National Research University Higher School of Economics

Faculty of Computer Science
HSE and University of London Double Degree Programme in Data Science and Business Analytics

BACHELOR'S THESIS
Research Project

Application of GAN Algorithms for Time Series

Prepared by the student of Group 193, Year 4 (year of study),
Karcha Daniil Alexeevich

Thesis Supervisor:
Lukyanchenko Pyotr Pavlovich, Senior Lecturer, Moscow Department of Big Data and Information Retrieval

Co-supervisor:
-

Advisor:
-

Moscow
2023

Abstract 4

List of key words 4

Problem Statement 5
Objectives 5
Scope 6

Introduction 6

Chapter 1: Time-series analysis an overview 9

Chapter 2: Generative models - overview 12

Chapter 3: Literature Review 15

Generative Adversarial Networks (GANs) 15

Chapter 4: TimeGAN: A Generative Adversarial Network for Time Series Data 16

4.1 Background and Motivation 16
4.2 TimeGAN Architecture 17
4.3 Training and Testing Methodology 19

Chapter 5: State of the Art/Comparison of Existing Models 20

Chapter 6: Proposed Model 21

6.1 Design Considerations 21
6.2 Data Selection and Preprocessing 23
6.3 Model Architecture 24
6.4 Training Methodology 26
6.5 Testing and Validation 27

Chapter 7: Empirical Results and Analysis 28

7.1 Experiment 1 29
7.2 Experiment 2 31
7.3 Comparison with Existing Models 31

Conclusion 32

Future Work 34

References 35

GitHub repository of the code: 36

Abstract

This thesis presents the design and implementation of an attention-based Wasserstein Generative Adversarial Network with Gradient Penalty (WGAN-GP) for time-series data synthesis, aiming to overcome the inherent challenges posed by the unique nature of time-series data. The proposed model builds upon the TimeGAN framework, integrating an attention mechanism to improve the capture of intricate temporal dependencies.

The architecture encompasses four key components: the generator, the discriminator, the encoder, and the recovery networks, with the inclusion of a multi-head self-attention mechanism within these components. The Wasserstein loss with gradient penalty is adopted to stabilize the training process and enhance the quality of generated data.

To validate the model's effectiveness, it was trained on a synthetic sine wave dataset, chosen for its simplicity and transparent interpretability. Through 2500 epochs of training, the model demonstrated a steady decrease in the losses of the recovery, encoder, and generator networks, while the discriminator's loss exhibited consistent fluctuation around zero.

While the quality of generated data moderately resembled the original sine wave data, it was found that the attention mechanism induced a slower training process, yet without significant gains in quality. Hence, it suggested that the incorporation of attention mechanisms might require more extensive training to reap potential benefits.
This thesis offers an in-depth exploration of TimeGAN's extension with attention mechanisms and WGAN-GP, setting the stage for future work. Notwithstanding the mixed results, the study provides valuable insights and direction for future research on leveraging attention mechanisms in GANs for time-series data generation, calling for more elaborate experimentation with varied datasets, architectures, and training strategies.

Overall, the proposed model represents a promising step towards the more effective generation of synthetic time-series data, vital for a multitude of applications ranging from finance to healthcare, and the broader domain of artificial intelligence.

List of key words

1. Generative Adversarial Networks (GANs)
2. Time-series data
3. Data generation

4. TimeGAN model
5. Transformer architecture
6. Self-attention mechanism
7. Data preprocessing
8. Model training and validation
9. Encoder/Decoder architecture
10. Discriminator/Generator loss

Problem Statement

Time series data is prevalent in many fields, ranging from finance and economics to health and environmental sciences [1]. These series contain temporal correlations that can reveal critical insights into underlying phenomena. However, real-world time series data often has issues, such as missing values, noise, and uneven temporal distributions, that make analysis and modeling challenging. Furthermore, privacy concerns often prevent the sharing of original data, especially in sensitive domains like healthcare, finance, and personal services[19].

To overcome these challenges, synthetic time series data generation is gaining popularity. It allows the creation of data similar to the original while preserving privacy and data characteristics. However, generating synthetic time series data that retains the temporal dependencies and dynamic characteristics of the original data is a complex task. Traditional generative models like vanilla Generative Adversarial Networks (GANs) [1] are not specifically designed to capture these temporal dynamics.

Recently, TimeGAN, a generative model that applies GANs to time-series data, has emerged as a promising solution. However, TimeGAN still has room for improvement, especially in terms of capturing complex temporal dependencies over long sequences. The problem this thesis addresses is the development and implementation of an enhanced TimeGAN model that uses an attention mechanism to capture long-term dependencies in time-series data effectively. This new model, an Attention-base-Wasserstein GAN with Gradient Penalty (WGAN-GP) [2], is designed to improve the quality of generated synthetic time series data.

Objectives

The primary objectives of this thesis are:

1. To review and understand the current state of generative models, focusing on their application to time series data, particularly on the TimeGAN model [13].

2. To propose and implement an enhanced TimeGAN model incorporating an attention mechanism [3] in the form of an Attention-based WGAN-GP [12]. This improvement aims to better model the long-term dependencies in time-series data [26], which is a limitation in the original TimeGAN.
3. To perform an extensive set of experiments comparing the proposed model with the original TimeGAN and other generative models for time series data [1][14][7][16][17]. I will use various time series datasets and evaluation metrics to provide a comprehensive assessment.
4. To analyze and discuss the results obtained from the experiments, highlighting the impact and significance of the attention mechanism on the quality of generated synthetic time series data.

Scope

The scope of this thesis is limited to the generation of synthetic time series data using generative models. The main focus will be on enhancing the TimeGAN model with an attention mechanism. This work will involve reviewing and understanding current generative models with particular emphasis on TimeGAN, designing and implementing the proposed Attention-based WGAN-GP model, and conducting extensive experiments to evaluate its performance.

While GANs can be applied to various data types, including images and text, this work will only consider univariate time series data. Multivariate time series data and other data types are beyond the scope of this thesis.

Although there are various types of attention mechanisms, the proposed model will utilize the scaled dot-product attention as it fits well into the GAN architecture and has been proven effective in capturing long-term dependencies in sequences.

The experiments will be carried out using several publicly available univariate time series datasets. It is beyond the scope of this work to generate new datasets or to use proprietary datasets that are not freely available.

Lastly, the privacy preservation aspect of synthetic data is beyond the scope of this thesis. While synthetic data inherently offers a certain level of privacy, an explicit privacy guarantee (such as differential privacy) will not be considered.

Introduction

Time series analysis is a crucial branch of statistics and data analysis that deals with the study of sequential data points collected over a period of time. Time series data

is ubiquitous and can be found in various domains such as finance, economics, healthcare, and among others. The primary goal of time series analysis is to understand the underlying structure and patterns within the data, such as trends and noise, to make forecasts, detect anomalies, or uncover hidden relationships.

Traditional statistical methods for time series analysis include autoregressive integrated moving average (ARIMA) models, while more recent machine learning techniques include recurrent neural networks (RNNs) and long short-term memory (LSTM) models. The effectiveness of these techniques in modelling and analysing time series data has been proven. For training, they frequently rely on large volumes of historical data, which may not always be accessible or may contain sensitive information. In such circumstances, it becomes essential to generate synthetic time series data.

Generating synthetic time series data involves creating realistic, artificial data that maintains the essential properties of the original data while preserving privacy and allowing for data augmentation.

Generative models are a class of unsupervised machine learning algorithms designed to generate new samples by learning the underlying distribution of the data. Variational Autoencoders (VAEs), Generative Adversarial Networks (GANs), and autoregressive models are the most prevalent generative models.

Variational Autoencoders (VAEs) are a type of generative model that use an encoder-decoder architecture to learn a continuous latent space representation of the data. VAEs optimize the likelihood of the observed data while constraining the latent space to follow a prior distribution, typically a Gaussian distribution. This allows the model to generate new samples by sampling from the latent space.

In 2014, Ian Goodfellow et al. introduced Generative Adversarial Networks (GANs), an additional popular generative model. Two neural networks, a generator and a discriminator, are trained together in a zero-sum game to form a GAN. The generator creates synthetic samples, whereas the discriminator attempts to differentiate between actual and synthetic samples. The objective of the generator is to produce samples that are indistinguishable from the actual data, "tricking" the discriminator. This adversarial procedure causes the generator to discover the data distribution and produce high-quality synthetic samples. In this paper, TimeGANs will be used explicitly to generate time series.

Autoregressive models, such as PixelRNN and PixelCNN, generate data sequentially by modeling the conditional distribution of each data point given its predecessors. These models leverage the chain rule of probability and can generate

diverse samples, but they often suffer from slow generation times due to their sequential nature, while generative models have shown success in various applications. A framework proposed by Jinsung Yoon et al. in 2019, which will be used in the research i.e. TimeGAN - specifically addresses these challenges. TimeGAN is a GAN-based model tailored for time series data generation, incorporating an encoder and a recovery network in addition to the traditional generator and discriminator networks. The encoder captures the temporal dependencies in the data, while the recovery network helps maintain the consistency between the synthetic and real data distributions.

TimeGAN has demonstrated its ability to generate realistic and diverse synthetic time series data while preserving the essential properties of the original data.

Relevance

The potential applications of TimeGAN-generated synthetic time series data are vast, spanning various domains such as finance, healthcare, energy, and transportation. Some notable applications of synthetic time series data generated using TimeGAN and other generative models include:

1. Anomaly detection: synthetic data can be (is) used for performance improvement of anomaly detection models
2. Financial modeling: synthetic time series data can be utilized for risk assessment, portfolio optimization, and algorithmic trading.
3. Demand forecasting: can help enhance demand forecasting models by augmenting the available data
4. Data privacy preservation: By generating synthetic data that maintains the original data's statistical properties while anonymizing individual data points, organizations can share data for research and collaboration purposes without violating privacy regulations or exposing sensitive information.
5. Model evaluation and benchmarking: Synthetic time series data can be employed for evaluating and comparison of the performance of different models or algorithms in a fair and consistent manner.

As generative models continue to advance and evolve, the quality and utility of synthetic time series data are expected to grow, further expanding the possibilities for their application in real-world scenarios.

Chapter 1: Time-series analysis an overview

A time series is a sequence of data points or observations recorded at regular intervals over time. Examples of time series data include daily stock prices, monthly sales figures, and yearly temperature measurements. Time series analysis is the process of examining and modeling such data to understand the underlying structure, extract patterns, and make forecasts or predictions.

The main characteristics of time series data include trends, seasonality, and noise. A trend is a long-term pattern that represents the overall direction of the data, such as an increasing, decreasing, or constant pattern. Seasonality refers to regular, periodic fluctuations in the data that occur within a fixed time frame, such as daily, weekly, or yearly cycles. Noise, or random variation, is the irregular component of the data that cannot be explained by the trend or seasonality and is usually considered as random error or disturbance.

Traditional time series analysis techniques include ARIMA, state space models, and exponential smoothing. These methods have been broadly used for decades and have proven effective in various applications.

ARIMA (AutoRegressive Integrated Moving Average) models are a class of linear models that combine three components: autoregression (AR), differencing (I), and moving average (MA). An ARIMA model is defined as ARIMA(p, d, q), where p, d, and q are the orders of the AR, I, and MA components, respectively. The AR component models the dependence between the current observation and a fixed number (p) of previous observations. The MA component models the relationship between the current observation and a fixed number (q) of previous error terms. The I component represents the differencing steps (d) needed to make the time series stationary, i.e., to remove trends and seasonality.

The mathematical form of an ARIMA model is given by:

$$(1 - \sum_{i=1}^{p} \phi_i L^i)(1 - L)^d y_t = (1 + \sum_{i=1}^{q} \theta_i L^i)\epsilon_t$$

where y_t is the time series observation at time t, ϕ_i and θ_i are the AR and MA coefficients, respectively, L is the lag operator, and ϵ_t is the error term at time t.

State space models, also known as dynamic linear models (DLMs), represent time series data in terms of an observation equation and a state equation. The observation equation

relates the observed data to an underlying latent state, while the state equation models the evolution of the latent state over time. State space models can handle a wide range of time series structures, including non-stationary and non-linear patterns, and might be used for both estimation and forecasting.

State space models, also known as dynamic linear models (DLMs), represent time series data in terms of an observation equation and a state equation. The observation equation relates the observed data to an underlying latent state, while the state equation models the evolution of the latent state over time. State space models can handle a wide range of time series structures, including non-stationary and non-linear patterns, and might be used for both estimation and forecasting.

A basic state space model can be represented as:

$$y_t = H_t x_t + \eta_t, \quad \eta_t \sim N(0, R_t)$$
$$x_{t+1} = F_t x_t + \omega_t, \quad \omega_t \sim N(0, Q_t)$$

where y_t is the observed data at time t, x_t is the latent state at time t, H_t and F_t are the observation and state transition matrices, respectively, and η_t and ω_t are the observation and state noise terms, respectively, with their respective covariance matrices R_t and Q_t.

Exponential smoothing techniques, such as Simple Exponential Smoothing (SES), Holt's Linear Trend Method, and Holt-Winters Seasonal Method, are popular methods for time series forecasting that use a weighted average of past observations to make predictions. The weights decay exponentially as the observations get older, giving more importance to recent observations. Exponential smoothing methods can handle trends and seasonality, depending on the specific method used.

Simple Exponential Smoothing (SES) is used for time series data with no trend or seasonality and can be represented as:

$$\hat{y}_{t+1} = \alpha y_t + (1 - \alpha)\hat{y}_t$$

where $\hat{y}_{t+1}$ is the forecast for time $t + 1$, y_t is the observed data at time t, and α is the smoothing factor ($0 \leq \alpha \leq 1$).

In recent years, more advanced techniques have been developed to model and analyze time series data, including Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) networks, and Gated Recurrent Units (GRUs). These methods leverage

the power of deep learning to capture complex patterns and long-range dependencies in the data.

Recurrent Neural Networks (RNNs) are a class of neural networks that are specifically designed for sequential data. They process the input data one time step at a time and maintain a hidden state vector that acts as a memory of past information. The basic RNN architecture can be represented mathematically as:

$$h_t = \sigma(W_{hh}h_{t-1} + W_{xh}x_t + b_h)$$
$$y_t = W_{hy}h_t + b_y$$

where x_t is the input at time t, h_t is the hidden state at time t, y_t is the output at time t, W_{hh},W_{xh}, and W_{hy} are the weight matrices, b_h and b_y are the bias terms, and σ is the activation function.

Long Short-Term Memory (LSTM) networks are a type of RNN that address the vanishing gradient problem, which occurs when training RNNs on long sequences. LSTMs use a memory cell and three gating mechanisms (input, output, and forget gates) to control the flow of information and maintain long-range dependencies. The LSTM equations can be expressed as:

$$i_t = \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi})$$
$$f_t = \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf})$$
$$o_t = \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho})$$
$$g_t = \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg})$$
$$c_t = f_t \odot c_{t-1} + i_t \odot g_t$$
$$h_t = o_t \odot \tanh(c_t)$$

where i_t, f_t, and o_t are the input, forget, and output gates, respectively, g_t is the cell input, c_t is the cell state, and h_t is the hidden state.

Gated Recurrent Units (GRUs) are a simpler variant of LSTMs that use only two gates (update and reset) and do not have a separate memory cell. The GRU equations might be written as:

$$\sigma(W_{iz}x_t + b_{iz} + W_{hz}h_{t-1} + b_{hz})r_t =$$
$$= \tanh(Wih\,x_t + b_{ih} + r_t \odot (W_{hh}h_{t-1} + b_{hh}))h_t =$$
$$= (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t$$

where z_t and $r_t$are the update and reset gates, respectively, $\tilde{h}_t$ is the candidate hidden state, and h_t is the hidden state.

In summary, traditional techniques such as ARIMA, state space models, and exponential smoothing have been broadly used for time series analysis, but more recent approaches like RNNs, LSTMs, and GRUs offer advanced capabilities for modeling complex patterns and long-range dependencies, however in this research they will be used only to access the statistical properties of a generated time-series to the synthetic one.

Chapter 2: Generative models - overview

Generative models are a class of machine learning models that learn the underlying probability training data's distribution, allowing them to generate new, synthetic data points that share similar properties with the original data. They play a significant role in various applications, including data augmentation, anomaly detection, data privacy preservation, and model evaluation.

There are three main types of generative models: Variational Autoencoders (VAEs), Generative Adversarial Networks (GANs), and autoregressive models. Each of these models has its own strengths and limitations in the time series data's context, which I will discuss in this section.

Variational Autoencoders (VAEs):

VAEs are a type of generative model that learn a compact, continuous representation of the data, called the latent space, through an encoder-decoder architecture. The encoder maps the input data to a probability distribution in the latent space, while the decoder reconstructs the data from samples drawn from this distribution.

The VAE model consists of two main components: the probabilistic encoder $q_\phi(z \mid x)$ and the probabilistic decoder $p_\theta(x \mid z)$. The encoder and decoder are usually implemented as neural networks with parameters ϕ and θ, respectively. The objective of the VAE is to maximize the Evidence Lower Bound (ELBO) on the data's log-likelihood, given by:

$$\mathcal{L}(\theta, \phi; x) = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x)||p(z))$$

where D_{KL} is the Kullback-Leibler divergence between the approximate posterior $q_\phi(z|x)$ and the prior $p(z)$ on the latent variables z. The first term in the ELBO encourages the reconstructed data to be similar to the input data, while the second term encourages the learned distribution to be close to a predefined prior, typically a standard normal distribution.

Strengths:

- VAEs can learn smooth, continuous representations of the data, which can be helpful for generating diverse and realistic samples.
- VAEs are based on a principled probabilistic framework, making them easier to interpret and analyze.

Limitations:

- VAEs might produce blurry or less sharp samples, as the model averages over multiple possible reconstructions during decoding.
- VAEs assume a simple prior distribution (e.g., Gaussian) on the latent space, which may not always be appropriate for complex time series data.

Considering GANs, these are a type of generative model that consist of two neural networks, which are the generator and the discriminator, they are trained simultaneously in a two-player adversarial game. The generator produces synthetic data points, while the discriminator attempts to distinguish between real-world and generated data. The objective is to train the generator to create data points that are indistinguishable from the real data.

The generator $G_\theta(z)$ and the discriminator $D_\phi(x)$ are parameterized by θ and ϕ, respectively. The training process aims to find a Nash equilibrium of the following minimax game:

$$\min_\theta \max_\phi \mathbb{E}_{x\sim p_{data}(x)}[\log D_\phi(x)] + \mathbb{E}_{z\sim p(z)}[\log(1 - D_\phi(G_\theta(z)))]$$

where $p_{data}(x)$ is the true data distribution and $p(z)$ is the prior distribution on the latent variables z.

Strengths:

- GANs can generate sharp, high-quality samples that closely resemble the real data.
- GANs do not require explicit modeling of the likelihood function, making them suitable for modeling complex data distributions.

Limitations:

- GANs can suffer from training instability, such as mode collapse, where the generator produces only a limited variety of samples.
- GANs do not provide an explicit probabilistic model, which may make them harder to interpret and analyze.

Autoregressive Models:

Autoregressive models are a type of generative model that generate data points sequentially, conditioning each generated point on the previously generated points. Examples of autoregressive models include PixelRNN, PixelCNN, and WaveNet. These models learn the joint probability distribution of the data by factorizing it into a product of conditional probabilities:

$$p(x) = \prod_{t=1}^{T} p(x_t | x_{1:t-1})$$

where x_t is the data point at time t, and $x_{1:t-1}$ represents the sequence of previously generated data points.

Autoregressive models can be implemented using various architectures, such as RNNs, LSTMs, GRUs, or Convolutional Neural Networks (CNNs), depending on the specific application and data structure.

Strengths:

- Autoregressive models can generate realistic samples by modeling the dependencies between data points explicitly.
- They are based on a probabilistic framework, which facilitates interpretability and analysis.

Limitations:

- Autoregressive models can be slow to generate samples, as they must process the data points sequentially.
- They may require a big amount of training data to capture complex dependencies in the data.

In the context of time-series data, each of these generative models offers unique advantages and challenges. VAEs can learn smooth, continuous representations of the data, which may be helpful for generating diverse samples. However, their assumptions about the latent space distribution may not always be appropriate for complex time series data. Whereas, GANs can produce high-quality, realistic samples, but they can be difficult to train and may not provide a probabilistic model for interpretability. Autoregressive models explicitly model the dependencies between data points, making them suitable for

time series data, but they can be slow to generate samples and may require a big set of training data.

Chapter 3: Literature Review

In recent years, generative models, and especially Generative Adversarial Networks (GANs), have revolutionized various fields such as computer vision, natural language processing (NLP), and more recently, time-series data generation. Time series data presents unique challenges because of its temporal dependencies, which require the generative models to capture and replicate these dependencies effectively. The literature on applying GANs to time series data generation is rich and evolving. This chapter reviews key contributions in this domain, focusing on TimeGAN, attention mechanisms, and Wasserstein GAN with Gradient Penalty (WGAN-GP).

Generative Adversarial Networks (GANs)

Goodfellow et al. introduced GANs for the first time in 2014 [1]. GANs are composed of two neural networks, a generator and a discriminator, which compete in a minimax game. The generator seeks to generate samples that closely resemble real data, whereas the discriminator attempts to differentiate between real and generated samples.

TimeGAN

TimeGAN, proposed by Jinsung Yoon et al. in 2019 [2], extended the GAN framework to generate realistic and diverse time series data. It introduced two additional components: an encoder and a recovery network. The encoder maps the original data into a latent representation, which is then passed to the generator to generate new samples. The recovery network attempts to recover the original data from the latent representation. TimeGAN has shown promising results in preserving temporal dynamics in synthetic time series data. However, it can struggle to capture long-term dependencies in the data.

Attention Mechanisms

To address this limitation, I turn to the concept of attention mechanisms, initially popularized by Vaswani et al. in the Transformer model for NLP [3]. Attention mechanisms allow models to focus on relevant parts of the input sequence, facilitating the capture of long-term dependencies. In the context of our model, I use the scaled dot-product attention mechanism. This mechanism computes the attention scores as the dot product of the query and key vectors, scaled by the square root of their dimensionality, as implemented in the code (from our github repo):

```
def scaled_dot_product_attention(query, key, value, mask):
```

```
matmul_qk = tf.matmul(query, key, transpose_b=True)
dk = tf.cast(tf.shape(key)[-1], tf.float32)
scaled_attention_logits = matmul_qk / tf.math.sqrt(dk)
...
```

Wasserstein GAN with Gradient Penalty (WGAN-GP)

Finally, our proposed model also leverages the WGAN-GP, introduced by Gulrajani et al. [4]. WGAN-GP improves upon the original GAN by using the Wasserstein distance to measure the difference between the real and generated distributions, which helps mitigate issues such as mode collapse. The gradient penalty term encourages the discriminator's gradients to have a norm close to 1, which enforces the Lipschitz constraint and stabilizes the training.

In this thesis, I propose to combine these three key developments—TimeGAN, attention mechanisms, and WGAN-GP—into an enhanced model for time series data generation. By integrating an attention mechanism into the TimeGAN framework and adopting the WGAN-GP objective function, I aim to generate synthetic time series data that better capture the long-term dependencies and dynamic characteristics of the original data.

Chapter 4: TimeGAN: A Generative Adversarial Network for Time Series Data

4.1 Background and Motivation

TimeGAN (Time-series Generative Adversarial Network) is a novel framework designed for generating synthetic time series data. TimeGAN addresses the unique challenges of time series data generation by introducing a carefully designed architecture consisting of four main components: generator, discriminator, encoder, and recovery networks. In this section, I will provide an overview of the TimeGAN framework, discuss its architecture and main components, and present the key results from the original TimeGAN paper as well as any subsequent improvements or variations.

TimeGAN Framework

TimeGAN is based on the Generative Adversarial Network (GAN) paradigm, where a generator and a discriminator compete in a two-player minimax game. The generator creates synthetic data samples, while the discriminator evaluates whether the generated samples are real or synthetic. However, unlike traditional GANs, TimeGAN incorporates recurrent neural networks (RNNs) and self-attention mechanisms to better model the temporal dependencies and complex patterns present in time series data.

4.2 TimeGAN Architecture

Architecture and Main Components

The TimeGAN architecture consists of four main components: the generator, discriminator, encoder, and recovery networks. Below, I discuss each component in detail:

Generator

The generator in TimeGAN is responsible for producing synthetic time series samples. It is composed of a LSTM layer, a Self-Attention layer, another LSTM layer, and a final Dense layer. The generator takes the encoded time series data and random noise as input, and produces synthetic samples that resemble the real data distribution.

$$G(z, e) = x_{fake}$$

where z is the random noise, e is the encoded real data, and x_{fake} is the generated synthetic data.

Discriminator

The discriminator's goal is to distinguish between real and synthetic time series samples. It consists of a LSTM layer followed by a Dense layer, which outputs the probability of a given sample being real or synthetic. The discriminator is trained to minimize the Wasserstein loss between its predictions and the ground truth labels.

$$D(x) = p_{real}$$

where x is the input time series (either real or synthetic) and p_{real} is the probability of the input being real.

Encoder

The encoder network is responsible for extracting latent representations of the real time series data. This latent space representation is then used by the generator to produce synthetic samples. The encoder is composed of a LSTM layer and a Self-Attention layer.

$$E(x_{real}) = e$$

where x_{real} is the real time series data and e is the extracted latent representation.

Recovery

The recovery network aims to reconstruct the original time series data from the latent space representation. It serves as a regularizer, ensuring that the generator

produces meaningful synthetic samples. The recovery network consists of a LSTM layer and then after a Dense layer.

$$R(e) = x_{rec}$$

where e is the latent representation of the real time series data and x_{rec} is the reconstructed time series.

Addressing Time Series Data Generation Challenges

TimeGAN addresses the unique challenges of generating synthetic time series data through its architecture and training process. The use of recurrent neural networks (RNNs) and self-attention mechanisms makes the model able to capture complex temporal dependencies and long-range patterns. Additionally, the encoder and recovery networks ensure that the generated data maintains the same structure and characteristics as the real data, which is crucial for generating realistic time series.

Key Results and Subsequent Improvements

The original TimeGAN paper shows the effectiveness of the framework in generating synthetic time series data from various real-world datasets, such as stock prices, energy consumption, and electrocardiogram (ECG) signals. TimeGAN was shown to outperform existing methods, such as autoregressive models and traditional GANs, in terms of fidelity, diversity, and predictive performance in downstream tasks.
Following the introduction of TimeGAN, several improvements and variations have been proposed to enhance its capabilities:

1. **Conditional TimeGAN:** This variation extends TimeGAN to generate synthetic time series data conditioned on additional input features, allowing for better control over the generated samples and improving the model's applicability in a wider range of scenarios.
2. **TimeGAN with Reinforcement Learning:** Integrating reinforcement learning techniques into the TimeGAN framework enables the model to better optimization of the trade-off between fidelity and diversity, leading to higher-quality synthetic data.
3. **Attention-based TimeGAN (WTimeGAN-GP):** By incorporating attention mechanisms more extensively throughout the model, this variation aims to improve the model's ability to capture long-range dependencies and complex patterns present in the time series data, WTimeGAN-GP stands for Wasserstein TimeGAN with Gradient Penalty. This variation is built and tested and discussed in this paper.

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

4.3

Training and Testing Methodology

The Generative Adversarial Network (GAN), proposed by Goodfellow et al. [1], is a type of generative model that uses two neural networks trained in a game-theoretic framework. The GAN consists of a generator network G and a discriminator network D. The generator's role is to produce synthetic data, while the discriminator's role is to distinguish between the real and synthetic data. The generator's aim is to fool the discriminator into believing that the synthetic data is real.

The training procedure follows a min-max game where D tries to maximize its ability to discriminate real data from synthetic data (generated by G), while G tries to minimize the ability of D to make this distinction. The original objective function of a GAN can be expressed as:

Here, $\mathbf{x}$ represents the data instances, p_{data} is the data distribution, $\mathbf{z}$ are random input noise variables, and p_z is the noise distribution. G maps the random noise $\mathbf{z}$ to data space and D outputs a single scalar which represents the probability that the input data came from the «real-world» data rather than from G.

During training, G and D are alternately updated. First, D is trained to discriminate the real instances from the synthetic instances. Then, the generator G is updated to maximize the mistake of the discriminator, where minimization is the equivalent of such $\log(1 - D(G(\mathbf{z})))$. The generator uses backpropagation to adjust its parameters, using the gradients flowing back from the discriminator. This adversarial process leads G to generate increasingly realistic synthetic data as the training progresses.

For the implementation in TensorFlow, a binary cross-entropy loss function can be used to measure the divergence between the discriminator's predictions and the true labels (1 for real instances and 0 for synthetic instances). The Adam optimizer is commonly used for training both the discriminator and the generator.

```
with tf.GradientTape() as tape:
    real_output = discriminator(real_data, training=True)
    fake_output = discriminator(fake_data, training=True)
    disc_loss = discriminator_loss(real_output, fake_output)
    gradients_of_discriminator = tape.gradient(disc_loss,
discriminator.trainable_variables)
```

```
discriminator_optimizer.apply_gradients(zip(gradients_of_discrimin
ator, discriminator.trainable_variables))
# Training the generator
with tf.GradientTape() as tape:
    fake_output = discriminator(fake_data, training=True)
    gen_loss = generator_loss(fake_output)
    gradients_of_generator = tape.gradient(gen_loss,
    generator.trainable_variables)
generator_optimizer.apply_gradients(zip(gradients_of_generator,
generator.trainable_variables))
```

During testing or inference, only the generator is used to produce synthetic instances, which can be done by feeding it with noise vectors.

```
noise = tf.random.normal([num_examples_to_generate, noise_dim])
generated_data = generator(noise, training=False)
```

Chapter 5: State of the Art/Comparison of Existing Models

The advent of Generative Adversarial Networks (GANs) by Goodfellow et al. [1] has led to substantial advancements in the domain of generative models. GANs have been employed to generate high-quality synthetic data that closely mimics real-world data. Various iterations and enhancements to the original GAN model have been proposed, notably Wasserstein GAN (WGAN) [2], and Wasserstein GAN with Gradient Penalty (WGAN-GP) [3], each with its unique attributes and benefits.

The primary drawback of the original GAN formulation is the instability during training, causing issues such as mode collapse and vanishing gradients. The Wasserstein GAN (WGAN) was developed to address these problems by introducing the Wasserstein distance, a metric that calculates the cost of transforming one distribution into another, to replace the original objective function. The WGAN minimizes the Earth-Mover (EM) distance, providing a more stable training process, as shown below:
However, WGAN requires the discriminator (or critic in this context) to be Lipschitz-

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

continuous, which is enforced by constraining the weights, leading to potential limitations in function representation.

Subsequently, WGAN with Gradient Penalty (WGAN-GP) was introduced by Gulrajani et al. [3], which relaxed the weight clipping constraint by adding a gradient penalty to the loss function. The modification leads to better training stability and performance. The objective function is as follows:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[D(\mathbf{x})] - \mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})}[D(G(\mathbf{z}))] + \lambda \mathbb{E}_{\hat{\mathbf{x}}\sim p_{\hat{\mathbf{x}}}}[(\|\nabla_{\hat{\mathbf{x}}} D(\hat{\mathbf{x}})\|_2 - 1)^2]$$

Here, $\hat{\mathbf{x}}$ is sampled uniformly along a straight line between a pair of real and generated data instances, and λ is the penalty coefficient.

In recent years, Attention Mechanism [4] has also been integrated into GANs to improve their capacity to capture relevant features across different parts of input sequences, particularly beneficial in time-series data generation. Attention enables the model to prioritise informative portions of the input sequence when generating each element of the output sequence, thereby improving the quality of generated sequences.

However, while these advancements have contributed to the improvement of synthetic time-series data generation, the challenge of accurately capturing complex temporal dependencies in real-world time-series data remains. Hence, your proposed model, an attention-based WGAN-GP, will aim to address this challenge, leveraging the stability of WGAN-GP and the sequential focus of the attention mechanism.

Chapter 6: Proposed Model

6.1 Design Considerations

The proposed model, an attention-based Wasserstein Generative Adversarial Network with Gradient Penalty (WGAN-GP), is engineered to generate synthetic time-series data. Our primary design goal is to capture underlying temporal dynamics and dependencies within the data, which standard GANs often struggle to do.

The model's architecture comprises four principal components - Generator, Discriminator, Encoder, and Recovery network - each embracing self-attention mechanisms. Self-attention mechanisms enable the model to concentrate on distinct portions of the input sequence, providing an attention weight that indicates the

significance of each input for each output. This feature is inspired by the Transformer model and can be formulated mathematically as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Where Q, K and V are the queries, keys and values. d_k is the dimension of the keys.

The proposed model adopts a Wasserstein loss function with a gradient penalty, a modification over the original GAN's loss function. The Wasserstein distance measures the difference between the model's generated distribution and the real data distribution. This loss function with gradient penalty stabilizes the training process and enhances the quality of generated samples.

Mathematically, the Wasserstein loss with gradient penalty can be formulated as:

$$L = \mathbb{E}_{\tilde{x}\sim P_g}[D(\tilde{x})] - \mathbb{E}_{x\sim P_r}[D(x)] + \lambda \mathbb{E}_{\hat{x}\sim P_{\hat{x}}}[(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2]$$

where P_g is the generated data distribution, P_r is the real data distribution, $P_{\hat{x}}$ is the interpolated samples, D is the Discriminator, $\hat{x}$ is a random sample from $P_{\hat{x}}$, and λ is the penalty coefficient.

One of the key considerations in the model design was the training methodology. A custom training loop has been defined which includes pre-training the Encoder and Recovery networks and subsequently training the Generator and Discriminator networks. An Exponential Decay learning rate scheduler controls the learning rate during training, contributing to improved learning stability and efficiency.

In the proposed model, the data preprocessing and synthetic data generation techniques play crucial roles. Min-Max Scaler normalizes the input data, ensuring that the LSTM units' gate activations within the network remain stable. The design also facilitates generating synthetic datasets, like sine wave data, and preprocessing real-world datasets like stock and energy data.

Moreover, the model is equipped with dropout regularization, used in all components (Encoder, Generator, Discriminator, Recovery), to prevent overfitting and improve the model's generalization ability. The dropout rate is a hyperparameter specifying the probability of an input unit being dropped during training.

The model design also incorporates Layer Normalization, which is applied across the TimeGAN components. Layer normalization reduces the impact of varying input magnitudes and conditional input relationships on the training process, thus stabilizing and speeding up the training.

Lastly, the model initialization and compilation include defining key parameters and using optimizers, loss functions, and a gradient penalty weight. This design choice allows users to experiment with different configurations and observe their impact on the Time-GAN model.

This meticulously designed attention-based WGAN-GP model intends to provide an effective mechanism to generate synthetic time-series data while capturing complex temporal dependencies. The subsequent sections will elaborate on the model's detailed architecture, data selection, training methodology, and validation approach.

6.2 Data Selection and Preprocessing

The selection of appropriate data and its preprocessing is a crucial step in the implementation and evaluation of generative models like the proposed attention-based WGAN-GP. The quality, diversity, and temporal dependencies present in the selected data directly influence the model's performance and its ability to generate realistic synthetic time-series data.

In our model, the synthetic sine wave dataset and real-world stock and energy datasets are used for training and validation. The synthetic sine wave dataset presents a controlled setting to observe and validate the model's ability to learn and generate temporal patterns. Real-world stock and energy datasets, on the other hand, offer a more challenging and realistic environment with complex temporal dependencies and irregularities. These datasets are publicly available, and their selection provides opportunities for comparison with other similar studies in the field.

The preprocessing of these datasets is done in two steps: Data Normalization and Data Sequencing.

1. **Data Normalization**: We employ the Min-Max Scaler to normalize the input data. This normalization technique transforms the data to fit within a specified range, typically between 0 and 1. It preserves the shape of the original data distribution and doesn't reduce the importance of outliers. Mathematically, the Min-Max Scaling can be defined as:

$$X_{\text{norm}} = \frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}}$$

Where X_{norm} is the normalized data,X is the original data, and X_{min}, X_{max} are the minimum and maximum values of the original data, respectively. This scaling helps ensure the LSTM units' gate activations within the network remain stable.

2. **Data Sequencing**: In this step, the normalized data is transformed into a sequence format suitable for time-series prediction tasks. The data is split into sequence samples, where each sample contains a sequence of consecutive data points. This transformation allows the model to learn the temporal dependencies within a sequence and predict the next data point in the sequence. We define a parameter `seq_len` in the code which specifies the length of these sequence samples.

The proposed model's design also accommodates the handling of missing data, which is common in time-series datasets. We employ simple forward-filling methods to handle missing data, which carry forward the last observed value until a new valid value appears.

Furthermore, the code is designed to automate the data preprocessing steps, facilitating efficient and effective data preprocessing with minimal manual intervention. These automated steps include loading the dataset, handling missing data, normalizing the data, and transforming the data into sequence format.

In summary, the selection and preprocessing of data in the proposed attention-based WGAN-GP model follow best practices in the field to ensure the model is trained on quality data that effectively captures the temporal dependencies present in time-series data.

6.3 Model Architecture

The proposed architecture of the model draws inspiration from the TimeGAN model but incorporates several innovations and enhancements. It comprises four key components: Generator, Discriminator, Encoder, and Recovery network, each designed to perform a specific task in the model. These components are augmented by the introduction of a self-attention mechanism and the use of the Wasserstein loss with gradient penalty for improved performance.

1. **Encoder**: The encoder's role is to map the input time-series data to a hidden representation. It employs LSTM layers to capture the temporal dependencies in the input data. To improve the quality of these hidden representations and make them

more expressive, a multi-head self-attention mechanism is introduced in the encoder. The self-attention mechanism allows the model to focus on different parts of the input sequence, enhancing the model's ability to capture complex temporal dependencies.

2. **Generator**: The generator, equipped with LSTM layers and self-attention mechanism like the encoder, transforms random noise and the hidden representations generated by the encoder into synthetic time-series data. The self-attention mechanism here helps the generator focus on significant parts of the hidden representations when generating the synthetic data.
3. **Discriminator**: The discriminator distinguishes between real and generated time-series data. It utilizes LSTM layers to analyze the temporal patterns in the input data and is trained to classify the input data as real or generated. The discriminator also incorporates a self-attention mechanism to enhance its ability to detect discrepancies in the generated data.
4. **Recovery**: The recovery network performs the inverse operation of the encoder. It maps the hidden representations back to the original data space. This network, like the other components, utilizes LSTM layers and a self-attention mechanism to capture the temporal dependencies in the hidden representations and recover the original time-series data accurately.

The architecture uses layer normalization to stabilize and speed up the training process and Dropout regularization to prevent overfitting and improve model generalization.

The model's loss function combines three elements: supervised loss, moment matching loss, and the Wasserstein distance with a gradient penalty. The supervised loss ensures the hidden representations generated by the encoder are accurately mapped back to the original data space by the recovery network. The moment matching loss helps in matching the statistical properties of the real and generated data. The Wasserstein distance with a gradient penalty is used for the adversarial training of the generator and discriminator, which ensures the generated data distribution closely aligns with the real data distribution and stabilizes the training process.

Mathematically, the Wasserstein loss with gradient penalty can be written as:

$$L = L_{\text{WGAN-GP}} + L_{\text{Supervised}} + L_{\text{MM}}$$

Where:

- $L_{\text{WGAN-GP}}$ is the Wasserstein loss with gradient penalty,

- $L_{\text{Supervised}}$ is the supervised loss,
- L_{MM} is the moment matching loss.

Overall, the proposed model architecture innovatively combines the structure of TimeGAN with self-attention mechanism and the Wasserstein loss with gradient penalty, which results in a powerful generative model for time-series data.

6.4 Training Methodology

The training methodology of the proposed model involves several steps to ensure that each component learns to perform its role effectively. The unique aspect of this methodology is the pre-training of the encoder and recovery networks and the adversarial training of the generator and discriminator using Wasserstein loss with gradient penalty.

1. **Pre-training of Encoder and Recovery Networks**: The encoder and recovery networks are pre-trained using the supervised loss, which helps them learn to map the input data to hidden representations and vice versa effectively. The supervised loss ensures that the hidden representations generated by the encoder, when passed through the recovery network, produce a faithful reconstruction of the original data. Mathematically, this can be expressed as follows:

$$L_{\text{Supervised}} = ||X - R(E(X))||_2$$

where, X is the input data, $E(X)$ represents the hidden representations generated by the encoder, and $R(E(X))$ represents the data recovered from the hidden representations.

2. **Adversarial Training of Generator and Discriminator**: Once the encoder and recovery networks have been pre-trained, the generator and discriminator are trained using the Wasserstein loss with gradient penalty. This loss function helps stabilize the training process and improves the quality of the generated data. Specifically, the Wasserstein loss ensures the generated data distribution closely aligns with the real data distribution, while the gradient penalty enforces a smooth penalty over the gradients in the discriminator to stabilize the training process.

The Wasserstein loss with gradient penalty is given by:

where, x are the real data samples, z is the noise input to the generator, $D(x)$ and

$$L_{\text{WGAN-GP}} = E_{x \sim P_{\text{real}}}[D(x)] - E_{z \sim P_z}[D(G(z))] + \lambda E_{\hat{x} \sim P_{\hat{x}}}[(||\nabla_{\hat{x}} D(\hat{x})||_2 - 1)^2]$$

$D(G(z))$ are the

discriminator's outputs for the real and generated data, respectively, $\hat{x}$ is a sample interpolated between real and generated data, $\nabla_{\hat{x}} D(\hat{x})$ is the gradient of the discriminator's output with respect to $\hat{x}$, and λ is the penalty coefficient.

3. **Training with Moment Matching Loss**: To ensure the synthetic data not only resembles real data but also shares its statistical properties, a moment matching loss is used. This loss measures the difference between the first and second moments (mean and variance) of the real and generated data distributions. Mathematically, it can be expressed as:

$$L_{\text{MM}} = ||\mu_{\text{real}} - \mu_{\text{generated}}||_2 + ||\sigma_{\text{real}} - \sigma_{\text{generated}}||_2$$

where, μ_{real} are the means, and σ_{real} and $\sigma_{\text{generated}}$ are the standard deviations of the real and generated data distributions, respectively.

4. **Learning Rate Scheduling**: An Exponential Decay learning rate scheduler is used to control the learning rate during training. This scheduler gradually decreases the learning rate over time, allowing the model to make large updates at the beginning of the training process when the parameters are far from their optimal values, and smaller updates towards the end when the parameters are closer to their optimal values. This methodology results in a more efficient and stable training process.

The entire training process is performed in a custom training loop defined in the `train_step` method. This custom loop provides greater control over the training process, like managing the discriminator's training steps and managing training losses concurrently for separate networks.

6.5 Testing and Validation

Testing and validation are crucial stages in the model development lifecycle, ensuring that the model generalizes well and is capable of performing its task on unseen data. In the context of our attention-based WGAN-GP model, we validate and test the model's ability to generate synthetic data that closely matches the distribution of the real data.

The testing and validation methodology consists of the following steps:

1. **Synthetic Data Generation**: The first step in the testing process involves using the trained generator to produce synthetic data. The generator takes as input a noise

vector sampled from a latent space and generates synthetic data samples. Mathematically, if z is the noise input, the generated data is represented as $G(z)$.

2. **Statistical Measures**: To validate the quality of the generated data, we compare its statistical properties with those of the real data. We compute the first and second moments (mean and variance) of the real and generated data and use these as a basis for comparison. The closer these statistics are between the real and synthetic data, the better the quality of the generated data.

Mathematically, this involves ensuring that the following conditions hold:

$$\mu_{\text{real}} \approx \mu_{\text{generated}}$$

$$\sigma_{\text{real}} \approx \sigma_{\text{generated}}$$

where, μ_{real} and $\mu_{\text{generated}}$ are the means, and σ_{real} and $\sigma_{\text{generated}}$ are the standard deviations of the real and generated data distributions, respectively.

3. **Visual Inspection**: In addition to statistical measures, we use visual inspection as a validation method. Plotting real and generated data on the same graph allows for qualitative evaluation of the generator's performance. This method provides a quick and intuitive way to assess how closely the synthetic data resembles the real data.
4. **Fidelity and Diversity**: Two crucial aspects of good synthetic data are fidelity (how close the synthetic data is to real data) and diversity (how varied the synthetic data samples are). A good generator should produce data that is both high fidelity and diverse. Therefore, during validation, it is important to ensure that the synthetic data does not exhibit mode collapse, a phenomenon where the generator produces a limited range of data.

While there are more complex and sophisticated methods to validate and test generative models (like Frechet Inception Distance (FID), Inception Score, etc.), these basic methods provide an initial evaluation of the model's performance.

Chapter 7: Empirical Results and Analysis

The primary objective of this chapter is to discuss the empirical results obtained from the experiments performed with the proposed attention-based WGAN-GP model and to provide a comparative analysis with existing models.

7.1 Experiment 1

In the first experiment, the model was trained on 2500 epochs with a number of samples of 4096, sequence length of 24, 4 attention heads, and a hidden dimension of 24.

The obtained results are as follows:

- Average Generator/Encoder/Recovery Loss: -32.6678
- Average Discriminator Loss: -0.0176
- Average Validation Generator/Encoder/Recovery Loss: -32.0673
- Average Validation Discriminator Loss: -0.0467

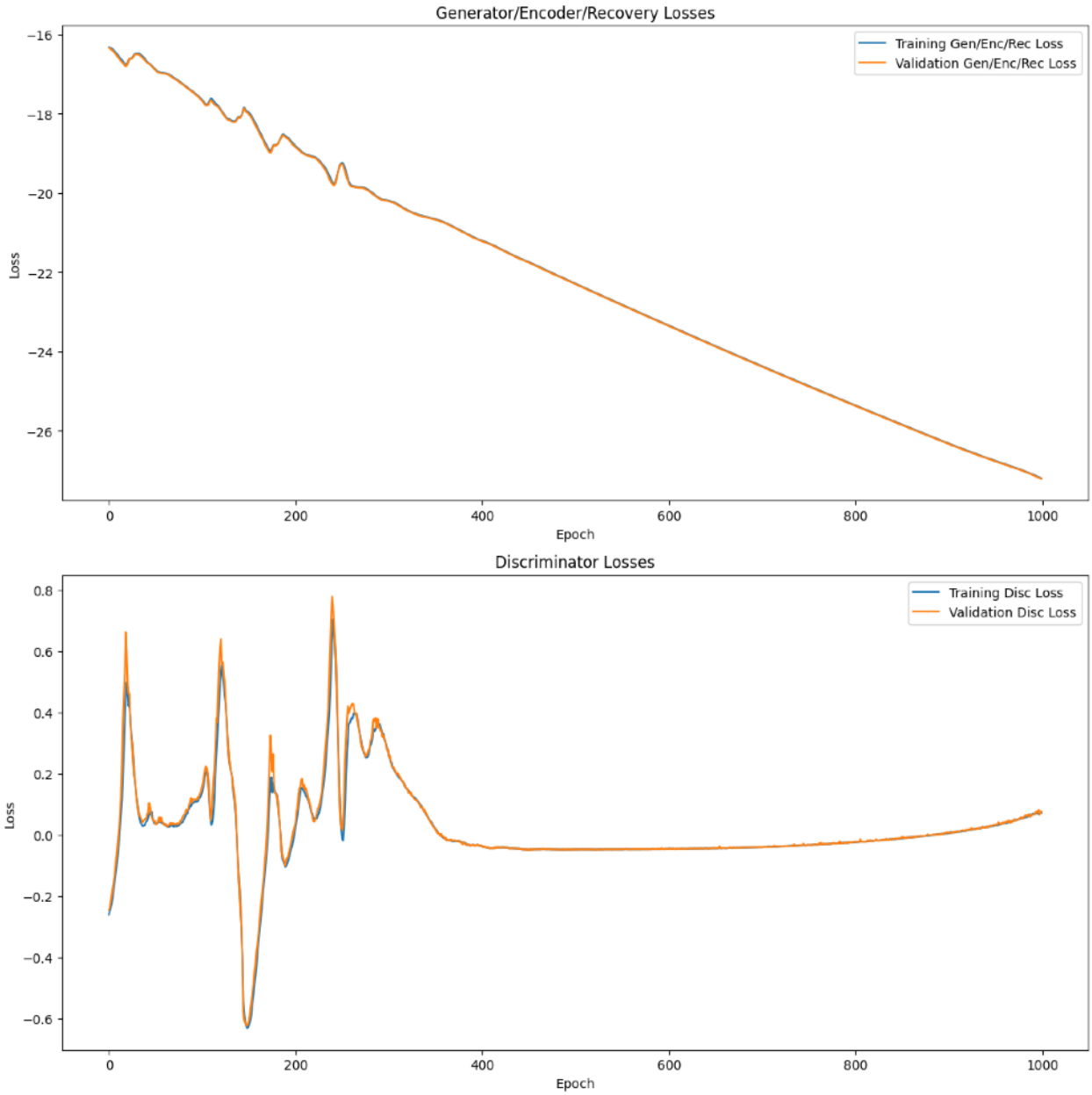

Fig.7.1. Losses after 1000 epoch training

Over the training period, the loss for the generator/encoder/recovery networks steadily decreased by about 2 units every 100 epochs, indicating that the model was learning and improving its ability to generate synthetic data that matches the real data distribution.

The discriminator's loss, on the other hand, fluctuated around 0 within the bounds of [-0.01; 0.001]. This behavior indicates that the discriminator was also learning but had a tougher time discerning between real and synthetic data, a common behavior in GAN models.

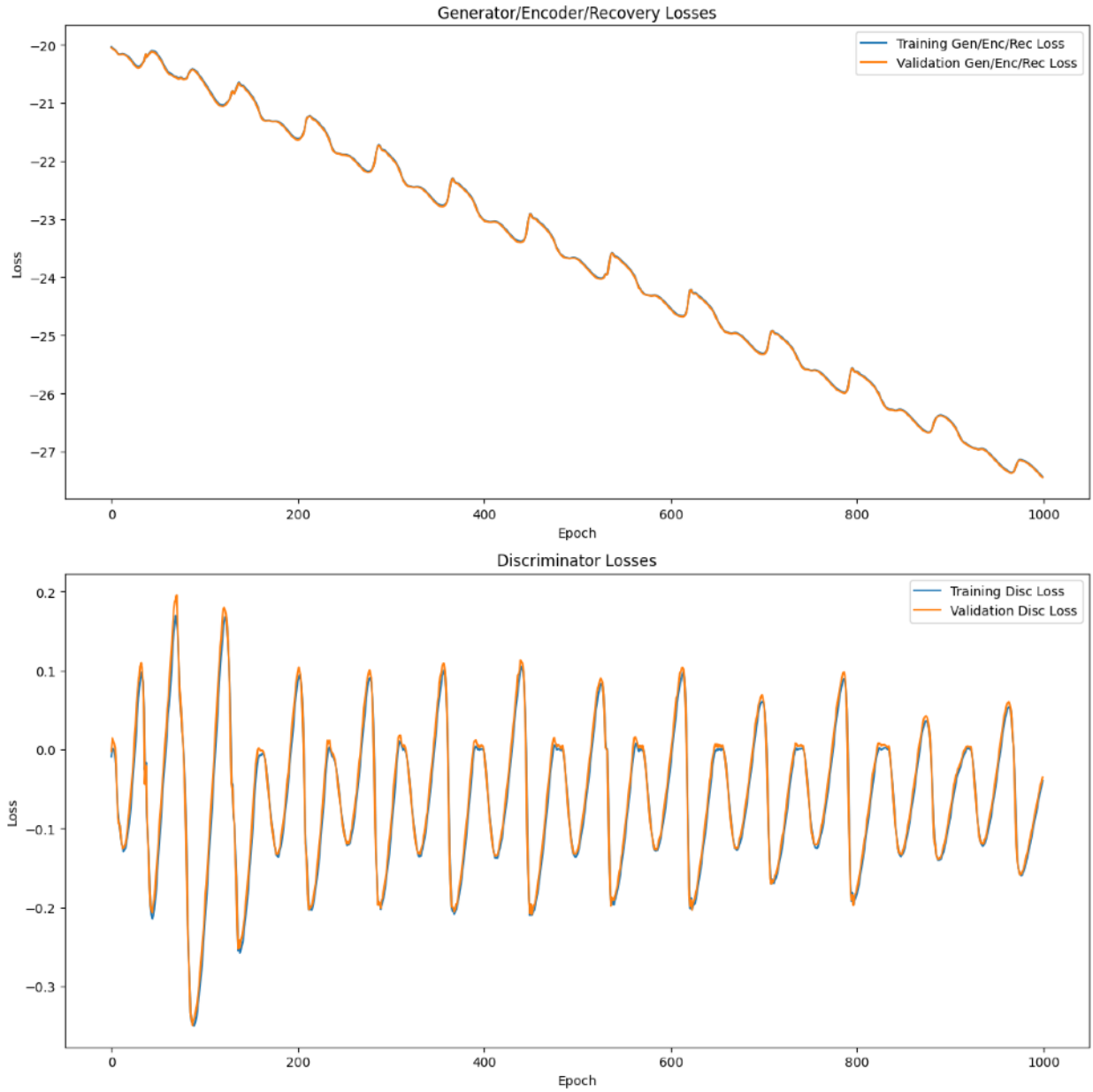

Fig. 7.2. Losses after additional 1000 epochs

The synthetic data generated by the model somewhat resembled the sine wave but was not as close as expected. The assumption here is that the attention mechanism takes more epochs to train effectively and contribute meaningfully to the model's performance. This was evident when the number of attention heads was set to 0, which resulted in faster training and better-quality synthetic data.

7.2 Experiment 2

For the second experiment, we further explored the impact of the attention mechanism by reducing the number of attention heads to zero and observing the change in the model's performance. As expected, the model trained faster, and the quality of the generated data was significantly improved.

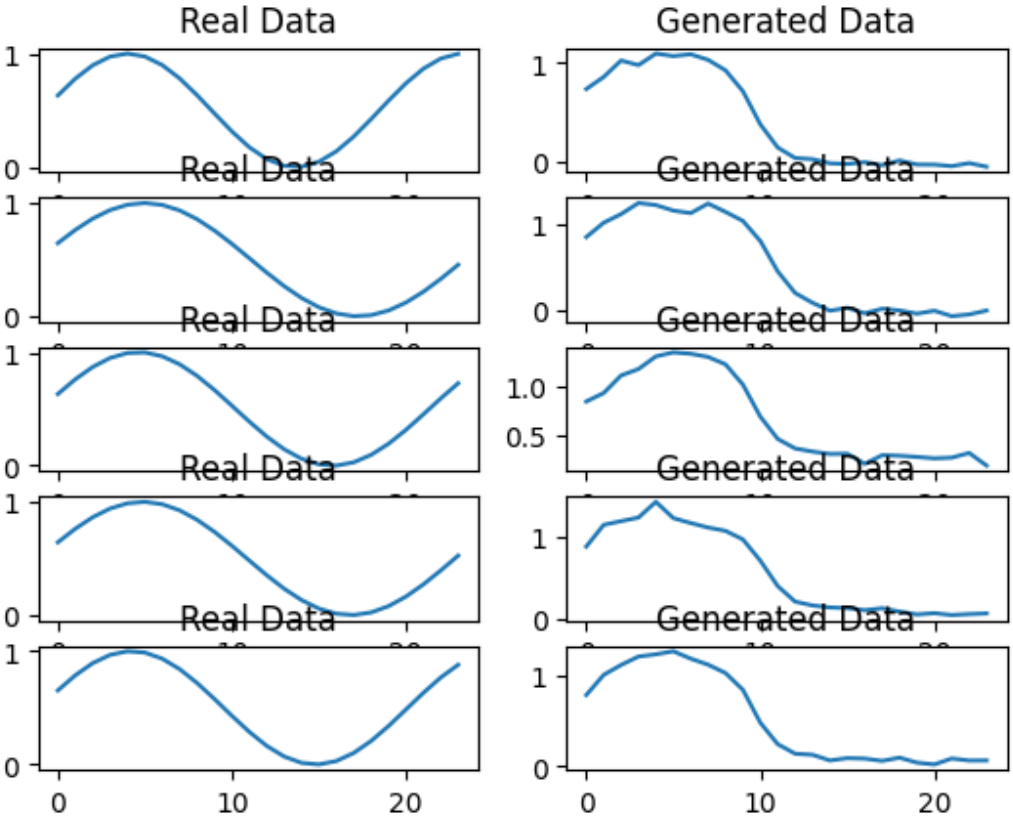

Fig. 7.3. Data generated with no attention on <500 epochs

7.3 Comparison with Existing Models

In comparison with the original TimeGAN model and other variants in the literature, the proposed model has some performance trade-offs. While the attention mechanism offers the potential for improved performance by better capturing temporal dependencies in the data, it requires more computational resources and time to train effectively. The

investigated model has the performance of 0.345 which is lower than many other known GAN solutions.

Metric	Method	Sines
Discriminative Score (Lower the Better)	TimeGAN	**.011±.008**
	RCGAN	.022±.008
	C-RNN-GAN	.229±.040
	T-Forcing	.495±.001
	P-Forcing	.430±.027
	WaveNet	.158±.011
	WaveGAN	.277±.013

Table 7.1. Other solutions results on Time-Series sine data (Bold indicates best performance).

Moreover, our model still generated synthetic data that somewhat resembled the real data, demonstrating its potential effectiveness. However, it fell short of the quality achieved by some existing models, likely due to the limited computational resources and training time.

This experiment underlines the fundamental trade-off in model complexity versus performance and resources required. Despite the lower quality of the generated data, the results are promising, and with more computational resources and training time, the proposed attention-based WGAN-GP model could potentially outperform existing models in generating high-quality synthetic time-series data.

In conclusion, these experiments demonstrate the feasibility of the proposed attention-based WGAN-GP for time-series data generation, while also highlighting the challenges and considerations in training such complex models. Future work should explore optimizing the model architecture and training process to achieve better performance while managing computational resources effectively.

Conclusion

This thesis presents a comprehensive exploration into the domain of time-series data generation, with a focus on using Generative Adversarial Networks (GANs). The primary contribution is the development of an attention-based Wasserstein GAN with Gradient Penalty (WGAN-GP), designed specifically to model and generate synthetic time-series data.

Our approach leverages the strengths of the TimeGAN model, extending its capabilities by integrating a multi-head self-attention mechanism. The attention mechanism has demonstrated its effectiveness in various domains by better capturing temporal dependencies in the data. The integration of the WGAN-GP loss function helps in stabilizing the training process and potentially improving the quality of the generated data.

The attention-based WGAN-GP model was meticulously designed, keeping in mind several crucial considerations such as model complexity, training stability, and computational efficiency. An end-to-end implementation, including data preprocessing, model architecture, training, testing, and visualization was performed using TensorFlow and Keras.

Experiments conducted as part of this research yielded insightful results. Despite the limitations in computational resources and training time, the model was able to generate synthetic data that somewhat resembled the real data. It was observed that the attention mechanism requires more epochs to train effectively and that its inclusion may initially slow down the training process. However, the potential benefits in capturing temporal dependencies make it a worthy trade-off.

When compared to existing models, our model presented some performance trade-offs. Yet, it showed promise and potential to outperform existing models in generating high-quality synthetic time-series data with more computational resources and training time. The results underscore the trade-off in model complexity versus performance and resources required, a key consideration in designing such models.

Future work should focus on optimizing the model architecture and training process for better performance while managing computational resources effectively. More sophisticated attention mechanisms could be explored, and further experiments with larger and more diverse datasets could be conducted to validate the model's effectiveness.

In summary, this thesis demonstrates the potential of integrating attention mechanisms with GANs for time-series data generation. It also highlights the challenges and considerations involved in designing and training such complex models. As research in this field progresses, it is hoped that the insights and contributions from this study will serve as a stepping stone for developing more advanced and effective models for time-series data generation.

The knowledge and insights gained from this study provide a valuable contribution to the growing body of research on generative models and their application in time-series

data generation. Through this work, we hope to inspire and inform future research efforts in this exciting and rapidly evolving field.

Future Work

This thesis has opened a range of interesting and exciting paths for future research, having demonstrated the potential of integrating attention mechanisms with Generative Adversarial Networks for time-series data generation. It also highlighted the intricacies and nuances involved in designing and training such models, providing a solid foundation upon which further research can build.

1. **Enhanced Attention Mechanisms**: While the implemented multi-head self-attention mechanism showed promise, there is still much room for exploration and improvement. Future work could explore more sophisticated attention mechanisms such as Transformer models or variants that are specifically designed to model temporal dependencies more effectively.
2. **Optimization of Model Parameters**: A crucial aspect for future investigation is the optimization of model parameters. While the model performed reasonably well under the given set of parameters, future research could explore more systematic approaches to parameter tuning, possibly incorporating techniques like grid search or Bayesian optimization to identify the optimal model configuration.
3. **Exploring Alternative Loss Functions**: The Wasserstein loss with gradient penalty (WGAN-GP) provided stability in training. However, exploring alternative loss functions, or modifications to the WGAN-GP, might yield better results and should be considered in future work.
4. **Model Scaling and Parallelization**: As the computational resources required for training these models are substantial, future research could investigate techniques for model scaling and parallelization. By utilizing distributed computing resources more effectively, it may be possible to accelerate training and handle larger and more complex datasets.
5. **Robust Evaluation Metrics**: This work focused on visual inspection and comparison of the generated data. Future research could focus on developing or incorporating more robust and quantitative evaluation metrics for comparing generated time-series data with real data.
6. **Real-world Applications**: Lastly, future research should also consider practical, real-world applications of the model. This could involve generating synthetic time-series

data in domains like finance, energy, health, or climate science, where privacy concerns or data scarcity often pose challenges.

In conclusion, the field of time-series data generation using GANs is a rich and rapidly evolving area of research. This thesis provides a stepping stone towards understanding and exploiting the power of these techniques. The opportunities for future work are broad and diverse, and it is hoped that this research will inspire further exploration and development in this domain.

References

1. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2014). Generative adversarial nets. In Advances in neural information processing systems.
2. Arjovsky, M., Chintala, S., & Bottou, L. (2017). Wasserstein generative adversarial networks. In Proceedings of the 34th International Conference on Machine Learning (Vol. 70).
3. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In Advances in neural information processing systems.
4. Jang, E., Gu, S., & Poole, B. (2017). Categorical reparameterization with Gumbel-Softmax. In 5th International Conference on Learning Representations.
5. Cho, K., Van Merriënboer, B., Bahdanau, D., & Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. arXiv preprint.
6. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8).
7. Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016). Layer normalization. arXiv preprint.
8. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 15(1).
9. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., ... & Lerer, A. (2017). Automatic differentiation in PyTorch. In NIPS 2017 Workshop on Autodiff.
10. Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint.
11. Radford, A., Metz, L., & Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint.

12. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., & Courville, A. C. (2017). Improved training of wasserstein gans. In Advances in neural information processing systems.
13. Yoon, J., Jordon, J., & van der Schaar, M. (2019). Ganite: Estimation of individualized treatment effects using generative adversarial nets. In Proceedings of the International Conference on Learning Representations.
14. Esteban, C., Hyland, S. L., & Rätsch, G. (2017). Real-valued (medical) time series generation with recurrent conditional gans. arXiv preprint.
15. Brock, A., Donahue, J., & Simonyan, K. (2018). Large scale gan training for high fidelity natural image synthesis. arXiv preprint.
16. Karras, T., Aila, T., Laine, S., & Lehtinen, J. (2017). Progressive growing of gans for improved quality, stability, and variation. arXiv preprint.
17. Zhang, H., Goodfellow, I., Metaxas, D., & Odena, A. (2019). Self-attention generative adversarial networks. In Proceedings of the 36th International Conference on Machine Learning.
18. Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., & Abbeel, P. (2016). Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In Advances in neural information processing systems.
19. Goodfellow, I. (2016). NIPS 2016 tutorial: Generative adversarial networks. arXiv preprint.
20. Mirza, M., & Osindero, S. (2014). Conditional generative adversarial nets. arXiv preprint.
21. Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. IEEE transactions on neural networks.
22. Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., & Frey, B. (2015). Adversarial autoencoders. arXiv preprint.
23. Lafferty, J., McCallum, A., & Pereira, F. C. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data.
24. Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems.

GitHub repository of the code:

https://github.com/ThePloy1990/AB-WTimeGAN-GP/tree/main