the section 2.7.2 these metrics are in addition to PCA and t-SNE plots that Yoon et al. (2019) also uses.

PCA was used as one of the two visual evaluation methods. To do this 2 dimensional PCA using the sklearn python library was carried out. This PCA process was applied to the time sequence to be tested, namely the generated and augmented asset time sequences and compared to the real asset time sequence. If the generated sequences conserved the features of the real data then the projections should overlap significantly. The t-SNE implementation used here was also the one in the sklearn library.

Visual assessments were carried out as described by Eckerli (2021) using code adapted from that supplied by him. Eckerli (2021) limits his evaluation to the comparison of "stylised facts". The process was enhanced by also including the visual evaluation tools described in Takahashi et al. (2019). The time series produced as described above (section 3.2.5) were passed to a python program that produced the various graphs to enable the assessment of the stylised facts described in section 2.7.2.

3.2.7 Discriminative and Predictive Metrics

Remlinger et al. (2022), Arnout et al. (2021), Santoro and Grilli (2022) and Lakshminarayanan et al. (2021) have all adopted the discriminative and predictive metrics developed by Yoon et al. (2019). Grilli and Santoro (2020) also adopt these metrics independently. Both these metrics are neural networks built using the python Tensorflow library following the implementation by Yoon et al. (2019).

The predictive model adopted by Yoon is essentially a regression algorithm. The algorithm must simulate a hidden function that gives an output Y when given an input X just like mathematical functions behave; for all points in a domain, corresponding points within a range are output. As opposed to mathematical functions that are explicit, a function in an ML regression system must be inferred by the system from the training data. Predictive systems in financial time series are a subset of regressors in that the input is a short time sequence and the output are points (or a single point) that should follow the input sequence. A large variety of regression algorithms have been investigated for prediction applications in the past (section 2.6.1). The solution chosen here follows that adopted by Yoon et al. (2019) that uses a gated recurrent unit (GRU) as a predictor. The mean absolute error (MAE) of the predictions (vs. the original sequence) is then output as the prediction metric.

It is worth noting that discriminative metrics have a history that goes back to the evaluation of GANs in computer vision (Goodfellow et al., 2020) because one of the two fundamental components is a discriminator. GANs were first used to generate pictures of specific objects (horses, human faces etc.). The problem of gauging the quality of this output concerned operators in the field as this automatic metric had to perform close to appraisal that was done by human judges (Salimans et al., 2016). One solution that was adopted was to measure if the generated image actually looked like the object it was supposed to mimic. This was done by employing Google's object classifier; the inception machine (Barratt and Sharma, 2018). Following this tradition (Yoon et al., 2019) developed his discriminative metric that is essentially a classifier network using a recurrent neural network (RNN) to judge whether the given time sequence is real or generated. This is no different from the discriminator networks used in GANs. The metric is given by:

$$(abs(classification\, accuracy - 0.5)) \tag{3.3}$$

3.2.8 Selection of parameters

Eckerli (2021) states that his experimental framework intended to use the GANs with the hyperparameters as suggested by the original authors; 'the black box' approach (section 3.2.4). One hyper parameter that was tested here is the number of epochs that the GANs were allowed to train on, namely 200 or 1000 epochs. As can be seen from the figure 3.2 below showing real vs. generated time series, the plots generated after 200 epochs are more realistic in terms of price excursions that those generated using 1000 epochs. This is an instance that highlights the fact that the loss-functions in GANs settle very quickly and thus the GAN has no way of determining when to stop and will thus carry on training indefinately. This is related to the evaluation problem mentioned in section 2.7.

Similar tests were carried out to establish whether the hyperparameters selected by Brock et al. (2018) for BigGAN gave suitable results when generating financial time series. The hyperparameters were:

1. Learning rates
2. Rl gradient penalty strength
3. Strength of the modified Orthogonal Regularization penalty in Generator

These preliminary tests using the orignal paper (Brock et al., 2018) hyperparameter ranges all indicated that the author's choice of hyperparameters were satisfactory.

3.2.9 Execution Details

The models were trained on the actual S&P500 time series of 5000 data points corresponding to the prices of the index from the 1st January 2000 to the end of 2020. The S&P500 was chosen as it condenses the performance of 500 of the largest companies listed on US stock exchanges. The index is thus a good benchmark on the performance of the US economy and it is widely used in the testing of financial models (Eckerli, 2021). The price time series was converted to log returns and this series was used to train the various

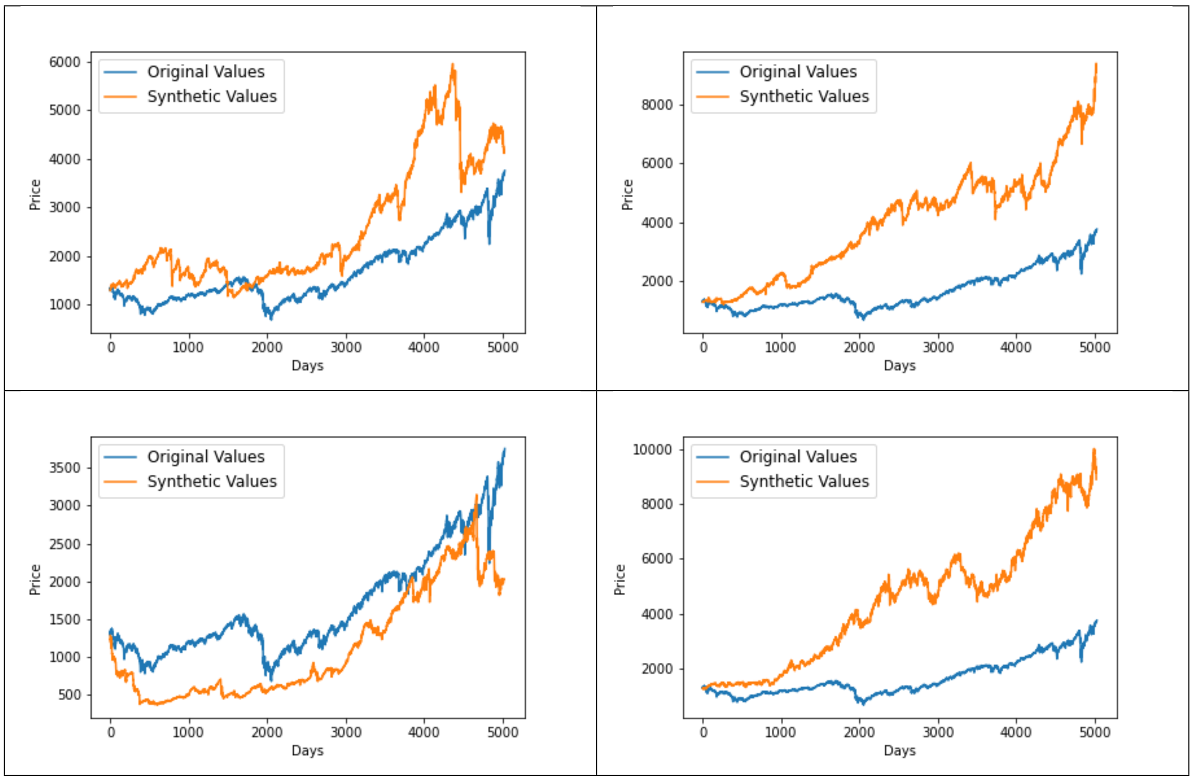


Figure 3.2: This figure shows real and synthetic time series for 200 epochs (left) and for 1000 epochs (right) for BigGAN (row 1), SigCWGAN (row 2)

GAN configurations. The resultant generated output was therefore a 20 year simulation of log returns.

3.3 Experiment 1b - Non-GAN method evaluation

3.3.1 Motivation

This experiment is intended to fulfil objective 1b as stated in section 1.2. The aim here is to appraise the quality of a number of non-GAN augmentation methods using the same evaluation criteria used in experiment 1a; namely the quantitative evaluation described in the paper by Yoon et al. (2019) followed by visual/qualitative assessments following the Eckerli (2021) and Takahashi et al. (2019) papers. This is being done to ensure that the GAN results from experiment 1a may be compared to those arising from the evaluation of non-GAN methods being carried out here. This will ensure that we may choose the most suitable synthesis method in order to proceed to experiment 2.

3.3.2 Actions undertaken

1. We use the evaluation methods developed by Yoon et al. (2019) to quantitatively evaluate a comprehensive selection of non-GAN methods.

2. We verify that the evaluateing obtained in task 1 is reflected in the quality of the visualisations using the frameworks of Eckerli (2021) and Takahashi et al. (2019).

3.3.3 Non-GAN Augmentation methods

Even though the majority of the literature surveyed used GANs as the preferred time series synthesis technology, there were papers that discussed other methods for time series synthesis (Fons et al., 2020), Fu et al. (2019), Liu et al. (2020) and Iwana and Uchida (2021a). Thus the necessity arose to appraise non-GAN methods of augmentation and compare their performance with the GAN results from experiment 1a. The Non-GAN augmentation methods that were used to achieve objective 1b were adapted from the code by Iwana and Uchida (2021b). The code was integrated into the Eckerli (2021) framework described in section 2.7.2 so that all the evaluation metrics (both visual and quantitative) operated seamlessly whether input was from a GAN or non-GAN generator.

3.3.4 Data

To replicate the chosen benchmark paper by YU and Li (2018), it was necessary to obtain the OHLC (open, high, low, close) data for various indices such as S&P500, SSEC, N225, HSI, and IXIC. Specifically, the S&P500 index was used as the reference data for experiments 1a and 1b. Following the recommendation of Ni et al. (2020) and their code, the OHLC data for these indices was sourced from the "realised library" provided by the Oxford-Man Institute. The acquired data was then stored in CSV files, with each file named after the corresponding asset and containing a column for dates.

3.3.5 Evaluation Criteria

The key function necessary to achieve objectives 1a and 1b (section 1.2) revolves around evaluating the fidelity of the generated time series compared to the original time series. This evaluation involves assessing both the statistical properties and, importantly, the temporal dependence of the time series. To accomplish this, two approaches have been employed. First, a series of visual assessments were conducted, following the methods described by Eckerli (2021), Takahashi et al. (2019), and utilizing the predictive and discriminative metrics developed by Yoon et al. (2019). These visual assessments complement the PCA and t-SNE plots utilized by Yoon et al. (2019), as described in section 2.7.2.

PCA was employed as one of the visual evaluation methods. It involved conducting a two-dimensional PCA using the sklearn Python library. The PCA process was applied to both the generated and augmented asset time sequences and compared against the real asset time sequence. If the generated sequences preserved the characteristics of the real data, the projections should exhibit significant overlap. The t-SNE implementation utilized in this context was also sourced from the sklearn library.

Visual assessments, as outlined by Eckerli (2021) and adapted from the provided code, were performed. Eckerli (2021) focused on comparing "stylized facts" in his evaluation. To enhance this process, the visual evaluation tools described in Takahashi et al. (2019) were also incorporated. The generated time series from section 3.2.5 were fed into a Python program that generated various graphs, enabling the assessment of the stylized facts detailed in section 2.7.2.

3.4 Experiment 2 - Improving daily return prediction using Synthetic data

3.4.1 Motivation

The second experiment is intended to fulfil objective 2 as stated in section 1.2. The aim here is to ascertain if the augmented data obtained by using the generative methods selected in the first experiment and added to real data actually provides model training and out-of-sample prediction improvement in the performance of the algorithm by YU and Li (2018).

3.4.2 Actions undertaken

1. The compilation of 20 augmented time sequences from the synthesised time sequences for all indices generated by the previous experiments.

2. The running of the benchmark paper code with the augmented time sequences from task 1.

3.4.3 Data

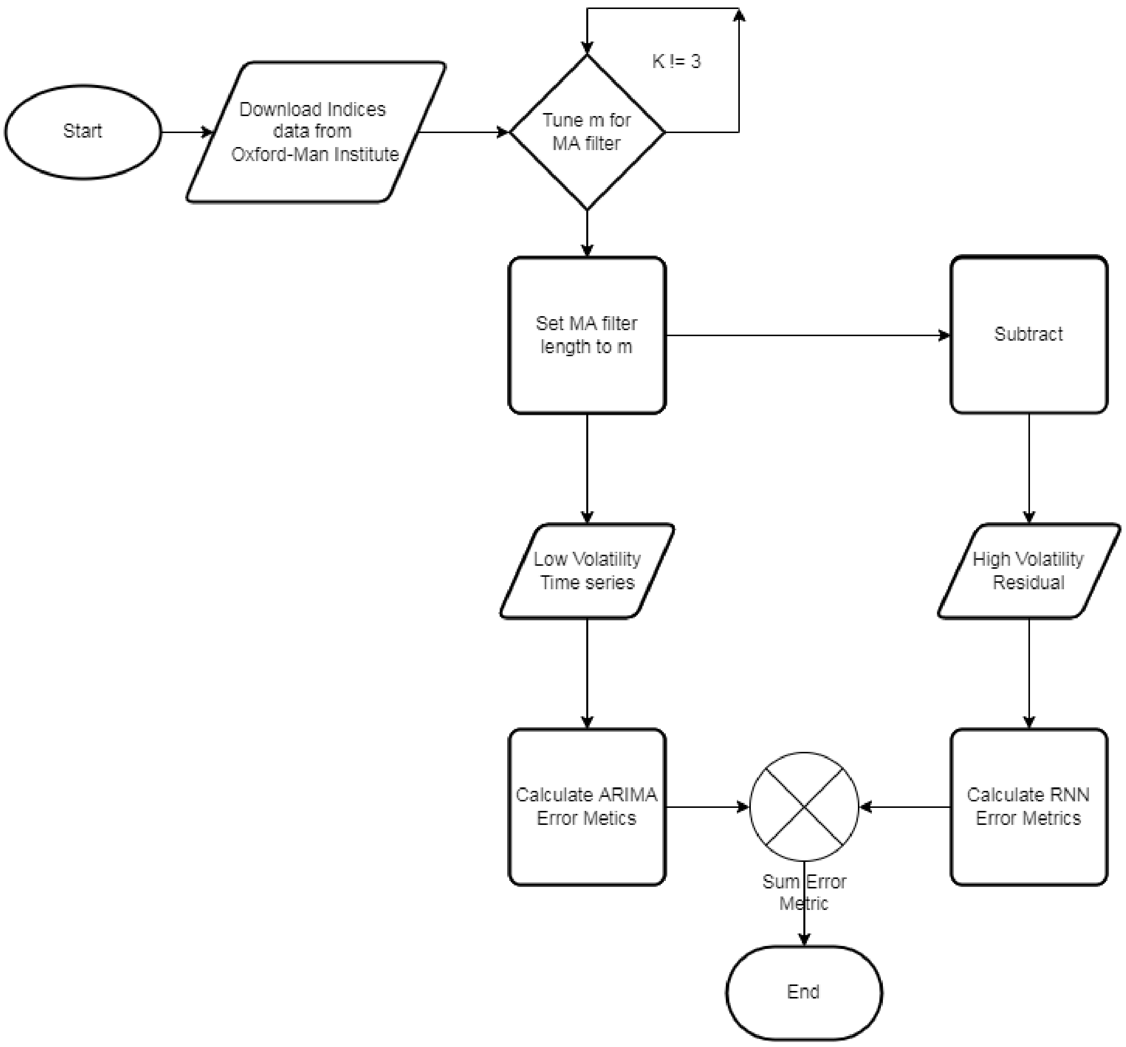

Start
Download Indices data from Oxford-Man Institute
Tune m for MA filter
K != 3
Set MA filter length to m
Subtract
Low Volatility Time series
High Volatility Residual
Calculate ARIMA Error Metics
Calculate RNN Error Metrics
Sum Error Metric
End

Figure 3.3: Experiment 2 Benchmark code flowchart

3.4.3.1 Data Description

The task to be carried out here follows from what was carried out in experiments 1a and 1b. This involved the production of synthesised time sequences obtained by the generation method that was found to give the best result when appraised by the Yoon et al. (2019) criteria (section 2.7.4). These synthesised time sequences were used to build the augmented time sequence data files that make up the inputs to the benchmark paper code. This augmented data consists of synthesised data followed by real data in the ratio of 3:1 one in one continuous CSV file.

Following from the generation of data described in section 3.2.5 all that is needed is to pass the particular CSV file to the benchmark algorithm code on the command line.

The experiment was carried out with 20 separately generated augmented time sequence files for each of the 5 indices (S&P500, SSEC, N225, HSI and IXIC). It was observed that 20 repetitions resulted in a stable figure of the standard deviation that the results generated, indicating that no substantial further variation would occur if more repetitions were carried out.

3.4.4 Replication of Benchmark Paper

The benchmark paper adopted in this dissertation is "Stock price prediction based on ARIMA-RNN combined model" by YU and Li (2018). The YU and Li (2018) paper uses daily closing prices of five indexes: the Standard & Poor's 500 index (S&P500), the NASDAQ index (IXIC), the Nikkei index (N225),the Hong Kong Hang Seng Index (HSI) and the Shanghai CompositeIndex (SSEC). The YU and Li (2018) paper describes how two time-honoured and well understood financial time series models (ARIMA and RNN) can be combined together to produce an improved model. The experiment described here is intended to achieve the aims set out in obective two (section 1.2).

A number of papers were considered before choosing the one by YU and Li (2018). Feng et al. (2019) studied the relationships between stock on the Nasdaq and NYSE exchanges in order to be able to evaluate over 2700 stocks on these markets. The trading strategy adopted was to buy the top 50 stocks and sell them after a day. The paper by Mudassir et al. (2020) looked at the prediction of Bitcoin prices. They used ANN and LSTM networks as their prediction models trained on multiple return intervals. Krauss et al. (2017) and Fischer and Krauss (2018) implemented a trading system that evaluated the S&P500 assets by focusing on the cross-correlations between these assets. All these benchmark paper candidates were discarded because they all necessitated the generation of large multiples of correlated time series. The fact that it is quite feasible to train DL networks with a high number of features does not necessarily mean that it is equally computationally viable to have a GAN produce the required data.

The code implementing the YU and Li (2018) paper was an adaptation of the code by Christopher Naimo that is available on GitHub[1]. This code required one input file in OHLC format. The data was the same as that used in Experiments 1a and 1b (section 3.2.3.1) and required no further processing. Python is used throughout the analysis process with the help of the libraries mentioned in section 3.6.

[1]https://github.com/cnaimo/hybrid-ARIMA-LSTM-model

3.4.5 Evaluation Criteria

The evaluation criteria used in this experiment follows that which was described in the benchmark paper YU and Li (2018), namely the MAE,MSE and MAPE error metrics that are briefly overviewed below.

3.4.5.1 Error Metrics

MAE Mean Absolute Error (MAE): MAE is a measure of the average error in a set of predicted values (i.e. between the prediction and observation), independently of their direction. The average is taken over the given data where all calculated differences have equal weight.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|x_i - x| \tag{3.4}$$

MSE Mean Squared Error (MSE): The mean squared error (MSE) also known as mean squared deviation (MSD) of a regressed or predicted quantity is given by the average of the squares of the errors; that is, the average squared difference between the predicted values and the observation.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \tag{3.5}$$

MAPE Mean Absolute Percentage Error (MAPE): The mean absolute percentage error (MAPE) is the average of the absolute percentage errors of the predicted values. An error is understood to be the predicted value subtracted from the actual, observed value. In Mape the absolute percentage errors are summed to give the metric.

$$MAPE = \frac{1}{n} \times \sum \left| \frac{actual\,value - forecast\,value}{actual\,value} \right| \tag{3.6}$$

There is a distinction that needs to be made clear regarding the meaning of the terms "synthetic data" and "augmented data". Synthetic data contains only artificial data that is produced by the procedures mentioned in this study (the GAN and non-GAN methods in sections 2.5 and 2.9.1). Augmented data is data where synthetic data is interspersed with real data resulting in a dataset that is larger than the original. The augmentation carried out in this study has not been used to enlarge the training set, but to ascertain whether synthetic data could satisfactorily substitute the real data in the training stage.

For the purpose of replication, the procedure described in the benchmark paper was carried out using real data so as to accurately follow the procedure described by YU and Li (2018).

Augmented data is what has been used in experiment 2 (section 3.4) to verify that the synthetic data behaved at least as well as the real data in terms of the evaluation criteria described in section (3.2.6). Thus, the procedure adopted by the benchmark paper was carried out using data that was all synthetic in the training phase and all real in the testing phase. Since the code used a single integral data set and proceeded to split this data into a training set and a testing set in the ratio of 3:1 (YU and Li, 2018), the augmented data was compiled by adding synthetic data followed by real data in the same ratio. The code was not modified to do this to preserve the integrity of the benchmark paper code, so the augmented data compilation was carried out separately.

The two underlying models been described in section 2.6.3 and section 2.6.4 in the Literature Review chapter.

3.5 Hardware Considerations

Development and experiments were carried out on a 64 core server motherboard with an NVIDIA Tesla K80 with 24GB of ram running Ubuntu 20.04.5 LTS. The development environment used was Anaconda version 4.13.0. Some of the code developed was based on that supplied by the various paper's authors. These were used in separate anaconda environments each built with the recommended python version and "requirements" file.

3.6 Software and Libraries

The libraries listed below have been employed extensively in the code used in this dissertation. Although there is some overlap in the functionality of the libraries listed below, only minimal changes were carried out to the code adopted from the various papers used in this investigation. Since the code was usually based on a particular framework (Tensorflow, Keras or Pytorch) substituting one framework for another would have involved a very onerous code rewrite that would be very prone to error and might not have preserved the operation of the original code.

Numpy: Numpy is primarily used to handle numeric arrays with a syntax that is very similar to Python arithmetic.

Pandas: Pandas enables the manipulation of tabular data in a manner that is reminiscent of SQL. It greatly aids the construction of data by allowing the merging of data from diverse sources and its manipulation, selection and pre-processing. It is used seamlessly with Numpy in the manipulation and processing of data in this work.

Matplotlib: Matplotlib is the python workhorse when it comes to data visualisation. The plots that have been presented here are mostly generated using this library.

Scikit-learn: Scikit-learn is a python library with a consistent API that is made up of a set of tools for ML and statistical modeling that includes regression, classification, dimensionality reduction and clustering capabilities.

Tensorflow, Keras and Pytorch: Tensorflow, Keras and Pytorch are the three libraries that were used to build the various Deep Learning networks used in this disseratation. Tensorflow has been developed by Google whereas Pytorch was developed by Meta (Facebook). Keras was originally an open source overlay to Tensorflow that has now been incorporated in Tesorflow 2 by Google. These libraries allow the programmer to construct almost any Neural Net architecture layer by layer, usually using just one line of code per layer.

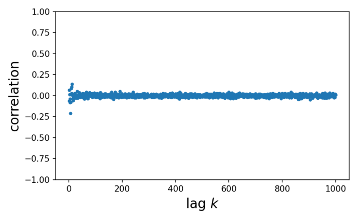
4 Results and Evaluation

4.1 Lack of Evaluation Methods

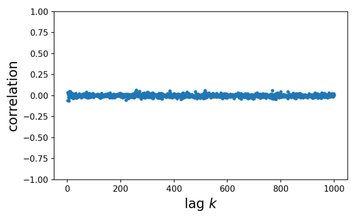
Barua (2019); Borji (2019); Ducoffe et al. (2019); Eckerli (2021); Jabbar et al. (2021); Koochali et al. (2020); Simonetto (2018) all have emphasised the fact that GAN evaluation, particularly in applications outside Computer Vision is highly problematic. The Inception score (Salimans et al., 2016) and Frechet Inception Distance (Heusel et al., 2017) metrics, that have become the accepted evaluation metrics for GANs used in CV are not applicable in the case were GANs are used to generate time series. An overview of this issue is given in section 2.7.

4.2 Visual Evaluations

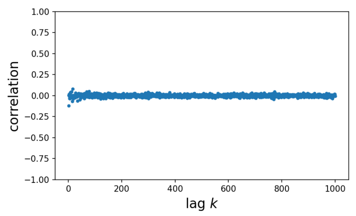
The plots were carried out using the time series generated by BigGAN and SigCWGAN after being trained on the S&P500 time series. S&P500 was chosen as this series is used in all trials and experiments performed here. The figures in this section should be compared with figure 2.7. This set of figures are the one published in Takahashi et al. (2019) paper and show the plots the authors generatted for the S&P500 index.



(a) SigCWGAN

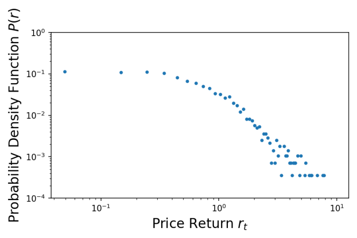


(b) BigGAN

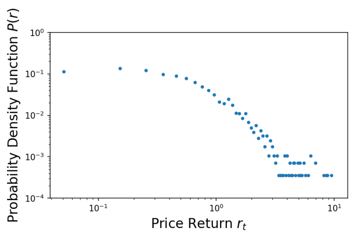


(c) Real S&P500

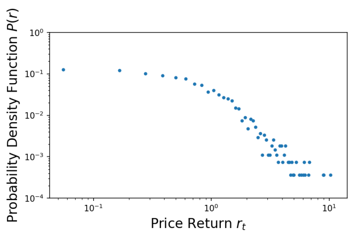
Figure 4.1: Linear Unpredictability



(a) SigCWGAN

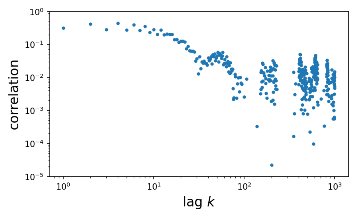


(b) BigGAN

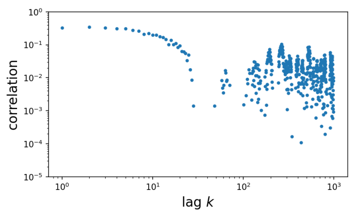


(c) Real S&P500

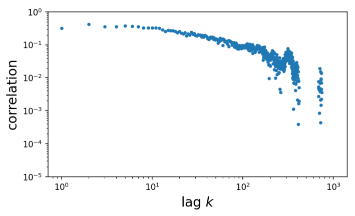
Figure 4.2: Heavy Tailed Distribution



(a) SigCWGAN



(b) BigGAN



(c) Real S&P500

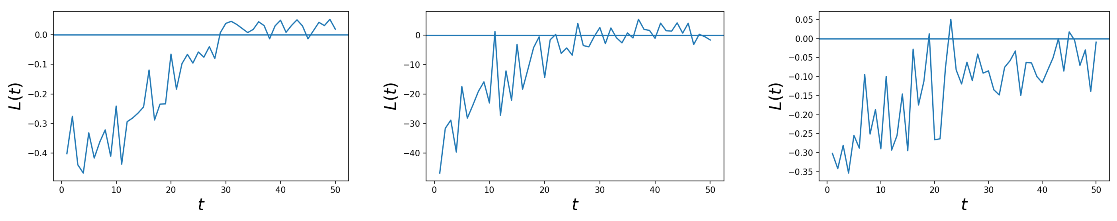
Figure 4.3: Volatility Clustering

Linear unpredictability: Autocorrelation of asset price is plotted against increasing time lags. The plots shown here in figure 4.1 show that there is essentailly no correlation at any time lag. This is an often quoted behavior that is often used to confirm the Efficient Market Hypothesis, section 2.2 i.e. that prices carry no information from the past.

Fat-tailed distribution: As has been discussed in section 2.7.2 the phenomenon of fat (or heavy) tails occurring in the distribution of returns may be observed using two different methods. The plots in figure 4.2 exhibit a power law decay as described in section 2.7.2. The Kurtosis figure 4.9 also highlights this phenomenon.

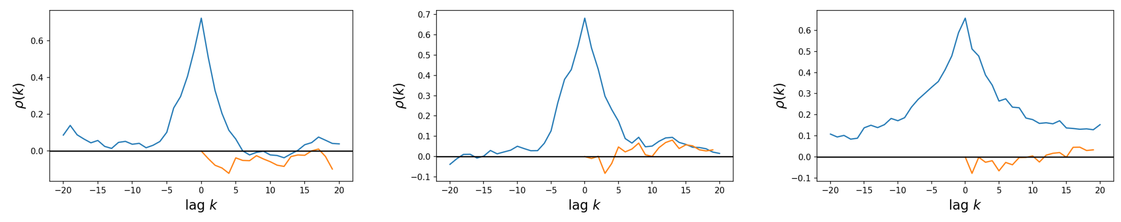
Volatility clustering: Figure 4.3 shows the average auto-correlation of price return in a log–log scale of the GAN synthesised data that can be compared with that of the real S&P500 time series. The power law decay is quite clear in figure 4.3 c and less so in figures 4.3 a and b. These plots indicate the effect of temporal characteristics (i.e. time dependence) that are typical in financial time series.

Leverage effect: All plots in Figures 4.4 show the leverage effect where they show a diminishing exponential decay (i.e. tending to zero), but it is more evident in the generated plots rather than the S&P500 plot.



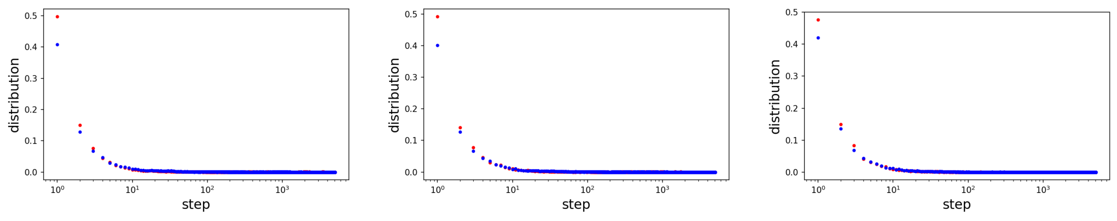
(a) SigCWGAN (b) BigGAN (c) Real S&P500

Figure 4.4: Leverage Effect



(a) SigCWGAN (b) BigGAN (c) Real S&P500

Figure 4.5: Coarse-Fine Volatility

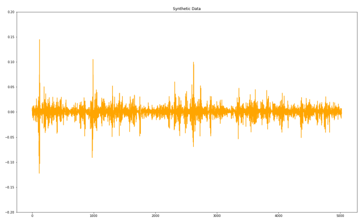


(a) SigCWGAN (b) BigGAN (c) Real S&P500

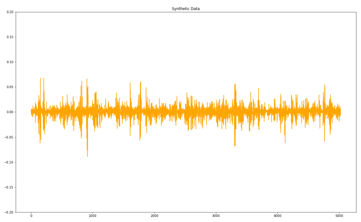
Figure 4.6: Gain-Loss Asymmetry

Coarse-fine volatility correlation: In figure 4.5 $\tau = 5$ is chosen for weekly time-scales. The asymmetry in the S&P500 plot at k = 1 is visible as the values move away from the zero level shown by the horizontal black dashed line. As has already been stated in this metric's overview 2.7.2 and also stated by Takahashi et al. (2019), this phenomenon is hard to discern in the behavior of single asset plots.

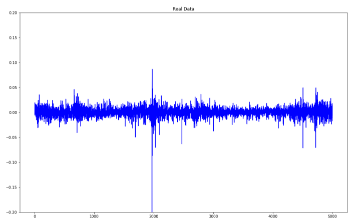
Gain/loss asymmetry: The results of our trials may be seen in figures 4.6. The plots do not satisfactorily show the purported behaviour that is evident in the Takahashi paper where the superimposed plots exhibit an evident shift.



(a) SigCWGAN

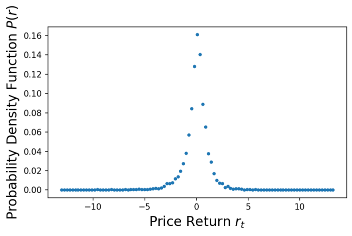


(b) BigGAN

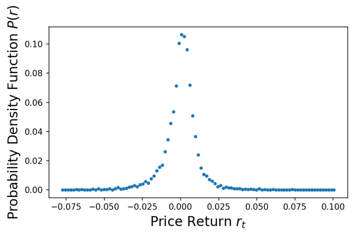


(c) Real S&P500

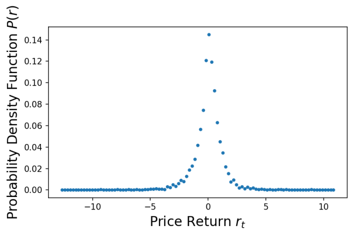
Figure 4.7: Returns



(a) SigCWGAN

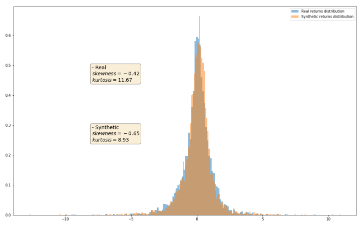


(b) BigGAN

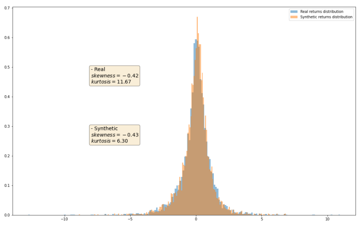


(c) Real S&P500

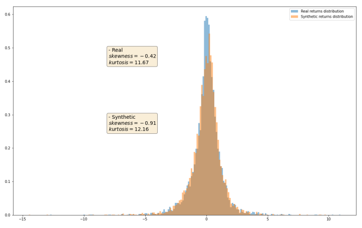
Figure 4.8: Linear Distribution



(a) SigCWGAN



(b) BigGAN



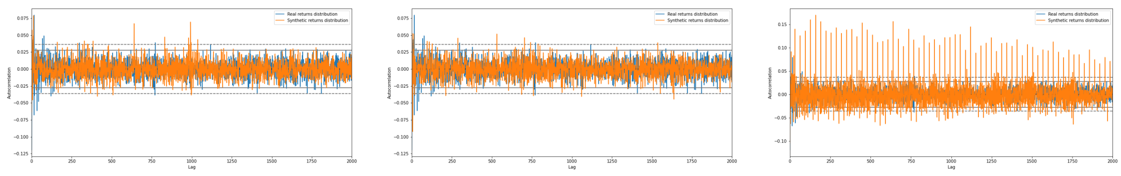
(c) WGAN-gp

Figure 4.9: Probabiity Density Function

Analysis of returns: The observation of figure 4.7 and 4.8 shows certain important characteristics. Assets returns, both generated and real are centred at zero and extend between -10% and 10%. These plots are very similar in their volatility distributions to that observed in the S&P500 plot.

4.2.0.1 Probability distribution, Kurtosis and Skewness:

In figure 4.9 the distribution of returns for S&P500 is superimposed on that of the GAN generated time sequence with the associated figures for skewness and kurtosis. The figure 4.9 shows that the two distributions overlap quite tightly implying that the synthetic sequence displays aggregational gaussianity (i.e. normal distribution) as does the real dis-



(a) SigCWGAN (b) BigGAN (c) WGAN-gp

Figure 4.10: Autocorrelation

tribution. This behaviour of the generated data is as would be expected of a satisfactorily performing GAN.

Skewness: The discussion of skewness in the literature review section 2.7.2.1 describes how skewness determines how by much the given distribution is off center. The observed prevalence of more significant downward movements in price implies that the skewness should be negative. The figures obtained confirm this.

Kurtosis: What should be noted in the statistics shown in table 4.1 is that the skewness datum for BigGAN is practically identical to the real result whereas the kurtosis figure is smaller. Both these figures substantially improve on the figures of SAGAN, DCGAN and WGAN-gp as reported by Echerli in his paper.

	Real S&P500	BigGan	SAGAN *	DCGAN *	WGAN-gp *
Skewness	-0.42	-0.43	-0.27	-0.14	-0.26
Kurtosis	11.69	6.3	3.62	2.38	5.6

Table 4.1: Skewness and Kurtosis Comparison. * Indicates Eckerli Paper figures

Autocorrelation: The WGAN-gp ACF plot that Eckerli shows in his paper has been replicated in figure 4.10. It has been reproduced here to show that WGAN-gp fails this test as there are several shifts that have an ACF that is well beyond the thresholds that are represented by the black dashed lines. BigGAN and SigCWGAN, in contrast, overshoot these limits as often as the real S&P500 sequence does, indicating the superior performance of BigGAN and SigCWGAN.

4.3 Experiments

Here we present the results arising from the application of Yoon's evaluative methodology (section 2.7.4) when applied to experiments 1a and 1b.

4.3.1 Data Preparation

As mentioned in section 3.2.3.1 the data used for all the experiments carried out here were obtained from the Oxford-Man Institute's "realised library". The fidelity requirement is essential in the training of GANs. This means that very little preprocessing is carried out on the raw OHLC (open, high, low, close) data as operations like SMOTE would affect the distribution and temporal characteristics of the input (training) data and consequently the fidelity of the GAN output. The datapoints that would be interpreted as NAN by the python code and libraries where deleted.

The data was loaded into a Pandas dataframe and a log return column was added. This column was necessary when generating synthetic data is synchronised with the real data (i.e. having the same starting price) as described in section 3.2.3.2.

Experiments 1a and 1b did not require augmented data but only synthetic data.

4.3.2 Experiment 1a

The Table 4.2 shows the results of Experiment 1a showing the discriminative and predictive sores of both GANs. As stated in the objective 1a in section 1.2, the aim of this experiment is to find the GAN architecture that gives the best performance according to the selected evaluation criteria as discussed in section 3.2.7. The trials were carried out for the five indices that formed the basis of experiment 2 below.

The table 4.2 shows that the best performing GAN is SigCWGAN.

The variability across indices was evident in all experiments performed and confirms the comment of Qiu et al. (2006) where he states that leverage effect is dependent on the observed market. We are quoting Qiu as a case where the researcher has found that certain (if not most) stylised facts exhibit different behaviours depending on the market being studied.

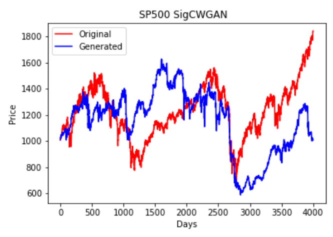
Another feature that arises from the observation of the figures in table 4.2 is that the results of each index are fairly close but the values across indices diverge. This, yet again confirms the notion behind Qui's observation.

The figures 4.11 and 4.12 show Real vs Generated:

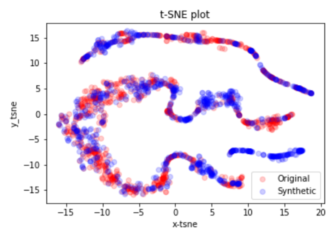
- Daily price plot

Index	GAN	Discriminative	Std. Dev.	Predictive	Std. Dev.
S&P500	SigCWGAN	**0.11470**	0.04217	**0.00558**	0.00068
	BigGAN	0.18637	0.05458	0.00598	0.00080
HSI	SigCWGAN	**0.13514**	0.09469	**0.01651**	0.02228
	BigGAN	0.22540	0.06360	0.06248	0.03253
IXIC	SigCWGAN	0.20608	0.04765	**0.01496**	0.00594
	BigGAN	**0.19445**	0.0651	0.04401	0.04059
N225	SigCWGAN	**0.12892**	0.05896	**0.01652**	0.00352
	BigGAN	0.14358	0.05589	0.07263	0.00833
SSEC	SigCWGAN	0.16983	0.04765	0.01052	0.00180
	BigGAN	**0.16533**	0.07170	**0.00976**	0.00126

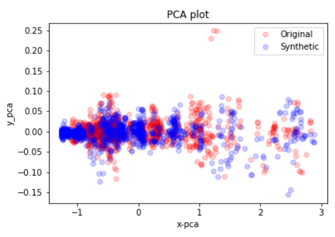
Table 4.2: S&P500 Yoon Evaluation



(a) Price Series

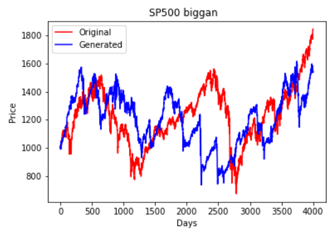


(b) t-SNE Plot

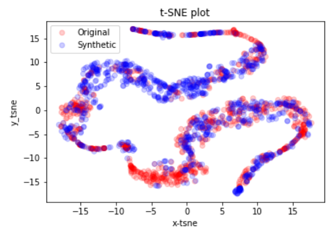


(c) PCA Plot

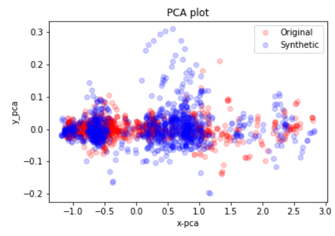
Figure 4.11: SigCWGAN



(a) Price Series



(b) t-SNE Plot

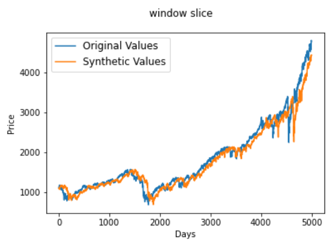


(c) PCA Plot

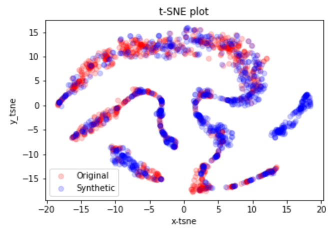
Figure 4.12: BigGAN

- t-SNE plot
- PCA plot

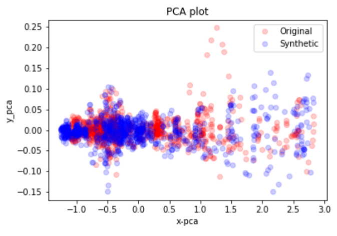
for BigGAN and SigCWGAN.
The plots shown are for the GANs used in this study, namely SigCWGAN and BigGAN.



(a) Time Series

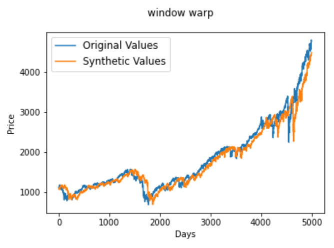


(b) t-SNE Plot

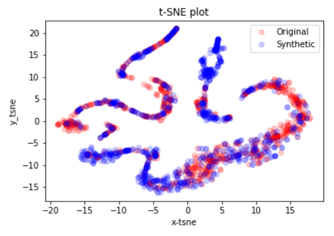


(c) PCA Plot

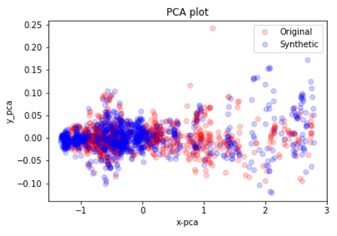
Figure 4.13: Window Slice



(a) Time Series



(b) t-SNE Plot



(c) PCA Plot

Figure 4.14: Window Warp

The Daily price plot shows that both the GANs studied here give plots that have a behaviour that is satisfactory in terms of reversion to the mean of returns and price excursion (i.e. plots do not rise or fall in a way that the real price never does)

The t-SNE plot is probably the most revealing in that careful observation will show that the SigGan and BigGan plots show a substantial amount of superposition of the red and blue spots with smaller areas that are predominantly of one colour. The spots that are pink indicate an overlap of the blue and red spots.

As mentioned in the overview of PCA in section 2.7.3, this plot is inferior to t-SNE in its ability to show the similarity (or lack thereof) between real and generated asset data sets. The SigCWGAN PCA plot does exhibit an evident overlap between blue and red spots.

4.3.3 Experiment 1b

Table 4.3 shows the results of the Yoon evaluation metrics when carried out using the non-GAN augmentation methods illustrated by Iwana and Uchida (2021a) as described in section 2.9.1. This experiment was carried out to fulfil objective 1b as expressed in section 1.2 where the synthetic performance of non-GAN methods are to be compared to that of GANs. As can be seen by comparing the results shown in table 4.3, none of

Method	Discriminative	Predictive
SigCWGAN	**0.11472**	**0.00564**
window_warp	0.14963	0.10504
window_slice	0.15412	0.10614

Table 4.3: Experiment 1b Results

these methods approached the performance of SigCWGAN generated S&P500 time sequences. This affirms, quite conclusively, that SigCWGAN has surpassed the results in the experiments carried out here and is thus the synthesis method that will be adopted to carry out experiment 2.

The diagrams in figure B.5 and figure B.9 show Time Series, t-SNE and PCA plots for both the implemented non-GAN synthetic methods.

The other non-GAN synthetic methods give results that are in a tight range both in the discriminative score and in the predictive score. This is consistent with the moderate plot excursions seen in time series figures for these methods.

The stylised fact plots for all the non-GAN methods investigated here may found in Appendix A. Preliminary visual assesments for non-GAN methods are given in appendices A and B.

4.3.4 Experiment 2 and the Benchmark Model

4.3.4.1 Benchmark Model

The benchmark model used in this study is due to YU and Li (2018)) and has been described in section 2.6.5. The preparation of data for the replication phase is outlined in section 3.4.4.

The results shown in table 4.4 were carried out using a code implementing the YU and Li (2018) paper by Christopher Naimo that is available on GitHub[1]. This code used LSTM instead of the RNN network that the benchmark paper authors used. The code was thus modified to use an RNN network. As can be seen from table 4.4 the MSE and MAE replication results improve on the ones quoted by the Shui paper for all indices except for the SSEC index. The MAPE results show that replication performs better than the benchmark for the SSEC and N225 indices. The MAPE result is close for the S&P500 index but not for the HSI and IXIC indices. The MAPE results are still very good in that since any result below 20 is seen in the industry as acceptable.

[1]https://github.com/cnaimo/hybrid-ARIMA-LSTM-model

		MSE	MAE	MAPE
HSI	Shui	13,650.66	86.94	0.37
	Replication	**11,241.46**	**78.93**	**0.55**
IXIC	Shui	616.32	18.52	0.35
	Replication	**606.48**	**19.30**	**0.62**
N225	Shui	82,102.50	160.78	0.94
	Replication	**65,884.52**	**203.43**	**0.76**
S&P500	Shui	459.98	19.08	0.30
	Replication	**408.98**	**15.74**	**0.36**
SSEC	Shui	111.16	8.88	0.75
	Replication	**246.00**	**11.76**	**0.45**

Table 4.4: Benchmark paper replication

4.3.4.2 Experiment 2

Experiment 2 has been carried out to fulfil objective 2 as described in section 1.2.

		MSE	MAE	MAPE
HSI	Replication	11,241.46	78.93	0.55
	Aug	**4,390.65**	**56.89**	**0.57**
IXIC	Replication	606.48	19.30	0.62
	Aug	**597.26**	**18.93**	**0.61**
N225	Replication	65,884.52	203.43	0.76
	Aug	**21,455.69**	**125.48**	**0.78**
S&P500	Replication	408.98	15.74	0.36
	Aug	**67.10**	**6.35**	**0.38**
SSEC	Replication	246.00	11.76	0.45
	Aug	**58.96**	**7.28**	**0.74**

Table 4.5: Experiment 2

The results shown in table 4.5 show that the augmented data produced by SigCW-GAN gives significant improvement over that obtained using the original data as seen in table 4.4. MSE and MAE results show considerable improvement from a minimum of 2% (IXIC, MAE) to a maximum of 76% (SSEC, MSE). The average MSE improvement is 58% whereas the average MAE improvement is 33%. The MAPE results for HSI, IXIC, N225 and S&P500 within a 3% margin, with the MAPE SSEC result being the only significant outlier. The unusual behaviour of the SSEC both here and in the replication require further investigation.

4.4 Summary

Experiments 1a and 1b were intended to search for a financial time series synthesis procedure that performed as closely as possible to real time sequences. The process involved visual as well as quantitative evaluation methods. The procedure to chose the evaluation methods themselves was in itself complex. After considering various visual methods that focused on stylised facts, the methods described by Yoon et al. (2019) were adopted as the definitive discriminant. The visual methods were used to corroborate the numeric results.

Table 4.2 presents the results of Experiment 1a, which aimed to determine the GAN architecture with the best performance based on selected evaluation criteria. The goal was outlined in objective 1a in section 1.2. SigCWGAN emerged as the best-performing GAN according to the provided table.

Table 4.3 demonstrates the results of the Yoon evaluation metrics for non-GAN augmentation methods, fulfilling objective 1b as expressed in section 1.2. None of these methods approached the performance of SigCWGAN in generating S&P500 time sequences, confirming SigCWGAN's superior performance. Hence, SigCWGAN was employed in Experiment 2.

The results shown in table 4.5 show that the augmented data produced by SigCWGAN gives significant improvement over that obtained using the original data as seen in Table 4.4. MSE and MAE results show considerable improvement from a minimum of 2% (IXIC, MAE) to a maximum of 76% (SSEC, MSE). The average MSE improvement is 58% whereas the average MAE improvement is 33%. The MAPE results for HSI, IXIC, N225 and S&P500 within a 3% margin, with the MAPE SSEC result being the only significant outlier.

As can be seen from the results obtained in experiment 2 the augmentation process using SigCWGAN of 5 market indexes gave very satisfactory results with regard to MSE, MAE and MAPE metrics. This strongly suggests that augmentation using SigCWGAN can be used to improve the performance of financial prediction models.

5 Conclusions

5.1 Revisiting the Aims and Objectives

Objective 1a: Our goal is to examine and contrast GAN architectures proposed in the literature for creating synthetic financial data. An overview of literature dealing with general data augmentation methods and data synthesis methods using GANs was carried out not only to establish a suitable set of synthesis and augmentation techniques but also to catalogue the various evaluation methods that were explored by the various authors. The investigation into evaluation methods revealed that this was carried out along two broad classes: visual evaluation and quantitative evaluation.

PCA and t-SNE seemed to garner the favour of the largest number of authors. These plots compare two data sets; the real and generated asset price time sequences and enable the observer to judge how close the stochastic behaviour of these sequences is. These plots do not, however, reveal if the sequences are displaying the behaviour that characterises financial time sequences. This behaviour has come to be known as "stylised facts" and financial time sequences have specific characteristics that are missing from a random walk generated under a Gaussian distribution. Thus, in addition to PCA and t-SNE a set of stylised fact plots was also used in our visual evaluations.

Following the practice adopted by a number of authors surveyed, it was decided to use the discriminative and predictive metrics suggested by Yoon et al. (2019) as the quantitative metric to assess the quality of the time sequences generated here. The "discriminative" value measures how well the neural network used in this metric is able to recognise portions of the generated time sequence as being similar to those in the real time sequence. The "predictive" metric measures if the predictive neural network can foretell the value of the n+1 data point when the previous n data points are input. Both the discriminative and predictive metrics are reasonable in that they do correlate with the utility of the application at hand.

The objective designated as 1b used the same criteria outlined above to determine the best non-GAN augmentation method. It is worth noting that these are augmentation methods and not synthetic methods in that the original time sequence is modified and not regenerated.

Objective 2: To determine whether integrating GANs with a financial prediction model can enhance the accuracy of stock price predictions. The paper by YU and Li (2018) required the completion of two tasks; the replication of the paper and the running of the ARIMA/RNN prediction system on the augmented data generated by the system chosen arising from the completion of objectives 1a and 1b. The two tasks were accomplished in a satisfactory manner.

Experiments 1a and 1b aimed to synthesize financial time series to emulate real data. Evaluation involved visual and quantitative assessments, with a focus on stylized facts. By using the discriminative and predictive metrics proposed by Yoon et al. (2019) as the definitive measures (that were then validated visually), SigCWGAN emerged as the top-performing GAN in experiment 1a. Experiment 1b showed that SigCWGAN significantly surpassed the non-GAN methods tested. Thus SigCWGAN was chosen as the augmentation technique for experiment 2. Results showed SigCWGAN's effectiveness across five market indexes using the standard metrics MSE, MAE, and MAPE, yielding satisfactory outcomes. The results obtained by using augmented data were particularly creditable in that the prediction error metrics obtained showed an appreciable improvement on the figures obtained both by YU and Li (2018) and our replication of that paper.

5.2 Critique and Limitations

Evaluation in Generative Adversarial Networks (GANs) poses several challenges and can be problematic for a number of reasons. GANs generate data that aims to be realistic, but defining objective metrics to assess the quality and fidelity of generated samples is difficult. Traditional evaluation metrics like accuracy or precision-recall are not directly applicable in GANs since there is no ground truth for comparison.

Evaluation in GANs often involves subjective human judgment. Visual inspection by humans is commonly used to assess the quality of generated samples, but this approach is inherently subjective and can vary among different evaluators. GAN evaluation often involves striking a balance between the diversity and quality of generated samples. Metrics that favour diversity might lead to low-quality samples, while those emphasizing quality may restrict the diversity of generated outputs. Evaluating GANs on large-scale datasets can be computationally expensive. Moreover, GANs may be sensitive to the specific characteristics of the training dataset, leading to biased evaluation results that may not generalize well to unseen data.

The question of evaluation is not unique to GANs, or indeed many other technologies. A telling comparison, from outside academia, may be made when observing how PC's, for example, are evaluated in the wider press. Here a number of benchmark measurements are adopted without there being any absolute standard. PC reviewers are familiar with what their community is using and try to use measurements that convey an analysis that is both understood and accepted by the this community. At the time of writing of this dissertation the community reviewing GAN performance was certainly smaller that for PC's, but this will certainly be growing as evidenced by the expansion that is occurring in the realm of AI. Another development that will very likely happen in the future is the trial and adoption of more and better metrics. It is, in our opinion, a mistake to try and reduce evaluation to one scalar value. Different metrics will highlight different qualities of the data generated by GANs. Thus, just as there is an evolution in the quality and application of GANs, there will very likely be a parallel evolution in GAN evaluation methods.

There is no doubt that the question of the evaluation of synthetic financial data still needs to be resolved to the satisfaction and consensus of most researchers in this field. The solution adopted here, namely to use the predictive and discriminative metrics by Yoon et al. (2019) may be seen as simply running one simple prediction system to obtain an early evaluation rather than using the prediction system that is being studied to do the same job. From the work carried out here it was evident that having an evaluation stage brought few advantages in terms of speed or accuracy. Thus the issue of evaluation is still open in our view.

A limitation that is unavoidable in the execution of research in pursuit of an academic degree is the constraint imposed by the amount of time that is available to carry out the activities necessary. This time constraint imposed a limitation of the possible scope in a number of areas. With regard to evaluation a selection from the possible methods had to be made, possibly excluding metrics that may have superior performance. Another limitation imposed by the time constraint is the range of input data that was investigated. Individual company stock data and data from diverse markets like futures, crypto currencies or commodities were not investigated in this work. Finally, it must be stated that even though a considerable effort was made to select the best GAN architectures for this inquiry, there is no guarantee that other architectures that may have merited attention were not overlooked.

5.3 Future Work

A few benchmark papers (Feng et al., 2019; Fischer and Krauss, 2018; Krauss et al., 2017; Mudassir et al., 2020) were excluded from this work due to the requirement of large and multivariate datasets, which could pose processing capacity challenges for the chosen computational infrastructure. However, there is a growing trend in utilizing GANs for predicting financial data, even though GAN generation demands more processing power than traditional machine learning (ML) prediction. As processing power becomes more accessible, more powerful synthetic data generation systems using GANs will likely emerge.

Expanding the research beyond computing limitations, algorithms should also focus on generating larger digital artefacts, such as images, videos, or simulated financial data. By enabling GANs to produce larger datasets, the work conducted in this study can be extended to larger multivariate datasets. Additionally, considering alternative architectures like Reinforcement Learning Networks (RL) may prove beneficial. RL networks trained on augmented data should be considered as a future endeavour.

One area that deserves attention in the financial field is the analysis of the wide variability in prediction and discrimination metrics of ML prediction systems across different assets and indices. Understanding the reasons behind these differences in accuracy, such as agent behaviour, prevalent trading strategies, or trading intensity, could provide valuable insights. GANs, known for their ability to extract and reproduce underlying stochastic behaviours, could potentially crystallize this information for comparative analysis and deductions. Architectures such as RL, Conditional GANs, and Attention Mechanisms are gaining attention and could help address these issues. RL stands out due to its ability to learn through interaction with an environment, making it effective in complex and dynamic environments with scarce labelled training data. RL agents consider long-term consequences, optimize behaviour based on rewards and penalties, balance exploration and exploitation, handle delayed rewards, adapt to changing environments, and make optimal decisions in uncertain and stochastic environments making them suitable for financial applications.

Conditional Generative Adversarial Networks (cGANs) extend the traditional GAN framework by introducing conditional input. The generator and discriminator receive additional information called the conditional vector, enabling controlled and targeted data generation. The generator uses random noise and the conditional vector to produce samples aligned with desired conditions. The discriminator considers the conditional vector to distinguish between real and generated data while evaluating how well the samples match the condition. By optimizing both the GAN loss and conditional loss, the generator learns to produce samples resembling real data that satisfy the desired conditions.

cGANs, as demonstrated in TimeGAN (Yoon et al., 2019), have been utilized in financial time sequence GANs to autonomously identify relevant features. They could also facilitate supervised teaching of relevant features, allowing the generation of time sequences with explicitly defined behaviours like what precedes the bursting of financial bubbles.

In Generative Adversarial Networks (GANs), attention mechanisms are often used to enhance the generation process and improve the quality of generated samples. Attention mechanisms in GANs typically involve introducing additional layers or modules within the generator network to focus on specific regions or features of the input data. These mechanisms allow the generator to allocate its generation capacity more selectively and effectively, emphasizing important parts of the data during the generation process. This is a desirable feature, as has been discussed above with regard to cGANs, where the network is able to focus on relevant features.

5.4 Potential Applicability

In the context of GAN-generated time series, there are several implications and potential applications in a range of areas. GAN-generated time series can be valuable in situations where data distributions change over time. By training GAN models on historical data, the generated time series can capture the characteristics and patterns of the past. This can be useful for tasks such as predictive analytics, anomaly detection, or generating synthetic data for simulation and scenario analysis in dynamic environments. Further research in this direction could focus on developing GAN architectures that adapt to changing data distributions, enabling the generation of time series that reflect the current context. Techniques like online learning, continual learning, or adaptive GANs could be explored to address the challenge of evolving contexts and facilitate the generation of time series that accurately capture the changing patterns. Adaptive GANs can have various architectures, such as incorporating attention mechanisms, utilizing adaptive optimization algorithms, or dynamically adjusting the model's architecture during training.

As the availability and size of time series datasets increase, there is a need to develop GAN models that can effectively handle larger-scale data. Scaling up GANs for time series generation requires addressing challenges such as computational efficiency, memory limitations, and model stability. Ongoing research in this area includes exploring distributed or parallel training techniques for GANs, model compression approaches, and architectural modifications that can handle larger input sequences. Additionally, incorporating attention mechanisms, such as self-attention or sparse attention, can help improve the scalability of GAN-generated time series by selectively attending to relevant parts of the sequence.

GAN-generated time series can be applied to entirely new domains or contexts. For example, trained GAN models on financial time series can be adapted for generating synthetic time series in different economic scenarios. Similarly, GANs trained on climate data can be used for generating synthetic weather patterns. Future research can explore the transferability of GAN-generated time series across different domains or contexts. This can involve investigating techniques like domain adaptation, domain generalization, or meta-learning to enable GAN models to adapt to new contexts with limited labelled data. Additionally, exploring interpretability and explainability techniques for GAN-generated time series can enhance trust and understanding when applying them to new domains.

5.5 Final Remarks

This work involved the search for a data synthesis method that would be able to substitute the real data used to train financial prediction systems. It must be said that the preliminary tests with various GAN solutions resulted in the prediction systems performing worse on synthetic data than on real data. It was very few GANs that gave usable results initially. The results obtained using SigCWGAN were very satisfactory not only because the synthetic data outperformed the real data in terms of prediction accuracy but also because it let us achieve our main objective i.e. showing that synthetic data can actually improve the performance of a financial prediction system.

References

Achelis, S. B. Technical analysis from a to z, 2001.

Agrawal, A., Gans, J. S., and Goldfarb, A. Exploring the impact of artificial intelligence: Prediction versus judgment. *Information Economics and Policy*, 47:1–6, 2019.

Aguilar-Rivera, R., Valenzuela-Rendón, M., and Rodríguez-Ortiz, J. Genetic algorithms and darwinian approaches in financial applications: A survey. *Expert Systems with Applications*, 42(21):7684–7697, 2015.

Akyash, M., Mohammadzade, H., and Behroozi, H. Dtw-merge: A novel data augmentation technique for time series classification. *arXiv preprint arXiv:2103.01119*, 2021.

Alexander, S. S. Price movements in speculative markets: Trends or random walks. *Industrial Management Review (pre-1986)*, 2(2):7, 1961.

An, G. The effects of adding noise during backpropagation training on a generalization performance. *Neural computation*, 8(3):643–674, 1996.

Antipov, G., Baccouche, M., and Dugelay, J.-L. Face aging with conditional generative adversarial networks. In *2017 IEEE international conference on image processing (ICIP)*, pages 2089–2093. IEEE, 2017.

Appadoo, S. S. Pricing financial derivatives with fuzzy algebraic models: A theoretical and computational approach. 2006.

Araujo, R. d. A. and Ferreira, T. A. A morphological-rank-linear evolutionary method for stock market prediction. *Information Sciences*, 237:3–17, 2013.

Arjovsky, M. and Bottou, L. Towards principled methods for training generative adversarial networks. *arXiv preprint arXiv:1701.04862*, 2017.

Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

Arnout, H., Bronner, J., and Runkler, T. Clare-gan: Generation of class-specific time series. In *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 01–08. IEEE, 2021.

Assefa, S. A., Dervovic, D., Mahfouz, M., Tillman, R. E., Reddy, P., and Veloso, M. Generating synthetic data in finance: opportunities, challenges and pitfalls. In *Proceedings of the First ACM International Conference on AI in Finance*, pages 1–8, 2020.

Athey, S., Imbens, G. W., Metzger, J., and Munro, E. Using wasserstein generative adversarial networks for the design of monte carlo simulations. *Journal of Econometrics*, 2021.

Atsalakis, G. S. and Valavanis, K. P. Forecasting stock market short-term trends using a neuro-fuzzy based methodology. *Expert systems with Applications*, 36(7):10696–10707, 2009.

Bachelier, L. Théorie de la spéculation. In *Annales scientifiques de l'École normale supérieure*, volume 17, pages 21–86, 1900.

Bai, S., Kolter, J. Z., and Koltun, V. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

Banko, M. and Brill, E. Scaling to very very large corpora for natural language disambiguation. In *Proceedings of the 39th annual meeting of the Association for Computational Linguistics*, pages 26–33, 2001.

Barber, B. M. and Odean, T. Boys will be boys: Gender, overconfidence, and common stock investment. *The quarterly journal of economics*, 116(1):261–292, 2001.

Barratt, S. and Sharma, R. A note on the inception score. *arXiv preprint arXiv:1801.01973*, 2018.

Barua, S. Towards improving the network architecture of gans and their evaluation methods, 2019.

Bell, D. E. Regret in decision making under uncertainty. *Operations research*, 30(5):961–981, 1982.

Bergmeir, C., Hyndman, R. J., and Benítez, J. M. Bagging exponential smoothing methods using stl decomposition and box–cox transformation. *International journal of forecasting*, 32(2):303–312, 2016.

Bezerra, P. C. S. and Albuquerque, P. H. M. Volatility forecasting via svr–garch with mixture of gaussian kernels. *Computational Management Science*, 14(2):179–196, 2017.

Bhatia, A., Chandani, A., and Chhateja, J. Robo advisory and its potential in addressing the behavioral biases of investors a qualitative study in indian context. *Journal of Behavioral and Experimental Finance*, 25:100281, 2020.

Biallas, M. and O'Neill, F. Artificial intelligence innovation in financial services. 2020.

Bishop, C. M. Training with noise is equivalent to tikhonov regularization. *Neural computation*, 7(1):108–116, 1995.

Black, F. Noise. *The journal of finance*, 41(3):528–543, 1986.

Blagus, R. and Lusa, L. Smote for high-dimensional class-imbalanced data. *BMC bioinformatics*, 14(1):1–16, 2013.

Blei, D. M., Ng, A. Y., and Jordan, M. I. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

Boden, M. A. *Artificial intelligence*. Elsevier, 1996.

Bodt, E., Cottrell, M., and Levasseur, M. Les réseaux de neurones en finance: Principe et revue de la littérature. *Finance*, 16(1):25–91, 1995.

Bollen, J., Mao, H., and Zeng, X. Twitter mood predicts the stock market. *Journal of computational science*, 2(1):1–8, 2011.

Bollerslev, T. Generalized autoregressive conditional heteroskedasticity. *Journal of econometrics*, 31(3):307–327, 1986.

Borji, A. Pros and cons of gan evaluation measures. *Computer Vision and Image Understanding*, 179:41–65, 2019.

Borovykh, A., Bohte, S., and Oosterlee, C. W. Conditional time series forecasting with convolutional neural networks. *arXiv preprint arXiv:1703.04691*, 2017.

Bouchaud, J.-P., Matacz, A., and Potters, M. Leverage effect in financial markets: The retarded volatility model. *Physical review letters*, 87(22):228701, 2001.

Bourlard, H. and Kamp, Y. Auto-association by multilayer perceptrons and singular value decomposition. *Biological cybernetics*, 59(4):291–294, 1988.

Box, G. E. and Jenkins, G. M. Time series analysis: Forecasting and control san francisco. *Calif: Holden-Day*, 1976.

Boyacioglu, M. A. and Avci, D. An adaptive network-based fuzzy inference system (anfis) for the prediction of stock market return: the case of the istanbul stock exchange. *Expert Systems with Applications*, 37(12):7908–7912, 2010.

Braun, J., Hausler, J., and Schäfers, W. Artificial intelligence, news sentiment, and property market liquidity. *Journal of Property Investment & Finance*, 2019.

Bredt, S. Artificial intelligence (ai) in the financial sector potential and public strategies. *Frontiers in Artificial Intelligence*, 2:

16, 2019.

Brock, A., Lim, T., Ritchie, J. M., and Weston, N. Neural photo editing with introspective adversarial networks. *arXiv preprint arXiv:1609.07093*, 2016.

Brock, A., Donahue, J., and Simonyan, K. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

Brophy, E., Wang, Z., She, Q., and Ward, T. Generative adversarial networks in time series: A survey and taxonomy. *arXiv preprint arXiv:2107.11098*, 2021.

Brynjolfsson, E., Rock, D., and Syverson, C. The productivity j-curve: How intangibles complement general purpose technologies. *American Economic Journal: Macroeconomics*, 13(1):333–72, 2021.

Bureš, M. Machine learning in algorithmic trading. 2021.

Burgess, N. Machine earning–algorithmic trading strategies for superior growth, outperformance and competitive advantage. *Outperformance and Competitive Advantage (March 29, 2021)*, 2021.

Bustos, O. and Pomares-Quimbaya, A. Stock market movement forecast: A systematic review. *Expert Systems with Applications*, 156:113464, 2020.

Cao, H., Tan, V. Y., and Pang, J. Z. A parsimonious mixture of gaussian trees model for oversampling in imbalanced and multimodal time-series classification. *IEEE transactions on neural networks and learning systems*, 25(12):2226–2239, 2014.

Carbune, V., Gonnet, P., Deselaers, T., Rowley, H. A., Daryin, A., Calvo, M., Wang, L.-L., Keysers, D., Feuz, S., and Gervais, P. Fast multi-language lstm-based online handwriting recognition. *International Journal on Document Analysis and Recognition (IJDAR)*, 23(2):89–102, 2020.

Carlsson, C. and Fullér, R. On possibilistic mean value and variance of fuzzy numbers. *Fuzzy sets and systems*, 122(2): 315–326, 2001.

Cavalcante, R. C., Brasileiro, R. C., Souza, V. L., Nobrega, J. P., and Oliveira, A. L. Computational intelligence and financial markets: A survey and future directions. *Expert Systems with Applications*, 55:194–211, 2016.

Chakraborti, A., Toke, I. M., Patriarca, M., and Abergel, F. Econophysics review: I. empirical facts. *Quantitative Finance*, 11 (7):991–1012, 2011.

Chaudhari, P., Agrawal, H., and Kotecha, K. Data augmentation using mg-gan for improved cancer classification on gene expression data. *Soft Computing*, 24:11381–11391, 2020.

Chen, H., Xiao, K., Sun, J., and Wu, S. A double-layer neural network framework for high-frequency forecasting. *ACM Transactions on Management Information Systems (TMIS)*, 7(4):1–17, 2017.

Chen, Q., Liang, B., and Wang, J. A comparative study of lstm and phased lstm for gait prediction. *Int J Artificial Intelli & App*, 10(4):57–66, 2019.

Chevyrev, I. and Kormilitzin, A. A primer on the signature method in machine learning. *arXiv preprint arXiv:1603.03788*, 2016.

Clarke, R. G., Krase, S., and Statman, M. Tracking errors, regret, and tactical asset allocation. *Journal of Portfolio Management*, 20(3):16, 1994.

Coelho, J., D'almeida, D., Coyne, S., Gilkerson, N., Mills, K., and Madiraju, P. Social media and forecasting stock price change. In *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, volume 2, pages 195–200. IEEE, 2019.

Cootner, P. *The Random Character of Stock Market Prices*. Risk classics library. M.I.T. Press, 1964. ISBN 9781899332847. URL `https://books.google.com.mt/books?id=lcfzAAAAMAAJ`.

REFERENCES

Cowles 3rd, A. and Jones, H. E. Some a posteriori probabilities in stock market action. *Econometrica, Journal of the Econometric Society*, pages 280–294, 1937.

Cross, J. G. et al. A theory of adaptive economic behavior. *Cambridge Books*, 2008.

Dau, H. A., Bagnall, A., Kamgar, K., Yeh, C.-C. M., Zhu, Y., Gharghabi, S., Ratanamahatana, C. A., and Keogh, E. The ucr time series archive. *IEEE/CAA Journal of Automatica Sinica*, 6(6):1293–1305, 2019.

De Bondt, W. P. Betting on trends: Intuitive forecasts of financial risk and return. *International Journal of forecasting*, 9(3): 355–371, 1993.

de Meer Pardo, F. *Enriching financial datasets with generative adversarial networks*. PhD thesis, Master's thesis, Delft University of Technology, the Netherlands, 2019.

de Meer Pardo, F., Schwendner, P., and Wunsch, M. Tackling the exponential scaling of signature-based generative adversarial networks for high-dimensional financial time-series generation. *The Journal of Financial Data Science*, 4(4): 110–132, 2022.

Debnath, A., Waghmare, G., Wadhwa, H., Asthana, S., and Arora, A. Exploring generative data augmentation in multivariate time series forecasting: Opportunities and challenges. *Solar-Energy*, 137:52–560, 2021.

Ding, X., Wang, Y., Wang, Z. J., and Welch, W. J. Efficient subsampling of realistic images from gans conditional on a class or a continuous variable. *Neurocomputing*, 2022.

Donahue, C., McAuley, J., and Puckette, M. Adversarial audio synthesis. *arXiv preprint arXiv:1802.04208*, 2018.

Duan, Y., Goodell, J. W., Li, H., and Li, X. Assessing machine learning for forecasting economic risk: Evidence from an expanded chinese financial information set. *Finance Research Letters*, 46:102273, 2022.

Ducoffe, M., Haloui, I., Gupta, J. S., and Supaero, I. Anomaly detection on time series with wasserstein gan applied to phm. *PHM Applications of Deep Learning and Emerging Analytics. International Journal of Prognostics and Health Management Reviewed (Special Issue)*, 2019.

Dumoulin, V., Shlens, J., and Kudlur, M. A learned representation for artistic style. *arXiv preprint arXiv:1610.07629*, 2016.

Eckerli, F. Generative adversarial networks in finance: an overview. *Available at SSRN 3864965*, 2021.

Efimov, D., Xu, D., Kong, L., Nefedov, A., and Anandakrishnan, A. Using generative adversarial networks to synthesize artificial financial datasets. *arXiv preprint arXiv:2002.02271*, 2020.

Eltoft, T. Data augmentation using a combination of independent component analysis and non-linear time-series prediction. In *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No. 02CH37290)*, volume 1, pages 448–453. IEEE, 2002.

Esteban, C., Hyland, S. L., and Rätsch, G. Real-valued (medical) time series generation with recurrent conditional gans. *arXiv preprint arXiv:1706.02633*, 2017.

Fama, E. F. The behavior of stock-market prices. *The journal of Business*, 38(1):34–105, 1965.

Fawaz, H. I. Deep learning for time series classification. *arXiv preprint arXiv:2010.00567*, 2020.

Fawaz, H. I., Forestier, G., Weber, J., Idoumghar, L., and Muller, P.-A. Data augmentation using synthetic data for time series classification with deep residual networks. *arXiv preprint arXiv:1808.02455*, 2018.

Feng, F., He, X., Wang, X., Luo, C., Liu, Y., and Chua, T.-S. Temporal relational ranking for stock prediction. *ACM Transactions on Information Systems (TOIS)*, 37(2):1–30, 2019.

Fernando, B., Fromont, E., Muselet, D., and Sebban, M. Supervised learning of gaussian mixture models for visual vocabulary generation. *Pattern Recognition*, 45(2):897–907, 2012.

Fields, T., Hsieh, G., and Chenou, J. Mitigating drift in time series data with noise augmentation. In *2019 International*

Conference on Computational Science and Computational Intelligence (CSCI), pages 227–230. IEEE, 2019.

Fischer, T. and Krauss, C. Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, 270(2):654–669, 2018.

Fischhoff, B. A little leaning: Confidence in multi-cue judgment tasks. *Attention and performance*, 8:122–138, 1980.

Flanegin, F. R. and Rudd, D. P. Should investments professors join the crowd. *Managerial Finance*, 2005.

Fons, E., Dawson, P., Zeng, X.-j., Keane, J., and Iosifidis, A. Evaluating data augmentation for financial time series classification. *arXiv preprint arXiv:2010.15111*, 2020.

Forestier, G., Petitjean, F., Dau, H. A., Webb, G. I., and Keogh, E. Generating synthetic time series to augment sparse datasets. In *2017 IEEE international conference on data mining (ICDM)*, pages 865–870. IEEE, 2017.

Franco-Pedroso, J., Gonzalez-Rodriguez, J., Cubero, J., Planas, M., Cobo, R., and Pablos, F. Generating virtual scenarios of multivariate financial data for quantitative trading applications. *The Journal of Financial Data Science*, 1(2):55–77, 2019.

Fu, B., Kirchbuchner, F., and Kuijper, A. Data augmentation for time series: traditional vs generative models on capacitive proximity time series. In *Proceedings of the 13th ACM International Conference on PErvasive Technologies Related to Assistive Environments*, pages 1–10, 2020.

Fu, R., Chen, J., Zeng, S., Zhuang, Y., and Sudjianto, A. Time series simulation by conditional generative adversarial net. *arXiv preprint arXiv:1904.11419*, 2019.

Furman, J. and Seamans, R. Ai and the economy. *Innovation policy and the economy*, 19(1):161–191, 2019.

Garliauskas, A. Neural network chaos and computational algorithms of forecast in finance. In *IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No. 99CH37028)*, volume 2, pages 638–643. IEEE, 1999.

Gervais, S. and Odean, T. Learning to be overconfident. *The review of financial studies*, 14(1):1–27, 2001.

Göçken, M., Özçalıcı, M., Boru, A., and Dosdoğru, A. T. Integrating metaheuristics and artificial neural networks for improved stock price prediction. *Expert Systems with Applications*, 44:320–331, 2016.

Goldberg, D. E. Gas in search, optimization and machine learning. *Kluwe r Academic Publishers, Boston, MA*, pages 1–40, 1989.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K., editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL `https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf`.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

Grietzer, P. A theory of vibe. *Glass-Bead, https://www. glass-bead. org/article/a-theory-of-vibe*, 2017.

Grilli, L. and Santoro, D. Generative adversarial network for market hourly discrimination. 2020.

Grossman, S. J. and Stiglitz, J. E. On the impossibility of informationally efficient markets. *The American economic review*, 70(3):393–408, 1980.

Gu, C.-S., Hsieh, H.-P., Wu, C.-S., Chang, R.-I., and Ho, J.-M. A fund selection robo-advisor with deep-learning driven market prediction. In *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, pages 2845–2850. IEEE, 2019.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. Improved training of wasserstein gans. *Advances in neural information processing systems*, 30, 2017.

Hartmann, K. G., Schirrmeister, R. T., and Ball, T. Eeg-gan: Generative adversarial networks for electroencephalograhic (eeg) brain signals. *arXiv preprint arXiv:1806.01875*, 2018.

Hasibi, R., Shokri, M., and Dehghan, M. Augmentation scheme for dealing with imbalanced network traffic classification using deep learning. *arXiv preprint arXiv:1901.00204*, 2019.

Hassan, M. R. A combination of hidden markov model and fuzzy model for stock market forecasting. *Neurocomputing*, 72 (16-18):3439–3446, 2009.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

Hiemstra, Y. Modeling structured nonlinear knowledge to predict stock market returns. *Chaos & Nonlinear Dynamics in the Financial Markets: Theory, Evidence and Applications, Irwin, Chicago, IL*, pages 163–175, 1995.

Hinton, G. E. and Roweis, S. Stochastic neighbor embedding. *Advances in neural information processing systems*, 15, 2002.

Hinton, G. E. and Sejnowski, T. J. Analyzing cooperative computation. In *Proceedings of the fifth annual conference of the cognitive science society*, pages 2554–2558, 1983.

Hinton, G. E. and Zemel, R. Autoencoders, minimum description length and helmholtz free energy. *Advances in neural information processing systems*, 6, 1993.

Hinton, G. E., Osindero, S., and Teh, Y.-W. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7): 1527–1554, 2006.

Hogenboom. Generation of synthetic financial time series with generative - adversarial networks. 2020.

Holland, J. H. *Adaptation in natural and artificial systems*. University of Michigan Press, Michigan, USA, 1975.

Hotelling, H. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24 (6):417, 1933.

Hu, H., Tang, L., Zhang, S., and Wang, H. Predicting the direction of stock markets using optimized neural networks with google trends. *Neurocomputing*, 285:188–195, 2018.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

Huang, Q., Kong, Z., Li, Y., Yang, J., and Li, X. Discovery of trading points based on bayesian modeling of trading rules. *World Wide Web*, 21(6):1473–1490, 2018.

Huberman, G. and Regev, T. Contagious speculation and a cure for cancer: A nonevent that made stock prices soar. *The Journal of Finance*, 56(1):387–396, 2001.

Hyland, S. L., Esteban, C., and Rätsch, G. Real-valued (medical) time series generation with recurrent conditional gans. *stat*, 1050(8), 2017.

Iwana, B. K. and Uchida, S. Time series classification using local distance-based features in multi-modal fusion networks. *Pattern Recognition*, 97:107024, 2020.

Iwana, B. K. and Uchida, S. An empirical survey of data augmentation for time series classification with neural networks. *Plos one*, 16(7):e0254841, 2021a.

Iwana, B. K. and Uchida, S. Time series data augmentation for neural networks by time warping with a discriminative teacher. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 3558–3565. IEEE, 2021b.

Jabbar, A., Li, X., and Omar, B. A survey on generative adversarial networks: Variants, applications, and training. *ACM*

Computing Surveys (CSUR), 54(8):1–49, 2021.

Jackwerth, J. C. and Rubinstein, M. Recovering probability distributions from option prices. *The journal of Finance*, 51(5): 1611–1631, 1996.

Jaitly, N. and Hinton, G. E. Vocal tract length perturbation (vtlp) improves speech recognition. In *Proc. ICML Workshop on Deep Learning for Audio, Speech and Language*, volume 117, page 21, 2013.

Javeri, I. Y., Toutiaee, M., Arpinar, I. B., Miller, T. W., and Miller, J. A. Improving neural networks for time series forecasting using data augmentation and automl. *arXiv preprint arXiv:2103.01992*, 2021.

Jensen, M. H., Johansen, A., and Simonsen, I. Inverse statistics in economics: the gain–loss asymmetry. *Physica A: Statistical Mechanics and its Applications*, 324(1-2):338–343, 2003.

Jolicoeur-Martineau, A. The relativistic discriminator: a key element missing from standard gan. *arXiv preprint arXiv:1807.00734*, 2018.

Jordon, J., Yoon, J., and Van Der Schaar, M. Pate-gan: Generating synthetic data with differential privacy guarantees. In *International conference on learning representations*, 2018.

Jørgensen, R. K. and Igel, C. Machine learning for financial transaction classification across companies using character-level word embeddings of text fields. *Intelligent Systems in Accounting, Finance and Management*, 28(3):159–172, 2021.

Jullum, M., Løland, A., Huseby, R. B., Ånonsen, G., and Lorentzen, J. Detecting money laundering transactions with machine learning. *Journal of Money Laundering Control*, 23(1):173–186, 2020.

Kamble, R. A. Short and long term stock trend prediction using decision tree. In *2017 International Conference on Intelligent Computing and Control Systems (ICICCS)*, pages 1371–1375. IEEE, 2017.

Kamycki, K., Kapuscinski, T., and Oszust, M. Data augmentation with suboptimal warping for time-series classification. *Sensors*, 20(1):98, 2019.

Karim, M. and Rahman, R. M. Decision tree and naive bayes algorithm for classification and generation of actionable knowledge for direct marketing. 2013.

Karras, T., Aila, T., Laine, S., and Lehtinen, J. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

Kim, B.-H. and Pyun, J.-Y. Ecg identification for personal authentication using lstm-based deep recurrent neural networks. *Sensors*, 20(11):3069, 2020.

Kim, J., Won, C., and Bae, J. K. A knowledge integration model for the prediction of corporate dividends. *Expert Systems with Applications*, 37(2):1344–1350, 2010.

Kim, J. W., Weistroffer, H. R., and Redmond, R. T. Expert systems for bond rating: a comparative analysis of statistical, rule-based and neural network systems. *Expert systems*, 10(3):167–172, 1993.

Kim, K.-j. Financial time series forecasting using support vector machines. *Neurocomputing*, 55(1-2):307–319, 2003.

Kim, K.-j. Toward global optimization of case-based reasoning systems for financial forecasting. *Applied intelligence*, 21(3): 239–249, 2004.

Kim, K.-j. and Han, I. Genetic algorithms approach to feature discretization in artificial neural networks for the prediction of stock price index. *Expert systems with Applications*, 19(2):125–132, 2000.

Kim, Y., Ahn, W., Oh, K. J., and Enke, D. An intelligent hybrid trading system for discovering trading rules for the futures market using rough sets and genetic algorithms. *Applied Soft Computing*, 55:127–140, 2017. ISSN 1568-4946. doi: https://doi.org/10.1016/j.asoc.2017.02.006.

Kimoto, T., Asakawa, K., Yoda, M., and Takeoka, M. Stock market prediction system with modular neural networks. In *1990 IJCNN international joint conference on neural networks*, pages 1–6. IEEE, 1990.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Kluwak, K. and Niżyński, T. Gait classification using lstm networks for tagging system. In *2020 IEEE 15th International Conference of System of Systems Engineering (SoSE)*, pages 295–300. IEEE, 2020.

Koochali, A., Dengel, A., and Ahmed, S. If you like it, gan it. probabilistic multivariate times series forecast with gan. *arXiv preprint arXiv:2005.01181*, 2020.

Krauss, C., Do, X. A., and Huck, N. Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the s&p 500. *European Journal of Operational Research*, 259(2):689–702, 2017.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Kumar, M. and Thenmozhi, M. Forecasting stock index returns using arima-svm, arima-ann, and arima-random forest hybrid models. *International Journal of Banking, Accounting and Finance*, 5(3):284–308, 2014.

Labiad, B., Berrado, A., and Benabbou, L. Machine learning techniques for short term stock movements classification for moroccan stock exchange. In *2016 11th International Conference on Intelligent Systems: Theories and Applications (SITA)*, pages 1–6. IEEE, 2016.

Labiad, B., Benabbou, L., and Berrado, A. Improving stock market intraday prediction by generative adversarial neural networks. 2021.

Lahmiri, S. A predictive system integrating intrinsic mode functions, artificial neural networks, and genetic algorithms for forecasting s&p500 intra-day data. *Intelligent Systems in Accounting, Finance and Management*, 27(2):55–65, 2020.

Laibson, D. Golden eggs and hyperbolic discounting. *The Quarterly Journal of Economics*, 112(2):443–478, 1997.

Lakshminarayanan, M., John, N. S., Channegowda, J., Raj, A., and Naaz, F. Devising high fidelity synthetic data using generative adversarial networks for energy storage systems. In *2021 IEEE Mysore Sub Section International Conference (MysuruCon)*, pages 202–205. IEEE, 2021.

Larson, A. B. Measurement of a random process in futures prices. In *Proceedings of the Annual Meeting (Western Farm Economics Association)*, volume 33, pages 101–112. JSTOR, 1960.

Lattner, S., Grachten, M., and Widmer, G. Imposing higher-level structure in polyphonic music generation using convolutional restricted boltzmann machines and constraints. *Journal of Creative Music Systems*, 2:1–31, 2018.

Lawrence, R. Using neural networks to forecast stock market prices. *University of Manitoba*, 333:2006–2013, 1997.

Le Guennec, A., Malinowski, S., and Tavenard, R. Data augmentation for time series classification using convolutional neural networks. In *ECML/PKDD workshop on advanced analytics and learning on temporal data*, 2016.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Lezmi, E., Roche, J., Roncalli, T., and Xu, J. Improving the robustness of trading strategy backtesting with boltzmann machines and generative adversarial networks. *Available at SSRN 3645473*, 2020.

Leznik, M., Michalsky, P., Willis, P., Schanzel, B., Östberg, P.-O., and Domaschka, J. Multivariate time series synthesis using generative adversarial networks. In *Proceedings of the ACM/SPEC International Conference on Performance Engineering*, pages 43–50, 2021.

Li, J., Wang, X., Lin, Y., Sinha, A., and Wellman, M. Generating realistic stock market order streams. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 727–734, 2020.

Li, W. and Mei, F. Asset returns in deep learning methods: An empirical analysis on sse 50 and csi 300. *Research in International Business and Finance*, 54:101291, 2020.

Li, X., Ngu, A. H. H., and Metsis, V. Tts-cgan: A transformer time-series conditional gan for biosignal data augmentation. *arXiv preprint arXiv:2206.13676*, 2022.

Li, X., Yang, L., Xue, F., and Zhou, H. Time series prediction of stock price using deep belief networks with intrinsic plasticity. In *2017 29th Chinese Control And Decision Conference (CCDC)*, pages 1237–1242. IEEE, 2017.

Li, Y., Jiang, W., Yang, L., and Wu, T. On neural networks and learning systems for business computing. *Neurocomputing*, 275:1150–1159, 2018.

Li, Y., Swersky, K., and Zemel, R. Generative moment matching networks. In *International conference on machine learning*, pages 1718–1727. PMLR, 2015.

Liang, Q., Rong, W., Zhang, J., Liu, J., and Xiong, Z. Restricted boltzmann machine based stock market trend prediction. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 1380–1387. IEEE, 2017.

Lichtenstein, S., Fischhoff, B., and Phillips, L. Calibration of probabilities: The state of the art to 1980. d. kahneman, p. slovic, a. tversky, eds. judgment under uncertainty: Heuristics and biases. 1982.

Lim, B. and Zohren, S. Time series forecasting with deep learning: A survey. *arXiv preprint arXiv:2004.13408*, 2020.

Liu, B., Zhang, Z., and Cui, R. Efficient time series augmentation methods. In *2020 13th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, pages 1004–1009. IEEE, 2020.

Liu, C., Ventre, C., and Polukarov, M. Synthetic data augmentation for deep reinforcement learning in financial trading. In *3rd ACM International Conference on AI in Finance*, pages 343–351, 2022.

Liu, G. Data quality problems troubling business and financial researchers: A literature review and synthetic analysis. *Journal of Business & Finance Librarianship*, 25(3-4):315–371, 2020.

Liu, J., Zhang, J., Ding, Y., Xu, X., Jiang, M., and Shi, Y. Pbgan: Partial binarization of deconvolution based generators. *arXiv preprint arXiv:1802.09153*, 2018a.

Liu, Q., Wang, C., Zhang, P., and Zheng, K. Detecting stock market manipulation via machine learning: Evidence from china securities regulatory commission punishment cases. *International Review of Financial Analysis*, 78:101887, 2021.

Liu, Y., Gopikrishnan, P., Stanley, H. E., et al. Statistical properties of the volatility of price fluctuations. *Physical review e*, 60(2):1390, 1999.

Liu, Y., Qin, Z., Wan, T., and Luo, Z. Auto-painter: Cartoon image generation from sketch by using conditional wasserstein generative adversarial networks. *Neurocomputing*, 311:78–87, 2018b.

Lo, A. W. and MacKinlay, A. C. Stock market prices do not follow random walks: Evidence from a simple specification test. *The review of financial studies*, 1(1):41–66, 1988.

Lo, A. W. and MacKinlay, A. C. Stock Market Prices Do Not Follow Random Walks: Evidence from a Simple Specification Test. *The Review of Financial Studies*, 1(1):41–66, 04 2015. ISSN 0893-9454. doi: 10.1093/rfs/1.1.41.

Lo, A. W., Mamaysky, H., and Wang, J. Foundations of technical analysis: Computational algorithms, statistical inference, and empirical implementation. *The journal of finance*, 55(4):1705–1765, 2000.

Luo, S., Lin, X., and Zheng, Z. A novel cnn-ddpg based ai-trader: Performance and roles in business operations. *Transportation Research Part E: Logistics and Transportation Review*, 131:68–79, 2019.

Ma, J., Wang, L., Zhang, Y.-R., Yuan, W., and Guo, W. An integrated latent dirichlet allocation and word2vec method for generating the topic evolution of mental models from global to local. *Expert Systems with Applications*, 212:118695, 2023.

Ma, T., Chen, J., and Xiao, C. Constrained generation of semantically valid graphs via regularizing variational autoencoders. *Advances in Neural Information Processing Systems*, 31, 2018.

Maddala, G. S. *Introduction to economics*. Macmillan, 1992.

Majhi, R., Panda, G., Sahoo, G., Panda, A., and Choubey, A. Prediction of s&p 500 and djia stock indices using particle swarm optimization technique. In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pages 1276–1282. IEEE, 2008.

Makhmudkhujaev, F. and Park, I. K. Generative adversarial networks with attention mechanisms at every scale. *IEEE Access*, 9:168404–168414, 2021.

Malkiel, B. G. *A random walk down Wall Street: the time-tested strategy for successful investing*. WW Norton & Company, 2019.

Mao, X., Li, Q., Xie, H., Lau, R. Y., Wang, Z., and Paul Smolley, S. Least squares generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2794–2802, 2017.

McCluskey, J. and Liu, J. Us financial market forecasting using data classification with features from global markets. In *2017 2nd International Conference on Image, Vision and Computing (ICIVC)*, pages 965–969. IEEE, 2017.

Metz, L., Poole, B., Pfau, D., and Sohl-Dickstein, J. Unrolled generative adversarial networks. *arXiv preprint arXiv:1611.02163*, 2016.

Milana, C. and Ashta, A. Artificial intelligence techniques in finance and financial markets: A survey of the literature. *Strategic Change*, 30(3):189–209, 2021.

Mingyue, Q., Cheng, L., and Yu, S. Application of the artifical neural network in predicting the direction of stock market index. In *2016 10th International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS)*, pages 219–223. IEEE, 2016.

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

Mogren, O. C-rnn-gan: Continuous recurrent neural networks with adversarial training. *arXiv preprint arXiv:1611.09904*, 2016.

Mudassir, M., Bennbaia, S., Unal, D., and Hammoudeh, M. Time-series forecasting of bitcoin prices using high-dimensional features: a machine learning approach. *Neural Computing and Applications*, pages 1–15, 2020.

Müller, U. A., Dacorogna, M. M., Davé, R. D., Olsen, R. B., Pictet, O. V., and Von Weizsäcker, J. E. Volatilities of different time resolutions—analyzing the dynamics of market components. *Journal of Empirical Finance*, 4(2-3):213–239, 1997.

Napate, S., Thakur, M., and B-School, D. Algorithmic trading and strategies. 11 2020.

Nazário, R. T. F., e Silva, J. L., Sobreiro, V. A., and Kimura, H. A literature review of technical analysis on stock markets. *The Quarterly Review of Economics and Finance*, 66:115–126, 2017.

Ni, H., Szpruch, L., Wiese, M., Liao, S., and Xiao, B. Conditional sig-wasserstein gans for time series generation. *arXiv preprint arXiv:2006.05421*, 2020.

Ni, H., Szpruch, L., Sabate-Vidales, M., Xiao, B., Wiese, M., and Liao, S. Sig-wasserstein gans for time series generation. In *Proceedings of the Second ACM International Conference on AI in Finance*, pages 1–8, 2021.

Niederhoffer, V. and Osborne, M. F. M. Market making and reversal on the stock exchange. *Journal of the American Statistical Association*, 61(316):897–916, 1966.

Nikolenko, S. I. *Synthetic data for deep learning*, volume 174. Springer, 2021.

Nikolopoulos, C. and Fellrath, P. A hybrid expert system for investment advising. *Expert Systems*, 11(4):245–250, 1994.

Oh, C., Han, S., and Jeong, J. Time-series data augmentation based on interpolation. *Procedia Computer Science*, 175: 64–71, 2020.

Ohashi, H., Al-Nasser, M., Ahmed, S., Akiyama, T., Sato, T., Nguyen, P., Nakamura, K., and Dengel, A. Augmenting wearable sensor data with physical constraint for dnn-based human-action recognition. In *ICML 2017 times series workshop*, pages 6–11, 2017.

Oord, A. v. d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

Osborne, M. F. Periodic structure in the brownian motion of stock prices. *Operations Research*, 10(3):345–379, 1962.

Pan, Q., Li, X., and Fang, L. Data augmentation for deep learning-based ecg analysis. In *Feature engineering and computational intelligence in ECG monitoring*, pages 91–111. Springer, 2020.

Pan, Z., Yu, W., Yi, X., Khan, A., Yuan, F., and Zheng, Y. Recent progress on generative adversarial networks (gans): A survey. *IEEE access*, 7:36322–36333, 2019.

Park, C.-H. and Irwin, S. H. The profitability of technical analysis: A review. 2004.

Park, C.-H. and Irwin, S. H. What do we know about the profitability of technical analysis? *Journal of Economic surveys*, 21 (4):786–826, 2007.

Patel, J., Shah, S., Thakkar, P., and Kotecha, K. Predicting stock and stock price index movement using trend deterministic data preparation and machine learning techniques. *Expert Systems with Applications*, 42(1):259–268, 2015. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2014.07.040.

Patuwo, E., Hu, M. Y., and Hung, M. S. Two-group classification using neural networks. *Decision Sciences*, 24(4):825–845, 1993.

Pau, L.-F. et al. Artificial intelligence in economics and management. In *International Workshop on Artificial Intelligence in Economics and Management (1985: Zurich, Switzerland)*. Sole distributors for the USA and Canada, Elsevier Science Pub. Co., 1986.

Pau, L.-F. and Tan, P. Y. Artificial intelligence in economics and finance: A state of the art-1994: The real estate price and assets and liability analysis case. *Handbook of computational economics*, 1:405–439, 1996.

Phua, P. K. H., Ming, D., and Lin, W. Neural network with genetic algorithms for stocks prediction. In *Fifth Conference of the Association of Asian-Pacific Operations Research Societies*, 2000.

Qiu, M. and Song, Y. Predicting the direction of stock market index movement using an optimized artificial neural network model. *PloS one*, 11(5):e0155133, 2016.

Qiu, T., Zheng, B., Ren, F., and Trimper, S. Return-volatility correlation in financial dynamics. *Physical Review E*, 73(6): 065103, 2006.

Quah, T.-S. and Srinivasan, B. Neural network for stock selection. In *APPLIED INFORMATICS-PROCEEDINGS-*, pages 29–31, 1999.

Radford, A., Metz, L., and Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Rashid, K. M. and Louis, J. Window-warping: a time series data augmentation of imu data for construction equipment activity identification. In *ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction*, volume 36, pages 651–657. IAARC Publications, 2019.

Refenes, A.-P., Zapranis, A., and Francis, G. Modeling stock returns in the framework of apt: a comparative study with regression models. In *Neural networks in the capital markets*, volume 7, pages 101–126. John Wiley & Sons Chichester, 1995.

Remlinger, C., Mikael, J., and Elie, R. Conditional loss and deep euler scheme for time series generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8098–8105, 2022.

Rode, D., Parikh, S., Friedman, Y., and Kane, J. An evolutionary approach to technical trading and capital market efficiency. *The Wharton School University of Pennsylvania*, 1, 1995.

Rosenblatt, F. *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning representations by back-propagating errors. *nature*, 323(6088): 533–536, 1986.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

Sakoe, H. and Chiba, S. Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing*, 26(1):43–49, 1978.

Salimans, T. and Kingma, D. P. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *Advances in neural information processing systems*, 29, 2016.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. Improved techniques for training gans. *Advances in neural information processing systems*, 29, 2016.

Santoro, D. and Grilli, L. Generative adversarial network to evaluate quantity of information in financial markets. *Neural Computing and Applications*, pages 1–18, 2022.

Saxe, A. M., McClelland, J. L., and Ganguli, S. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

Schmidhuber, J. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

Schumann, M. and Lohrbach, T. Comparing artificial neural networks with statistical methods within the field of stock market prediction. In *[1993] Proceedings of the Twenty-sixth Hawaii International Conference on System Sciences*, volume 4, pages 597–606. IEEE, 1993.

Schwartz, R. A. and Whitcomb, D. K. The time-variance relationship: Evidence on autocorrelation in common stock returns. *The Journal of Finance*, 32(1):41–55, 1977.

Shap, K. Artificial intelligence in financial trading, 1987.

Shiller, R. J. Irrational exuberance. In *Irrational exuberance*. Princeton university press, 2015.

Shorten, C. and Khoshgoftaar, T. M. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1): 1–48, 2019.

Siegelmann, H. T. Theoretical foundations of recurrent neural networks. Technical report, Rutgers University, 1993.

Simonetto, L. Generating spiking time series with generative adversarial networks: an application on banking transactions. 2018.

Simonovsky, M. and Komodakis, N. Graphvae: Towards generation of small graphs using variational autoencoders. In *Artificial Neural Networks and Machine Learning–ICANN 2018: 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018, Proceedings, Part I 27*, pages 412–422. Springer, 2018.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Smith, K. E. and Smith, A. O. Conditional gan for timeseries generation. *arXiv preprint arXiv:2006.16477*, 2020.

Snow, D. Mtss-gan: Multivariate time series simulation generative adversarial networks. *Available at SSRN*, 2020.

Sønderby, C. K., Caballero, J., Theis, L., Shi, W., and Huszár, F. Amortised map inference for image super-resolution. *arXiv preprint arXiv:1610.04490*, 2016.

Sornette, D. Why stock markets crash. In *Why Stock Markets Crash*. Princeton university press, 2009.

Steiner, M. and Wittkemper, H.-G. Neural networks as an alternative stock market model. 1995.

Steven Eyobu, O. and Han, D. S. Feature representation and data augmentation for human activity classification based on wearable imu sensor data using a deep lstm neural network. *Sensors*, 18(9):2892, 2018.

Strong, R. A behavioral investigation of three paradigms in finance. *Unpublished manuscript, University of Maine, College of Business Administration*, 1987.

Subramanian, V., Hung, M. S., and Hu, M. Y. An experimental evaluation of neural networks for classification. *Computers & operations research*, 20(7):769–782, 1993.

Such, F. P., Rawal, A., Lehman, J., Stanley, K., and Clune, J. Generative teaching networks: Accelerating neural architecture search by learning to generate synthetic training data. In *International Conference on Machine Learning*, pages 9206–9216. PMLR, 2020.

Sun, H., Deng, Z., Chen, H., and Parkes, D. C. Decision-aware conditional gans for time series data. *arXiv preprint arXiv:2009.12682*, 2020.

Sun, L., Su, T., Liu, C., and Wang, R. Deep lstm networks for online chinese handwriting recognition. In *2016 15th international conference on frontiers in handwriting recognition (icfhr)*, pages 271–276. IEEE, 2016.

Susskind, J. M., Hinton, G. E., Movellan, J. R., and Anderson, A. K. Generating facial expressions with deep belief nets. *Affective computing, emotion modelling, synthesis and recognition*, 2008(5):421–440, 2008.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

Taddy, M. The technological elements of artificial intelligence. In *The economics of artificial intelligence: An agenda*, pages 61–87. University of Chicago Press, 2018.

Takagi, H. and Pallez, D. Paired comparison-based interactive differential evolution. In *2009 World Congress on Nature & Biologically Inspired Computing (NaBIC)*, pages 475–480. IEEE, 2009.

Takahashi, S., Chen, Y., and Tanaka-Ishii, K. Modeling financial time-series with generative adversarial networks. *Physica A: Statistical Mechanics and its Applications*, 527:121261, 2019.

Tay, F. E. and Cao, L. Application of support vector machines in financial time series forecasting. *omega*, 29(4):309–317, 2001a.

Tay, F. E. H. and Cao, L. J. A comparative study of saliency analysis and genetic algorithm for feature selection in support vector machines. *Intelligent Data Analysis*, 5(3):191–209, 2001b.

Teng, X., Wang, T., Zhang, X., Lan, L., and Luo, Z. Enhancing stock price trend prediction via a time-sensitive data augmentation method. *Complexity*, 2020, 2020.

Thakur, A. *Approaching (almost) any machine learning problem*. Abhishek Thakur, 2020.

Thavaneswaran, A., Thiagarajah, K., and Appadoo, S. S. Fuzzy coefficient volatility (fcv) models with applications. *Mathematical and Computer Modelling*, 45(7-8):777–786, 2007.

Thavaneswaran, A., Appadoo, S. S., and Paseka, A. Weighted possibilistic moments of fuzzy numbers with applications to garch modeling and option pricing. *Mathematical and Computer Modelling*, 49(1-2):352–368, 2009.

Theis, L., Oord, A. v. d., and Bethge, M. A note on the evaluation of generative models. *arXiv preprint arXiv:1511.01844*, 2015.

Tieleman, T. and Hinton, G. Lecture 6.5-rmsprop. *COURSERA: Neural networks for machine learning*, 2012.

Tiwari, R., Srivastava, S., and Gera, R. Investigation of artificial intelligence techniques in finance and marketing. *Procedia Computer Science*, 173:149–157, 2020.

Tizziano, A. Direct reinforcement learning for the dva hedging through recurrent generative adversarial networks for dataset augmentation. 2018.

Tokic, D. Blackrock robo-advisor 4.0: When artificial intelligence replaces human discretion. *Strategic Change*, 27(4): 285–290, 2018.

Torralba, A., Fergus, R., and Freeman, W. T. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 30(11):1958–1970, 2008.

Trippi, R. R. and Turban, E. *Neural networks in finance and investing: Using artificial intelligence to improve real world performance*. McGraw-Hill, Inc., 1992.

Tsaih, R., Hsu, Y., and Lai, C. C. Forecasting s&p 500 stock index futures with a hybrid ai system. *Decision support systems*, 23(2):161–174, 1998.

Tseng, H.-Y., Jiang, L., Liu, C., Yang, M.-H., and Yang, W. Regularizing generative adversarial networks under limited data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7921–7931, 2021.

Um, T. T., Pfister, F. M., Pichler, D., Endo, S., Lang, M., Hirche, S., Fietzek, U., and Kulić, D. Data augmentation of wearable sensor data for parkinsons disease monitoring using convolutional neural networks. In *Proceedings of the 19th ACM international conference on multimodal interaction*, pages 216–220, 2017.

Van der Maaten, L. and Hinton, G. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

Van Eyden, R. The application of neural networks in the forecasting of share prices (finance and technology publishing, haymarket, va). 1996.

Van Liebergen, B. et al. Machine learning: A revolution in risk management and compliance? *Journal of Financial Transformation*, 45:60–67, 2017.

van Rhijn, J. Generating asset paths for financial sdes with gans. 2020.

Vapnik, V. N. An overview of statistical learning theory. *IEEE transactions on neural networks*, 10(5):988–999, 1999.

Violante, A. An introduction to t-sne with python example, Aug 2018. URL `https://medium.com/@violante.andre/an-introduction-to-t-sne-with-python-example-47e6ae7dc58f`.

Wang, J.-J., Wang, J.-Z., Zhang, Z.-G., and Guo, S.-P. Stock index forecasting based on a hybrid model. *Omega*, 40(6): 758–766, 2012.

Wang, Y. and Yan, G. Survey on the application of deep learning in algorithmic trading. *Data Science in Finance and Economics*, 1(4):345–361, 2021.

Wang, Z., Healy, G., Smeaton, A. F., and Ward, T. E. Use of neural signals to evaluate the quality of generative adversarial network performance in facial image generation. *Cognitive Computation*, 12(1):13–24, 2020a.

Wang, Z., She, Q., Smeaton, A. F., Ward, T. E., and Healy, G. Synthetic-neuroscore: Using a neuro-ai interface for evaluating generative adversarial networks. *Neurocomputing*, 405:26–36, 2020b.

Wang, Z., She, Q., and Ward, T. E. Generative adversarial networks in computer vision: A survey and taxonomy. *ACM Computing Surveys (CSUR)*, 54(2):1–38, 2021.

Wang, Z., Yan, W., and Oates, T. Time series classification from scratch with deep neural networks: A strong baseline. In *2017 International joint conference on neural networks (IJCNN)*, pages 1578–1585. IEEE, 2017.

Wen, Q., Sun, L., Song, X., Gao, J., Wang, X., and Xu, H. Time series data augmentation for deep learning: A survey. *arXiv preprint arXiv:2002.12478*, 2020.

White, H. Economic prediction using neural networks: The case of ibm daily stock returns. In *ICNN*, volume 2, pages 451–458, 1988.

White, T. Sampling generative networks. *arXiv preprint arXiv:1609.04468*, 2016.

Wiatrak, M., Albrecht, S. V., and Nystrom, A. Stabilizing generative adversarial networks: A survey. *arXiv preprint arXiv:1910.00927*, 2019.

Wiese, M., Knobloch, R., Korn, R., and Kretschmer, P. Quant gans: deep generation of financial time series. *Quantitative Finance*, 20(9):1419–1440, 2020.

Xu, G., Xing, G., Jiang, J., Jiang, J., and Ke, Y. Arrhythmia detection using gated recurrent unit network with ecg signals. *Journal of Medical Imaging and Health Informatics*, 10(3):750–757, 2020.

Xu, Q., Huang, G., Yuan, Y., Guo, C., Sun, Y., Wu, F., and Weinberger, K. An empirical study on evaluation metrics of generative adversarial networks. *arXiv preprint arXiv:1806.07755*, 2018.

Yang, H., Chan, L., and King, I. Support vector machine regression for volatile stock market prediction. In *International conference on intelligent data engineering and automated learning*, pages 391–396. Springer, 2002.

Yoo, P. D., Kim, M. H., and Jan, T. Machine learning techniques and use of event information for stock market prediction: A survey and evaluation. In *International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06)*, volume 2, pages 835–841. IEEE, 2005.

Yoon, J., Jarrett, D., and Van der Schaar, M. Time-series generative adversarial networks. 2019.

Yoon, Y. and Swales, G. Predicting stock price performance: A neural network approach. In *Proceedings of the twenty-fourth annual Hawaii international conference on system sciences*, volume 4, pages 156–162. IEEE, 1991.

Yoon, Y., Swales Jr, G., and Margavio, T. M. A comparison of discriminant analysis versus artificial neural networks. *Journal of the Operational Research Society*, 44(1):51–60, 1993.

YU, S.-L. and Li, Z. Stock price prediction based on arima-rnn combined model. *DEStech Transactions on Social Science, Education and Human Science*, 03 2018. doi: 10.12783/dtssehs/icss2017/19384.

Zhang, D. and Khoreva, A. Progressive augmentation of gans. *arXiv preprint arXiv:1901.10422*, 2019.

Zhang, H., Goodfellow, I., Metaxas, D., and Odena, A. Self-attention generative adversarial networks. In *International conference on machine learning*, pages 7354–7363. PMLR, 2019a.

Zhang, H., Zhang, Z., Odena, A., and Lee, H. Consistency regularization for generative adversarial networks. *arXiv preprint arXiv:1910.12027*, 2019b.

Zhang, J., Cui, S., Xu, Y., Li, Q., and Li, T. A novel data-driven stock price trend prediction system. *Expert Systems with Applications*, 97:60–69, 2018.

Zhang, N., Lin, A., and Shang, P. Multidimensional k-nearest neighbor model based on eemd for financial time series forecasting. *Physica A: Statistical Mechanics and its Applications*, 477:161–173, 2017.

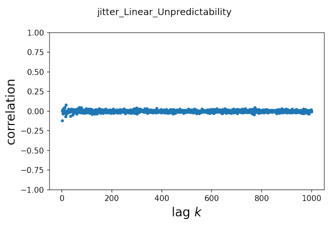
Zhong, X. and Enke, D. Forecasting daily stock market return using dimensionality reduction. *Expert Systems with Applications*, 67:126–139, 2017.

Zhou, W., Chen, J., and Huang, Y. Co-citation analysis and burst detection on financial bubbles with scientometrics approach. *Economic research-Ekonomska istraživanja*, 32(1):2310–2328, 2019.

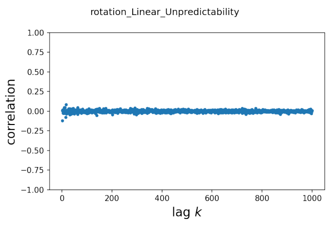
Zhou, X., Pan, Z., Hu, G., Tang, S., and Zhao, C. Stock market prediction on high-frequency data using generative adversarial nets. *Mathematical Problems in Engineering*, 2018, 2018.

Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

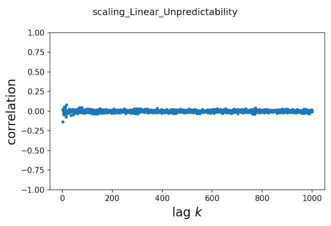
A Non-GAN Stylised Facts



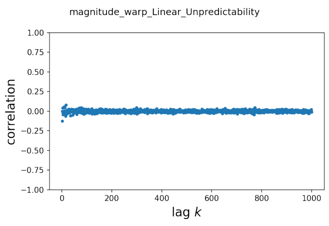
(a) Jitter



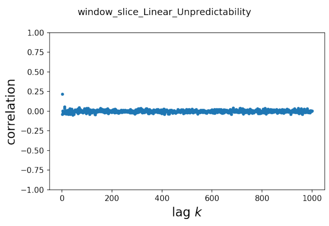
(b) Rotation



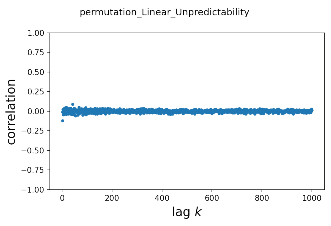
(c) Scaling



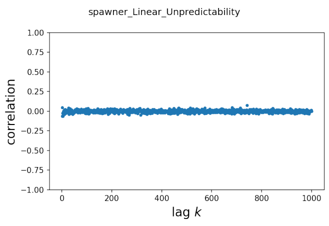
(d) Magnitude Warp



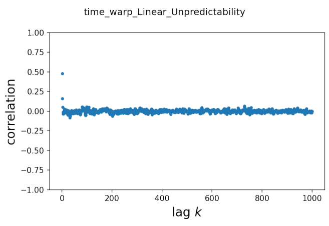
(e) widow Slice



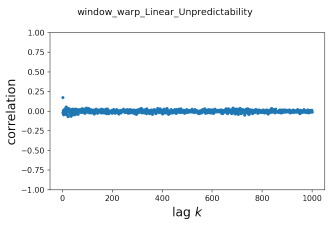
(f) Permutation



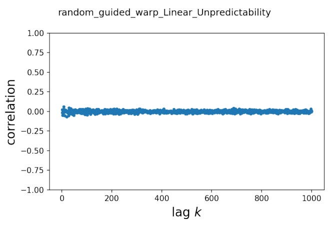
(g) Spawner



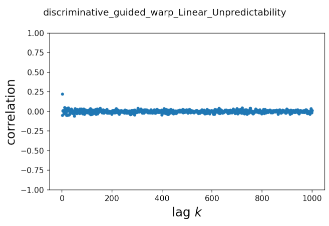
(h) Time Warp



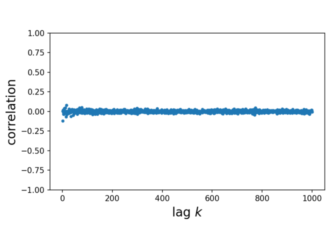
(i) Window Warp



(j) Random Guided Warp

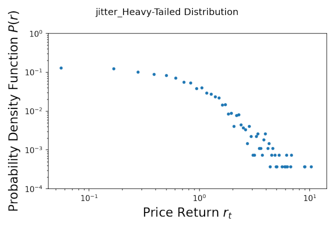


(k) Discrim. Guided Warp

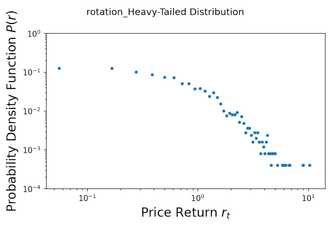


(l) Real SP500

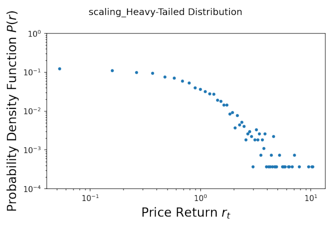
Figure A.1: Linear Unpredictability



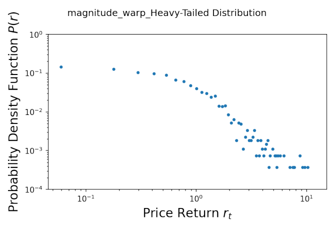
(a) Jitter



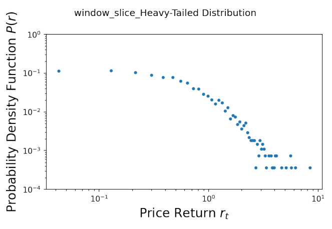
(b) Rotation



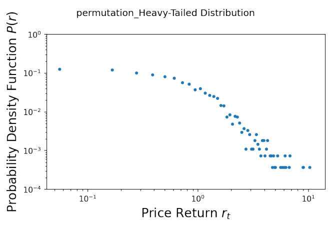
(c) Scaling



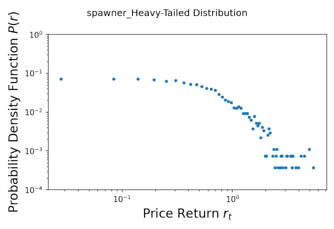
(d) Magnitude Warp



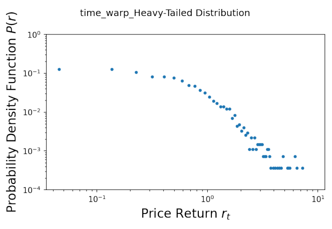
(e) Winow Slice



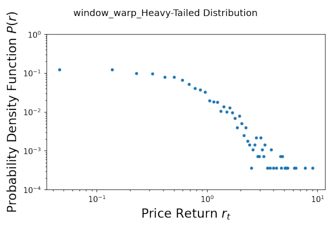
(f) Permutation



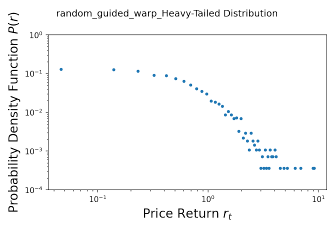
(g) Spawner



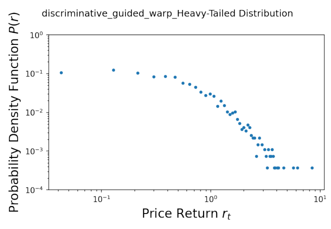
(h) Time Warp



(i) Window Warp

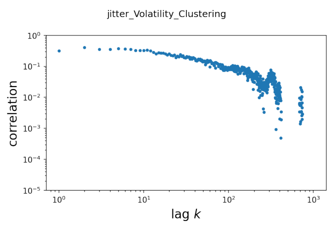


(j) Random Guided Warp

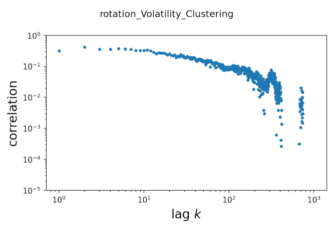


(k) Discrim. Guided Warp

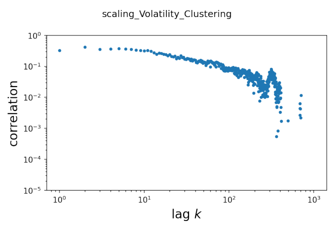
Figure A.2: Heavy-Tailed Distribution



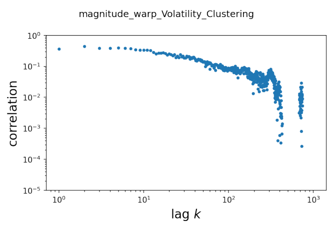
(a) Jitter



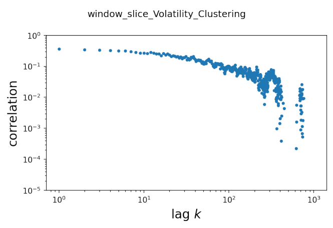
(b) Rotation



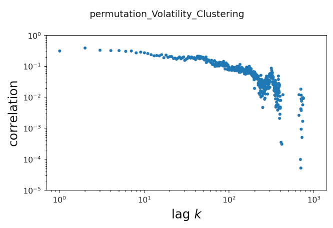
(c) Scaling



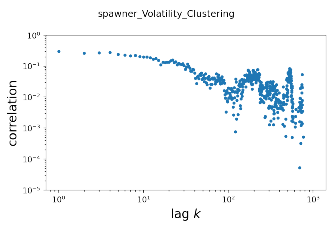
(d) Magnitude Warp



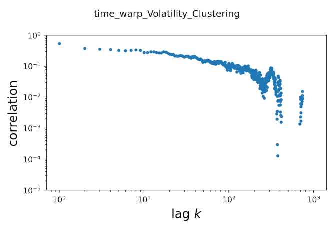
(e) Winow Slice



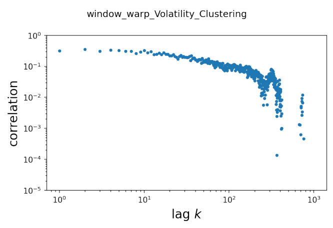
(f) Permutation



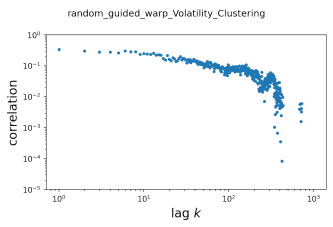
(g) Spawner



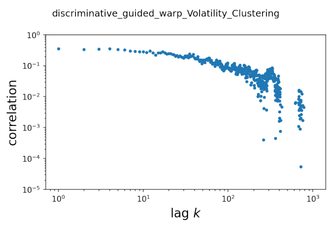
(h) Time Warp



(i) Window Warp

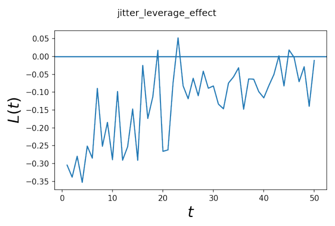


(j) Random Guided Warp

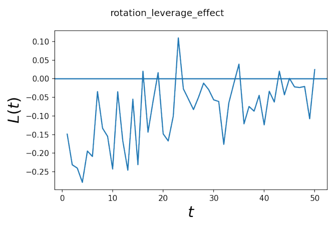


(k) Discrim. Guided Warp

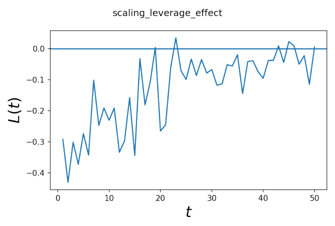
Figure A.3: Volatility Clustering



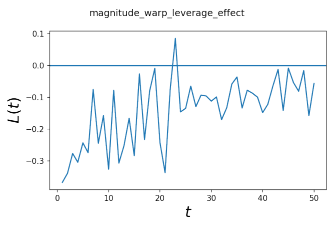
(a) Jitter



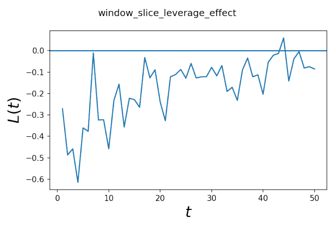
(b) Rotation



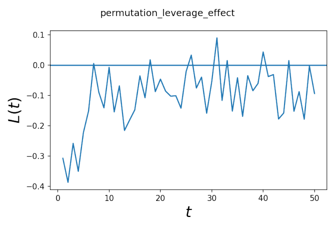
(c) Scaling



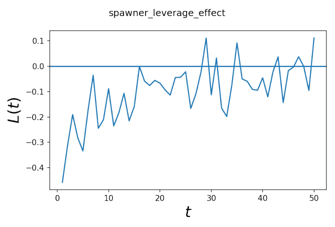
(d) Magnitude Warp



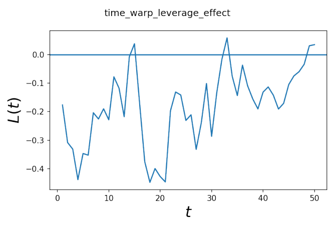
(e) Winow Slice



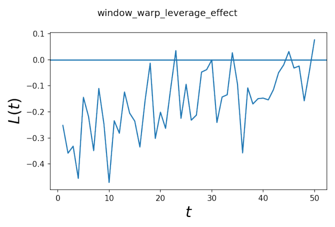
(f) Permutation



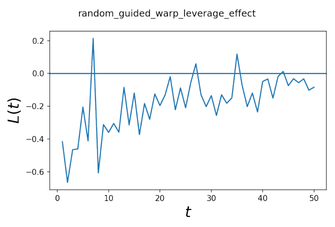
(g) Spawner



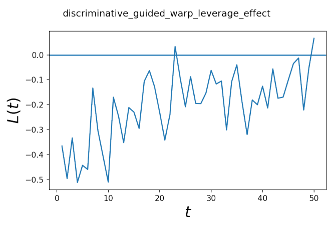
(h) Time Warp



(i) Window Warp

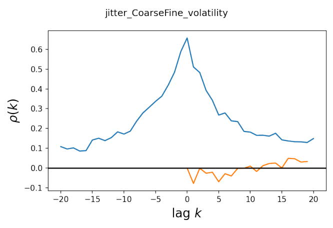


(j) Random Guided Warp

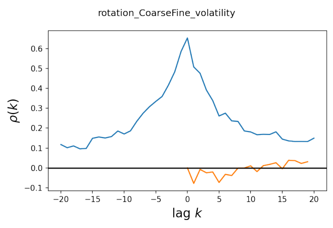


(k) Discrim. Guided Warp

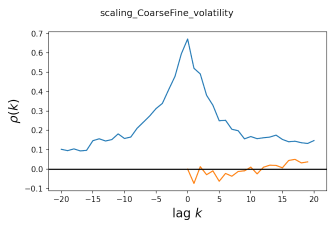
Figure A.4: Leverage Effect



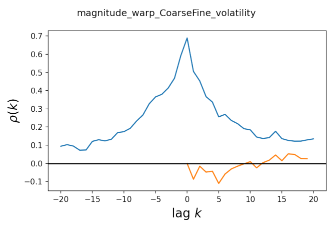
(a) Jitter



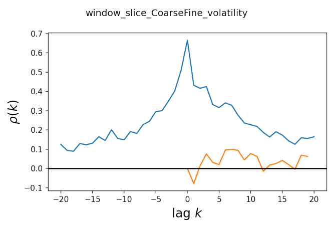
(b) Rotation



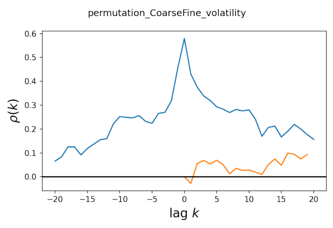
(c) Scaling



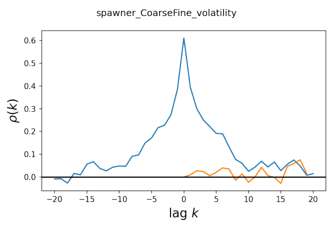
(d) Magnitude Warp



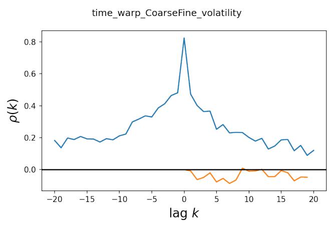
(e) Winow Slice



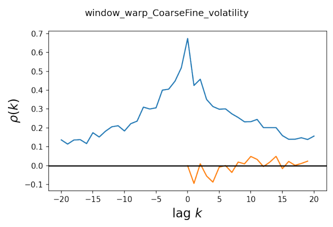
(f) Permutation



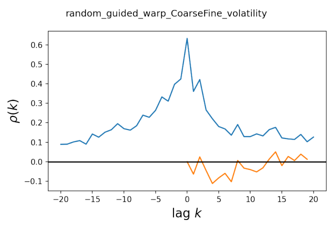
(g) Spawner



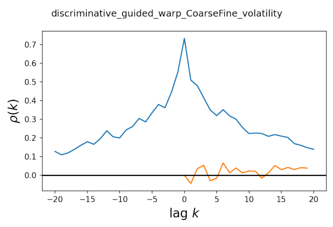
(h) Time Warp



(i) Window Warp

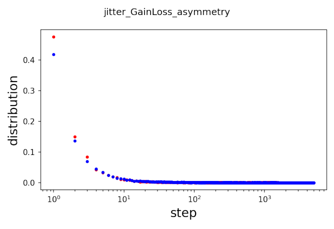


(j) Random Guided Warp

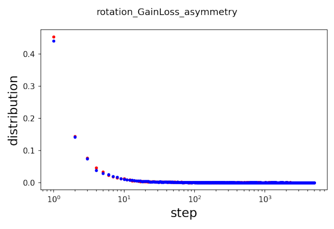


(k) Discrim. Guided Warp

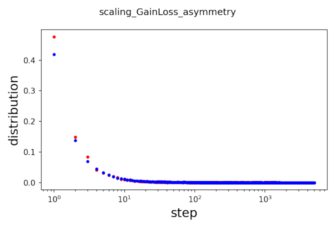
Figure A.5: Coarse-Fine Volatility



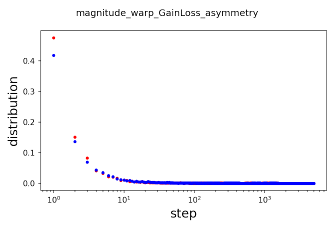
(a) Jitter



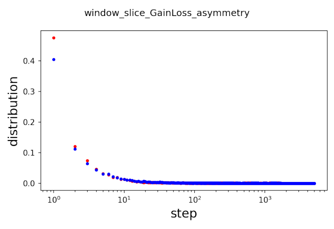
(b) Rotation



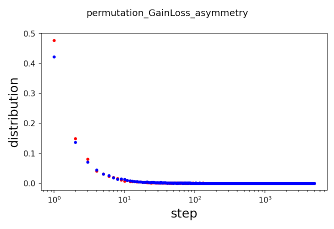
(c) Scaling



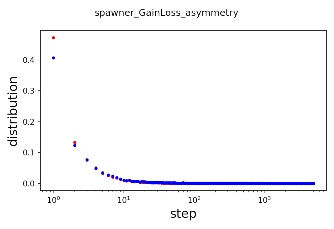
(d) Magnitude Warp



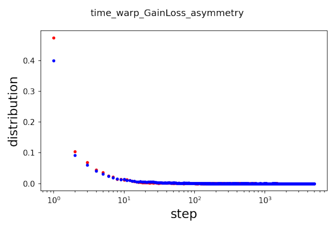
(e) Winow Slice



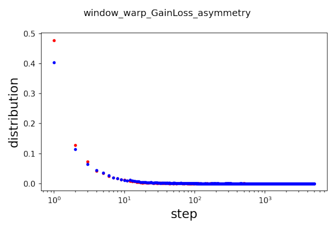
(f) Permutation



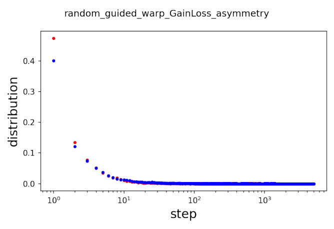
(g) Spawner



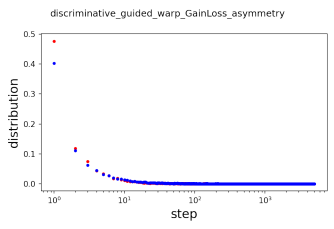
(h) Time Warp



(i) Window Warp

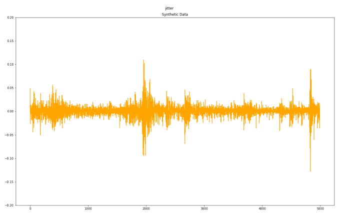


(j) Random Guided Warp

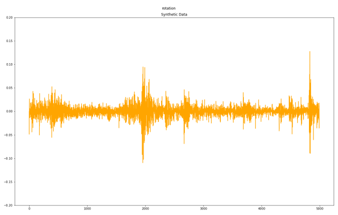


(k) Discrim. Guided Warp

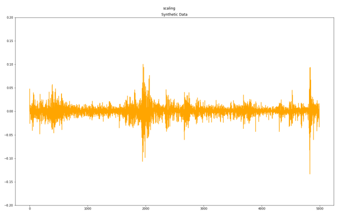
Figure A.6: Gain-Loss Asymmetry



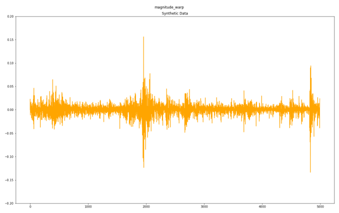
(a) Jitter



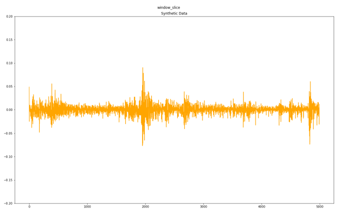
(b) Rotation



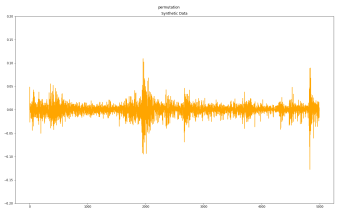
(c) Scaling



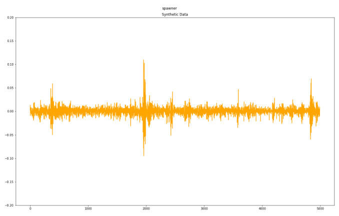
(d) Magnitude Warp



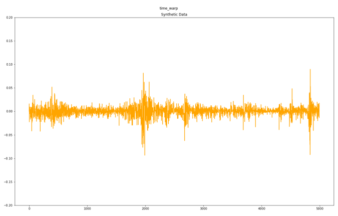
(e) Winow Slice



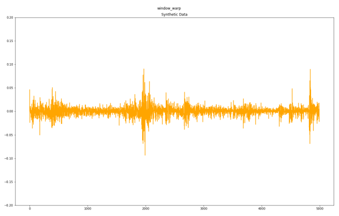
(f) Permutation



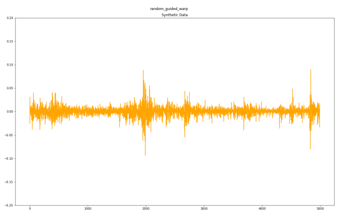
(g) Spawner



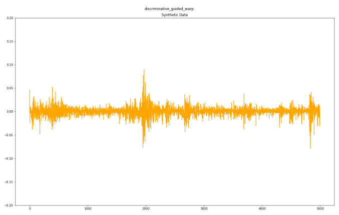
(h) Time Warp



(i) Window Warp

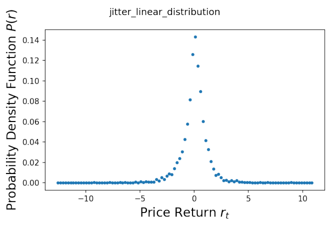


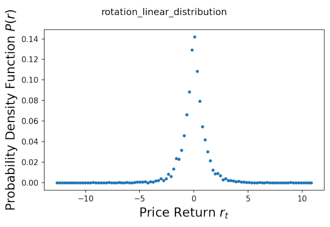
(j) Random Guided Warp

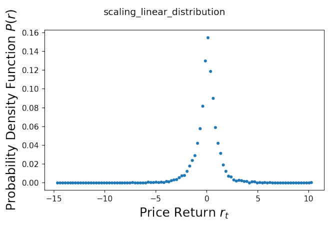


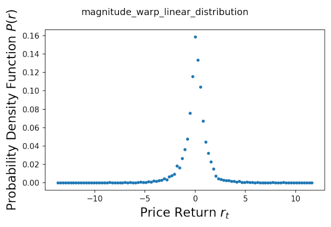
(k) Discrim. Guided Warp

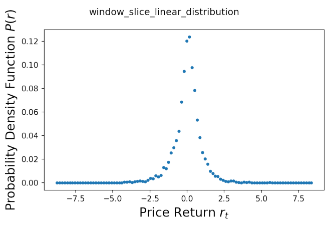
Figure A.7: Log Returns

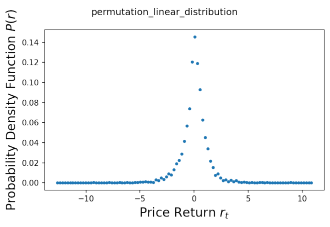

(a) Jitter

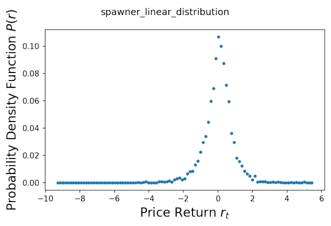

(b) Rotation

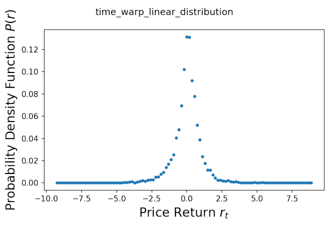

(c) Scaling

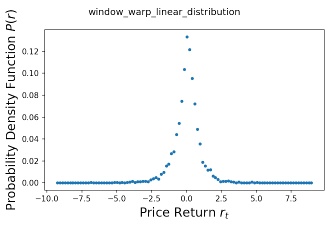

(d) Magnitude Warp

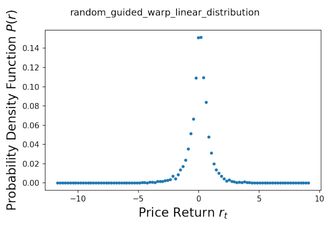

(e) Winow Slice


(f) Permutation


(g) Spawner


(h) Time Warp


(i) Window Warp


(j) Random Guided Warp

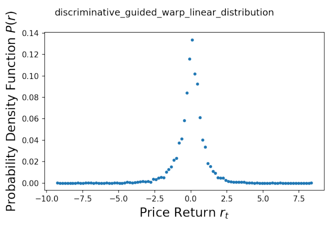
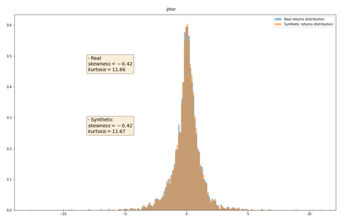
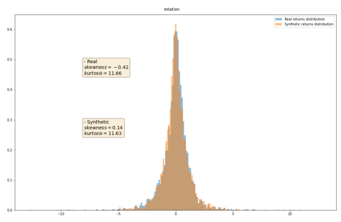

(k) Discrim. Guided Warp

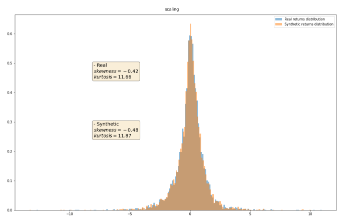
Figure A.8: Linear Distribution



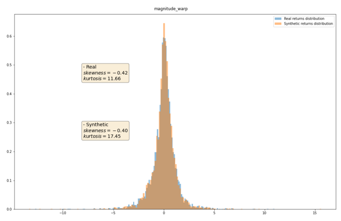
(a) Jitter



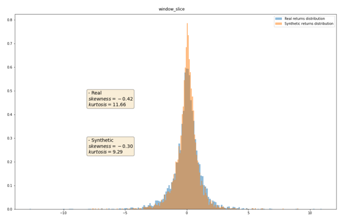
(b) Rotation



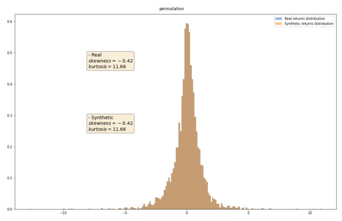
(c) Scaling



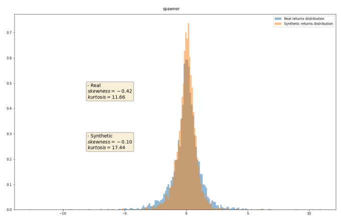
(d) Magnitude Warp



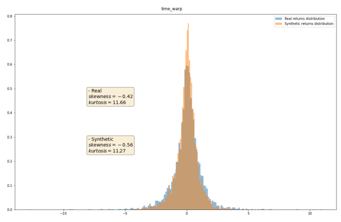
(e) Winow Slice



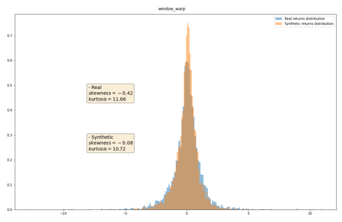
(f) Permutation



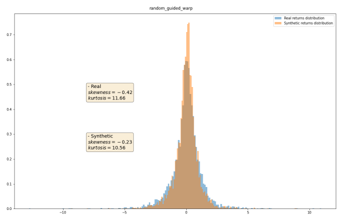
(g) Spawner



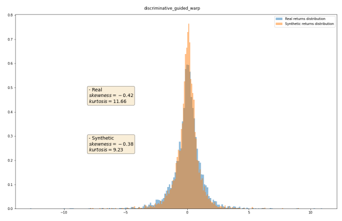
(h) Time Warp



(i) Window Warp

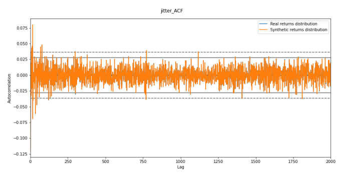


(j) Random Guided Warp

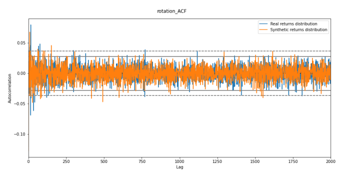


(k) Discrim. Guided Warp

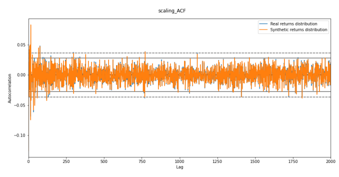
Figure A.9: Probability Density Function, Skewness and Kurtosis



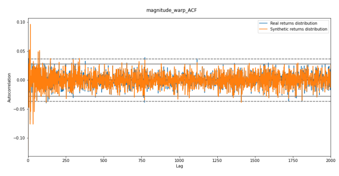
(a) Jitter



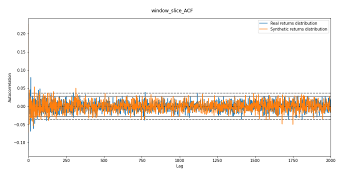
(b) Rotation



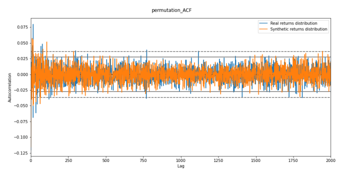
(c) Scaling



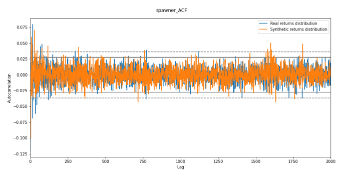
(d) Magnitude Warp



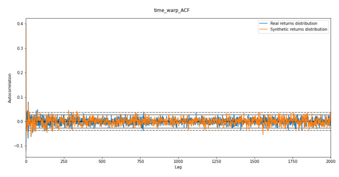
(e) Winow Slice



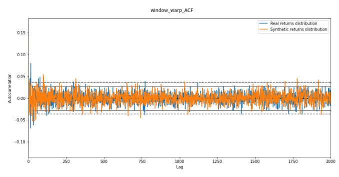
(f) Permutation



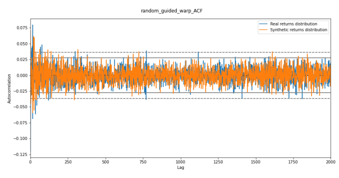
(g) Spawner



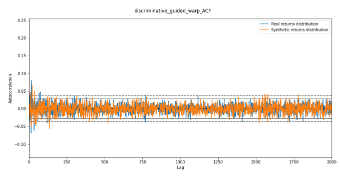
(h) Time Warp



(i) Window Warp



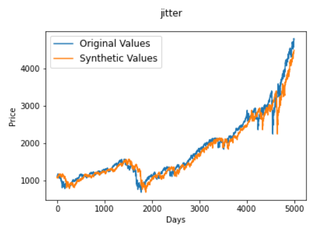
(j) Random Guided Warp



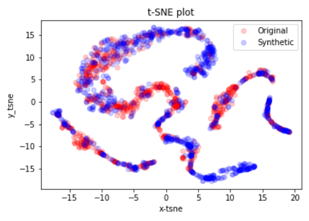
(k) Discrim. Guided Warp

Figure A.10: Autocorrelation Function

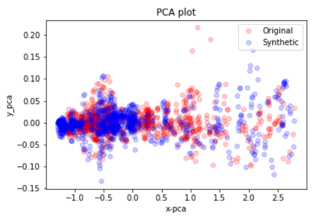
B Preliminary non-GAN Plots



(a) Time Series

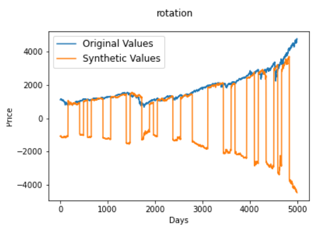


(b) t-SNE Plot

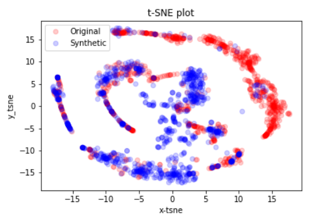


(c) PCA Plot

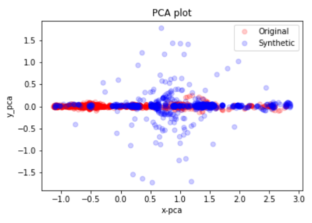
Figure B.1: Jitter



(a) Time Series

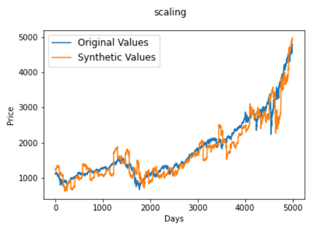


(b) t-SNE Plot

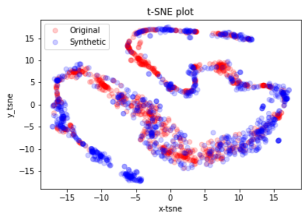


(c) PCA Plot

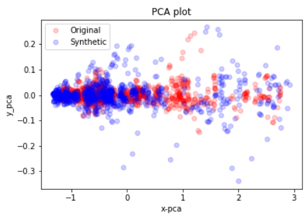
Figure B.2: Rotation



(a) Time Series

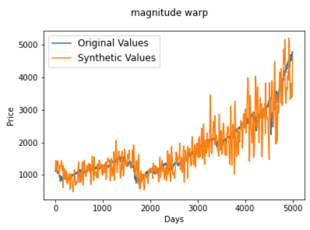


(b) t-SNE Plot

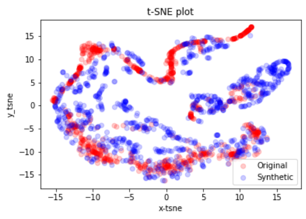


(c) PCA Plot

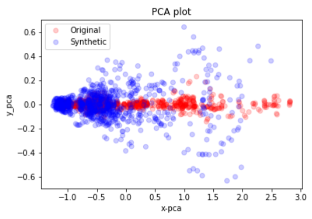
Figure B.3: Scaling



(a) Time Series

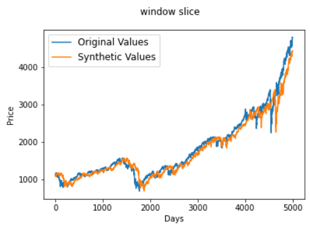


(b) t-SNE Plot

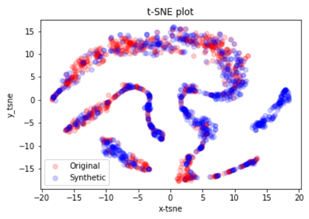


(c) PCA Plot

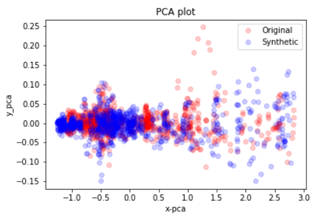
Figure B.4: Magnitude Warp



(a) Time Series

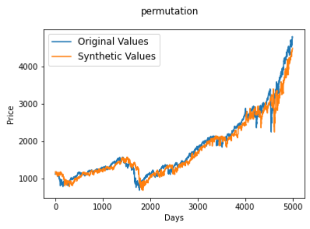


(b) t-SNE Plot

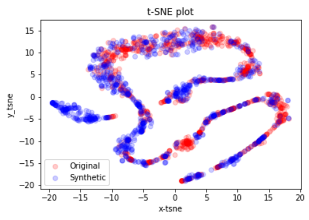


(c) PCA Plot

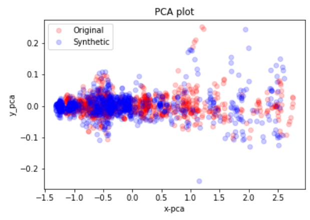
Figure B.5: Window Slice



(a) Time Series

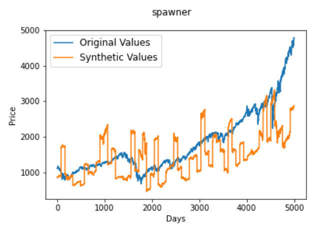


(b) t-SNE Plot

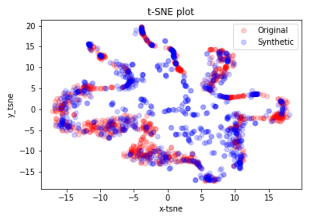


(c) PCA Plot

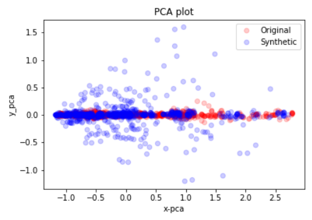
Figure B.6: Permutation



(a) Time Series

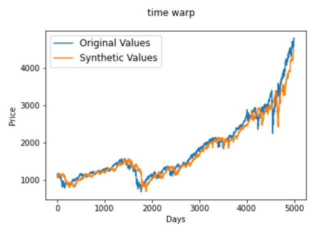


(b) t-SNE Plot

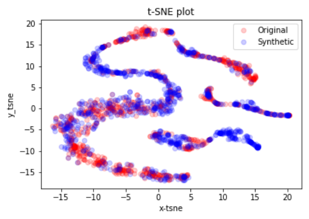


(c) t-SNE Plot

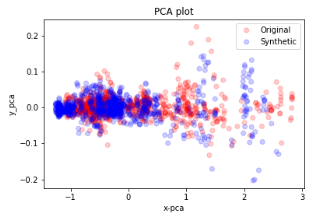
Figure B.7: Spawner



(a) Time Series

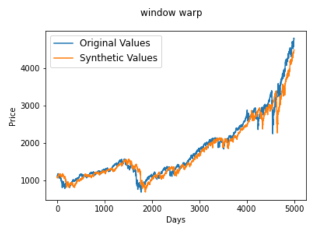


(b) t-SNE Plot

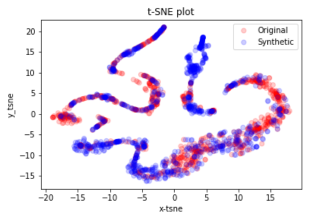


(c) PCA Plot

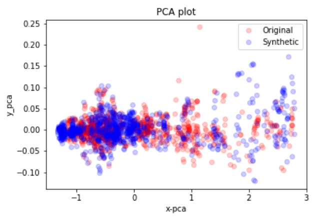
Figure B.8: Time Warp



(a) Time Series

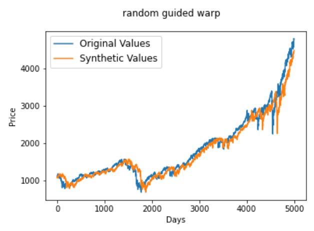


(b) t-SNE Plot

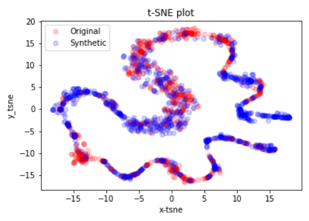


(c) PCA Plot

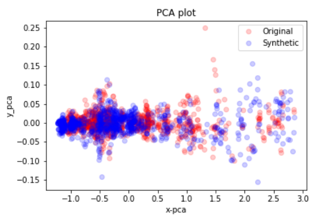
Figure B.9: Window Warp



(a) Time Series

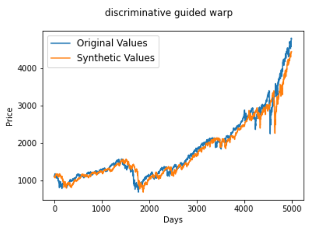


(b) t-SNE Plot

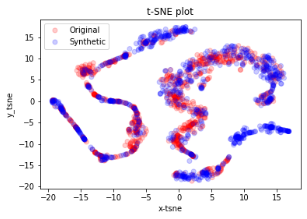


(c) PCA Plot

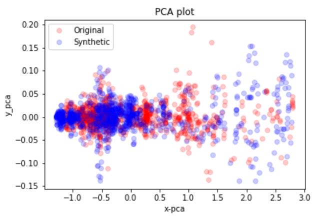
Figure B.10: Random Guided Warp



(a) Time Series



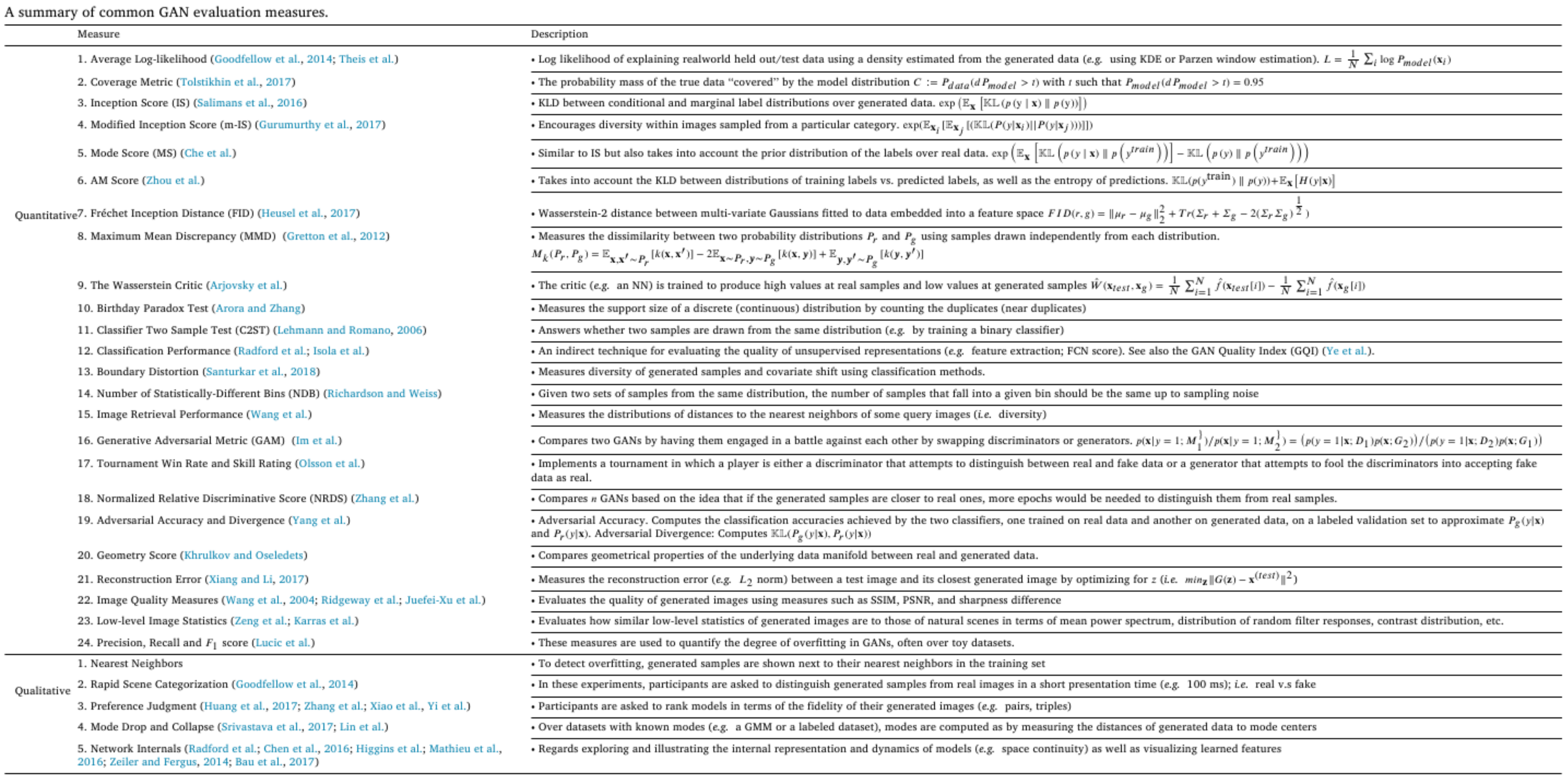
(b) t-SNE Plot



(c) PCA Plot

Figure B.11: Discrim. Guided Warp

C GAN Evaluation Methods



A summary of common GAN evaluation measures.

	Measure	Description
	1. Average Log-likelihood (Goodfellow et al., 2014; Theis et al.)	• Log likelihood of explaining realworld held out/test data using a density estimated from the generated data (*e.g.* using KDE or Parzen window estimation). $L = \frac{1}{N}\sum_i \log P_{model}(\mathbf{x}_i)$
	2. Coverage Metric (Tolstikhin et al., 2017)	• The probability mass of the true data "covered" by the model distribution $C := P_{data}(dP_{model} > t)$ with t such that $P_{model}(dP_{model} > t) = 0.95$
	3. Inception Score (IS) (Salimans et al., 2016)	• KLD between conditional and marginal label distributions over generated data. $\exp\left(\mathbb{E}_{\mathbf{x}}\left[\mathbb{KL}\left(p\left(y \mid \mathbf{x}\right) \parallel p\left(y\right)\right)\right]\right)$
	4. Modified Inception Score (m-IS) (Gurumurthy et al., 2017)	• Encourages diversity within images sampled from a particular category. $\exp(\mathbb{E}_{\mathbf{x}_i}[\mathbb{E}_{\mathbf{x}_j}[(\mathbb{KL}(P(y\|\mathbf{x}_i)\|\|P(y\|\mathbf{x}_j)))]])$
	5. Mode Score (MS) (Che et al.)	• Similar to IS but also takes into account the prior distribution of the labels over real data. $\exp\left(\mathbb{E}_{\mathbf{x}}\left[\mathbb{KL}\left(p\left(y \mid \mathbf{x}\right) \parallel p\left(y^{train}\right)\right)\right] - \mathbb{KL}\left(p\left(y\right) \parallel p\left(y^{train}\right)\right)\right)$
	6. AM Score (Zhou et al.)	• Takes into account the KLD between distributions of training labels vs. predicted labels, as well as the entropy of predictions. $\mathbb{KL}(p(y^{\text{train}}) \parallel p(y)) + \mathbb{E}_{\mathbf{x}}\left[H(y\|\mathbf{x})\right]$
Quantitative	7. Fréchet Inception Distance (FID) (Heusel et al., 2017)	• Wasserstein-2 distance between multi-variate Gaussians fitted to data embedded into a feature space $FID(r,g) = \\|\mu_r - \mu_g\\|_2^2 + Tr(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{\frac{1}{2}})$
	8. Maximum Mean Discrepancy (MMD) (Gretton et al., 2012)	• Measures the dissimilarity between two probability distributions P_r and P_g using samples drawn independently from each distribution. $M_k(P_r, P_g) = \mathbb{E}_{\mathbf{x},\mathbf{x}' \sim P_r}[k(\mathbf{x},\mathbf{x}')] - 2\mathbb{E}_{\mathbf{x}\sim P_r, \mathbf{y}\sim P_g}[k(\mathbf{x},\mathbf{y})] + \mathbb{E}_{\mathbf{y},\mathbf{y}' \sim P_g}[k(\mathbf{y},\mathbf{y}')]$
	9. The Wasserstein Critic (Arjovsky et al.)	• The critic (*e.g.* an NN) is trained to produce high values at real samples and low values at generated samples $\hat{W}(\mathbf{x}_{test}, \mathbf{x}_g) = \frac{1}{N}\sum_{i=1}^{N}\hat{f}(\mathbf{x}_{test}[i]) - \frac{1}{N}\sum_{i=1}^{N}\hat{f}(\mathbf{x}_g[i])$
	10. Birthday Paradox Test (Arora and Zhang)	• Measures the support size of a discrete (continuous) distribution by counting the duplicates (near duplicates)
	11. Classifier Two Sample Test (C2ST) (Lehmann and Romano, 2006)	• Answers whether two samples are drawn from the same distribution (*e.g.* by training a binary classifier)
	12. Classification Performance (Radford et al.; Isola et al.)	• An indirect technique for evaluating the quality of unsupervised representations (*e.g.* feature extraction; FCN score). See also the GAN Quality Index (GQI) (Ye et al.).
	13. Boundary Distortion (Santurkar et al., 2018)	• Measures diversity of generated samples and covariate shift using classification methods.
	14. Number of Statistically-Different Bins (NDB) (Richardson and Weiss)	• Given two sets of samples from the same distribution, the number of samples that fall into a given bin should be the same up to sampling noise
	15. Image Retrieval Performance (Wang et al.)	• Measures the distributions of distances to the nearest neighbors of some query images (*i.e.* diversity)
	16. Generative Adversarial Metric (GAM) (Im et al.)	• Compares two GANs by having them engaged in a battle against each other by swapping discriminators or generators. $p(\mathbf{x}\|y=1; M_1^{'})/p(\mathbf{x}\|y=1; M_2^{'}) = \left(p(y=1\|\mathbf{x}; D_1)p(\mathbf{x}; G_2)\right)/\left(p(y=1\|\mathbf{x}; D_2)p(\mathbf{x}; G_1)\right)$
	17. Tournament Win Rate and Skill Rating (Olsson et al.)	• Implements a tournament in which a player is either a discriminator that attempts to distinguish between real and fake data or a generator that attempts to fool the discriminators into accepting fake data as real.
	18. Normalized Relative Discriminative Score (NRDS) (Zhang et al.)	• Compares n GANs based on the idea that if the generated samples are closer to real ones, more epochs would be needed to distinguish them from real samples.
	19. Adversarial Accuracy and Divergence (Yang et al.)	• Adversarial Accuracy. Computes the classification accuracies achieved by the two classifiers, one trained on real data and another on generated data, on a labeled validation set to approximate $P_g(y\|\mathbf{x})$ and $P_r(y\|\mathbf{x})$. Adversarial Divergence: Computes $\mathbb{KL}(P_g(y\|\mathbf{x}), P_r(y\|\mathbf{x}))$
	20. Geometry Score (Khrulkov and Oseledets)	• Compares geometrical properties of the underlying data manifold between real and generated data.
	21. Reconstruction Error (Xiang and Li, 2017)	• Measures the reconstruction error (*e.g.* L_2 norm) between a test image and its closest generated image by optimizing for z (*i.e.* $min_{\mathbf{z}}\\|G(\mathbf{z}) - \mathbf{x}^{(test)}\\|^2$)
	22. Image Quality Measures (Wang et al., 2004; Ridgeway et al.; Juefei-Xu et al.)	• Evaluates the quality of generated images using measures such as SSIM, PSNR, and sharpness difference
	23. Low-level Image Statistics (Zeng et al.; Karras et al.)	• Evaluates how similar low-level statistics of generated images are to those of natural scenes in terms of mean power spectrum, distribution of random filter responses, contrast distribution, etc.
	24. Precision, Recall and F_1 score (Lucic et al.)	• These measures are used to quantify the degree of overfitting in GANs, often over toy datasets.
	1. Nearest Neighbors	• To detect overfitting, generated samples are shown next to their nearest neighbors in the training set
Qualitative	2. Rapid Scene Categorization (Goodfellow et al., 2014)	• In these experiments, participants are asked to distinguish generated samples from real images in a short presentation time (*e.g.* 100 ms); *i.e.* real v.s fake
	3. Preference Judgment (Huang et al., 2017; Zhang et al.; Xiao et al.; Yi et al.)	• Participants are asked to rank models in terms of the fidelity of their generated images (*e.g.* pairs, triples)
	4. Mode Drop and Collapse (Srivastava et al., 2017; Lin et al.)	• Over datasets with known modes (*e.g.* a GMM or a labeled dataset), modes are computed as by measuring the distances of generated data to mode centers
	5. Network Internals (Radford et al.; Chen et al., 2016; Higgins et al.; Mathieu et al., 2016; Zeiler and Fergus, 2014; Bau et al., 2017)	• Regards exploring and illustrating the internal representation and dynamics of models (*e.g.* space continuity) as well as visualizing learned features

Figure C.1: GAN Evaluation Methods - Taken from Borji (2019)

D BigGAN Layer Summary

Table D.1: BigGAN generator

Layer	Output Shape	Param #	Connected to
input_101	[(None, 100)]	0	
spectral_normalization_1312	(None, 256)	26112	input_101[0][0]
reshape_84	(None, 2, 128)	0	spectral_normalization_1312[0][0]
batch_normalization_1008	(None, 2, 128)	512	reshape_84[0][0]
re_lu_312	(None, 2, 128)	0	batch_normalization_1008[0][0]
up_sampling1d_192	(None, 4, 128)	0	re_lu_312[0][0]
spectral_normalization_1313	(None, 4, 64)	41088	up_sampling1d_192[0][0]
batch_normalization_1009 (Batch	(None, 4, 64)	256	spectral_normalization_1313[0][0]
up_sampling1d_193	(None, 4, 128)	0	reshape_84[0][0]
re_lu_313 (ReLU)	(None, 4, 64)	0	batch_normalization_1009[0][0]
spectral_normalization_1315	(None, 4, 64)	8320	up_sampling1d_193[0][0]
spectral_normalization_1314	(None, 4, 64)	20608	re_lu_313[0][0]
tf.__operators__.add_296	(None, 4, 64)	0	spectral_normalization_1315[0][0]
batch_normalization_1010	(None, 4, 64)	256	tf.__operators__.add_296[0][0]
re_lu_314	(None, 4, 64)	0	batch_normalization_1010[0][0]
up_sampling1d_194	(None, 8, 64)	0	re_lu_314[0][0]
spectral_normalization_1316	(None, 8, 32)	10304	up_sampling1d_194[0][0]
batch_normalization_1011	(None, 8, 32)	128	spectral_normalization_1316[0][0]
up_sampling1d_195	(None, 8, 64)	0	tf.__operators__.add_296[0][0]
re_lu_315	(None, 8, 32)	0	batch_normalization_1011[0][0]
spectral_normalization_1318	(None, 8, 32)	2112	up_sampling1d_195[0][0]
spectral_normalization_1317	(None, 8, 32)	5184	re_lu_315[0][0]
tf.__operators__.add_297	(None, 8, 32)	0	spectral_normalization_1318[0][0]

(continued…)

Layer	Output Shape	Param #	Connected to
spectral_normalization_1320	(None, 8, 4)	136	tf.__operators__.add_297[0][0]
spectral_normalization_1319	(None, 8, 4)	136	tf.__operators__.add_297[0][0]
max_pooling1d_144	(None, 4, 4)	0	spectral_normalization_1320[0][0]
tf.linalg.matmul_144	(None, 8, 4)	0	spectral_normalization_1319[0][0]
spectral_normalization_1321	(None, 8, 16)	544	tf.__operators__.add_297[0][0]
tf.nn.softmax_40	(None, 8, 4)	0	tf.linalg.matmul_144[0][0]
max_pooling1d_145	(None, 4, 16)	0	spectral_normalization_1321[0][0]
tf.linalg.matmul_145	(None, 8, 16)	0	tf.nn.softmax_40[0][0]
spectral_normalization_1322	(None, 8, 32)	576	tf.linalg.matmul_145[0][0]
residual_link_72	(None, 8, 32)	1	tf.__operators__.add_297[0][0]
batch_normalization_1012	(None, 8, 32)	128	residual_link_72[0][0]
re_lu_316	(None, 8, 32)	0	batch_normalization_1012[0][0]
up_sampling1d_196	(None, 16, 32)	0	re_lu_316[0][0]
spectral_normalization_1323	(None, 16, 16)	2592	up_sampling1d_196[0][0]
batch_normalization_1013	(None, 16, 16)	64	spectral_normalization_1323[0][0]
up_sampling1d_197	(None, 16, 32)	0	residual_link_72[0][0]
re_lu_317	(None, 16, 16)	0	batch_normalization_1013[0][0]
spectral_normalization_1325	(None, 16, 16)	544	up_sampling1d_197[0][0]
spectral_normalization_1324	(None, 16, 16)	1312	re_lu_317[0][0]
tf.__operators__.add_298	(None, 16, 16)	0	spectral_normalization_1325[0][0]
batch_normalization_1014	(None, 16, 16)	64	tf.__operators__.add_298[0][0]
re_lu_318	(None, 16, 16)	0	batch_normalization_1014[0][0]
up_sampling1d_198	(None, 32, 16)	0	re_lu_318[0][0]
spectral_normalization_1326	(None, 32, 8)	656	up_sampling1d_198[0][0]
batch_normalization_1015	(None, 32, 8)	32	spectral_normalization_1326[0][0]
up_sampling1d_199	(None, 32, 16)	0	tf.__operators__.add_298[0][0]
re_lu_319	(None, 32, 8)	0	batch_normalization_1015[0][0]
spectral_normalization_1328	(None, 32, 8)	144	up_sampling1d_199[0][0]
spectral_normalization_1327	(None, 32, 8)	336	re_lu_319[0][0]
tf.__operators__.add_299	(None, 32, 8)	0	spectral_normalization_1328[0][0]
batch_normalization_1016	(None, 32, 8)	32	tf.__operators__.add_299[0][0]
re_lu_320	(None, 32, 8)	0	batch_normalization_1016[0][0]

Table D.2: BigGAN discriminator

Layer	Output Shape	Param #	Connected to
input_102	[(None, 32, 1)]	0	
spectral_normalization_1330	(None, 32, 8)	56	input_102[0][0]
leaky_re_lu_760	(None, 32, 8)	0	spectral_normalization_1330[0][0]
spectral_normalization_1331	(None, 16, 8)	336	leaky_re_lu_760[0][0]
spectral_normalization_1332	(None, 16, 8)	32	input_102[0][0]
tf.__operators__.add_300	(None, 16, 8)	0	spectral_normalization_1331[0][0]
batch_normalization_1017	(None, 16, 8)	32	tf.__operators__.add_300[0][0]
leaky_re_lu_761	(None, 16, 8)	0	batch_normalization_1017[0][0]
spectral_normalization_1333	(None, 16, 16)	672	leaky_re_lu_761[0][0]
batch_normalization_1018	(None, 16, 16)	64	spectral_normalization_1333[0][0]
leaky_re_lu_762	(None, 16, 16)	0	batch_normalization_1018[0][0]
spectral_normalization_1335	(None, 8, 16)	288	tf.__operators__.add_300[0][0]
spectral_normalization_1334	(None, 8, 16)	1312	leaky_re_lu_762[0][0]
tf.__operators__.add_301	(None, 8, 16)	0	spectral_normalization_1335[0][0]
spectral_normalization_1337	(None, 8, 2)	36	tf.__operators__.add_301[0][0]
spectral_normalization_1336	(None, 8, 2)	36	tf.__operators__.add_301[0][0]
max_pooling1d_146	(None, 4, 2)	0	spectral_normalization_1337[0][0]
tf.linalg.matmul_146	(None, 8, 4)	0	spectral_normalization_1336[0][0]
spectral_normalization_1338	(None, 8, 8)	144	tf.__operators__.add_301[0][0]
tf.nn.softmax_41	(None, 8, 4)	0	tf.linalg.matmul_146[0][0]
max_pooling1d_147	(None, 4, 8)	0	spectral_normalization_1338[0][0]
tf.linalg.matmul_147	(None, 8, 8)	0	tf.nn.softmax_41[0][0]
spectral_normalization_1339	(None, 8, 16)	160	tf.linalg.matmul_147[0][0]
residual_link_73	(None, 8, 16)	1	tf.__operators__.add_301[0][0]
batch_normalization_1019	(None, 8, 16)	64	residual_link_73[0][0]
leaky_re_lu_763	(None, 8, 16)	0	batch_normalization_1019[0][0]
spectral_normalization_1340	(None, 8, 32)	2624	leaky_re_lu_763[0][0]
batch_normalization_1020	(None, 8, 32)	128	spectral_normalization_1340[0][0]
leaky_re_lu_764	(None, 8, 32)	0	batch_normalization_1020[0][0]
spectral_normalization_1342	(None, 4, 32)	1088	residual_link_73[0][0]
spectral_normalization_1341	(None, 4, 32)	5184	leaky_re_lu_764[0][0]

(continued…)

Layer	Output Shape	Param #	Connected to
tf.__operators__.add_302	(None, 4, 32)	0	spectral_normalization_1342[0][0]
batch_normalization_1021	(None, 4, 32)	128	tf.__operators__.add_302[0][0]
leaky_re_lu_765	(None, 4, 32)	0	batch_normalization_1021[0][0]
spectral_normalization_1343	(None, 4, 64)	10368	leaky_re_lu_765[0][0]
batch_normalization_1022	(None, 4, 64)	256	spectral_normalization_1343[0][0]
leaky_re_lu_766	(None, 4, 64)	0	batch_normalization_1022[0][0]
spectral_normalization_1345	(None, 2, 64)	4224	tf.__operators__.add_302[0][0]
spectral_normalization_1344	(None, 2, 64)	20608	leaky_re_lu_766[0][0]
tf.__operators__.add_303	(None, 2, 64)	0	spectral_normalization_1345[0][0]
batch_normalization_1023	(None, 2, 64)	256	tf.__operators__.add_303[0][0]
leaky_re_lu_767	(None, 2, 64)	0	batch_normalization_1023[0][0]
spectral_normalization_1346	(None, 2, 128)	41216	leaky_re_lu_767[0][0]
batch_normalization_1024	(None, 2, 128)	512	spectral_normalization_1346[0][0]
leaky_re_lu_768	(None, 2, 128)	0	batch_normalization_1024[0][0]
spectral_normalization_1348	(None, 1, 128)	16640	tf.__operators__.add_303[0][0]
spectral_normalization_1347	(None, 1, 128)	82176	leaky_re_lu_768[0][0]
tf.__operators__.add_304	(None, 1, 128)	0	spectral_normalization_1348[0][0]
batch_normalization_1025	(None, 1, 128)	512	tf.__operators__.add_304[0][0]
leaky_re_lu_769	(None, 1, 128)	0	batch_normalization_1025[0][0]
spectral_normalization_1349	(None, 1, 128)	82176	leaky_re_lu_769[0][0]
batch_normalization_1026	(None, 1, 128)	512	spectral_normalization_1349[0][0]
leaky_re_lu_770	(None, 1, 128)	0	batch_normalization_1026[0][0]
spectral_normalization_1350	(None, 1, 128)	82176	leaky_re_lu_770[0][0]
tf.__operators__.add_305	(None, 1, 128)	0	tf.__operators__.add_304[0][0]
batch_normalization_1027	(None, 1, 128)	512	tf.__operators__.add_305[0][0]
leaky_re_lu_771	(None, 1, 128)	0	batch_normalization_1027[0][0]
tf.math.reduce_sum_24	(None, 128)	0	leaky_re_lu_771[0][0]
spectral_normalization_1351	(None, 1)	130	tf.math.reduce_sum_24[0][0]