The Signature-Wasserstein GAN for Time Series Generation and Beyond

Baoren Xiao

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

of

University College London.

Department of Mathematics

University College London

October 29, 2023

Acknowledgements

I would like to express my deepest gratitude and appreciation to many individuals who have supported and guided me through my doctoral journey.

First, of foremost, I am sincerely grateful to my supervisor, Prof. Hao Ni, for her unwavering support, invaluable guidance, and continuous encouragement. Her expertise, insightful feedback, and dedication have been instrumental in shaping the direction of my research and pushing me to achieve my best. Without her, my PhD and this thesis would not be possible.

I am also grateful to all my academic collaborators and colleagues for their invaluable support and collaboration throughout my research journey; in particular, Shujian Liao, Hang Lou, Jiajie Tao, Magnus Wiese, Marc Sabate-Vidales, Lukasz Szpruch. Their insightful discussions, constructive feedback, and willingness to share their expertise have greatly enriched my work and contributed to its overall quality.

Last but not least, I owe my thanks to my family members for their financial and emotional support. I will be forever grateful for your infinite help throughout my life.

To everyone who has played a part, big or small, in shaping my academic and personal growth, I am sincerely grateful. Thank you for being a part of this incredible journey.

Abstract

Time series is a vital source of information in many prominent domains such as finance, medicine, and geophysics. However, the acquisition of those time-series data is difficult due to high collection costs and privacy constraints. In recent years, generative models have been at the heart of a viable solution to this problem. Generative adversarial networks (GANs) that are originally designed for image generation have also shown remarkable success in generating realistic-looking time series. However, most GANs suffer from unstable training processes and high computational costs due to the min-max problem in their framework. To overcome this problem, we propose conditional Signature-based Wasserstein GAN (SigCWGAN), a novel framework that incorporates signature methods into GANs. The signature method provides an efficient and mathematically principled approach to extracting features from time-series data. SigCWGAN uses the conditional Sig-W_1 metric as the loss function, which turns computationally challenging GAN min-max problems into supervised learning, and hence saves computational time. We validate our proposed model on both synthetic data generated by popular quantitative risk models and empirical financial data. Our results demonstrate that SigCWGAN outperforms state-of-the-art benchmarks in terms of measures of similarity and predictive ability.

Furthermore, we introduce MCGAN, a more general framework that enhances generator performance. MCGAN incorporates mean squared error (MSE) and the Monte-Carlo method into the generative loss function, providing strong supervision for the generator. As shown in this thesis, MCGAN considerably improves the training stability while retaining the optimality of the original GAN. Numerical experiments on synthetic and empirical datasets showcase the superior performance and better training stability of MCGAN compared to the original GAN.

Keywords: Rough Path Theory, Generative Adversarial Networks, Time-series Generation, Monte Carlo Method.

Impact Statement

The main motivation of this thesis lies in the increased interest in the financial time-series generation in the financial industry. The work presented in this thesis addresses the challenging problem of generating realistic time-series data based on historical information. By proposing novel frameworks, namely SigCWGAN and MCGAN, this work advances the field of generative modeling by overcoming key limitations in the training process and generation performance of conventional GANs.

The adoption of the signature method in SigCWGAN enables efficient and principled feature extraction from time-series data. Additionally, the proposed Sig-W_1 metric provides an efficient technique to measure the discrepancy between two time-series distributions. The use of the conditional Sig-W_1 metric as the loss function in SigCWGAN transforms the GAN's min-max problem into supervised learning, resulting in improved computational efficiency and faster convergence. Empirical evaluations on synthetic and real-world financial datasets demonstrate that SigCWGAN outperforms state-of-the-art benchmarks in terms of similarity measures and predictive capabilities.

Additionally, the proposed MCGAN framework introduces the integration of mean squared error (MSE) and the Monte Carlo method into the generative loss function. This enhancement significantly improves the training stability of the generator while preserving the optimality of the original GAN. Numerical experiments validate the superior performance and enhanced training stability of MCGAN over the conventional GAN. MCGAN is a more general framework and can be also used in other types of conditional generation tasks like conditional image generation.

These two models provide practical implications for various industries by providing reliable and efficient methods for generating realistic time-series data. These advancements have the potential to facilitate better risk modeling, decision-making processes, and scenario analysis in the fields of finance, medicine, and geophysics. Moreover, the proposed frameworks contribute to the broader field of generative modeling, paving the way for further development in the generation of complex and high-fidelity time-series data.

In general, this research increases the potential of generative modeling techniques and significantly advances both academia and industry. The proposed frameworks offer improved efficiency, stability, and performance, thus opening new avenues for applications in diverse fields and fostering advancements in the understanding and utilization of time-series data.

Contents

1 Introduction **19**

1.1 Main contributions 21

1.2 Outline 22

2 Preliminary **24**

2.1 Rough path theory 24

2.1.1 Path with finite p-variation 24

2.1.2 Tensor algebra space 27

2.1.3 Signature of a path 29

2.1.4 Important properties of signature 30

2.1.5 Expected signature of stochastic processes 34

2.1.6 Path augmentation 35

2.2 Generative adversarial network (GAN) 38

2.2.1 Classical generative adversarial network 38

2.2.2 Wasserstein generative adversarial network (WGAN) 41

2.2.3 Conditional generative adversarial network 43

2.2.4 Problem of GAN training 44

3 Signature-based W_1 metric **46**

3.1 Sig-W_1 metric 46

3.2 Conditional Sig-W_1 metric 53

4 Signature-based conditional WGAN **55**

4.1 Conditional AR-FNN generator 56

4.2 Signature-based loss function 58
4.2.1 Learning the conditional expected signature under true measure 59
4.3 Algorithm 61
4.4 Numerical results 62
4.4.1 Vector autoregressive model 66
4.4.2 ARCH model 68
4.4.3 S&P 500 and DJI market data 69
4.5 Conclusions 73

5 Monte-Carlo GAN 74
5.1 Motivation 74
5.2 Discriminability 77
5.3 Monte Carlo GAN 83
5.3.1 Relation to f-divergence 87
5.3.2 Improved stability 89
5.3.3 A toy example: Dirac-GAN 92
5.4 Related work 93
5.4.1 Relation to feature matching 94
5.5 MCGAN for conditional time-series generation 95
5.5.1 Algorithm 97
5.6 Numerical experiments 97
5.6.1 Vector autoregressive model 99
5.6.2 ARCH model 100
5.6.3 Stock dataset 103
5.7 Conclusions 107

6 Conclusion and future direction 109
6.1 Conclusion 109
6.2 Discussion and future direction 110
6.2.1 About SigCWGAN 110

6.2.2 About MCGAN 111

Appendices **113**

A Rough Paths **113**

A.1 *p*-rough paths 113

A.2 Expected signature of stochastic processes 115

B Basics of neural networks **116**

B.1 AR-FNN architecture 118

C Supplementary numerical results **121**

C.1 Implementation and hyper-parameters 121

C.1.1 Truncation level of signature in SigCWGAN 121

C.2 VAR(1) dataset 122

C.3 ARCH(k) 129

C.4 SPX and DJI dataset 134

Bibliography **141**

List of Figures

2.1 A simple representation of a generative adversarial network (GAN) architecture 39

3.1 The top row displays blue and red samples from two distributions μ and ν respectively for fixed θ_1, and different values of θ_2. 52

4.1 Illustration of AR-FNN generator. 56

4.2 The illustration of the flowchart of SigCWGAN. 62

4.3 (Left) The distributional metric (*abs_metrics*) comparison; (Right) the R^2 (TSTR) comparison. VAR(1) data is generated for $\phi = 0.8$ and $\sigma = 0.8$. 66

4.4 Evolution of three test metrics: abs metric, ACF metric, and cross-correlation metric. Each color represents the metric of one dimension. The results are for the 3-dimensional VAR(1) model for $\phi = 0.8$ and $\sigma = 0.8$. 67

4.5 Comparison of the marginal distributions of one long sampled path (80,000 steps) with the real distribution using VAR(1) model for $\phi = 0.8, \sigma = 0.8$. 68

4.6 Comparison of all models' performance in fitting the conditional distribution of future time series given one past path sample. The real and generated paths are plotted in red and blue respectively with the shaded area as the 95% confidence interval. The real samples are synthesized from VAR(1) model for $d = 3$, $\phi = 0.8$ and $\sigma = 0.8$. 68

4.7 (Left) The distributional metric (acf_score) comparison; (Right) the R^2 error (TSTR) comparison using ARCH(k) data for $k = 2,3,4$. 70

4.8 ACF/PACF plot of squared residuals for ARCH(2) model. Here x-axis represents the lag value (with a maximum lag equal to 10) and the y-axis represent the corresponding auto- correlation/partial auto-correlation. The length of the real/generated time series used to compute the ACF is 1000. The number in the bracket under each model is the sum of the absolute difference between the correlation coefficients computed from real (dashed line) and generated (solid line) samples. 70

4.9 Comparison of the marginal distributions of the generated SigCWGAN paths and the SPX and DJI data. 71

4.10 ACF/PACF plot of absolute and squared returns for SPX (green line) and DJI (orange line). Here x-axis represents the lag value (with a maximum lag equal to 10) and the y-axis represents the corresponding auto-correlation/partial auto-correlation. The length of the real/generated time series used to compute the ACF is 1000. The number in the bracket under each model is the sum of the absolute difference between the correlation coefficients computed from real (dashed line) and synthetic (solid line) samples. 72

5.1 One illustration on the discriminability of the discriminator. In this example, $\mu \sim \mathcal{N}(-2,1), \nu_\theta \sim \mathcal{N}(2,1)$. The optimal discriminator is given by $D^{\phi^*}(x) = \frac{p_\mu(x)}{p_\mu(x)+p_{\nu_\theta}(x)}$, and the weaker version of discriminator $\hat{D}^{\phi^*}$ is defined in (5.18). 84

5.2 Convergence behavior of the Dirac-GAN. The starting point of iteration is marked in red. The total number of iterations is 2000. 93

5.3 Flowchart of generator training phase in MCGAN algorithm. 99

5.4 Comparison of the performance across all parameter sets (d,ϕ,σ) and benchmarks. 101

5.5 Comparison of training dynamics of MCGAN (blue) and RCGAN (orange). All four dynamics: (1) l^2 norm of generator gradient; (2) abs metric;(3) ACF metric;(4) cross-correlation metric is computed via the moving average (solid line) and standard deviation (shaded area) using window size 50. The training dataset is synthesized from VAR(1) model for $d = 3$, $\phi = 0.2$ and $\sigma = 0.8$. 102

5.6 Comparison of all models' performance in fitting the conditional distribution of future time series given one past path sample. The real and generated paths are plotted in red and blue respectively with the shaded area as the 95% confidence interval. The training dataset is synthesized from VAR(1) model for $d = 3$, $\phi = 0.8$ and $\sigma = 0.8$. 102

5.7 (Left) The distributional metric (abs metric) comparison; (Right) the R^2 error (TSTR) comparison using ARCH(k) data for $k = 2,3,4$. 103

5.8 ACF/PACF plot of squared residuals for ARCH(2) model. Here x-axis represents the lag value (with a maximum lag equal to 10) and the y-axis represents the corresponding auto- correlation/partial auto-correlation. The length of the real/generated time series used to compute the ACF is 1000. The number in the bracket under each model is the sum of the absolute difference between the correlation coefficients computed from real (dashed line) and generated (solid line) samples. 104

5.9 Comparison of R^2 relative error (%) with different lags. All models are trained on SPX and DJI data. 105

5.10 ACF/PACF plot of absolute and squared returns for SPX (green line) and DJI (orange line). Here x-axis represents the lag value (with a maximum lag equal to 10) and the y-axis represents the corresponding auto-correlation/partial auto-correlation. The length of the real/generated time series used to compute the ACF is 1000. The number in the bracket under each model is the sum of the absolute difference between the correlation coefficients computed from real (dashed line) and synthetic (solid line) samples. 106

5.11 Comparison of the marginal distributions of the generated MCGAN paths and the SPX and DJI data. 106

B.1 Architecture of residual layer. 119

D.1 Example development of the considered distances and score functions during training for the 1-dimensional VAR(1) model with autocorrelation coefficient $\phi = 0.8$ and co-variance parameter $\sigma = 0.8$. The colors blue and orange indicate the relevant distance/score for each dimension. 125

D.2 Example development of the considered distances and score functions during training for the 2-dimensional VAR(1) model with autocorrelation coefficient $\phi = 0.8$ and co-variance parameter $\sigma = 0.8$. The colors blue and orange indicate the relevant distance/score for each dimension. 126

D.3 Example development of the considered distances and score functions during training for the 3-dimensional VAR(1) model with autocorrelation coefficient $\phi = 0.8$ and co-variance parameter $\sigma = 0.8$. The colors blue and orange indicate the relevant distance/score for each dimension. 127

D.4 ACF plot for each channel on the 3-dimensional VAR(1) dataset with autocorrelation coefficient $\phi = 0.8$ and co-variance parameter $\sigma = 0.8$. Here x-axis represents the lag value (with a maximum lag equal to 100) and the y-axis represents the corresponding auto-correlation. The length of the real/generated time series used to compute the ACF is 1000. The number in the bracket under each model is the sum of the absolute difference between the correlation coefficients computed from real (dashed line) and generated (solid line) samples. 128

D.5 The plot displays a comparison of the marginal distribution on the linear- and log-scale, as well as the fit of the auto-correlation for ARCH(3) data. Histograms and the auto-correlation of the data are indicated in blue and the fits from the generator are colored in orange. 132

D.6 ACF plot for each channel on the ARCH(2) dataset. Here x-axis represents the lag value (with a maximum lag equal to 100) and the y-axis represents the corresponding auto-correlation. The length of the real/generated time series used to compute the ACF is 1000. The number in the bracket under each model is the sum of the absolute difference between the correlation coefficients computed from real (dashed line) and generated (solid line) samples. 133

D.7 Example development of the considered distances and score functions during training for SPX data. 136

D.8 Example development of the considered distances and score functions during training for SPX and DJI data. 137

D.9 Comparison of real and synthetic cross-correlation matrices for SPX/ SPX and DJI data. On the far left the real cross-correlation matrix from SPX/ SPX and DJI log-return and log-volatility data is shown. *x*/*y*-axis represents the feature dimension while the color of the $(i, j)^{th}$ block represents the correlation of $X_t^{(i)}$ and $X_t^{(j)}$. Observe that the historical correlation between log returns and log volatility is negative, indicating the presence of leverage effects, i.e. when log returns are negative, log volatility is high. 138

D.10 ACF plot for each channel on the SPX/DJI dataset. Here *x*-axis represents the lag value (with a maximum lag equal to 100) and *y*-axis represents the corresponding auto-correlation. The length of the real/generated time series used to compute the ACF is 1000. The number in the bracket under each model is the sum of the absolute difference between the correlation coefficients computed from real (dashed line) and generated (solid line) samples. 139

D.11 Example paths of SPX log returns generated by each model. Since the path of DJI log returns is similar to that of SPX, there is no need to make another plot for DJI. 140

List of Tables

4.1 Numerical results of the stock datasets. In each cell, the left/right numbers are the results for the SPX data/ the SPX and DJI data respectively. We use the relative error of TSTR R^2 against TRTR R^2 as the R^2 metric. 72

4.2 Results of coverage ratio test applied to SPX and DJI stock returns. In each cell, the left/right numbers are the results for the SPX/ DJI data respectively. A ratio that is closer to 5% indicates a more accurate 95% VaR estimate. The VaR estimate of each model for each observation is obtained using 5000 generated samples. 73

5.1 List of common discriminative loss functions that satisfy strict discriminability 84

5.2 Numerical results of the stock datasets. In each cell, the left/right numbers are the results for the SPX data/ the SPX and DJI data respectively. We use the relative error of TSTR R^2 against TRTR R^2 as the R^2 metric. 104

5.3 Results of coverage ratio test applied to SPX and DJI stock returns. In each cell, the left/right numbers are the results for the SPX/ DJI data respectively. A ratio that is closer to 5% indicates a more accurate 95% VaR estimate. The VaR estimate of each model for each observation is obtained using 5000 generated samples. 107

D.1 Monte Carlo sample size used in SigCWGAN and MCGAN for each example. 121

D.2 Truncation level of signature used in SigCWGAN for each example. 121
D.3 Numerical results of VAR(1) for $d = 1$ 122
D.4 Numerical results of VAR(1) for $d = 2$ 123
D.5 Numerical results of VAR(1) for $d = 3$ 124
D.6 Numerical results of the ARCH(k) datasets. 130
D.7 Relative R^2 error (%) of the ARCH(3) model for different lag values. 131
D.8 Numerical results of the stocks datasets. 135
D.9 Relative R^2 error (%) of the SPX data for different lag values. 138
D.10 Relative R^2 error (%) of the SPX and DJI data for different lag values. 138
D.11 Autocorrelation metric for the stock datasets for different lag values. In each cell, the left/right numbers are the result for the SPX data/ the SPX and DJI data respectively. 139

Chapter 1

Introduction

Financial services generate a huge amount of data that is complex and varied. These datasets contain some highly sensitive and personally identifiable attributes of customers, imposing significant restrictions on their utilization or dissemination for research purposes. It is therefore imperative to investigate the generation of synthetic data for numerous applications in the finance industry. Assefa et al. [1] highlighted the growing need for effective synthetic data generation in the financial domain, which focuses on three main areas for the academic community: 1) Generating realistic synthetic datasets; 2) Measuring the similarities between real and generated datasets; 3) Ensuring the generative process to satisfy any privacy constraints. Bellovin et al. [2] also pointed out that synthetic data can serve as a valid and privacy-conscious alternative to raw data.

In particular, synthetic time series can facilitate the training and validation of data-driven risk models and enable data sharing while upholding privacy requirements. Generative adversarial networks (GANs) [3], initially designed for synthetic image generation, have been growing popular in time-series generation since its birth. Many quantitative researchers have investigated the usage of GAN in quantitative risk management. Buehler et al. [4] constructed realistic equity option market simulators based on GANs and demonstrated that GANs can be applied for multivariate financial time-series generation. Inspired by the recent success of generative adversarial networks, Magnus et al. [5] proposed Quant GAN, whose generator is explicitly constructed such that the stochastic process induced by the generator allows a transition to its risk-neutral distribution, facilitating pricing tasks. We refer the readers to [6, 7, 8, 9] for generative modeling perspective of some classical problems in quantitative risk management.

While generative modeling has been highly successful in generating samples from seemingly high-dimensional probability measures, off-the-shelf techniques, such as generative adversarial networks (GANs) [3], struggle to capture the temporal dependence of joint probability distributions induced by time-series data. Furthermore, the min-max objective function of classical GANs makes them notoriously difficult to tune [10, 11, 12]. Many techniques have been developed to stabilize the GAN training [13, 14, 15]. However, those techniques either introduce considerable computational costs or heavily rely on hyperparameter tuning. Hence training instability and unsatisfactory performance remains a problem.

To address these challenges and improve the performance of GANs, we propose the signature-based method as a mathematically principled feature extraction machinery that emerged from the theory of rough paths. Rough path theory, developed in the mid-90s by Terry Lyons [16], is a generalization of the notion of a smooth path allowing to construct a robust solution to controlled differential equations driven by irregular signals. The signature of a path, as the core of rough path theory, is defined as the collection of iterated integrals of the path and is considered as an efficient feature extraction. The signature method has drawn the attention of quantitative researchers in the financial industry. Buehler et al. [7] showed how the signature of a path can be combined with a variational autoencoder framework, which provides a powerful way for encoding and evaluating financial time series and outperforms returns-based data generation by reducing the overfitting when available training data is scare. In other areas, like landmark-based human action recognition, Yang et al. [17] considered the evolving landmark data as a high-dimensional path and applied signature transform for both spatial structure and temporal dynamics to generate discriminative features, which achieve comparable state-of-the-art performance to the advanced deep networks. Liao et al. [18] enhanced the RNN model by incorporating log signature into the model and lifting the discrete-time model into a continuous-time one. We aim to combine the signature method and generative models to enhance the performance of WGAN for conditional time-series generation

1.1 Main contributions

Part of the work, the SigCWGAN model, presented in this thesis is a joint work published in [19] and [20]. My contribution mainly focuses on two parts: (1) the numerical implementation of the proposed conditional SigWGAN algorithm and its applications on financial datasets; (2) the design of the MCGAN algorithm, its theoretical guarantee, and all the numerical experiments. The major contribution of our work lies in three aspects:

Firstly, we propose a novel metric, Sig-W_1 metric, which is inspired by the W_1 metric and used to measure the difference between the two measures. Thanks to the universality of signature, we are able to derive the analytic formula for Sig-W_1 metric and hence reduce the min-max formulation of GAN into a supervised learning problem. We also propose the conditional Sig-W_1 metric which takes the past path as additional information and is constructed as the difference between the conditional expected signatures under two different measures.

Secondly, for the conditional time-series generation task, we propose Signature-based Conditional Wasserstein GAN (SigCWGAN) based on the conditional Sig-W_1 metric. It is efficient for data streams sampled at high frequency and significantly reduces the computational cost as no min-max problem is involved. We also design the AR-FNN network as a conditional generator that maps the past path and noise vectors into the future path and capture the auto-regressive nature of time series.

Thirdly, we introduce a general framework called Monte-Carlo GAN (MCGAN) that enhances the conditional generation performance from the perspective of the generator. MCGAN incorporates mean square error (MSE) and the Monte-Carlo method into the generative loss function, which measures the distance between the expected discriminative scores of real and generated samples, providing stronger supervision for the generator. Benefiting from the strong supervision, this novel generative loss function shows improved training stability meanwhile retaining the optimality of the original GANs where fake measure coincides with real measure. This MCGAN framework is not limited to time-series generation; it offers broader applicability to other conditional generation tasks, including image generation.

1.2 Outline

The outline of this thesis is as follows:

In Chapter 2, we first overview the key elements of rough path theory. In particular, we introduce the definition of the signature and its important properties. Then we introduce the conventional GAN framework and W_1 metric. We explain the motivation for improving the conventional GAN algorithm.

In Chapter 3, we start from the W_1 metric on the signature space and propose a novel metric called Sig-W_1 metric, which can be used to quantify the distance between two measures induced by time series. Then we derive its analytic form from the universality of signature. For a conditional generative model, we also introduce the conditional Sig-W_1 metric that takes the past path as a conditional variable.

In Chapter 4, we proposed the SigCWGAN model. It employs conditional Sig-W_1 metric as the loss function. To compute this metric, we apply the OLS regression to estimate the real conditional expected signature. Then the conditional generator, whose architecture is specified as an AR-FNN network, is trained to minimize the estimated conditional Sig-W_1 metric. The numerical experiments on both synthetic and empirical datasets show that our SigCWGAN model achieves significantly better performance in capturing the conditional distribution and has much more stable training dynamics than other baselines.

In Chapter 5, we introduce the Monte Carlo GAN (MCGAN) to improve the GAN performance from the perspective of the generator. The MCGAN incorporates MSE and the Monte-Carlo method into the generative loss function, which provides strong supervision for generator training. We proved that this strong supervision allows the generator to learn the target measure with a relatively weak condition imposed on the (non-optimal) discriminator. Additionally, we showed that MCGAN has better training stability than the original GAN given a noisy discriminator. To demonstrate the effectiveness of MCGAN, we also conducted numerical experiments on both synthetic and empirical datasets, highlighting the improved stability and superior performance achieved by MCGAN.

In Chapter 6, we make a conclusion of our work and discuss some limitations of SigCWGAN and MCGAN, based on which we propose avenues for potential future research to address these concerns.

Chapter 2

Preliminary

In this chapter, we start with an introduction to the basics of rough path theory and then summarize the framework of generative adversarial neural networks (GANs) for synthetic data generation. The key object of rough path theory, called the signature of a path, provides a mathematically principled and universal feature of time series (paths). Both rough path theory and GANs serve as an important foundation for the proposed SigWGAN algorithm.

2.1 Rough path theory

The core of rough path theory is called the signature of a path [16] that is defined as a collection of iterated integrals of the path and is considered as an efficient feature extraction of time series. Before we go through the details of the signature, we first introduce the p-variation that measures the roughness of a path and the space where the signature lives. We follow all the notations from [16].

2.1.1 Path with finite p-variation

Let E be a Banach space endowed with a norm denoted by $|\cdot|$ and $J := [s,t]$ be a compact time interval. Let $X : J \to E$ denote a E-valued continuous path and the finite time partition on J is defined as: $\mathcal{D} := (t_0, t_1, t_2, \ldots, t_r) \in J$, where $s = t_0 \leq t_1 \leq \ldots \leq t_r = t$. We first introduce the *p-variation* as a measure of the roughness of a path.

Definition 2.1.1 (p-variation)**.** Let $p \geq 1$ be a real number. Let $X : J \to E$ be a continuous path. The p-variation of X on the interval J is defined by

$$\|X\|_{p,J} = \left[\sup_{\mathcal{D}\in J} \sum_{j=0}^{r-1} |X_{t_{j+1}} - X_{t_j}|^p \right]^{\frac{1}{p}},$$

where the supremum is taken over any finite time partition $\mathcal{D}$ of J.

Remark 2.1.1. We emphasize here that the *supremum* is taken over all subdivisions of J, not a limit as the mesh of the subdivision tends to zero.

The p-variation $\|X\|_{p,J}$ is equal to zero if and only if X is a constant path. Below are some examples of paths with finite p-variation.

Example 2.1.1. Any continuously differentiable path $X : J \to E$ has finite 1-variation.

Example 2.1.2. For example, a Brownian motion has finite p-variation for $p > 2$ a.s., but it has infinite p-variation for $p \in [1,2]$.

For each $p \geq 1$, let $\mathcal{V}^p(J,E)$ denote the subset of $C^0(J,E)$, the space of E-valued continuous path on time interval J, consisting of those paths which have finite p-variation. For each $X \in \mathcal{V}^p(J,E)$, we define the p-variation norm as:

$$\|X\|_{\mathcal{V}^p(J,E)} = \|X\|_{p,J} + \sup_{t\in J} \|X_t\|.$$

Then $(\mathcal{V}^p(J,E), \|\cdot\|_{\mathcal{V}^p(J,E)})$ is a linear subspace of $C^0(J,E)$ endowed with p-variation topology. And it holds that, for $1 \leq p \leq q$,

$$\mathcal{V}^1(J,E) \subset \mathcal{V}^p(J,E) \subset \mathcal{V}^q(J,E) \subset C^0(J,E)$$

To avoid any ambiguity, we call $\|X\|_{p,J}$ the p-variation of X on J and $\|X\|_{\mathcal{V}_p(J,E)}$ the p-variation norm of X on J. Let E and W be two Banach spaces and J a compact time interval. Suppose $X \in \mathcal{V}^1(J,E)$ and $Y \in \mathcal{V}^1(J,W)$ satisfy the following linear rough differential equation,

$$dY_t = BY_t dX_t, \quad Y_0 \in W, \tag{2.1}$$

where $B : E \to \mathbf{L}(W)$ is a linear map from E to $\mathbf{L}(W)$ (the set of linear map from W to itself). The existence and uniqueness of the solution to the linear rough differential equation (2.1) can be found in [16]. Now our question is: what information about the path $\{X_t\}_{t>0}$ should we know in order to approximate the solution Y with a given precision?

To answer this question, let us apply the Picard iteration to solve this linear rough differential equation(2.1), which will result in a sequence of functions $(Y_{\cdot}^n)_{n\geq 0}$. For all $t \in [0,T]$, we set $Y_t^0 = I$,

$$Y_t^1 = Y_0 + \int_0^t BY_u^0 dX_u = Y_0 + \int_0^t B(dX_u).$$

Let us do this again and set $B^{\otimes k}(x_1 \otimes \cdots \otimes x_k) = B(x_k)\cdots B(x_1)$, we get

$$Y_t^2 = Y_0 + B\int_0^t dX_u + B^{\otimes 2}\int_{0<u_1<u_2<t} dX_{u_1} \otimes dX_{u_2}.$$

Keep iterating this process, we can get a sequence $\{Y^n\}_{n\in\mathbf{N}}$, where

$$Y_t^n = Y_0 + \sum_{k=1}^{n} B^{\otimes k}\int_{0<u_1<\cdots<u_k<t} dX_{u_1} \otimes \cdots \otimes dX_{u_k}.$$

The fact (as described in Lemma 2.1.3) that

$$\left\| \int_{0<u_1<\cdots<u_k<T} dX_{u_1} \otimes \cdots \otimes dX_{u_k} \right\| \leq \frac{\|X\|_{1,[0,T]}^k}{k!},$$

ensures the convergence of the iterative procedure above. We hence have a rapidly convergent expansion of the solution Y to (2.1):

$$Y_t = Y_0 + \sum_{k=1}^{\infty} B^{\otimes k}\int_{0<u_1<\cdots<u_k<t} dX_{u_1} \otimes \cdots \otimes dX_{u_k},$$

for all $t \in [0,T]$.

This helps us to answer the original question that in order to estimate the solution Y, the information we need on X is the collection of its iterated integrals denoted as $(X^0_{0,T}, X^1_{0,T}, X^2_{0,T}, \cdots, X^k_{0,T}, \cdots)$, where

$$X^k_{0,T} = \int_{0<u_1<\cdots<u_k<T} dX_{u_1} \otimes \cdots \otimes dX_{u_k} \in E^{\otimes k}.$$

Remark 2.1.2. Regarding the definition of the iterated integral for a path with finite p-variation, these iterated integrals can be defined as Young's integral [21] when $p < 2$. Young's integral is simply a limit of Riemann sums as for the Riemann-Stiletjes integral. In the case of $p > 2$, we can define the iterated integrals of a path with finite p-variation as a limit of iterated integrals of a path with bounded variation in p-variation topology. We refer those interested readers to [16].

2.1.2 Tensor algebra space

As a successive tensor series, the collection of iterated integrals of a path X can be identified with the space of homogeneous non-commuting polynomials. In this subsection, we introduce the space of power series called *tensor space* and use the convention $E^{\otimes 0} = \mathbb{R}$.

Definition 2.1.2 (Tensor algebra space)**.** The extended tensor algebra or formal series of tensors over E, denoted by $T((E))$, is defined to be the space of the following sequence,

$$T((E)) = \{(a_0, a_1, ..., a_n, ...) | a_n \in E^{\otimes n}\}.$$

It is equipped with two operations, an addition $+$ and a product $\otimes$, defined as follows. Let $\mathbf{a} = (a_i)_{i=0}^{\infty}, \mathbf{b} = (b_i)_{i=0}^{\infty} \in T((E))$, then

$$\mathbf{a} + \mathbf{b} = (a_i + b_i)_{i=0}^{\infty}$$
$$\mathbf{a} \otimes \mathbf{b} = (\sum_{k=0}^{i} a_i \otimes b_{i-k})_{i=0}^{\infty}$$

The space T((E)) endowed with the two operations and the natural action of R by $\lambda \mathbf{a} = (\lambda a_i)_{i=0}^{\infty}$ is a real non-commutative unital algebra, with unit $\mathbf{1} = (1, 0, 0, ...)$. The element $\mathbf{a}$ is invertible if and only if $a_0 \neq 0$, whose inverse is given by

$$\mathbf{a}^{-1} = \frac{1}{a_0}\sum_{n\geq 0}(1-\frac{\mathbf{a}}{a_0})^{\otimes n}$$

It is often important to look only at finitely many terms of an element of $T(E)$, we define the truncated tensor space $T^{(n)}(E)$ of order n as the quotient algebra of $T(E)$ as follows.

Definition 2.1.3. Let $n \geq 1$ be an integer. The truncated tensor algebra of order n of E is defined as the quotient algebra

$$T^{(n)}(E) = T((E))/B_n,$$

where $B_n = \{\mathbf{a} = (a_0, a_1, \ldots)|a_0 = \ldots = a_n = 0\}$. The canonical homomorphism $T((E)) \to T^{(n)}(E)$ is denoted by π_n.

The homomorphism π_n consists simply in forgetting the terms of degree greater than n.

For our analysis to work, we further introduce the admissible norm of tensor powers defined as follows.

Definition 2.1.4. We say that the tensor powers of E are endowed with an admissible norm $|.|$, if the following conditions hold:

1. For each $n \geq 1$, the symmetric group $\mathbb{S}_n$ acts by isometry on $E^{\otimes n}$, i.e.

$$|\sigma v| = |v| \quad \forall v \in E^{\otimes n}, \quad \forall \sigma \in \mathbb{S}_n;$$

2. The tensor product has norm 1, i.e. $\forall n, m \geq 1$,

$$|v \otimes w| \leq |v||w| \quad \forall v \in E^{\otimes n}, \quad \forall w \in E^{\otimes m}.$$

We then introduce the l_p norm of $T((E))$ for some $p \geq 1$. Let $(e_i)_{i=1}^d$ be a canonical basis of $E = \mathbb{R}^d$. Then we have a canonical basis of $T((E))$ by $(e_I :=$

$e_{i_1} \otimes \cdots e_{i_n})_{n \in \mathbb{R}}$. Therefore any element a in $T((E))$ can be written as

$$a = \sum_{n \in \mathbb{N}} \sum_{I=(i_1,\cdots,i_n) \in \{1,\cdots,d\}^n} a_I e_I. \tag{2.2}$$

The l_p norm of $a \in T((E))$ is denoted by $||a||_p$ and defined as

$$||a||_p = \left(\sum_{n \in \mathbb{N}} \sum_{I=(i_1,\cdots,i_n) \in \{1,\cdots,d\}^n} |a_I|^p \right)^{1/p}. \tag{2.3}$$

Similarly, we define the l_p norm of the algebraic dual space $T((E))^*$. Define the canonical basis of the dual space $T((E))^*$, i.e.

$$(e_I^*)_{I=(i_1,\cdots,i_n) \in \{1,\cdots,d\}^n,\ n \in \mathbb{N}}$$

by $\langle e_{I_1}^*, e_{I_2} \rangle = \mathbf{1}_{I_1 = I_2}$. Then the l_q norm of $L \in T((E))^*$ is denoted by $||L||_q$ and defined as

$$||L||_q = \left(\sum_{n \in \mathbb{N}} \sum_{I=(i_1,\cdots,i_n) \in \{1,\cdots,d\}^n} |L_I|^q \right)^{1/q}. \tag{2.4}$$

2.1.3 Signature of a path

After introducing the tensor algebra space, we can notice that the collection of iterated integral of path X on the time interval J: $(X_J^0, X_J^1, X_J^2, \ldots, X_J^k, \ldots)$ takes its value in $T((E))$. We name the collection of the iterated integral as the signature of path X.

Definition 2.1.5 (The signature of a path)**.** Let $X : J \to E$ be a continuous path of finite p-variation such that the following integration makes sense. The signature of X denoted by $S(X)_J$ is defined as an infinite series of X_J^k, i.e. $S(X)_J = (1, X_J^1, \ldots, X_J^k, \ldots)$, where

$$X_J^k = \int \cdots \int_{u_1 < \cdots < u_k;\, u_1, \ldots, u_k \in J} dX_{u_1} \otimes \cdots \otimes dX_{u_k}, \quad \forall k \geq 1.$$

Let $S_k(X)_J := \pi_k(S(X)_J)$ denote the truncated signature of X of degree k, i.e.

$S_k(X)_J = (1, X_J^1, \ldots, X_J^k)$.

Remark 2.1.3. [16] The signature of a path is defined as an element of a group, which is a subset of $T((E))$, defined as

$$\tilde{T}((E)) = \{\mathbf{a} \in T((E)) | a_0 = 1\}.$$

Remark 2.1.4. The lower levels of signature have their geometric interpretation. For instance, the first level signature X_J^1 is simply the increment of the path X, i.e. $X_t - X_s$, while the second level signature represents the signed area, also named as Lévy area, enclosed by the curve X and the cord connecting the ending and starting point of the path X.

The concept of a rough path is a generalization of the signature of a path and it is defined in a way to capture the analytic and algebraic properties of the signature of the path. A p-rough path is a multiplicative functional of degree $\lfloor p \rfloor$ with finite p-variation [16]. A geometric p-rough path is a p-rough path that can be represented as the limit of 1-rough path in p-variation distance, which we denote as $G\Omega_p(E)$. The definition of p-rough path and geometric p-rough path can be found in Appendix A.1. In this thesis, we only focus on those paths in $\mathcal{V}^1(J,E)$ so that their signature is well defined.

2.1.4 Important properties of signature

In this subsection, we summarize the main properties of signatures, which are important for statistical inference. The first property, named *multiplicative property*, asserts that the signature is a homomorphism. First, we define the concatenation of two paths.

Definition 2.1.6. Let $X : [0,s] \to E$ and $Y : [s,t] \to E$ be two continuous paths. Their concatenation is the path $X * Y$ defined by

$$(X * Y)_u = \begin{cases} X_u, & u \in [0,s]; \\ X_s + Y_u - Y_s, & u \in [s,t], \end{cases}$$

where $0 \leq s \leq t$.

For the concatenation of two paths, we have the following multiplicative property for their signature.

Theorem 2.1.1 (Chen's identity)**.** Fix $p \in [1,2)$ and $0 \leq s \leq t$. Let $X \in \mathcal{V}^p([0,s],E)$ and $Y \in \mathcal{V}^p([s,t],E)$, then it holds that

$$S(X * Y) = S(X) \otimes S(Y)$$

Remark 2.1.5. Chen's identity claims that the signature of the concatenation of two paths is just the tensor product of their signature, which makes the signature a group homomorphism.

The second property of the signature is the *shuffle product property*. Suppose $(e_1, \cdots, e_d)$ is a basis for the finite dimensional space E, and $(e_1^*, \cdots, e_d^*)$ is a basis for the dual space E^*. Then, the elements $(e_I = e_{i_1} \otimes \cdots \otimes e_{i_n})_{I=(i_1,\cdots,i_n)\in\{1,\cdots,d\}^n}$ form a basis of $E^{\otimes n}$, and the elements $(e_I^* = e_{i_1}^* \otimes \cdots \otimes e_{i_1}^*)_{I=(i_1,\cdots,i_n)\in\{1,\cdots,d\}^n}$ can be identified with the dual basis of $E^{\otimes n}$, then we may form a real number as

$$\phi_I(X) = e_I^*(S(X)) = \int \cdots \int_{0<u_1<\cdots<u_n<T} e_{i_1}^*(dX_{u_1}) \cdots e_{i_n}^*(dX_{u_n}).$$

Definition 2.1.7. We define the set $\mathbb{S}_{m,n}$ of (m,n) shuffles to be the subset of permutations in the symmetric group $\mathbb{S}_{m+n}$ defined by

$$\mathbb{S}_{m,n} = \{\sigma \in \mathbb{S}_{m+n} : \sigma(1) < \cdots < \sigma(m), \sigma(m+1) < \cdots < \sigma(m+n)\}.$$

The shuffle product property states that the point-wise product of two linear forms e_I^* and e_J^* is again a linear form which we denote as $e_I^* \sqcup\!\sqcup\, e_J^*$.

Theorem 2.1.2 (Shuffle product property)**.** Let $I = (i_1, \ldots, i_m)$ and $J = (j_1, \ldots, j_n)$ be two arbitrary indices. For every path $X \in \mathcal{V}^p(J,E)$, it holds that

$$\phi_I(X)\phi_J(X) = (\phi_I \sqcup\!\sqcup\, \phi_J)(X) = \sum_{\sigma \in \mathbb{S}_{m,n}} e_{k_{\sigma^{-1}(1)}}^* \cdots e_{k_{\sigma^{-1}(m+n)}}^*(S(X)).$$

The third property of the signature is *invariance under time reparameterization* (Lemma 1.6, [16]), which means reparameterizing a path inside the time interval does not change its signature.

Lemma 2.1.1 (Invariance under time reparameterization)**.** Let $X \in \mathcal{V}_p([0,T],E)$ and $\lambda : [0,T] \to [T_1,T_2]$ be a non-decreasing surjection and define $X_t^{\lambda} := X_{\lambda_t}$ for the reparamterization of X under λ. Then for every $s,t \in [0,T]$,

$$S(X)_{\lambda_s,\lambda_t} = S(X^{\lambda})_{s,t}.$$

The invariance under time reparameterization indicates that the signature feature can reduce dimension massively by removing the redundancy caused by the speed of traversing the path. It is very useful for applications where the output is invariant w.r.t. the speed of an input path, e.g. online handwritten character recognition and video classification.

Another important property of the signature is *uniqueness*. It was first proved for any continuous path of finite 1- variation [22], which has been extended to the case of a geometric p-rough path for some $p \geq 1$ [23]. For simplicity, we only state the results for $\mathcal{V}^1(J,E)$.

Theorem 2.1.3. Let $X \in \mathcal{V}^1(J,E)$. Then $S(X)$ determines X up to the tree-like equivalence.

A tree-like path can be regarded as a null path that can be canceled by itself, and we say path X and Y are tree-like equivalent if the concatenation of X and $\overleftarrow{Y}$ (the inverse path of Y) is a tree-like path. One can prove that $S(X * \overleftarrow{Y}) = 1$ if X and Y are tree-like equivalent. The precise definition of tree-like equivalence can be found in [16]. The following lemma provides a sufficient condition for the uniqueness of the signature.

Lemma 2.1.2. Let $X \in \mathcal{V}^1(J,E)$ with a fixed starting point and at least one coordinate of X is a monotone function. Then $S(X)$ determines X uniquely.

For a general class of multi-dimensional stochastic processes like Brownian motion, we refer the readers to [24] and [25] for a sufficient condition of the uniqueness of the signature. The uniqueness of the signature is important, as it ensures itself to be a discriminative feature set of unparameterized streamed data.

Let $\mathcal{S}(\mathcal{V}^p(J,E))$ denote the range of signature of all path in $\mathcal{V}^p(J,E)$. Note that the signature map, defined on $\mathcal{V}^1(J,E)$, is continuous with respect to the 1-variation topology. Given a function f on the signature space, one naturally might wonder how to approximate $f(a)$ for $a \in \mathcal{S}(\mathcal{V}^p(J,E))$. The signature of X provides insight into this problem. It is proved in [26] that any continuous function f on the compact subset of $\mathcal{S}(\mathcal{V}^p(J,E))$ can be approximated arbitrarily well by a linear functional.

Theorem 2.1.4 (Universality theorem)**.** Suppose $f : S_1 \to \mathbb{R}$ is a continuous function where S_1 is a compact subset of $\mathcal{S}(\mathcal{V}^p(J,E))$. Then for every $\varepsilon > 0$, there exists a linear functional $L \in T((E))^*$ such that for every $a \in S_1$,

$$|f(a) - L(a)| \leq \varepsilon.$$

Proof. Let $L(S_1)$ denote a family of all linear functions in $T((E))^*$ restricted to S_1. By the shuffle product property of signatures (Theorem 2.1.2), $L(S_1)$ is an algebra. Since the 0th term of the signature is always 1, this algebra contains constant functions. Moreover, it separates the points. (Details can be found in the proof of Corollary 2.16 in [16].) By Stone-Weierstrass theorem, $L(S_1)$ is dense in the space of continuous functions on S_1. $\square$

Theorem 2.1.4 applies to any subspace topology on $\mathcal{S}(\mathcal{V}^p(J,E))$, which is inherited from the Hausdorff topology on $T((E))$, that is finer than the weak topology. The universality of signature is important as it provides a way to estimate any continuous function on the path by using linear regression on the signature feature set.

Lastly, we state the decay rate of the signature for the path of finite 1-variation. However, there is also a similar statement of the factorial decay for the case of paths of finite p-variation [16].

Lemma 2.1.3 (Factorial decay of the signature [16])**.** Let $X \in \mathcal{V}^1(J,E)$. Then for all $m \geq 0$, it holds that

$$|\pi_m(S(X))| \leq \frac{\|X\|_{p,J}^m}{m!}.$$

2.1.5 Expected signature of stochastic processes

In this subsection, we introduce the expected signature. Given a filtered probability space $(\Omega, \mathcal{F}, (\mathcal{F}_t)_{t\in[0,T]}, \mathbb{P})$, where $(\mathcal{F}_t)_{t\in[0,T]}$ is a filtration, let X be a E-valued stochastic process, naturally one will be interested in the expectation of the signature of path $X(\omega)$ for $\omega \in \Omega$, which we denote as $\mathbb{E}(S(X(\omega)))$.

Definition 2.1.8. Given a filtered probability space $(\Omega, \mathcal{F}, (\mathcal{F}_t)_{t\in[0,T]}, \mathbb{P})$, X is a E-valued stochastic process. Suppose that for every $\omega \in \Omega$, the signature of $X(\omega)$ denoted by $S(X(\omega))$ (or $\mathbf{X}(\omega)$) is well defined and under the probability measure $\mathbb{P}$, its expectation denoted by $\mathbb{E}[S(X(\omega))]$ is finite. We call $\mathbb{E}[S(X(\omega))]$ the expected signature of X.

As the signature of a path can be thought of as non-commutative monomials on the path space, the expected signature of a random path plays a similar role as that the moment-generating function of a random variable does. The following theorem gives a sufficient condition that the expected signature determines the law on the signatures. Here $G\Omega_p(E)$ is the geometric p-rough path. The detailed definition of p-geometric rough path can be found in Appendix A.1.

Theorem 2.1.5 (Proposition 6.1, [27])**.** Let $\mathbf{X}$ and $\hat{\mathbf{X}}$ be two random variables taking values in $G\Omega_p(E)$ for some $p \geq 1$ such that $\mathbb{E}[\mathbf{X}] = \mathbb{E}[\hat{\mathbf{X}}]$ and $\mathbb{E}[\mathbf{X}]$ has an infinite convergence radius. Then $\mathbf{X} \overset{D}{=} \hat{\mathbf{X}}$.

Let $\Omega_0(J,E)$ denote the space of continuous d dimensional paths of finite variation starting from the origin, whose 0^{th} coordinate being time dimension. We endow $\Omega_0(J,E)$ with the 1-variation metric. When restricting the path space to $\Omega_0(J,E)$, the signature S is a bijective and continuous map, the measure μ on the path space induces the measure on the signature space $\tilde{\mu}$ by the push-forward of μ, i.e., $\tilde{\mu}(B) = (S_{\#}\mu)(B) = \mu(S^{-1}(B))$ for B is the σ-algebra of $\mathcal{S}(\Omega_0(J,E))$.

Lemma 2.1.4. Let μ, ν be two measures defined on the path space $\Omega_0(J,E)$. The corresponding induced measures of the signature are denoted by $\tilde{\mu}$ and $\tilde{\nu}$ respectively.

$$\tilde{\mu} = \tilde{\nu} \Longleftrightarrow \mu = \nu.$$

Proof. This is an immediate result of the bijective property of the signature map $S : \Omega_0(J,E) \to \mathcal{S}(\Omega_0(J,E))$. $\square$

Combined with the uniqueness of the signature, the Theorem 2.1.5 claims that the expected signature $\mathbb{E}(S(X(\omega)))$ can characterize the law of a stochastic process X, which makes the expected signature a very efficient feature set for time-series.

2.1.6 Path augmentation

In practice, the signature method is usually combined with path augmentations [28]. Different path augmentations can encode extra information into the signature of the augmented path which makes the signature a useful candidate for the feature set of a path. There are three main uses of path augmentation: (1) to remove the signature invariance to translation and/or reparameterization; (2) to reduce the dimension of the paths so that higher orders of the signature are reachable; (3) to extract useful information from the sequence.

In our work, we use the following augmentation methods:

- **Time augmentation** [26]: Let $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^{d \times n}$ be an observed sequence, time jointed transformation is defined as:

  $$\Phi(\mathbf{x}) = \big((t_1, x_1), (t_2, x_2), ..., (t_n, x_n)\big) \in \mathbb{R}^{d \times n},$$

  where $t_1 < t_2 < \ldots < t_n$ are increasing time index. It is obvious from the definition that it includes timestamps as an additional coordinate. The extra timestamp can bring us two key properties: (1) it ensures the uniqueness of the signature, as shown in [22]; (2) it provides information about the parameterization of the time series and hence removes the invariance to reparameterization.

- **Cumulative sum augmentation** [29] : This augmentation is defined to map the observed sequence $\mathbf{x}$ into its cumulative sum:

$$\Phi(\mathbf{x}) = \big(S_0, S_1, S_2, ..., S_n\big) \in \mathbb{R}^{d\times(n+1)},$$

where $S_t := \sum_{i=1}^{t} x_i, \forall t \in \{1,\cdots,n\}$ and $S_0 = \mathbf{0} \in \mathbb{R}^d$. Notice that we added $\mathbf{0}$ as the base point of S_t, which can introduce the sensitivity of the signature to the translation of the sequence.

- **Lead-lag augmentation** [29, 30, 31]: Lead-lag augmentation can be employed to capture the quadratic variation by transforming the sequence to

$$\Phi(\mathbf{x}) = \big((x_1,x_1),(x_2,x_1),(x_2,x_2),(x_3,x_2),(x_3,x_3),...,(x_n,x_n)\big) \in \mathbb{R}^{d\times 2\times n}.$$

It is easy to verify that the Lévy area of the augmented path $\Phi(\mathbf{x})$ is the quadratic variation of the path, i.e.

$$A_{\text{Lead-lag}} = \frac{1}{2}\sum_{i=1}^{n-1}(x_{i+1}-x_i)^2,$$

This information is encoded in the truncated signature of the augmented path at level 2 as the second term of the log-signature of the augmented path is simply the Lévy area and the signature and log-signature have one-to-one correspondence.

Combining cumulative sum and Lead-lag augmentation, the truncated signature of the augmented path at level 2, will give us information on the mean and variance of the sequence. To be specific, a bit of algebra can bring us:

$$\hat{\mathbb{E}}(X) = \frac{1}{n}S^{(1)},$$
$$\hat{\mathbb{E}}(X^2) = \frac{1}{n}S^{(1,2)} - \frac{1}{n}S^{(2,1)}.$$

Similarly, it is straightforward to demonstrate that higher-order statistical moments can be encoded into higher-order truncated signatures of the embedded path.

- **Lag-added augmentation**: The m-lag added transformation of sequence $\mathbf{x}$ is defined as follows:

$$\Phi(\mathbf{x}) = (y_1, y_2, \ldots, y_{n-m}) \in \mathbb{R}^{d\times(m+1)\times(n-m)},$$

  where $y_t = (x_t, \cdots, x_{t+m})$, $\forall t \in \{1, \ldots, n-m\}$. Similarly, by combining lag-added augmentation with cumulative sum and lead-lag augmentation, the signature of the augmented path can obtain information on the temporal dependency of the sequence.

We refer readers to [28, 29, 32] for a more precise definition of the above path augmentations. To use the signature of a path as the feature set to differentiate two measures on the path space, the only requirement for the way of embedding a discrete time series to a continuous path is that this embedding needs to ensure the bijection between the time series and its signature. For example, both time augmentation and lead-lag transformation can ensure the one-to-one correspondence between the time series and the signature. In numerical experiments, we choose the above augmentations such that the signature of the augmented path has *uniqueness* and *universality*, and it encodes useful information like statistical moments and temporal dependency. In our analysis in the following chapters, we only use the time-augmented path to embed the discrete time series X to a continuous path for ease of discussion. Also, we denote the space of augmented paths of finite p-variation by $\Omega_0^p(J,E)$ for ease of notation.

2.2 Generative adversarial network (GAN)

Generative adversarial network (GAN), designed by Ian Goodfellow et al. [3] in 2014, is a deep-learning-based generative model that learns the distribution of the target random variable. GAN is originally applied to generate high-quality images, but has been widely used for time-series data generation these years, eg. stock data [5][33] and medical data [34]. A brief introduction to the basics of neural networks can be found in Appendix B.

2.2.1 Classical generative adversarial network

In this subsection, we will introduce the classical generative adversarial network (GAN).

Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space, in the context of GAN, $\mathcal{Z}$ denotes the latent space, and $Z \in \mathcal{Z}$ denotes latent noise with the known distribution $\mu_Z \in \mathcal{P}(\mathcal{Z})$. $\mathcal{X}$ denotes the target space and $\mu \in \mathcal{P}(\mathcal{X})$ denotes a target distribution of observed data. The goal of GAN is to learn the target distribution μ.

To achieve this, GAN plays a min-max game between two networks: *Generator* and *Discriminator*. *Generator* $G^\theta : \mathcal{Z} \to \mathcal{X}$ is a parameterized map transporting the latent distribution μ_Z to the model distribution defined as the push-forward measure $\nu_\theta := G^\theta_\# \mu_Z(X) = \mu_Z((G^\theta)^{-1}(X))$ induced by G^θ. Here $\theta \in \Theta$ is the parameter set of the generator; *Discriminator* $D^\phi : \mathcal{X} \to \mathbb{R}$ is usually a parameterized map from the target space to the real space. As designed to discriminate between real data and synthetic data, $D^\phi(X)$ can be used to approximate the probability that X comes from the target distribution μ rather than model distribution ν_θ, and we refer the output of discriminator $D^\phi(X)$ as the *discriminator score* of X. Both generator and discriminator can be constructed by the neural networks like FNN or RNN introduced in Appendix B. Additionally, we assume the generator to be differentiable with respect to θ and the discriminator to be differentiable with respect to ϕ and x.

Both generator and discriminator are optimized to solve the following min-max

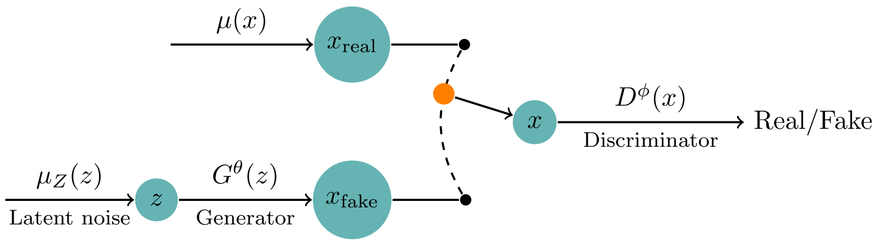


Figure 2.1: A simple representation of a generative adversarial network (GAN) architecture

game as in the original GAN algorithm [35]:

$$\min_{\theta}\max_{\phi} V(G^\theta, D^\phi) = \mathbb{E}_\mu\left[\log(D^\phi(X))\right] + \mathbb{E}_{\nu_\theta}\left[\log(1 - D^\phi(X))\right], \tag{2.5}$$

where $D^\phi : \mathcal{X} \to (0,1)$.

In this min-max game, the discriminator is trained to maximize the binary cross-entropy (BCE) loss that measures the distance between the real measure μ and fake measure ν_θ, and the generator is trained to minimize the BCE such that the generated samples can fool out the discriminator. If we assume both μ and ν_θ are continuous with densities p_μ and p_{ν_θ}, then by solving the maximization problem for ϕ, we can actually derive the optimal discriminator as

$$D^{\phi^*}(x) = \frac{p_\mu(x)}{p_\mu(x) + p_{\nu_\theta}(x)}. \tag{2.6}$$

for $x \in \mathcal{X}$. Given this optimal discriminator, the training objective (2.5) can be interpreted as minimizing *Jensen-Shannon divergence* between μ and ν_θ, shifted by a constant term $\log(4)$:

$$V(G^\theta, D^{\phi^*}) = \mathrm{JSD}(\mu\|\nu_\theta) - \log(4) = \mathrm{KL}(\mu\|\bar{\nu}) + \mathrm{KL}(\nu_\theta\|\bar{\nu}) - \log(4), \tag{2.7}$$

where $\bar{\nu} = \frac{\mu+\nu}{2}$ and $\mathrm{KL}(\cdot|\cdot)$ denotes the Kullback-Leibler divergence. Both JS and KL divergence are methods of measuring the similarity between two probability distributions. They are non-negative and equal to 0 if and only if two distributions coincide.

Nevertheless, it is also well-known that GAN objective (2.5) suffers from the saturating (or gradient vanishing) problem, meaning when the discriminator is sufficiently accurate, it can cause the gradient of the generator to vanish (close to 0), which makes the updates to the generator consistently worse. This saturation problem can be solved via a new objective function for the generator. Notice that the second term in (2.5) is independent of μ, and minimizing the second term in (2.5) is the same as maximizing the logits of the generated sample, hence an equivalent objective can be derived as

$$\begin{aligned} &\max_{\phi} \mathbb{E}_{\mu}\left[\log(D^{\phi}(X))\right] + \mathbb{E}_{\nu_\theta}\left[\log(1-D^{\phi}(X))\right], \\ &\min_{\theta} \mathbb{E}_{\nu_\theta}\left[-\log(D^{\phi}(X))\right]. \end{aligned} \tag{2.8}$$

This new objective function is proven in [36] to overcome the saturation problem, and hence named as *Non-saturating loss*.

For the convenience of discussion, we follow the notation in [37] to rewrite the GAN objective (2.8) in a more generic form. The training objective involves optimizing discriminative loss L_D and generative loss L_G can be rewritten as:

$$\begin{aligned} &\max_{\phi} L_D(\phi;\theta) = \mathbb{E}_{\mu}\left[f_1(D^{\phi}(X))\right] + \mathbb{E}_{\nu_\theta}\left[f_2(D^{\phi}(X))\right], \\ &\min_{\theta} L_G(\theta;\phi) = \mathbb{E}_{\nu_\theta}\left[h(D^{\phi}(X))\right], \end{aligned} \tag{2.9}$$

where f_1, f_2, and h are real-valued functions that can be chosen depending on the adversarial loss (or metric) we use. To retrieve the objective in (2.8), we can set $f_1(x) = -h(x) = \log(x)$ and $f_2(x) = \log(1-x)$.

Remark 2.2.1. In practical implementation, instead of training both discriminator and generator at the same time, we alternatively train discriminator and generator by computing their gradients and updating their respective parameters. To get a close approximation of the optimal discriminator, it is common to compute the discriminator's gradient multiple times and ascent the parameter ϕ. This training rule applies to most of GAN variants and those presented in our paper.

Due to the instability and unsatisfactory performance of (2.5), numerous studies have been conducted to explore different approaches to quantify the divergence or distance between μ and ν_θ, which has been a source of improved loss functions that stabilize the training and make high-fidelity fake samples [38, 39, 40, 41]. For example, A widely used adversarial loss is called *hinge loss* [42], where $f_1(w) = f_2(-w) = -\max(0, 1-w)$, and $h(w) = -w$. It has a geometric interpretation that generator parameters update along the normal vector direction of the separating hyperplane learned by the discriminator. A recent study has shown that combined with *spectral normalization* of weights in discriminator [14], hinge loss greatly improves generation performance and has become a mainstay in recent state-of-the-art GANs like SN-GAN and BigGAN [43]. Apart from that, WGAN [44] is also an innovative improvement to traditional GANs which we will introduce in detail in the following subsection.

2.2.2 Wasserstein generative adversarial network (WGAN)

The discontinuity of the Jensen-Shannon divergence is usually regarded as the source of instability in GAN training. To tackle this problem and inspired by the *Wasserstein-1 (Earth-Mover)* distance, Martin Arjovsky et al. [44] designed *Wasserstein Generative Adversarial Network* (WGAN) to improve the stability of GAN. WGAN provides a loss function named Wasserstein-1 (W_1) metric which measures the distance between the target distribution μ and model distribution ν_θ.

To define the W_1 metric, we first introduce the Lipschitz norm of a functional $f : \Omega \to \mathbb{R}$, where Ω is a generic metric space. The Lipschitz norm of f denoted by $||f||_{Lip,\Omega}$ is defined as

$$||f||_{Lip,\Omega} := \sup_{x \neq y, x,y \in \Omega} \frac{|f(x) - f(y)|}{d(x,y)}, \tag{2.10}$$

where $d(\cdot,\cdot)$ is a metric defined on Ω.

The Kantorovich-Rubinstein dual representation of W_1 metric defines a distance between two measures μ and ν, denoted by $W_1(\mu,\nu)$ as follows:

$$W_1(\mu,\nu) = \sup_{||f||_{\mathrm{Lip}} \leq 1} \left(\mathbb{E}_\mu[f(X)] - \mathbb{E}_\nu[f(X)]\right), \tag{2.11}$$

where the supremum is taken over all the 1-Lipschitz functions $f : \Omega \to \mathbb{R}$ with its Lipschitz norm smaller than 1.

WGAN aims to train a model that induces distribution ν_θ such that $W_1(\mu, \nu_\theta)$ is small enough. Similar to GAN, the WGAN is also composed of two neural networks. The first one is generator $G^\theta : \mathcal{Z} \to \mathcal{X}$, a parameterized map transporting the latent distribution μ_Z to the model distribution ν_θ where $\theta \in \Theta$ is a parameter set. The second one is the test function f in the definition of W_1 metric (2.11) replaced by a neural network D^ϕ with the parameter set $\phi \in \Phi$. Training the generator entails solving the min-max problem. Indeed, to find optimal $(\theta^\star, \phi^\star)$ one needs to solve

$$\min_\theta \max_{|D^\phi|_{\text{Lip}} \leq 1} \left(\mathbb{E}_\mu[D^\phi(X)] - \mathbb{E}_{\nu_\theta}[D^\phi(X)] \right). \tag{2.12}$$

To enforce the Lipschitz constraint on function D^ϕ, Arjovsky et al. [44] propose to clip the weights of D^ϕ to lie within a compact space $[-1, 1]$. The set of functions satisfying this constraint is a subset of the 1-Lipschitz functions which depends on the architecture of D^ϕ. However, as pointed out in [45], weight clipping in WGAN might give rise to optimization difficulties. Furthermore, even if optimization succeeds, the critic obtained can have a pathological value surface. Gulrajani et al. [45] also proved that optimal test function D^ϕ has unit gradient norm everywhere under μ and ν_θ, and therefore propose an alternative to penalize the gradient norm of the test function with respect to the input. This algorithm is often referred to as WGAN-GP due to the existence of the gradient penalty term, and its training objective can be given as:

$$\begin{aligned} &\max_\phi L_D(\phi; \theta) = \mathbb{E}_\mu[D^\phi(X)] - \mathbb{E}_{\nu_\theta}[D^\phi(X)] - \lambda \mathbb{E}_{\hat{\mu}}[(\|\nabla_x D^\phi(X)\|_2 - 1)^2], \\ &\min_\theta L_G(\theta; \phi) = \mathbb{E}_{\nu_\theta}[-D^\phi(X)]. \end{aligned} \tag{2.13}$$

Here the measure $\hat{\mu}$ is defined as sampling along straight lines between pairs of points sampled from μ and ν. WGAN-GP is empirically shown to outperform WGAN and enable stable training of a wide range of neural network architectures.

2.2.3 Conditional generative adversarial network

Although the GAN model is capable of generating realistic samples, it has no control over the type of samples generated. *Conditional Generative Adversarial Network* [46] is a conditional version of the generative adversarial network which has recently become one of the most popular models in generative model research. Conditional GAN is of so much interest as it allows us to condition both the discriminator and generator with additional information like class labels and past paths such that we have better control of the generated samples. The goal of conditional GAN is to learn the conditional distribution of the target variable $X \in \mathcal{X}$ given any conditioning variable $Y \in \mathcal{Y}$. To this goal, both discriminator and generator can be naturally extended to incorporate the conditional information Y, i.e. $D : \Phi \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$. and $G : \Theta \times \mathcal{Z} \times \mathcal{Y} \to \mathcal{X}$ respectively, and both are trained to optimize the following objective function:

$$\max_{\phi} L_D(\phi;\theta) := \mathbb{E}_{\mu}\left[f_1(D^{\phi}(X,Y))\right] + \mathbb{E}_{\nu_\theta}\left[f_2(D^{\phi}(X,Y))\right],$$
$$\min_{\theta} L_G(\theta;\phi) := \mathbb{E}_{\nu_\theta}\left[h(D^{\phi}(X,Y))\right],$$

where f_1, f_2 and h are real-valued functions corresponding to different adversarial losses.

In recent conditional image generation research, cGAN can be roughly divided into two groups depending on the way it incorporates the class information: *Classification-based* and *Projection-based* cGAN. *Classification-based* cGAN [47, 48] facilitate an auxiliary classifier to feed in the class information. Apart from the original min-max objective, both the discriminator and generator are also trained to maximize the log-likelihood of the correct class. Although classification-based cGAN has shown satisfactory results, it suffers an early mode collapse problem and the training can become very unstable as the number of classes in the dataset increases. In this circumstance, Miyato et al. [49] proposed *Projection-based* cGAN that incorporates the class information by projecting the class label into the feature set. It is based on the observation that the optimal solution for the adversarial loss in the original GAN can be rewritten as the sum of two log-likelihood ratios. These two log-likelihood ratios are then modeled as two parametric functions, one of which is fed class information using an embedding matrix. For more information on conditional image generation task, we refer the interested readers to [47, 49, 46, 50, 43, 48].

In the field of conditional time-series generation research, most of the work feeds the conditional information into the discriminator by simply concatenating the conditional information along with the target variable. Hyland et al. [34] propose RCGAN that makes use of RNN to produce realistic medical time series. In RCGAN, the conditional information, class label in this case, is fed into RNN by concatenation at each time step. Lu et al. [51] use a gated recurrent unit (GRU) in the generator with a smooth conditional matrix to generate high-quality EHR data and uncommon disease. Ramponi et al. [52] propose TCGAN where both generator and discriminator are conditioned on the sampling timestamps of time series, which can be used as a data augmentation method for time series with irregular sampling.

Despite these advancements, little work has ever been done to condition the model on past time series and generate realistic future time series. This is our main focus that we will investigate in the subsequent chapters and serves as part of the innovation of this thesis.

2.2.4 Problem of GAN training

In practice, the min-max problem is solved by iterating gradient descent-ascent algorithms and its convergence can be studied using tools from game theory [11, 53]. Nonetheless, a considerable gap exists between theoretical results and actual performance for the training of GAN. It is well known that first-order methods that are typically used in practice to solve the min-max problem might not converge, even if the convex-concave case [54, 55, 56], which usually results in mode collapse, undamped oscillations in training dynamics. Consequently, the adversarial training is notoriously difficult to tune [10, 11], and generalization error is very sensitive to the choice of discriminator and hyper-parameters as it was demonstrated in large scale study in [12].

To combat this instability, much effort has been put into developing regularization methods: [57, 13] suggest penalizing the gradient norm of the discriminator so that the sensitivity of the discriminator to input is close to 0 as it should be at the Nash equilibrium; Miyao et al. [14] propose spectral normalization that constrains the spectral norm of each layer in the discriminator network to control its Lipschitz constant; Mescheder et al. [13] also suggest that adding Gaussian instance noise [58] to the input can make the training process converge and improve the stability; Zhou et al. [50] shows that weight decay [59] that penalizes the norm of weights of a neural network can improve the generalization of neural networks and hence prevent overfitting problems. However, weight decay can also deteriorate the model performance to some extent. And those regularization methods like gradient penalty are also hard to tune and meanwhile increase computational cost. The main purpose of this thesis is to stabilize the GAN training and improve the generation performance.

Chapter 3

Signature-based W_1 metric

As commented in Section 2.2.4, the min-max objective function of generative adversarial networks makes them notoriously difficult to tune. Furthermore, long time series hugely increase the dimension of target space, which makes GAN more difficult to train and struggle to capture the temporal dependence of joint probability distributions induced by time-series data. To overcome these challenges, we develop a signature-based W_1 metric that originates from the signature to characterize the measure induced by time series and allows turning computationally challenging GAN min-max problem into supervised learning while generating high-fidelity samples.

3.1 Sig-W_1 metric

As introduced in Section 2.1.3, the uniqueness and universality of the (expected) signature make it an efficient feature set to characterize the law of stochastic process. In this section, we combine the signature feature with the W_1 metric and propose a new Signature-based Wasserstein-1 (Sig-W_1) metric on the measures on the path space $\mathcal{X} = \Omega_0^1([0,T],\mathbb{R}^d)$ to achieve better computation efficiency.

Let μ and ν be two compactly supported measures on the path space $\Omega_0(J,E)$ endowed with 1-variation topology such that the corresponding induced measures on the signature space $\tilde{\mu}$ and $\tilde{\nu}$ respectively have a compact support $\mathcal{K} \subset \mathcal{S}(\Omega_0(J,E)) \subset T((E))$. Naturally one may consider applying the W_1 metric on the signature space to define a distance between the induced measures $\tilde{\mu}$ and $\tilde{\nu}$, which we denote by $W_1^{\mathrm{Sig}}(\mu,\nu)$ and define as

$$W_1^{\mathrm{Sig}}(\mu,\nu) := \quad W_1(\tilde{\mu},\tilde{\nu}) = \sup_{||f||_{Lip,\mathcal{K}}\leq 1} \mathbb{E}_{S\sim\tilde{\mu}}[f(S)] - \mathbb{E}_{S\sim\tilde{\nu}}[f(S)] \tag{3.1}$$

$$= \sup_{||f||_{Lip,\mathcal{K}}\leq 1} \mathbb{E}_{X\sim\mu}[f(S(X))] - \mathbb{E}_{X\sim\nu}[f(S(X))]. \tag{3.2}$$

The W_1 metric on the signature features space denoted by W_1^{Sig} has the advantage over W_1 metric on the path space that it significantly reduces the time dimension of the problem, especially for high-frequency data. However, the practical challenges of solving the min-max problem remain.

The universality of the signature Theorem 2.1.4 provides us with a way to reduce the min-max problem into analytic form. The supremum in (3.2) implies that there exists a sequence of $f_n : \mathcal{K} \to \mathbb{R}$ with bounded Lipschitz norm to attain the supremum $W_1^{\mathrm{Sig}}(\mu,\nu)$, i.e., $\forall\varepsilon > 0$, there exist a sequence of function $\{f_n\}_{n=1}^{\infty}$ and a integer N such that when $n > N$,

$$|\int_{\mathcal{K}} f_n(S)\mu(dS) - f_n(S)\nu(dS) - W_1^{\mathrm{Sig}}(\mu,\nu)| \leq \varepsilon.$$

For each f_n, the universality of the signature implies that $\forall\varepsilon > 0$, there also exists a linear functional $L_n : \mathcal{K} \to \mathbb{R}$ to approximate f_n uniformly, i.e.

$$|\int_{\mathcal{K}} f_n(S)\mu(dS) - f_n(S)\nu(dS) - \left(\int_{\mathcal{K}} L_n(S)\mu(dS) - L_n(S)\nu(dS))\right)| \leq 2\varepsilon.$$

As $L_n : \mathcal{K} \to \mathbb{R}$ is linear, there is a natural extension of L_n mapping from $T((E))$ to $\mathbb{R}$. This implies that the supremum over the linear functional with Lipschitz norm less than one is a good approximation to W_1^{Sig}.

Motivated by the above observation, we can restrict the admissible set of f in (3.2) to be linear functionals $L : \mathcal{K} \to \mathbb{R}$ to approximate W_1^{Sig}. Hence we propose the Sig-W_1 metric as below.

Definition 3.1.1 (Sig-W_1 metric)**.** For two measures μ,ν on the path space $\Omega_0(J,\mathbb{R}^d)$ such that their induced measures $\tilde{\mu}$ and $\tilde{\nu}$ respectively has a compact support $\mathcal{K} \subset \mathcal{S}(\Omega_0(J, \mathbb{R}^d))$, the Sig-$W_1$ metric between μ and ν is defined as

$$\text{Sig-}W_1(\mu, \nu) = \sup_{||L||_{Lip} \leq 1, L \text{ is a linear functional}} \left(\mathbb{E}_{S \sim \tilde{\mu}}[L(S)] - \mathbb{E}_{S \sim \tilde{\nu}}[L(S)] \right).$$

Remark 3.1.1. Despite the motivation of Sig-W_1 from the approximation of W_1^{Sig}, it is hard to establish the theoretical results on the link between these two metrics. The main difficulty comes from that the uniform approximation of the continuous function f by a linear map L on $\mathcal{K}$ does not guarantee the closeness of their Lipschitz norms. We conjecture that in general $W_1^{Sig}(\mu, \nu)$ is not equal to Sig-$W_1(\mu, \nu)$. However, it would be interesting but technically challenging to find out the sufficient conditions such that these two metrics coincide.

Dealing with the Lipschitz norm is usually intractable. However, thanks to the linear functional L, we can simplify the Lipschitz norm of L as follows:

$$||L||_{Lip} := \sup_{x \neq y, x, y \in T((E))} \frac{|L(x-y)|}{D(x,y)} = \sup_{x \neq y, x, y \in T((E))} \frac{|L(x-y)|}{||x-y||_p} = \sup_{||a||_p = 1} |La|,$$

where we specify $D(\cdot, \cdot)$ as the l^p distance, i.e. $D(x,y) = ||x-y||_p$ for $p \geq 1$. In order to eliminate the supremum and derive the analytic formula for the Sig-W_1 metric, we introduce the following lemma.

Lemma 3.1.1. For any $p, q > 1$ such that $\frac{1}{p} + \frac{1}{q} = 1$, and any linear functional $L \in T((E))^*$, it holds that

$$\sup_{||a||_p = 1} |La| = ||L||_q, \tag{3.3}$$

Similarly, for any $a \in T((E))$, it holds that

$$\sup_{||L||_q \leq 1} |La| = \sup_{||L||_q = 1} |La| = ||a||_p. \tag{3.4}$$

The l^p norm of $T((E))$ and l^q norm of $T((E))^*$ are defined in the usual sense, which can found in Section 2.1.2.

Proof. Let $(e_I = e_{i_1} \otimes \cdots e_{i_n})_I$ be the canonical basis of $T((E))$. For any $a \in T((E))$, we write $\mathbf{a} = (a_I)$, i.e. $a = \sum_I a_I e_I$. Then $\left(e_I^* = e_{i_1}^* \otimes \cdots e_{i_n}^*\right)_{I=(i_1,\cdots,i_n)}$ is the basis of $T((E))^*$ and we can write $L = \sum_I l_I e_I^*$.

To prove Eq. (3.3), we apply the Lagrange multiplier method to solve the constraint optimization of maximizing La with the constraint $||\mathbf{a}||_p = 1$. More specifically, we solve the following unconstrained optimization

$$\sup_{a,\lambda} \mathcal{L}(\mathbf{a},\lambda) := \sup_{\mathbf{a},\lambda} |La| + \lambda \left(\sum_I |a_I|^p - 1 \right),$$

where $L \neq 0$.By the first order condition, the optimal (a^*, λ^*) satisfy the below equations:

$$\frac{\partial \mathcal{L}}{\partial a_I} = (\text{sign}(a_I l_I) l_I + (\lambda (p|a_I|^{p-1} \text{sign}(a_I)))) = 0, \forall I$$
$$\frac{\partial \mathcal{L}}{\partial \lambda} = \sum_I |a_I|^p = 1.$$

Then we obtain that $\mathbf{a}^* = (a_I^*)_I$ with $a_I^* = \text{sign}(l_I) \frac{|l_I|^{\frac{1}{p-1}}}{(\sum_I |l_I|^{p/(p-1)})^{1/p}} = \text{sign}(l_I) \frac{|l_I|^{\frac{1}{p-1}}}{(\sum_I |l_I|^q)^{1/p}}$. Then it follows that

$$l_I a_I^* = |l_I| \cdot \frac{|l_I|^{\frac{1}{p-1}}}{(\sum_I |l_I|^q)^{1/p}} = \frac{|l_I|^q}{(\sum_I |l_I|^q)^{1/p}} \geq 0;$$

Hence,

$$|L\mathbf{a}^*| = L\mathbf{a}^* = \sum_I l_I a_I^* = \frac{\sum_I |l_I|^q}{(\sum_I |l_I|^q)^{1/p}} = (\sum_I |l_I|^q)^{1-1/p} = (\sum_I |l_I|^q)^{1/q} = ||L||_q.$$

By Hôlder's inequality,

$$\sup_{||a||_p=1} La \leq \sup_{||a||_p=1} ||a||_p ||L||_q = ||L||_q,$$

and the supremum $||L||_q$ is obtained when $a = a^*$. We complete the proof of Eq. (3.3).

The proof of Eq. (3.4) is similar to the above. We only need to show the supremum taken over $||L|| = 1$ is the same as that $||L|| \leq 1$. Similarly to the above, when $L^* := (l_I^*)$ with

$$l_I^* = \operatorname{sign}(a_I)\frac{|a_I|^{\frac{1}{p-1}}}{(\sum_I |a_I|^q)^{1/p}}, \tag{3.5}$$

$L^*(a)$ attains the supremum $\sup_{||L||_q=1} L(a) = ||a||_p$ and $||L^*||_q = 1$. By Hôlder's inequality,

$$\sup_{||L||_q \leq 1} La \leq \sup_{||L||_q \leq 1} ||a||_p ||L||_q \leq ||a||_p. \tag{3.6}$$

As $\sup_{||L||_q \leq 1} La$ can not exceed $||a||_p$ and $L^*(a) = ||a||_p$, it follows

$$\sup_{||L||_q \leq 1} L(a) = ||a||_p.$$

□

By exploiting the linearity of the functional $L : T((E)) \to \mathbb{R}$, we can compute the Lipschitz norm of L analytically for D is the l^p norm of $T((E))$ without the need of numerical optimization. By Lemma 3.1.1, the Lipschitz norm of L is l^p norm of L that can be given as

$$||L||_{Lip} := \sup_{x \neq y, x, y \in T((E))} \frac{|L(x-y)|}{||x-y||_p} = \sup_{||a||_p = 1} |La| = ||L||_q,$$

where $\frac{1}{p} + \frac{1}{q} = 1$ and $D(x,y) = ||x-y||_p$ with some $p \geq 1$.

The simplification of the Lipschitz norm enables us to derive an analytic formula for the corresponding Sig-W_1 metric. Remember that the Lipschitz norm of L in the Sig-W_1 metric is restricted to be less than one. By the simplification of the Lipschitz norm mentioned above, it is equivalent to restricting $||L||_q$ to be less than one. Combining this observation with Lemma 3.1.1, we are able to derive an analytic formula for the Sig-W_1 metric.

Lemma 3.1.2. For two measures μ, ν on the path space $\Omega_0(J,\mathbb{R}^d))$ such that their induced measures $\tilde{\mu}$ and $\tilde{\nu}$ respectively has a compact support $\mathcal{K} \subset \mathcal{S}(\Omega_0(J,\mathbb{R}^d))$. Then it holds that

$$\text{Sig-}W_1(\mu,\nu) = ||\mathbb{E}_{S\sim\tilde{\mu}}[S] - \mathbb{E}_{S\sim\tilde{\nu}}[S]||_p = ||\mathbb{E}_{X\sim\mu}[S(X)] - \mathbb{E}_{X\sim\nu}[S(X)]||_p. \quad (3.7)$$

Proof. Let a linear functional $L : \mathcal{K} \to \mathbb{R}$ endowed with the Lipschitz norm when $D(x,y) = ||x-y||_p$, which is the same as L_q norm. Let $a := (\mathbb{E}_{S\sim\tilde{\mu}}(S) - \mathbb{E}_{S\sim\tilde{\nu}}(X))$ and $a = (a_I)_I$. Then by Lemma 3.1.1, one derive the analytic formula of Sig-W_1 metric as follows:

$$\text{Sig-}W_1(\mu,\nu) = \sup_{||L||_q\leq 1} L(\mathbb{E}_{S\sim\tilde{\mu}}(S)) - L(\mathbb{E}_{S\sim\tilde{\nu}}(S)) = \sup_{||L||_q\leq 1} L(a) = ||a||_p,$$

where $L = \sum_I l_I e_I^*$. □

By Lemma 3.1.2, we obtain the analytic formula for the Sig-W_1 metric. This Sig-W_1 metric is a generalization of the one proposed in [19] by considering the general l^p metric of the signature space.

In practice, one needs to truncate the infinite-dimensional signature to a finite degree for numerical computation of Sig-$W_1(\mu,\nu)$. The factorial decay of the signature enables us to approximate the signature in (3.7) by its truncated signature up to degree M for a sufficiently large M. Therefore we propose to define the truncated Sig-$W_1(\mu,\nu)$ metric up to a degree M as follows:

$$\text{Sig-W}_1^{(M)}(\mu,\nu) := ||\mathbb{E}_{X\sim\mu}[S_M(X)] - \mathbb{E}_{X\sim\nu}[S_M(X)]||_p. \quad (3.8)$$

The following toy example illustrates the relationship between the Sig-W_1 distance and the W_1 distance between two path distributions.

Example 3.1.1. Let $X = (X_t)_{t\in[0,T]}$,$\hat{X} = (\hat{X}_t)_{t\in[0,T]}$ be two 1-dimensional GBMs given by,

$$dX_t = \theta_1 X_t dt + \sigma X_t dW_t,\ X_0 = 1;$$
$$d\hat{X}_t = \theta_2 \hat{X}_t dt + \sigma \hat{X}_t d\hat{W}_t,\ \hat{X}_0 = 1,$$

with the same volatility but with possibly different drifts θ_1, θ_2. Let μ, ν be the laws of $X, \hat{X}$. We fix $\sigma = 0.1, \theta_1 = 0.02$, and for $\theta_2 = 0.02 + 0.025j, j = 0, \ldots, 4$. When θ_2 is increasing, the discrepancy between X and $\hat{X}$ is increasing. We calculate three distances, i.e. $W_1^{\text{path space}}$, $W_1^{\text{Sig space}}$ and Sig-W_1 to quantify the distance between X and $\hat{X}$ for different θ_2, which all increase when enlarging θ_2 as expected. Since Sig-$W_1^{(M)}(\mu, \nu)$ admits an analytic solution, it is cheaper to calculate than $W_1^{\text{path space}}(\mu, \nu)$ and $W_1^{\text{Sig space}}(\mu, \nu)$, where one needs to parameterize f by a neural network and optimize its weights. We observe in Figure 3.1 how these three values increase with a similar rate as θ_2 increases.

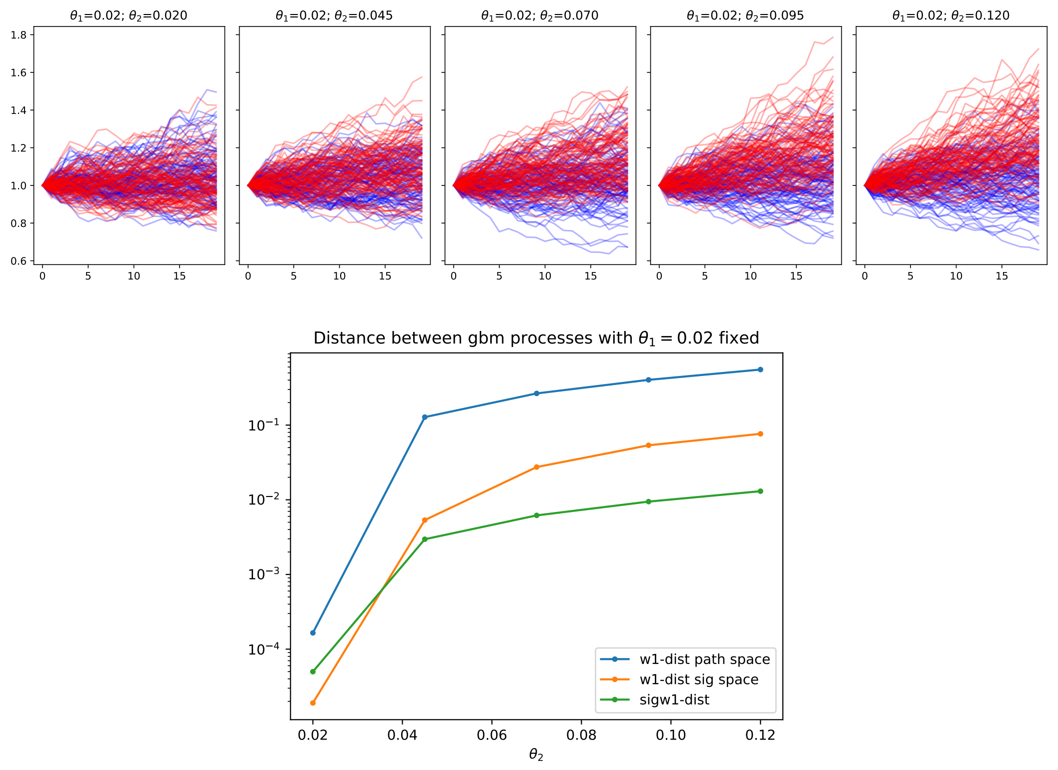


Figure 3.1: The top row displays blue and red samples from two distributions μ and ν respectively for fixed θ_1, and different values of θ_2.

3.2 Conditional Sig-W_1 metric

In reality, one might be interested in predicting the future given the past information, for instance, given current (or past) stock price, investors may want to predict the stock price in the future. In the last section, we propose the Sig-W_1 metric to measure the distance between two measures μ and ν on the path space. However, the Sig-W_1 metric does not rely on any past path information. In this section, we generalize the Sig-W_1 metric to the conditional setting where we condition explicitly on the past information based on the Sig-W_1 metric.

Let J_1, J_2 be compact and disjoint time intervals such that $J = J_1 \cup J_2$, and $t_1 < t_2, \forall t_1 \in J_1, \forall t_2 \in J_2$. Throughout this section we assume that X is a $\Omega_0^1(J;\mathbb{R}^d)$-valued random variable and denote by X_{J_i} the stochastic process constrained to the interval J_i, for $i = 1,2$.

Given a past path $y \in \Omega_0^1(J_1;\mathbb{R}^d)$, we would like to generalize (3.7) to quantify a distance of the conditional measures $\mu(x) = \mu(X_{J_2} \mid X_{J_1} = y)$ and $\nu(x) = \nu(X_{J_2} \mid X_{J_1} = y)$. This can be done by changing the expectations in (3.7) to conditional expectations where the condition is the conditioning path $x \in \Omega_0^1(J_1;\mathbb{R}^d)$. We define the conditional Sig-W_1 metric as the the Sig-W_1 metric between two conditonal measures $\mu(y)$ and $\nu(y)$, i.e.

$$\text{cSig-W}_1(\mu,\nu,y) := \text{Sig-W}_1(\mu(y),\nu(y)) = \|\mathbb{E}_{\mu(y)}[S(X_{J_2})] - \mathbb{E}_{\nu(y)}[S(\hat{X}_{J_2})]\|_p \tag{3.9}$$

Similar to the unconditional case, the cSig-W_1 metric has the advantage of approximating conditional W_1 metric on the signature space effectively without any optimization and it preserves the discriminative ability to distinguish different conditional distributions.

Theorem 3.2.1 (Characteristic property of cSig-W_1 metric)**.** Suppose $\mu(y)$ and $\nu(y)$ be two conditional measures, under which the expected signature of X_{future} given

$X_{\text{past}} = y$ has infinite radius of convergence respectively for some $y \in \mathbb{R}^{d \times p}$. Then

$$\text{cSig-W}_1(\mu, \nu, y) = 0 \Longleftrightarrow \mu(y) \overset{d}{=} \nu(y)$$

Proof. This is a direct consequence of the characteristic property of the expected signature [27] for the conditional law case. □

The definition of infinite radius of convergence of expected signature can be found in Definition A.2.1. However, it is challenging to establish a general condition to guarantee the infinite radius of convergence (ROC). In fact, the study of the expected signature of stochastic processes is an active area of research. For example, the expected signature of fractional Brownian motion for the Hurst parameter $H \geq 1/2$ is shown to have the infinite ROC [60, 61], whereas the ROC of the expected signature of stopped Brownian motion up to the first exit domain is finite [62]. Theorem 6.3 in [27] provides a sufficient condition for the infinite ROC of the expected signature, potentially offering an alternative way to show the infinite ROC without directly examining the decay rate of the expected signature.

In practice, it is usually intractable to compute the cSig-W_1 metric between real measure $\mu(y)$ and fake measure $\nu(y)$. That is due to the fact that most of the time series we can get access to in reality consist of one long trajectory, such as historical stock price. In this case, each past path is associated with one single future path, and it is hence impossible to compute the conditional expected signature by averaging the signature of future paths given one past path. This limitation is also one of the most challenging problems in the conditional time-series generation area. In the following chapter, we address this challenge by proposing a novel conditional time-series generation framework that is able to estimate the conditional expected signature in an efficient way.

Chapter 4

Signature-based conditional WGAN

In Chapter 3, we propose the cSig-W_1 metric that can measure the distance between two conditional measures $\mu(y)$ and $\nu(y)$, and no optimization is involved. Now we are ready to present our signature-based conditional WGAN (SigCWGAN) that can learn the target conditional distribution induced by the time series and generate high-fidelity samples based on past information. More importantly, by utilizing the analytic form of the Sig-W_1 metric, it reduces the min-max problem of the conventional GAN framework into supervised learning tasks.

Let $(\Omega, \mathcal{F}, (\mathcal{F}_t)_{t\geq 0}, \mathbb{P})$ be a filtered probability space, and $\{X_t\}_{t\in\mathbb{N}}$ a stationary stochastic process that takes value in $\mathbb{R}^d$. We assume that the conditional distribution of X_{t+1} given the filtration $\mathcal{F}_t$, only depends on the p-lagged values of X_t which we denote by $X_{\text{past},t} := (X_{t-p+1}, \cdots, X_t) \in \mathbb{R}^{d\times p} := \mathcal{Y}$. We are interested in generating samples from the distribution of $X_{\text{future},t} := (X_{t+1}, \cdots, X_{t+q}) \in \mathbb{R}^{d\times q} := \mathcal{X}$ for any given q, conditioned on p-lagged value $X_{\text{past},t}$ which we denote by $\mu_{p,q}(y) := \text{Law}(X_{\text{future},t} | X_{\text{past},t} = y)$. Here that the conditional measure $\mu_{p,q}(y)$ is time-homogeneous is a direct consequence of the stationarity of X.

Suppose that we have access to a long realization $(x_1, \dots, x_T) \in \mathbb{R}^{d\times T}$ of X, then we are able to get N number of past / future paths samples pairs $(X_{\text{past}}, X_{\text{future}})$ in rolling window fashion, i.e.

$$(x^{(i)}_{\text{past}}, x^{(i)}_{\text{future}}) = (x_{i:i+p-1}, x_{i+p:i+p+q-1}),$$

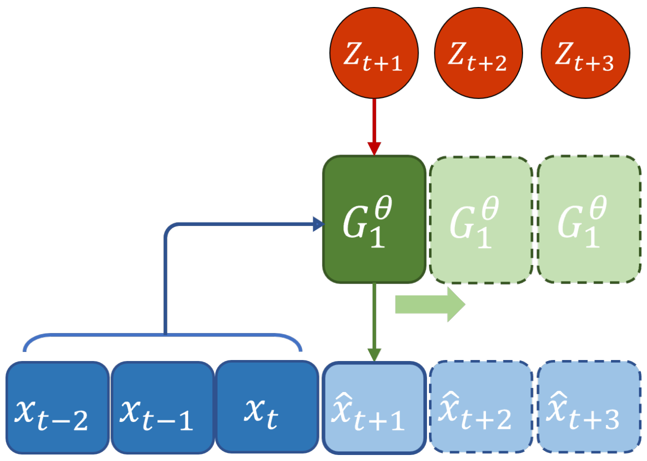


Figure 4.1: Illustration of AR-FNN generator.

where $i \in \{1,\dots,N\}$ and $N = T - q - p + 1$. The stationarity of X implies that for every $i \in \{1,\dots,N\}$, past / future sample pairs $(x^{(i)}_{\text{past}}, x^{(i)}_{\text{future}})$ is identically distributed, i.e.

$$\left(x^{(i)}_{\text{past}}, x^{(i)}_{\text{future}}\right) \overset{d}{\sim} \left(X_{\text{past}}, X_{\text{future}}\right).$$

And $x^{(i)}_{\text{future}}$ is a sample drawn from the conditional distribution $\mu_{p,q}(x^{(i)}_{\text{past}})$. The goal of our conditional generative model is to learn the conditional measure $\mu_{p,q}(X_{\text{past}})$ induced by the past/future sample pairs $(x^{(i)}_{\text{past}}, x^{(i)}_{\text{future}})_{i=1}^{N}$.

4.1 Conditional AR-FNN generator

We start by specifying the generator. Motivated by the auto-regressive type models in time series literature, we assume that the $\mathbb{R}^d$-valued stochastic process $\{X_t\}_{t=1}^{T}$ satisfies

$$X_{t+1} = g(X_{\text{past},t}, \varepsilon_{t+1})$$

where $(\varepsilon_t)_t$ are i.i.d. random variables and $E\left[\varepsilon_{t+1}|\mathcal{F}_t\right] = 0$ and $g : \mathcal{Y} \times \mathcal{Z} \to \mathbb{R}^d$ is a continuous but unknown function. Time series of such kind include the autoregressive model (AR) and the Autoregressive conditional heteroskedasticity (ARCH) model. The objective of SigCWGAN is to generate the joint distribution of the future time series $X_{\text{future},t}$ given the past time series $X_{\text{past},t}$.

The proposed conditional generator is designed to capture the autoregressive structure of the target time series by using the past path $X_{\text{past},t}$ as additional input for the generator. It is defined as a parameterized map $G^\theta : \mathcal{Y} \times \mathcal{Z}^q \to \mathcal{X}$ with Θ being a parameter space that takes the latent noise $Z := (Z_{t+1}, \ldots, Z_{t+q}) \in \mathcal{Z}^q$ and the past path $X_{\text{past},t} \in \mathcal{Y}$ to predict q-step future path $\hat{X}_{\text{future},t} \in \mathcal{X}$ so that the conditional distribution of $\hat{X}_{\text{future},t}$ is as close as possible to $\mu(X_{\text{future},t} | X_{\text{past},t} = y)$.

We first consider a step-1 conditional generator $G_1^\theta(\cdot,\cdot) : \mathcal{Y} \times \mathcal{Z} \to \mathbb{R}^d$, which takes the past path x and the noise vector Z_{t+1} to generate a random variable to mimic the conditional distribution of step-1 forecast $\mu(X_{t+1} | X_{\text{past},t} = y)$. Here the noise vector Z_{t+1} has the standard normal distribution in $\mathcal{Z} = \mathbb{R}^{d_Z}$, where d_Z is the dimension of the latent noise.

One can generate the future time series of arbitrary length $q \geq 1$ given x_{past} by applying $G_1^\theta(\cdot,\cdot)$ in a rolling window fashion with i.i.d. noise vector $(Z_t)_t$ as follows. Given $x_{\text{past}} = (\mathrm{x}_1, \cdots, \mathrm{x}_p) \in \mathbb{R}^{d \times p}$, we define time series $(\hat{x}_t)_t$ inductively: we first initialize the first p term $\hat{x}$ as x_{past}, and then for $t > p$, use $G_1^\theta(\cdot,\cdot)$ with the p-lagged value of $\hat{x}_t$ conditioning variable and the noise Z_t to generate $\hat{x}_{t+1}$, in formula,

$$\hat{x}_t = \begin{cases} \mathrm{x}_t, & \text{if } t \leq p; \\ G_1^\theta(\underbrace{\hat{x}_{t-p}, \cdots, \hat{x}_{t-1}}_{p \text{ lagged values of } \hat{x}_t}, Z_t), & \text{if } t > p, \end{cases} \tag{4.1}$$

where the function $G_1 : \Theta \times \mathcal{Y} \times \mathcal{Z} \to \mathbb{R}^d$ is represented by a feed-forward neural network with residual connections and parametric ReLUs as activation functions. The detailed description of residual connection and ReLU activation function can be found in Appendix B.1

Therefore, we obtain the step-q conditional generator, denoted by $G_q^\theta(\cdot,\cdot) : \mathcal{Y} \times \mathcal{Z}^q \to \mathbb{R}^{d \times q}$ and defined by $x_{\text{past}} \mapsto (\hat{x}_{p+1}, \cdots, \hat{x}_{p+q})$, where $(\hat{x}_{p+1}, \cdots, \hat{x}_{p+q})$ is defined in (4.1). We omit q in G_q^θ for simplicity. See Algorithm 1 for the pseudo-code of generating the next q-step forecast using G^θ. Hence we name the generator G^θ as a conditional AR-FNN generator due to its auto-regressive nature and the conditional law induced by $\mu_z \in \mathcal{P}(\mathcal{Z})$ is defined as $\nu_{p,q}^\theta(y) = \text{Law}(\hat{X}_{\text{future}} | X_{\text{past}} = y)$.

Given a collection of past / future sample pairs $(x^{(i)}_{\text{past}}, x^{(i)}_{\text{future}})_{i=1}^{N}$ that induces the conditional measure $\mu_{p,q}(y)$, the our aim is to find the optimal parameter θ such that $\nu^{\theta}_{p,q}(y)$ is as close as possible to the real measure $\mu_{p,q}(y)$ with respect to a proper metric. For simplicity, we just denote $\nu_{\theta}(y)$ and $\mu(y)$ respectively for the rest of this thesis. We also provide a detailed description of the AR-FNN network in Appendix B.1.

Algorithm 1 Pseudocode of Generating the next q-step forecast using G^{θ}

```
    Input: $x_{t-p+1:t}, G^{\theta}_{1}$
    Output: $\hat{x}_{t+1:t+q}$
1: $\hat{x}_{\text{future}} \leftarrow$ a matrix of zeros of dimension $d \times q$.
2: $\hat{x} \leftarrow$the concatenation of $x_{t-p+1:t}$ and $\hat{x}_{\text{future}}$.
3: for $i = 1 : q$ do
4:     We sample $Z_i$ from the iid standard normal distribution.
5:     $\hat{x}_{t+i} = G^{\theta}_{1}(\hat{x}_{t+i-p:t+i-1}, Z_i)$.
6: end for
7: return $\hat{x}_{t+1:t+q}$
```

4.2 Signature-based loss function

To train the conditional generator G^{θ}, we need a proper loss function. Our proposed SigCWGAN incorporates conditional Sig-W_1 metric (3.9) as the loss function that is able to measure the distance between two conditional distribution $\mu(y)$ and $\nu(y)$ and hence quantify the goodness of the conditional generator G^{θ}. More importantly, computing the conditional Sig-W_1 metric doesn't involve any optimization, which hence makes the algorithm more efficient. Based on the conditional Sig-W_1 metric the loss function is defined as follows:

$$\begin{aligned} L(\theta|(x^{(i)}_{\text{past}}, x^{(i)}_{\text{future}})_{i=1}^{N}) &= \mathbb{E}_{X_{\text{past}} \sim \mu}\left[\text{cSig-W}_1(\mu(X_{\text{past}}), \nu_{\theta}(X_{\text{past}}))\right] \\ &= \mathbb{E}_{X_{\text{past}} \sim \mu}\left\|\mathbb{E}_{\mu(X_{\text{past}})}[S(X_{\text{future}})] - \mathbb{E}_{\nu_{\theta}(X_{\text{past}})}[S(X_{\text{future}})]\right\|_2 \end{aligned} \tag{4.2}$$

4.2.1 Learning the conditional expected signature under true measure

Notice that, in order to compute the signature-based loss (4.2), we need to estimate the conditional expected signature under the true measure from data. This can be done via Monte Carlo simulation if, for every single past path, we have a larger number of corresponding future path samples. However, it is infeasible in practice. For empirical datasets like stock price data, we only have access to the time series of a long history where one past path corresponds to one single future path.

As a result, we need to estimate the conditional expected signature under the true measure from data. This estimation problem can be viewed as a linear regression task, where the regressor and response variable are the signature of the past path and future path respectively [26].

Hence we propose the linear models on the truncated signature to learn the conditional expected signature under the empirical true measure. We apply an ordinary least squares regression (OLS) to regress signatures of future paths on signatures of past paths.

More specifically, given a long realization of $x := (x_1, \cdots, x_T) \in \mathbb{R}^{d\times T}$ and fixed window size of the past and future path $p, q > 0$, we construct the samples of past / future path pairs $(X_{\text{past}}, X_{\text{future}})$ in a rolling window fashion, where the i^{th} sample is given by

$$\left(x^{(i)}_{\text{past}}, x^{(i)}_{\text{future}}\right) = \left(x_{i:i+p-1}, x_{i+p:i+p+q-1}\right).$$

Assuming stationarity and memorylessness property of the time series, we ensure that the path outside the window has no impact on the conditional distribution, and past/future sample pairs $(x^{(i)}_{\text{past}}, x^{(i)}_{\text{future}})_{i=1}^{N}$ are identically distributed, and we are also able to compute the samples pairs of the signature of the past / future path that is identically distributed,

$$\left(S_{L_1}(x^{(i)}_{\text{past}}), S_{L_2}(x^{(i)}_{\text{future}})\right) \overset{d}{\sim} \left(S_{L_1}(X_{\text{past}}), S_{L_2}(X_{\text{future}})\right).$$

Here, L_1 and L_2 are the truncated degree of the signature of the past path and future path respectively, which can be chosen by cross-validation in terms of fitting result. One may refer to [63] for further discussion on the choice of the degree of the signature truncation.

In principle, linear regression methods on the signature space could be applied to solve this problem using the data constructed above. If we assume that

$$S_{L_2}(X^{(i)}_{\text{future}}) = L(S_{L_1}(X^{(i)}_{\text{past}})) + \varepsilon_i, \tag{4.3}$$

where the noise $\varepsilon_i \overset{iid}{\sim} \varepsilon$ and $\mathbb{E}[\varepsilon_i | X^{(i)}_{\text{past}}] = 0$, then an ordinary least squares regression (OLS) can be applied by using the above samples pairs of the signature of the past / future path.

Remark 4.2.1. In general, the assumption we made above might not be satisfied, in which case the OLS could be potentially replaced by other sophisticated regression models. This simple linear regression model on the signature space achieves satisfactory results on the numerical examples of this thesis.

The design matrix of the OLS regression is hence constructed as

$$\mathbf{X} = \left(S_{L_1}(x^{(1)}_{\text{past}}), \dots, S_{L_1}(x^{(N)}_{\text{past}})\right)^T \in \mathbb{R}^{N \times F(d,L_1)}.$$

where $F(d,L) = \sum_{k=0}^{L}(d+1)^k$ denotes the number of signature terms that are computed when truncating at order L and considering an d-dimensional path in $\Omega_0^1(J,\mathbb{R}^d)$. Furthermore, the response matrix is defined as

$$\mathbf{Y} = \left(S_{L_2}(x^{(1)}_{\text{future}}), \dots, S_{L_2}(x^{(N)}_{\text{future}})\right) \in \mathbb{R}^{N \times F(d,L_2)},$$

where L_2 is the truncated degree of the signature of the future path. Setting up the OLS regression problem we define the OLS regressor as

$$\hat{L}_\mu := \underset{L \in \mathcal{L}(T^{(L_1)}(\mathbb{R}^{d+1}), T^{(L_2)}(\mathbb{R}^{d+1}))}{\text{argmin}} \|L\mathbf{X} - \mathbf{Y}\|_2.$$

Then the conditional expected signature under the true measure $\mathbb{E}_{\mu(X_{\text{past}})}[S_{L_2}(X_{\text{future}})]$ can be approximated by $\hat{L}_\mu(S_{L_1}(X_{\text{past}}))$. We emphasize that the estimation of $\mathbb{E}_{\mu(X_{\text{past}})}[S_{L_2}(X_{\text{future}})]$ is one-off and can be done prior to the generative learning. It is in striking contrast to the conditional WGAN learning, which requires learning $\mathbb{E}_{\mu(X_{\text{past}})}[D^\phi(X_{\text{future}}, X_{\text{past}})]$ every time the discriminator D^ϕ is updated, and hence saves significant computational cost.

On the other hand, given the conditional generator G^θ, the fake conditional expected signature $\mathbb{E}_{\nu_\theta(X_{\text{past}})}[S_{L_2}(\hat{X}_{\text{future}})]$ can be estimated by Monte Carlo method. Hence the empirical loss function can be given as follows:

$$L\left(\theta \middle| \left(x_{\text{past}}^{(i)}, x_{\text{future}}^{(i)}\right)_{i=1}^{N}\right) = \frac{1}{N}\sum_{i=1}^{N} ||\hat{L}_\mu(S_{L_1}(x_{\text{past}}^{(i)}))) - \mathbb{E}_{\hat{\nu}_i}[S_{L_2}(X_{\text{future}})]||_2 \tag{4.4}$$

where $\hat{\nu}_i$ is the approximation to the fake conditional measure $\nu_\theta(x_{\text{past}}^{(i)})$, which is computed via Monte Carlo method.

Remark 4.2.2. All the discrete time series involved should be embedded into path space via linear interpolation and time augmentation so that we are able to compute their signature and ensure its uniqueness (by Lemma 2.1.2).

Given the empirical loss function (4.4), the parameter θ of the conditional generator G^θ is updated via stochastic gradient decent algorithm until the loss converges or the training reaches the maximal number of iterations.

4.3 Algorithm

Based on the proposed SigCGWAN, we present its algorithm as follows:

First, as described in Section 4.2.1, we construct samples of past/future pairs in a rolling window fashion with fixed window size of $p+q$ and obtain the OLS estimator $\hat{L}_\mu$ using the whole training set to estimate conditional expected signature under true measure $\mathbb{E}_\mu[S_M(X_{\text{future}})|X_{\text{past}} = y]$. Second, in each training epoch, we randomly sample a mini-batch of past/future sample pair $(x_{\text{past}}^{(i)}, x_{\text{future}}^{(i)})_{i=1}^{B}$, where B is the batch size. The generator G^θ takes the p-lagged value $x_{\text{past}}^{(i)}$ and latent noise z to predict q-step future paths, i.e. $\hat{x}_{\text{future}}^{(i)} = G^\theta(x_{\text{past}}^{(i)}, z)$. Then we apply Monte Carlo method to estimate the conditional expected discriminator score under fake measure for every past sample $\{x_{\text{past}}^{(i)}\}_{i=1}^{N}$, i.e. $\mathbb{E}_{\hat{\nu}_i}[S_{L_2}(X_{\text{future}})] \approx \mathbb{E}_{\nu^{\theta}(x_{\text{past}}^{(i)})}[S_{L_2}(X_{\text{future}})]$.

Given the estimator $\hat{L}_{\mu}$ and $\hat{\nu}_i$, we are able to compute the loss given in (4.4) and update the model parameters of the generator G^{θ} using an optimizer until it converges or the maximum number of training epochs is reached. The pseudo-code of SigCWGAN is listed in in Algorithm 2, and we present the flowchart of SigCWGAN algorithm in Figure 4.2.

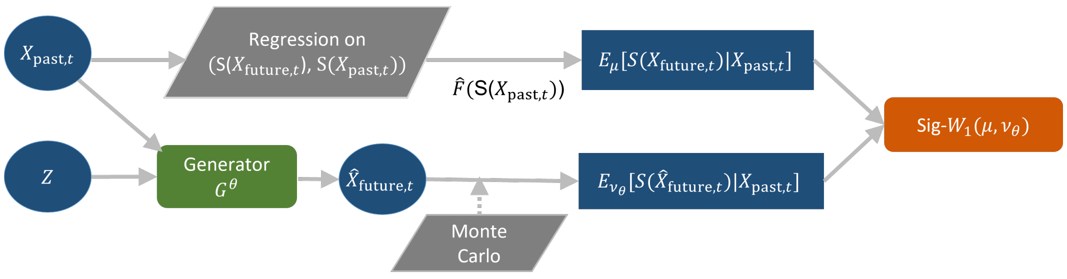

$X_{\text{past},t}$
Regression on $(S(X_{\text{future},t}), S(X_{\text{past},t}))$
$\hat{F}(S(X_{\text{past},t}))$
$E_{\mu}[S(X_{\text{future},t})|X_{\text{past},t}]$
Sig-$W_1(\mu, \nu_{\theta})$
Z
Generator G^{θ}
$\hat{X}_{\text{future},t}$
$E_{\nu_{\theta}}[S(\hat{X}_{\text{future},t})|X_{\text{past},t}]$
Monte Carlo

Figure 4.2: The illustration of the flowchart of SigCWGAN.

4.4 Numerical results

To benchmark with SigCWGAN, we choose three representative generative models for the time-series generation, i.e. (1) TimeGAN, (2) RCGAN - a conditional GAN, and (3) GMMN - an unconditional MMD with a Gaussian kernel. Furthermore, for the stock dataset, we compare the proposed SigCWGAN with the Generalized autoregressive conditional heteroskedasticity model (GARCH), which is a popular econometric time series model.

To assess the goodness of the fitting of a generative model, we consider three main criteria (a) the marginal distribution of time series; (b) the temporal and feature dependence; (c) the usefulness - synthetic data should be as useful as the real data when used for the same predictive purposes (i.e. train-on-synthetic, test-on-real). The test metrics are defined below.

In the following, we describe the calculation of the test metrics precisely. Let $(X_t)_{t=1}^{T}$ denote a d-dimensional time series sampled from the real target distribution. We first extract the past/future pairs $(X_{t-p+1:t}, X_{t+1:t+q})_{t\in\mathcal{T}}$, where $\mathcal{T}$ is the set of time indexes. Given the generator G^{θ}, for each input sample $(X_{t-p+1:t})$, we

Algorithm 2 Algorithm for training SigCWGAN

```
Input:
    $\{x_i \in \Omega_0^1(J;\mathbb{R}^d) : i \in \{1,\dots N\}\}$: input time series,
    $L_1, L_2 \in \mathbb{N}$: order of truncated signature of the past/future path,
    $J_1, J_2$: time interval of past/future path,
    $\alpha_\theta \in \mathbb{R}_{>0}$: learning rate,
    $N$: the number of epochs,
    $B \in \mathbb{N}, B \le N$: batch size,
    $N_{MC}$: the number of Monte Carlo samples.
Output:
    $\theta$: approximation of the optimal parameters of the generator $G$
 1: Compute truncated signature of the past and future paths: $(S_{L_1}(x_{J_1}^{(i)}), S_{L_2}(x_{J_2}^{(i)}))_{i=0}^N$.
 2: Compute linear regression coefficient $\hat{L}$ using the truncated signature $(S_{L_1}(x_{J_1}^{(i)}), S_{L_2}(x_{J_2}^{(i)}))_{i=0}^N$.
 3: Initialize the model parameter $\theta$ of the generator $G$.
 4: for $i = 1 : N$ do
 5:     Sample set $J$ of $B$ random indices without repetition from $\{1,\dots,N\}$.
 6:     Initialize the loss function $\Gamma(\theta) \leftarrow 0$
 7:     for $j = 1 : J$ do
 8:         Simulate $N_{MC}$ samples of the simulated future path $\{\hat{x}_{J_2}^{(n)}\}_{n=1}^{N_{MC}}$ by the generator $G^\theta$ given the past path $x_{J_1}^{(j)}$.
 9:         Compute
                $$\hat{\mathbb{E}}_{X \sim \nu_\theta(x_{J_1}^{(j)})}[S_{L_2}(X)] \leftarrow \frac{1}{N_{\mathrm{MC}}} \sum_{n=1}^{N_{\mathrm{MC}}} S_{L_2}(\hat{x}_{J_2}^{(n)}).$$
10:         update the loss function
                $$\Gamma(\theta) \leftarrow \frac{1}{|I|} \sum_{j \in J} \|\hat{L}_\mu(S_{L_1}(x_{J_1}^{(j)})) - \hat{\mathbb{E}}_{X \sim \nu_\theta(x_{J_1}^{(j)})}[S_{L_2}(X)]\|_2$$
11:     end for
12:     Compute gradient and update generator parameters via (stochastic) gradient descent
                $$\theta \leftarrow \theta - \alpha_\theta \nabla_\theta \Gamma(\theta)$$
13: end for
14: return $\theta$
```

generate one sample of the q-step forecast $\hat{X}_{t+1,t+q}^{(t)}$ (if G^θ is not conditional generator, we generate a sample of q-step forecast $\hat{X}_{t+1,t+q}^{(t)}$ without any conditioning variable.). The synthetic data generated by G^θ is given by $\{\hat{X}_{t+1,t+q}^{(t)}\}_t$, which we use to compute the test metrics.

- **Metric on marginal distribution**: The empirical density function (epdf) of real data and synthetic data are computed based on their histograms. When talking about epdf, we mean each bin's raw count divided by the total number of counts and the bin width. For each feature dimension $i \in \{1, \cdots, d\}$, we denote the epdfs of real data and synthetic data as $\hat{df}^i_r$ and $\hat{df}^i_G$ respectively. Here the epdfs of synthetic date $\hat{df}^i_G$ is computed on the bins derived from the histogram of real data. The metric on marginal distribution, named as **abs metric**, is defined as the absolute difference of those two epdfs averaged over feature dimension, i.e.

$$\frac{1}{d}\sum_{i=1}^{d} |\hat{df}^i_r - \hat{df}^i_G|_1,$$

where $|\hat{df}^i_r - \hat{df}^i_G|_1$ is computed as the l_1 distance between the epdfs of real and synthetic data on each bin. Notice that although **abs metric** cannot give a fully point-separating metric on the space of measure, it can still provide a general description of the similarity between two set of data given a reasonable number of bins. Considering the computational cost, we set number of bins to 50 in our implementation.

- **Temporal dependency**: We use the absolute error of the auto-correlation estimator by real data and synthetic data as the metric to assess the temporal dependency and name it as **ACF metric**. For each feature dimension $i \in \{1, \ldots, d\}$, we compute the auto-covariance of the i^{th} coordinate of time series data X with lag value k under real measure and synthetic measure resp, denoted by $\rho^i_r(k)$ and $\rho^i_G(k)$. Here Then the estimator of the lag-1 auto-correlation of the real/synthetic data is given by $\frac{\rho^i_r(1)}{\rho^i_r(0)}/\frac{\rho^i_G(1)}{\rho^i_G(0)}$. The ACF metric is defined to be the absolute difference of lag-1 auto-correlation given as follows:

$$\frac{1}{d}\sum_{i=1}^{d}\left|\frac{\rho_r^i(1)}{\rho_r^i(0)}-\frac{\rho_G^i(1)}{\rho_G^i(0)}\right|.$$

In addition, we present the ACF plot, which illustrates the autocorrelation of each coordinate of the time series with different lag values. The synthetic data's quality is evaluated by how closely its ACF plot resembles that of the real data, as it indicates the synthetic data's ability to capture long-term temporal dependencies.

- **Feature dependency**: For $d > 1$, we assess the feature dependency by using the l_1 norm of the difference between cross-correlation matrices and name it as **correlation metric**. Let $\tau_r^{i,j}$ and $\tau_G^{i,j}$ denote the correlation of the i^{th} and j^{th} feature of time series under real measure and synthetic measure resp. The correlation metric between the real data and synthetic data is given by l_1 norm of the difference between two correlation matrices, i.e.

$$\sum_{i=1}^{d}\sum_{j=1}^{d}|\tau_r^{i,j}-\tau_G^{i,j}|.$$

- **Predictive score**: In order to be useful, the synthetic data should inherit the predictive characteristics of the original, meaning that the synthetic data should be just as useful as the real data when used for the same predictive purpose (i.e. train-on-synthetic, test-on-real). To measure the usefulness of the synthetic data, we follow [34] and [33] and consider the problem of predicting next-step temporal vectors using the lagged values of time series using the real data and synthetic data. First, we train a supervised learning model on real data to predict next-step values and evaluate it in terms of R^2(TRTR). Then we train the same supervised learning model on synthetic data and evaluate it on the real data in terms of R^2 (TSTR). The closer two R^2 are, the better the generative model is. The predictive score is then defined as the R^2 **relative error**. This test metric is reasonable because it demonstrates the ability of the synthetic data to be used for real applications.

4.4.1 Vector autoregressive model

To demonstrate the model's ability to generate realistic multi-dimensional time series in a controlled environment, we consider synthetic data generated by the Vector Autoregressive (VAR) model, in the d-dimensional VAR(1) model, time series are defined recursively for $t \in \{1, ..., T-1\}$ through the following equation:

$$X_{t+1} = \phi X_t + \varepsilon_{t+1},$$

where $(\varepsilon_t)_{t=1}^{T}$ are i.i.d. Gaussian-distributed random variables with co-variance matrix $\sigma \mathbf{1} + (1-\sigma)\mathbf{I}$; $\mathbf{I}$ is a $d \times d$ identity matrix. Here, the coefficient $\phi \in [-1, 1]$ controls the auto-correlation of the time series and $\sigma \in [0, 1]$ the correlation of the d features.

In our benchmark, we investigate the dimensions $d = 1, 2, 3$ and various (σ, ϕ). Across all dimensions, we observe that the SigCWGAN has a comparable performance or outperforms the baseline models in terms of the metrics defined above. Furthermore, we find that as the dimension increases the performance of SigCWGANs exceeds baselines. We illustrate this finding in Figure 4.3 (Right) which shows the relative error of TSTR R^2 when varying the dimensionality of VAR(1). Observe that the SigCWGAN remains a very low relative error, but the performance of the other models deteriorates significantly, especially the GMMN.

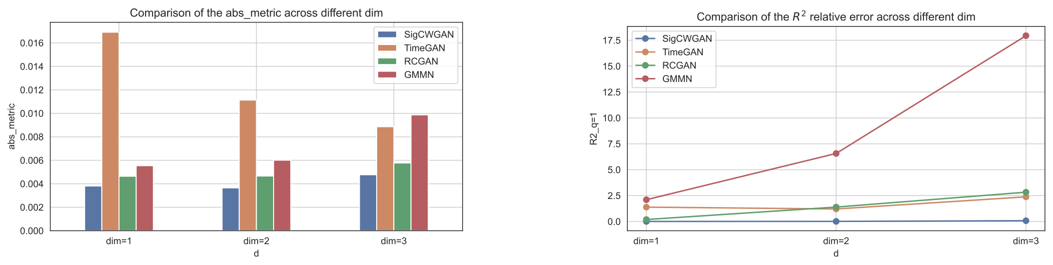
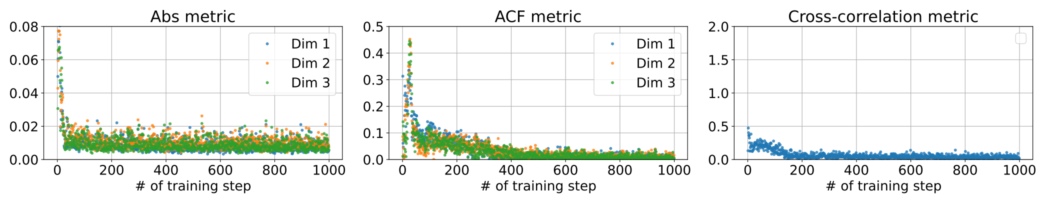
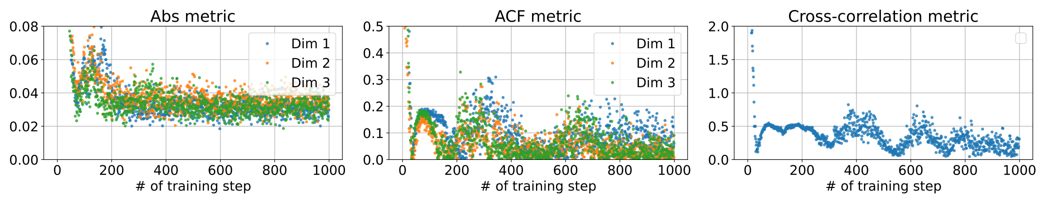


Figure 4.3: (Left) The distributional metric (*abs_metrics*) comparison; (Right) the R^2 (TSTR) comparison. VAR(1) data is generated for $\phi = 0.8$ and $\sigma = 0.8$.

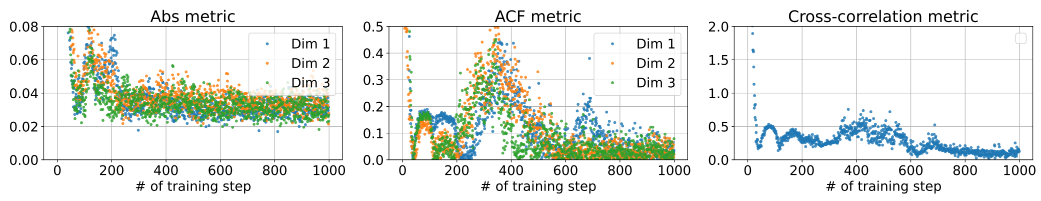
Figure 4.4 shows the development of the abs metric, ACF metric, and cross-correlation metric through the course of training for the 3-dimensional VAR(1)



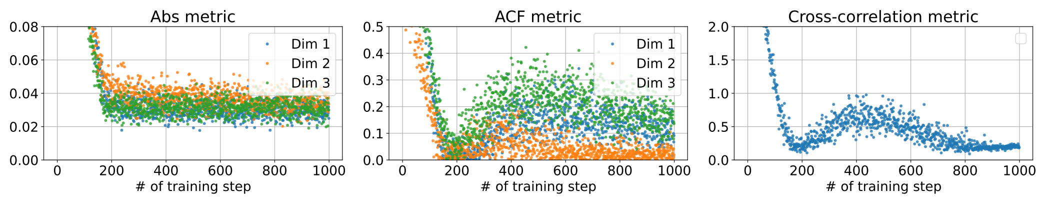
(a) SigCWGAN



(b) TimeGAN



(c) RCGAN



(d) GMMN

Figure 4.4: Evolution of three test metrics: abs metric, ACF metric, and cross-correlation metric. Each color represents the metric of one dimension. The results are for the 3-dimensional VAR(1) model for $\phi = 0.8$ and $\sigma = 0.8$.

model. While the ACF metrics of the baseline models oscillate heavily, the SigCWGAN ACF metric and cross-correlation metric converge nicely toward zero. Also, the GMMN abs metric converges nicely to a certain level. However, in contrast, its ACF metric does not converge. This highlights the stability and usefulness of the Sig-W_1 distance as a loss function. Furthermore, the SigCWGAN has the advantage of generating a realistic long-time series over the other models, which is reflected by the marginal density function of a synthetic sampled path of 80,000 steps are much closer to that of real data than baselines in Figure 4.5.

To better show our model's capability in capturing the conditional law, we include Figure 4.6 which compares the real and synthetic distribution of future time series given one past path sample. The shaded area represents the 95% confidence interval. We can see that compared with other models, SigCWGAN has a better fitting of the expected values (as represented by the central line within the shaded area) and confidence interval, which emphasizes the model's superior performance in capturing conditional laws.

To emphasize the strengths of SigCWGAN in capturing the long-term temporal dependency, we also present the long-term ACF plot of synthetic data in Figure D.4. We can observe that SigCWGAN demonstrates the best fitting of long-term ACF. More details of the numerical results can be found in Appendix C.2.

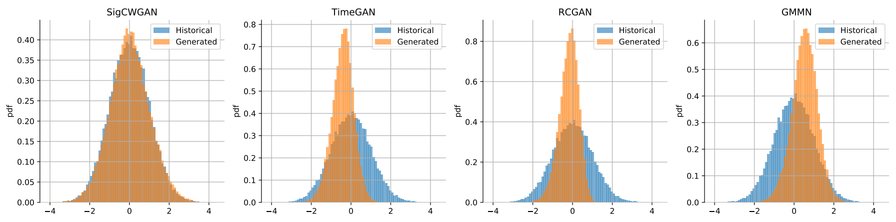


Figure 4.5: Comparison of the marginal distributions of one long sampled path (80,000 steps) with the real distribution using VAR(1) model for $\phi = 0.8, \sigma = 0.8$.

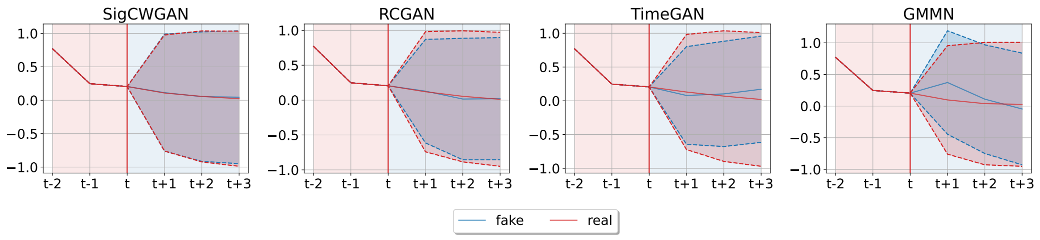


Figure 4.6: Comparison of all models' performance in fitting the conditional distribution of future time series given one past path sample. The real and generated paths are plotted in red and blue respectively with the shaded area as the 95% confidence interval. The real samples are synthesized from VAR(1) model for $d = 3$, $\phi = 0.8$ and $\sigma = 0.8$.

4.4.2 ARCH model

To assess the non-linear temporal dependency, we also choose synthetic data generated by *Autoregressive Conditional Heteroskedasticity* (ARCH) model [64]. In the ARCH(k) model, time series are defined recursively for $t \in \{1, ..., T-1\}$ through the following equation:

$$\begin{aligned} \sigma_t^2 &= \alpha_0 + \sum_{i=1}^{k} \alpha_i X_{t-i}^2, \\ X_t &= \sigma_t \varepsilon_t, \end{aligned} \tag{4.5}$$

where $\{\varepsilon_t\}_{t=1}^{T}$ are iid Gaussian-distributed random variables, and $\alpha_i > 0$, $i = 0, 1, \ldots, k$ are coefficients that measure the extent to which the past values affect current volatility.

We implement extensive experiments on ARCH(k) with different $k-$lag values, i.e. $k \in \{2,3,4\}$, and choose $\alpha_i = 0.2$ for $i = 1, \ldots, k$. We choose the optimal degree of signature 3. The comparison of the ACF metric is plotted in Figure 4.7. We can see that for all different k values, SigCGWAN significantly outperforms other baselines, highlighting its strength in capturing non-linear temporal dependency. Also, the predictive score comparison in Figure 4.7 indicates that SigCWGAN achieves an extremely low and stable predictive score among all k values, showcasing the stronger predictive power of SigCWGAN. The complete numerical results are summarized in Table D.6. The best results among all the models are highlighted in bold.

For ARCH model, it is more interesting to investigate the ACF/PACF of the squared residuals. The ACF/PACF analysis of squared residuals is a key aspect of assessing the performance of generative model trained on ARCH data. In Figure 4.8, we show the ACF/PACF plot of the squared residuals generated by each model. Notably, in the PACF plot of ARCH(2) models, We can observe two significant values at lag 1 and 2. Our results demonstrate that the SigCWGAN outperforms other models by better capturing the temporal dependency in the squared residuals.

4.4.3 S&P 500 and DJI market data

To assess the performance of our method on the empirical data, we again choose the dataset of the S&P 500 index (SPX) and Dow Jones index (DJI) and their realized volatility, which is retrieved from the Oxford-Man Institute's "realized library" [1]. We aim to generate a time series of both the log return of the close prices and the

[1]`https://oxford-man.ox.ac.uk`

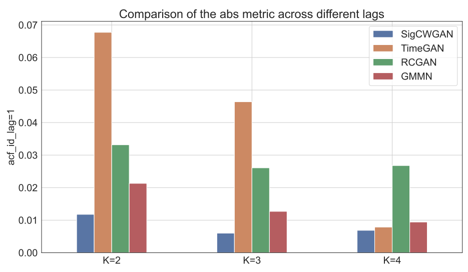
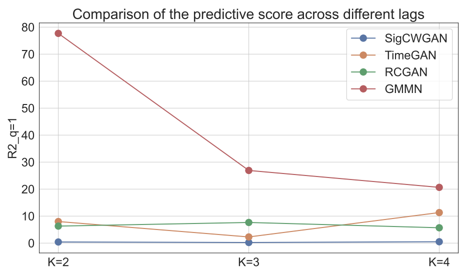
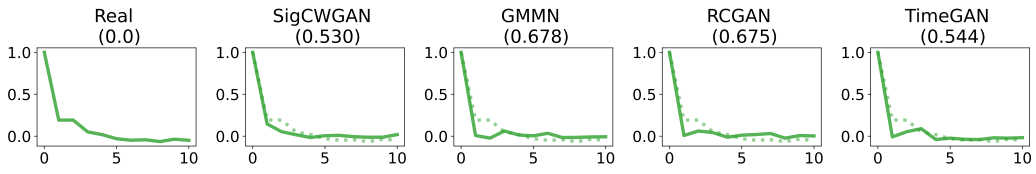
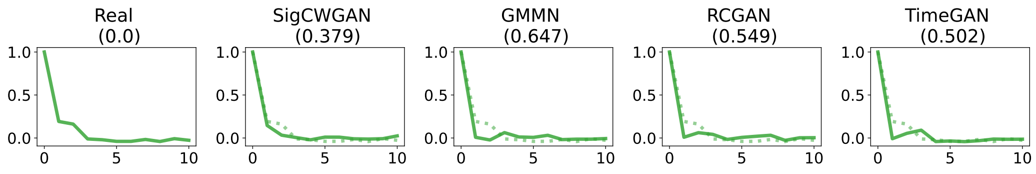


Figure 4.7: (Left) The distributional metric (acf_score) comparison; (Right) the R^2 error (TSTR) comparison using ARCH(k) data for $k = 2, 3, 4$.



(a) ACF of squared residuals.



(b) PACF of squared residuals.

Figure 4.8: ACF/PACF plot of squared residuals for ARCH(2) model. Here x-axis represents the lag value (with a maximum lag equal to 10) and the y-axis represent the corresponding auto- correlation/partial auto-correlation. The length of the real/generated time series used to compute the ACF is 1000. The number in the bracket under each model is the sum of the absolute difference between the correlation coefficients computed from real (dashed line) and generated (solid line) samples.

log of median realized volatility of (a) the SPX only; (b) the SPX and DJI. Table 4.1 shows that SigCWGAN achieves superior or comparable performance to the other baselines. The SigCWGAN generates the realistic synthetic data of the SPX and DJI data shown by the marginal distribution comparison with that of real data in Figure 4.9. For the SPX-only data, GMMN performs better than our model in terms of the fitting of marginal distribution, but it suffers from poor auto-correlation and feature correlation. It is worth noting that our SigCWGAN model outperforms GARCH, the classical and widely used time series model in econometrics, on both the SPX and SPX/DJI data, as shown in Table Table 4.1. The poor performance of the GARCH model could be attributed to its parametric nature and the potential issues of model misspecification when applied to empirical data.

It will be also interesting to investigate the ACF/PACF of synthetic absolute and squared returns, which provide insights into the volatility clustering property of stock returns. We can see from Figure 4.10 that GMMN failed to capture the temporal dependency of absolute and squared returns. In contrast, SigCWGAN demonstrates decent performance in fitting the ACF/PACF of absolute and squared returns. Hence, our results demonstrate that SigCWGAN does better in capturing the temporal dependency in (both absolute and squared) stock returns. This fact can also be observed from Figure D.11 where the synthetic log return path generated by SigCWGAN clearly shows the volatility clustering phenomenon. The ACF plot of synthetic returns can also be found in Figure D.10.

We also include in this section the coverage ratio test of SPX/DJI returns. The coverage ratio test is a statistical test used to assess the accuracy and reliability of the estimated Value at Risk (VaR) value. To perform the test, we initially compute the 95% VaR for both 1 and 5-day ahead returns. We then compute the ratio of the number of exceedances to the total number of observations. If the 95% VaR estimate is correct, the ratio should be close to 5%. We summarize the results of the coverage ratio test in Table 4.2. We can observe that RCGAN has the closest ratio for 1-day ahead return of SPX (4.96%) and DJX (4.42%) to the target of 5%, indicating a more accurate VaR estimate for 1-day ahead returns. In contrast, SigCWGAN, while achieving the best ratio of 5-day ahead return of DJI (4.02%), tends to have an over-optimistic VaR estimate for 1-day ahead returns, resulting in ratios of 1-day ahead returns of SPX (8.04%) and DJI (9.11%) that are far from 5%.

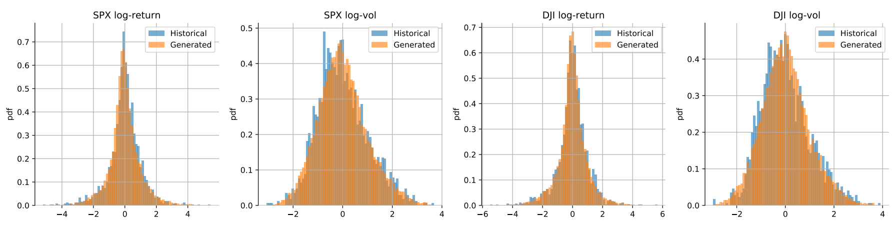
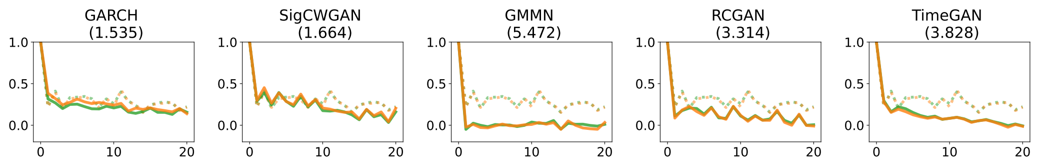
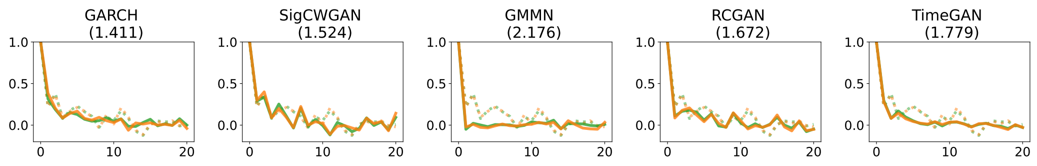


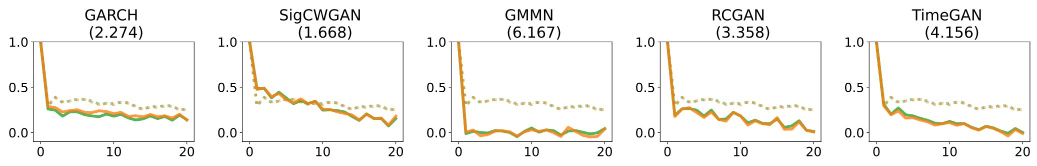
Figure 4.9: Comparison of the marginal distributions of the generated SigCWGAN paths and the SPX and DJI data.



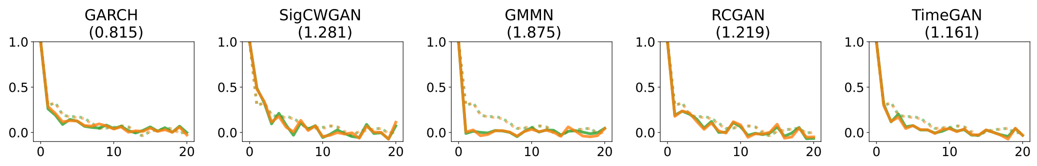
(a) ACF of squared returns.



(b) PACF of squared returns.



(c) ACF of absolute returns.



(d) PACF of absolute returns.

Figure 4.10: ACF/PACF plot of absolute and squared returns for SPX (green line) and DJI (orange line). Here x-axis represents the lag value (with a maximum lag equal to 10) and the y-axis represents the corresponding auto-correlation/partial auto-correlation. The length of the real/generated time series used to compute the ACF is 1000. The number in the bracket under each model is the sum of the absolute difference between the correlation coefficients computed from real (dashed line) and synthetic (solid line) samples.

Metrics	abs metric	ACF score	correlation	$R^2(\%)$	Sig-W_1
SigCWGAN	0.01468,**0.00879**	**0.02693**,0.03988	**0.00638**,**0.10354**	5.56765,**5.28866**	**4.74507**,**5.18462**
TimeGAN	0.01064,0.01136	0.0341,**0.03147**	0.0101,0.11342	8.5833,7.13335	4.75276,5.22316
RCGAN	0.01134,0.00933	0.04192,0.03367	0.0926,0.13699	**3.74567**,5.75112	4.74862,5.20083
GMMN	**0.00986**,0.01438	0.05483,0.06668	0.07498,0.28339	11.37772 ,13.7324	4.75251,5.19631
GARCH	0.01583,0.01670	0.05392, 0.05337	0.15791, 0.7290	12.1253, 12.5686	4.75825, 5.25344

Table 4.1: Numerical results of the stock datasets. In each cell, the left/right numbers are the results for the SPX data/ the SPX and DJI data respectively. We use the relative error of TSTR R^2 against TRTR R^2 as the R^2 metric.

Models	Ratio of 1-day ahead returns	Ratio of 5-day ahead returns
SigCWGAN	8.04%,9.11%	3.62%,**4.02%**
TimeGAN	6.70%,8.31%	**5.63%**,6.70%
RCGAN	**4.95%**,**4.42%**	3.62%,3.88%
GMMN	6.70%,6.70%	6.43%,6.30%
GARCH	9.70%,8.75%	7.63%,6.70%

Table 4.2: Results of coverage ratio test applied to SPX and DJI stock returns. In each cell, the left/right numbers are the results for the SPX/ DJI data respectively. A ratio that is closer to 5% indicates a more accurate 95% VaR estimate. The VaR estimate of each model for each observation is obtained using 5000 generated samples.

4.5 Conclusions

In this chapter, we developed the signature-based conditional Wasserstein GAN for time series generation. SigCWGAN utilizes a signature-based loss function, namely Sig-W_1 metric, which is an explicit approximation of W_1 metric using the signature features space. It eliminates the problem of having to approximate a costly discriminator/critic and, as a consequence, dramatically turns the min-max game into supervised learning. The generator in SigCWGAN is specified as an AR-FNN generator that is designed to capture the autoregressive structure of the target time series. We validate the advantages of SigCWGAN on both synthetic and empirical datasets.

As demonstrated in Section 4.4, SigCWGAN achieves state-of-the-art results on both synthetic and empirical datasets. On synthetic datasets such as VAR and ARCH datasets. SigCWGAN shows dominantly better performance in terms of distributional metric (abs metric) and predictive score (R^2 relative error). Additionally, SigCWGAN dramatically improves the training stability and has much faster convergence. On S&P 500 and DJI dataset, SigCWGAN achieves either the best or comparable performance in terms of all test metrics compared to other baselines and is effective in generating high-fidelity stock data.

Chapter 5

Monte-Carlo GAN

The SigCWGAN algorithm, proposed in Chapter 4, enhances generative performance by using signature features in the discriminative loss function and reducing the min-max problem into supervised learning. In this chapter, we introduce a novel algorithm called Monte-Carlo GAN (MCGAN) that improves generation performance from the perspective of the generator. MCGAN utilizes the mean squared error (MSE) as the generative loss function to measure the distance between real and fake discriminator scores. The strong supervision provided in the MSE allows the generator to learn the target measure μ with a relatively weaker condition imposed on the (non-optimal) discriminator. Additionally, it is shown in Theorem 5.3.2 that the innovative generative loss function in MCGAN improves training stability compared to the original GAN. We also employ a toy example called Dirac-GAN to better illustrate the advantages of MCGAN. Numerical experiments conducted on both synthetic and empirical datasets further demonstrate the effectiveness and improved stability of MCGAN.

5.1 Motivation

In this section, we explain the motivation of the MCGAN model and the intuition behind MSE.

Bottleneck of SigCWGAN: Despite the superior performance of SigCWGAN shown in Section 4.4, it still has some potential problems. As proposed in Section 4.2.1, a regression module is employed to estimate the conditionally expected signature. Nevertheless, one potential bottleneck of this approach is that utilizing function approximation on the signature feature may lead to unsatisfactory performance of estimating the conditionally expected signature under the true measure, and hence the poor estimator of the conditional Sig-W_1 metric. Moreover, although the linear models on the signature space have universality, they might be not always effective. Similar to polynomial regression, such models might be unstable and result in overfitting problems in the case of a very high degree of the truncated signature.

Challenges in continuous setting: The same problem happens to most GAN variants in conditional time-series generation tasks. GAN variants such as WGAN usually require estimation of the conditionally expected discriminator score under real measures. Take WGAN as an example, ones needs to estimate $\mathbb{E}_{\mu(x_{\text{past}})}[D^{\phi}(X_{\text{future}}, X_{\text{past}})]$ in order to compute the following conditional W_1 distance:

$$W_1(\mu, \nu_\theta) := \max_{|D^{\phi}|_{\text{Lip}} \leq 1} \mathbb{E}_{\mu(X_{\text{past}})}[D^{\phi}(X_{\text{future}}, X_{\text{past}})] - \mathbb{E}_{\nu_\theta(X_{\text{past}})}[D^{\phi}(X_{\text{future}}, X_{\text{past}})].$$

In the continuous setting studied in this thesis, one past path corresponds to only one future path, which makes the one-sample estimator unreliable. It usually demands employing additional function approximation to the conditional expectation, which will introduce additional bias and make the overall algorithm much harder to tune.

To tackle these challenges, we are interested in developing a new algorithm to avoid the direct estimation for the conditional expectation under true measure while achieving superior performance in conditional time series generation tasks.

In the following, let us explain the general framework of learning the conditional distribution induced by time series without explicitly estimating the conditionally expected signature under the true measure. Let $\mathcal{X}$ be the target space, $\mathcal{Y}$ the conditioning space, and $\mathcal{Z}$ the latent space. The conditional generative model we are interested in is to use the generator $G^{\theta} : \mathcal{Z} \times \mathcal{Y} \to \mathcal{X}$ to learn the conditional distribution of future paths $X_{\text{future}} \in \mathcal{X}$ given past paths $X_{\text{past}} \in \mathcal{Y}$. Let us consider the following mean-square optimization problem:

$$\min_{\xi} \mathbb{E}_{\mu}[|D^{\phi}(X_{\text{future}}, X_{\text{past}}) - \xi(X_{\text{past}})|^2], \tag{5.1}$$

where ξ is a measurable function and $D^{\phi} : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ is a discriminator parameterized by ϕ. From the Doob-Dynkin Lemma, we know that this conditional expectation is a measurable function of X_{past} and the minimum of (5.1) is obtained by $\xi^*(X_{\text{past}}) = \mathbb{E}_{\mu(X_{\text{past}})}[D^{\phi}(X_{\text{future}}, X_{\text{past}})]$, where $\mu(X_{\text{past}})$ denotes the conditional measure of the future path given the past path X_{past}. This fact inspires us to use the MC estimator of $\mathbb{E}_{\nu_\theta(X_{\text{past}})}[D^{\phi}(X_{\text{future}}, X_{\text{past}})]$ in place of $\xi(X_{\text{past}})$ in (5.1) and use the mean square error (MSE) to train the generator. Thus the novel generative loss function can be defined as:

$$L_G(\theta; \phi) = \mathbb{E}_{\mu}[|D^{\phi}(X_{\text{future}}, X_{\text{past}}) - \mathbb{E}_{\nu_\theta(X_{\text{past}})}[D^{\phi}(X_{\text{future}}, X_{\text{past}})]|^2]. \tag{5.2}$$

This innovative loss function reframes the conventional generator training into a mean-square optimization problem by computing the MSE between real and fake discriminator scores. It requires no additional least square regression to estimate the real conditional expectation and can avoid the estimation bias. In the ideal scenario, the optimal generator to (5.2) given D^{ϕ} will result in the equality:

$$\mathbb{E}_{\mu(X_{\text{past}})}[D^{\phi}(X_{\text{future}}, X_{\text{past}})] = \mathbb{E}_{\nu_\theta(X_{\text{past}})}[D^{\phi}(X_{\text{future}}, X_{\text{past}})]. \tag{5.3}$$

In practice, the conditional expectation of discriminator score under fake measure can be estimated with the Monte Carlo method. Let us consider the input-output pair $\{(x^{(i)}_{\text{future}}, x^{(i)}_{\text{past}})\}_{i=1}^N$ that are sampled from the same joint distribution $\mathcal{P}(\mathcal{X}, \mathcal{Y})$, the loss function can thus be approximated via:

$$L_G(\theta; \phi) \approx \frac{1}{N} \sum_{i=1}^{N} |D^{\phi}(x^{(i)}_{\text{future}}, x^{(i)}_{\text{past}}) - \underbrace{\frac{1}{N_{MC}} \sum_{j=1}^{N_{MC}} D^{\phi}(G^{\theta}(x^{(i)}_{\text{past}}, z^{(j)}_t), x^{(i)}_{\text{past}})}_{\approx \mathbb{E}_{\nu_\theta(x^{(i)}_{\text{past}})}[D^{\phi}(X_{\text{future}}, x^{(i)}_{\text{past}})]} |^2, \tag{5.4}$$

where N_{MC} is the sample size used for Monte-Carlo estimation. We name this model the conditional Monte Carlo (MCGAN) due to the usage of the Monte Carlo estimator of conditionally expected discriminator score under the fake measure.

Notice that (5.3) would not necessarily lead to $\nu_\theta = \mu$. It requires a certain property of the discriminator to ensure the match of ν_θ and μ. This property plays an important role in the generator training using (5.2), and it should reflect the discriminative power of the discriminator to distinguish between two different measures. Discriminators with weak discriminative power, let us say a constant function, usually drag down the training of the generator due to the incorrect or useless information provided. A candidate discriminator should possess the ability to differentiate between the real distribution and fake distribution in a certain sense, ensuring that the minimized MSE in (5.1) is meaningful. However, how to define this property, how to obtain this property, and whether this loss function can guarantee the optimality of $\nu_\theta = \mu$ remain questions. We will answer these questions in the following sections.

5.2 Discriminability

To better illustrate the advantage of our model and also for clarity of notation, we set aside the conditioning variable for now. Considering the fact that all arguments around the mean-square optimization problem (5.1) also hold in the unconditional case, we use the unconditional generation task instead to focus on the property of the discriminator and showcase the benefits of MSE.

Recall that in the unconditional generation task, our goal is to use the generator $G^\theta : \mathcal{Z} \to \mathcal{X}$ to learn the distribution of target variable $X \in \mathcal{X}$ such that ν_θ induced by G^θ can approximate the real conditional measure μ. Throughout this chapter, we assume that both μ and ν_θ are absolutely continuous with densities p_μ and p_{ν_θ}, respectively. The generative loss function using MSE can be defined as

$$L_G(\theta;\phi) = \mathbb{E}_\mu[|D^\phi(X) - \mathbb{E}_{\nu_\theta}[D^\phi(X)]|^2], \tag{5.5}$$

where $D^\phi : \mathcal{X} \to \mathbb{R}$ is the discriminator.

The discriminator is trained by maximizing the discriminative loss function. The discriminative loss function is usually defined as the objective function of certain divergences such as JS divergence in GAN [35], and computing such divergence involves finding the optimal discriminator that maximizes the objective function. See [65] for a comprehensive description of the discriminative loss function. The discriminative loss functions are usually formulated as:

$$L_D(\phi;\mu,\nu_\theta) := \mathbb{E}_\mu\left[f_1(D^\phi(X))\right] + \mathbb{E}_{\nu_\theta}\left[f_2(D^\phi(X))\right],$$

where f_1 and f_2 are real-valued functions on $\mathbb{R}$. Here we include μ,ν_θ in the notation of L_D to emphasize its dependence on these two measures. In GAN training, the optimal discriminator is the one that minimizes the discriminative loss function and is further used in the generative loss function to train the generator.

It is important to notice that the optimal discriminator also depends on μ and ν_θ as well, and minimizing the discriminative loss involves finding the optimal parameter ϕ^*_{μ,ν_θ} specific to μ and ν_θ. To clarify this point, we give the following definition.

Definition 5.2.1 (Optimal discriminator)**.** Let $L_D : \Phi \times \mathcal{P}(\mathcal{X}) \times \mathcal{P}(\mathcal{X}) \to \mathbb{R}$ denote a discriminative loss function. A discriminator

$$\begin{aligned} \mathcal{P}(\mathcal{X}) \times \mathcal{P}(\mathcal{X}) \times \mathcal{X} &\to \mathbb{R} \\ (\mu,\nu,x) &\mapsto D^{\phi^*_{\mu,\nu}}(x), \end{aligned} \tag{5.6}$$

where $\phi^*_{\cdot,\cdot} : \mathcal{P}(\mathcal{X}) \times \mathcal{P}(\mathcal{X}) \to \Phi$, is said the *optimal discriminator* of L_D if for any two measures $\mu,\nu \in \mathcal{P}(\mathcal{X})$, it satisfies that

$$L_D(\phi^*_{\mu,\nu};\mu,\nu) = \max_\phi L_D(\phi;\mu,\nu).$$

The parameter $\phi^*_{\mu,\nu}$ represents the specific parameterization of the optimal discriminator for the given measures μ and ν. It can be regarded as a map $\phi^*_{\cdot,\cdot} : \mathcal{P}(\mathcal{X}) \times \mathcal{P}(\mathcal{X}) \to \Phi$ that maps a pair of probability measures into a certain parameterization. By this definition, we emphasize the dependence of the optimal discriminator on the measures μ and ν.

Although an optimal discriminator would have perfect discrimination performance, it may not always be achievable in practice. However, in the case of using MSE as the generative loss function, there is no need for an optimal discriminator as the MSE already provides more information via comparing the difference between the expected discriminator score under real and fake measures. Nevertheless, we still expect the discriminator, to some extent, to shed light on the discrepancy between two different measures, and we name this property as *discriminability*.

Definition 5.2.2 (Discriminability)**.** A discriminator

$$\begin{aligned} \mathcal{P}(\mathcal{X}) \times \mathcal{P}(\mathcal{X}) \times \mathcal{X} &\to \mathbb{R} \\ (\mu, \nu, x) &\mapsto D^{\phi_{\mu,\nu}}(x), \end{aligned} \tag{5.7}$$

where $\phi_{\cdot,\cdot} : \mathcal{P}(\mathcal{X}) \times \mathcal{P}(\mathcal{X}) \to \Phi$, is said to have *discriminability* if for any two different continuous measures $\mu, \nu \in \mathcal{P}(\mathcal{X})$, it satisfies that

$$\int_{\mathcal{A}^{\mu,\nu}} D^{\phi_{\mu,\nu}}(x)(p_\mu(x) - p_\nu(x))dx \neq 0 \tag{5.8}$$

where p_μ and p_ν are the densities of μ and ν respectively, and $\mathcal{A}^{\mu,\nu} := \{x \in \mathcal{X} : p_\mu(x) \neq p_\nu(x)\}$. We denote the set of discriminators with discriminability as $\mathcal{D}_{\text{Dis}}$.

The discriminability can be interpreted as the discriminator's ability to detect the discrepancy between μ and ν over the set $\mathcal{A}^{\mu,\nu}$ where μ differs from ν. However, the integration in (5.8) makes it hard to verify. To better understand the discriminability, we introduce *strict discriminability* that is more intuitive and straightforward to verify.

Definition 5.2.3 (Strict Discriminability)**.** A discriminator

$$\begin{aligned} \mathcal{P}(\mathcal{X}) \times \mathcal{P}(\mathcal{X}) \times \mathcal{X} &\to \mathbb{R} \\ (\mu, \nu, x) &\mapsto D^{\phi_{\mu,\nu}}(x), \end{aligned} \tag{5.9}$$

where $\phi_{\cdot,\cdot} : \mathcal{P}(\mathcal{X}) \times \mathcal{P}(\mathcal{X}) \to \Phi$, is said to have *strict discriminability* if there exist two constants $a \in \{-1, 1\}$ and $c \in \mathbb{R}$ such that for any two different continuous measures $\mu, \nu \in \mathcal{P}(\mathcal{X})$, it satisfies that

$$a(D^{\phi_{\mu,\nu}}(x) - c)(p_\mu(x) - p_\nu(x)) > 0, \tag{5.10}$$

for all $x \in \mathcal{A}^{\mu,\nu} := \{x \in \mathcal{X} : p_\mu(x) \neq p_\nu(x)\}$. We denote the set of discriminators with strict discriminability as $\mathcal{D}_{\mathrm{Dis}^*}$.

The strict discriminability of discriminator can be interpreted as the ability to distinguish between ν and μ point-wisely over $\mathcal{A}^{\mu,\nu}$ by telling the sign (or the opposite sign) of $p_\mu(x) - p_\nu(x)$. In (5.10), if $a = 1$, the constant c can be regarded as a criterion in the sense that $D^{\phi_{\mu,\nu}}(x) - c$ is positive only when $p_\mu(x) > p_\nu(x)$ and vice versa. Compared with discriminability, this strict one is more straightforward to verify, and it holds that $\mathcal{D}_{\mathrm{Dis}^*} \subset \mathcal{D}_{\mathrm{Dis}}$.

This (strict) discriminability covers a variety of optimal discriminators in GAN variants. We present some examples by specifying the value of c for each optimal discriminator.

- **GAN [35]**: GAN employs BCE as the discriminative loss function defined as

$$L_D(\phi; \mu, \nu_\theta) = \mathbb{E}_\mu \left[\log(D^\phi(X))\right] + \mathbb{E}_{\nu_\theta} \left[\log(1 - D^\phi(X))\right]. \tag{5.11}$$

  As proven in [35], the optimal discriminator given binary cross-entropy loss can be derived as:

$$D^{\phi^*_{\mu,\nu_\theta}}(x) = \frac{p_\mu(x)}{p_\mu(x) + p_{\nu_\theta}(x)}. \tag{5.12}$$

  Let us consider function $f(l) = \frac{1}{1+l}$ for $l > 0$. Notice that $f(l) > 1/2$ when $l < 1$ and vice versa. Also notice that $D^{\phi^*_{\mu,\nu_\theta}}(x) = f(\frac{p_{\nu_\theta}(x)}{p_\mu(x)})$, it is easy to verify that $(D^{\phi^*_{\mu,\nu_\theta}}(x) - 1/2)(p_\mu(x) - p_{\nu_\theta}(x)) > 0$ when $p_\mu(x) \neq p_{\nu_\theta}(x)$.

- **Least Square GAN [38]**: LSGAN employs least square loss function defined as follows:

$$L_D(\phi;\mu,\nu_\theta) = -\mathbb{E}_\mu\left[(D^\phi(X)-\alpha)^2\right] - \mathbb{E}_{\nu_\theta}\left[(D^\phi(X)-\beta)^2\right],$$

where $\alpha, \beta \in \mathbb{R}$, and $\alpha \neq \beta$. The optimal discriminator is given as

$$D^{\phi^*_{\mu,\nu_\theta}}(x) = \frac{\alpha p_\mu(x) + \beta p_{\nu_\theta}(x)}{p_\mu(x) + p_{\nu_\theta}(x)}, \tag{5.13}$$

Similarly, by the fact that $D^{\phi^*}(x) = f(\frac{p_{\nu_\theta}(x)}{p_\mu(x)})$, where $f(l) = \frac{\alpha+\beta l}{1+l}$, we can verify that this discriminator also has (strict) discriminability in the sense that $\text{sign}(\alpha-\beta)(D^{\phi^*_{\mu,\nu_\theta}}(x) - \frac{\alpha+\beta}{2})(p_\mu(x) - p_{\nu_\theta}(x)) > 0$ when $p_\mu(x) \neq p_{\nu_\theta}(x)$.

- **Geometric GAN [42]**: Hinge loss function is defined as

$$L_D(\phi;\mu,\nu_\theta) = -\mathbb{E}_\mu\left[\max(0, 1-D^\phi(X))\right] - \mathbb{E}_{\nu_\theta}\left[\max(0, 1+D^\phi(X))\right].$$

By Lemma B.1 in [42], it is straightforward to show that the optimal discriminator can be derived as:

$$D^{\phi^*_{\mu,\nu_\theta}}(x) = 2\mathbb{1}_{\{p_\mu(x) \geq p_{\nu_\theta}(x)\}} - 1. \tag{5.14}$$

It is clear that $D^{\phi^*}(x) = f(\frac{p_\mu(x)}{p_{\nu_\theta}(x)})$, where $f(l) = 2\mathbb{1}_{\{l\geq 1\}} - 1$, and $D^{\phi^*}(x)(p_\mu(x) - p_{\nu_\theta}(x)) > 0$ when $p_\mu(x) \neq p_{\nu_\theta}(x)$.

- **Energy-based GAN [66]**: Energy-based loss function is defined as

$$L_D(\phi;\mu,\nu_\theta) = -\mathbb{E}_\mu\left[D^\phi(X)\right] - \mathbb{E}_{\nu_\theta}\left[\max(0, m-D^\phi(X))\right].$$

where $m > 0$. By Lemma 1 in [66], the optimal discriminator given energy-based loss function can be derived as:

$$D^{\phi^*_{\mu,\nu_\theta}}(x) = m\mathbb{1}_{\{p_\mu(x) < p_{\nu_\theta}(x)\}}. \tag{5.15}$$

It is straight forward to verify that $-m(D^{\phi^*_{\mu,\nu_\theta}}(x) - \frac{m}{2})(p_\mu(x) - p_{\nu_\theta}(x)) > 0$ when $p_\mu(x) \neq p_{\nu_\theta}(x)$.

- ***f*-GAN [67]**: In f-GAN, variational lower bound (VLB) on the f-divergence $\mathrm{Div}_f(\mu||\nu_\theta)$ is used in the generative-adversarial approach to mimic the target distribution ν_θ. Let $f : \mathbb{R}_+ \to \mathbb{R}$ be a convex, lower-semicontinuous function. In f-GAN, the discriminative loss is defined as the variational lower bound on certain f-divergence:

$$L_D(\phi; \mu, \nu_\theta) = \mathbb{E}_\mu \left[D^\phi(X) \right] - \mathbb{E}_{\nu_\theta} \left[f^*(D^\phi(X)) \right], \tag{5.16}$$

where f^* is the convex conjugate function of f. Under mild conditions on the function f [68], the maximum of (5.16) is achieved when

$$D^{\phi^*_{\mu,\nu_\theta}}(x) = f' \left(\frac{p_\mu(x)}{p_{\nu_\theta}(x)} \right), \tag{5.17}$$

where f' is the first order derivative of f and increasing due to the convexity of f. Consequently, we can choose $c = f'(1)$ such that $(D^{\phi^*_{\mu,\nu_\theta}}(x) - c)(p_\mu(x) - p_{\nu_\theta}(x)) > 0$ when $p_\mu(x) \neq p_{\nu_\theta}(x)$. A more detailed list of f-divergence can be found in [67].

In Table 5.1, we also provide a list summarizing those optimal discriminators of discriminative loss functions in some commonly used GAN variants along with the values of a and c.

Although the (strict) discriminability can be obtained by training the discriminator via certain discriminative loss functions, it is worth emphasizing that the discriminator does not necessarily need to reach its optimum to obtain discriminability. In the case of BCE in (5.11), we can easily find a non-optimal discriminator with discriminability as follows:

$$\begin{aligned} \hat{D}^{\phi^*_{\mu,\nu_\theta}}(x) &= D^{\phi^*_{\mu,\nu_\theta}}(x) - \frac{1}{5}\left(D^{\phi^*_{\mu,\nu_\theta}}(x) - \frac{1}{2} \right) \mathbf{1}_{\{p_\mu(x) > p_{\nu_\theta}(x)\}} \\ &= \frac{p_\mu(x)}{p_\mu(x) + p_{\nu_\theta}(x)} - \frac{1}{5}\left(\frac{p_\mu(x)}{p_\mu(x) + p_{\nu_\theta}(x)} - \frac{1}{2} \right) \mathbf{1}_{\{p_\mu(x) > p_{\nu_\theta}(x)\}}. \end{aligned} \tag{5.18}$$

It is also straightforward to verify that $(\hat{D}^{\phi^*_{\mu,\nu_\theta}}(x) - 1/2)(p_\mu(x) - p_{\nu_\theta}(x)) > 0$, and $\hat{D}^{\phi^*_{\mu,\nu_\theta}}(x) \in (0,1)$ for all $x \in \mathcal{X}$ as permitted by the feasible set. Hence it has (strict) discriminability although it is not optimal.

To better illustrate this point, we also present the following 1-D example.

Example 5.2.1. Suppose both μ and ν_θ are 1-D Gaussian measures with different means, i.e. $\mu \sim \mathcal{N}(-2,1), \nu_\theta \sim \mathcal{N}(2,1)$. The optimal discriminator D^{ϕ^*} that maximizes binary cross-entropy loss (BCE) in (5.11) is given by $D^{\phi^*}(x) = \frac{p_\mu(x)}{p_\mu(x)+p_{\nu_\theta}(x)}$ as shown in Figure 5.1. In this case, it has (strict) discriminability in the sense that $(D^{\phi^*}(x) - \frac{1}{2})\big(p_\mu(x) - \nu_\theta(x)\big) > 0$ when $p_\mu(x) \neq \nu_\theta(x)$. This indicates that the discriminator $D^{\phi^*}(x)$ shifted by $\frac{1}{2}$ is capable of telling the sign of $p_\mu(x) - \nu_\theta(x)$. However, a non-optimal discriminator can also have discriminability. The $\hat{D}^{\phi^*}$ defined as (5.18) is such an example shown in Figure 5.1. We can verify that $\hat{D}^{\phi^*}$ will not achieve the minimum of BCE in (5.11) and hence is not optimal, but it still satisfies that $(\hat{D}^{\phi^*}(x) - \frac{1}{2})\big(p_\mu(x) - \nu_\theta(x)\big) > 0$ when $p_\mu(x) \neq \nu_\theta(x)$. In other words, there is no need for discriminators to achieve their optimum to have discriminability.

The discriminability can be regarded as a weaker condition imposed on the discriminator than optimality. While an optimal discriminator would perfectly classify all real and generated samples, it may not always be achievable or necessary in practice. By imposing discriminability as a weaker condition on the discriminator, the focus is on training a discriminator that is competent enough to provide meaningful feedback on the discrepancy between μ and ν_θ. Thanks to the supervised learning nature of MSE, it can utilize this feedback from the discriminator and provide strong supervision to the generator to learn the target distribution. .

5.3 Monte Carlo GAN

Now that we have defined discriminability, the desired property of the discriminator, we are ready to present our model formally.

Recall that the target measure is $\mu \in \mathcal{P}(\mathcal{X})$, and the goal is to train the generator $G^\theta : \mathcal{Z} \to \mathcal{X}$ such that the induced fake measure $\nu_\theta \in \mathcal{P}(\mathcal{X})$ is sufficiently close to μ. To reach this goal, we propose Monte-Carlo GAN (MCGAN) via the following

Name	Discriminative loss	$D^*(x)$	a	c
GAN	Binary cross-entropy	$\frac{p_\mu(x)}{p_\mu(x)+p_{\nu_\theta}(x)}$	1	$1/2$
LSGAN	Least square loss	$\frac{\alpha p_\mu(x)+\beta p_{\nu_\theta}(x)}{p_\mu(x)+p_{\nu_\theta}(x)}$	$\text{sign}(\alpha-\beta)$	$\frac{\alpha+\beta}{2}$
Geometric GAN	Hinge loss	$2\mathbb{1}_{\{p_\mu(x)\geq p_{\nu_\theta}(x)\}}-1$	1	0
Energy-based GAN	Energy-based loss	$m\mathbb{1}_{\{p_\mu(x)<p_{\nu_\theta}(x)\}}$	$\text{sign}(-m)$	$\frac{m}{2}$
f-GAN	VLB on f-divergence	$f'\left(\frac{p_\mu(x)}{p_{\nu_\theta}(x)}\right)$	1	$f'(1)$

Table 5.1: List of common discriminative loss functions that satisfy strict discriminability

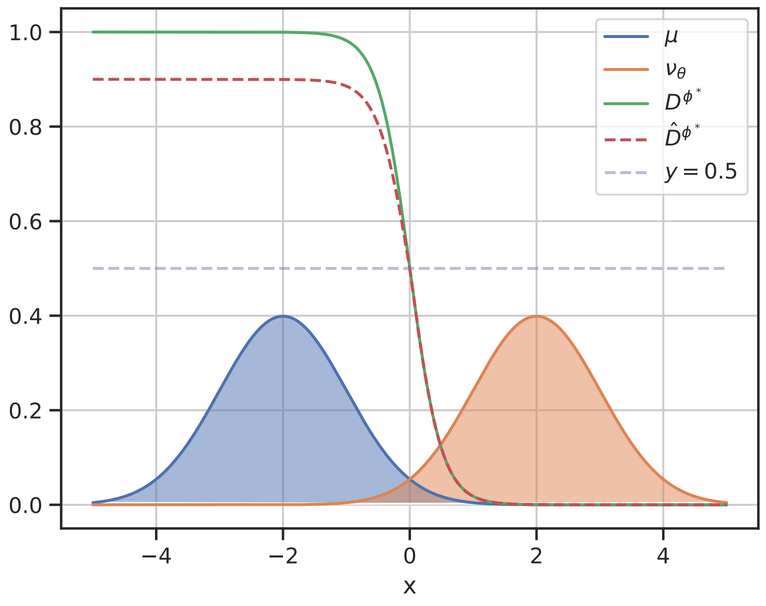


Figure 5.1: One illustration on the discriminability of the discriminator. In this example, $\mu \sim \mathcal{N}(-2,1), \nu_\theta \sim \mathcal{N}(2,1)$. The optimal discriminator is given by $D^{\phi^*}(x) = \frac{p_\mu(x)}{p_\mu(x)+p_{\nu_\theta}(x)}$, and the weaker version of discriminator $\hat{D}^{\phi^*}$ is defined in (5.18).

training objective:

$$\begin{aligned}\max_{\phi\in\Phi} L_D(\phi;\mu,\nu_\theta) &= \max_{\phi\in\Phi} \mathbb{E}_\mu\left[f_1(D^\phi(X))\right] + \mathbb{E}_{\nu_\theta}\left[f_2(D^\phi(X))\right], \\ \min_{\theta\in\Theta} L_G(\theta;\phi,\mu) &= \min_{\theta\in\Theta} \mathbb{E}_\mu[|D^\phi(X) - \mathbb{E}_{\nu_\theta}[D^\phi(X)]|^2],\end{aligned} \tag{5.19}$$

where functions f_1 and f_2 depend on the choice of the discriminative loss function, and the discriminator is trained to maximize the discriminative loss function $L_D(\phi;\mu,\nu_\theta)$ to obtain the discriminability.

MCGAN benefits from the strong supervision lying in MSE, which provides more control over the gradient behavior during the training. In conventional GAN objective (2.9), the generator parameter θ is updated using the following scheme:

$$\theta_{n+1} = \theta_n - \lambda \mathbb{E}_{z\sim\mu_z}\left[h'(G^{\theta_n}(z))\nabla_\theta G^\theta(z)^T|_{\theta=\theta_n}\cdot\nabla_x D^\phi(G^{\theta_n}(z))\right], \tag{5.20}$$

where λ is the learning rate, and h is a continuous function depending on the chosen discriminative loss. This updating scheme only allows the generator to update as much as its parametric family and update step permits. The generator parameter θ will only stop updating when the generator and the discriminator both reach the equilibrium point (θ^*,ϕ^*) where $\nabla_x D^\phi(x) = 0, \forall x \in \operatorname{supp}\mu$ [13]. However, reaching this equilibrium point (θ^*,ϕ^*) is challenging in practice. Instead, the sensitivity $\nabla_x D^\phi$ tends to fluctuate throughout the training process and increase in its norm as the discriminator overfits. As a result, the generator struggles to reach the optimal parameter θ^* and gets stuck with undamped oscillations due to its reliance on the sensitivity $\nabla_x D^\phi$.

In the case of MCGAN, the updating scheme of generator parameter θ can be derived as:

$$\begin{aligned}\theta_{n+1} =& \theta_n - \lambda\frac{\partial L_G}{\partial\theta}|_{\theta=\theta_n}\\ =& \theta_n - 2\lambda\left(\mathbb{E}_\mu[D^\phi(X)] - \mathbb{E}_{\nu_{\theta_n}}[D^\phi(X)]\right)H(\theta_n,\phi),\end{aligned} \tag{5.21}$$

where

$$H(\theta,\phi) = \mathbb{E}_{z\sim p(z)}[\nabla_\theta G^\theta(z)^T\cdot\nabla_x D^\phi(G^\theta(z))].$$

Here the gradient contains information about not only $\nabla_x D^\phi(x)$ but also the discrepancy between the expected discriminator scores under two measures μ and ν_θ. In this case, even if (θ,ϕ) fails to reach the equilibrium point where $\nabla_x D^\phi(x) = 0, \forall x \in \operatorname{supp}\mu$, it is possible for the generator to reach its optimum by aligning the expected discriminator scores under μ and ν_θ. The discriminability defined in Definition 5.2.2 can guarantee that optimizing the MSE given a discriminator of discriminability can lead to $\mu = \nu_\theta$.

Now we claim the optimality of $\mu = \nu_\theta$ in the following theorem.

Theorem 5.3.1. Let $\phi'_{\cdot,\cdot} : \mathcal{P}(\mathcal{X}) \times \mathcal{P}(\mathcal{X}) \to \Phi$ be a parameterization map such that $D^{\phi'_{\cdot,\cdot}} : \mathcal{P}(\mathcal{X}) \times \mathcal{P}(\mathcal{X}) \times \mathcal{X} \to \mathbb{R}$ has discriminability, i.e. $D^{\phi'_{\cdot,\cdot}} \in \mathcal{D}_{\text{Dis}}$. If θ^* is a global minimizer of $L_G(\theta; \phi'_{\mu,\nu_\theta}, \mu)$ defined in (5.19), then $\nu_{\theta^*} = \mu$.

Proof. Given $\phi \in \Phi$, consider the following function

$$f(\xi;\phi) = \mathbb{E}_\mu[(D^\phi(X) - \xi)^2].$$

It is easy to verify that its global minimizer satisfies that

$$\arg\min_\xi f(\xi;\phi) = \mathbb{E}_\mu[D^\phi(X)],$$

and it is unique due to strict convexity. If θ^* is a global minimizer of $L_G(\theta;\phi,\mu)$, we must have

$$\mathbb{E}_{\nu_{\theta^*}}[D^\phi(X)] = \mathbb{E}_\mu[D^\phi(X)]. \tag{5.22}$$

Given a parameterization map $\phi'_{\cdot,\cdot} : \mathcal{P}(\mathcal{X}) \times \mathcal{P}(\mathcal{X}) \to \Phi$ and $D^{\phi'_{\cdot,\cdot}} \in \mathcal{D}_{\text{Dis}}$, if θ^* is a global minimizer of $L_G(\theta; \phi'_{\mu,\nu_\theta}, \mu)$, it should also satisfy that

$$\mathbb{E}_{\nu_{\theta^*}}[D^{\phi'_{\mu,\nu_{\theta^*}}}(X)] = \mathbb{E}_\mu[D^{\phi'_{\mu,\nu_{\theta^*}}}(X)]. \tag{5.23}$$

Since $D^{\phi'_{\cdot,\cdot}} \in \mathcal{D}_{\text{Dis}}$,

$$\begin{aligned}
&\mathbb{E}_{\nu_\theta}[D^{\phi'_{\mu,\nu_\theta}}(X)] - \mathbb{E}_\mu[D^{\phi'_{\mu,\nu_{\theta^*}}}(X)] \\
=&\int_{\mathcal{A}^{\mu,\nu_\theta}} D^{\phi'_{\mu,\nu_\theta}}(x)(p_\mu(x) - p_{\nu_\theta}(x))dx \\
\neq&0,
\end{aligned} \tag{5.24}$$

for every different μ and ν_θ. Hence equality (5.23) holds if and only if $\nu_{\theta^*} = \mu$, which completes the proof. □

Theorem 5.3.1 claims that to learn the data distribution μ, we do not restrict the discriminator to reach its optimum; the discriminability is sufficient. This again benefits from the strong supervision provided by MSE.

In this scenario, the gradient of θ incorporates additional information beyond the discriminator's sensitivity with respect to the input. It also captures the discrepancy between the expected discriminator scores under μ and ν_θ. Thanks to the discriminability, this discrepancy provides useful feedback on the difference between ν_θ and μ. Even if the discriminator is not optimal or (θ, ϕ) is not at the equilibrium point, the generator can still learn the target distribution by minimizing this discrepancy.

5.3.1 Relation to f-divergence

As mentioned before, minimizing the MSE results in matching the expected discriminator scores under real and fake measures as in (5.23). We claimed in Theorem 5.3.1 that given a discriminator of discriminability, matching the expected discriminator scores yields the match of μ and ν_θ. However, it is not clear what this discrepancy between expected real and fake discriminator scores actually means in the context of GAN training. In this section, we explore its relation to f-divergence by specifying the discriminator.

As mentioned in Section 2.2.1, the optimal discriminator for BCE loss is given by:

$$D^{\phi^*}(x) = \frac{p_\mu(x)}{p_\mu(x) + p_{\nu_\theta}(x)}. \tag{5.25}$$

Here we omit μ and ν_θ in the notation of D^{ϕ^*} for conciseness. Given this optimal discriminator, the objective function in (2.5) can be interpreted as *Jensen-Shannon divergence* between μ and ν_θ, subtracting a constant term $\log(4)$ as shown in (2.7). The generator is therefore trained to minimize the Jensen-Shannon divergence. Similarly, Mao et al. [38] showed that optimizing LSGANs yields minimizing the Pearson χ^2 divergence between real and fake measures.

Given (5.25), we are also able to explore the connection between MCGAN and f-divergence. As proven in Lemma 5.3.1, when considering the optimal discriminator (5.25), the difference between real and fake expected discriminator output in (5.21) can be interpreted as an f-divergence between μ and $\bar{\nu}$, where $\bar{\nu} := \frac{(\mu+\nu)}{2}$

represents the averaged measure with density $p_{\bar{\nu}}(x) := \frac{p_\mu(x)+p_{\nu_\theta}(x)}{2}$.

Lemma 5.3.1. Given the optimal discriminator in (5.25), optimizing the MCGAN objective (5.19) is equivalent to minimizing the square of f-divergence:

$$\nabla_\theta L_G(\theta;\phi^*) = \nabla_\theta [\mathrm{Div}_f(\mu|\bar{\nu})]^2, \tag{5.26}$$

where $f(x) = x(x-1)$.

Proof. Given the optimal discriminator in (5.25), we have

$$\mathbb{E}_\mu[D^{\phi^*}(X)] - \mathbb{E}_{\nu_\theta}[D^{\phi^*}(X)] = \int_{\mathrm{supp}\,\mu \cup \mathrm{supp}\,\nu_\theta} \frac{p_\mu(x)}{p_\mu(x)+p_{\nu_\theta}(x)}(p_\mu(x)-p_{\nu_\theta}(x))dx.$$

Let $\bar{\nu} := \frac{\mu+\nu_\theta}{2}$ be the averaged measure defined on $\mathrm{supp}\,\mu \cup \mathrm{supp}\,\nu_\theta$, then we have

$$\begin{aligned}
\mathbb{E}_\mu[D^{\phi^*}(X)] - \mathbb{E}_{\nu_\theta}[D^{\phi^*}(X)] &= \int_{\mathrm{supp}\,\mu \cup \mathrm{supp}\,\nu_\theta} \frac{p_\mu(x)}{2p_{\bar{\nu}}(x)} \frac{(p_\mu(x)-p_{\nu_\theta}(x))}{p_{\bar{\nu}}(x)} p_{\bar{\nu}}(x)dx \\
&= \int_{\mathrm{supp}\,\mu \cup \mathrm{supp}\,\nu_\theta} \frac{p_\mu(x)}{p_{\bar{\nu}}(x)}\left(\frac{p_\mu(x)}{p_{\bar{\nu}}(x)} - 1\right) p_{\bar{\nu}}(x)dx \\
&= \int_{\mathrm{supp}\,\mu \cup \mathrm{supp}\,\nu_\theta} f\left(\frac{p_\mu(x)}{p_{\bar{\nu}}(x)}\right) p_{\bar{\nu}}(x)dx \\
&= \mathrm{Div}_f(\mu\|\bar{\nu}),
\end{aligned}$$

where $f(x) := x(x-1)$ is a convex function and $f(1) = 0$. Therefore, $\mathrm{Div}_f(\mu\|\bar{\nu})$ is a well-defined f-divergence. Furthermore, we can observe that the gradient of the generator objective function $L_G(\theta;\phi^*)$ equals the gradient of the squared error between the expected discriminator scores under real and fake measures:

$$\begin{aligned}
\nabla_\theta L_G(\theta;\phi^*) &= \nabla_\theta \mathbb{E}_\mu \left| D^{\phi^*}(X) - \mathbb{E}_{\nu_\theta}[D^{\phi^*}(X)]\right|^2 \\
&= \nabla_\theta \left|\mathbb{E}_\mu[D^{\phi^*}(X)] - \mathbb{E}_{\nu_\theta}[D^{\phi^*}(X)]\right|^2 \\
&= \nabla_\theta [\mathrm{Div}_f(\mu|\bar{\nu})]^2,
\end{aligned}$$

which completes the proof. □

Lemma 5.3.1 establishes a connection between MCGAN and f-divergence, illustrating that matching the expected scores is equivalent to minimizing the f-divergence between real and fake measures.

5.3.2 Improved stability

Incorporating MSE into the generative loss function has more advantages than just optimality.

Stability is a well-known issue in GAN training, and it arises due to several factors. The non-saturating loss was first proposed in [35] to address the gradient vanishing problem where the generator struggles to learn when the discriminator becomes highly accurate. While the non-saturating loss allows the generator to receive a non-zero gradient when the discriminator is accurate, it still suffers from severe instability issue that has been widely experienced in practice. Arjovsky et al. [36] provides insights into this instability issue. The instability is analyzed by modeling the inaccurate discriminator as an optimal discriminator perturbed by a centered Gaussian process. Given this noisy version of the optimal discriminator, it can be shown that the gradient of non-saturating loss follows a centered Cauchy distribution with infinite mean and variance, which leads to massive and unpredictable updates of the generator parameter. Hence it can be regarded as the source of training instability.

By following the same idea, we can prove that the proposed generative loss function in MCGAN can overcome the instability issue. However, we would not restrict the random noise of the discriminator to be Gaussian, which is a rather strong assumption.

Theorem 5.3.2. Let D^{ϕ^*} be the optimal discriminator for the given discriminative loss L_D. Furthermore, let us consider the noisy version of the optimal discriminator $D^{\phi_\varepsilon}(x)$ defined as follows:

$$D^{\phi_\varepsilon}(x) = D^{\phi^*}(x) + \varepsilon_1(x)$$

and

$$\nabla_x D^{\phi_\varepsilon}(x) = \nabla_x D^{\phi^*}(x) + \varepsilon_2(x),$$

$\forall x \in \mathcal{X}$, where $\varepsilon_1(x)$ and $\varepsilon_2(x)$ are two uncorrelated and centered random noise that are indexed by x and independent for every x.[1] Now, given $L_G(\theta;\phi)$ defined in (5.19), the following holds:

(a) $\mathbb{E}[\nabla_\theta L_G(\theta;\phi_\varepsilon)] = \nabla_\theta L_G(\theta;\phi^*)$;

(b) The variance of $\nabla_\theta L_G(\theta;\phi_\varepsilon)$ depends on the difference between μ and ν_θ. Specifically, when $\nu_{\theta^*} = \mu$, we have $\nabla_\theta L_G(\theta^*;\phi_\varepsilon) = 0$ almost surely.

Proof. Since $D^{\phi_\varepsilon}(x) = D^{\phi^*}(x) + \varepsilon_1(x)$ and $\nabla_x D^{\phi_\varepsilon}(x) = \nabla_x D^{\phi^*}(x) + \varepsilon_2(x)$, we have

$$\begin{aligned}
\nabla_\theta L_G(\theta;\phi_\varepsilon) &= \Big(\mathbb{E}_{\nu_\theta}[D^{\phi_\varepsilon}(X)] - \mathbb{E}_\mu[D^{\phi_\varepsilon}(X)]\Big) H(\theta,\phi_\varepsilon) \\
&= \Big(\mathbb{E}_{\nu_\theta}[D^{\phi^*}(X)] - \mathbb{E}_\mu[D^{\phi^*}(X)] + \mathbb{E}_{\nu_\theta}[\varepsilon_1(X)] - \mathbb{E}_\mu[\varepsilon_1(X)]\Big) H(\theta,\phi_\varepsilon) \\
&= (\Delta(\theta,\phi^*) + \bar{\varepsilon}_1(\theta)) H(\theta,\phi_\varepsilon),
\end{aligned}$$

where

$$\Delta(\theta,\phi) = \mathbb{E}_{\nu_\theta}[D^\phi(X)] - \mathbb{E}_\mu[D^\phi(X)]$$

and

$$\bar{\varepsilon}_1(\theta) = \mathbb{E}_{\nu_\theta}[\varepsilon_1(X)] - \mathbb{E}_\mu[\varepsilon_1(X)].$$

Because

$$\begin{aligned}
H(\theta,\phi_\varepsilon) =& \mathbb{E}_{z\sim\mu_z}[(\nabla_\theta G^\theta(z))^T \cdot \nabla_x D^{\phi_\varepsilon}(G^\theta(z))] \\
=& H(\theta,\phi^*) + \mathbb{E}_{z\sim\mu_z}[(\nabla_\theta G^\theta(z))^T \cdot \varepsilon_2(\hat{x})] \\
=& H(\theta,\phi^*) + \bar{\varepsilon}_2(\theta),
\end{aligned}$$

where $\hat{x} = G^\theta(z)$ and

$$\bar{\varepsilon}_2(\theta) = \mathbb{E}_{z\sim\mu_z}[(\nabla_\theta G^\theta(z))^T \cdot \varepsilon_2(\hat{x})].$$

[1]This assumption of centered random noise is made due to the fact that as the approximation gets better, this error looks more and more like centered random noise due to the finite precision [36].

Hence we have

$$\begin{aligned}\nabla_\theta L_G(\theta;\phi_\varepsilon) =& \Delta(\theta,\phi^*)H(\theta,\phi^*)+\bar{\varepsilon}_1(\theta)H(\theta,\phi^*)+\Delta(\theta,\phi^*)\bar{\varepsilon}_2(\theta)+\bar{\varepsilon}_1(\theta)\bar{\varepsilon}_2(\theta)\\ =& \nabla_\theta L_G(\theta;\phi^*)+\bar{\varepsilon}_1(\theta)H(\theta,\phi^*)+\Delta(\theta,\phi^*)\bar{\varepsilon}_2(\theta)+\bar{\varepsilon}_1(\theta)\bar{\varepsilon}_2(\theta).\end{aligned}$$

Since $\bar{\varepsilon}_1(\theta)$ are linear combinations of centered random noises, it is also a centered noise. Similarly, $\bar{\varepsilon}_2(\theta)$ has zero mean as $\varepsilon_2(x)$ are independent for all x. Moreover, the expectation of $\bar{\varepsilon}_1(\theta)\bar{\varepsilon}_2(\theta)$ is also zero since $\varepsilon_1(x)$ and $\varepsilon_2(x)$ are uncorrelated for all x. Hence the mean of $\nabla_\theta L_G(\theta;\phi_\varepsilon)$ equals to $\nabla_\theta L_G(\theta;\phi^*)$. By the definition of $\Delta(\theta,\phi)$ and $\bar{\varepsilon}_1(\theta)$, its variance also depends on the difference between μ and ν_θ, which completes the proof. $\square$

Theorem 5.3.2 implies that given the noisy discriminator D^{ϕ_ε}, the expected value of the gradient of $L_G(\theta;\phi_\varepsilon)$ in (5.19) is the accurate gradient given by the optimal discriminator. More importantly, its variance is determined by the discrepancy between μ and ν_θ: as ν_θ approaches μ, the variance of $\bar{\varepsilon}_1(\theta)$ and $\Delta(\theta,\phi^*)\bar{\varepsilon}_2(\theta)$ tend to 0, indicating better training stability compared with non-saturating loss.

In summary, the MCGAN algorithm offers the following two key advantages in terms of generator training:

- **Relaxed Requirement for Optimal Discriminator:** The MCGAN can relax the requirement for the optimal discriminator. Instead, it focuses on the discriminability of the discriminator described in Definition 5.2.2. This relaxation allows the generator to successfully learn the real data distribution without relying on the discriminator being optimal.

- **Improved Stability of Gradient**: MCGAN introduces an innovative generative loss function that provides more reliable supervision for generator training. It improves the training stability by controlling the variance of the generator gradient that is directly related to the difference between the real and fake measures.

Overall, the relaxation of the optimal discriminator requirement and the improved training stability through the generative loss function make MCGAN a powerful and effective framework for training generators in GANs. By leveraging discriminability and providing reliable supervision, MCGAN enables the generator to learn the desired data distribution more efficiently and achieve better performance.

5.3.3 A toy example: Dirac-GAN

To better demonstrate the advantages of MCGAN, we take Dirac-GAN [13] as an example.

Dirac-GAN was a simple yet representative counterexample employed to illustrate the instability of unregularized GAN training [13, 69]. In Dirac GAN, the fake measure induced by the generator is defined as $p_{\nu_\theta}(x) = \delta(x-\theta)$ where $\delta(\cdot)$ is the Dirac delta function, and the discriminator is defined as $D^{\phi}(x) = \phi x$. Here, θ and ϕ are the parameters of the generator and discriminator respectively. The target distribution is given by Dirac distribution where $p_{\mu}(x) = \delta(x-c)$ with $c = 0$. In this case, the unique equilibrium point of the training objective is given by $\phi = \theta = 0$.

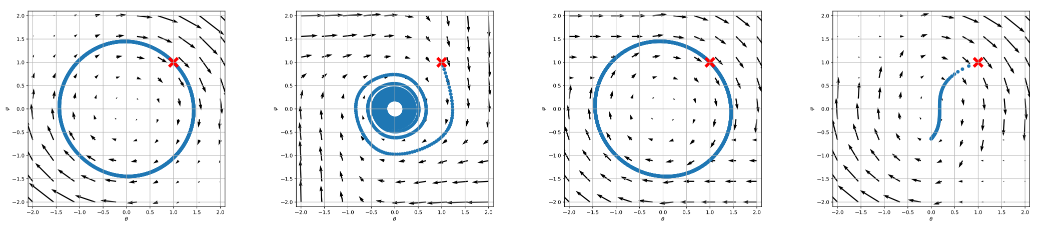
For unregularized GAN training, the updating scheme of training objective (2.9) can be derived as:

$$\begin{aligned} \phi_{n+1} &= \phi_n + \lambda f_2'(\phi_n\theta_n)\theta_n, \\ \theta_{n+1} &= \theta_n - \lambda h'(\phi_n\theta_n)\phi_n. \end{aligned} \tag{5.27}$$

For MCGAN, the updating scheme of the training objective (5.19) is given by:

$$\begin{aligned} \phi_{n+1} &= \phi_n + \lambda f_2'(\phi_n\theta_n)\theta_n, \\ \theta_{n+1} &= \theta_n - 2\lambda(\phi_n\theta_n - \phi_n c)\phi_n. \end{aligned} \tag{5.28}$$

In the unregularized GAN training, Figure 5.2 demonstrates that (a) GAN, (b) NSGAN, and (c) Geometric GAN, which utilize *binary cross-entropy loss*, *non-saturating loss* and *hinge loss* respectively, all fail to converge to the equilibrium point $(0,0)$. This failure can be attributed to the dependence on the value of ϕ in the updating scheme. When ϕ fails to converge to zero, θ keeps updating even if it has already reached zero. In turn, the non-zero θ further encourages ϕ to update away from 0, resulting in an endless loop and the failure of both the generator and discriminator.



(a) GAN **(b)** NSGAN **(c)** Geometric GAN **(d)** MCGAN

Figure 5.2: Convergence behavior of the Dirac-GAN. The starting point of iteration is marked in red. The total number of iterations is 2000.

However, (d) MCGAN is able to address this issue by leveraging stronger supervision. Despite the failure of the discriminator, the generator parameter θ still successfully converged to the optimal value 0 as the difference term $\phi_n\theta_n - \phi_n c$ in (5.28) provides additional information to the generator, allowing it to justify how far the fake sample currently is from the real sample as perceived by the discriminator and adjust its parameter accordingly.

5.4 Related work

It is not the first time that the generative loss function has attracted researchers' attention. As already explored in the original GAN paper [35], the generative loss function in the original min-max objective (2.5) suffers from the saturating issue where optimal discriminator can cause zero gradients of generator almost everywhere. Non-saturating loss is therefore proposed to address this issue, and this alternative generative loss function has been shown to have better performance and convergence than the original proposal [70, 36]. However, the non-saturating loss still suffers from training instability and mode-dropping problems as it assigns an extremely low cost on mode-dropping [36]. Additionally, one main cause of the failure of convergence to Nash equilibrium is the lack of communication between the generator and discriminator. Both networks have to communicate with each other to enhance individual performance until reaching the equilibrium [71]. Therefore, Metz et al. [72] propose to model the dynamic of the discriminator parameter as a function of the generator parameter and design a new generator objective with respect to an k-step unrolled optimization of the discriminator. This allows the generator to capture the reaction of the discriminator toward a change in the generator, which hence reduces the tendency of the generator to engage in mode collapse. However, this method requires additional steps of forward and backward propagation, resulting in considerably increased computational costs. Different from those approaches mentioned above, our proposed MCGAN enhances the interaction between the discriminator and generator by incorporating MSE into the generative loss function, which is more straightforward to implement and has better training stability.

Interestingly, the concept of matching the expected discriminator scores was previously explored in [65]. It is shown that under certain conditions, those distributions that minimize the adversarial divergence to target distribution are exactly those that are indistinguishable from target distribution by all discriminator networks. In other words, minimizing those adversarial divergences in these GAN variants is equivalent to matching the expected discriminator scores under all discriminator networks, which is referred to as *generalized moment matching*. Our work explains in detail what kinds of benefits it can bring to GAN training by matching the expected discriminator scores. Specifically, we show that in order to learn the target distribution, it is unnecessary to match the expected discriminator scores under all discriminator networks or the optimal one, those with discriminability can already provide sufficient information to guide the generator training as shown in Theorem 5.3.1.

5.4.1 Relation to feature matching

In order to enhance the generative performance, a feature matching approach is proposed in [73], which adds to the generative loss function an additional cost that matches the statistic of the real and generated samples given by the activation on an intermediate layer of the discriminator. The generator hence is trained to generate fake samples that reflect the statistics (features maps) of real data rather than just maximizing its discriminator scores.

To be specific, suppose we have a feature map ψ that maps each $x \in \mathcal{X}$ to a feature vector $\psi(x) = (\psi_1(x), \psi_2(x), \ldots, \psi_n(x)) \in \mathbb{R}^n$ where each $\psi_i \in C_b(\mathcal{X})$, then the feature matching approach adds to the generative loss function an additional cost defined as:

$$R_{\mathrm{fm}}(\theta;\phi) = \|\mathbb{E}_{\mu}[\psi(x)] - \mathbb{E}_{\nu_\theta}[\psi(x)]\|_2^2. \tag{5.29}$$

Although empirical results already indicated the effectiveness of the feature matching method, it lacks a theoretical guarantee that minimizing the difference of features can help us reach the Nash equilibrium or optimality $\nu_\theta = \mu$. Hence (5.29) is commonly used as a regularization term rather than an individual loss function like the one proposed in our MCGAN.

In the case of MCGAN, if we construct the discriminator as a linear transformation of the feature map, i.e. $D^{\phi} = \phi^T(\psi(x), \mathbf{1})$ where $\phi \in \mathbb{R}^{n+1}$ is a linear functional. Given the novel generative loss function in (5.19), we have

$$\nabla_\theta L_G(\theta;\phi) = \nabla_\theta |\mathbb{E}_{\mu}[\phi^T(\psi(x), \mathbf{1})] - \mathbb{E}_{\nu_\theta}[\phi^T(\psi(x), \mathbf{1})]|^2.$$

Here, the generator is also trained to match the feature maps of real samples and fake samples, but in a weighted average way. By using the linear transformation on the feature map, the discriminator is trained to focus on the most relevant features and assign them larger weights while assigning relatively smaller weights to less important features. As a result, the generator is trained to match the feature maps more efficiently and effectively.

This weighted average matching approach facilitates the learning of the important features in the data distribution. Through the discriminator's linear transformation, the generator can better capture the salient attributes of the real data distribution, resulting in the generation of more authentic and high-quality samples.

5.5 MCGAN for conditional time-series generation

Now that we have discussed those advantages of MCGAN in the unconditional generation task, it is straightforward to apply the MCGAN framework to the conditional time-series generation task.

Let us first revisit the problem setting. Let $(\Omega, \mathcal{F}, (\mathcal{F}_t)_{t\geq 0}, \mathbb{P})$ be a filtered probability space, and $\{X_t\}_{t\in\mathbb{N}}$ a stationary stochastic process that takes value in $\mathbb{R}^d$. In this case, the target value is defined as future path $X_{\text{future},t} := (X_{t+1},\cdots,X_{t+q}) \in \mathbb{R}^{d\times q} := \mathcal{X}$, and the conditioning variable is the p-lagged values of X_t which we denote by $X_{\text{past},t} := (X_{t-p+1},\cdots,X_t) \in \mathbb{R}^{d\times p} := \mathcal{Y}$. We are interested in learning the conditional law given p-lagged value $X_{\text{past},t}$ which we denote by $\mu_{p,q}(y) := \text{Law}(X_{\text{future},t}|X_{\text{past},t} = y), \forall y \in \mathcal{Y}$. In other words, our goal is to train the generator such that the conditional law $\nu^{\theta}_{p,q}(y)$ induced by the generator can best approximate $\mu_{p,q}(y)$. For clarity, we omit the p and q values in $\mu_{p,q}(y)$ and $\nu^{\theta}_{p,q}(y)$, and use $\mu(y)$ and $\nu_\theta(y)$ instead.

Based on the MCGAN algorithm, we propose the following training objectives for the conditional time-series generation task:

$$\begin{aligned}
&\max_{\phi\in\Phi} L_D(\phi;\theta) = \mathbb{E}_\mu\left[f_1(D^\phi(X_{\text{future}},X_{\text{past}}))\right] + \mathbb{E}_{\nu_\theta}\left[f_2(D^\phi(X_{\text{future}},X_{\text{past}}))\right],\\
&\min_{\theta\in\Theta} L_G(\theta;\phi) = \mathbb{E}_\mu[|D^\phi(X_{\text{future}},X_{\text{past}}) - \mathbb{E}_{\nu^\theta(X_{\text{past}})}[D^\phi(X_{\text{future}},X_{\text{past}})]|^2],
\end{aligned} \tag{5.30}$$

where functions f_1 and f_2 depend on the choice of the discriminative loss function, and $\mathbb{E}_\mu$ denotes expectation taken over $(X_{\text{future}},X_{\text{past}}) \sim \mu$, $\mathbb{E}_{\mu(X_{\text{past}})}$ denotes conditional expectation taken over $X_{\text{future}} \sim \mu(X_{\text{past}})$. Here the subscript t is omitted for brevity. The discriminator $D^\phi : \mathcal{X}\times\mathcal{Y} \to \mathbb{R}$ is trained by a certain type of discriminative loss function $L_D(\phi;\theta)$ to obtain the discriminability described in Definition 5.2.2.

As discussed before, one advantage of using this generative loss function is that it requires no additional approximation to the conditionally expected discriminator scores of real samples as needed when computing conditional W_1 distance or conditional Sig-W_1 metric, which leads to more efficient and stable generator training in MCGAN. Similar to its unconditional version, conditional MCGAN has the ability to learn the target distribution with a relatively weaker discriminator and is capable of improving the training stability, which we would not bother to restate through additional proof.

Remark 5.5.1. The extension from unconditional GAN to conditional GAN is trivial. In a conditional GAN, the law of conditioning variable is fixed for both real and synthetic distribution. This means that modeling the conditional distribution in cGAN is, in essence, equivalent to modeling the joint distribution of the conditioning variable and the target variable. Consequently, all the result and insights obtained from the unconditional GAN can be naturally extended to the conditional case.

5.5.1 Algorithm

As described in Section 4.2.1, we first construct samples of past/future pair from a long realization of X in a rolling window fashion with fixed window size of $p+q$. For each training epoch, the MCGAN training process is divided into two phases: discriminator training and generator training. In the discriminator training phase, we first randomly sample a mini-batch of real past/future path pairs $\{(x^i_{\text{past}}, x^i_{\text{future}})\}_{i=1}^B$ from the training dataset where B is the batch size, then we compute the discriminative loss using these real past/future paths and update ϕ.

In the generator training phase, we again randomly sample another mini-batch of real past/future path pairs $\{(x^i_{\text{past}}, x^i_{\text{future}})\}_{i=1}^B$ from the training set. For each past path x^i_{past}, we apply the MC method to estimate the conditional expectation of discriminator scores of the generated future path given x^i_{past}. This estimate is denoted as $\hat{\mathbb{E}}_{\nu^\theta(x^i_{\text{past}})}[D^\phi(X_{\text{future}}, x^i_{\text{past}})]$. Then we compute the generative loss with this estimate, i.e. the MSE between the discriminator scores of real future path $D^\phi(x^i_{\text{future}}, x^i_{\text{past}})$ and conditionally expected discriminator scores of generated future path $\hat{\mathbb{E}}_{\nu^\theta(x^i_{\text{past}})}[D^\phi(X_{\text{future}}, x^i_{\text{past}})]$, and update the parameter of generator θ based on the generative loss. A flowchart of the generator training phase in the MCGAN algorithm is provided in Figure 5.3.

The pseudo-code of the MCGAN algorithm for conditional time-series generation task is presented in Algorithm 3.

5.6 Numerical experiments

To benchmark with MCGAN, we choose RCGAN for the conditional time-series generation task as presented in Section 4.4. To investigate the effectiveness of the

Algorithm 3 Algorithm of MCGAN

```
Input:
    $N$: the number of epochs;
    $N_D$: number of discriminator iterations per generator iteration;
    $B \in \mathbb{N}$: batch size;
    $N_{MC}$: the number of Monte Carlo samples;
    $f, g$: choice of discriminative loss;
    $\alpha_\phi, \alpha_\theta$: learning rate for discriminator and generator.
Output:
    $(\theta^*, \phi^*)$: approximation of the optimal parameters of the generator and discriminator.
 1: Initialize the model parameter $(\theta, \phi)$ of the generator $G$ and discriminator $D$.
 2: for $n = 1 : N$ do
 3:     for $n_d = 1 : N_D$ do
 4:         Sample batch $\{(x^i_{\text{past}}, x^i_{\text{future}})\}_{i=1}^B \sim p_\mu(X_{\text{past}}, X_{\text{future}})$
 5:         Generate fake paths $\{(x^i_{\text{past}}, \hat{x}^i_{\text{future}})\}_{i=1}^B \sim p_{\nu_\theta}(X_{\text{past}}, \hat{X}_{\text{future}})$
 6:         Compute discriminative loss:
                $$L_D(\phi;\theta) \leftarrow \frac{1}{B}\sum_{i=1}^B f(D^\phi(x^i_{\text{future}}, x^i_{\text{past}})) + \frac{1}{B}\sum_{i=1}^B g(D^\phi(\hat{x}^i_{\text{future}}, x^i_{\text{past}}));$$
 7:         Update parameter of discriminator:
                $$\phi \leftarrow \phi + \alpha_\phi \nabla_\phi L_D(\phi;\theta);$$
 8:     end for
 9:     Sample batch $\{(x^i_{\text{past}}, x^i_{\text{future}})\}_{i=1}^B \sim p_\mu(X_{\text{past}}, X_{\text{future}})$;
10:     Estimate the conditional expectation for each past path $x^i_{\text{past}}$:
            $$\hat{\mathbb{E}}_{\nu_\theta(x^i_{\text{past}})}[D^\phi(X_{\text{future}}, x^i_{\text{past}})] \leftarrow \frac{1}{N_{mc}}\sum_{j=1}^{N_{mc}} D^\phi(G^\theta(x^i_{\text{past}}, z^{(j)}), x^i_{\text{past}});$$
11:     Compute generative loss:
            $$L_G(\theta;\phi) \leftarrow \frac{1}{B}\sum_{i=1}^B (D^\phi(x^i_{\text{future}}, x^i_{\text{past}}) - \hat{\mathbb{E}}_{\nu_\theta(x^i_{\text{past}})}[D^\phi(X_{\text{future}}, x^i_{\text{past}})])^2;$$
12:     Update parameter of generator:
            $$\theta \leftarrow \theta - \alpha_\theta \nabla_\theta L_G(\theta;\phi);$$
13: end for
```

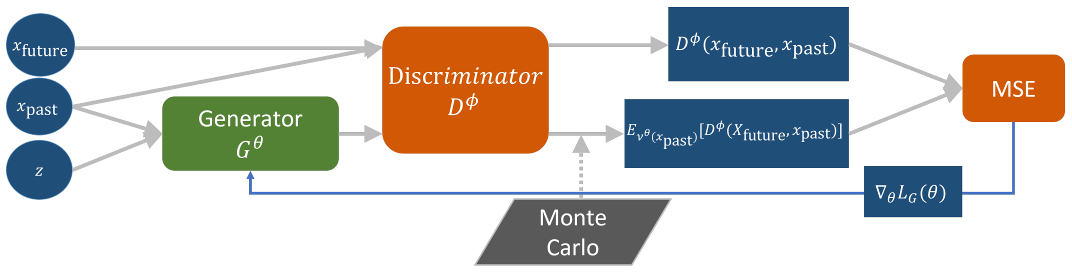

x_{future}
x_{past}
z
Generator G^θ
Discriminator D^ϕ
$D^\phi(x_{\text{future}}, x_{\text{past}})$
$E_{\nu^\theta(x_{\text{past}})}[D^\phi(X_{\text{future}}, x_{\text{past}})]$
MSE
$\nabla_\theta L_G(\theta)$
Monte Carlo

Figure 5.3: Flowchart of generator training phase in MCGAN algorithm.

proposed generative loss function in MCGAN, both MCGAN and RCGAN use the same discriminative loss, i.e., binary cross-entropy loss. In both cases, the discriminator scores of real and fake samples are first computed, and then these discriminator scores are used to compute the binary cross-entropy loss and update the discriminator parameters. The only difference between MCGAN and RCGAN is the generator training phase where RCGAN will update the generator by minimizing BCE while MCGAN computes the MSE by following the procedure described in Algorithm 3. For all experiments, the Monte-Carlo size used in MCGAN implementation is set to 1000. We also include GMMN, TimeGAN, and SigCWGAN as baseline models as did in Section 4.4. The hyper-parameter setting of each dataset can be found in Section C.1.

In terms of test metrics, we again employ three main criteria:(a) the marginal distribution of time series; (b) the temporal and feature dependence; (c) the usefulness - synthetic data should be as useful as the real data when used for the same predictive purposes (i.e. train-on-synthetic, test-on-real). The definition of these test metrics can be found in Section 4.4.

5.6.1 Vector autoregressive model

To demonstrate the model's ability to generate realistic multi-dimensional time series in a controlled environment, we again consider synthetic data generated by the Vector Autoregressive (VAR) model as we did in Section 4.4.1.

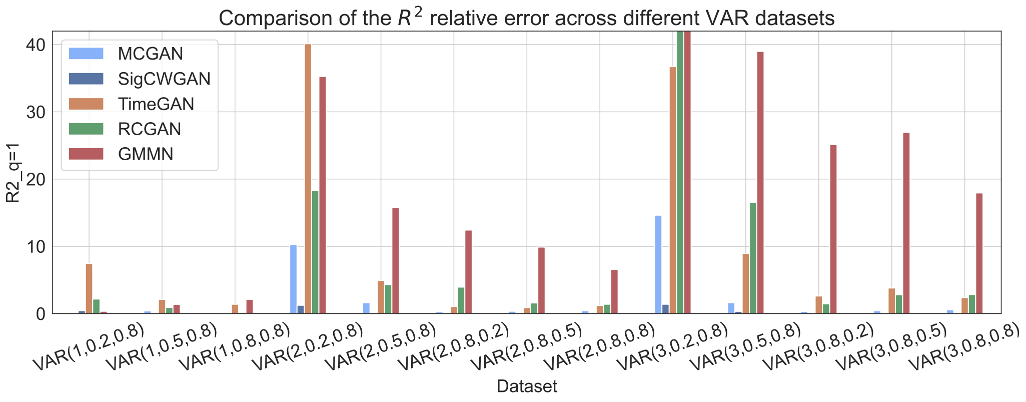
In our benchmark, we investigate the dimension $d = 1,2,3$ and various (ϕ, σ). Across all dimensions, we observe that the MCGAN considerably improves the performance of RCGAN in terms of all the test metrics. Specifically, MCGAN considerably outperforms RCGAN in terms of the predictive discriminator score. As shown in Figure 5.4a, compared with RCGAN, MCGAN hugely improves the R^2 error in all parameter sets. When $d = 3, \phi = 0.5, \sigma = 0.8$, the MCGAN can reduce the R^2 relative error from 16.5% in RCGAN to only 1.6%. The performance gap is rather huge in the multi-dimensional case. Figure 5.4b shows the correlation metrics of all baselines. The MCGAN achieves a much lower correlation error while RCGAN appears not to fully capture the cross-correlation in the three-dimensional case. A similar pattern can be observed in the comparison of ACF metric and density metric illustrated in Figure 5.4c and Figure 5.4d.

Moreover, we validate the improved training stability of MCGAN when compared with RCGAN. Figure 5.5 shows the development of the norm of generator gradient, abs metric, ACF metric, and cross-correlation metric through the course of training for the 3-dimensional VAR(1) model. Observe that the generator gradient in RCGAN oscillates heavily during the training process and ends up with a non-zero gradient at the end of training. However, MCGAN has a much more stable gradient dynamic, and thanks to the difference term in (5.19), the gradient converges nicely toward zero at the end of the training, which indicates better training behavior. In terms of the other three metrics, although the dynamics of MCGAN also oscillate at the beginning, they become increasingly stable as the generator gets accurate and converges to a better level than that of RCGAN, which again highlights the stability and usefulness of the newly proposed generative loss (5.19) in MCGAN.

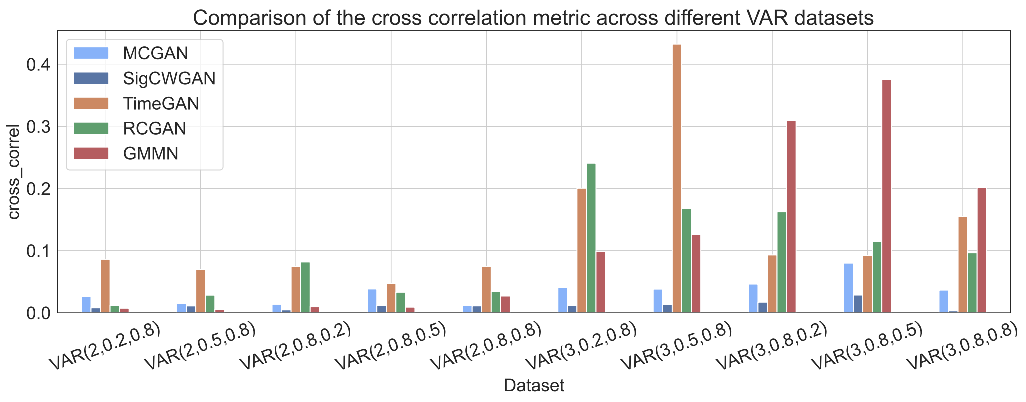
Apart from that, we also show the effectiveness of MCGAN in capturing the target conditional distribution. Figure 5.6 illustrates that similar to SigCWGAN, the MCGAN has a better fitting than other baselines in terms of conditional law as the estimated mean (and standard deviation) is closer to that of the true model compared with other baselines. More details of the numerical results including long-term ACF plots of synthetic data can be found in Appendix C.2

5.6.2 ARCH model

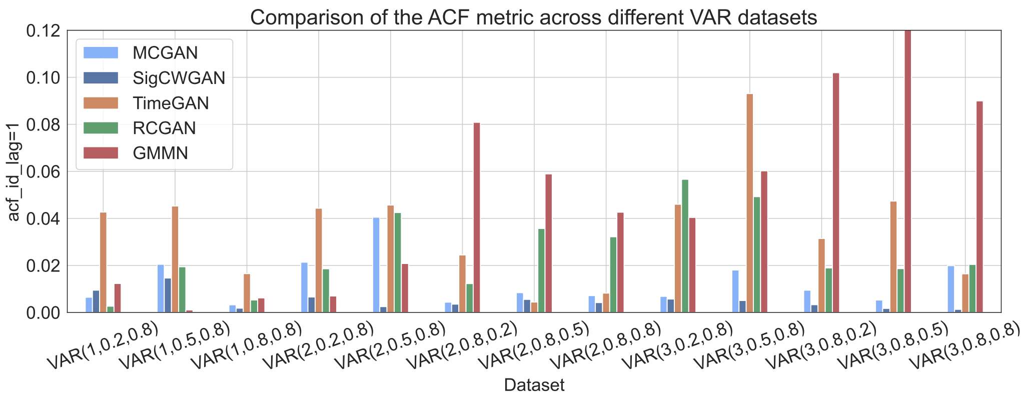
We also implement extensive experiments on ARCH(k) with different $k-$lag values, i.e. $k \in \{2, 3, 4\}$ as in Section 4.4.2. The numerical results are summarized in Ta-



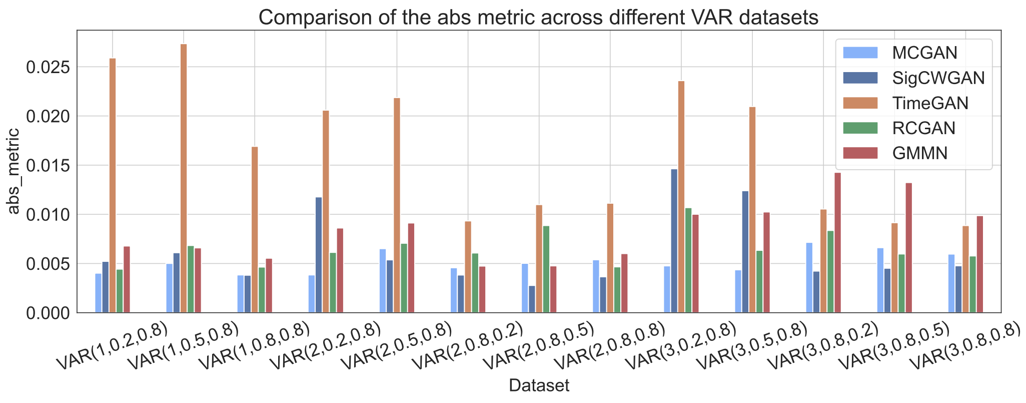
(a) R^2 relative error.



(b) Cross-correlation metric.



(c) ACF metric.



(d) Abs metric.

Figure 5.4: Comparison of the performance across all parameter sets (d, ϕ, σ) and benchmarks.

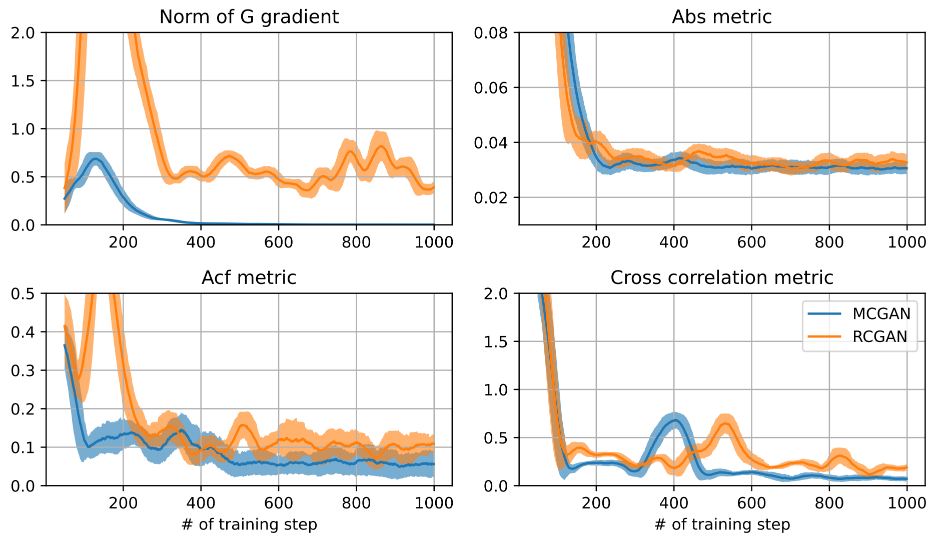


Figure 5.5: Comparison of training dynamics of MCGAN (blue) and RCGAN (orange). All four dynamics: (1) l^2 norm of generator gradient; (2) abs metric;(3) ACF metric;(4) cross-correlation metric is computed via the moving average (solid line) and standard deviation (shaded area) using window size 50. The training dataset is synthesized from VAR(1) model for $d = 3$, $\phi = 0.2$ and $\sigma = 0.8$.

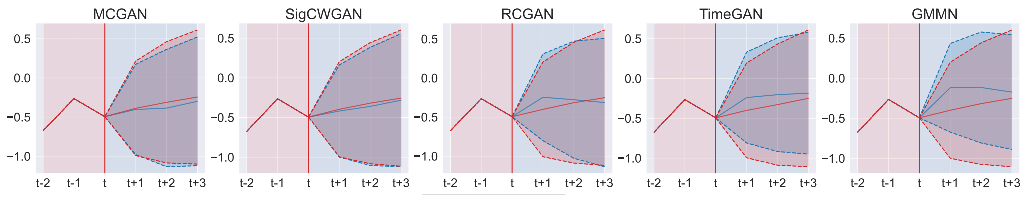


Figure 5.6: Comparison of all models' performance in fitting the conditional distribution of future time series given one past path sample. The real and generated paths are plotted in red and blue respectively with the shaded area as the 95% confidence interval. The training dataset is synthesized from VAR(1) model for $d = 3$, $\phi = 0.8$ and $\sigma = 0.8$.

ble D.6. Although RCGAN achieves better cross-correlation metrics than MCGAN, it suffers from much worse predictive performance and auto-correlation.

The distribution metrics (abs metric) of models across different k values are plotted in Figure 5.7. Across all k values, we observe that the MCGAN outperforms all baseline models including SigCWGAN in terms of the abs metric. In particular, we find that RCGAN attains an abs metric of 0.01113 when $k = 3$. By using MSE along with the Monte-Carlo method, MCGAN is able to achieve a value of 0.00489 in terms of the abs metric, which reduces the RCGAN abs metric by more than half. This showcases the effectiveness of MCGAN in capturing the target distribution.

Similarly, in the R^2 comparison for different k values plotted in Figure 5.7, we can see that for all different k values, MCGAN consistently outperforms RCGAN in terms of R^2 relative error, highlighting its improvement on predictive power when compared with RCGAN. The complete numerical results are summarized in Table D.6. The best results among all the models are highlighted in bold.

In Figure 5.8, we also compare the ACF/PACF plot of the squared residuals generated by each model. While MCGAN exhibits some proficiency in capturing temporal dependencies at small lags, our observations reveal a limitation in its performance when considering lags greater than 5. Ideally, the ACF/PACF at larger lags should approach zero. However, MCGAN deviates from this expectation, displaying notable values at larger lags. In this case, MCGAN falls short compared to other models concerning the ACF/PACF of squared residuals.

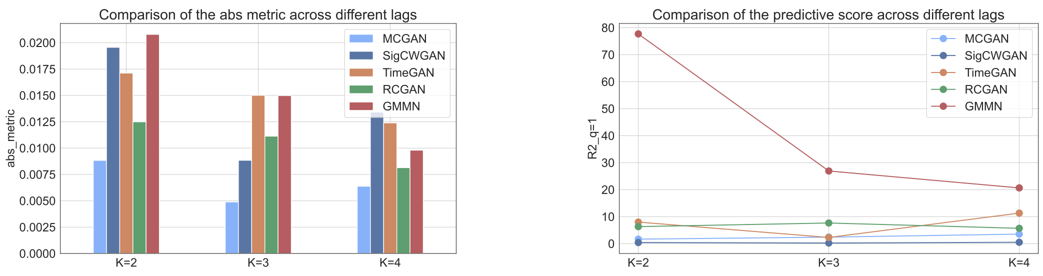
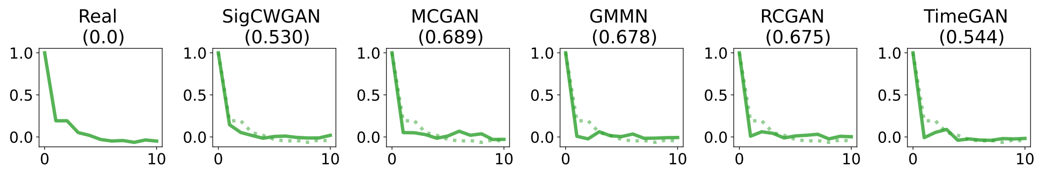


Figure 5.7: (Left) The distributional metric (abs metric) comparison; (Right) the R^2 error (TSTR) comparison using ARCH(k) data for $k = 2, 3, 4$.

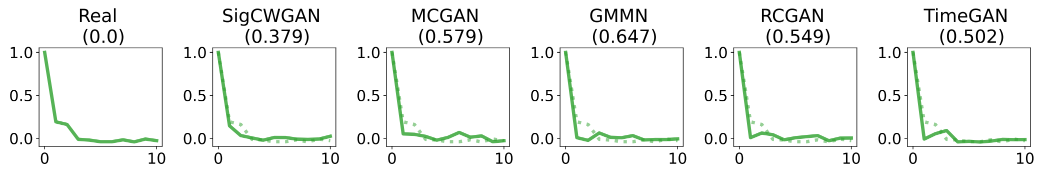
5.6.3 Stock dataset

As demonstrated in Section 4.4.3, we also assess the performance of MCGAN on the empirical data. We train the model on the log return of the close prices and the log of realized volatility of the S&P 500 index (SPX) and Dow Jones index (DJI), which is retrieved from the Oxford-Man Institute's "realized library". Data is normalized before being fed into the model.

Table 5.2 shows that MCGAN achieves superior or comparable performance to the other baselines. For the SPX-only data, MCGAN performs better than RCGAN in terms of the fitting of marginal distribution and feature correlation. Specifically, MCGAN achieves 0.05641 of correlation metric which is 40% lower than 0.0926



(a) ACF of squared residuals.



(b) PACF of squared residuals.

Figure 5.8: ACF/PACF plot of squared residuals for ARCH(2) model. Here x-axis represents the lag value (with a maximum lag equal to 10) and the y-axis represents the corresponding auto- correlation/partial auto-correlation. The length of the real/generated time series used to compute the ACF is 1000. The number in the bracket under each model is the sum of the absolute difference between the correlation coefficients computed from real (dashed line) and generated (solid line) samples.

obtained by RCGAN. When the MCGAN is outperformed by RCGAN (for example, abs metric and ACF discriminator score for SPX and DJI data), the difference is not significant.

Metrics	abs metric	ACF metric	correlation	R^2 error(%)	Sig-W_1
MCGAN	**0.00915**,0.00937	0.03665,0.03386	0.05641,0.12938	4.20975,2.93112	**4.73513**,5.19871
RCGAN	0.01134,0.00933	0.04192,0.03367	0.0926,0.13699	5.06443,5.4847	4.74862,5.20083
SigCWGAN	0.01468,**0.00879**	**0.02693**,0.03988	**0.00638**,**0.10354**	6.25606,5.28866	4.74507,**5.18462**
TimeGAN	0.01064,0.01136	0.0341,**0.03147**	0.0101,0.11342	**3.79664**,**2.90929**	4.75276,5.22316
GMMN	0.00986,0.01438	0.05483,0.06668	0.07498,0.28339	11.37772 ,13.7324	4.75251,5.19631
GARCH	0.01583,0.01670	0.05392, 0.05337	0.15791, 0.7290	12.1253, 12.5686	4.75825, 5.25344

Table 5.2: Numerical results of the stock datasets. In each cell, the left/right numbers are the results for the SPX data/ the SPX and DJI data respectively. We use the relative error of TSTR R^2 against TRTR R^2 as the R^2 metric.

The advantages of MCGAN are highlighted in terms of predictive discriminator scores. As shown in Figure 5.9, MCGAN consistently achieves better R^2 error (mostly lower than 4%) of different lags compared to RCGAN. Although TimeGAN has the best R^2 error at lag 1, its performance deteriorates fast as the lag increases, which indicates its poor ability to generate realistic longer time series.

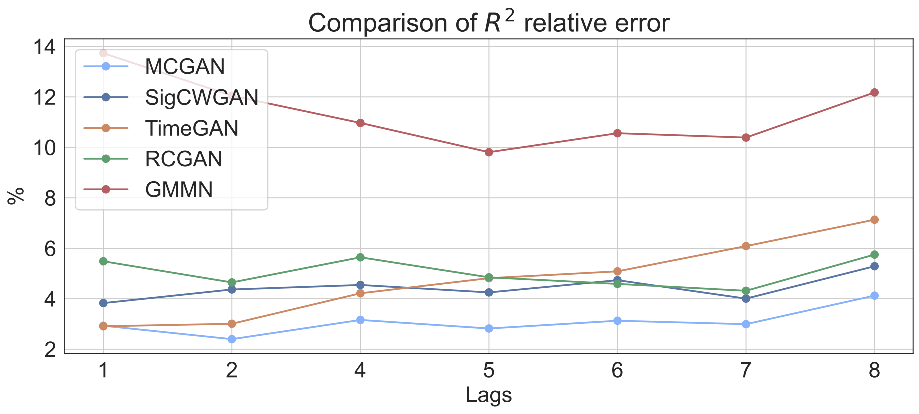
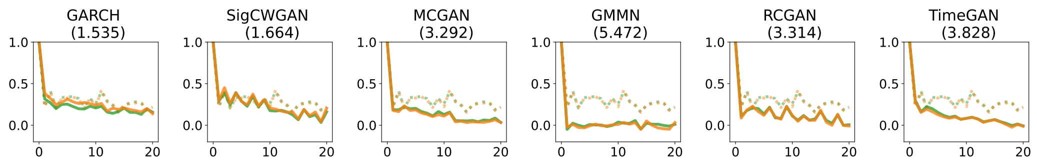


Figure 5.9: Comparison of R^2 relative error (%) with different lags. All models are trained on SPX and DJI data.

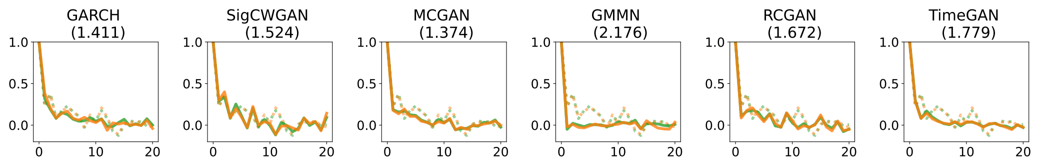
In Figure 5.10, we also present a comparison of the ACF/PACF of synthetic absolute and squared returns. We can see from Figure 5.10 that when compared with RCGAN, MCGAN achieved considerable improvement in capturing the temporal dependency of absolute and squared returns. To be specific, the MCGAN demonstrates a decent fitting of ACF/PACF of both absolute and squared returns at small lags, outperforming RCGAN in this regard. It is important to note, however, that both MCGAN and RCGAN struggle to capture the long-term ACF of both squared and absolute returns.

In the context of the coverage ratio test of SPX/DJI returns, We have summarized the results of the coverage ratio test in Table 5.3. Notably, MCGAN achieves the second-best ratio for 1-day ahead returns, with a result of 4.29% for both SPX and DJI. Furthermore, MCGAN excels by achieving the best ratio for 5-day ahead returns, precisely 4.02%, equaling that of SigCWGAN. These results indicate that, in contrast to SigCWGAN, which exhibited a poor VaR estimate for 1-day ahead returns, MCGAN demonstrates decent 95% VaR estimates for both 1-day and 5-day ahead returns.

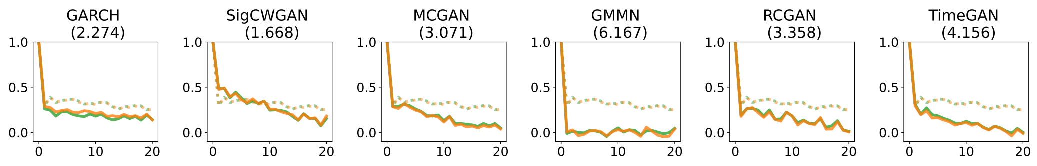
In Figure 5.11, we plot the marginal distribution of the generated data by MCGAN compared with that of real SPX and DJI data, which shows that MCGAN is able to generate realistic synthetic data of the SPX and DJI data. The example path



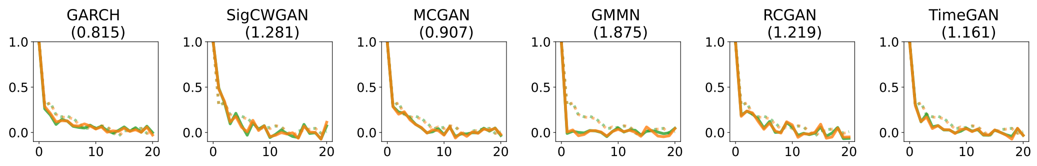
(a) ACF of squared returns.



(b) PACF of squared returns.



(c) ACF of absolute returns.



(d) PACF of absolute returns.

Figure 5.10: ACF/PACF plot of absolute and squared returns for SPX (green line) and DJI (orange line). Here x-axis represents the lag value (with a maximum lag equal to 10) and the y-axis represents the corresponding auto-correlation/partial auto-correlation. The length of the real/generated time series used to compute the ACF is 1000. The number in the bracket under each model is the sum of the absolute difference between the correlation coefficients computed from real (dashed line) and synthetic (solid line) samples.

of generated log returns can be found in Figure D.11. Other details of numerical results can also be found in Appendix C.4.

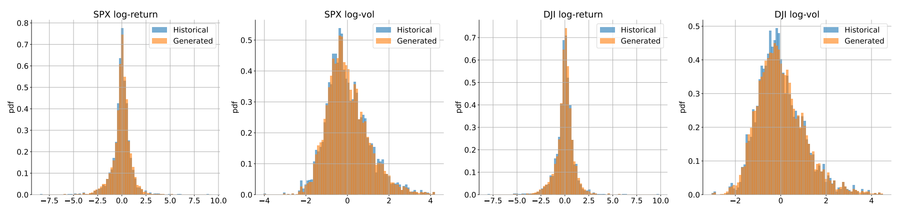


Figure 5.11: Comparison of the marginal distributions of the generated MCGAN paths and the SPX and DJI data.

Models	Ratio of 1-day ahead returns	Ratio of 5-day ahead returns
SigCWGAN	8.04%,9.11%	3.62%,**4.02**%
MCGAN	4.29%,4.29%	3.35%,**4.02**%
TimeGAN	6.70%,8.31%	**5.63%**,6.70%
RCGAN	**4.95%**,**4,42%**	3.62%,3.88%
GMMN	6.70%,6.70%	6.43%,6.30%
GARCH	9.70%,8.75%	7.63%,6.70%

Table 5.3: Results of coverage ratio test applied to SPX and DJI stock returns. In each cell, the left/right numbers are the results for the SPX/ DJI data respectively. A ratio that is closer to 5% indicates a more accurate 95% VaR estimate. The VaR estimate of each model for each observation is obtained using 5000 generated samples.

5.7 Conclusions

In this chapter, we introduced MCGAN, a more general framework that enhances the generator's performance. MCGAN incorporates mean squared error (MSE) and the Monte-Carlo method into the generative loss function, providing strong supervision for the generator. As shown in this chapter, in order to learn the target distribution, we no longer require an optimal discriminator of a certain discriminative loss. Instead, we expect the discriminator to obtain discriminability, a weaker condition on the discriminator, such that when combined with MSE, the discriminability will provide sufficient information to guide the generator training. Additionally, this innovative generative loss function has better training stability compared with non-saturating loss. A toy example named Dirac-GAN is employed to illustrate the advantage of MCGAN. In this example, while other unregularized GANs fail to converge to the equilibrium point, the generator of MCGAN still successfully reaches the optimum regardless of the failure of the discriminator, which is attributed to the strong supervision provided by MSE. For the conditional time-series task, MC-GAN can also avoid the additional least square regression used in SigCWGAN and conditional WGAN to estimate the conditional expectation under the real measure, which considerably improves the efficiency of the algorithm.

Numerical experiments on synthetic and empirical datasets presented in Section 5.6 also validate the superiority of MCGAN compared with other baselines. On VAR datasets, MCGAN consistently improves the performance of RCGAN in terms of all test metrics. Moreover, MCGAN has better training stability and faster convergence than RCGAN. Compared with other baseline models, it also better captures the conditional distribution of the VAR dataset. On the ARCH dataset, MCGAN also achieved the best abs metrics across all k values. On the stock dataset, it is observed that MCGAN considerably outperforms RCGAN in terms of abs metric and cross-correlation metric. Additionally, the MCGAN consistently demonstrates significantly lower R^2 relative error across different lag values, indicating better short-term and long-term predictive power of MCGAN. All these state-of-the-art results show the effectiveness and advantages of MCGAN in generating realistic time series.

Chapter 6

Conclusion and future direction

6.1 Conclusion

To sum up, we proposed two methods in this thesis to improve the generative performance of GANs for time-series generation. Specifically, from the discriminator perspective, we propose SigCWGAN which is inspired by the Wasserstein distance in signature feature space and provides an efficient and robust algorithm for conditional time-series generation. SigCWGAN turns the conventional GAN framework into supervised learning, which dramatically simplifies the training and improves generative performance. From the generator perspective, we propose a more general conditional generative model, namely the MCGAN algorithm, where we design a novel generative loss function that utilizes MSE and Monte Carlo method to compute the difference between the real and fake expected discriminator scores. Benefiting from the strong supervision in MSE, this proposed generative loss function enables the generator to learn the target distribution with a relatively weaker condition imposed on the discriminator. Additionally, it is shown to have better training stability compared to the original GAN.

Numerical results shown in Sections 4.4 and 5.6 demonstrate that both SigCWGAN and MCGAN can achieve state-of-the-art performance on both synthetic and empirical datasets. These results allow us to conclude that the proposed SigCWGAN and MCGAN are empirically useful in learning the conditional distributions induced by time series, especially in capturing the temporal dependency. Additionally, both of them substantially improve the training stability in the time-series generation task compared with other baseline models.

6.2 Discussion and future direction

While SigCWGAN and MCGAN have shown impressive performance, it is important to acknowledge their limitation and potential problems. In this section, we discuss some potential problems of SigCWGAN and MCGAN and explore future directions.

6.2.1 About SigCWGAN

In spite of the superior performance of SigCWGAN, it still has some potential problems. As already discussed in Section 5.1, the regression module we proposed in Section 4.2.1 to estimate the conditionally expected signature has the risk of unsatisfactory estimation of the expected signature under the true measure. And it might suffer from overfitting problems in the case of a very high degree of the signature. Moreover, the residual assumption of linear regression models might not be fulfilled in practice. As claimed in [26], the covariance of the residual under the linear signature models has a special algebraic structure: the covariance is uniquely determined by the expected signature via shuffle product. Therefore, the homoscedasticity assumption is violated, which can result in a biased estimator of the expected signature.

Despite the motivation of Sig-W$_1$ from the approximation of W_1^{Sig}, it is hard to establish the theoretical results on the link between these two metrics. The main difficulty comes from that the uniform approximation of the continuous function f by a linear map L on $\mathcal{K}$ does not guarantee the closeness of their Lipschitz norms. We conjecture that in general $W_1^{Sig}(\mu,\nu)$ is not equal to Sig-W$_1(\mu,\nu)$. However, it would be interesting but technically challenging to find out sufficient conditions such that these two metrics coincide [20].

Apart from that, we are also interested in applying SigCWGAN in high-dimensional time-series generation [74, 75, 76, 77]. However, SigCWGAN suffers from the curse of dimensionality as the dimension $\left(\sum_{i=0}^{k} d^i = \frac{d^{k+1}-1}{d-1}\right)$ of the truncated signature grows geometrically in the dimension of the path d. For this reason, the current SigCWGAN algorithm might not be able to perform high-dimensional time-series generation tasks efficiently. To combat this challenge, one potential solution is to apply dimension reduction techniques in path augmentation like coordinates projection and random projections [28, 78]. The random projection consists of multiple random affine transformations that map the high-dimension time series into a smaller space, while coordinates projection can be used to compute the signature of multiple subsets of the coordinates individually. Although one can easily incorporate these dimension reduction techniques into the SigCWGAN framework, the efficiency of the algorithm remains an open question.

6.2.2 About MCGAN

As for the MCGAN model, it provides a more general framework than SigCWGAN and can be easily extended to other conditional generation tasks like conditional image generation [79, 43, 80]. However, one potential problem of MCGAN is the over-fitting of the generator. Especially when training sets are very limited, the generator tends to generate only a handful of samples that can achieve large discriminator output. Consequently, generated samples are of extremely low diversity no matter what noise you feed into the generator. In that case, we suggest applying weight decay [59] that is used to limit the weight norm and hence improve the network's generalization. The effectiveness of weight decay in preventing the generator from over-fitting has already been empirically demonstrated in [50]. By applying weight decay on the generator network, we will be able to improve the diversity of the generated samples under the usage of MSE.

Additionally, it will be intriguing to explore how the MC size affects the generation performance of MCGAN. Although a larger MC size will give us a better estimator of conditional expectation, it might not lead to a better generalization of the model. For example, if we fix the real samples and consider a sufficiently large MC sample size, then in every training epoch we will have a very stable and accurate MC estimator of the conditional expectation of fake discriminator output. In this case, the job of the generator is rather easy, it only needs to output a constant sample such that the discriminator output given this sample is equal to the average of the discriminator output of those fixed real samples, which leads to a terrible generalization of the generator. This phenomenon is also connected to large-batch training where there is a degradation in the quality of the model. Keskar et al. [81] empirically investigate this problem. It is found that the small-batch (SB) method usually achieves better test accuracy compared to the large-batch (LB) method although the SB training accuracy is a bit worse. This lack of generalization is due to the fact that the LB method tends to converge to sharp minimizers of the objective function that is highly sensitive to the training set. In contrast, the SB method will consistently converge to a flat minimizer which is less sensitive to the training set and leads to a better generalization of the model. To overcome the degradation in generalization when using the LB method, a prospective remedy includes dynamic sampling where the batch size is increased gradually as the iteration progresses [82, 83]. Similarly, in order to improve the generalization of the generator while using a large MC sample size, we can also apply this warm-start method where we start with a relatively smaller MC sample size at the beginning, then a larger size at the later stage of training. It will also be interesting to theoretically prove how the MC size affects the generalization of the generator.

Appendix A

Rough Paths

For the sake of precision, we introduce in this chapter the definition of p-rough path and geometric p-rough path. Again, we follow the notations from [16].

A.1 p-rough paths

Let $p \geq 1$ be a real number and $X : J \to E$ be a continuous path. Recall that the p-variation of X on the interval J is defined by

$$||X||_{p,J} = \left[\sup_{\mathcal{D} \subset J} \sum_{j=0}^{r-1} \left| X_{t_{j+1}} - X_{t_j} \right|^p \right]^{\frac{1}{p}}, \tag{A.1}$$

where the supremum is taken over any time partition of J, i.e. $\mathcal{D} = (t_1, t_2, \cdots, t_r)$. [1]

Let $\mathcal{V}^p(J,E)$ denote the range of any continuous path mapping from J to E of finite p-variation. The larger the p-variation is, the rougher a path is. For each $p \geq 1$, the p-variation norm of a path $X \in \mathcal{V}^p(J,E)$ is denoted by $||X||_{p-var}$ and defined as follows:

$$||X||_{p-var} = ||X||_{p,J} + \sup_{t \in J} ||X_t||.$$

Before we go through the definition of p-rough paths, we first introduce the *Control* function.

[1] Let $J = [s,t]$ be a closed bounded interval. A time partition of J is an increasing sequence of real numbers $\mathcal{D} = (t_0, t_1, \cdots, t_r)$ such that $s = t_0 < t_1 < \cdots < t_r = t$. Let $|\mathcal{D}|$ denote the number of time points in $\mathcal{D}$, i.e. $|\mathcal{D}| = r + 1$. $\Delta\mathcal{D}$ denotes the time mesh of $\mathcal{D}$, i.e. $\Delta\mathcal{D} := \max_{i=0}^{r-1}(t_{i+1} - t_i)$.

Definition A.1.1 (Control)**.** Let Δ_T denote the simplex $\{(s,t) \in [0,T]^2 : 0 \leq s \leq t \leq T\}$. A *control function* on $J = [0,T]$ is a continuous non-negative function ω on Δ_T which is super-additive in the sense that

$$\omega(s,t) + \omega(t,u) \leq \omega(s,u), \quad \forall s \leq t \leq u \in J$$

and for which $\omega(t,t) = 0$ for all $t \in J$.

Now we focus on collections of elements of $T((E))$ which satisfy Chen's identity. Such collections are called *multiplicative functionals* and the point of the theory of rough paths is to take them as the fundamental objects driving differential equations.

Definition A.1.2 (Multiplicative functional)**.** Let $n \geq 1$ be an integer. Let $X : \Delta_T \to T^{(n)}(V)$ be a continuous map. For each $(s,t) \in \Delta_T$, denote by $X_{s,t}$ the image by X of (s,t) and write

$$X_{s,t} = (X^0_{s,t}, X^1_{s,t}, \dots, X^n_{s,t}) \in R \oplus V \oplus V^{\otimes 2} \oplus \dots \oplus V^{\otimes n}.$$

The function X is called a *multiplicative functional of degree n* in V if $X^0_{s,t} = 1$ for all $(s,t) \in \Delta_T$ and

$$X_{s,u} \otimes X_{u,t} = X_{s,t} \quad \forall s,u,t \in [0,T], s \leq u \leq t.$$

So far, the only condition imposed on a functional X has been Chen's identity, which is a purely algebraic condition. We are now going to introduce a notion of p-variation for multiplicative functionals.

Definition A.1.3. Let $p \geq 1$ be a real number and $n \geq 1$ be an integer. Let $\omega : [0,T] \to [0,\infty)$ be a control. Let $X : \Delta_T \to T^{(n)}(V)$ be a multiplicative functional. We say that X has finite p-variation on Δ_T controlled by ω if

$$||X^i_{s,t}|| \leq \frac{\omega(s,t)^{\frac{i}{p}}}{\beta(\frac{i}{p})!} \quad \forall i = 1 \dots n, \quad \forall (s,t) \in \Delta_T,$$

where

$$\beta = p^2 \left(1 + \sum_{r=3}^{\infty} \left(\frac{2}{r-2} \right)^{\frac{\lfloor p \rfloor + 1}{p}} \right).$$

In general, we say that X has finite p-variation if there exists a control ω such that the conditions above are satisfied.

A p-rough path is then defined to be a multiplicative functional of finite p-variation and degree $\lfloor p \rfloor$.

Definition A.1.4 (p-rough path)**.** Let V be a Banach space. Let $p \geq 1$ be a real number. A *p-rough path* in V is a multiplicative functional of degree $\lfloor p \rfloor$ in V with finite p-variation. The space of p-rough paths is denoted by $\Omega_p(V)$.

Given $p \geq q$, a q-rough path can be extended to a multiplicative functional of degree $\lfloor p \rfloor$ with finite q-variation, hence finite p-variation. In particular, a path with bounded variation can be considered canonically as a p-rough path for every $p \geq 1$.

Definition A.1.5 (Geometric p-rough path)**.** A *geometric p-rough path* is a p-rough path that can be expressed as a limit of 1-rough paths in the p-variation distance. The space of geometric p-rough paths in V is denoted by $G\Omega_p(V)$.

For a more detailed description, we refer the interested readers to [16].

A.2 Expected signature of stochastic processes

Definition A.2.1. Let X denote a stochastic process, whose signature is well defined almost surely. Assume that $\mathbb{E}[S(X)]$ is well defined and finite. We say that $\mathbb{E}[S(X)]$ has infinite radius of convergence, if and only if for every $\lambda \geq 0$,

$$\sum_{n \geq 0} \lambda^n |\Pi_n(\mathbb{E}[S(X)])| < \infty.$$

Appendix B

Basics of neural networks

In recent years, neural networks have been very popular in machine learning, which has gained wide attention. It is one way to construct the function with a forecasting ability in supervised learning problems which has been successfully applied to many real world problems. The simplest neural network (NN) consists of two different layers: one input layer where the data flows in and one output layer where the prediction $\hat{y} \in \mathbb{R}^e$ is made, which can be formulated as:

$$\hat{y} = \sigma(\mathbf{W}x + b), \quad \forall x \in \mathbb{R}^d,$$

where $\mathbf{W} \in \mathbb{R}^{e\times d}$ is the weight matrix, $b \in \mathbb{R}^d$ is the bias vector and σ is called the activation function which can add non-linearity to the neural network. Let the set of parameters be denoted as θ where $\theta = \{\mathbf{W}, b\}$. The process of training a neural network is indeed the following optimization problem:

$$\arg\min_{\theta} \mathcal{L}(\theta | \{x_i, y_i\}_{i=1}^N),$$

where the function $\mathcal{L}$ is called the *loss* function which measures the difference between the ground-truth y_i and the prediction $\hat{y}_i$ given x_i.

To make the model more complex, one can increase the number of layers. The NN with L layers is a nonlinear map $h_L : \mathbb{R}^d \to \mathbb{R}^d$ which receives the input x and maps it to the output $\hat{y}$ such that $\hat{y} := h_L(x)$. It usually consists of three different layers: the input layer, the hidden layer, and the output layer. Here we present two examples.

Feedforward Neural Network Conventionally, the feedforward neural network (FNN) or deep neural network (DNN) is a NN with $L > 2$. Let $l \in \{1, \cdots, L-1\}$ be the index for hidden layers and n_l be the number of neurons, i.e. the number of elements in the output vector of the l^{th} layer. Layers in FNN are defined in the following recursive way:

1. Input layer: $h_0 : \mathbb{R}^d \to \mathbb{R}^d$,

   $$h_0(x) = x, \quad \forall x \in \mathbb{R}^d;$$

2. Hidden layer: $h_l : \mathbb{R}^{n_{l-1}} \to \mathbb{R}^{n_l}$

   $$h_l(x) = \sigma_l(\mathbf{W}_l h_{l-1}(x) + b_l), \quad \forall x \in \mathbb{R}^d$$

   where $\mathbf{W}_l \in \mathbb{R}^{n_l \times n_{l-1}}$ is the weight matrix, $b_l \in \mathbb{R}^{n_l}$ is the bias term and σ_l is called the activation function.

3. Output layer: $h_L : \mathbb{R}^{n_{L-1}} \to \mathbb{R}^e$

   $$h_L(x) = \sigma_L(\mathbf{W}_L h_{L-1}(x) = b_L), \quad \forall x \in \mathbb{R}$$

   where $\mathbf{W}_L \in R^{e \times n_{L-1}}$ is the weight matrix, $b_L \in \mathbb{R}^e$ is the bias term and σ_L is the activation function.

Thus the output of FNN is $\hat{y} = h_L(x)$ and the parameter set is $\theta := \{(\mathbf{W}_l, b_l)\}_{l=1}^{L}$. The feedforward neural network is the simplest type of artificial neural network devised. In FNN, the information moves in only one direction—forward, as its name implies, from the input layer, through the hidden layer, and to the output layer.

Recurrent Neural Networks As a descendant of FNN, recurrent neural networks (RNN) have a totally different structure, where the data in RNN would flow backward from the output of the hidden layer to the layer itself. Hence RNN is useful when dealing with time-dependent data, such as solutions of stochastic differential equations, human actions, hand-written characters, etc. The RNN is also composed of three types of layers, i.e. the input layer, the hidden layer, and the output layer. RNN takes the sequential input data $\{x_t\}_{t=1}^T$, where $x_t \in \mathbb{R}^d$, and compute the output $\{o_t\}_{t=1}^T$, where $o_t \in \mathbb{R}^e$ using Equation (B.1):

$$\begin{aligned} h_t &= \sigma(\mathbf{U}x_t + \mathbf{W}h_{t-1}), \\ o_t &= q(\mathbf{V}h_t), \end{aligned} \tag{B.1}$$

where $\{h_t\}_{t=1}^T$ is the hidden layer output with $h_t \in \mathbb{R}^h$ and $\mathbf{U} \in \mathbb{R}^{h\times d}$, $\mathbf{W} \in \mathbb{R}^{h\times h}$ and $\mathbf{V} \in \mathbb{R}^{e\times h}$ are model parameters, and σ and q are two activation functions in the hidden layer and output layer respectively. We denote the RNN model as $R((x_t)_t)$. When training the RNN, the prediction is $\hat{y}_t := R((x_t)_t)$ and $\theta := \{\mathbf{U}, \mathbf{W}, \mathbf{V}\}$ is the parameter set.

The model parameters of the neural network are updated during the training to minimize the loss, and the choice of the loss function depends on the goal we want to achieve. For generative models like GAN [35], the loss function is binary cross entropy which is a measure of the difference between two probability measures for a given random variable or set of events. The model parameters are updated to learn the target distribution in such a way that the model can be used to generate new samples that persist with the same property as the original dataset does.

B.1 AR-FNN architecture

In this section, we give a detailed description of the AR-FNN architecture below. For this purpose, let us start with defining the employed transformations, namely the parametric rectifier linear unit (PReLU) and the residual layer.

Definition B.1.1 (Parametric rectifier linear unit)**.** The parametrised function $\phi_\alpha \in C(\mathbb{R}, \mathbb{R}), \alpha \geq 0$ defined as

$$\phi_\alpha(x) = \max(0, x) + \alpha \min(0, x)$$

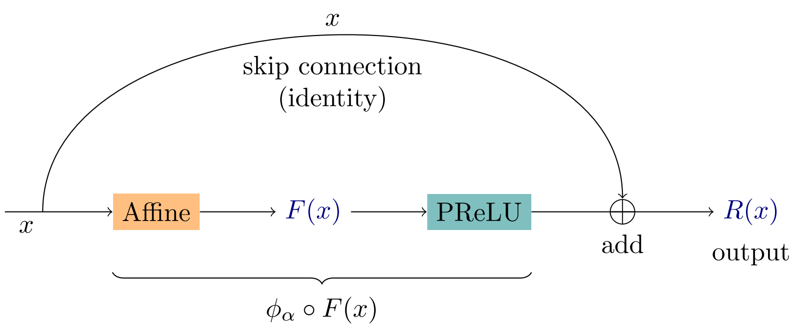


Figure B.1: Architecture of residual layer.

is called *parametric rectifier linear unit* (*PReLU*).

Definition B.1.2 (Residual layer)**.** Let $F : \mathbb{R}^n \to \mathbb{R}^n$ be an affine transformation and $\phi_\alpha, \alpha \geq 0$ a PReLU. The function $R : \mathbb{R}^n \to \mathbb{R}^n$ defined as

$$R(x) = x + \phi_\alpha \circ F(x),$$

where ϕ_α is applied component-wise, is called *residual layer*.

We provide an illustration of residual layer architecture in Figure B.1.

The AR-FNN is defined as a composition of PReLUs, residual layers, and affine transformations. Its inputs are the past p-lags of the d-dimensional process we want to generate as well as the d-dimensional noise vector. A formal definition is given below.

Definition B.1.3 (AR-FNN)**.** Let $d, p \in \mathbb{N}$, $A_1 : \mathbb{R}^{d(p+1)} \to \mathbb{R}^{n_1}$, $A_4 : \mathbb{R}^{n_3} \to \mathbb{R}^d$ be affine transformations, $\phi_\alpha, \alpha \geq 0$ a PReLU and $R_i : \mathbb{R}^{n_{i-1}} \to \mathbb{R}^{n_i}$ for $i = 2, 3$ two residual layers. Then the function $\mathrm{ArFNN} : \mathbb{R}^{dp} \times \mathbb{R}^d \to \mathbb{R}^d$ defined as

$$\mathrm{ArFNN}(x, z) = A_4 \circ R_3 \circ R_2 \circ \phi_\alpha \circ A_1(xz),$$

where xz denotes the concatenated vectors x and z, is called *autoregressive feedforward neural network* (*AR-FNN*).

In our implementation, the hidden dimension is set as $n_1 = n_2 = n_3 = 50$. The pseudo-code of generating the next q-step forecast using G^{θ} can be found in Algorithm 1.

Appendix C

Supplementary numerical results

C.1 Implementation and hyper-parameters

The implementation of those numerical experiments included in this thesis can be found in the GitHub repository [1]. For all the numerical experiments, we use the following hyper-parameters: length of past path $p = 3$, length of future path $q = 3$, batch size $B = 200$, generator learning rate $\alpha_\theta = 1 \times 10^{-4}$, discriminator learning rate $\alpha_\phi = 2 \times 10^{-4}$, number of discriminator optimization steps per generator iteration $N_d = 3$. Regarding the Monte Carlo sample size, MCGAN and SigCWGAN share the same MC sample size for each dataset, which we list in Table D.1.

Dataset	VAR(d=1)	VAR(d=2)	VAR(d=3)	ARCH	SPX	SPX/DJI
MC size	500	1000	1000	2000	1000	1000

Table D.1: Monte Carlo sample size used in SigCWGAN and MCGAN for each example.

C.1.1 Truncation level of signature in SigCWGAN

We also provide Table D.2 that summarizes the truncation level of the signature used in SigCWGAN for each example. The signature of future paths and past paths share the same truncation level.

Dataset	VAR(d=1)	VAR(d=2)	VAR(d=3)	ARCH	SPX	SPX/DJI
Level	3	2	2	3	3	2

Table D.2: Truncation level of signature used in SigCWGAN for each example.

[1] `https://github.com/Baoren1996/SigWGANandBeyond`

C.2 VAR(1) dataset

We conduct the extensive experiments on VAR(1) with different hyper-parameter settings, i.e. $d \in \{1,2,3\}$, $\sigma, \phi \in \{0.2,0.5,0.8\}$.

Test metrics of different models We apply MCGAN, SigCWGAN, and the other above-mentioned methods on the VAR(1) dataset with various hyperparameter settings. The summary of the test metrics of all models on d dimensional VAR(1) data for $d = 1,2,3$ can be found in Table D.3, D.4 and D.5 respectively.

Table D.3: Numerical results of VAR(1) for $d = 1$

	Temporal Correlations		
Settings	$\phi = 0.2$	$\phi = 0.5$	$\phi = 0.8$
Metric on marginal distribution			
SigCWGAN	0.00522	0.00610	**0.00381**
MCGAN	**0.00402**	**0.00501**	0.00384
TimeGAN	0.0259	0.02735	0.01691
RCGAN	0.00443	0.00683	0.00464
GMMN	0.00678	0.00659	0.00554
Absolute difference of lag-1 autocorrelation			
SigCWGAN	0.00947	0.01464	**0.00182**
MCGAN	0.00648	0.02047	0.00324
TimeGAN	0.04269	0.04526	0.01651
RCGAN	**0.00266**	0.01943	0.00531
GMMN	0.01232	**0.00106**	0.00618
Relative R^2 error (%)			
SigCWGAN	0.45011	**0.12953**	**0.00654**
MCGAN	**0.15403**	0.39642	0.03417
TimeGAN	7.44523	2.12036	1.38983
RCGAN	2.16534	0.93133	0.19214
GMMN	0.34882	1.36565	2.10632
Sig-W_1 distance			
SigCWGAN	0.69598	**1.09869**	**2.34807**
MCGAN	**0.69529**	1.10365	2.35118
TimeGAN	0.71696	1.12885	2.37692
RCGAN	0.69653	1.0995	2.35203
GMMN	0.70083	1.10592	2.3526

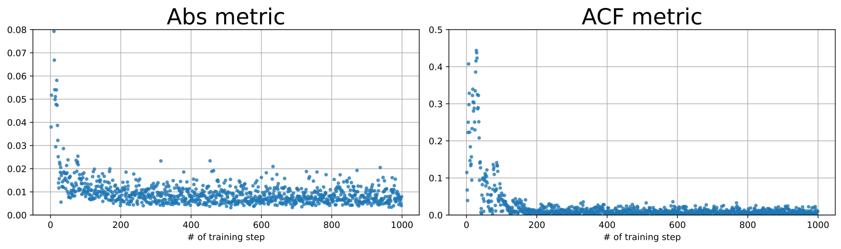
Table D.4: Numerical results of VAR(1) for $d = 2$

	Temporal Correlations (fixing $\sigma = 0.8$)			Feature Correlations (fixing $\phi = 0.8$)		
Settings	$\phi = 0.2$	$\phi = 0.5$	$\phi = 0.8$	$\sigma = 0.2$	$\sigma = 0.5$	$\sigma = 0.8$
Metric on marginal distribution						
SigCWGAN	0.01177	**0.00537**	**0.00365**	**0.00383**	**0.00277**	**0.00365**
MCGAN	**0.00384**	0.00651	0.00538	0.00457	0.00502	0.00538
TimeGAN	0.02059	0.02187	0.01113	0.00933	0.01099	0.01113
RCGAN	0.00613	0.00706	0.00466	0.00607	0.00886	0.00466
GMMN	0.00861	0.00912	0.00601	0.00474	0.00476	0.00601
Absolute difference of lag-1 autocorrelation						
SigCWGAN	**0.00658**	**0.00248**	**0.00419**	**0.00353**	0.00555	**0.00419**
MCGAN	0.02137	0.04051	0.00716	0.00438	0.00840	0.00716
TimeGAN	0.04433	0.04567	0.00822	0.02446	**0.00442**	0.00822
RCGAN	0.01857	0.04249	0.03218	0.01227	0.03571	0.03218
GMMN	0.00699	0.02081	0.04263	0.08085	0.05893	0.04263
L_1-norm of real and generated cross-correlation matrices						
SigCWGAN	0.00804	0.01113	**0.01122**	**0.00476**	0.01198	**0.01122**
MCGAN	0.02653	0.01502	0.01149	0.01381	0.03842	0.01149
TimeGAN	0.08622	0.07002	0.07494	0.07455	0.04685	0.07494
RCGAN	0.01200	0.02846	0.03460	0.08187	0.03317	0.03460
GMMN	**0.00745**	**0.00565**	0.02705	0.00973	**0.00917**	0.02705
Relative R^2 error (%).						
SigCWGAN	**1.24036**	**0.09027**	**0.01252**	**0.01381**	**0.01248**	**0.01252**
MCGAN	10.24923	1.61850	0.42136	0.26449	0.35319	0.42136
TimeGAN	40.1273	4.92783	1.21018	1.05100	0.89636	1.21018
RCGAN	18.33682	4.31191	1.39435	3.94201	1.58417	1.39435
GMMN	35.25094	15.76457	6.56956	12.42385	9.88914	6.56956
Sig-W_1 distance						
SigCWGAN	**1.92823**	2.42590	**3.60068**	**3.02208**	3.23497	**3.60068**
MCGAN	1.93087	**2.42466**	3.61617	3.02879	3.2390	3.61617
TimeGAN	1.98070	2.47622	3.63571	3.04472	3.26746	3.63571
RCGAN	1.93333	2.43379	3.61464	3.03564	**3.21083**	3.61464
GMMN	1.94517	2.43949	3.60922	3.02910	3.23898	3.60922

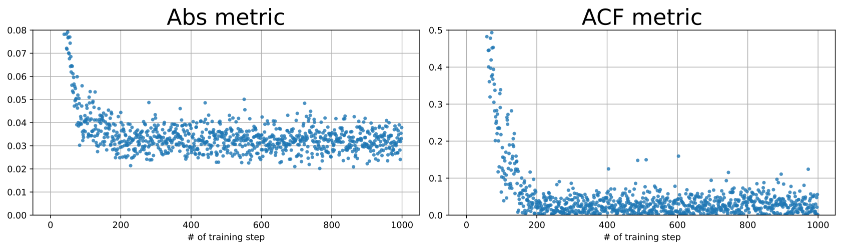
Training stability Figures D.1 to D.3 demonstrate the stability of the SigCWGAN and MCGAN optimization in terms of training iterations in contrast to other baselines, in particular two baselines involving the min-max game optimization.

Table D.5: Numerical results of VAR(1) for $d = 3$

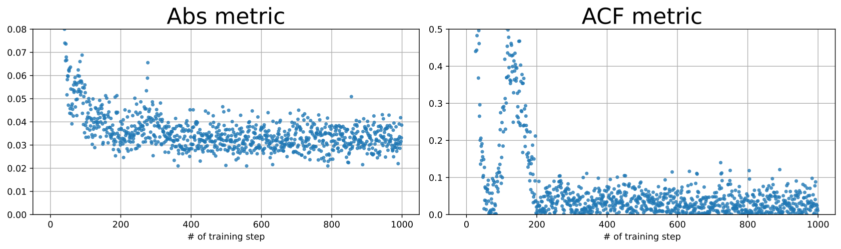
	Temporal Correlations (fixing $\sigma = 0.8$)			Feature Correlations (fixing $\phi = 0.8$)		
Settings	$\phi = 0.2$	$\phi = 0.5$	$\phi = 0.8$	$\sigma = 0.2$	$\sigma = 0.5$	$\sigma = 0.8$
Metric on marginal distribution						
SigCWGAN	0.01463	0.01240	**0.00477**	**0.00423**	**0.00452**	**0.00477**
MCGAN	**0.00476**	**0.00436**	0.00596	0.00715	0.00661	0.00596
TimeGAN	0.02359	0.02096	0.00886	0.01054	0.00915	0.00886
RCGAN	0.01068	0.00634	0.00577	0.00836	0.00597	0.00577
GMMN	0.01001	0.01024	0.00987	0.01427	0.01323	0.00987
Absolute difference of lag-1 autocorrelation						
SigCWGAN	**0.00570**	**0.00508**	**0.00131**	**0.00330**	**0.00172**	**0.00131**
MCGAN	0.00684	0.01805	0.0199	0.00947	0.00529	0.0199
TimeGAN	0.04601	0.09309	0.01643	0.03144	0.04736	0.01643
RCGAN	0.05663	0.04925	0.02041	0.01894	0.01863	0.02041
GMMN	0.04041	0.06024	0.08998	0.10196	0.13395	0.08998
L_1-norm of real and generated cross-correlation matrices						
SigCWGAN	**0.01214**	**0.01311**	**0.00317**	**0.01715**	**0.02862**	**0.00317**
MCGAN	0.04076	0.03819	0.03659	0.04631	0.08001	0.03659
TimeGAN	0.20056	0.43239	0.15509	0.09314	0.09228	0.15509
RCGAN	0.24082	0.16809	0.09657	0.16257	0.11514	0.09657
GMMN	0.09850	0.12638	0.20142	0.3096	0.37507	0.20142
Relative R^2 error (%).						
SigCWGAN	**1.393190**	**0.34009**	**0.07690**	**0.05323**	**0.03498**	**0.07690**
MCGAN	14.63272	1.62564	0.56412	0.32549	0.42388	0.56412
TimeGAN	36.71498	8.94899	2.38110	2.61944	3.80723	2.38110
RCGAN	70.69909	16.50512	2.83140	1.44543	2.79532	2.83140
GMMN	152.87792	38.97992	17.94085	25.12542	26.93346	17.94085
Sig-W_1 distance						
SigCWGAN	11.57390	**17.66105**	30.46722	30.75008	25.24824	30.46722
MCGAN	11.57797	17.69298	30.48571	**30.73360**	**25.22319**	30.48571
TimeGAN	11.88320	18.09083	30.70047	30.86857	25.36035	30.70047
RCGAN	**11.53368**	17.72101	**30.40070**	30.74105	25.30295	**30.40070**
GMMN	11.61313	17.73444	30.59544	30.79028	25.38754	30.59544



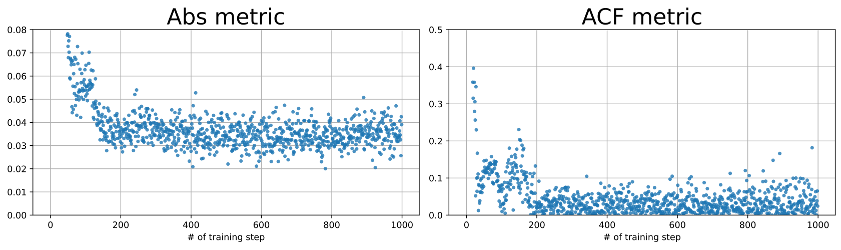
(a) SigCWGAN



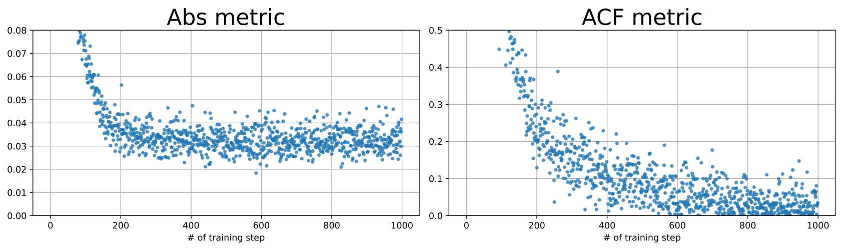
(b) MCGAN



(c) RCGAN

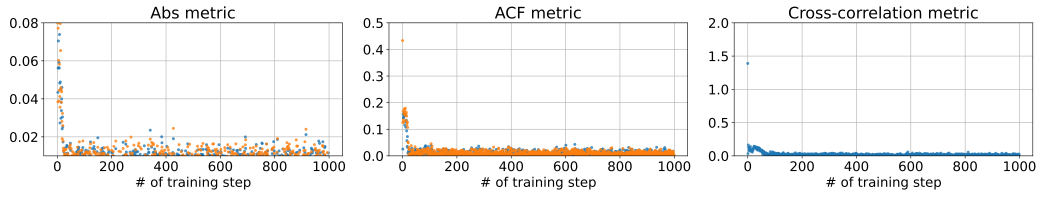


(d) TimeGAN

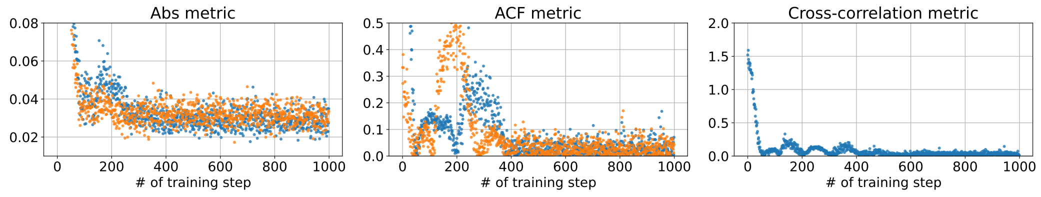


(e) GMMN

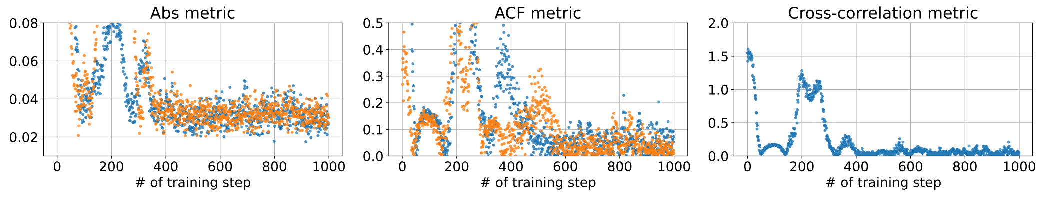
Figure D.1: Example development of the considered distances and score functions during training for the 1-dimensional VAR(1) model with autocorrelation coefficient $\phi = 0.8$ and co-variance parameter $\sigma = 0.8$. The colors blue and orange indicate the relevant distance/score for each dimension.



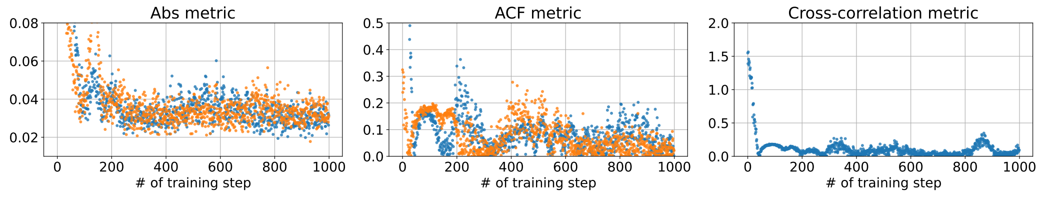
(a) SigCWGAN



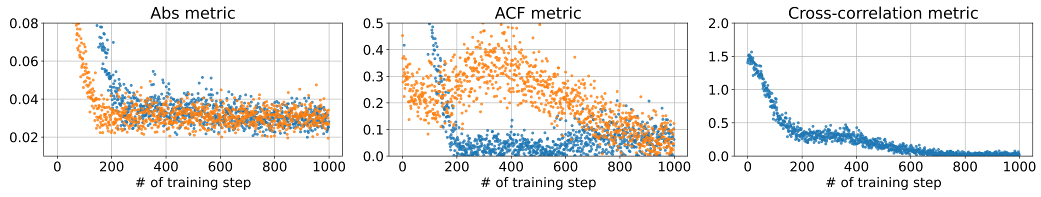
(b) MCGAN



(c) RCGAN

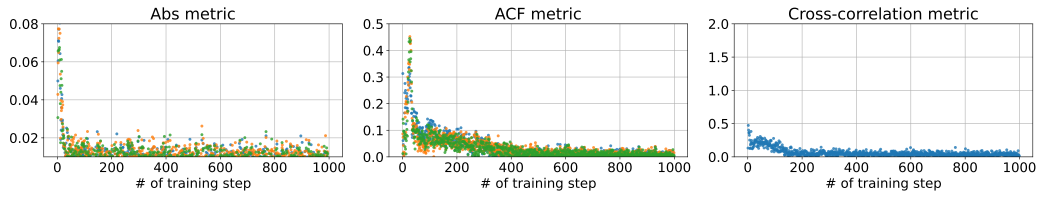


(d) TimeGAN

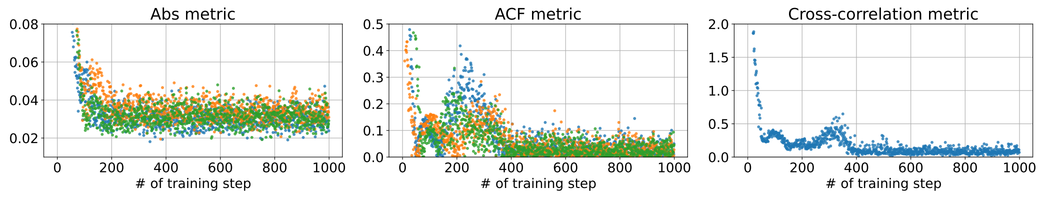


(e) GMMN

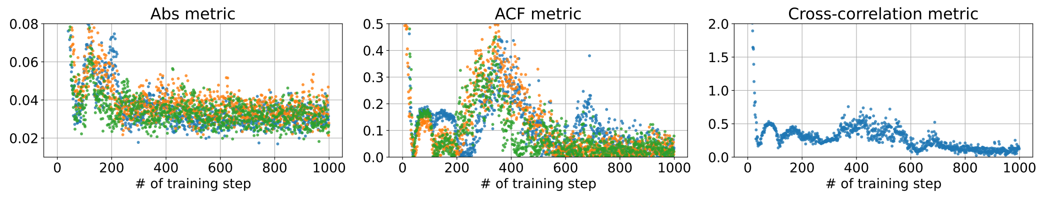
Figure D.2: Example development of the considered distances and score functions during training for the 2-dimensional VAR(1) model with autocorrelation coefficient $\phi = 0.8$ and co-variance parameter $\sigma = 0.8$. The colors blue and orange indicate the relevant distance/score for each dimension.



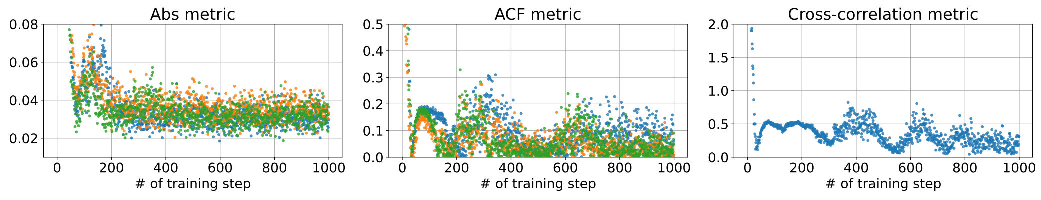
(a) SigCWGAN



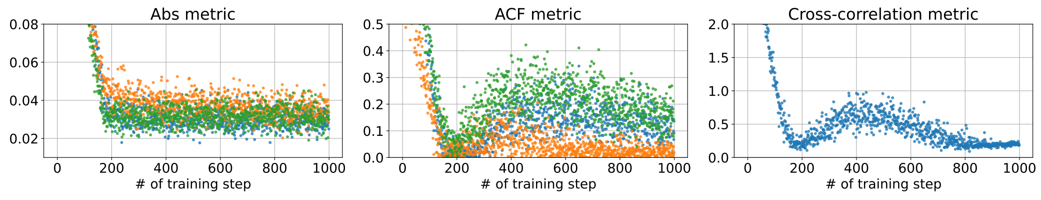
(b) MCGAN



(c) RCGAN



(d) TimeGAN



(e) GMMN

Figure D.3: Example development of the considered distances and score functions during training for the 3-dimensional VAR(1) model with autocorrelation coefficient $\phi = 0.8$ and co-variance parameter $\sigma = 0.8$. The colors blue and orange indicate the relevant distance/score for each dimension.

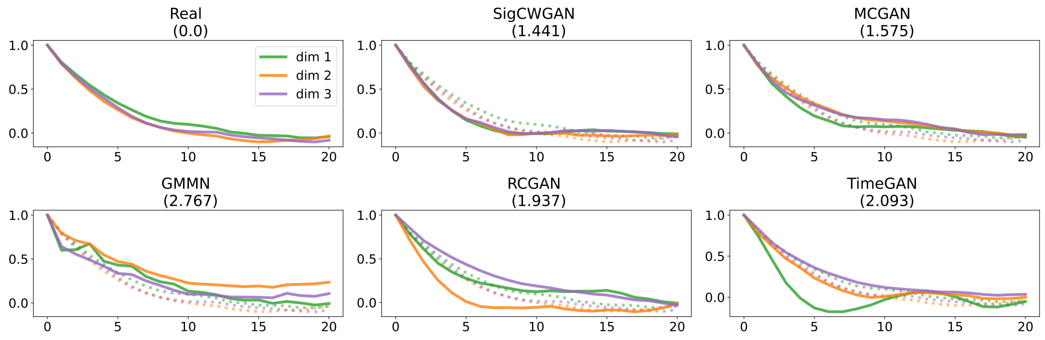


Figure D.4: ACF plot for each channel on the 3-dimensional VAR(1) dataset with autocorrelation coefficient $\phi = 0.8$ and co-variance parameter $\sigma = 0.8$. Here x-axis represents the lag value (with a maximum lag equal to 100) and the y-axis represents the corresponding auto-correlation. The length of the real/generated time series used to compute the ACF is 1000. The number in the bracket under each model is the sum of the absolute difference between the correlation coefficients computed from real (dashed line) and generated (solid line) samples.

C.3 ARCH(k)

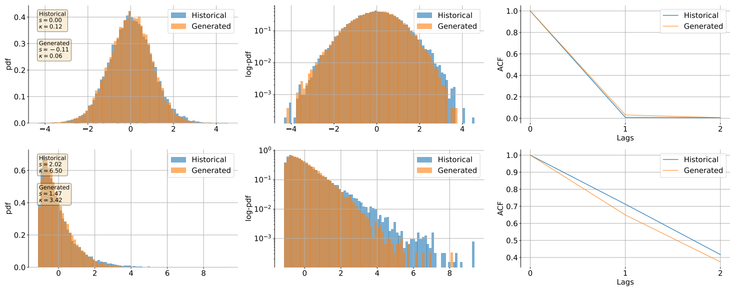
We implement extensive experiments on ARCH(k) with different $k-$lag values, i.e. $k \in \{2,3,4\}$. We choose the optimal degree of signature as 3. The numerical results are summarized in Table D.6. The best results among all the models are highlighted in bold.

Table D.6: Numerical results of the ARCH(k) datasets.

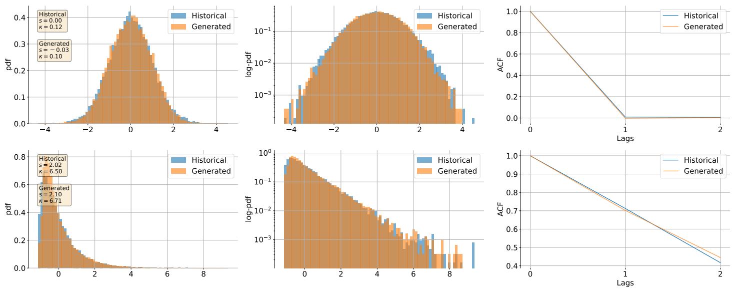
Settings	$k = 2$	$k = 3$	$k = 4$
Metric on marginal distribution			
SigCWGAN	0.01956	0.00884	0.01342
MCGAN	**0.00883**	**0.00489**	**0.00638**
TimeGAN	0.01712	0.01501	0.01239
RCGAN	0.01249	0.01113	0.00814
GMMN	0.02079	0.01498	0.00980
Absolute difference of lag-1 autocorrelation			
SigCWGAN	**0.01178**	**0.00601**	0.00687
MCGAN	0.05063	0.05070	**0.00647**
TimeGAN	0.06774	0.04640	0.00788
RCGAN	0.03317	0.02608	0.02679
GMMN	0.02131	0.01270	0.00943
l_1-norm of real and generated cross correlation matrices			
SigCWGAN	0.01002	0.00991	**0.02645**
MCGAN	0.04777	0.03374	0.05829
TimeGAN	**0.00243**	0.01321	0.12278
RCGAN	0.00674	**0.00775**	0.02941
GMMN	0.04594	0.05310	0.03074
Relative error (%) of R^2 obtained from TSTR.			
SigCWGAN	**0.42229**	**0.22901**	**0.51255**
MCGAN	1.66906	2.41386	3.53569
TimeGAN	7.99938	2.28691	11.3179
RCGAN	6.30669	7.65128	5.68634
GMMN	77.6923	26.9197	20.6545
Sig-W_1 distance			
SigCWGAN	**2.78657**	**4.25746**	**7.10808**
MCGAN	2.80282	4.27829	7.11126
TimeGAN	2.83394	4.29551	7.12979
RCGAN	2.80822	4.27600	7.11534
GMMN	2.85250	4.30627	7.14542

Lag / Model	1	2	3	4	5	6	7	8
SigCWGAN	**0.22901**	**0.34059**	**0.61954**	**0.74292**	**0.74742**	**0.79215**	**1.57154**	**1.03204**
MCGAN	2.41386	4.06995	6.30312	6.72356	8.65506	10.22342	12.17741	12.58088
TimeGAN	2.28691	6.19728	15.37614	30.2916	46.70249	62.8842	79.76519	91.16927
RCGAN	7.65128	6.85297	9.81539	11.52681	13.94434	15.80158	18.10313	15.93167
GMMN	26.9197	28.68219	39.458	53.78034	53.56911	57.89666	58.25836	58.76633

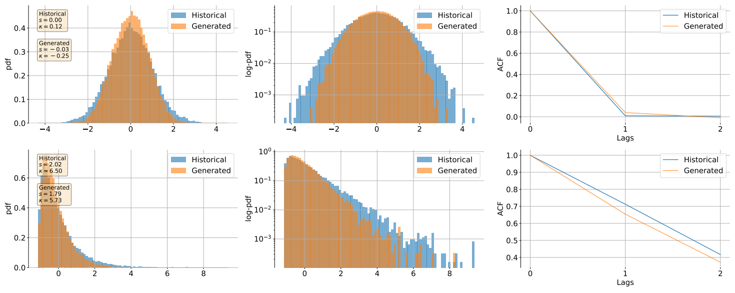
Table D.7: Relative R^2 error (%) of the ARCH(3) model for different lag values.



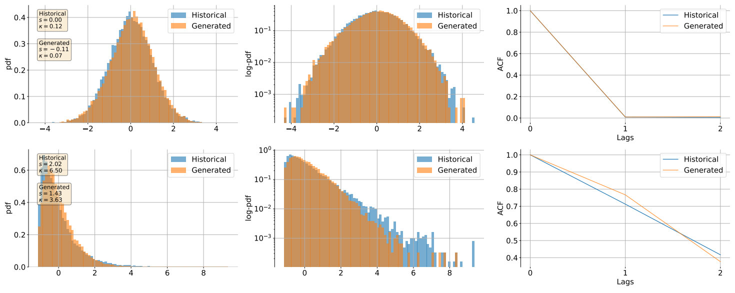
(a) MCGAN



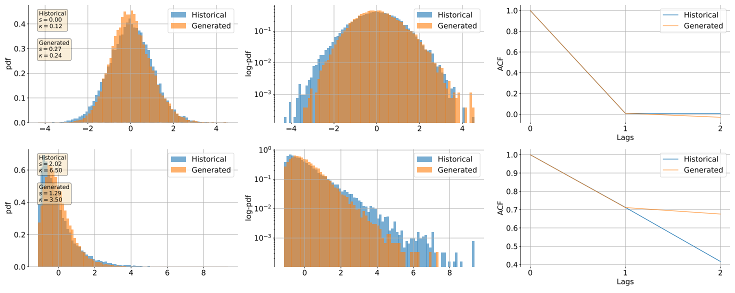
(b) SigCWGAN



(c) TimeGAN



(d) RCGAN



(e) GMMN

Figure D.5: The plot displays a comparison of the marginal distribution on the linear- and log-scale, as well as the fit of the auto-correlation for ARCH(3) data. Histograms and the auto-correlation of the data are indicated in blue and the fits from the generator are colored in orange.

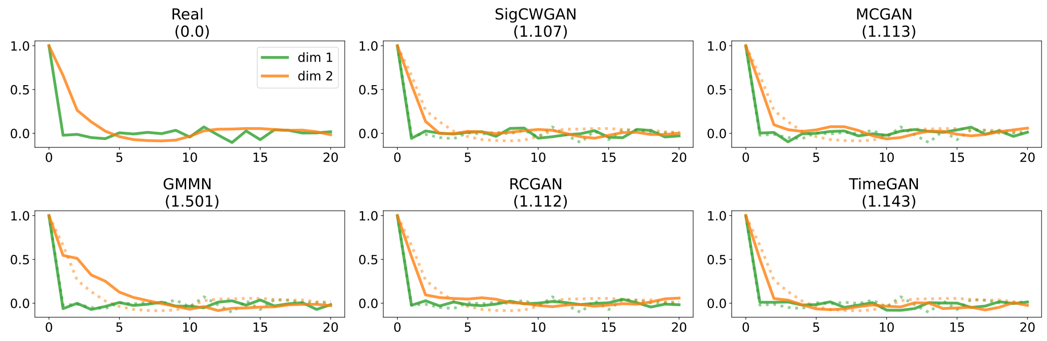


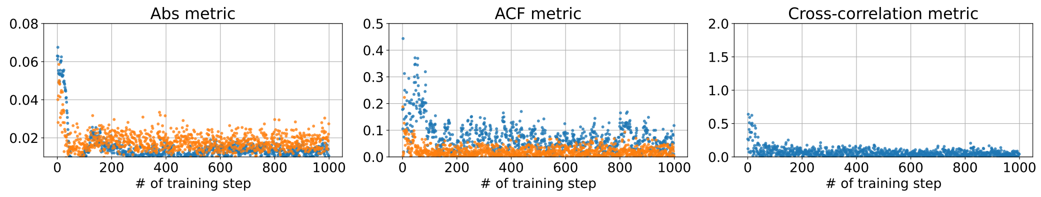
Figure D.6: ACF plot for each channel on the ARCH(2) dataset. Here *x*-axis represents the lag value (with a maximum lag equal to 100) and the *y*-axis represents the corresponding auto-correlation. The length of the real/generated time series used to compute the ACF is 1000. The number in the bracket under each model is the sum of the absolute difference between the correlation coefficients computed from real (dashed line) and generated (solid line) samples.

C.4 SPX and DJI dataset

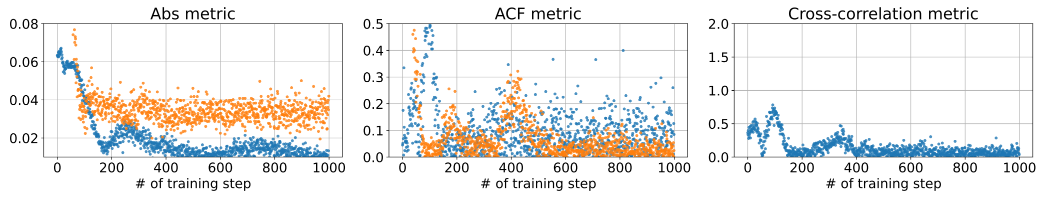
Here we provide the supplementary results on the SPX and DJI datasets. The summary of test metrics of different models is given by Table D.8. The test metrics over the training process of each method on (1) the SPX dataset and (2) the SPX and DJI dataset can be found in Figure D.7 and Figure D.8. The fitting of different models in terms of the cross-correlation matrix of the log-return and log-realized volatility of SPX/ SPX and DJI are presented in Figure D.9.

Table D.8: Numerical results of the stocks datasets.

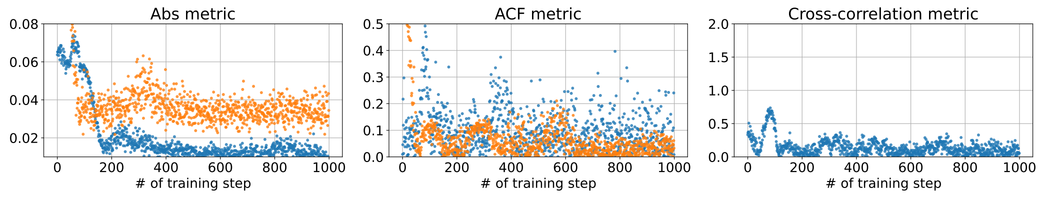
Data type	SPX	SPX + DJI
Metric on marginal distribution		
SigCWGAN	0.01468	**0.00879**
MCGAN	**0.00915**	0.00937
TimeGAN	0.01064	0.01136
RCGAN	0.01134	0.00933
GMMN	0.00986	0.01438
GARCH	0.01583	0.01670
Absolute difference of lag-3 autocorrelation		
SigCWGAN	**0.02693**	0.03988
MCGAN	0.03665	0.03386
TimeGAN	0.0341	**0.03147**
RCGAN	0.04192	0.03367
GMMN	0.05483	0.06668
GARCH	0.05392	0.05337
L_1-norm of real and generated cross-correlation matrices		
SigCWGAN	**0.00638**	**0.10354**
MCGAN	0.05641	0.12938
TimeGAN	0.0101	0.11342
RCGAN	0.0926	0.13699
GMMN	0.07498	0.28339
GARCH	0.15791	0.72901
Relative R^2 error (%).		
SigCWGAN	5.56765	5.28866
MCGAN	3.87395	**4.12438**
TimeGAN	8.5833	7.13335
RCGAN	**3.74567**	5.75112
GMMN	5.33373	12.17658
GARCH	12.1253	12.5686
Sig-W_1 distance		
SigCWGAN	4.74507	5.18462
MCGAN	**4.73513**	5.19871
TimeGAN	4.75276	5.22316
RCGAN	4.74362	5.20083
GMMN	4.75251	**5.17637**
GARCH	4.75825	5.25344



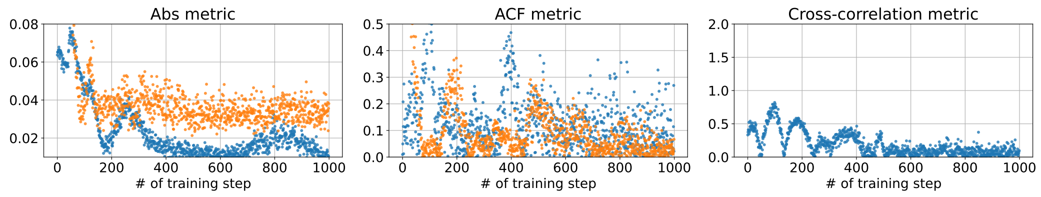
(a) SigCWGAN



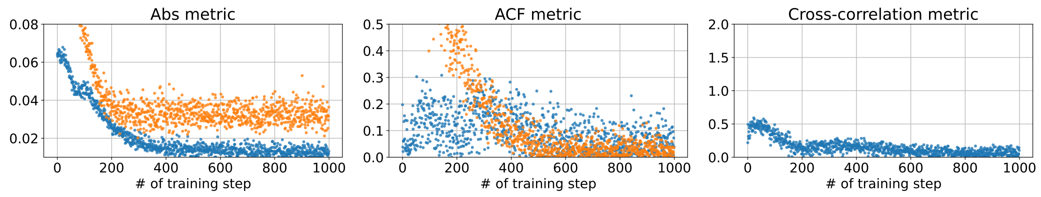
(b) MCGAN



(c) TimeGAN

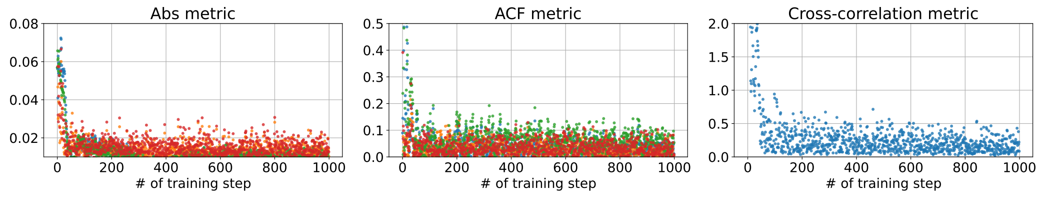


(d) RCGAN

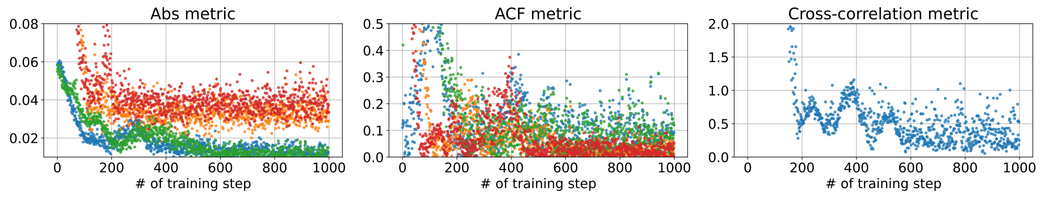


(e) GMMN

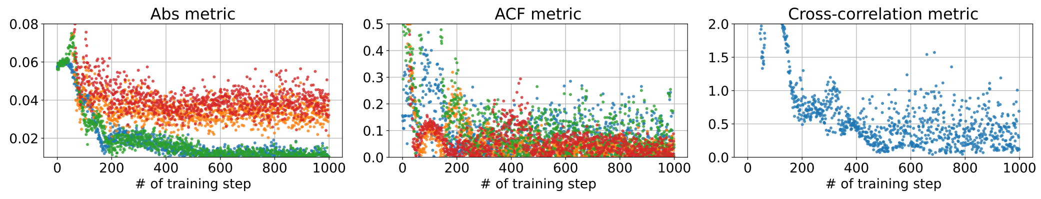
Figure D.7: Example development of the considered distances and score functions during training for SPX data.



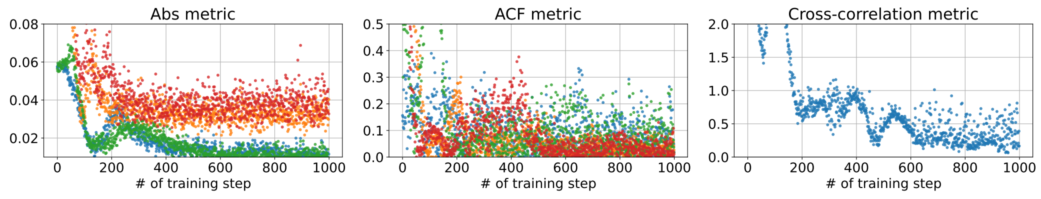
(a) SigCWGAN



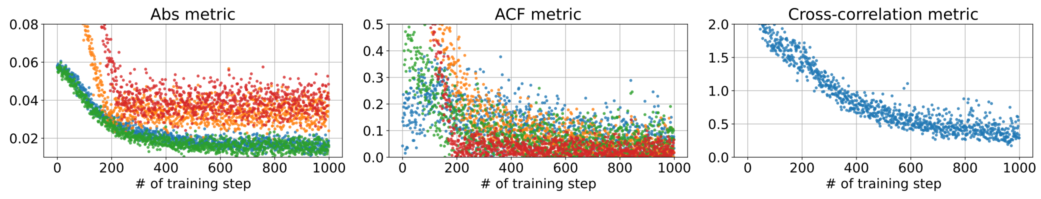
(b) MCGAN



(c) TimeGAN



(d) RCGAN



(e) GMMN

Figure D.8: Example development of the considered distances and score functions during training for SPX and DJI data.

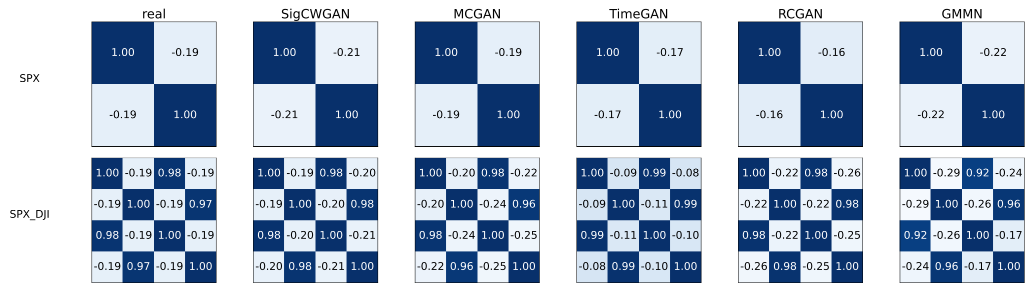


Figure D.9: Comparison of real and synthetic cross-correlation matrices for SPX/ SPX and DJI data. On the far left the real cross-correlation matrix from SPX/ SPX and DJI log-return and log-volatility data is shown. x/y-axis represents the feature dimension while the color of the $(i, j)^{th}$ block represents the correlation of $X_t^{(i)}$ and $X_t^{(j)}$. Observe that the historical correlation between log returns and log volatility is negative, indicating the presence of leverage effects, i.e. when log returns are negative, log volatility is high.

Model \ Lag	1	2	3	4	5	6	7	8
SigCWGAN	6.25606	**3.22038**	5.2733	4.42471	3.82947	3.8639	4.57777	5.56765
MCGAN	4.20975	3.39397	3.38214	**3.39932**	**3.28247**	**3.07611**	**3.32665**	3.87395
TimeGAN	**3.79664**	3.8673	3.83001	4.11029	4.67948	5.1156	6.09387	8.5833
RCGAN	5.06443	4.20889	**3.33699**	3.68433	4.0931	3.65047	3.58021	**3.74567**
GMMN	11.37772	8.38272	8.0536	6.11822	5.17814	4.99595	5.1035	5.33373

Table D.9: Relative R^2 error (%) of the SPX data for different lag values.

Model \ Lag	1	2	3	4	5	6	7	8
SigCWGAN	3.82897	4.36748	3.67481	4.54616	4.25157	4.73639	4.00567	5.28866
MCGAN	2.93112	**2.39898**	3.62457	**3.15994**	**2.82198**	**3.12895**	**2.99333**	**4.12438**
TimeGAN	**2.90929**	3.0085	**3.51063**	4.21676	4.81641	5.08649	6.08654	7.13335
RCGAN	5.4847	4.64733	6.34131	5.64255	4.84939	4.58869	4.3158	5.75112
GMMN	13.73239	12.04712	10.90208	10.96817	9.80816	10.5603	10.38836	12.17658

Table D.10: Relative R^2 error (%) of the SPX and DJI data for different lag values.

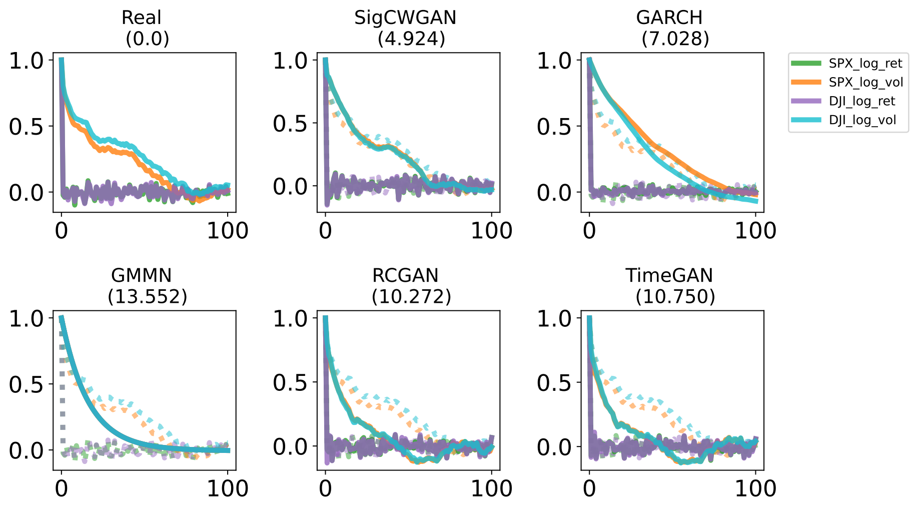
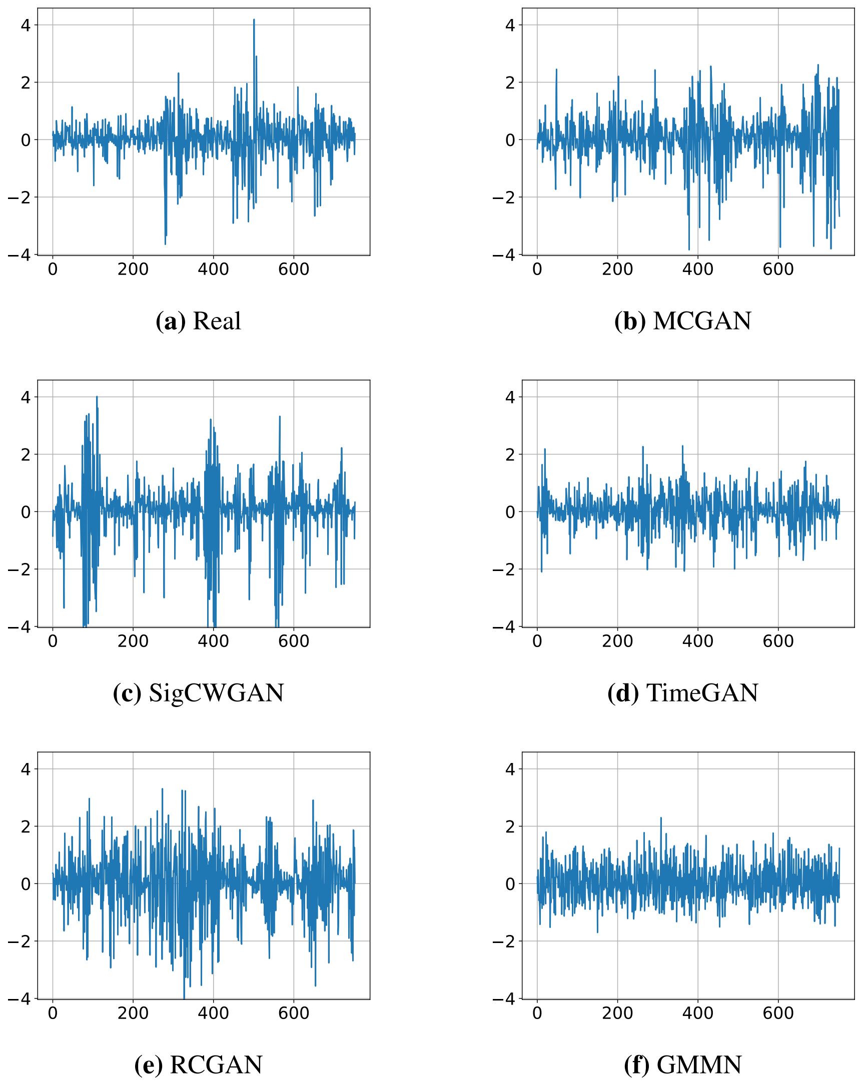


Figure D.10: ACF plot for each channel on the SPX/DJI dataset. Here *x*-axis represents the lag value (with a maximum lag equal to 100) and *y*-axis represents the corresponding auto-correlation. The length of the real/generated time series used to compute the ACF is 1000. The number in the bracket under each model is the sum of the absolute difference between the correlation coefficients computed from real (dashed line) and generated (solid line) samples.

Lag Model	1	2	3	4	5
SigCWGAN	**0.02048**,0.01567	0.04697,**0.01118**	**0.02693**,0.03988	0.06516,0.04697	0.08845,0.04504
MCGAN	0.02392,0.02257	**0.01844**,0.03866	0.03665,0.03386	0.0593,0.04871	0.07881,0.05291
TimeGAN	0.02676,**0.00927**	0.02752,0.01752	0.0341,**0.03147**	**0.04489,0.03729**	**0.04396,0.04433**
RCGAN	0.03085,0.01663	0.03568,0.01747	0.04192,0.03367	0.06414,0.04773	0.06858,0.05919
GMMN	0.02301,0.01929	0.02236,0.00836	0.05483,0.06668	0.07812,0.12326	0.08671,0.12287

Table D.11: Autocorrelation metric for the stock datasets for different lag values. In each cell, the left/right numbers are the result for the SPX data/ the SPX and DJI data respectively.



(a) Real

(b) MCGAN

(c) SigCWGAN

(d) TimeGAN

(e) RCGAN

(f) GMMN

Figure D.11: Example paths of SPX log returns generated by each model. Since the path of DJI log returns is similar to that of SPX, there is no need to make another plot for DJI.

Bibliography

[1] Samuel A Assefa, Danial Dervovic, Mahmoud Mahfouz, Robert E Tillman, Prashant Reddy, and Manuela Veloso. 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.

[2] Steven M Bellovin, Preetam K Dutta, and Nathan Reitinger. Privacy and synthetic datasets. *Stan. Tech. L. Rev.*, 22:1, 2019.

[3] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[4] Magnus Wiese, Lianjun Bai, Ben Wood, and Hans Buehler. Deep hedging: learning to simulate equity option markets. *Available at SSRN 3470756*, 2019.

[5] Magnus Wiese, Robert Knobloch, Ralf Korn, and Peter Kretschmer. Quant gans: deep generation of financial time series. *Quantitative Finance*, 20(9):1419–1440, 4 2020.

[6] Christa Cuchiero, Wahid Khosrawi, and Josef Teichmann. A generative adversarial network approach to calibration of local stochastic volatility models. *Risks*, 8(4):101, 2020.

[7] Hans Buehler, Blanka Horvath, Terry Lyons, Imanol Perez Arribas, and Ben Wood. A data-driven market simulator for small data environments. *arXiv preprint arXiv:2006.14498*, 2020.

[8] Adriano Koshiyama, Nick Firoozye, and Philip Treleaven. Generative adversarial networks for financial trading strategies fine-tuning and combination. *Quantitative Finance*, 0(0):1–17, 2020.

[9] Patryk Gierjatowicz, Marc Sabate-Vidales, David Siska, Lukasz Szpruch, and Zan Zuric. Robust pricing and hedging via neural sdes. *Available at SSRN 3646241*, 2020.

[10] Farzan Farnia and Asuman Ozdaglar. Gans may have no nash equilibria. *arXiv preprint arXiv:2002.09124*, 2020.

[11] Eric V Mazumdar, Michael I Jordan, and S Shankar Sastry. On finding local nash equilibria (and only local nash equilibria) in zero-sum games. *arXiv preprint arXiv:1901.00838*, 2019.

[12] Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, and Olivier Bousquet. Are gans created equal? a large-scale study. *arXiv preprint arXiv:1711.10337*, 2017.

[13] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? In *International conference on machine learning*, pages 3481–3490. PMLR, 2018.

[14] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

[15] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[16] Terry J Lyons, Michael Caruana, and Thierry Lévy. *Differential equations driven by rough paths*. Springer, 2007.

[17] Weixin Yang, Terry Lyons, Hao Ni, Cordelia Schmid, Lianwen Jin, and Jiawei Chang. Leveraging the path signature for skeleton-based human action recognition. *arXiv preprint arXiv:1707.03993*, 2017.

[18] Shujian Liao, Terry Lyons, Weixin Yang, and Hao Ni. Learning stochastic differential equations using rnn with log signature features. *arXiv preprint arXiv:1908.08286*, 2019.

[19] Hao Ni, Lukasz Szpruch, Marc Sabate-Vidales, Baoren Xiao, Magnus Wiese, and Shujian Liao. Sig-wasserstein gans for time series generation. *arXiv preprint arXiv:2111.01207*, 2021.

[20] Hao Ni, Lukasz Szpruch, Magnus Wiese, Shujian Liao, and Baoren Xiao. Conditional sig-wasserstein gans for time series generation. *arXiv preprint arXiv:2006.05421*, 2020.

[21] Laurence C Young. An inequality of the hölder type, connected with stieltjes integration. 1936.

[22] Ben Hambly and Terry Lyons. Uniqueness for the signature of a path of bounded variation and the reduced path group. *Annals of Mathematics*, pages 109–167, 2010.

[23] Horatio Boedihardjo, Xi Geng, Terry Lyons, and Danyu Yang. The signature of a rough path: uniqueness. *Advances in Mathematics*, 293:720–737, 2016.

[24] Horatio Boedihardjo and Xi Geng. The uniqueness of signature problem in the non-markov setting. *Stochastic Processes and their Applications*, 125(12):4674–4701, 2015.

[25] Yves Le Jan and Zhongmin Qian. Stratonovich's signatures of brownian motion determine brownian sample paths. *Probability Theory and Related Fields*, 157(1-2):209–223, 2013.

[26] Daniel Levin, Terry Lyons, and Hao Ni. Learning from the past, predicting the statistics for the future, learning an evolving system. *arXiv preprint arXiv:1309.0260*, 2013.

[27] Ilya Chevyrev and Terry Lyons. Characteristic functions of measures on geometric rough paths. *Annals of Probability*, 44:4049–4082, 2016.

[28] James Morrill, Adeline Fermanian, Patrick Kidger, and Terry Lyons. A generalised signature method for multivariate time series feature extraction. *arXiv preprint arXiv:2006.00873*, 2020.

[29] Ilya Chevyrev and Andrey Kormilitzin. A primer on the signature method in machine learning. *arXiv preprint arXiv:1603.03788*, 2016.

[30] Guy Flint, Ben Hambly, and Terry Lyons. Discretely sampled signals and the rough hoff process. *Stochastic Processes and their Applications*, 126(9):2593–2614, 2016.

[31] Weixin Yang, Terry Lyons, Hao Ni, Cordelia Schmid, and Lianwen Jin. Developing the path signature methodology and its application to landmark-based human action recognition. In *Stochastic Analysis, Filtering, and Stochastic Optimization: A Commemorative Volume to Honor Mark HA Davis's Contributions*, pages 431–464. Springer, 2022.

[32] Terry Lyons and Harald Oberhauser. Sketching the order of events. *arXiv preprint arXiv:1708.09708*, 2017.

[33] Jinsung Yoon, Daniel Jarrett, and Mihaela van der Schaar. Time-series generative adversarial networks. In *Advances in Neural Information Processing Systems*, pages 5509–5519, 2019.

[34] Cristóbal Esteban, Stephanie L Hyland, and Gunnar Rätsch. Real-valued (medical) time series generation with recurrent conditional gans. *arXiv preprint arXiv:1706.02633*, 2017.

[35] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[36] Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. *arXiv preprint arXiv:1701.04862*, 2017.

[37] Shengyu Zhao, Zhijian Liu, Ji Lin, Jun-Yan Zhu, and Song Han. Differentiable augmentation for data-efficient gan training. *Advances in Neural Information Processing Systems*, 33:7559–7570, 2020.

[38] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2794–2802, 2017.

[39] Ilya Kavalerov, Wojciech Czaja, and Rama Chellappa. cgans with multi-hinge loss. *arXiv preprint arXiv:1912.04216*, 2019.

[40] Aude Genevay, Marco Cuturi, Gabriel Peyré, and Francis Bach. Stochastic optimization for large-scale optimal transport. *Advances in neural information processing systems*, 29, 2016.

[41] Gintare Karolina Dziugaite, Daniel M Roy, and Zoubin Ghahramani. Training generative neural networks via maximum mean discrepancy optimization. *arXiv preprint arXiv:1505.03906*, 2015.

[42] Jae Hyun Lim and Jong Chul Ye. Geometric gan. *arXiv preprint arXiv:1705.02894*, 2017.

[43] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

[44] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

[45] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. *Advances in neural information processing systems*, 30, 2017.

[46] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[47] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. In *International conference on machine learning*, pages 2642–2651. PMLR, 2017.

[48] Minguk Kang, Woohyeon Shim, Minsu Cho, and Jaesik Park. Rebooting acgan: Auxiliary classifier gans with stable training. *Advances in Neural Information Processing Systems*, 34:23505–23518, 2021.

[49] Takeru Miyato and Masanori Koyama. cgans with projection discriminator. *arXiv preprint arXiv:1802.05637*, 2018.

[50] Peng Zhou, Lingxi Xie, Bingbing Ni, Cong Geng, and Qi Tian. Omni-gan: On the secrets of cgans and beyond. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14061–14071, 2021.

[51] Chang Lu, Chandan K Reddy, Ping Wang, Dong Nie, and Yue Ning. Multi-label clinical time-series generation via conditional gan. *arXiv preprint arXiv:2204.04797*, 2022.

[52] Giorgia Ramponi, Pavlos Protopapas, Marco Brambilla, and Ryan Janssen. T-cgan: Conditional generative adversarial network for data augmentation in noisy time series with irregular sampling. *arXiv preprint arXiv:1811.08295*, 2018.

[53] Tianyi Lin, Chi Jin, and Michael Jordan. On gradient descent ascent for nonconvex-concave minimax problems. In *International Conference on Machine Learning*, pages 6083–6093. PMLR, 2020.

[54] Constantinos Daskalakis and Ioannis Panageas. The limit points of (optimistic) gradient descent in min-max optimization. *arXiv preprint arXiv:1807.03907*, 2018.

[55] Constantinos Daskalakis, Andrew Ilyas, Vasilis Syrgkanis, and Haoyang Zeng. Training gans with optimism. *arXiv preprint arXiv:1711.00141*, 2017.

[56] Panayotis Mertikopoulos, Christos Papadimitriou, and Georgios Piliouras. Cycles in adversarial regularized learning. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2703–2717. SIAM, 2018.

[57] Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. The numerics of gans. *Advances in neural information processing systems*, 30, 2017.

[58] Casper Kaae Sønderby, Jose Caballero, Lucas Theis, Wenzhe Shi, and Ferenc Huszár. Amortised map inference for image super-resolution. *arXiv preprint arXiv:1610.04490*, 2016.

[59] Anders Krogh and John Hertz. A simple weight decay can improve generalization. *Advances in neural information processing systems*, 4, 1991.

[60] Riccardo Passeggeri. On the signature and cubature of the fractional brownian motion for h¿ 12. *Stochastic Processes and their Applications*, 130(3):1226–1257, 2020.

[61] Thomas Fawcett. *Problems in stochastic analysis: connections between rough paths and non-commutative harmonic analysis*. PhD thesis, University of Oxford, 2002.

[62] Siran Li and Hao Ni. Expected signature of stopped brownian motion on d-dimensional c2, α-domains has finite radius of convergence everywhere: $2 \leq d \leq 8$. *Journal of Functional Analysis*, 282(12):109447, 2022.

[63] Adeline Fermanian. Functional linear regression with truncated signatures. *Journal of Multivariate Analysis*, page 105031, 2022.

[64] Robert F Engle. Autoregressive conditional heteroscedasticity with estimates of the variance of united kingdom inflation. *Econometrica: Journal of the econometric society*, pages 987–1007, 1982.

[65] Shuang Liu, Olivier Bousquet, and Kamalika Chaudhuri. Approximation and convergence properties of generative adversarial learning. *Advances in Neural Information Processing Systems*, 30, 2017.

[66] Junbo Zhao, Michael Mathieu, and Yann LeCun. Energy-based generative adversarial network. *arXiv preprint arXiv:1609.03126*, 2016.

[67] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. *Advances in neural information processing systems*, 29, 2016.

[68] XuanLong Nguyen, Martin J Wainwright, and Michael I Jordan. Estimating divergence functionals and the likelihood ratio by convex risk minimization. *IEEE Transactions on Information Theory*, 56(11):5847–5861, 2010.

[69] Kun Xu, Chongxuan Li, Jun Zhu, and Bo Zhang. Understanding and stabilizing gans' training dynamics using control theory. In *International Conference on Machine Learning*, pages 10566–10575. PMLR, 2020.

[70] William Fedus, Mihaela Rosca, Balaji Lakshminarayanan, Andrew M Dai, Shakir Mohamed, and Ian Goodfellow. Many paths to equilibrium: Gans do not need to decrease a divergence at every step. *arXiv preprint arXiv:1710.08446*, 2017.

[71] Abdul Jabbar, Xi Li, and Bourahla Omar. A survey on generative adversarial networks: Variants, applications, and training. *ACM Computing Surveys (CSUR)*, 54(8):1–49, 2021.

[72] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. *arXiv preprint arXiv:1611.02163*, 2016.

[73] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29, 2016.

[74] Rajat Sen, Hsiang-Fu Yu, and Inderjit S Dhillon. Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting. *Advances in neural information processing systems*, 32, 2019.

[75] Xiajun Jiang, Ryan Missel, Zhiyuan Li, and Linwei Wang. Sequential latent variable models for few-shot high-dimensional time-series forecasting. In *The Eleventh International Conference on Learning Representations*.

[76] Matteo Barigozzi, Haeran Cho, and Dom Owens. Fnets: Factor-adjusted network estimation and forecasting for high-dimensional time series. *arXiv preprint arXiv:2201.06110*, 2022.

[77] Jörg P Bachmann and Johann-Christoph Freytag. High dimensional time series generators. *arXiv preprint arXiv:1804.06352*, 2018.

[78] Terry Lyons and Harald Oberhauser. Sketching the order of events. 8 2017.

[79] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8110–8119, 2020.

[80] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

[81] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

[82] Richard H Byrd, Gillian M Chin, Jorge Nocedal, and Yuchen Wu. Sample size selection in optimization methods for machine learning. *Mathematical programming*, 134(1):127–155, 2012.

[83] Michael P Friedlander and Mark Schmidt. Hybrid deterministic-stochastic methods for data fitting. *SIAM Journal on Scientific Computing*, 34(3):A1380–A1405, 2012.