# Generalized Energy Based Time Series Models

RBTW0[1]

MSc Computational Statistics and Machine Learning

Supervisor: Brooks Paige

Submission date: Day Month Year

[1]**Disclaimer:** This report is submitted as part requirement for the MSc Computational Statistics and Machine Learning degree at UCL. It is substantially the result of my own work except where explicitly indicated in the text. The report will be distributed to the internal and external examiners, but thereafter may not be copied or distrbuted except with permission from the author.

**Abstract**

With the development of more advanced deep learning architectures, it has become critical that sufficient data is available to use models effectively. Often fields that rely on time-series data suffer from serious data-shortage which can limit the use of these models and stagnate research. This has motivated development of models that can generate synthetic data and augment the data-sets to allow for effective analysis. One such approach, that has shown promising performance in image generation, is generalized energy based models. In this thesis we explore the use of these models in time-series generation.

For this we present two generalized energy based models, one based based on attention mechanisms and the other on recurrent neural networks. To assess the performance of the models we used a range of real-life and generated data-sets. While the results for the recurrent based model was inconclusive, we found that by adopting this framework the attention-based model can generate time-series data the better resemble the underlying data, compared to samples from the equivalent GAN. Additionally, we demonstrated that this gain can be directly attributed to incorporating the energy into the sampling process.

# Contents

# Chapter 1

# Introduction

This dissertation aims to explore the application of Generalized Energy Based Models in time-series generation. In this chapter we provide motivation for this avenue of research as well outlining the specific objectives we wish to achieve.

## 1.1 Generative Modelling

If we look at nature under the lens of probability we can think of any observed data as samples from an unknown underlying distribution. The main goal of *generative modelling* is to learn an approximation to this intractable probability distribution. The motivation for this is that if we are able to learn a good approximation then we can use the model in downstream tasks such as sampling, density estimation and representation learning. In this dissertation we are interested in parametric approximations, where models with a finite/fixed set of parameters are used to model the observed underlying data. In this setting, the model is learned by minimizing the *distance* between the model distribution and the underlying data distribution, over the set of parameters of the family of model distributions. Specifically, let $\mathbb{P}_\theta$ and $\mathbb{P}_r$ denote the model and underlying distribution respectively, the optimization problem is represented as follows

$$min_{\theta \in \Theta} d(\mathbb{P}_r, \mathbb{P}_\theta) \tag{1.1}$$

where $\Theta$ is the parametric family of models, $\theta$ is the parameters of the model and $d(.)$ is a distance between two probability distributions. As the underlying distribution is unknown to us we often cannot directly minimize this distance so we instead employ proxy methods

using a set of independent and identically distributed (i.i.d) samples, $\mathcal{D}$, from $\mathbb{P}_r$ that we refer to as the training data.

## 1.2 Deep Generative Models

Often the data of interest in generative modelling is high dimensional and complex, such as images, text or time-series. Choosing the parametric family of models is therefore challenging as it needs to be complex enough to capture the structure of the data while still being computationally tractable. Recent works have advocated the use of deep neural networks due to their inherent ability to approximate functions in high dimensions effectively. In this case the models are called *Deep Generative Models*(DGMs) and the parameters $\theta$ are the weights of the neural network. In this setting the weights are found using *stochastic optimization* methods. In general, DGMs can be divided in two classes; *explicit/prescribed* models whereby the density function $p_\theta(\mathbf{x})$ of the model distribution is explicitly defined and *implicit* models whereby it's not explicitly defined[31]. DGMs have been used to generate a range of data including images, text, music etc. In this dissertation we are focusing on generating *time-series* data.

## 1.3 Time-series data

A time-series is a set of data points indexed in time order. Time-series data is incredibly common in major fields such as medicine, finance, retail, and economics. We denote a multivariate time-series of length $T$ as $\mathbf{x}_{1:T} = \mathbf{x}_1, ..., \mathbf{x}_T$ with $\mathbf{x}_t \in \mathcal{X}$, $\forall 0 \leq t \leq T-1$. In the generative modelling setting the aim is to use a training set $\mathcal{D} = \{\mathbf{x}^n_{1:T}\}_{n=1}^N$ comprising of $N$ time-series to estimate a density function $p_\theta(\mathbf{x}_{1:T})$ that is a good approximation of the underlying density $p_r(\mathbf{x}_{1:T})$. The time order adds an extra challenge to generative modelling as complex correlations may exist across the time-steps as well as across the features. Therefore we are not only tasked with capturing the joint distribution $p_r(\mathbf{x}_{1:T})$, such that the synthetic data is indistinguishable from real data, but also capturing the conditional distribution $\prod_t^T p_r(\mathbf{x}_t|\mathbf{x}_1, ..., \mathbf{x}_{t-1})$ such that the temporal dynamics are attended to and the synthetic data can be used in downstream tasks.

## 1.4 Time Series Generation Problem

To understand the purpose of generating time-series data, we first look at their applications. One of the core applications of synthetic time-series is *data augmentation*. Data augmentation is a broad term for methods used to increase the size of a data-set. This could be through adding modified versions of the existing underlying data or in this case, generating new synthetic data. As well as acting as a regularizer and helping reduce overfitting, data augmentation is advantageous when the data-set is small, imbalanced or access to it is restricted. This is particularly the case with medical and clinical data where the data might be costly, scarce or concerning private information, preventing researchers from openly sharing it. Additionally, with the development and wide adoption of bigger deep learning models in these domains it has become critical that the sufficient amount of data is available.

Another application is *imputation*, where missing or corrupted sectionss of the data is replaced by the synthetic data. In real-life data, errors in the data-sets or incomplete streams is a frequent issue that can cause problems in downstream analytics. A common method with dealing with this problem is to stop using the streams with the missing or corrupted data. While a reliable way of ensuring no corrupted data gets processed it is inefficient and can lead to the data shortage problems we encountered in our discussion of data augmentation. It is therefore crucial that reliable synthetic data is available such that the streams can be utilized to the fullest and the quality of the downstream analytics is maximised.

Synthetic time-series have also found use in detecting anomalies or outliers in real time-series data. *Anomaly detection* is an important area of research with wide application including detecting underlying conditions in medical data, irregularities in financial behaviour or even possible cyber-attacks.

While this list of applications of synthetic time-series is not exhaustive, it demonstrates that access to good quality synthetic time-series data is crucial for the development of a range of fields and in turn, highlights the importance of the further development of the models that generate them.

## 1.5 MSc dissertation objectives

In recent years we have seen the development of powerful DGMs that have allowed us to generate incredibly realistic synthetic data. One such development is Generalized Energy

Based Models[5] which combines ideas from explicit and implicit models. In this work, they demonstrate that this approach can outperform the equivalent networks in image generation tasks. Given the importance of high quality synthetic time-series data, outlined in Section 1.4, it's natural for us to experiment with this framework in the time-series domain. This leads us to specific aim of this dissertation:

*Implement and evaluate a Generalized Energy Based Model for time-series generation*

Within this there are several objectives that we aim to achieve:

1. Review the current state-of-art Deep Generative Models and their application in the time-series domain
2. Introduce Generalised Energy Based Models and provide motivation for their use in this domain
3. Implement a Generalised Energy Based Model for time-series generation
4. Evaluate the performance of the models using a series of data-sets
5. Investigate the effectiveness of the approach across different architectures
6. Propose the direction for future works

## 1.6 Chapter Summary

In this chapter we have introduced and provided motivation for the main aim of this dissertation. We first introduced the generative modelling before looking at the relevance and complexity of time-series generation. We finally presented the specific objectives we wish to achieve in this dissertation. The rest of the dissertation is divided into different chapters that aim to achieve these objectives.

# Chapter 2

# Literature Review and Background

This chapter aims at providing the reader with an overview of current deep generative models and their application in the time-series domain. In doing so we not only motivate the discussion on generalized energy based models but provide some background for the model and experiment design used later in this dissertation. First we look at models that explicitly define the density function, before looking at those that don't and instead rely on sampling from it. At the end of the chapter, we review how these generated samples can be evaluated.

## 2.1 Background

An important part of generative modelling is minimizing the distance between the model distribution and the underlying data distribution. To quantify distances between statistical objects, such as probability distributions, we often use *statistical distances*[1]. This section aims to provide background to these statistical distances.

### 2.1.1 Statistical divergences

An example of a statistical distance is the *statistical divergence*. Specifically, given a differentiable statistical manifold $\mathcal{M}$, a divergence on $\mathcal{M}$ is a function $D : \mathcal{M}\text{x}\mathcal{M} \rightarrow \mathbb{R}$

[1]Statistical distances are similar to traditional distance metrics but do not have to satisfy all the distance axioms. Due to this, they can be seen as a more general case of distance metrics.

satisfying [4]:

$$1. \quad D(P,Q) \geq 0 \quad \forall P,Q \in \mathcal{M} \tag{2.1}$$

$$2. \quad D(P,Q) = 0 \quad \text{if and only if } P = Q \tag{2.2}$$

In the setting of this dissertation, $\mathcal{M}$ is the space of a parametric family of probability distributions, and $P$ and $Q$ are distributions in this space. Intuitively, we understand this as a distance measure that is always positive and is only 0 when the distributions are equal. As opposed to other distance measures, a divergence does not have to be symmetric.

### 2.1.2 $\phi$-divergences

One widely adopted family of statistical divergences are the *$\phi$-divergences* [26, 3]. Given two distributions P and Q over domain $\mathcal{X}$ with density functions $p$ and $q$, the $\phi$ -divergence is defined,

$$D_\phi(P||Q) = \int_{\mathcal{X}} q(x)\phi\left(\frac{p(x)}{q(x)}\right)dx, \tag{2.3}$$

where $\phi : \mathbb{R} \to \mathbb{R}$ is a convex and lower semi-continuous function satisfying $\phi(1) = 0$[3]. Several well-known measures are then obtained by appropriately choosing $\phi$, including the Hellinger distance where $\phi(t) = (\sqrt{t}-1)^2$, total variation distance where $\phi(t) = |t-1|$, $\chi^2$-divergence where $\phi(t) = (t-1)^2$ etc. Arguably the most powerful[54] $\phi$-divergence is the Kullback-Leibler (KL) divergence [73] where $\phi(t) = t\log t$, such that

$$D_{KL}(P||Q) = \int_{\mathcal{X}} q(x)\log\left(\frac{p(x)}{q(x)}\right)dx \tag{2.4}$$

Informally, the KL divergence or *relative entropy* can be viewed as measuring the expected excess surprise from using Q to model P [65]. It is heavily used in machine learning as well as major fields such as fluid mechanics, neuroscience and bioinformatics. This divergence is not symmetric and can be infinite, which may cause issues in some applications. This motivated the Jensen-Shannon (JS) divergence,

$$D_{JS}(P||Q) = \frac{1}{2}D_{KL}(P||M) + \frac{1}{2}D_{KL}(Q||M), \tag{2.5}$$

where $M = \frac{1}{2}(P + Q)$. Informally, we can view the JS divergence as a symmetric and smoothed version of the KL divergence. Given these two additionally properties the JS-divergence now qualifies as a distance metric and thus its square root is often called the Jensen–Shannon distance. Similarly to the KL divergence, the JS divergence has been widely adopted, being applied in fields such as bioinformatics, social sciences and machine learning. Despite their success, $\phi$-divergences have some significant limitations. One such limitation is that they are hard to estimate, especially in high-dimension [126]. An additional limitation can be seen if we consider the instance when P and Q have *dis-joint support*[2]. Here the divergence is non-informative as the ratio of $p$ on $q$ is not well defined over all the domain $\mathcal{X}$.

### 2.1.3 Integral Probability Metrics

Another popular family of statistical distances are the *Integral Probability Metrics*(IPMs). Given $\mathcal{F}$, a set of functions $f : \mathcal{X}$ to $\mathbb{R}$, then

$$d_{\mathcal{F}}(P, Q) = \sup_{f \in \mathcal{F}} \mathbb{E}_{x \sim P}[f(x)] - \mathbb{E}_{x \sim Q}[f(x)] \tag{2.6}$$

is defined as the integral probability metric between $P$ and $Q$ corresponding to function class $\mathcal{F}$[98]. As IPMs do not depend on the ratio of the distributions, they can be used to provide meaningful distances, even when the support is dis-joint. Another advantage they hold over $\phi$-divergences is that, with certain conditions on $\mathcal{F}$ [115], they can be very simple to estimate, even in high dimension. One popular IPM is the Wasserstein-1(W1) or *Earth Mover*(EM) distance,

$$W_1(P, Q) = \inf_{\gamma \in \prod(P,Q)} \mathbb{E}_{(x,y) \sim \gamma}[||x - y||] \tag{2.7}$$

where $\gamma \in \prod(P, Q)$ denotes the set of all joint distributions $\gamma(x, y)$ whose marginals are P and Q [66]. This form of the distance is intractable, but by using the Kantorovich-Rubinstein duality argument, as in [37], it can reformulated into the solution of a maxi-

[2]The support of a function is the set of points where the function of the points is non- zero. In the probabilistic sense, the support of a distribution can then be thought of as the set of random variable values that have a density. If the supports are dis-joint then there are no values that are in both of the sets. Informally, we can think of it as the distributions not overlapping.

mization

$$W_1(P,Q) = \sup_{||f||_L \leq 1} \mathbb{E}_{x\sim P}[f(x)] - \mathbb{E}_{x\sim Q}[f(x)] \tag{2.8}$$

where the $||f||_L \leq 1$ is the set of all 1-Lipschitz functions $f : \mathcal{X} : \mathbb{R}$ A function $f$ is a *K-Lipschitz* function if there exists a real constant $K \geq 0$ such that, for all $x, y \in \mathbb{R}$

$$|f(x) - f(y)| \leq K|x - y| \tag{2.9}$$

Intuitively, this means that the derivative of $f$ is less than or equal to K everywhere. In this form we see that $W_1(P,Q) = d_{\mathcal{F}}(P,Q)$ where $\mathcal{F}$ is the set of 1-Lipschitz functions. In can be shown that, although the relative strength of the topologies of the KL and JS divergence are stronger than that of the W1 distance, the W1 distance is the only informative statistical distance when the distributions are dis-joint or have little overlap[6].

Given this background, we now return to the generative modelling setting where we review the current the literature on DGMs and their application in the time-series domain. First we will consider the models where the density is explicitly defined.

## 2.2 Explicit Density Models

If we recall from the last chapter, learning a deep generative model in the parametric setting consists of minimizing the distance between the model $\mathbb{P}_\theta$ and underlying data distribution $\mathbb{P}_r$. If we consider using the KL divergence then we can then reformulate the optimization problem as follows;

$$\begin{aligned} min_{\theta\in\mathcal{M}} d(\mathbb{P}_r, \mathbb{P}_\theta) &= min_{\theta\in\mathcal{M}} D_{KL}(\mathbb{P}_r, \mathbb{P}_\theta) && (2.10)\\ &= min_{\theta\in\mathcal{M}} \mathbb{E}_{\mathbf{x}\sim p_r}[\log p_r(\mathbf{x}) - \log p_\theta(\mathbf{x})] && (2.11)\\ &= max_{\theta\in\mathcal{M}} \mathbb{E}_{\mathbf{x}\sim p_r}[\log p_\theta(\mathbf{x})] && (2.12)\\ &= max_{\theta\in\mathcal{M}} \mathcal{L}(\theta) && (2.13) \end{aligned}$$

where $\mathcal{L}(\theta)$ is the log-likelihood of the model under parameters $\theta$. In this section we consider models where $p_\theta$ is explicitly defined and thus this maximum likelihood approach can be used as a proxy for our main goal of approximating the underlying data distribution. We first review the models that have densities that are computationally intractable and

rely on approximate methods for learning, before looking at those that are constructed to guarantee tractable densities.

### 2.2.1 Energy-Based Models

The first class of explicitly defined density models we look at are Energy-Based Models (EBMs)[76]. EBMs define the density $p_\theta(x)$ over the whole space $\mathcal{X} \subset \mathbb{R}^d$ via the Gibbs-Boltzmann distribution [75];

$$p_\theta(x) = \frac{\exp\{-E_\theta(x)\}}{Z_\theta}, \qquad Z_\theta = \int \exp\{-E_\theta(x)\}dx, \tag{2.14}$$

where $Z_\theta$, also called the *partition* function, is the normalizing constant and $E_\theta : \mathcal{X} \to \mathbb{R}$ is called the *energy* function. The goal of EBMs is to learn an energy function that admits a model distribution that best approximates the underlying distribution. As we have shown, this can be achieved by maximizing the expected log-likelihood

$$\mathcal{L}(\theta) = \mathbb{E}_{x\sim p_r}[\log p_\theta(x)] \tag{2.15}$$

$$= \mathbb{E}_{x\sim p_r}[-E_\theta(x)] - \log Z_\theta \tag{2.16}$$

As the partition function $Z_\theta$ is often intractable, computing the gradient

$$\nabla_\theta \mathcal{L}(\theta) = -\mathbb{E}_{x\sim p_r}[\nabla_\theta E_\theta(x)] - \nabla_\theta \log Z_\theta \tag{2.17}$$

is challenging due to the $\nabla_\theta \log Z_\theta$ term. A popular approach to circumvent this is to use *Contrastive Divergence*[52] methods. Here samples from the model distribution are used to approximate this learning gradient. Specifically, with some algebraic manipulation[52], it can be shown that $\nabla_\theta \log Z_\theta$ can be expressed as an expectation with respect to $p_\theta$

$$\nabla \log Z_\theta = \mathbb{E}_{x'\sim p_\theta}[-\nabla_\theta E_\theta(x')] \tag{2.18}$$

Which can be numerically approximated using samples from the model distribution $p_\theta(x)$. Substituting this back into (2.17) we find

$$\nabla_\theta \mathcal{L}(\theta) = -\mathbb{E}_{x\sim p_r}[\nabla_\theta E_\theta(x)] + \mathbb{E}_{x'\sim p_\theta}[\nabla_\theta E_\theta(x')] \tag{2.19}$$

Intuitively, this gradient maximises the probability of samples from the underlying distribution, by decreasing their corresponding energy, as well as minimising the probability of samples from the model distribution, by increasing their corresponding energy. Since the value of the partition function is unknown, $p_\theta(x)$ cannot be directly sampled from, so instead Markov Chain Monte Carlo (MCMC)[109] techniques are utilized to transform the training data into data from the model distribution. However, often the data of interest is high dimensional and classical MCMC techniques applied in this space are impractical due to the long mixing times[48].

To overcome this, recent works [35] have proposed the use of a class of MCMC techniques called *Langevin Dynamics*. Here new states are proposed using evaluations of the gradient of the target probability density function combined with interjections of random noise. Specifically, let $p_T$ denote the density function of the target distribution we wish to obtain an i.i.d set of samples from, the *Overdamped Langevin dynamics* are

$$dX_t = \nabla \log p_T(X_t) + \sqrt{2} dW_t, \tag{2.20}$$

where $W_t$ is a standard Brownian motion. Similarly to generic MCMC, as $t \to \infty$, the generated distribution approaches a stationary distribution equivalent to the target distribution[99]. To obtain a sampler, these dynamics are approximated using a discrete time method. One of the simplest and widely adopted is the Euler–Maruyama method[70]. Here the samples are updated through the following iterative process

$$X_0 := x_0, \qquad X_{k+1} := X_k + \lambda \nabla \log p_T(X_k) + \sqrt{2\lambda}\epsilon_k, \tag{2.21}$$

where $\epsilon_k$ is an independent multivariate gaussian, $\lambda > 0$ is a fixed learning rate and $x_0$ is determined by the user. Another example of Langevin dynamics are the *Kinetic Langevin dynamics*, which develops on the former by introducing a momentum variable;

$$dX_t = V_t dt, \qquad dV_t = -\gamma V_t dt + u(\nabla \log p_T(X_t))dt + \sqrt{2\gamma u} dW_t \tag{2.22}$$

where $\gamma$ is a friction variable, $u$ is the inverse mass and $V_t$ is the momentum vector.

Another popular class of sampling methods are *stochastic optimization*[108] methods. Here

the samples are updated as follows

$$X_0 := x_0, \qquad X_{k+1} := X_k + \frac{\lambda_k}{2} \nabla \log p_T(X_k) \tag{2.23}$$

where similar to before $x_0$ is determined by the user. To ensure convergence, $\lambda_k$ has to satisfy the following conditions

$$\sum_{k=1}^{\infty} \lambda_t = \infty \qquad \sum_{k=1}^{\infty} \lambda_t^2 < \infty \tag{2.24}$$

Stochastic Gradient Langevin Dynamics (SGLDs)[128] combines both the overdamped Langevin sampler and the stochastic optimization sampler. SGLDs are a common method for sampling from EBMs and have allowed them to scale to high dimensional data. Specifically the update in the EBM setting is

$$X_0 := x_0, \qquad X_{k+1} := X_k - \frac{\lambda_k}{2} \nabla E_\theta(X_k) + \epsilon_k \tag{2.25}$$

where $x_0$ is generally sampled from a uniform distribution. Despite these advances, training EBMs is often slow due to the repetitive sampling from the model distribution. This motivated approaches that side-step directly maximising the likelihood. One approach is *score matching*[59], which instead aims to minimize the Fisher divergence [63, Definition 1.13] between $\mathbb{P}_r$ and $\mathbb{P}_\theta$

$$\mathcal{L}(\theta) = \frac{1}{2} \mathbb{E}_{p_r(\mathbf{x})} ||\nabla_{\mathbf{x}} \log p_\theta(\mathbf{x}) - \nabla_{\mathbf{x}} \log p_r(\mathbf{x})||_2^2 \tag{2.26}$$

where $\nabla_{\mathbf{x}} \log p_\theta(\mathbf{x})$ is called the *score function* of the data. This divergence has shown promising results in a variety of statistics and machine learning applications [57, 58] and is ideal for this setting as it does not depend on the intractable partition function. However the score function is often not available so various methods are utilized to approximate it including Langevin score matching[64], spectral approximation[14] and denoising score matching[79]. Due to these recent advances, EBMs have seen comparable results to other generative models in a range of tasks including image generation[35]. One advantage EBMs have over other generative models is it's simplicity. Here only one network is trained which results in a balanced training and fewer model parameters. Unbalanced training, which can occur when training several network concurrently, can lead training instabilities and unrealistic generated samples[74]. In the time-series domain EBMs have succesfully been

applied to multivariate time-series forecasting. Specifically, ScoreGrad[131], consisting of a time series feature extraction module and conditional stochastic differential equation based score matching module, achieved state-of-the art results on a range of data-sets.

### 2.2.2 Variational Auto-Encoders

As we have seen one of the limitations of energy-based models is that sampling is often challenging and mixing times can make it impractical. In this section we consider Variational Auto-Encoders(VAEs)[69], a class of explicit generative models that utilize neural networks to explicitly learn a *representation* of the input data. Here the aforementioned problems are circumvented by explicitly sampling from a data distribution in a single pass. Before we look further into the structure of the a VAE we first must introduce the standard auto-encoder. To this end, let us first define the feature-extracting *encoder* function

$$E_\phi : \mathcal{X} \to \mathcal{Z} \tag{2.27}$$

the maps from the input space $\mathcal{X}$ to a latent/feature space $\mathcal{Z}$. Given $x \in \mathcal{X}$ and $E_\phi$ we define $z = E_\phi(x)$ as the *latent variable* or *representation*. Similarly, let us define the *decoder* function

$$D_\theta : \mathcal{Z} \to \mathcal{X} \tag{2.28}$$

that maps from the latent space to the input data space. We can then define the *reconstruction* $x' = D_\theta(z)$ with $x' \in \mathcal{X}$. The central goal of *auto-encoders*[72] is to learn the parameters $\theta$ and $\phi$ such that the reconstruction loss

$$L(\theta, \phi) := \mathbb{E}_{x \sim p_r}[d(x, D_\theta(E_\phi(x)))] \tag{2.29}$$

is minimised. Here $d : \mathcal{X}$ x $\mathcal{X} \to [0, \infty)$ is a *reconstruction error* function such that $d(x, x')$ measures the differences between $x \in \mathcal{X}$ and $x' \in \mathcal{X}$. To capture the meaningful structure of the data, it is important to prevent the auto-encoder from simply learning the identity function and duplicating the data. One strategy that achieves this is constraining the latent dimension to be smaller than that of the data dimension. With this constraint the autoencoder is said to be *undercomplete*.

Another motivation for this can be seen if we consider the *manifold hypothesis*. Here

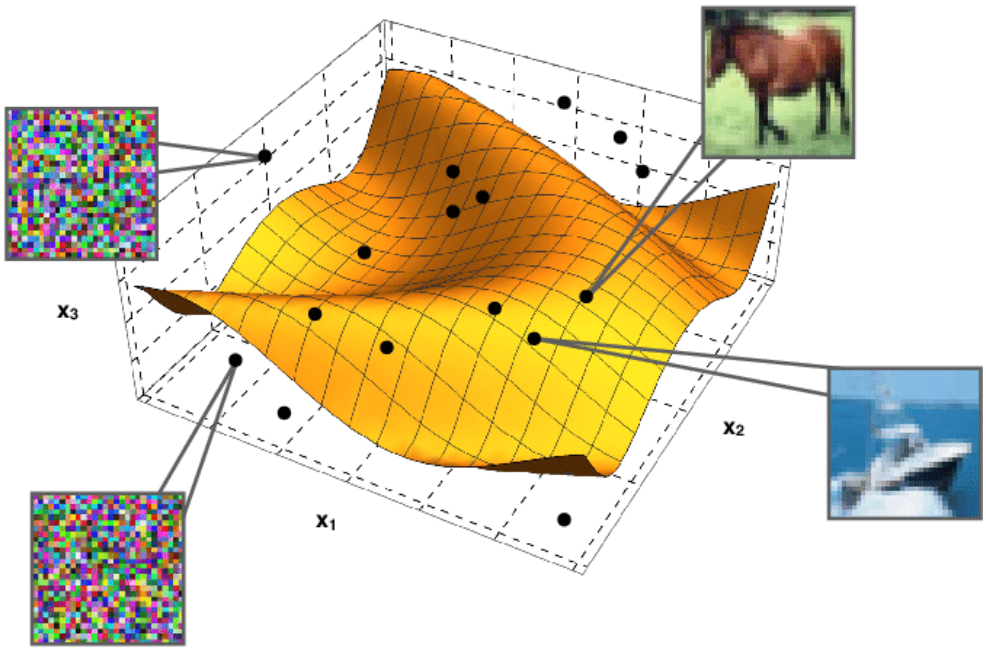



Figure 2.1: Illustrative example of the low dimensional manifold in the input space using CIFAR10 samples. In this high dimensional space most points will not resemble a plausible image. The points we interpret as images tend to reside on a hidden lower dimensional manifold, here shown as the yellow surface. Credit: [44]

real-world data presented in high dimensional spaces are expected to concentrate in a much lower dimensional manifold[18]. Figure 2.1 illustrates the notion of this in the image domain. Similarly, in the time-series domain the temporal dynamics of the data is often driven by a small amount of low-dimensional factors of variation[134]. This particularly apparent when there are strong dependencies in data [118]. By learning the low-dimensional support of the data it is believed that one can better model the structure of the underlying distribution that generated it. An implication of this hypothesis is that, if different data-sets concentrate in different low-dimensional manifolds then their distributions are more likely to have dis-joint support.

Despite being powerful reconstruction tools, the traditional auto-encoder defined above is not considered a generative model as the latent space has no distributional interpretation. VAEs overcome this by regularising the latent space to ensure that it has the sufficient properties for data generation. Specifically, the encoder of the VAE is now trained to output the parameters of a distribution over the latent space. The decoder then maps latent vector samples from this distribution to the data space. The density of a VAE can

then be expressed

$$p_\theta(x) = \int_z p_\theta(x|z)p_\theta(z)dz \tag{2.30}$$

where $p_\theta(z)$ is the prior latent distribution and $p_\theta(x|z)$ is the latent based model [69]. However, this integral is often intractable so instead a lower bound on the log-likelihood is optimized. Formally, let $q_\phi(z|x) = argmin_q D_{KL}(q_\phi(z|x)||p_\theta(z|x))$ be a tractable approximation of the true posterior distribution, then

$$\ln p_\theta(x) \geq -D_{KL}(q_\phi(z|x)||p_\theta(z)) + \mathbb{E}_{q_\phi(z|x)}[\ln p_\theta(x|z)] \tag{2.31}$$

$$= \mathcal{L}(\theta, \phi; x) \tag{2.32}$$

where $\mathcal{L}$ is referred to as the *evidence lower bound*(ELBO)[109] and is optimized with respect respect to $\phi$ and $\theta$. Full derivation can be found in [69]. While effective on small scale data-sets, VAEs often struggle with complex high-dimensional data-sets [34]. A number of recent works have improved the quality of the generated samples through various techniques including improving the lower bound[15], using the W1 distance for the regularizer instead of the KL [119] and improving the priors distributions [120, 103].

### 2.2.3 Neural Density Estimators

We now look at constructing models that explicitly define density functions that are computationally tractable. Namely, *neural density estimators* are approaches that use neural networks to directly estimate the density[86]. Naturally, these are more suited to applications involving explicitly evaluating densities, rather than generating data. Furthermore, we regularly see these models utilised as components in larger data generating models, instead of explicitly generating the data themselves. The challenge in this setting is constructing a model that is flexible enough to capture the complexity of the densities while still being computational tractability. The two main families of neural density estimators are autoregressive models and normalizing flows[3].

As we have seen, often the intractability comes from the normalisation of the density in high dimensions. One way to overcome this is by restricting the class of models to those

[3]For the sake of brevity we only consider autoregressive models in this dissertation

that obey the *autoregressive* property;

$$p_\theta(\mathbf{x}) = p_\theta(\mathbf{x}_1) \prod_{t=2}^{T} p_\theta(\mathbf{x}_t|\mathbf{x}_{<t}), \tag{2.33}$$

where here the joint density has been decomposed into a product of conditional distributions using the chain rule of probability. This is a natural way to model temporal data as not only is it inherently auto-regressive, but in using this factorization it explicitly attends to the conditional distributions in the data. The maximum likelihood objective (2.13) is then broken down into a series of prediction problems

$$max_{\theta \in \mathcal{M}} \mathbb{E}_{\mathbf{x} \sim p_r}[\log p_\theta(\mathbf{x})] = max_{\theta \in \mathcal{M}} \mathbb{E}_{\mathbf{x} \sim p_r}[\log[p_\theta(\mathbf{x}_1) \prod_{t=2}^{T} p_\theta(\mathbf{x}_t|\mathbf{x}_{<t})]] \tag{2.34}$$

$$= max_{\theta \in \mathcal{M}} \mathbb{E}_{\mathbf{x} \sim p_r}[\log p_\theta(\mathbf{x}_1) + \sum_{t=2}^{T} \log p_\theta(\mathbf{x}_t|\mathbf{x}_{<t})] \tag{2.35}$$

However, in getting rid of the intractability problem we gain another as we find that the space complexity for representing each conditional $p_\theta(\mathbf{x}_t|\mathbf{x}_{<t})$ in full form grows exponentially with the size of the sequence. *Autoregressive Generative Models*[4] address this problem by specifying the conditionals as parameterized functions, whose parameters are a function of a hidden state $h_t$. As each conditional has a fixed number of parameters, the space complexity only grows linearly.

### Masked Multilayer Perceptrons

One such statistically and computationally efficient approach to this parameterization is to use multi-layer perceptrons (MLPs). Alternatively, this can be thought of as *masking* the weights on an MLP based auto-encoder. The first approach to adopt this was the masked auto-encoder for distribution estimator(MADE)[41], where they achieved competitive performance in a range of unsupervised distribution and density estimation tasks by placing fixed time masks on the MLP so as to explicitly enforce the auto-regressive property. The neural autoregressive density estimator(NADE)[122] then developed this by making the masks time-dependant and introducing a hidden layer. Since then there have been numerous approaches that has further increased performance [104, 50]. In the time-

[4]It is common to also see them referred to as Autoregressive Density Estimators. The distinction is largely down to the application of the models.

series domain, ExtraMAE [135] demonstrated that by introducing an *extrapolator* into the masked auto-encoder framework, it can outperform the state-of-the-art benchmarks in time-series generation.

### Recurrent Neural Networks

We next consider a family of autoregressive models[5] that model sequential data by encoding information on previous terms *recursively* through the hidden states. In this architecture, the hidden state at step t, $h_t$, is a function of the previous hidden state, $h_{t-1}$, and the input of the model at that step. Namely, Recurrent Neural Networks (RNNs)[111] are neural networks that achieve this recursion by feedback connections along the temporal sequence. The conditional probability for a simple RNN with one hidden layer is then parameterized as follows[22]

$$p_\theta(\mathbf{x}_t|\mathbf{x}_{<t}) = g_\theta(h_{t-1}) \tag{2.36}$$

where $\{h_t\}_{t=0}^T$ is the set of inner states recursively defined as follows

$$h_0 := 0, \quad h_s = f_\theta(h_{s-1}, a_s) \quad \forall s = 1, ..., t \tag{2.37}$$

$a_s$ is the input of the sequence at time-step s and $f_\theta$ and $g_\theta$ are learned functions, usually sigmoidal activation units. This parameter sharing across the model allows for context information to be propagated through multiple time steps [45]. Additionally the sequential nature of processing allows for sequences of variable length to be handled.

RNNs are trained using a backpropagation technique called backpropagation through time (BPTT) [96]. Here the network is first *unfolded* along the input sequence, to which the standard backpropagation algorithm is applied sequentially to the unfolded layers. Due to this unfolding, BPTT is especially susceptible to the vanishing/exploding gradient problem[55] as the error is required to propagate further than in other architectures. One popular way of mitigating these problems is by utilizing *Long Short-Term Memory*(LSTM)[56] or *Gated Recurrant Unit*(GRU)[21] networks.

[5]In its simplest form, represented by (2.37), some authors do not technically class RNNs as autoregressive models. They define autoregressive models as models where the output at time t depends *directly* on previous outputs $\mathbf{x}_{<t}$ rather than only on hidden states. RNNs can therefore be made to be auto-regressive by conditioning on previous outputs. A simple example of is $h_s = f_\theta(h_{s-1}, x_{s-1}, a_s)$ where now the output at time-step $s$ depends directly on the previous outputs

LSTMs are complex activation units that alleviate the vanishing gradient problem by allowing the gradients to flow unchanged through the networks. To this end each LSTM unit generally consists of a *memory cell*, a linear activation node where the information is stored, and several *gates*;

$$i_t = \sigma(W_i a_t + U_i h_{t-1} + b_i) \tag{2.38}$$

$$f_t = \sigma(W_f a_t + U_f h_{t-1} + b_f) \tag{2.39}$$

$$o_t = \sigma(W_o a_t + U_o h_{t-1} + b_o) \tag{2.40}$$

that control the flow of information into and out of this memory cell [56, 42]. Here $a_t \in \mathbb{R}^d$ denotes the input to the LSTM unit at time t, $h_t \in (-1, 1)^h$ denotes the hidden state of the LSTM unit at time t, $\sigma$ is the sigmoid function and $W \in \mathbb{R}^{hxd}, U \in \mathbb{R}^{hxh}$ and $b \in \mathbb{R}^h$ are the learnable weight matrices and bias vectors. Intuitively, the *input gate* $i_t$ determines whether new inputs should be stored in the memory cell while the *forget gate* $f_t$ determines which information can be erased from memory cell and finally the *output gate* $o_t$ determines which contents of the memory cell should be output as $x_t$, as well as to the next hidden state $h_{t-1}$. Specifically, let $c_t$ denote the memory cell state at time $t$ then

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \tag{2.41}$$

where $\odot$ is the Hadamard operator which performs element-wise multiplication. The next time-step's hidden state vector is then calculated as follows

$$h_t = o_t \odot \sigma(c_t) \tag{2.42}$$

Given this updated hidden state, the conditional probability is then found as in (2.36). As the gradients now backpropagate further without being diminished, the network can better capture the long range dependencies in the data. Despite these controls, LSTM networks have still been found to suffer from the exploding gradient problem [17].

Similar to LSTMs, GRUs aims the mitigate the gradient problems by controlling the flow of the information. Where they differ from LSTMs is that GRUs do not not have a memory

cell and instead rely on *reset* and *update* gates[21];

$$z_t = \sigma(W_z a_t + U_z h_{t-1} + b_z) \tag{2.43}$$

$$r_t = \sigma(W_r a_t + U_r h_{t-1} + b_r) \tag{2.44}$$

The next layer-wise hidden state vector is then calculated as follows

$$\hat{h}_t = \phi(W_h a_t + U_h(r_t \odot h_{t-1}) + b_h) \tag{2.45}$$

$$h_t = \sigma z_t \odot \hat{h}_t + (1 - z_t) \odot h_{t-1} \tag{2.46}$$

Intuitively, the update gate determines how much of the past information is to be kept and passed onto future states and the reset gate, similar to the forget gate in LSTMs, determines how much of the past information to forget. GRUs have been found to be more computationally efficient than LSTMs, due to their less complex structure[21]. Furthermore, they have been found to perform as well as or even better than LSTMs in a range of sequence modelling tasks on smaller data-sets[21]. In theory however, given sufficient data, LSTMs should outperform GRUs in modeling long-distance relations[16].

RNNs have historically been the neural network of choice for sequential data. They have successfully been applied to a range of domains including natural language modeling [90], speech recognition [8], robotics [88], finance[112] etc. Despite their advantages and wide-spread adoption, RNNs have some significant drawbacks. As well as the training instability issues already discussed, the sequential computation of the hidden states can make sampling from an RNN slow. This often limits their ability in applications that require real-time data generation.

### Causal Convolutions

An alternative autoregressive approach that does not not rely on sequential sampling are *causal convolutions*. Causal convolutions are defined as convolutions where the output at a specifc time only depends on the previous sequence elements[6]. A popular class of causal convolution architecture used in auto-regressive generative models are *temporal convolution networks* (TCNs) [9]. It's been found that TCNs, also referred to as *WaveNETs*, are able to capture long-range dependencies in sequences more effectively than LSTMs or GRUs

[6]For the sake of brevity the full mathematical definition of causal convolutions has been omitted. For a rigorous definition and great application please follow [129]

[9]. In addition to this, training TCNs is very stable as they they do not suffer from the vanishing/exploding gradient problem. However, as they do not process sequentially they can only capture a set max sequence length, called the *receptive field size*. To handle larger sequences, the model needs are larger receptive field size, which may cause memory issues. Models based on these architectures have since been developed to generate data in several domains [102, 32, 129].

### Attention Based

We next consider models that utilize *attention-mechanisms* to generate data. Similar to in our discussion of auto-encoders, these models are not inherently autoregressive, but regularly use masking techniques when modelling sequential data to enforce the property. In psychology, attention is *the concentration of awareness on some phenomenon to the exclusion of other stimuli*[1]. Similarly, in neural networks, attention-mechanisms attempt to enhance important parts of the data, by focusing on it more, while diminishing other parts, by focusing on it less. The model learns which parts of the data is important by looking at it in context of the rest of the data.

Before we formalise our understanding of attention mechanisms we first introduce *value vectors*. Given a set of items, $i$, the value vectors $v_i$ are just vector representations of these elements. Here the set of items could include words in a sentence or sections of an image. Attention mechanisms are just weighted averages of these value vectors

$$c = \sum_i w_i v_i \tag{2.47}$$

Intuitively, the larger the weight $w_i$ the more the attention-mechanism focuses on $v_i$ to generate its output. To prevent it from simply scaling the value vector the weights are normalised such that $\sum_i w_i = 1$ .The goal of attention mechanisms is to learn these weights as to focus on the important parts of the set of values so that the output performs well in specific tasks.

While there has been numerous proposals in how the set of weights should be calculated, in this section we'll follow [124] and look at the widely adopted *query-key* mechanism. Here for each attention mechanism there is corresponding *query* vector $q$ and for each value vector there is a corresponding *key* vector $k_i$. Similarly to the value vector these are just vector representations of elements in a set. Each weight is then calculated using a

*compatibility* function of the query vector and a key vector;

$$w_i = f(q, k_i) \tag{2.48}$$

One way to understand this is by viewing it as a retrieval process. It is analogous to retrieving a value for a query based on a key in a database. For a more real world example we imagine we are searching for something on eBay, a popular online market place. Here we'll type our request (*query*) into the search bar and the search engine will map this value against a set of the attributes of the products in their database (*keys*). From this, it will then present the best matched corresponding products (*values*). The compatibility function can be seen as quantifying how similar the query is with a specific key. As the values are all in vector form a natural and popular choice for quantifying similarity is taking the dot product of the vectors. In practice, the attention mechanism is computed on a set of queries simultaneously. Let us denote the packed together queries, keys and values by matrices Q, K and V respectively, then the *dot-product attention* is defined

$$\mathrm{A}(Q, K, V) = softmax\left(QK^T\right)V \tag{2.49}$$

When each key vector is equal to it's corresponding value vector then it's referred to as *self-attention*.

Since their inception attention mechanisms have become an essential part of effective sequence modelling. One of their inherent advantages is that they model the dependencies of the elements in the sequence regardless of their distance from one another [7][67] . This allows for effective modelling of long range dependencies compared to RNNs, where more emphasis is put on elements being close to one another, or CNNs, where an increase of operations to capture further points often makes it difficult. In early works, these mechanisms were often used in conjunction with recurrent or convolutional networks. Here they showed good performance in a wide variety of tasks including machine translation[7], visual object classification[93] and speech recognition[23].
However, recent works have advocated models that rely entirely on attention mechanisms. One such state-of-the-art model is the Transformer[124]. Here they follow a general encoder-decoder structure using stacked self-attention blocks and fully connected layers for both the encoder and decoder architectures[124]. As it does not process the data sequentially, it largely circumvent the vanishing gradient issues present in training RNNs. In this work, the authors used *Scaled Dot-Product Attention*, which, similarly to (2.49) relies on

dot-product operator to quantify similarity, but differs by scaling the inner dot-product;

$$\mathrm{A}_s(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{dk}}\right)V \tag{2.50}$$

where $d_k$ is the dimension of the keys. It's scaled in the fashion to prevent the product from growing too large, which diminishes the gradient and prevents effective learning[124]. Additionally, they introduce *multi-head attention*, where scaled dot-product attention[7] mechanisms are performed in parallel on a different sets of queries, keys and values;

$$\text{MultiHead(Q,K,V)} = Concat(head_1, ..., head_h)W^O \tag{2.51}$$

$$\text{where } head_i = A_s(QW_i^Q, KW_i^K, VW_i^V) \tag{2.52}$$

where $W_i^Q$, $W_i^K$, $W_i^V$ and $W^O$ are learned matrices [124]. The motivation for this is to leverage information from different representations of the data. A visualisation of the different heads of this attention in the image domain can be seen in Figure 2.2.

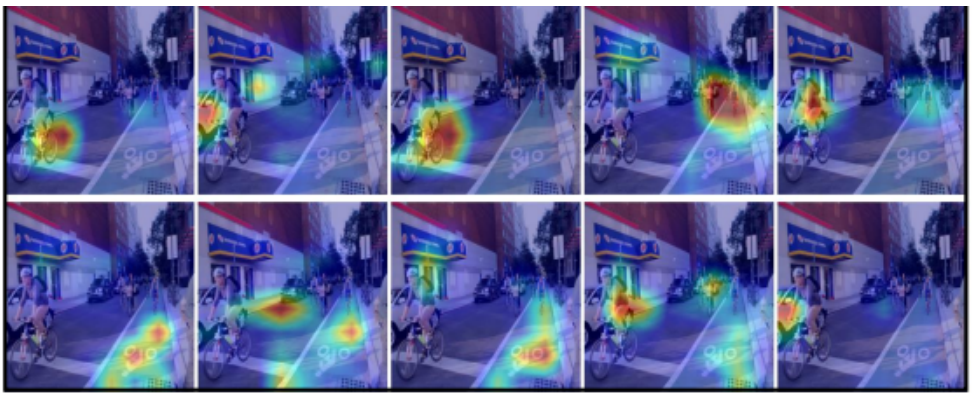


Figure 2.2: Image attention heatmap plots, where each subplot is a different head. Here the warmer colours correspond to areas of larger attention. Credit [105]

Here we see that across the different heads, the mechanism focuses on different aspects of the image. Concatenating the heads then results in the multi-head attention considering all the different representations. The learned matrix $W^0$ then determines which parts of all these heads contribute more to the final attention.

Transformer models have quickly become state-of-the-art in many tasks in the natural language domain, consistently out-performing RNN and CNN models. Popular examples

[7]While scaled-dot product attention was used in the original paper, any attention mechanism can be used to construct this module, depending on the application

include BERT[30], GPT-2/3[107] and RoBERTa[83]. For this reason they've become increasingly utilized in larger generative models that have seen impressive results in such domains as images[19, 20], videos[132] and time-series[40, 78, 114].

## 2.3 Implicit Density Models

We now look at the models that can be trained without explicitly using a density function. Since they don't directly access $p_\theta$, these models are mainly used for data generation rather than density estimation. To this end we define the deterministic generator function[8]

$$G_\theta : \mathcal{Z} \to \mathcal{X} \tag{2.53}$$

that maps samples from a tractable latent distribution $\eta$, to the data space $\mathcal{X}$. This generator is trained to map to points that *resemble* the underlying data. Similarly to auto-encoders, the latent space dimension will generally be smaller than that of the data space dimension to capture the low dimensional structure of the data[9]. An implicit generative model (IGM) is a family of probability distributions $\mathbb{G}_\theta$ parameterized by this learned generator function. Sampling from $\mathbb{G}_\theta$ is achieved by first sampling from the latent distribution and then applying $G_\theta$;

$$x \sim \mathbb{G} \leftrightarrow x = G_\theta(z), \qquad z \sim q \tag{2.54}$$

In theory an effective likelihood can be derived using the derivative of the cumulative distribution function [95], however given the dimensionality of the latent space, its highly intractable. Where in the explicit model case, when faced with intractability, often approximations of the likelihood are used, here we do not have the liberty. Instead we side-step this issue and rely on a learning approach based on discriminating real from synthetic data using hypothesis and two-sample testing. We will now looks at the most popular of these approaches, Generative Adversarial Networks.

[8] When using neural networks this function is often called a *neural sampler*. By this definition the decoder and some normalizing flows are also neural samplers

[9] Where this approach differs to auto-encoders is that this is an *unsupervised* learning problem. In unsupervised learning the algorithm learns from data that is neither classified nor labled. Autoencoders are trained by minimizing a loss between a sample and its reconstruction, which can be considered a *semi-supervised* learning problem.

### 2.3.1 Generative Adversarial Networks

Generative Adversarial Networks (GANs)[46] are a class of IGMs where the generator is trained *adversarially* by pitting it against a *discriminator* function;

$$D_\phi : \mathcal{X} \to \mathbb{R} \tag{2.55}$$

In the original paper[46], this discriminator was a binary classifier trained to determine whether the data was sampled from the underlying distribution or from the generator. Concurrently, the generator was trained to produce samples that the discriminator thinks is from the underlying data. Formally, learning the network is achieved by maximising $\mathcal{D}(\phi) := \mathbb{E}_{\mathbf{x}\sim p_r}[logD_\phi(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_z}[log(1 - D_\phi(G_\theta(\mathbf{z})))]$ over the set of parameters $\Phi$ of the discriminator, while minimizing $\mathcal{G}(\theta) := \mathbb{E}_{\mathbf{z}\sim p_z}[log(1 - D_\phi(G_\theta(\mathbf{z})))]$ over the set of parameters $\Theta$ of the generator. In this setting the generator and discriminator are playing a two-player minimax game[87, pp. 176–180] with the following objective

$$\min_\theta \max_\phi V(\phi, \theta) = \mathbb{E}_{\mathbf{x}\sim p_r}[logD_\phi(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_z}[log(1 - D_\phi(G_\theta(\mathbf{z})))] \tag{2.56}$$

The motivation for this can not only be seen in the game theoretic sense, but also if we consider the loss of the GAN when the discriminator is optimum.

$$\mathcal{L}(\theta, \phi^*) = 2D_{JS}(\mathbb{P}_r||\mathbb{P}_\theta) - 2\log 2 \tag{2.57}$$

where $D_{JS}$ is the Jensen-Shannon divergence (2.5). For full derivation see [46]. Similarly to our discussion of explicit density models, we can see this GAN approach as a proxy for our main goal of approximating the underlying distribution. Whereas in the maximum likelihood case the distance minimised is the KL divergence, here it's the JS divergence. Thus with sufficient capacity and data, the generator can recover the underlying data distribution.

#### Training Instability

The main challenges of training GANs is their instability and sensitivity to hyper-parameters. Two well studied issues are *diminishing/vanishing gradients* and *mode collapse*. To understand the vanishing gradient problem, in this context, we return to the loss when the discriminator is optimum (2.57). When there is not substantial overlap between $\mathbb{P}_r$ and $\mathbb{P}_\theta$, learning gradient will be 0 as the the JS divergence will be constant. In this case the

generator will cease to learn. As we have discussed, this is especially common in high dimensional data as the data concentrates in low-dimensional manifolds.

In the original work[46] the authors highlight that this vanishing gradient problem can be overcome through the minimization of $\mathbb{E}_{\mathbf{x}\sim p_\theta}[D_\phi(\mathbf{x})]$ for training $G_\theta$. However this leads to another problem, mode collapse. Mode collapse can be understood by considering the following form for the alternate loss when the discriminator is optimum

$$\mathcal{L}(\theta, \phi^*) = -\mathbb{E}_{x\sim p_\theta} \log[D_{\phi^*}(x)] \tag{2.58}$$

$$= D_{KL}(\mathbb{P}_\theta||\mathbb{P}_r) - 2D_{JS}(\mathbb{P}_r||\mathbb{P}_\theta) + 2\log 2 + \mathbb{E}_{x\sim p_r} \log[D_{\phi^*}(x)] \tag{2.59}$$

We find that the learning gradient is only affected by the first two terms in (2.58), as the last two don't depend on $\theta$. Additionally we see that this loss is dominated by the *reverse* KL divergence $D_{KL}(\mathbb{P}_\theta||\mathbb{P}_r)$ as $D_{JS}$ is bounded in [0, log 2]. Optimizing $\theta$ with the reverse KL divergences concentrates the probability mass on single modes to avoid the lower-probability areas. This manifests itself as the generator producing the same safe samples instead of producing a reasonable range.

### Variants

In original paper[46] MLPs were used for both the generator and the discriminator architecture. Since then, there have been many architecture-variants that have increased the generated sample quality and allowed the framework to be utilized in a range of domains[29, 10, 91]. However, it can be argued that the problems outlined in the training instability section cannot be solved through architecture design as they ultimately arise from the choice of loss function [127]. This motivated the research into loss-variant GANs, that extend the training procedure to different loss measures. One such development is Wasserstein GAN (WGAN) [6], where they utilize the W1 distance (2.7) as the loss measure. As we have discussed, IPMs posses many qualities that make them informative *critics*[10]. In particular, as they can provide informative distances for dis-joint distributions, they largely alleviate the vanishing gradient and mode collapse problem, increasing training stability. In this work they also find that by enforcing the Lipschitz constraint on the critic, there is a much more meaningful gradient for training the generator and in turn the GAN performs better. Another IPM that has seen success as the critic is the MMD.

[10] As the metrics are not used to discriminate they are referred to as critics.

Specifically, MMD GAN[11] has been shown to match the performance of WGAN with smaller architecture requirements resulting in a simpler and faster training.

Despite WGAN improving the stability of the training of GANs, it still often generates poor-quality samples. This is often attributed to the use of *weight clipping* to enforce the Lipschitz contraint on the critic. Here the weights of the model are clipped to ensure they are always in a user defined range [-c,c]. Implementing this weight clipping has been found to bias the critic towards simpler functions [47]. Alternatively, WGAN-GP[47] proposes enforcing this constraint through a *gradient penalty*. Specifically, they use objective

$$V(\phi, \theta) = \mathbb{E}_{x \sim \mathbb{P}_r}[D_\phi(x)] - \mathbb{E}_{z \sim \mathbb{P}_z}[D_\phi(G_\theta(z))] + \lambda \mathbb{E}_{\hat{x} \sim \mathbb{P}_{\hat{x}}}[(||\nabla_{\hat{x}} D_\phi(\hat{x})||_2 - 1)^2] \tag{2.60}$$

where samples are drawn from $\mathbb{P}_{\hat{x}}$ by sampling uniformly along straight lines between samples from the underlying distribution and samples from the generator distribution. The first two terms in this objective are the original WGAN critic loss while the last term is the penalty term. The $\lambda$ variable is then used to control how much the critic get penalised for having a large gradient norm. The authors showed that this set up outperforms the standard WGAN as well as having more stable training.

Since their inception GANs have been applied to a range of domains including natural language modelling [39][62], semantic segmentation [84][33] and computer vision[85][133]. They have achieved significant success in the latter, with state-of-the-art performances is tasks such as video generation [125], image-to-image translation [81] and photo-realistic image generation[77]. In the following section we will look at how this framework can be used in the time-series domain.

### Time-Series GANs

Similarly to the explicit models, the original GANs for sequential data relied mainly on RNN-based architectures. The first, C-RNN-GAN[94], generated sequential music data using stacked LSTMs layers for both the generator and discriminator architectures. Here the generator receives samples from the latent distribution at each time step and, conditioned on the previous outputs, generates data for that time step. The C-RNN-GAN model architecture is shown on Figure 2.3. Overall, it was found to be a successful in learning the characteristics of the continuous sequential data but the authors said there needs to be better metrics to better evaluate and compare the quality of the generated samples.

Specific details on the architecture for C-RNN-GAN can be found in Appendix A.

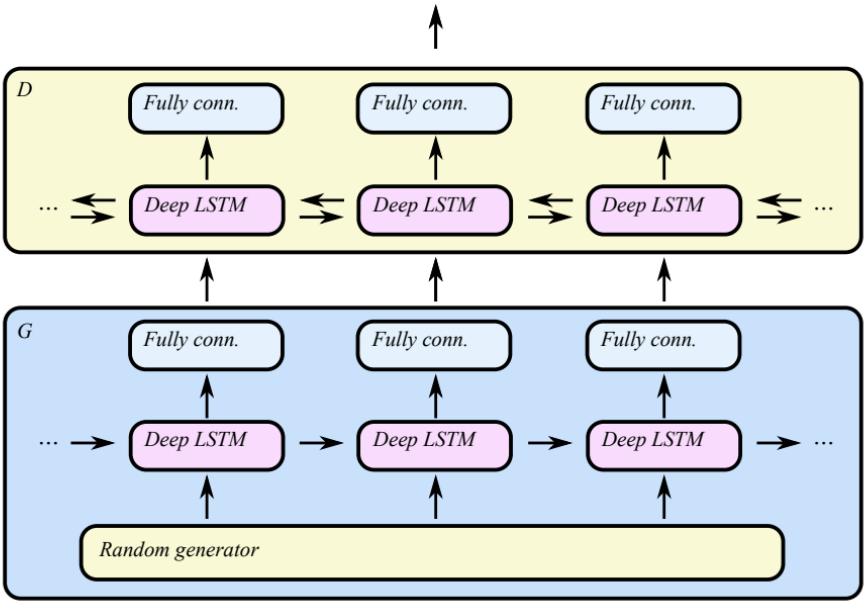



Figure 2.3: C-RNN-GAN architecture diagram from original paper[78]

Recurrent Conditional GAN (RCGAN) [38] then developed this architecture further by conditioning on additional auxiliary information as well as removing the dependence on the output of the generator. Here the focus was on generating medical time-series intended to be used in downstream tasks. Through several qualitative and quantitative metrics they demonstrated that RCGANs can generate time-series useful for supervised training.

Due to their better capturing of long-range dependencies in time-series compared to RNNs, recent works have advocated the use of causal convolutions in the GAN framework. The first model to implement them was WaveGAN [32], which developed ideas from DCGAN to synthesis raw-waveform audio. QuantGAN [129] then developed this further by utilizing TCNs with skip connections in both the generator and discriminator networks with the goal of generating financial time-series. It was shown that this GAN approach can outperform more traditional models from mathematical finance, but the authors raised the issue of a lack of unifying metric to evaluate the model's performance.

However, it can be argued that relying solely on a single adversarial loss may not be suf-

ficient to ensure the network captures the temporal correlations of the time-series[134]. This motivated approaches whereby additional losses and networks are added that explicitly train the model to attend to the unique characteristics of data. Namely, TimeGAN[134] introduces a stepwise supervised loss to the framework, using the training data as supervision, to encourage the model to capture the conditional distributions in the data. In addition to this, they introduced an embedding network to learn the lower-dimensional factors of variation in the data and reduce the dimensions of the learning space. They found that TimeGAN outperformed state-of-the-art benchmarks including C-RNN-GAN, RCGAN, WaveGAN and Professor Forcing on a range of real-life and synthetic data-sets

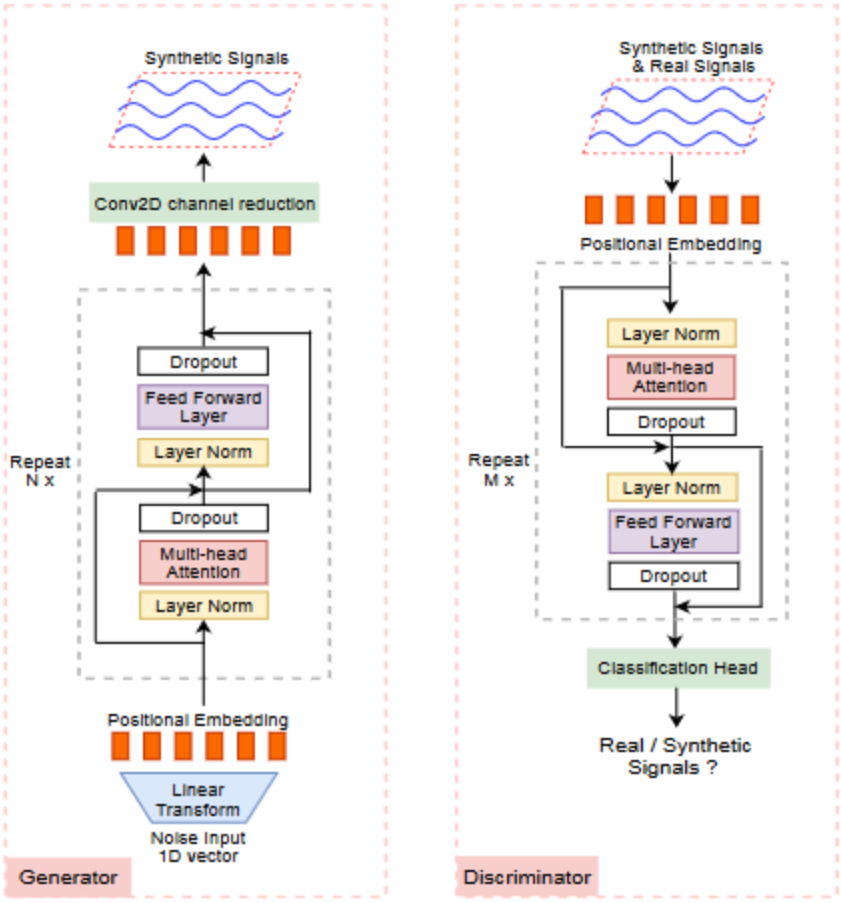



Figure 2.4: TTS-GAN architecture diagram from original paper [78]

Following the success of attention-based models in sequence modelling tasks, numerous works have proposed incorporating these mechanisms into GAN frameworks for time-series generation[40, 106]. One such framework is TTS-GAN[78], where the transformer encoder

architecture is used for both the generator and discriminator. Figure 2.4 shows the model architecture of TTS-GAN. Similarly to RCGAN, the focus in this work is on generating medical time-series data to be used in downstream tasks. Performance was assessed against the then state-of-the-art TimeGAN, with TTS-GAN outperforming TimeGAN in 7 out of the 10 datasets. Specific details on the architecture for TTS-GAN can be found in Appendix B. Another transformer based time-series GAN is TsT-GAN[114]. Here they develop on the ideas in TimeGAN and introduce two extra networks and several losses to the GAN framework to better capture the global distribution and conditional time-series dynamics.

Time-series GANs are a natural choice for synthesizing medical and physiological data as not only do they outperform alternatives in certain classification and recognition tasks [138] but samples generated can adhere to stricter privacy constraints [80]. It's for these reasons that we've seen them widely adopted in this domain [38, 49, 28, 139]. Another field that relies on heavily on time-series data but regularly suffers from insufficient or unavailable real data is finance. Here GANs have seen impressive results, regularly outperforming traditional mathematical finance models in quality of the generated financial time-series [129][117]. Along with the training stability, one of the biggest challenges of utlizing GANs, and especially time-series GANs, is that they are difficult to evaluate.

**Evaluation**

A ramification of relying on a learning approach based on discriminating between real and synthetic data is that there is no single objective loss function. This can make assessing the progress of training or quality of the models challenging. While there have been plenty of evaluation measures proposed [12, 13], researchers have not yet agreed which measures best capture the performance of the models. This problem is particularly apparent when evaluating time series GANs as most of the metrics proposed in the literature have been developed for the computer vision domain. Current measures can be split into two categories; qualitative, where the measures are not numerical and often involve human evaluation and quantitative, where the quality of the generated samples are summarized numerically.

The simplest and most characteristic qualitative evaluation is to use human reviewers to compare the visual quality and diversity of the generated samples. One drawback in this approach is its subjectivity; using reviewers to assess the quality of the samples inherently include biases from the reviewer. Another significant drawback is that it requires that each of the reviewers have knowledge of the target domain which might be infeasible with more

complex objectives. In practice this approach is slow due to the limited by the amount of samples a reviewer can evaluate in a reasonable time. Examples of these methods include *Rating and Preference Judgment* where the reviewers rank or compare real and generated samples and *Rapid Scene Categorization* which is generally the same process as Rating and Preference Judgment but the samples will only be presented to the reviewers for a very small amount of time.

When using human reviewers to evaluate the performance of image-based and other media-based GANs, often the samples are compared and ranked directly. However, time-series data is often difficult to interpret in a human psycho-perpetual sense so instead it's common to evaluate low-dimensional representations of the data. Two popular techniques for this dimension reduction are t-distributed stochastic neighbor embedding(t-SNE)[123] analysis and Principal Component Analysis(PCA)[20].

PCA reduces the dimensions through changing of basis of the data using a limited number of *principle components*. The principle components of the data are a sequence of unit vectors where each vector in the sequence is the direction of best fit[11] of the data, while being orthogonal to the previous vectors. Equivalently, we can view this as transforming the data using a new coordinate system that best preserves the variance in the data.

t-SNE is a non-parametric approach that reduces the dimensions by directly optimizing a set of low-dimensional points. To understand t-SNE we first need to define a stochastic neighbor embedding(SNE)[53]. Given a set of high dimensional points $\{\mathbf{x}_n\}$, SNE first constructs a probability distribution over pairs of these points using their relative *similarity*. Specifically for $i \neq j$, we define the similarity of $x_j$ to $x_i$ to be

$$p_{j|i} = \frac{\exp(-||\mathbf{x}_i - \mathbf{x}_j||^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(-||\mathbf{x}_i - \mathbf{x}_j||^2/2\sigma_i^2)} \tag{2.61}$$

where $p_{i|i} = 0$ and $\sigma_i$ is calculated such that only a set number of the closest points to $x_i$ have a non-negligible similarity. The probability is then computed by symmetrizing the similarities

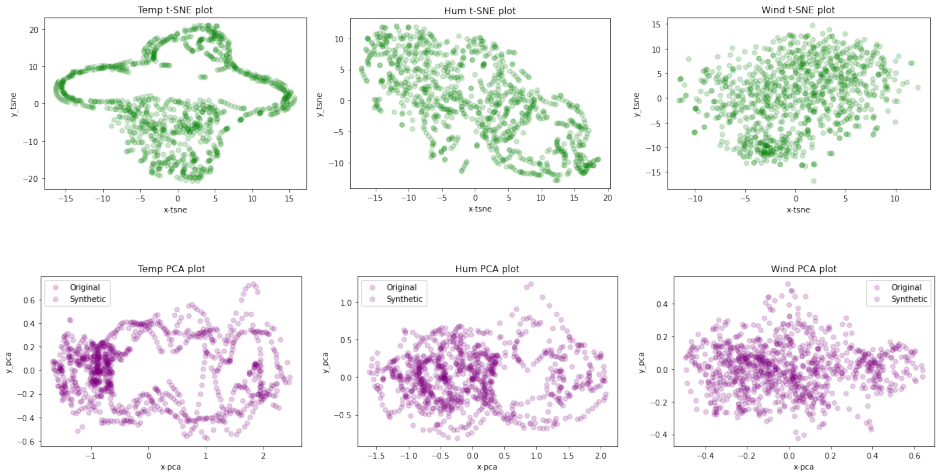
$$p_{ij} = (p_{i|j} + p_{j|i})/2N. \tag{2.62}$$

[11]Here the direction of best fit of the data corresponds to finding the line that minimizes the squared perpendicular distance points in the space and the line. It can be shown [] that this objective is equivalent to the maximizing of the variance ().

The goal of SNE is find a set of low-dimensional points $\{\mathbf{y}_n\}$ that best represent $\{\mathbf{x}_n\}$. SNE achieves this by minimizing the KL divergence between P, as we have defined above, and Q, where $q_{ij}$ is the similarity of $\mathbf{y}_j$ to $\mathbf{y}_i$. In the original work [53], this low-dimensional similarity was measured using the Gaussian kernel, t-SNE then extended this by proposing the use of the t-distribution kernel, such that

$$q_{ij} = \frac{(1 + ||\mathbf{y}_i - \mathbf{y}_j||^2)^{-1}}{\sum_k \sum_{l \neq k} (1 + ||\mathbf{y}_k - \mathbf{y}_l||^2)^{-1}} \tag{2.63}$$

By minimizing this divergence the algorithm aligns the lower dimensional points as to capture the similarities in the high dimensional points. Visualisations of both the t-SNE and PCA plots for a number of time-series can be seen in Figure 2.5




(a) Mean temperature (b) Humidity (c) Wind speed

Figure 2.5: t-SNE and PCA visualistions. Here we consider three time-series from Delhi Weather data-set [2], recording the mean temperature (a), humidity (b) and wind speed (c) in Delhi. The first row, in green, are the corresponding t-SNE plots for each time-series and the second row, in purple, are the corresponding PCA plots.

In the original GAN paper [46] the authors quantitatively summarized the generated images using the Average Log-likelihood, also called the *Parzen density estimation*. However this has been found to not be effective as an evaluation metric for GANs as it is bias towards trivial models. Since then a range of quantitative measures have been proposed for a wide variety of target-domains. Specifically, two widely adopted metrics that were first utilized in the image-domain but have since permeated across domains are the Inception Score(IS)[116] and the Frechet Inception Distance(FID)[51] score. These are calculated us-

ing a pre-trained image-classification network called Inception v3 [116]. Another popular approach, which has application across domains, is to use IPMs. Specifically, the MMD has been shown to be a suitable evaluation metric in several domains including the time-series domain. Additionally, the sliced-wasserstein distance[71], which approximates the general wasserstein distance, has been been shown to generalise well to the sequential data and is regularly adopted in that domain .

In the time-series domain, often classical measures associated with time-series analysis are used to quantitatively evaluate the generated samples including Pearson Correlation Coefficient (PCC), Mean Squared Error (MSE/RMSE), Mean Absolute Error (MAE) etc. The limitation of these metrics is that they measure very particular characteristics of the time-series so often a range of metrics are needed to determine the overall fidelity. A lot of metrics will depend on the target-domain; in finance its common to use the autocorrelation function score or DY metric where as in audio generation popular metrics include Normalised Source-to-Distortion Ratio and Source-to-artifact ratio.

Using separate networks to assess the performance of the generated samples in is a common approach to quantifying the performance of GANs. As we have seen, both the IS and FID score are computed using an additional pre-trained network. Another method is *Train on Synthetic, Test on Real/Train on Real, Test on Synthetic*. Here, given a supervised task that can be defined on the domain, a data-set of either synthetic or real data is used to train a model to perform this task. The performance of held-out sets from both data-sets are then measured using the trained models. The TSTR framework is training the model with the synthetic data and evaluating with the real data and visa versa. For a benchmark of how the the true data performs in the tasks, both the synthetic and real held out sets are also evaluated on the corresponding synthetic and real trained models. The motivation for this method is to quantify how well the synthetic data performs in tasks compared to the real data. This is particularly relevant if the generated samples are intended to be used in downstream tasks.

Compared to other DGMs, GANs often generate better quality and more diverse samples. This can be attributed to their ability to handle complex density functions and lack of deterministic bias. As we have seen the main disadvantage of using GANs is that the training is unstable and generally long. Sampling from GANs, compared to autoregressive models, is very quick as it requires one pass through the network.

## 2.4 Chapter Summary

In this chapter we provided an overview of current deep generative models and their application in the time-series domain. First we looked at several explicit models including energy based models, variational auto-encoders and neural density estimators before looking at an important implicit density model, the generative adversarial network. In our discussion we have not only looked at the ideas behind the main generative models but looked at the smaller models that often feature as core components. Through this, we have not only we explicitly achieved the first objective of this dissertation but set the foundations for the achieving the second and third objective.

# Chapter 3

# Generalized Energy Based Models

We now turn our attention to the main models of interest in this dissertation. Namely, this chapter aims to give an introduction to Generalized Energy Based Models. We begin by first providing motivation for the models by assessing the limitations present in several explicit and implicit DGMs. We then introduce the models using the definition provided to us in the original work and look at how the authors proposed they should be trained. Finally we will provide details on how one can sample from these models and review the experiments performed in the original paper.

We note an important disclaimer that a lot of the definitions, derivations and figures in this chapter are from the original paper[5] introducing the models. This chapter therefore not only serves to introduce the models but the work in that paper. For that reason, throughout the chapter we will refer to the Arbel et al. as *the authors*. For more information on these powerful models we advise the reader to go check it out.

## 3.1 Motivation

In the previous chapter, we introduced several explicit and implicit deep generative models. Within this we dicussed EBMs , a class of explicit models that utilizes energy functions that associate realistic data with low scalar values and unrealistic data with high scalar values. We found these models to be stable and effective way to represent complex high dimensional data. A limitation of EBMs is that the energies specify a probability density over the whole data space $\mathcal{X}$. While not an issue if the high dimensional data resides in the whole space, we often find that this is not the case and it concentrates in some lower dimensional

manifold. This means that putting mass on the whole space will inevitably result in *blurry* samples as the model has not learned this low-dimensional support. Conversely, in our discussion of implicit models, we found IGMs to be incredibly effective in learning this lower dimensional support but unable to refine the mass on this support to produce more probable samples. This motivated the creation of *Generalized Energy Based Models*(GEBMs)[5] which combines the benefits of both implicit and explicit models.

## 3.2 Background

Before we formally introduce GEBMs, we first need to understand an important component in the training process, the KL Approximate Lower-bound Estimate. For this, we first provide a variational estimation of the $\phi$-divergences before looking at the particular case of the KL-divergence.

### 3.2.1 Variational Estimation of $\phi$-divergences

As we have discussed, a major limitation of using $\phi$-divergences is that they are non-informative if the distributions have disjoint support. To overcome this it is common to instead use the a *variational lower bound* of the divergences. In this section we will follow the divergence estimation procedure presented in [100]. For completeness, first we provide some basic definitions from convex analysis. Given a convex function $f$, we define it's *subdifferential* [110, p. 242] at point $t \in \mathbb{R}$ as the set

$$\partial f(t) := \{z \in \mathbb{R} | f(s) \geq f(t) + z(s-t) \quad \forall s \in \mathbb{R}\} \tag{3.1}$$

In the case that f is differentiable at point t, then the subdifferential at point t is the differential at that point, such that $\partial f(t) = f'(t)$ . Additionally if $f$ is convex and lower-semicontinuous, we define its *conjugate dual function* or *fenchel dual* as

$$f^*(t) = \sup_{u \in dom_f} \{ut - f(u)\} \tag{3.2}$$

This function has the property $f^{**} = f$, therefore we can represent $f$ as

$$f(u) = \sup_{t \in dom_{f*}} \{tu - f^*(t)\} \tag{3.3}$$

By definition the $\phi$ function of the $\phi$-divergences is convex and lower-semicontinuous, so their definition can be reformulated using the dual of $\phi$ as follows

$$D_\phi(P||Q) = \int_{\mathcal{X}} q(x)\phi\left(\frac{p(x)}{q(x)}\right)dx \tag{3.4}$$

$$= \int_{\mathcal{X}} q(x) \sup_{t\in dom_{\phi*}} \left\{ t\frac{p(x)}{q(x)} - \phi^*(t) \right\} dx \tag{3.5}$$

Given this definition, a lower bound for the divergence can be yielded[1] by applying Jensen's inequality[61] and restricting the class of functions to a subset of all functions,

$$D_\phi(P||Q) \geq \sup_{f\in\mathcal{H}} \left( \int_{\mathcal{X}} p(x)f(x)dx - \int_{\mathcal{X}} q(x)\phi^*(f(x))dx \right) \tag{3.6}$$

$$= \sup_{f\in\mathcal{H}} (\mathbb{E}_{x\sim P} f(x) - \mathbb{E}_{x\sim Q}\phi^*(f(x))), \tag{3.7}$$

where $\mathcal{H}$ is an arbitrary class of functions $f : \mathcal{X} \to \mathbb{R}$. This bound is *tight*, such that equality in the supremum is attained, if $f(x) \in \partial\phi\left(\frac{q(x)}{p(x)}\right)$ for any $x \in \mathcal{X}$ [100]. This is the variational lower bound of the $\phi$-divergence.

### 3.2.2 KL Approximate Lower-bound Estimate

We now find the variational lower bound of the KL divergence. In this case, the conjugate dual of the $\phi$ function is

$$\phi^*(t) = \begin{cases} -1 - \log(-t) & \text{if } t < 0 \\ +\infty & \text{otherwise} \end{cases} \tag{3.8}$$

Substituting this dual into the lower bound (3.6) we find the *KL Approximate Lower-bound Estimate*(KALE)

$$D_{KL}(P||Q) \geq \sup_{f\in\mathcal{H}} (-\mathbb{E}_{x\sim P} f(x) + 1 - \mathbb{E}_{x\sim Q} \exp(-f(x))) \tag{3.9}$$

$$= KALE(P||Q), \tag{3.10}$$

[1] Either of these admit a lower bound

also referred to as the KALE divergence, between P and Q. Given a sample from both distributions this can then be estimated by the following estimator

$$KALE(P||Q) \approx \sup_{f \in \mathcal{H}} \left[ -\frac{1}{n}\sum_{i=1}^{n} f(x_i) - \frac{1}{m}\sum_{j=1}^{m} \exp(-f(y_i)) \right] + 1 \tag{3.11}$$

Here we see that the definition of the KALE divergence resembles that of the IPMs (2.6). Indeed, it's been shown that the KALE divergence share similar empirical properties with the IPMs, that makes it a useful measure in the learning domain [43]. The first is, like the IPMs, the KALE remains well defined for mutually singular distributions and the second is that the KALE inherits a lot of the topological strength of the KL divergence. The combination of these make the KALE divergence especially suited to when distributions are supported on low-dimensional manifolds.

We will now provide details of the topological properties of KALE found in [5]. Given compact domain $\mathcal{X}$, $\mathcal{H}$ is dense in the space $\mathrm{C}(\mathcal{X})$[2] of continuous functions on $\mathcal{X}$ wrt $||\cdot||_\infty$. If $f \in \mathcal{H}$ then $-f \in \mathcal{H}$ and $cf \in \mathcal{H}$ for $0 \leq c \leq C_{max}$. Then

$$KALE(P||Q) \geq 0 \quad and \quad KALE(P||Q) = 0 \quad \text{iff} \quad P = Q \tag{3.12}$$

Informally, this result shows us that the KALE correctly informs us once the distributions match. However it doesn't tell us how well this divergence measures when the distributions don't match. If we add the additional requirements that all functions in $\mathcal{H}$ are L-Lipschitz then

$$KALE(P||Q^n) \to 0 \quad \text{iff} \quad Q^n \to P \text{ under the weak topology,} \tag{3.13}$$

with proofs for both statements provided in [5, Appendix C.2.1]. This shows that as Q approaches P then KALE goes to 0 in a well defined way. These properties further reinforce our belief that the KALE divergence can be used as a meaningful critic.

[2]Lucky for us it's been shown that this dense assumption holds when the function class $\mathcal{H}$ contains feed-forward neural networks[136]. This allows us to use a neural network approach to determine the KALE.

## 3.3 Generalized Energy-Based Model

A Generalized Energy Based Models (GEBMs), $\mathbb{Q}$, consists of two components: a base distribution $\mathbb{B}_\theta$, often chosen to be an Implicit Generative Model, and an energy function $E : \mathcal{X} \to \mathbb{R}$ defined over a subset $\mathcal{X}$ of $\mathbb{R}^d$.

In the case that the base is an IGM, sampling from $\mathbb{B}_\theta$ is achieved by first sampling $z$ from a simple latent distribution $\eta$ and then applying a learnable function $B_\theta$. Where this differs from a conventional IGM is that the samples from this base distribution, $x$, are re-weighted by the importance weights determined by the energy function. Analogously to EBMs, the density of the GEBM is then defined

$$q_{\mathbb{B},E}(x) = \frac{\exp(-E(x))}{Z_{\mathbb{B},E}}, \quad Z_{\mathbb{B},E} = \int exp(-E(x))d\mathbb{B}_\theta(x), \tag{3.14}$$

From this we can then express the distribution of the GEBM, as in the original work

$$\mathbb{Q}(dx) = exp(-E(x) - A_{\mathbb{B},E})\mathbb{B}(dx), \tag{3.15}$$

where $A_{\mathbb{B},E} = \log(Z_{\mathbb{G},E})$ is the log-partition. As such, $q_{\mathbb{B},E}(x)$ is a *Radon-Nikodym derivative*[101] of $\mathbb{Q}$ wrt $\mathbb{B}_\theta$. To understand the advantages of this approach we consider the visualisations in Figure 3.1. Here the authors conducted an experiment where they used a GAN, GEBM and EBM to model a distribution supported on a line. We see that the GAN (b) learns the correct support of the distribution but puts equal density across the line, whereas the ground truth density (a) is higher at the ends. Contrarily, we see that the EBM (b) puts higher density at the end similar to the ground truth, but fails to learn the underlying shape of the distribution. The GEBM (c) however not only learns the correct support but the density on the support.

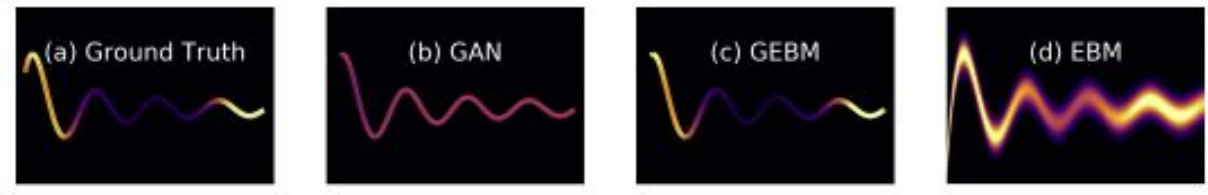



Figure 3.1: Visualisation of the performance of GEBM, GAN and EBM in a simple density estimation task. Credit [5]

## 3.4 Learning GEBMs

Here we'll describe the training procedure the authors proposed for GEBMs. In a similar manner to GANs, training is undertook in an alternating fashion between it's two components. Specifically, the procedure alternates between the *energy learning* step, where the base $\mathbb{B}$ is fixed and E is learned through maximising a likelihood on the fixed support, and a *base learning* step, where now the energy is fixed and $\mathbb{B}$ is learned.

### 3.4.1 Energy Learning

As we have seen in our discussion of explicit density models, given a explicitly defined density, a natural choice for learning the energy is to maximize the likelihood of the model. However, as $\mathbb{B}$ is supported on a low dimensional manifold, the conventional likelihood maximisation cannot be done as its density will not be defined over the whole space. Instead, the authors introduce a generalized notion of likelihood to maximise. Specifically they define the *expected* $\mathbb{B}$*-log-likelihood* under a target distribution $\mathbb{P}_r$ of a GEBM $\mathbb{Q}$ with base $\mathbb{B}$ and energy E as

$$\mathcal{L}_{\mathbb{P}_r,\mathbb{B}}(E) := \int \log(q_{\mathbb{B},E}(x))d\mathbb{P}_r(x) = -\int E(x)d\mathbb{P}_r(x) - A_{\mathbb{B},E} \tag{3.16}$$

The maximum generalized likelihood energy $E^*$ is then learned through maximising (3.16) w.r.t E. This likelihood can be estimated through the following estimator

$$\hat{\mathcal{L}}_{\mathbb{P}_r,\mathbb{B}}(E) = -\frac{1}{N}\sum_{n=1}^{N} E(x^{(n)}) - \log\left(\frac{1}{M}\sum_{m=1}^{M}\exp(-E(y^{(m)}))\right) \tag{3.17}$$

where $\{x^{(n)}\}_{1:N}$ and $\{y^{(m)}\}_{1:M}$ are i.i.d samples from the underlying distribution $\mathbb{P}_r$ and the base $\mathbb{B}$ respectively[121, 82]. However, an issue arises when training the model using mini-batch stochastic optimization methods. When training using mini-batches, small samples of the data are used to gradually update the parameters of the network. If a sample is too small then the last term is often a poor estimate of $A_{\mathbb{B},E}$. The authors overcome this by also maximizing over a variational parameter used to estimate $A_{\mathbb{B},E}$. Specifically, they use the convexity of the exponential function to determine a lower-bound of the log-partition

$$-A_{\mathbb{B},E} \geq -A - \exp(-A + A_{\mathbb{B},E}) + 1 \tag{3.18}$$

for $A \in \mathbb{R}$. Here equality is only achieved when $A = A_{\mathbb{B},E}$. They then use this lower-bound to determine a lower bound for the generalized likelihood

$$\mathcal{L}_{\mathbb{P}_r,\mathbb{B}}(E) = -\int E(x)d\mathbb{P}_r(x) - A_{\mathbb{B},E} \tag{3.19}$$

$$\geq -\int E(x)d\mathbb{P}_r(x) - A - \exp(-A + A_{\mathbb{B},E}) + 1 \tag{3.20}$$

$$= -\int (E(x) + A)d\mathbb{P}_r(x) - \int \exp(-(E(x) + A))d\mathbb{B}(x) + 1 \tag{3.21}$$

where they denote (3.21) as the functional $\mathcal{F}_{\mathbb{P},\mathbb{B}}$(E+A). As maximising (3.18) over A recovers $A = A_{\mathbb{B},E}$, so we find that that maximising $\mathcal{F}_{\mathbb{P},\mathbb{B}}$(E + A) over A recovers $\mathcal{L}_{\mathbb{P},\mathbb{B}}(E)$. Thus maximising $\mathcal{F}_{\mathbb{P},\mathbb{B}}$ over both E and A yields the maximum likelihood energy $E^*$. This functional can be estimated using the following estimator;

$$\hat{\mathcal{F}}_{\mathbb{P}_r,\mathbb{B}}(E + A) = -\frac{1}{N}\sum_{n=1}^{N} E(x^{(n)} + A) - \frac{1}{M}\sum_{m=1}^{M} \exp(-E(y^{(m)} + A)) + 1, \tag{3.22}$$

which contrary to (3.17), is well suited to mini-batch stochastic gradient methods. To summarise, the energy is learned by jointly maximising $\mathcal{F}_{\mathbb{P},\mathbb{B}}$(E+A) over the parameters $\Phi$ of the energy network.

### 3.4.2 Base Learning

To learn the base $\mathbb{B}$, the authors propose minimizing the expected $\mathbb{G}$-log-likelihood (3.16) evaluated at the optimal $E^*$. To understand the motivation for this we first notice that this likelihood evaluated at $E^*$ is equivalently the KALE divergence (3.9) between $\mathbb{P}$ and $\mathbb{B}$

$$KALE(\mathbb{P}_r||\mathbb{B}) = \sup_{(E,A)\in\mathcal{E}x\mathbb{R}} \mathcal{F}_{\mathbb{P}_r,\mathbb{B}}(E + A) \tag{3.23}$$

where the function class is the energy functions $E \in \mathcal{E}$. As we have seen in Section 3.2.2, the KALE divergence has desirable convergence properties that makes it a natural choice for the criterion. As we're in the parametric setting, the base is learned by minimizing $\mathcal{K}(\theta) := \text{KALE}(\mathbb{P}||\mathbb{B}_\theta)$ over the parameters $\Theta$ of the base generator function. If we recall the from Section 3.2.2, one of the requirements for the distributions to converge under

weak topology when the KALE is minimized is that all functions in $\mathcal{H}$ are L-Lipschitz. To train the base it is therefore necessary to regularise the energies to ensure this Lipschitz assumption is met. As we have seen in our discussion on the regularised variants of GANS, this regularisation can be achieved through methods such as spectral normalization and the gradient penalty.

## 3.5 Sampling from GEBMs

Given the unreliability[27] and inefficiency[48] of methods that sample in the the data-space, the authors instead propose sampling from a *posterior latent* distribution over the latent space

$$v(z) := \eta(z) \exp(-E(B(z)) - A_{\mathbb{B},E}) \tag{3.24}$$

using MCMC and then applying the base generator function to the samples such that

$$x \sim \mathbb{Q} \quad \leftrightarrow \quad x = B_\theta(z), \qquad z \sim v \tag{3.25}$$

Sampling from this distribution allows the GEBM to learn a richer latent disitibution. Specifically, where IGMs usually have to distort a *unimodal* simple latent distribution into a *multimodel* distribution[137], GEBMs instead allow the latent noise to be multimodel. The benefits of this are two-fold; first, as we have discussed, it allows the model to put differing mass on regions of the support, and second that it requires less architecture in order to distort the distribution [5].

Despite these benefits, sampling from the posterior latent distribution is often hard due to the dependence on the complex energy and base functions. Similar to EBMs, the authors overcome this through the use of Langevin MCMC. Specifically they consider two samplers. The first is *overdamped samplers*, where the samples are updated according the the iterative process (2.21), where here the target distribution is the posterior distribution $v$. Specifically, the Unadjusted Langevin Algorithm (ULA) is given by the following iterative process

$$Z_0 := z_0, \quad Z_{k+1} = Z_k + \lambda \nabla \log v(Z_k) + \sqrt{2\lambda}\epsilon_k \tag{3.26}$$

$$= Z_k + \lambda(\nabla \log \eta(Z_k) - \nabla E(G(Z_k))) + \sqrt{2\lambda}\epsilon_k \tag{3.27}$$

As well they consider *kinetic samplers*, where the samples updated according to the iterative process (2.22). Specifically, they consider the algorithm from, which they refer to as Kinetic Langevin Algorithm(KLA).

## 3.6 Experiments

In the original work the authors conducted two experiments; first in image generation and second in density estimation. For relevance to this dissertation we only look at the image generation experiment. Here they assessed the performance of the GEBM by comparing the FID scores of samples generated via the KLA, to samples from the base of the GEBM, without incorporating energy, as well as against samples from the GAN with the equivalent architecture. Here for the base and energy architectures they used the generator and discriminator architecture of SNGAN[92]. They found that by incorporating the energy it results in lower (better) scores. Additionally, they found that the scores of the samples from the base was comparable to that the scores of the samples from the GAN which implies that the performance gain was not just from adopting the KALE objective.

## 3.7 Chapter Summary

In this chapter we have provided the reader with an overview generalized energy based models. In this discussion, we first gave motivation for their creation before looking at how they can be trained and sampled from. Finally we discussed the experiments conducted in the original paper to demonstrate their ability. In doing so we have not only achieved the third objective of the dissertation, but provided the foundations for the fourth objective.

# Chapter 4

# Experiments

In this chapter we outline the experiments we conduct to achieve the goals of the dissertation. To this end, first we provide details on the models and the motivation behind our choices. Following that, we provide details on the metrics and data-sets that were used to assess the performance of the models. Finally we provide specific implementation details so that the experiments may be recreated.

## 4.1 Models

Following the intuition of the experiments performed in the original work, we base the base and energy architecture of the time-series GEBMs on the generator and discriminator architectures of a time-series GAN. In this dissertation we are not only interested in investigating whether a GEBM approach to time-series generation is viable, but also investigating the effectiveness of the approach across different architectures. To this end, we considered two network architectures for the base and energy, one based on RNNs and the other based on attention-mechanisms. These are a natural choice as they are the most widely adopted in time-series GAN literature. To assess the performance of the models we compare the quality of the samples from the GEBMs, sampled following the ULA (3.26), to samples from the base of the GEBMs, without using the energy information. This comparison will allow us to determine whether incorporating the energy learnt in the training process produces more realistic time-series.

Additionally, for both architectures, we considered the performance of the samples from their respective GANs with the same generator and discriminator architecture. This will allow us to determine the overall utlity of the models as well as how much the performance

change can be attributed to using the KALE objective.

#### Recurrent Based

For the base and energy of the recurrent-based GEBM we used the generator and discriminator of C-RNN-GAN [Appendix A]. Here a 5-dimensional uniform distribution is used for the latent noise $\eta$ at each time-step. Although there are better performing RNN based GANs, we choose this simple architecture as our objective is to demonstrate the effect of a GEBM approach rather than to achieve state-of-the-art results. We will refer to the samples generated by this model as **R-GEBM**. For the samples from the base of the GEBM only we will refer to as **R-KALE** and those from the standard GAN we'll refer to **R-GAN**

#### Attention Based

For the base and energy of the attention-based GEBM we used the generator and discriminator of TTS-GAN [Appendix B]. Here a 100-dimensional gaussian was used for the latent noise $\eta$. Similarly to the recurrant based model, we chose this simple architecture to demonstrate the effect of a GEBM approach rather than to achieve state-of-the-art results. We will refer to the samples generated by this model as **T-GEBM**. Similarly, the samples from the base of the GEBM only we will refer to as **T-KALE** and those from the standard GAN we'll refer to **T-GAN**

## 4.2 Evaluation Methodology

As we have previously discussed, there is not a standard metric for measuring the quality of generated time-series. Here we follow [134][114] and measure performance by qualitatively/quantitatively assessing whether the generated samples posses three desirable qualities a synthetic time-series should have;

#### Fidelity

First we want the samples from the model to be indistinguishable from the underlying data. To assess this we find the *discriminator score*. To compute this score we first label a set of synthetic samples and a set of samples from the underlying distribution. After combining the data and splitting into a train set and a hold out *test set*, we train a simple

binary classification model using the train set in a supervised fashion. The discriminator score is the jaccard score[1] [97] between the predicted labels of the test set and it's real labels. A high score indicates that the post-hoc model can easily distinguish between the samples. Intuitively, the better samples are those that achieve lower scores.

**Usefulness**

As we have mentioned, the synthetic data is often generated to be used in downstream inference. It is therefore crucial that the samples from the model have the same predictive characteristics as the underlying data. To measure this we use a *predicitive score*. Similarly to discriminative score, the predictive score is computed using a post-hoc network trained in a supervised fashion. With this score however, we follow the TS/TR evaluation methodology. Specifically, we first train a regression model on the synthetic data where it predicts the next time-step value given the previous values. We then evaluate the trained model on the underlying data. The predictive score is the mean absolute error (MAE) between the predictions and the real data. Here the better samples are those that get low scores as it will be similar to predictive quality of the real samples.

**Diversity**

Finally we want the distribution of the samples from the model to resemble that of the underlying data. To assess this we qualitatively assess *Visualisations* of the t-SNE and PCA of both models. The better samples are those that have closer visual characteristics as the underlying data.

## 4.3 Datasets

Time-series data can be characterized using a range of properties including periodicity, trend, linearity, level of noise and correlation across time and features. In order to assess the performance of the energy models we'll use a range of data-sets with different combinations of these properties. In all the experiments we consider time-series of length 32.

[1]Here the Jaccard score corresponds to the proportion of correct labels.

### 4.3.1 Real Time-Series Data

In the first experiment we assess the performance of the models on a set of real-life time-series data-sets. Often real-life data is messy and contains complex dependencies that are hard to model. Assessing the models on these data-sets allows us to not only to determine their ability with regard to the specific qualities of the data-sets, but their overall practical capacity. To allow for fair comparison we follow [134] and transform the features in pre-processing by scaling each feature to the unit range.

#### Stock

First we consider a dataset that is aperiodic with highly correlated features. The *stock* dataset[2] consists of daily high, low, opening, closing and adjusted closing prices stock prices for the Google ticker collected between 2004 and 2019. We remove the column corresponding to volume as part of prepossessing. This dataset consists of 5 features and 3685 datapoints. Figure 4.1 shows a sample from the data-set.

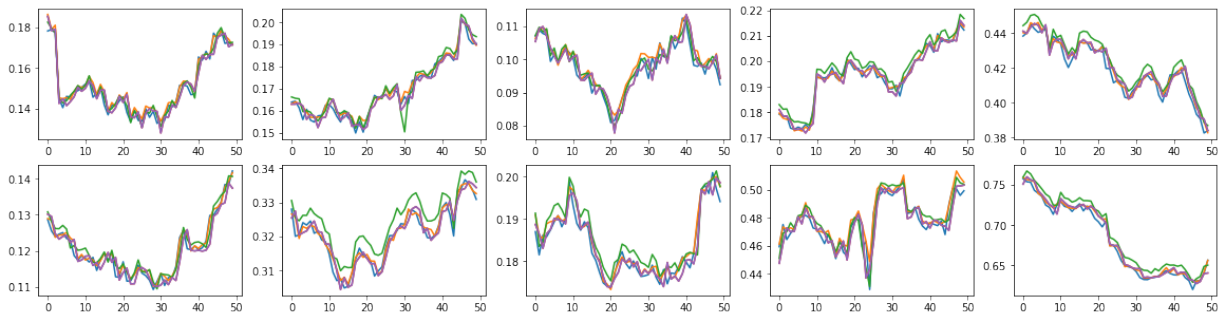


Figure 4.1: A sample from the stock data-set

#### Energy

Next we consider a high-dimensional data-set that is noisy periodic with a lot of correlated features. The energy dataset is the UCI Appliances Energy Prediction data-set [36] includes features such as energy consumption, humidity and temperature. This data-set consists of 28 features and 19735 datapoints. Figure 4.2 shows a sample from the data-set.

[2]Obtained from the implementation of [134]

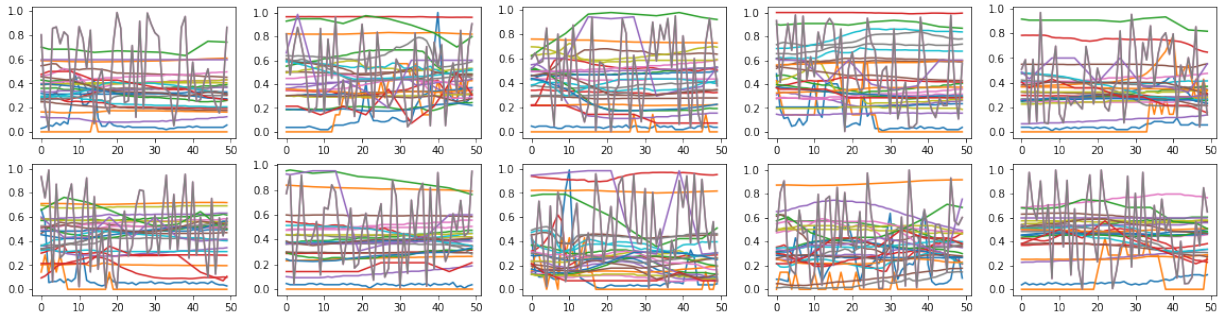


Figure 4.2: A sample from the energy data-set.

### Chickenpox

Finally we consider a high dimensional data-set with a small amount of datapoints. The chickenpox data-set [36] records the weekly amount of chickenpox cases in different areas around Hungary. Naturally, these features are high correlated. As we have discussed one of the core applications for synthetic time-series is augmenting smaller data-sets so that further analysis can be done. Assessing the performance of the models on this data-set will allow us to determine whether they are suited to this crucial goal. This data-set consists of 20 features and 521 datapoints. Figure 4.3 shows a sample from the data-set.

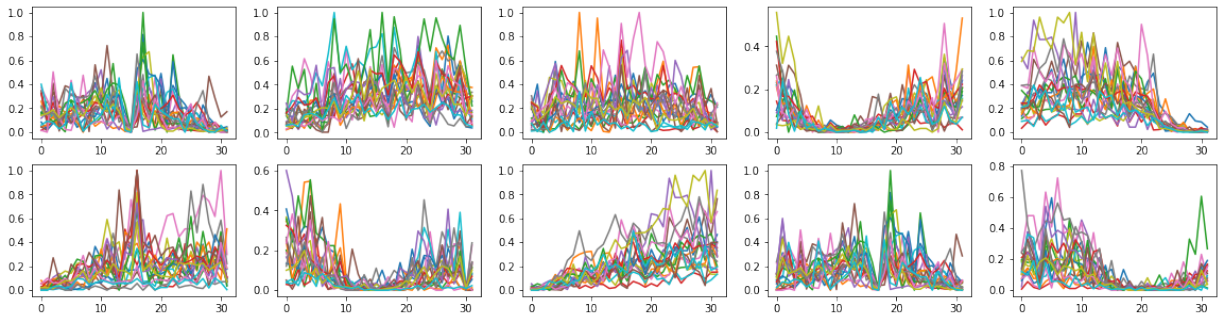


Figure 4.3: A sample from the chickenpox data-set

## 4.3.2 Constructed Time-Series Data

In the second experiment we assess the performance of the models on a series of constructed data-sets. This allows us to control and assess specific characteristics of the time series to determine if the GEBM better attends to these characteristics.

#### Autoregressive Gaussian Models

Here we consider a data-set with differing correlation across the time-steps and across the features. The motivation for this is to access whether the generalized energy based approach better captures the variable temporal dynamics in the time-series. As discussed, this is important as to make the synthetic data useful in downstream tasks. We follow [134] and create the data-set as follows

$$\mathbf{x}_t = \phi \mathbf{x}_{t-1} + \mathbf{n} \qquad \text{where } \mathbf{n} \sim \mathcal{N}(\mathbf{0}, \sigma \mathbf{1} + (1-\sigma)\mathbf{I}) \tag{4.1}$$

where $\phi \in [0, 1]$ controls the correlation across time steps and $\sigma \in [-1, 1]$ controls the correlation across features. This dataset consists of 3 features and 3000 datapoints.

In order to ensure the validity of the results we repeat all the experiments 3 times. We will then report the mean and standard error of the quantitative scores.

## 4.4 Implementation Details

In this section we provide the implementation details for the experiments. Python 3 was used for all the programming elements of this dissertation with PyTorch specifically being used for the deep learning aspects. The code for the GEBM framework, TTS-GAN networks and C-RNN-GAN network are from the following repositories[89][60][24]. Specific details on both the attention and recurrent based architectures can be found in Appendix B and A respectively. The code for this dissertation is available from the following repository[25].

#### Training

For the GEBMs, we train the models by alternating between 5 gradient steps on the energy to 1 on the base. The motivation for this can be seen if we consider the base objective outlined in Section 3.4.2. To learn the base, the generalized likelihood *evaluated at the optimum energy* is used as the loss. To ensure that energy is optimum on a given support it requires numerous gradient steps. Without this the training would be bias and the GEBM would achieve subpar results[5]. Where we differ from the original work on image generation is that we found more optimum results when the learning rate for the energy is significantly less than that of the base. To enforce the Lipschitz assumptions required for

(3.13) we include a gradient penalty as in [113]. Here we use penalty parameter $\lambda = 0.15$ for both models across all the data-sets. We train the model using the Adam optimizer[68] with the learning rates(LR) and parameters $(\beta_1, \beta_2)$ shown in Table 4.1

| Architecture | Dataset | Base LR | Energy LR | Parameters |
|---|---|---|---|---|
| Attention based | Stock | 0.0005 | 0.0001 | (0.5,0.99) |
| | Energy | 0.0005 | 0.0001 | (0.5,0.99) |
| | Chickenpox | 0.0005 | 0.0001 | (0.5,0.99) |
| | Gaus | 0.0001 | 0.00002 | (0.5,0.99) |
| Recurrent based | Stock | 0.0005 | 0.0001 | (0.5,0.99) |
| | Energy | 0.0001 | 0.00002 | (0.5,0.99) |
| | Chickenpox | 0.0001 | 0.00002 | (0.5,0.99) |
| | Gaus | NA | NA | NA |

Table 4.1: LRs and parameters for the GEBMs.

For both GANs we train the models by alternating between single gradient steps for the generator and discriminator. Similarly to the case of GEBMs we train the model using the Adam optimizer with the learning rates(LR) and parameters $(\beta_1, \beta_2)$ shown in Table 4.2. For all the models we decrease the learning rates by a factor of $\gamma = 0.8$ every 20,000

| Architecture | Dataset | Gen LR | Dis LR | Parameters |
|---|---|---|---|---|
| Attention based | Stock | 0.0001 | 0.0001 | (0.5,0.99) |
| | Energy | 0.0001 | 0.0001 | (0.5,0.99) |
| | Chickenpox | 0.0005 | 0.0005 | (0.5,0.99) |
| | Gaus | 0.0001 | 0.0001 | (0.5,0.99) |
| Recurrent based | Stock | 0.0001 | 0.0001 | (0.5,0.99) |
| | Energy | 0.0001 | 0.0001 | (0.5,0.99) |
| | Chickenpox | 0.0001 | 0.0001 | (0.5,0.99) |
| | Gaus | NA | NA | NA |

Table 4.2: LRs and parameters for the GEBMs.

gradient steps of the base/generator. The training performance is monitored using the predictive scores from a held out set of latent variables sampled before the training starts. For the post-hoc networks that compute the discriminator and predictive scores, we use two layer GRUs(2.43), with hidden dimension size equal to the input dimension. For all models we train for 50,000 generator/base gradient steps and use a batch size of 128.

### Sampling

Once we have trained the model we then rescale the energy by a factor of 100 as in the original image generation experiment[5]. We then sample from the posterior latent distribution following the Unadjusted Langevin Algorithm (3.26). Here we perform 1000 iterations with an initial step-size of $\lambda = 10^{-6}$, which halves every 200 iterations. The sampling performance is monitored using the predictive scores of the MCMC samples.

## 4.5 Chapter Summary

In this chapter we provided details on the experiments we conducted to achieve the main aim of the dissertation. First we provided details on the models and data-sets used in the experiments, before providing specific implementation details. Here we explicitly achieve the third objective of the dissertation as well as providing justification for the implementation.

# Chapter 5

# Results

In this chapter we evaluate the performance of the models in the experiments outlined in the previous chapter. First we assess the performance of the models on the real-life data-sets, before assessing the performance of a subset of the models on the generated data-sets. At the end of the chapter we will discuss and critique the findings from both of the experiments.

## 5.1 Real Time Series Data

### 5.1.1 Discriminative and Predictive Scores

The discriminative and predictive scores for each of the models are shown in Table 5.1. Here the results in **bold** are the lowest within each model class. As we have discussed, a lower score is evidence that the model performs better then those with a higher score.

Here we see that T-GEBM consistently achieves the lowest predictive scores across all the attention-based models. As well as this, T-KALE outperforms T-GAN in all the data-sets. For all the models the predictive scores had relatively small standard errors which implies that these differences are significant. Similarly, in all but the stock data-set, T-GEBM achieves the lowest discriminative score. In the stock data-set both T-KALE and T-GAN outperform it with T-KALE achieving the lowest. In both the energy and chickenpox data-set the discriminator scores were similar for T-KALE and T-GAN. Across all the data-sets the discriminator score results had relatively large standard deviations which implies that the results are quite unstable.

| Metric | Method | Stocks | Energy | Chickenpox |
|---|---|---|---|---|
| Predictive Score | T-GEBM | **0.009** ±0.000 | **0.094** ±0.001 | **0.076** ±0.002 |
| | T-KALE | 0.011 ±0.000 | 0.096 ±0.002 | 0.081 ±0.001 |
| | T-GAN | 0.013 ±0.000 | 0.098±0.002 | 0.088 ±0.001 |
| | R-GEBM | **0.011** ±0.001 | **0.092** ±0.002 | 0.112 ±0.003 |
| | R-KALE | 0.013 ±0.000 | 0.094 ±0.001 | 0.114 ±0.004 |
| | R-GAN | 0.019 ±0.001 | 0.094 ±0.002 | **0.106** ±0.003 |
| Discriminative Score | T-GEBM | 0.806 ±0.094 | **0.910** ±0.057 | **0.723** ±0.117 |
| | T-KALE | **0.763** ±0.133 | 0.981 ±0.017 | 0.805 ±0.169 |
| | T-GAN | 0.771 ±0.191 | 1.000 ±0.000 | 0.770 ±0.142 |
| | R-GEBM | **0.870** ±0.076 | **1.000** ±0.000 | 0.821 ±0.106 |
| | R-KALE | 0.881 ±0.087 | **1.000** ±0.000 | **0.784** ±0.109 |
| | R-GAN | 0.900 ±0.051 | **1.000** ±0.000 | 0.801 ±0.118 |

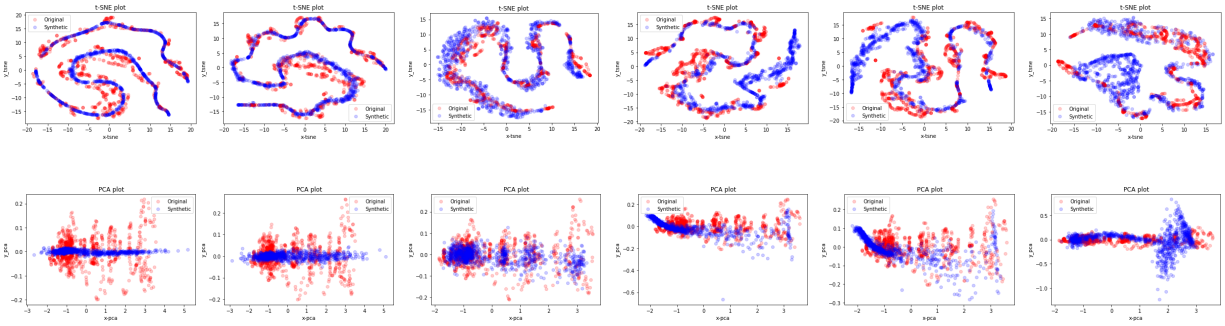
Table 5.1: The discriminative and predictive scores of the models on the real-life data-sets. Here the scores are displayed; mean score over the runs ± standard error of the mean.

For the recurrent-based models, we find that R-GEBM outperforms both R-KALE and R-GAN in terms of the predictive score on the stock and energy data-sets. On the chickenpox data-set R-GAN achieved the lowest score with R-KALE achieving the highest score. Similarly to attention-based case, the predictive scores had relatively small standard deviations. The results for the discriminative scores of the recurrent-based models are mixed. On the stock data-set, R-GEBM outperforms both R-KALE and R-GAN, whereas on the chickenpox data it achieves the highest scores with R-KALE achieving the lowest. On the energy data-set all the scores have the same value.

### 5.1.2 Visualizations

The first row of each of the figures are the t-SNE plots and the second row are the PCA plots. The blue in each of the subplots denotes the samples generated by the models whereas the red denotes samples from the underlying data. As we have discussed, we measure the performance of the model by assessing how well the plot for the models resembles the plot for the underlying data. Larger visualizations can be found in Appendix C

The visualisations of the performance of both the attention and recurrent based models on the stocks data-set can be seen in Figure C.6. We first look at the performance of the attention-based models (a)-(c). Through the t-SNE plots we see that, although all the

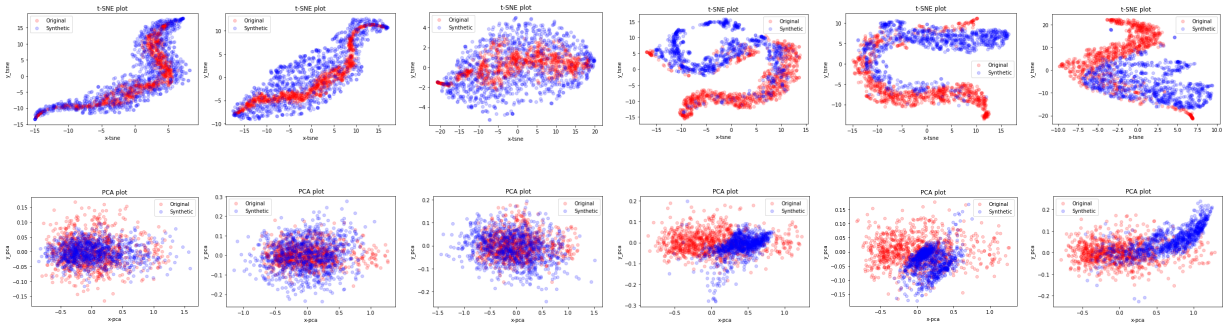


(a) T-GEBM (b) T-KALE (c) T-GAN (d) R-GEBM (e) R-KALE (f) R-GAN

Figure 5.1: Visualisations of the performance of the models on the stock data-set.

models follow the general shape of the underlying data, there are significant differences between them. Specifically, we see that the samples generated from T-GAN (c) are spread out and varied around the underlying shape whereas those from T-KALE (b) are more concentrated along a line that follows the shape. This is a possible indication of over-fitting. Furthermore, we see that the samples from T-GEBM (a) further concentrate on this line. From this it is difficult to assess the performance as both are different from the underlying data in different ways. Through the PCA plot we again see the that the samples from T-KALE have concentrated on a line running through the shape generated by the real data and that the samples from T-GEBM further concentrates on this line. Here, as opposed to the t-SNE plots, it is more evident that the samples from T-GAN best resembles the underlying data and T-GEBM least resembles the underlying data.
For the performance of the recurrent-based models (d)-(f) we find that, in both the t-SNE and PCA plots, although the samples from R-GEBM (d) and R-KALE (e) are again more concentrated then the samples from R-GAN (f), they generally achieve better coverage and resemblance of the underlying data. Furthermore we note that although there is a slight improvement in coverage in the PCA plots from R-KALE to R-GEBM there is no measurable difference in resemblance between the t-SNE plots.

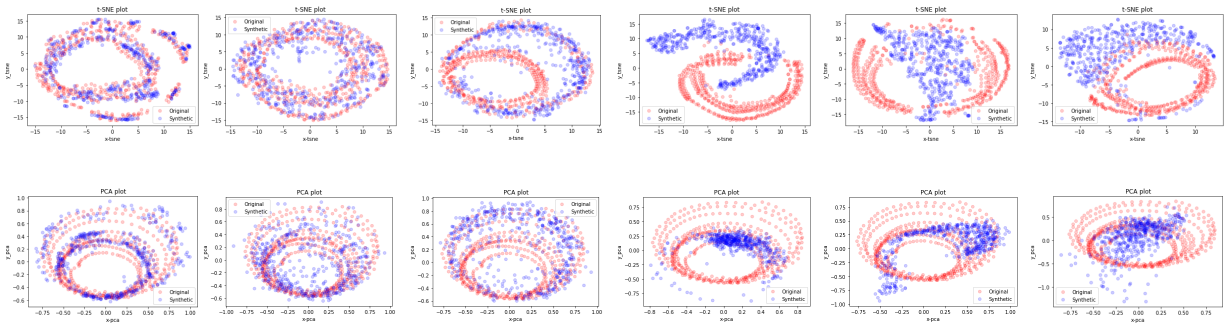
The visualisations of the performance of both the attention and recurrent based models on the energy data-set can be seen in Figure 5.2. We first look at the performance of the attention-based models (a)-(c). Here we see through the t-SNE visualisations that the synthetic data-set generated by T-GEBM (a) better resembles the underlying data, compared to that of T-KALE (b) and T-GAN (c). In evaluating the PCA plots for these models we find that, despite T-GEBM being a bit more concentrated, there is little difference to report.



(a) T-GEBM (b) T-KALE (c) T-GAN (d) R-GEBM (e) R-KALE (f) R-GAN

Figure 5.2: Visualisations of the performance of the models on the energy data-set.

For the recurrent-based models (d)-(f), we find that, similarly to the stock data-set, the samples from R-GEBM (d) are more concentrated than those of R-KALE (e) and R-GAN (f). This holds true for both the t-SNE and PCA plots. While it can be argued that the samples from R-GEBM best resemble the underlying data, we find these gains to be minimal.



(a) T-GEBM (b) T-KALE (c) T-GAN (d) R-GEBM (e) R-KALE (f) R-GAN

Figure 5.3: Visualisations of the performance of the models on the chickenpox data-set.

The visualisations of the performance of both the attention and recurrent based models on the chickenpox data-set can be seen in Figure 5.3. Similarly to before, we first look at the performance of the attention-based models (a)-(c). Here we see through both the t-SNE and PCA that, whereas both T-GEBM (a) and T-KALE (b) follow the general cyclical shapes well, T-GEBM is a lot more concentrated in areas than T-KALE. Similarly to the stock data-set, it is difficult to assess the performance as both are different from the underlying data in different ways. Conversely, we see that while T-GAN (c) does well in covering and concentrating to the outer rings of the shape, it largely ignores the smaller inside rings. For the recurrent based models (d)-(f), in both the t-SNE and PCA

visualisation, we see a lack of coverage for all the models implying that they failed to converge to a meaningful solution.

## 5.2 Constructed Time Series Data

Due to the high time cost of training the recurrent models and general instability observed in the results of the real life data-sets, we perform the second experiment on the attention-based models only. Additionally as we found the differences on the visualisations to be insignificant, we only report the predictive and discriminative score.

The discriminative and predictive scores for each of the models are shown in Table 5.2. Here the results in **bold** are the lowest within each model class.

| | Temporal Correlations (fixing $\sigma = 0.8$) | | | Temporal Correlations (fixing $\sigma = 0.8$) | | |
|---|---|---|---|---|---|---|
| Method | $\phi = 0.2$ | $\phi = 0.5$ | $\phi = 0.8$ | $\sigma = 0.2$ | $\sigma = 0.5$ | $\sigma = 0.8$ |
| Predictive score | | | | | | |
| T-GEBM | **1.112** ±.002 | **.874** ±.001 | **.783** ±.001 | **.792** ±.002 | **.784** ±.001 | **.783** ±.001 |
| T-KALE | 1.114 ±.002 | .879 ±.001 | .787 ±.001 | **.792** ±.001 | .792 ±.001 | .787 ±.001 |
| T-GAN | 1.202 ±.003 | .879 ±.002 | .789 ±.002 | .795 ±.002 | .790 ±.001 | .789 ±.002 |
| Discriminative score | | | | | | |
| T-GEBM | .692 ±.082 | **.614** ±.080 | **.607** ±.050 | .680 ±.043 | **.642** ±.032 | **.607** ±.050 |
| T-KALE | **.690** ±.072 | .641 ±.043 | .674 ±.043 | .685 ±.042 | .656 ±.056 | .674 ±.043 |
| T-GAN | .760 ±.091 | .701 ±.056 | .622 ±.059 | **.661**±.067 | .679 ±.047 | .622±.059 |

Table 5.2: The discriminative and predictive scores of the models on the constructed data-sets. Here the scores are displayed; mean score over the runs ± standard error of the mean.

We find that T-GEBM consistently achieves the lowest predictive scores across all but one of the configurations of the data-set, where on this configuration it shares the lowest with T-KALE. Similarly to the real-lie data-sets, the standard error of the scores are very low which implies these results are significant. Additionally, we find that T-GEBM achieves the lowest discriminative scores in all but two configurations. Across the models we see comparable results between T-KALE and T-GAN. Similarly to the real-life data, the standard error of the discriminative scores across all the data-sets are very high which implies instability.

## 5.3 Discussion

In this discussion, we first consider the results from the experiment with the real-life data-sets. Here we found that, with the exception of the recurrent-based models on the chickenpox data-set, the samples from the GEBMs achieved the lowest predictive score compared to the samples from the base of the GEBMs and samples from the GANs. Through the visualisations, we find that this exception is due to models having failed to converge to a meaningful solution, making the comparison of the recurrent-based models on the chickenpox data-set is inconclusive.

Overall the standard error of the predictive scores across the experiment was very low. This is evidence of not only the reliability of the metric but the significance of the results. However, despite its consistency, there is evidence that it has limitations for use assessing the overall quality of the samples. An example of this can be seen in the results for the attention-based models on the stocks data-set where, despite the lowest predictive score, the samples from the GEBM least resemble the underlying data and achieve the highest discriminative score. This is possibly due to the model *over-fitting* the data and the predictive score not quantifying this as worse performance. However, generally we found that a lower predictive score resulted in better coverage in the visualisations and lower discriminative score.
Through the predictive scores, this experiment provides evidence that, compared to GANs of the same architecture, GEBMs generate samples that better attend to the predictive characteristics of the underlying data. Furthermore, it shows that this improvement is not soley due to the KALE critic. These gains can be seen across both attention and recurrent based architectures.

Through the visualisations, we found that the samples from the GEBMs were generally more concentrated then those from the base of the GEBMs or from the GANs. In some cases, such as attention-based models on the stock data-set, this corresponded in worse quality samples, but in the majority of cases this meant the models better resembled the underlying data. Generally, this metric proved difficult to compare similar performances.

Through the discriminative scores, we found mixed results for the performance of the GEBMs. For the attention-based models, the samples from the GEBM achieve the lowest scores in all but the stock data-set but for the recurrent-based model we found that the samples from the GEBM only achieve the lowest score in the stock data-set. As we have discussed, the poor performance of the attention-based GEBM on the stock data-set may

be attributed to the model over-fitting data. Additionally, for the recurrent-based model, it can be argued that the scores in the energy and chickenpox data-sets are inconclusive, as they are either uninformative or the models failed to converge to a meaningful solution.

Using these results we can also assess the performance and limitations of the discriminative score as a metric. Overall we found the standard error of the scores across the experiment to be very high. This implies that the metric is quite unstable. In designing the experiment we found that by increasing the capacity of the discriminator, the results became more stable but on some data-sets it became too powerful and it always correctly discriminated between the samples, leading to non-informative results, such as for the recurrent models on the energy data-set. The current results reflect this trade-off between stablility and informativity. Overall, we feel that with the mean of the score being used, we can still use these scores to assess the performance of the models. We see evidence in their utility in the results for the attention-based models on the stock data-set; here the possibly overfit and least visually similar samples from the GEBM achieves the lowest predictive score but the highest discriminative score. Overall the discriminative scores of the network provided evidence that, compared to the GAN of the same architecture, the attention-based GEBMs generate samples that better resemble the underlying data. Furthermore, it shows that this better resemblance is not soley due to the KALE critic.

In general we found the training of the recurrent-based networks to be a lot more difficult then the attention-based. This is possibly due to the sensitivity on the parameters. We found this issue to be present in both the training of GEBMs and GANs. A downside of the GEBM approach, for both attention and recuurent architectures, is the training time. As the GEBM was trained 5 gradient steps to 1 between the energy and the base it took significantly more time than the GAN approach. This, coupled with the slow training of the RNNs due to PyTorch implementation difficulties and the time constraints of this dissertation, resulted limited attempts of different parameters for the recurrent models.

Overall, through all the metrics, we find that by incorporating the energy into the attention-based model the sample quality increases. However, the same cannot be said for the recurrent-based models as the their was not sufficient results to form a conclusion

In the constructed data experiment we found that, in all configurations of the autoregressive gaussian, the samples from the GEBMs achieved the lowest predictive score compared to the samples from the base of the GEBMs and samples from the GANs. Additionally

we found that, in over half of the configurations the samples from the GEBMs achieved the lowest discriminative score. These results provide further evidence of the increase in sample quality by incorporating the energy.

## 5.4 Chapter summary

In this chapter we provided details on the results of the experiments of this dissertation. First, we objectively reviewed the results for both experiments before summarising and critiquing the findings. We found that, while there is evidence that attention-based GEBM outperforms its respective GAN, there is inconclusive evidence for the recurrent-based GEBM. In this chapter we have therefore achieved the fourth objective but failed the achieve the fifth objective.

# Chapter 6

# Conclusion

In this thesis we have implemented and evaluated two generalized energy based models for time-series generation, namely T-GEBM, an attention-based GEBM and R-GEBM, a recurrent-based GEBM. We assessed the performance of the models using a range of real-life and generated time-series data-sets and found that, while the results for R-GEBM was mixed due to implementation difficulties, the time-series generated by T-GEBM outperformed those generated by the equivalent GAN, with the same generator and discriminator architecture.

The main contribution of this dissertation is the demonstration of the utility of GEBMs in the time-series domain. Within this; through experiments on multiple real-life data-sets we showed that this approach can be utilized in a range of applications. Furthermore we demonstrated that, compared to GANS, not only do the time-series generated from GEBMs better resemble the underlying data but they better capture the predictive characteristics as well. This is particular important if the data is to be used in downstream inference. Finally, validating the empirical results in the original work on image generation, we showed through comparison with samples from the base of the GEBM only, that this performance gain can be attributed to the incorporation of the energy in the sampling procedure.

Despite these contributions there were several limitations in the report. The biggest issue we found was the performance of the metrics. As well as being fairly unstable, we found evidence that the metrics used in the experiments can be conflicting. As we have seen, this is a recurring theme in the literature for time-series generation. Moving forward with this particular research we hope to experiment with the use of alternative metrics such as MMD or FID.

A further limitation of the report is that is that only fixed length time-series were used. This limits it's use in certain real life applications. A future direction for this work could be to extend the framework to handle time-series of variable length. One way to achieve this is through investigating how the KALE critic can be better applied to sequential data. Another potential avenue of research is extending the the sampling procedure to a more auto-regressive approach, whereby the energy of time-series at a specific time is conditioned on the previous time steps.

In many applications the generated time-series data, as well as being indistinguishable from the real data, must contain privacy guarantees such that it is impossible to link it back to the individual samples. This is particularly important in medical fields. For this reason, in future works we hope to develop a generalized energy framework with privacy-preserving mechanisms such as differential privacy(DP)[130]. As well, we hope to start utilizing metrics that explicitly measure the privacy preserved. Through this we will be better able to assess the utility of these models in the sensitive fields.

Although it wasn't the purpose of this dissertation to achieve state-of-the-art results, we have demonstrated that by adopting this generalized energy approach we can possibly outperform the respective state-of-the-art GANs in time-series generation. Often these state-of-the-art models utilize extra losses and networks to explicitly encourage the generator to attend to the unique features of the time series. In future works we wish experiment with extending the equivalent additions to the GEBM framework to see if the same performance increase is achieved.

# Bibliography


[1] Attention definition. https://www.britannica.com/science/attention. Accessed: 2022-08-16.

[2] MS Windows NT delhi weather data. https://www.kaggle.com/datasets/mahirkukreja/delhi-weather-data. Accessed: 2022-09-01.

[3] S. M. Ali and S. D. Silvey. A general class of coefficients of divergence of one distribution from another. *Journal of the Royal Statistical Society. Series B (Methodological)*, 28(1):131–142, 1966.

[4] Shun-ichi Amari and Hiroshi Nagaoka. *Methods of Information Geometry*, volume 191. 01 2000.

[5] Michael Arbel, Liang Zhou, and Arthur Gretton. Generalized energy based models, 2020.

[6] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan, 2017.

[7] Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. January 2015. 3rd International Conference on Learning Representations, ICLR 2015 ; Conference date: 07-05-2015 Through 09-05-2015.

[8] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2014.

[9] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling, 2018.

[10] David Berthelot, Thomas Schumm, and Luke Metz. Began: Boundary equilibrium generative adversarial networks, 2017.

[11] Mikołaj Bińkowski, Danica J. Sutherland, Michael Arbel, and Arthur Gretton. Demystifying mmd gans, 2018.

[12] Ali Borji. Pros and cons of gan evaluation measures, 2018.

[13] Ali Borji. Pros and cons of gan evaluation measures: New developments, 2021.

[14] John P. Boyd. *Chebyshev and Fourier Spectral Methods*. Dover Books on Mathematics. Dover Publications, Mineola, NY, second edition, 2001.

[15] Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders, 2015.

[16] Roberto Cahuantzi, Xinye Chen, and Stefan Güttel. A comparison of lstm and gru networks for learning symbolic sequences, 2021.

[17] Ovidiu Calin. *Deep Learning Architectures: A Mathematical Approach*. Springer Publishing Company, Incorporated, 1st edition, 2020.

[18] Lawrence Cayton. Algorithms for manifold learning. 2005.

[19] Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T. Freeman. Maskgit: Masked generative image transformer, 2022.

[20] Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1691–1703. PMLR, 13–18 Jul 2020.

[21] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches, 2014.

[22] Yo Joong Choe, Jaehyeok Shin, and Neil Spencer. Probabilistic interpretations of recurrent neural networks. 2017.

[23] Jan Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. Attention-based models for speech recognition, 2015.

[24] cjbayron. c-rnn-gan.pytorch. https://github.com/cjbayron/c-rnn-gan.pytorch, 2020.

[25] ConnorWatts. Timegebm. https://github.com/ConnorWatts/TimeGEBM, 2022.

[26] I. CSISZAR. Information-type measures of difference of probability distributions and indirect observation. *Studia Scientiarum Mathematicarum Hungarica*, 2:229–318, 1967.

[27] Pierre Del Moral, Arnaud Doucet, and Ajay Jasra. Sequential monte carlo samplers. *Journal of the Royal Statistical Society Series B*, 68:411–436, 02 2006.

[28] Anne Marie Delaney, Eoin Brophy, and Tomas E. Ward. Synthesis of realistic ecg using generative adversarial networks, 2019.

[29] Emily Denton, Soumith Chintala, Arthur Szlam, and Rob Fergus. Deep generative image models using a laplacian pyramid of adversarial networks, 2015.

[30] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018.

[31] Peter J. Diggle and Richard J. Gratton. Monte carlo methods of inference for implicit statistical models. *Journal of the Royal Statistical Society. Series B (Methodological)*, 46(2):193–227, 1984.

[32] Chris Donahue, Julian McAuley, and Miller Puckette. Adversarial audio synthesis, 2018.

[33] Hao Dong, Simiao Yu, Chao Wu, and Yike Guo. Semantic image synthesis via adversarial learning, 2017.

[34] Alexey Dosovitskiy and Thomas Brox. Generating images with perceptual similarity metrics based on deep networks. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[35] Yilun Du and Igor Mordatch. Implicit generation and generalization in energy-based models, 2019.

[36] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[37] D. Edwards. On the kantorovich–rubinstein theorem. *International Journal of Biological Macromolecules - INT J BIOL MACROMOL*, 29, 12 2011.

[38] Cristóbal Esteban, Stephanie L. Hyland, and Gunnar Rätsch. Real-valued (medical) time series generation with recurrent conditional gans, 2017.

[39] William Fedus, Ian Goodfellow, and Andrew M. Dai. MaskGAN: Better text generation via filling in the . *In International Conference on Learning Representations*, 2018.

[40] Weilong Fu, Ali Hirsa, and Jörg Osterrieder. Simulating financial time series using attention, 2022.

[41] Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. Made: Masked autoencoder for distribution estimation. 2015.

[42] Felix Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. *Neural computation*, 12:2451–71, 10 2000.

[43] Pierre Glaser, Michael Arbel, and Arthur Gretton. Kale flow: A relaxed kl gradient flow for probabilities with disjoint support, 2021.

[44] Sebastian Goldt, Marc Mezard, Florent Krzakala, and Lenka Zdeborova. Modeling the influence of data structure on learning in neural networks: The hidden manifold model. *Physical Review X*, 10, 12 2020.

[45] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. Book in preparation for MIT Press.

[46] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[47] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans, 2017.

[48] Martin B. Haugh. A tutorial on markov chain monte-carlo and bayesian modeling. *Computer Science Educator: Courses*, 2021.

[49] Debapriya Hazra and Yung-Cheol Byun. Synsiggan: Generative adversarial networks for synthetic biomedical signal generation. *Biology*, 9(12):441, Dec 2020.

[50] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners, 2021.

[51] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[52] Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002.

[53] Geoffrey E Hinton and Sam Roweis. Stochastic neighbor embedding. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems*, volume 15. MIT Press, 2002.

[54] Cheng BK Hobson, A. A comparison of the shannon and kullback information measures, 1973.

[55] Sepp Hochreiter. Untersuchungen zu dynamischen neuronalen netzen. 1991.

[56] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, nov 1997.

[57] Chris Holmes and Stephen Walker. Assigning a value to a power likelihood in a general bayesian model, 2017.

[58] Jonathan H. Huggins, Trevor Campbell, Mikołaj Kasprzak, and Tamara Broderick. Practical bounds on the error of bayesian posterior approximations: A nonasymptotic approach, 2018.

[59] Aapo Hyvärinen. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(24):695–709, 2005.

[60] imics lab. tts-gan. https://github.com/imics-lab/tts-gan, 2022.

[61] J. L. W. V. Jensen. Sur les fonctions convexes et les inégalités entre les valeurs moyennes. *Acta Mathematica*, 30(none):175 – 193, 1906.

[62] Nikolay Jetchev, Urs Bergmann, and Roland Vollgraf. Texture synthesis with spatial generative adversarial networks, 2016.

[63] OT Johnson. *Information theory and the central limit theorem*. Imperial College Press, 2004. Other identifier: ISBN-13: 9781860944734.

[64] Alexia Jolicoeur-Martineau, Rémi Piché-Taillefer, Rémi Tachet des Combes, and Ioannis Mitliagkas. Adversarial score matching and improved sampling for image generation, 2020.

[65] D. S. Jones. Elementary information theory. 1979.

[66] L. V. Kantorovich. Mathematical methods of organizing and planning production. *Management Science*, 6(4):366–422, jul 1960.

[67] Yoon Kim, Carl Denton, Luong Hoang, and Alexander M. Rush. Structured attention networks, 2017.

[68] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[69] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2013.

[70] Peter Kloeden and Eckhard Platen. *The Numerical Solution of Stochastic Differential Equations*, volume 23. 01 2011.

[71] Soheil Kolouri, Kimia Nadjahi, Umut Simsekli, Roland Badeau, and Gustavo K. Rohde. Generalized sliced wasserstein distances, 2019.

[72] Mark A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, 37(2):233–243, 1991.

[73] S. Kullback and R. A. Leibler. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79 – 86, 1951.

[74] Karol Kurach, Mario Lucic, Xiaohua Zhai, Marcin Michalski, and Sylvain Gelly. A large-scale study on regularization and normalization in gans, 2018.

[75] L. D. (Lev Davidovich) Landau. *Statistical physics / by L.D. Landau and E.M. Lifshitz ; translated from the Russian by J.R. Sykes and M.J. Kearsley.* Their Course of theoretical physics ; v. 9, pt. 2. Pergamon Press, Oxford ;, 3d ed. / by e.m. lifshitz and l.p. pitaevskiĭ. edition, 1980.

[76] Yann LeCun, Sumit Chopra, Raia Hadsell, Fu Jie Huang, and et al. A tutorial on energy-based learning. In *PREDICTING STRUCTURED DATA*. MIT Press, 2006.

[77] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network, 2016.

[78] Xiaomin Li, Vangelis Metsis, Huangyingrui Wang, and Anne Hee Hiong Ngu. Tts-gan: A transformer-based time-series generative adversarial network, 2022.

[79] Zengyi Li, Yubei Chen, and Friedrich T. Sommer. Learning energy-based models in high-dimensional spaces with multi-scale denoising score matching, 2019.

[80] Zinan Lin, Vyas Sekar, and Giulia Fanti. On the privacy properties of gan-generated samples, 2022.

[81] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks, 2017.

[82] Shuang Liu, Olivier Bousquet, and Kamalika Chaudhuri. Approximation and convergence properties of generative adversarial learning, 2017.

[83] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[84] Pauline Luc, Camille Couprie, Soumith Chintala, and Jakob Verbeek. Semantic segmentation using adversarial networks. 2016.

[85] Liqian Ma, Xu Jia, Qianru Sun, Bernt Schiele, Tinne Tuytelaars, and Luc Van Gool. Pose guided person image generation, 2017.

[86] Malik Magdon-Ismail and Amir Atiya. Neural networks for density estimation. In M. Kearns, S. Solla, and D. Cohn, editors, *Advances in Neural Information Processing Systems*, volume 11. MIT Press, 1998.

[87] Michael Maschler, Eilon Solan, and Shmuel Zamir. *Game Theory*. Cambridge University Press, 2013.

[88] Hermann Mayer, Faustino Gomez, Daan Wierstra, Istvan Nagy, Alois Knoll, and Jurgen Schmidhuber. A system for robotic heart surgery that learns to tie knots using recurrent neural networks. volume 22, pages 543 – 548, 11 2006.

[89] MichaelArbel. Generalizedebm. https://github.com/MichaelArbel/GeneralizedEBM, 2021.

[90] Tomas Mikolov and Geoffrey Zweig. Context dependent recurrent neural network language model. In *2012 IEEE Spoken Language Technology Workshop (SLT)*, pages 234–239, 2012.

[91] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets, 2014.

[92] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks, 2018.

[93] Volodymyr Mnih, Nicolas Heess, Alex Graves, and Koray Kavukcuoglu. Recurrent models of visual attention, 2014.

[94] Olof Mogren. C-rnn-gan: Continuous recurrent neural networks with adversarial training. *ArXiv*, abs/1611.09904, 2016.

[95] Shakir Mohamed and Balaji Lakshminarayanan. Learning in implicit generative models, 2016.

[96] Michael Mozer. A focused backpropagation algorithm for temporal pattern recognition. *Complex Systems*, 3, 01 1995.

[97] Allan H. Murphy. The finley affair: A signal event in the history of forecast verification. *Weather and Forecasting*, 11(1):3 – 20, 1996.

[98] Alfred Müller. Integral probability metrics and their generating classes of functions. *Advances in Applied Probability*, 29(2):429–443, 1997.

[99] Mikio Namiki. *Stochastic Quantization*. Lecture Notes in Physics Monographs.

[100] 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, nov 2010.

[101] Otton Martin Nikodým. Sur une généralisation des intégrales de m. j. radon. *Fundamenta Mathematicae*, 15:131–179.

[102] Aaron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. Conditional image generation with pixelcnn decoders, 2016.

[103] Bo Pang, Tian Han, Erik Nijkamp, Song-Chun Zhu, and Ying Nian Wu. Learning latent space energy-based prior model, 2020.

[104] George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation, 2017.

[105] Geondo Park, Chihye Han, Wonjun Yoon, and Daeshik Kim. Mhsan: Multi-head self-attention network for visual semantic embedding, 2020.

[106] Jeha Paul, Bohlke-Schneider Michael, Mercado Pedro, Kapoor Shubham, Singh Nirwan Rajbir, Flunkert Valentin, Gasthaus Jan, and Januschowski Tim. Psa-gan: Progressive self attention gans for synthetic time series, 2021.

[107] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[108] Herbert Robbins and Sutton Monro. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 22(3):400 – 407, 1951.

[109] Christian Robert and George Casella. A short history of markov chain monte carlo: Subjective recollections from incomplete data. *Statistical Science*, 26(1), feb 2011.

[110] R. Tyrrell Rockafellar. *Convex analysis*. Princeton Mathematical Series. Princeton University Press, Princeton, N. J., 1970.

[111] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[112] Kady Sako, Berthine Nyunga, and Paulo Rodrigues. Neural networks for financial time series forecasting. *Entropy*, 24:657, 05 2022.

[113] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans, 2016.

[114] Padmanaba Srinivasan and William J. Knottenbelt. Time-series transformer generative adversarial networks, 2022.

[115] Bharath K. Sriperumbudur, Kenji Fukumizu, Arthur Gretton, Bernhard Schölkopf, and Gert R. G. Lanckriet. On integral probability metrics, -divergences and binary classification, 2009.

[116] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision, 2015.

[117] Shuntaro Takahashi, Yu Chen, and Kumiko Tanaka-Ishii. Modeling financial time-series with generative adversarial networks. *Physica A: Statistical Mechanics and its Applications*, 527:121261, 2019.

[118] Louis Thiry, Michael Arbel, Eugene Belilovsky, and Edouard Oyallon. The unreasonable effectiveness of patches in deep convolutional kernels methods. 2021.

[119] Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schoelkopf. Wasserstein auto-encoders, 2017.

[120] Dustin Tran, Rajesh Ranganath, and David M. Blei. The variational gaussian process, 2015.

[121] Yuta Tsuboi, H. Kashima, S. Hido, Steffen Bickel, and M. Sugiyama. Direct density ratio estimation for large-scale covariate shift adaptation. *Journal of Information Processing*, 17:138–155, 01 2009.

[122] Benigno Uria, Marc-Alexandre Côté, Karol Gregor, Iain Murray, and Hugo Larochelle. Neural autoregressive distribution estimation, 2016.

[123] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.

[124] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[125] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics, 2016.

[126] Qing Wang, S.R. Kulkarni, and S. Verdu. Divergence estimation of continuous distributions based on data-dependent partitions. *IEEE Transactions on Information Theory*, 51(9):3064–3074, 2005.

[127] Zhengwei Wang, Qi She, and Tomás E. Ward. Generative adversarial networks in computer vision: A survey and taxonomy. *ACM Comput. Surv.*, 54(2), feb 2021.

[128] Max Welling and Yee Whye Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, page 681–688, Madison, WI, USA, 2011. Omnipress.

[129] Magnus Wiese, Robert Knobloch, Ralf Korn, and Peter Kretschmer. Quant gans: deep generation of financial time series. *Quantitative Finance*, 20:1–22, 04 2020.

[130] Liyang Xie, Kaixiang Lin, Shu Wang, Fei Wang, and Jiayu Zhou. Differentially private generative adversarial network, 2018.

[131] Tijin Yan, Hongwei Zhang, Tong Zhou, Yufeng Zhan, and Yuanqing Xia. Scoregrad: Multivariate probabilistic time series forecasting with continuous energy-based generative models, 2021.

[132] Wilson Yan, Yunzhi Zhang, Pieter Abbeel, and Aravind Srinivas. Videogpt: Video generation using vq-vae and transformers, 2021.

[133] Chao Yang, Xin Lu, Zhe Lin, Eli Shechtman, Oliver Wang, and Hao Li. High-resolution image inpainting using multi-scale neural patch synthesis, 2016.

[134] Jinsung Yoon, Daniel Jarrett, and Mihaela Schaar. Time-series generative adversarial networks. 12 2019.

[135] Mengyue Zha, SiuTim Wong, Mengqi Liu, Tong Zhang, and Kani Chen. Time series generation with masked autoencoder, 2022.

[136] Pengchuan Zhang, Qiang Liu, Dengyong Zhou, Tao Xu, and Xiaodong He. On the discrimination-generalization tradeoff in gans, 2017.

[137] Xuhong Zhang, Toby C. Cornish, Lin Yang, Tellen D. Bennett, Debashis Ghosh, and Fuyong Xing. Generative adversarial domain adaptation for nucleus quantification in images of tissue immunohistochemically stained for ki-67. *JCO Clinical Cancer Informatics*, (4):666–679, 2020. PMID: 32730116.

[138] Zixing Zhang, Jing Han, Kun Qian, Christoph Janott, Yanan Guo, and Björn Schuller. Snore-gans: Improving automatic snore sound classification with synthesized data. *IEEE Journal of Biomedical and Health Informatics*, 24:300–310, 01 2020.

[139] Fei Zhu, Ye Fei, Yuchen Fu, Quan Liu, and Bairong Shen. Electrocardiogram generation with a bidirectional lstm-cnn generative adversarial network. *Scientific Reports*, 9, 05 2019.

# Appendix A

# C-RNN-GAN

In this appendix we aim to provide specific implementation and architecture details on C-RNN-GAN[94]. To that end, we will first look the loss function, before looking at the generator architecture and discriminator architecture presented original work.

## A.1 Loss

For a sequence length m, the authors proposed the following loss functions for the generator G and discriminator D;

$$L_G = \frac{1}{m}\sum_{i=1}^{m}\log(1 - D(G(z^i))) \tag{A.1}$$

$$L_D = \frac{1}{m}\sum_{i=1}^{m}[-\log D(x^i)\log(1 - D(G(z^i)))] \tag{A.2}$$

where $z^i$ is a sequence of k dimensional uniform random vectors, and $x^i$ is a sample time-series from the training data.

## A.2 Generator

The generator consists of an unidirectionsl LSTM network with depth 2 and 350 hidden units for each LSTM cell. The input to the cell at each time-step is a concatenation of the input random vector z(i) to the generator and the output of the previous time-step cell. The output of each cell in G are then fed into a fully connected layer which is then passed

through a sigmoid activation function.

## A.3 Discriminator

Similarly to the generator, the discriminator consists of an LSTM network with depth 2 and 350 hidden units for each LSTM cell. Where it differs from the generator is that it's bidirectional. Therefore the input to the cell at each time-step is a output of the generator at that time-step as well as the output for the previous and subsequent cells. The output of each cell in D are then fed into a fully connected layer which is then passed through a sigmoid activation cell. All the outputs are then average to reach a global decision for the sequence.

# Appendix B

# TTS-GAN

In this appendix we aim to provide specific implementation and architecture details on TTS-GAN[78]. To that end, we will first look the loss function, before looking at the generator architecture and discriminator architecture presented original work.

## B.1 Loss

In this work the authors propose the following losses for the network

$$L_{real} = \frac{1}{n}\sum_{i}^{n}(D(x^i) - 1)^2 \tag{B.1}$$

$$L_{fake} = \frac{1}{n}\sum_{i}^{n}(D(G(z^i)))^2 \tag{B.2}$$

$$L_D = L_{real} + L_{fake} \tag{B.3}$$

$$L_G = \frac{1}{n}\sum_{i}^{n}(D(G(z^i)) - 1)^2 \tag{B.4}$$

where n is the size of the mini-batch, $\{x^i\}$ is the set of real datapoints and $\{z^i\}$ is a set of samples from an N dimensional uniform distribution.

## B.2 Generator

The generator first maps each latent variable to an M dimensional embedding. These embeddings are then divided into patches and repeatedly input into a transformer encoder

block. Each transformer block consists first of a multi-head attention block[124] and then a feed forward layer. Following this, the output is then transformed into a sequence the same size as the original data through a 2D convolution with kernel size (1,1).

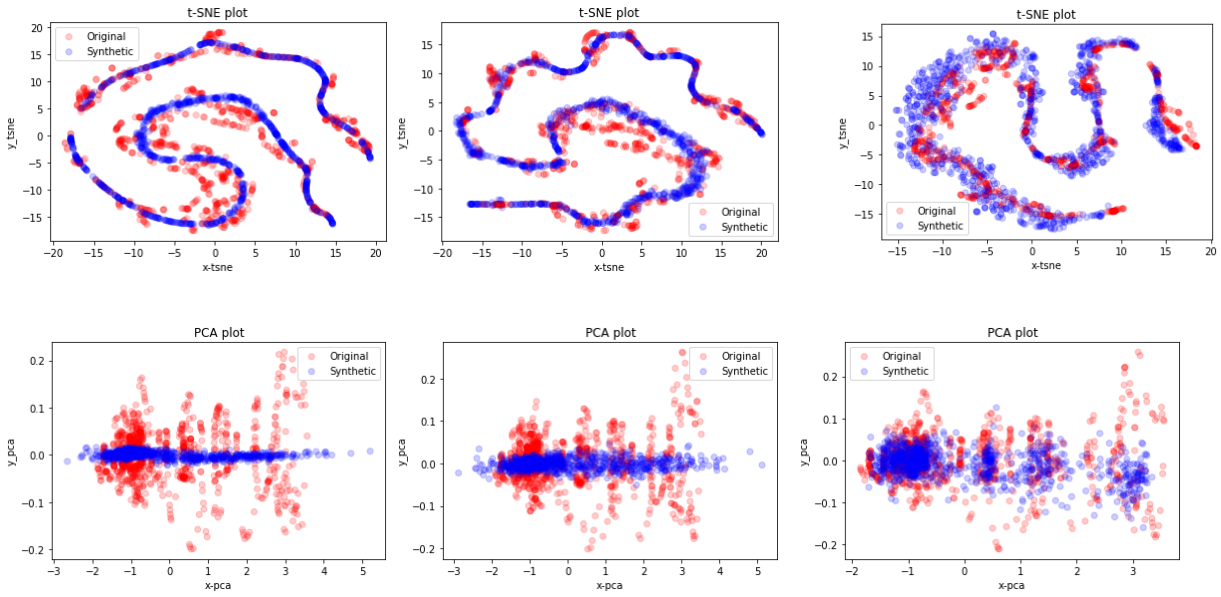
## B.3 Discriminator

Similarly, the discriminator then maps the data in the original data space to an M dimensional embedding. These embeddings are then divided into patches and repeatedly input into a transformer encoder block. Following this, the output is then passed through a single classification head, consisting of a fully connected layer with two out features, corresponding to real and fake.

# Appendix C

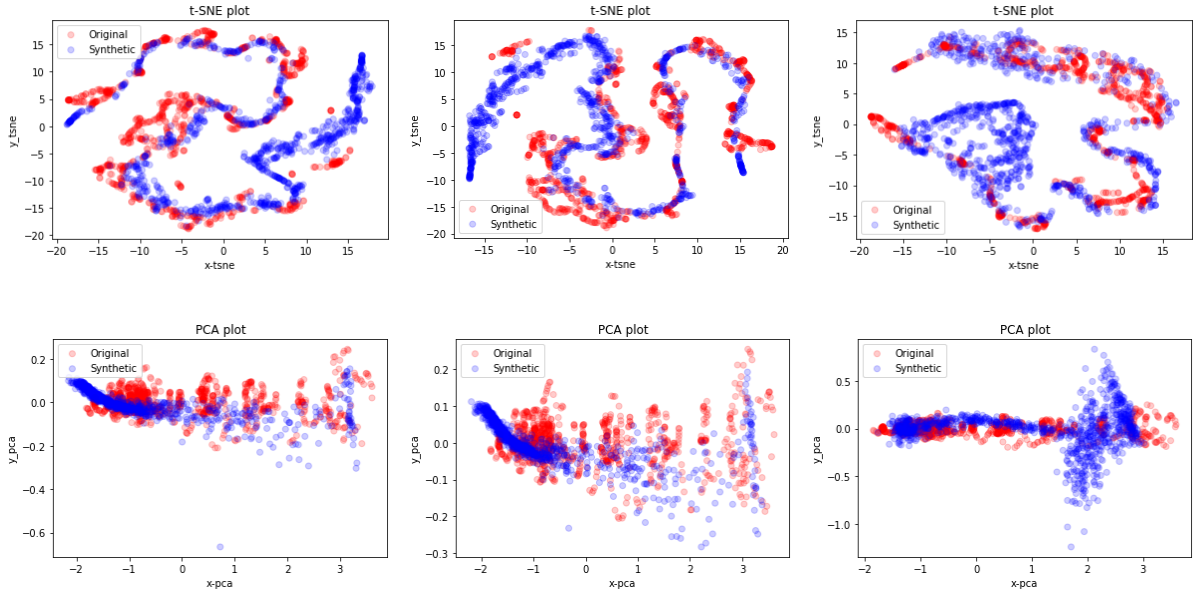
# Visualisations

Here we provide larger visualisations to accompany the report.



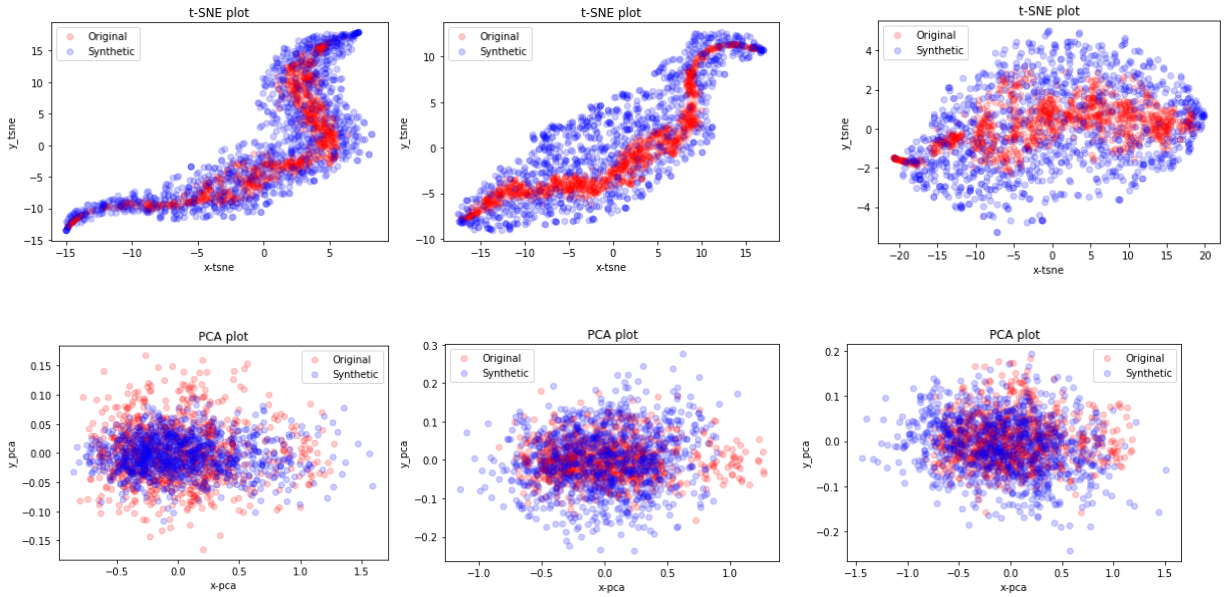

(a) T-GEBM (b) T-KALE (c) T-GAN

Figure C.1: Visualisations of the performance of the attention-based models on the stock data-set.



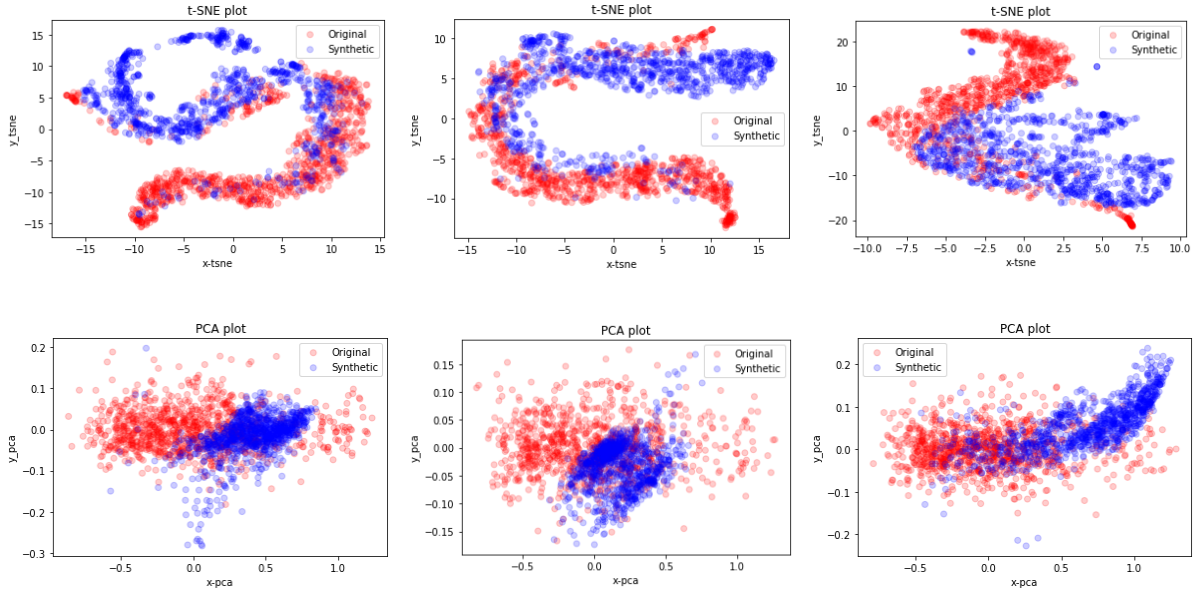

(a) R-GEBM (b) R-KALE (c) R-GAN

Figure C.2: Visualisations of the performance of the recurrent-based models on the stock data-set.



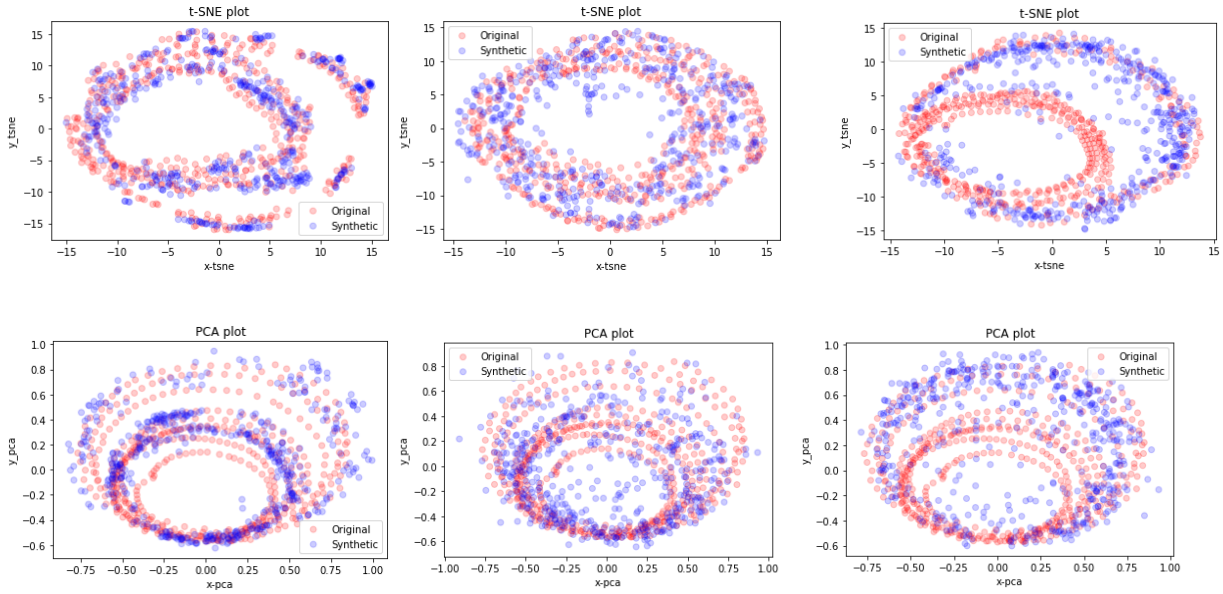

(a) T-GEBM (b) T-KALE (c) T-GAN

Figure C.3: Visualisations of the performance of the attention-based models on the energy data-set.



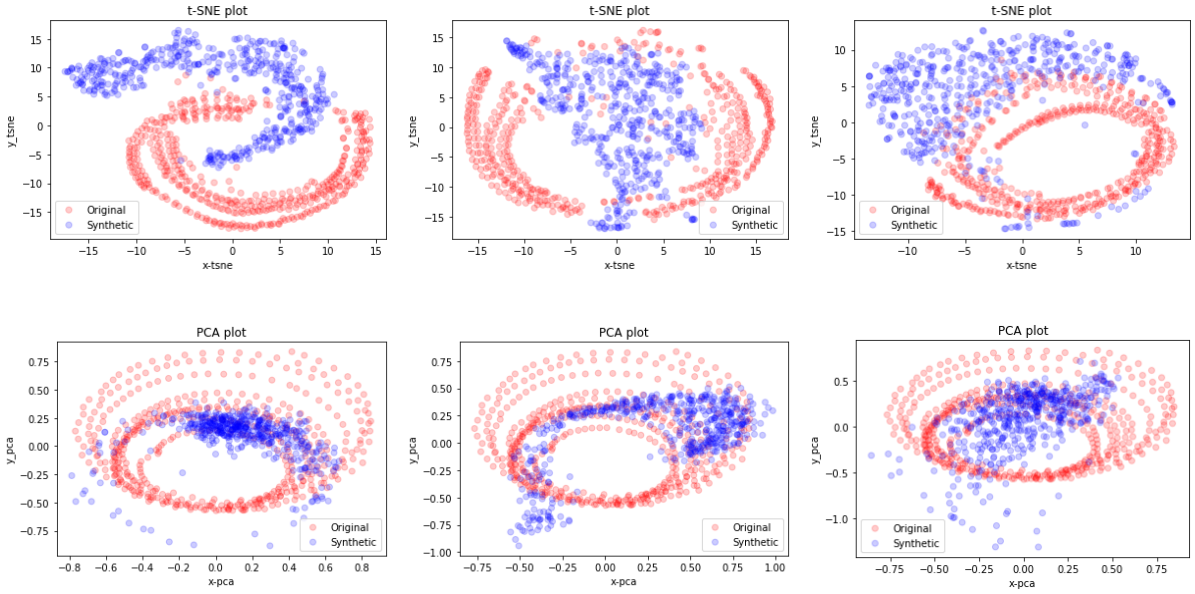

(a) R-GEBM (b) R-KALE (c) R-GAN

Figure C.4: Visualisations of the performance of the recurrent-based models on the energy data-set.




(a) T-GEBM (b) T-KALE (c) T-GAN

Figure C.5: Visualisations of the performance of the attention-based models on the chickenpox data-set.




(a) R-GEBM (b) R-KALE (c) R-GAN

Figure C.6: Visualisations of the performance of the recurrent-based models on the chciken-pox data-set.