AEC-GAN: Adversarial Error Correction GANs for Auto-Regressive Long Time-Series Generation

Lei Wang, Liang Zeng, Jian Li
Institute for Interdisciplinary Information Sciences (IIIS), Tsinghua University
{wanglei20, zengl18}@mails.tsinghua.edu.cn, lijian83@mail.tsinghua.edu.cn

Abstract

Large-scale high-quality data is critical for training modern deep neural networks. However, data acquisition can be costly or time-consuming for many time-series applications, thus researchers turn to generative models for generating synthetic time-series data. In particular, recent generative adversarial networks (GANs) have achieved remarkable success in time-series generation. Despite their success, existing GAN models typically generate the sequences in an auto-regressive manner, and we empirically observe that they suffer from severe distribution shifts and bias amplification, especially when generating long sequences. To resolve this problem, we propose Adversarial Error Correction GAN (AEC-GAN), which is capable of dynamically correcting the bias in the past generated data to alleviate the risk of distribution shifts and thus can generate high-quality long sequences. AEC-GAN contains two main innovations: (1) We develop an error correction module to mitigate the bias. In the training phase, we adversarially perturb the realistic time-series data and then optimize this module to reconstruct the original data. In the generation phase, this module can act as an efficient regulator to detect and mitigate the bias. (2) We propose an augmentation method to facilitate GAN's training by introducing adversarial examples. Thus, AEC-GAN can generate high-quality sequences of arbitrary lengths, and the synthetic data can be readily applied to downstream tasks to boost their performance. We conduct extensive experiments on six widely used datasets and three state-of-the-art time-series forecasting models to evaluate the quality of our synthetic time-series data in different lengths and downstream tasks. Both the qualitative and quantitative experimental results demonstrate the superior performance of AEC-GAN over other deep generative models for time-series generation.

1 Introduction

Time-series forecasting has been widely applied in both academia and industry, such as disease propagation (Kapoor et al. 2020), energy deployment (Zhou et al. 2021), and quantitative investment (Ding et al. 2020). Recent research has achieved significant progress in time-series forecasting with the development of the temporal convolutional network (TCN) (Bai, Kolter, and Koltun 2018) and transformer (Vaswani et al. 2017). With the increasing model capacity, large-scale high-quality data is required to fully realize the potential of time-series forecasting (Lim and Zohren 2021). For sufficient data, synthesizing data via deep generative models has achieved remarkable progress and has attracted much attention (reviewed in Brophy et al. (2021)).

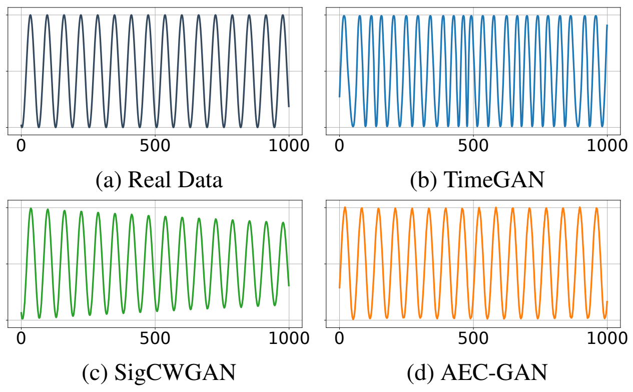


Figure 1: (a) is the time series used for training. (b-d) are the synthetic time series auto-regressively generated by corresponding methods. The time series generated by (b) TimeGAN exhibits unstable frequency, and that generated by (c) SigCWGAN suffers from amplitude attenuation.

Generative Adversarial Network (GAN) (Goodfellow et al. 2014) is a versatile generative framework for generating data and has shown excellent performance in computer vision (Karras et al. 2020b, 2021). Meanwhile, much effort has been devoted to applying GANs to time-series generation (Brophy et al. 2021). To name a few: (1) In a recursive manner, Recurrent Condition GAN (RCGAN) (Esteban, Hyland, and Rätsch 2017) and QuantGAN (Wiese et al. 2020) exploit LSTM or TCN to replace the Convolutional Neural Network (CNN) for time-series generation. (2) Considering the temporal nature, TimeGAN (Yoon, Jarrett, and Van der Schaar 2019) borrows ideas from auto-regressive models and explicitly models the temporal transitions $p(\boldsymbol{x}_t|\boldsymbol{x}_{1:t-1})$ in the latent space. SigCWGAN (Ni et al. 2020) captures the auto-regressive structure of sequences and builds a high-quality conditional generative model.

Despite the encouraging progress in time-series generation, we argue that generating long sequences in an auto-regressive manner still exhibits the inherent difficulty of distribution shifts. To demonstrate this problem, we implement

Copyright © 2023, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

several time-series GANs to generate a sine function. When scaled to 1000 generation steps in the generation phase, the generated time series of TimeGAN and SigCWGAN exhibit unstable frequency or attenuated amplitude, as shown in Fig. 1. This phenomenon amplifies along with the auto-regressive generation. Due to cumulative errors in the sequence generation, the long sequence generation is sensitive to the distribution shifts and inapplicable for the downstream long sequence time-series forecasting (LSTF) task.

To address this issue, we propose a novel conditional GAN, named Adversarial Error Correction GAN (AEC-GAN). AEC-GAN refines generated sequences (regarded as the conditioning variables) for high-quality long sequence generation auto-regressively. Most importantly, we develop a scalable error correction module to mitigate the bias in the past generated data. The module is optimized to reconstruct the realistic conditioning variables perturbed by an adversarial attack algorithm. As the module recursively operates on the condition in the generation phase, it enables the GAN model to scale to the high-quality long sequence generation based on auto-regressively refined conditions. Moreover, we find that the adversarial examples can be used as augmented data to stabilize the GAN's training.

The contributions of our paper are summarized as follows:

- We argue that auto-regressively generating long time series is sensitive to distribution shifts, and we have empirically observed severe distribution shifts in the long sequence generation of existing time-series GANs.
- We propose an error correction module that dynamically modifies the generated data to mitigate the risk of distribution shifts. Hence, AEC-GAN can auto-regressively generate high-quality data with arbitrary lengths to adapt to various downstream forecasting tasks.
- We theoretically explain how the error correction module improves the long sequence generation in the auto-regressive setting. In addition, we propose an augmentation method to stabilize the GAN's training by introducing adversarial examples in training.
- We conduct extensive experiments on six widely used datasets, and the results show that AEC-GAN outperforms other time-series GANs at different lengths. We apply the synthetic data to downstream forecasting tasks and boost the performances of three time-series forecasting models by 17.9% on average.

2 Related Work

2.1 Generative Adversarial Network

Generative Adversarial Networks (GANs) (Goodfellow et al. 2014) integrate generator and discriminator as a two-player game, where the generator learns to generate vivid contents to mislead the discriminator's judgements. WGAN (Arjovsky, Chintala, and Bottou 2017) and LS-GAN (Mao et al. 2017) propose new objectives which minimize the divergence between the realistic and synthetic distributions to stabilize the GAN's training. When available data is limited, optimizing GANs is more challenging, since the discriminator overfits quickly and breaks the equilibrium (Karras et al. 2020a). To address this issue, Jiang et al. (2021), Karras et al. (2020a) and Tran et al. (2021) adopt augmentation for the training data to prevent overfitting.

Recently, GANs have attracted extensive attention in the time-series generation. A broad literature (Esteban, Hyland, and Rätsch 2017; Wiese et al. 2020; Yoon, Jarrett, and Van der Schaar 2019) deploys LSTM or TCN to design network architectures to adapt to domain-specific generation tasks, such as limit order simulation (Li et al. 2020), audio generation (Donahue, McAuley, and Puckette 2018), and biomedical signal (Hazra and Byun 2020). Inspired by auto-regressive models (Hamilton 2020), recent efforts (Yoon, Jarrett, and Van der Schaar 2019; Ni et al. 2020; Jarrett, Bica, and van der Schaar 2021) have been devoted to learning conditional dynamics, which is essential in downstream time-series analysis. For the long sequence generation, PSA-GAN (Paul et al. 2021) adopts a progressive growing architecture that adds additional modules during training.

2.2 Adversarial Attacks and Defenses

Adversarial attack algorithms usually add imperceptible perturbations to the inputs to mislead the neural networks (NNs) into giving undesirable outputs (Chakraborty et al. 2018). These subtly crafted inputs are well-known as adversarial examples (Szegedy et al. 2013). PGD-attack (Madry et al. 2017) is an effective attack algorithm that exploits projected gradient descent to search for efficient attack directions. Furthermore, adversarial training (Goodfellow, Shlens, and Szegedy 2014) mixes the adversarial examples into the original training dataset to defend against corresponding attacks.

Adversarial algorithms have also been widely used in GANs. Xiao et al. (2018) propose to generate adversarial examples based on GANs. Liu and Hsieh (2019) reveal how adversarial training and GANs can promote each other. Technically, they propose Rob-GAN, which creates adversarial examples to fool the discriminator. In contrast, the adversarial examples in our AEC-GAN are created for the input conditions and serve as fake conditions, aiming to assist the training of the error correction module.

3 Preliminary

In this paper, the time-series dataset $\boldsymbol{S}$ contains a sequence of data points with d dimensions, i.e., $\boldsymbol{S} = \{\boldsymbol{x}_t \in \mathbb{R}^d\}_{1 \leq t \leq T}$. We suppose the time-series dataset obeys a conditional distribution $\mathcal{P}(\boldsymbol{x}|\boldsymbol{c})$ where $\boldsymbol{c}$ represents the knowledge of the context, e.g., $\boldsymbol{x}_{t+1:t+q} \sim \mathcal{P}(\boldsymbol{x}_{t+1:t+q}|\boldsymbol{c}_t)$. Following PSA-GAN (Paul et al. 2021), the knowledge of the context is represented as a sequence, i.e., $\boldsymbol{c}_t = \boldsymbol{x}_{t-p+1:t}$, where p is the context length. We denote the distribution of the context $\boldsymbol{c}$ as $\mathcal{P}(\boldsymbol{c})$.

Our goal is to generate high-quality time-series data with *arbitrary lengths* given a short context $\boldsymbol{c}_t$ (context length is p) sampled from history data. We develop a conditional GAN (modeling $\mathcal{P}(\boldsymbol{x}|\boldsymbol{c})$), which contains a generator G_θ and a discriminator D_ϕ parameterized by θ and ϕ respectively. In the training phase, given the context $\boldsymbol{c}_t = \boldsymbol{x}_{t-p+1:t}$, the GAN model is optimized to generate a time series of the target length q, i.e., $\hat{\boldsymbol{x}}_{t+1:t+q}$ (we use $\hat{}$ to denote the generated data). Specifically, the one-step generator G_θ

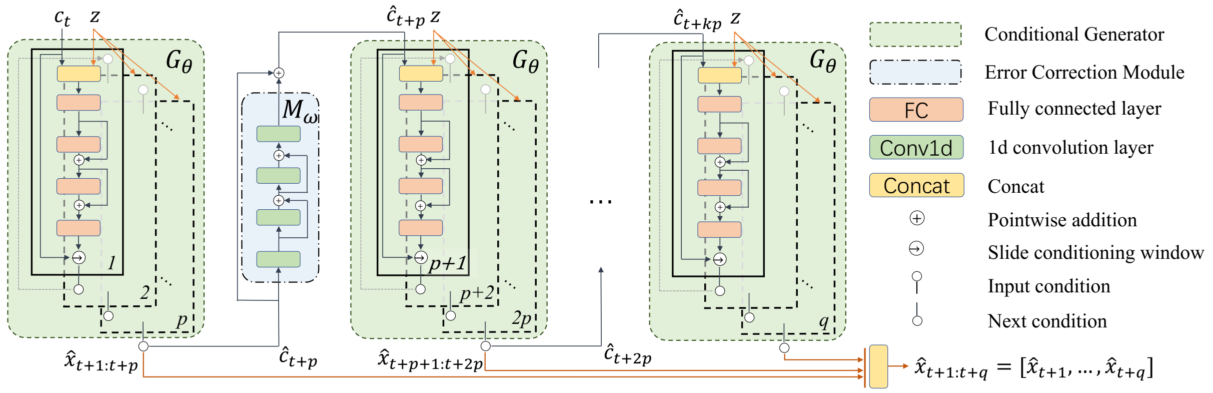


Figure 2: The network architecture of generation in AEC-GAN. The generator G_θ generates sequences auto-regressively and the correction module M_ω modifies the conditioning variable per p steps. *Slide conditioning window* means updating the conditioning variable by the newly generated data. The network parameters are shared across the generation of each step.

first transforms a Gaussian noise vector $\boldsymbol{z}_{t+1} \in \mathbb{R}^d$ (drawn from $\mathcal{N}(\mathbf{0}, \boldsymbol{I}_{d\times d})$) into the a generated data point $\hat{\boldsymbol{x}}_{t+1}$ given the condition $\boldsymbol{c}_t$, i.e., $\hat{\boldsymbol{x}}_{t+1} = G_\theta(\boldsymbol{z}_{t+1}|\boldsymbol{c}_t) \in \mathbb{R}^d$. Then G_θ generate $\hat{\boldsymbol{x}}_{t+2} = G_\theta(\boldsymbol{z}_{t+2}|\hat{\boldsymbol{c}}_{t+1})$ based on the updated condition $\hat{\boldsymbol{c}}_{t+1}$, where $\hat{\boldsymbol{c}}_{t+1} = [\boldsymbol{x}_{t-p+2:t}, \hat{\boldsymbol{x}}_{t+1}]$. Auto-regressively, we obtain the target sequence $\hat{\boldsymbol{x}}_{t+1:t+q} = [G_\theta(\boldsymbol{z}_{t+1}|\boldsymbol{c}_t), ..., G_\theta(\boldsymbol{z}_{t+q}|\hat{\boldsymbol{c}}_{t+q-1})]$. For brevity, we overload the notation G_θ to denote the generations $\hat{\boldsymbol{x}}_{t+1:t+q}$ as $G_\theta(\boldsymbol{z}_{t+1:t+q}|\boldsymbol{c}_t)$. The discriminator D_ϕ is optimized to distinguish generated sequence $\hat{\boldsymbol{x}}_{t+1:t+q}$ from real sequence $\boldsymbol{x}_{t+1:t+q}$, given the condition $\boldsymbol{c}_t$. For more realistic data, $D_\phi(\cdot, \boldsymbol{c}_t) : \mathbb{R}^{d\times q} \times \mathbb{R}^{d\times p} \to \mathbb{R}$ is desired to give a higher discriminant score. In the rest of the paper, we omit the subscripts of $\boldsymbol{z}$ unless required. The GAN's objective function (Mirza and Osindero 2014) is shown in function (1). After training, G_θ can generate sequences step by step after feeding in a context $\boldsymbol{c}_t$, without generation length limitation.

$$\begin{aligned}\min_\theta \max_\phi \mathop{\mathbb{E}}_{\substack{\boldsymbol{c}_t\sim\mathcal{P}(\boldsymbol{c})\\ \boldsymbol{x}_{t+1:t+q}\in\mathcal{P}(\boldsymbol{x}|\boldsymbol{c}_t))}} &\Big[\log D_\phi(\boldsymbol{x}_{t+1:t+q}, \boldsymbol{c}_t)\Big] + \\ &\mathop{\mathbb{E}}_{\substack{\boldsymbol{c}_t\sim\mathcal{P}(\boldsymbol{c})\\ \boldsymbol{z}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I})}} \Big[\log\Big(1 - D_\phi(G_\theta(\boldsymbol{z}|\boldsymbol{c}_t), \boldsymbol{c}_t)\Big)\Big].\end{aligned} \tag{1}$$

4 Adversarial Error Correction GAN

In this section, we introduce AEC-GAN (Adversarial Error Correction GAN) to generate high-quality sequences with arbitrary lengths. First, we show that an auto-regressive generation process will inevitably deviate from the actual distribution as the generated length increases. Then, we introduce an error correction module to mitigate the distribution shifts and propose an augmentation method to facilitate training.

4.1 Distribution Shifts in Existing GANs

A conditional GAN can naturally generate sequences of arbitrary lengths auto-regressively (Sec. 3). Fig. 1 shows the auto-regressively generated sine functions by several time-series GANs. It shows that TimeGAN and SigCWGAN suffer from unstable frequency and attenuated amplitude, and this phenomenon amplifies as the generation proceeds.

We present the following Prop. 1 to explain this phenomenon. Briefly speaking, for an $AR(1)$ process (Hamilton 2020), [1] the KL-divergence between the L_{th} step generation and the underlying distribution is $\Omega(\delta L)$, where L is generation length and δ is estimation error in terms of max eigenvalues. The proof of Prop. 1 is detailed in the appendix.

Proposition 1 *Assume the ground-truth generation process is an $AR(1)$ process without drift, $\boldsymbol{x}_t = \boldsymbol{\Phi}\boldsymbol{x}_{t-1} + \boldsymbol{a}_t$, where $\boldsymbol{\Phi} \in \mathbb{R}^{d\times d}$ is symmetric and $\boldsymbol{a}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$. We denote $\hat{\boldsymbol{x}}_t = \hat{\boldsymbol{\Phi}}\hat{\boldsymbol{x}}_{t-1} + \hat{\boldsymbol{a}}_t$ as the estimated generation process, where $\hat{\boldsymbol{\Phi}} \in \mathbb{R}^{d\times d}$ is also symmetric and $\hat{\boldsymbol{a}}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$. The max eigenvalues of $\boldsymbol{\Phi}$ and $\hat{\boldsymbol{\Phi}}$ are denoted as λ and $\hat{\lambda}$, and they differ by a small quantity δ, i.e., $|1 - \hat{\lambda}/\lambda| = \delta$. Generating L steps time-series data from $\boldsymbol{x}_0$, if $\lambda \geq 1$, then*

$$D_{KL}\Big(p(\boldsymbol{x}_L|\boldsymbol{x}_0)||p(\hat{\boldsymbol{x}}_L|\boldsymbol{x}_0)\Big) = \Omega(\delta L), \tag{2}$$

where $D_{KL}(\,\cdot\,||\,\cdot\,)$ denotes the KL-divergence, $p(\cdot)$ denotes the probability density function.

Prop. 1 indicates that generating long sequences is sensitive to distribution shifts, since L gets larger as the generation proceeds. Thus, we propose the following error correction module to alleviate this problem.

4.2 Error Correction Module

In this section, we introduce an *error correction module* $M_\omega(\cdot) : \mathbb{R}^{d\times p} \to \mathbb{R}^{d\times p}$, parameterized by ω, to mitigate the bias in the past generated sequences. Given the condition $\boldsymbol{c}_t \in \mathbb{R}^{d\times p} \sim \mathcal{P}(\boldsymbol{c})$ to generate time-series data beginning from the time t, the generator G_θ first generates a p-step sequence $\hat{\boldsymbol{x}}_{t+1:t+p}$ in a rolling way, which is regarded as the next condition $\hat{\boldsymbol{c}}_{t+p}$ for the following generation. M_ω modifies the condition $\hat{\boldsymbol{c}}_{t+p}$ with $\hat{\boldsymbol{c}}_{t+p} + M_\omega(\hat{\boldsymbol{c}}_{t+p})$. Then the generator generates the following p-step sequence $\hat{\boldsymbol{x}}_{t+p+1:t+2p}$ given the refined condition $\hat{\boldsymbol{c}}_{t+p} + M_\omega(\hat{\boldsymbol{c}}_{t+p})$. After this, M_ω modifies the new condition $\hat{\boldsymbol{c}}_{t+2p}$ (i.e., $\hat{\boldsymbol{x}}_{t+p+1:t+2p}$) again.

[1]An $AR(p)$ process can be converted to an $AR(1)$ process with higher dimensions (Hamilton 2020).

We repeat the procedure until we obtain the target length q. In general, M_ω modifies the generated condition $\hat{\boldsymbol{c}}_{t+kp}$ with $\hat{\boldsymbol{c}}_{t+kp}+M_\omega(\hat{\boldsymbol{c}}_{t+kp})$ every time generating a p-step subsequence in the auto-regressive generation process, and the generator proceeds to the next round generation given the modified condition $\hat{\boldsymbol{c}}_{t+kp}+M_\omega(\hat{\boldsymbol{c}}_{t+kp})$, where $k \in \mathbb{Z}_+$. In such a way, M_ω mitigates the bias adaptively. Finally, we gather the generated $\hat{\boldsymbol{x}}_{t+1:t+q}$ as the final output denoted by $G_\theta^M(\boldsymbol{z}|\boldsymbol{c}_t)$. The generation process with correction is detailed in Algorithm 1 and illustrated in Fig. 2.

To optimize the module M_ω, we perturb the realistic condition $\boldsymbol{c}_t$ with a slight perturbation $\boldsymbol{\delta}$ and expect $M_\omega(\boldsymbol{c}_t+\boldsymbol{\delta})$ to remove the perturbation. Specifically, $\boldsymbol{\delta}$ is created from the PGD-attack (Madry et al. 2017) within an l_2 radius δ_{max}, as shown in Eq. (3). More discussions about the kinds of perturbations can be found in the appendix.

$$\boldsymbol{\delta} = \underset{||\boldsymbol{\delta}||_2 \le \delta_{max}}{\arg\min} \log\Big[1 - D_\phi(G_\theta(\boldsymbol{z}|\boldsymbol{c}_t), \boldsymbol{c}_t+\boldsymbol{\delta})\Big]. \quad (3)$$

Besides modifying the adversarial example (i.e., $\boldsymbol{c}_t+\boldsymbol{\delta}$), $M_\omega(\boldsymbol{c}_t)$ is also supposed to output $\mathbf{0}$ for realistic $\boldsymbol{c}_t$. Hence, the objective function $\min_\omega \mathcal{L}_M(\omega)$ for M_ω is as follows:

$$\min_\omega \mathop{\mathbb{E}}_{\boldsymbol{c}_t \in \mathcal{P}(\boldsymbol{c})} \big[\mathcal{L}_2(M_\omega(\boldsymbol{c}_t+\boldsymbol{\delta}), -\boldsymbol{\delta}) + \mathcal{L}_2(M_\omega(\boldsymbol{c}_t), \mathbf{0})\big], \quad (4)$$

where $\mathcal{L}_2(\cdot,\cdot)$ denotes the MSE loss. With M_ω to mitigate bias, we can generate high-quality long sequences, and we explain the reason in what follows.

Intuitively, there exists a discrepancy between the training and testing phases. In the training phase, G_θ only generates sequences with a target length q, as shown in the objective function (1). In the testing phase, G_θ is desired to generate arbitrary length sequences auto-regressively, especially the long sequences. For this issue, we argue that the error correction module can bridge the gap between generation lengths in the training and testing phases to improve the quality of long sequence generation.

We specify the generation length in training as p for convenience of analysis, i.e., $G_\theta(\boldsymbol{z}|\boldsymbol{c}_t) \in \mathbb{R}^{d\times p}$. In the testing phase, suppose we are required to generate sequences with length Kp, given $\boldsymbol{c}_p \sim \mathcal{P}(\boldsymbol{c}_p)$ (i.e., $\boldsymbol{x}_{1:p}$). Concretely, the generator G_θ first generates a p-step sequence $\hat{\boldsymbol{c}}_{2p}$ (i.e., $\hat{\boldsymbol{x}}_{p+1:2p}$) based on $\boldsymbol{c}_p$ and then auto-regressively generates $\hat{\boldsymbol{c}}_{kp+p}$ (i.e., $\hat{\boldsymbol{x}}_{kp+1:kp+p}$) based on the refined generated data $\hat{\boldsymbol{c}}_{kp}+M_\omega(\hat{\boldsymbol{c}}_{kp})$, where $2 \le k \le K$. To assess the generation quality, we first assume that the realistic sequences are $\boldsymbol{c}_{kp+p} = \boldsymbol{x}_{kp+1:kp+p} \sim \mathcal{P}(\boldsymbol{x}_{kp+1:kp+p}|\boldsymbol{c}_{kp})$ (note that $\boldsymbol{c}_p \sim \mathcal{P}(\boldsymbol{c}_p)$). Then we calculate the generator's loss $\mathcal{L}_{test}(\theta,\phi,\omega)$ for the generated sequence, i.e., the loss of the discriminator's judgement about whether the generated sequence $\hat{\boldsymbol{x}}_{kp+1:kp+p} = G_\theta(\boldsymbol{z}|\hat{\boldsymbol{c}}_{kp}+M_\omega(\hat{\boldsymbol{c}}_{kp}))$ is drawn from the ground-truth $\mathcal{P}(\boldsymbol{x}_{kp+1:kp+p}|\boldsymbol{c}_{kp})$, as follows:

$$\mathop{\mathbb{E}}_{\substack{\boldsymbol{z}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I})\\ \boldsymbol{c}_p,\dots,\boldsymbol{c}_{Kp}}} \sum_{k=2}^{K} \log\Big[1 - D_\phi(G_\theta(\boldsymbol{z}|\hat{\boldsymbol{c}}_{kp}+M_\omega(\hat{\boldsymbol{c}}_{kp})), \boldsymbol{c}_{kp})\Big]. \quad (5)$$

We denote $\boldsymbol{\delta}_k = \hat{\boldsymbol{c}}_{kp} - \boldsymbol{c}_{kp}$ (related to $\boldsymbol{z}$ and $\boldsymbol{c}_{kp}$) as the generation bias. Then, the generation part of the equation (5) can be rewritten as follows:

$$G_\theta(\boldsymbol{z}|\boldsymbol{c}_{kp}+\boldsymbol{\delta}_k+M_\omega(\boldsymbol{c}_{kp}+\boldsymbol{\delta}_k)) \quad (6)$$

Algorithm 1: Auto-Regressive Generation

```
  Input: Condition $\boldsymbol{c}_t \in \mathbb{R}^{d\times p}$, generation length $q$,
         generator $G_\theta$, error-correction module $M_\omega$.
  Output: $\hat{\boldsymbol{x}}_{t+1:t+q} \in \mathbb{R}^{d\times q}$.
1 Init $\hat{\boldsymbol{c}}_t = \boldsymbol{c}_t$.
2 for $i$ = 0,...,q-1 do
3     if $i+1 \equiv 0 \pmod p$ then
4         $\hat{\boldsymbol{c}}_{t+i} \leftarrow \hat{\boldsymbol{c}}_{t+i} + M_\omega(\hat{\boldsymbol{c}}_{t+i})$
5     $\boldsymbol{z}_{t+i+1} \leftarrow \mathcal{N}(\mathbf{0},\boldsymbol{I}) \in \mathbb{R}^d$.
6     $\hat{\boldsymbol{x}}_{t+i+1} \leftarrow G_\theta(\boldsymbol{z}_{t+i+1}|\hat{\boldsymbol{c}}_{t+i}) \in \mathbb{R}^d$.
7     $\hat{\boldsymbol{c}}_{t+i+1} \leftarrow pop(\hat{\boldsymbol{c}}_{t+i}) \cup \hat{\boldsymbol{x}}_{t+i+1}$.   ▷ $pop(\cdot)$ means
        to remove the earliest step.
8 Return: $\hat{\boldsymbol{x}}_{t+1:t+q} = [\hat{\boldsymbol{x}}_{t+1}, \cdots, \hat{\boldsymbol{x}}_{t+q}]$.
```

For brevity, we let $\boldsymbol{\epsilon}_{kp} = \boldsymbol{\delta}_k + M_\omega(\boldsymbol{c}_{kp}+\boldsymbol{\delta}_k)$. With a powerful M_ω to mitigate the bias, $\boldsymbol{\epsilon}_{kp}$ is small. Then with Taylor expansion, we get the following upper bound:

$$\begin{aligned}\mathcal{L}_{test} &= \mathop{\mathbb{E}}_{\substack{\boldsymbol{z}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I})\\ \boldsymbol{c}_p,\dots,\boldsymbol{c}_{Kp}}} \sum_{k=2}^{K} \log\Big[1 - D_\phi(G_\theta(\boldsymbol{z}|\boldsymbol{c}_{kp}+\boldsymbol{\epsilon}_{kp}), \boldsymbol{c}_{kp})\Big] \\ &\le \mathop{\mathbb{E}}_{\substack{\boldsymbol{z}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I}),\\ \boldsymbol{c}_p,\dots,\boldsymbol{c}_{Kp}}} \sum_{k=2}^{K} \Big\{ \log\Big[1 - D_\phi(G_\theta(\boldsymbol{z}|\boldsymbol{c}_{kp}), \boldsymbol{c}_{kp})\Big] \\ &\quad + \Big\| \frac{\partial \log(1-D_\phi)}{\partial D_\phi} \frac{\partial D_\phi}{\partial G_\theta} \frac{\partial G_\theta}{\partial \boldsymbol{c}_{kp}} \cdot \boldsymbol{\epsilon}_{kp} \Big\| + O(\|\boldsymbol{\epsilon}_{kp}\|^2) \Big\}.\end{aligned} \quad (7)$$

Since $G_\theta(\boldsymbol{z}|\boldsymbol{c}_{kp})$ generates data with the length specified in training given the real $\boldsymbol{c}_{kp}$, the first term represents the generator's loss in training: $\mathcal{L}_G(\boldsymbol{z}, \boldsymbol{c}_t; \theta, \phi) = \log[1 - D_\phi(\hat{\boldsymbol{x}}, \boldsymbol{c}_t)]$, where $\hat{\boldsymbol{x}}$ represents the generated data (i.e., $G_\theta(\boldsymbol{z}|\boldsymbol{c}_t)$). Then,

$$\begin{aligned}\mathcal{L}_{test} \le \mathop{\mathbb{E}}_{\substack{\boldsymbol{z}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I}),\\ \boldsymbol{c}_p,\dots,\boldsymbol{c}_{Kp}}} \sum_{k=2}^{K} \Big\{ &\mathcal{L}_G(\boldsymbol{z}, \boldsymbol{c}_{kp}; \theta, \phi) \\ &+ \|\nabla_{\hat{\boldsymbol{x}}} \mathcal{L}_G\| \cdot \|\nabla_{\boldsymbol{c}_{kp}} G_\theta\| \cdot \|\boldsymbol{\epsilon}_{kp}\| + O(\|\boldsymbol{\epsilon}_{kp}\|^2) \Big\}.\end{aligned} \quad (8)$$

From above, $\mathcal{L}_{test}$ is bounded by the training loss and some terms related to $\|\boldsymbol{\epsilon}_{kp}\|$, and we explicitly minimize the $\|\boldsymbol{\epsilon}_{kp}\|$ in the objective function (4). With the module M_ω, we can optimize the upper bound of the loss $\mathcal{L}_{test}$ to generate longer sequences with high quality, rather than being limited to the length specified in training. In addition, the gradients $\|\nabla_{\boldsymbol{c}} G_\theta\|$ and $\|\nabla_{\hat{\boldsymbol{x}}} \mathcal{L}_G\|$ also affect the upper bound, and we conduct experiments to quantify them in Sec. 5.4.

4.3 Augmentation Method

In this section, we propose an augmentation method to facilitate the GAN's training. Generally, GANs only generate $\hat{\boldsymbol{x}}_\tau^1 = G_\theta(\boldsymbol{z}_\tau|\boldsymbol{c}_t)$ to mislead D_ϕ, where τ denotes a time span (e.g., $t+1:t+q$). We propose to enrich the generations for training the GANs by using adversarial examples (i.e., $\boldsymbol{c}_t+\boldsymbol{\delta}$) and M_ω, detailed as in Eq. (9),

$$\begin{aligned} &\hat{\boldsymbol{x}}_\tau^1 = G_\theta(\boldsymbol{z}_\tau|\boldsymbol{c}_t) \quad &&\hat{\boldsymbol{x}}_\tau^2 = G_\theta(\boldsymbol{z}_\tau|\boldsymbol{c}_t+\boldsymbol{\delta}) \\ &\hat{\boldsymbol{x}}_\tau^3 = G_\theta^M(\boldsymbol{z}_\tau|\boldsymbol{c}_t) \quad &&\hat{\boldsymbol{x}}_\tau^4 = G_\theta^M(\boldsymbol{z}_\tau|\boldsymbol{c}_t+\boldsymbol{\delta}). \end{aligned} \quad (9)$$

	ETTh1			ETTh2			US Births			ILI		
Methods	ACF	Skew	Kurt	ACF	Skew	Kurt	ACF	Skew	Kurt	ACF	Skew	Kurt
QuantGAN	0.403	4.370	8.323	0.742	3.140	18.27	0.433	4.409	18.57	0.703	4.770	21.80
TimeGAN	0.636	2.596	2.103	0.169	2.117	3.013	0.844	0.315	0.822	0.804	27.54	5498
Cot-GAN	0.153	2.082	7.451	0.157	2.138	6.419	0.250	1.198	2.445	**0.312**	1.792	8.560
SigCWGAN	0.155	3.021	7.081	0.142	**1.973**	3.281	0.116	2.593	9.284	0.448	**0.550**	1.930
AEC-GAN	**0.055**	**1.911**	**0.714**	**0.115**	2.014	**2.441**	**0.036**	**0.109**	**0.061**	0.329	0.929	**0.802**

Table 1: Quantitative comparison of generation quality (lower is better, best results are in bold). All values represent the MSE error of statistical indicators corresponding to synthetic and realistic sequences.

Among these data, $\hat{\boldsymbol{x}}_\tau^1$ denotes the generated data optimized in inequality (8), $\hat{\boldsymbol{x}}_\tau^2$ and $\hat{\boldsymbol{x}}_\tau^4$ encourage the G_θ to resist perturbation (i.e., $\boldsymbol{\delta}$) in the condition, and $\hat{\boldsymbol{x}}_\tau^3$ denotes the generation in the testing phase. The objective function for G_θ, i.e., $\min_\theta \mathcal{L}_G^{Aug}(\theta, \phi, \omega)$, consists of four parts as follows:

$$\mathop{\mathbb{E}}_{\substack{\boldsymbol{c}_t\sim\mathcal{P}(\boldsymbol{c})\\ \boldsymbol{z}\sim\mathcal{N}(\boldsymbol{0},\boldsymbol{I})}}\Big[\log\big(1-D_\phi(\hat{\boldsymbol{x}}_\tau^1,\boldsymbol{c}_t)\big)+\log\big(1-D_\phi(\hat{\boldsymbol{x}}_\tau^3,\boldsymbol{c}_t)\big) \tag{10}$$
$$\log\big(1-D_\phi(\hat{\boldsymbol{x}}_\tau^2,\boldsymbol{c}_t+\boldsymbol{\delta})\big)+\log\big(1-D_\phi(\hat{\boldsymbol{x}}_\tau^4,\boldsymbol{c}_t+\boldsymbol{\delta})\big)\Big].$$

Augmented with these generations, the objective function $\max_\phi \mathcal{L}_D^{Aug}(\theta, \phi, \omega)$ for D_ϕ can be formulated as follows:

$$\max_\phi \mathop{\mathbb{E}}_{\substack{\boldsymbol{c}_t\sim\mathcal{P}(\boldsymbol{c})\\ \boldsymbol{x}_\tau\sim\mathcal{P}(\boldsymbol{x}|\boldsymbol{c}_t)}}\Big[\log D_\phi(\boldsymbol{x}_\tau,\boldsymbol{c}_t)\Big]+\mathcal{L}_G^{Aug}(\theta,\phi,\omega). \tag{11}$$

It is worth noting that the slight perturbation $\boldsymbol{\delta}$ aims to fool the discriminator's judgement, since it contradicts the discriminator's objective, as shown in Eq. (3). Hence, optimizing the discriminator to distinguish the $\boldsymbol{c}_t + \boldsymbol{\delta}$ acts as a regularizer to prevent overfitting, as shown in Eq. (11).

In the training phase, we jointly optimize G_θ, D_ϕ and M_ω. After training, G_θ^M can generate sequences with arbitrary lengths by feeding in a context $\boldsymbol{c}_t$ sampled from the training set.

In summary, through an error correction module, we can adaptively mitigate the distribution shifts and bridge the gap between the training and testing phases (Sec. 4.2). Moreover, we propose an augmentation method to further facilitate the model's training (Sec. 4.3). These parts together comprise our AEC-GAN.

4.4 Implementation

We utilize the conditional AR-FNN (Ni et al. 2020) to implement the generator G_θ and discriminator D_ϕ, which use the past sequences as conditioning variables. For error correction module M_ω, we implement it with a one-dimensional convolutional neural network (Conv1d) and refine the conditions per p steps generation as shown in Fig. 2.

5 Experiments

In this section, we conduct extensive experiments to demonstrate the effectiveness of our methods. To guide the analysis, we answer the following questions progressively.

- **RQ1:** How is the generation quality compared to other deep generative models?
- **RQ2:** How does the generated data perform in the downstream forecasting task?
- **RQ3:** Why does AEC-GAN work effectively?

5.1 Experiment Settings

Dataset We conduct experiments on six widely used temporal datasets: ETTh1, ETTh2, ETTm1, ETTm2 (Zhou et al. 2021), US Births (Godahewa et al. 2021) and ILI (Centers for Disease Control and Prevention 2020), covering three practical domains: energy, population and disease. All these datasets are split into training (80%) and testing (20%) sets in chronological order. In the training phase, we let $p/q = 168/336$ for ETTh* and US Birth, $p/q = 96/192$ for ETTm*, and $p/q = 18/36$ for ILI.

Benchmarks We compare AEC-GAN with the following well-known time-series GANs: **QuantGAN** (Wiese et al. 2020), **TimeGAN** (Yoon, Jarrett, and Van der Schaar 2019), **Cot-GAN** (Xu et al. 2020) and **SigCWGAN** (Ni et al. 2020). Furthermore, we adopt three time-series forecasting models: **SCINet** (Liu et al. 2021), **Informer** (Zhou et al. 2021), and **Autoformer** (Wu et al. 2021), to assess the performance of the generated time-series data in the downstream tasks.

Metrics We adopt the following indicators to assess the quality of the generated data statistically.

- **ACF**: We calculate the autocorrelation function (ACF) with the delay value ranging from 1 to 100:

$$\left\{ACF(k)=\frac{Cov(\hat{\boldsymbol{x}}_t,\hat{\boldsymbol{x}}_{t-k})}{Var(\hat{\boldsymbol{x}}_t)}\Big|1\le k\le 100\right\}. \tag{12}$$

- **Skew / Kurt**: We compute the skewness and kurtosis of the marginal distribution of generated data:

$$Skew=\mathbb{E}\left[\frac{(\hat{\boldsymbol{x}}_t-\boldsymbol{\mu}_{\hat{x}})^3}{\boldsymbol{\sigma}_{\hat{x}}}\right], Kurt=\mathbb{E}\left[\frac{(\hat{\boldsymbol{x}}_t-\boldsymbol{\mu}_{\hat{x}})^4}{\boldsymbol{\sigma}_{\hat{x}}}\right], \tag{13}$$

  where $\boldsymbol{\mu}_{\hat{x}} = \mathbb{E}[\hat{\boldsymbol{x}}_t]$, $\boldsymbol{\sigma}_{\hat{x}} = \sqrt{Var(\hat{\boldsymbol{x}}_t)}$.

- **FD**: Following (Paul et al. 2021), we use the FD (Fréchet Distance (Fréchet 1957)) score to assess the generation quality. A lower FD score means the synthetic sequences are closer to the realistic sequences. We encode each sequence with unsupervised representations learned by (Franceschi, Dieuleveut, and Jaggi 2019). Given the mean and covariance of realistic and synthetic representations, i.e., $(\boldsymbol{M}_r, \boldsymbol{C}_r)$ and $(\boldsymbol{M}_s, \boldsymbol{C}_s)$, FD score is obtained as follows:

$$FD=||\boldsymbol{M}_r-\boldsymbol{M}_s||_2^2+Tr(\boldsymbol{C}_r+\boldsymbol{C}_s-2(\boldsymbol{C}_r\boldsymbol{C}_s)^{1/2}). \tag{14}$$

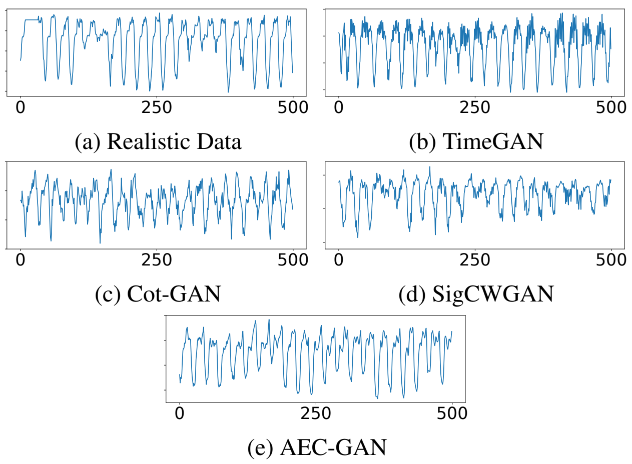
GANs		TimeGAN	Cot-GAN	SigCWGAN	Ours
ETTh1	256	3.945	1.747	0.255	**0.203**
	512	4.001	2.190	0.469	**0.274**
	1024	3.913	2.850	1.042	**0.384**
	4000	3.854	4.811	5.606	**0.752**
ETTh2	256	0.969	1.043	0.455	**0.357**
	512	1.151	1.250	0.767	**0.592**
	1024	1.514	1.707	1.341	**0.945**
	4000	2.893	4.133	3.424	**2.563**
ETTm1	256	1.884	1.383	0.173	**0.113**
	512	1.817	1.471	0.281	**0.155**
	1024	1.700	1.587	0.478	**0.245**
	4000	1.448	1.933	1.200	**0.870**
ETTm2	256	2.365	1.123	0.211	**0.087**
	512	2.437	1.398	0.385	**0.135**
	1024	2.443	1.752	0.770	**0.296**
	4000	**2.327**	2.541	3.515	2.896
US Births	256	0.395	1.329	0.116	**0.050**
	512	0.453	1.777	0.226	**0.067**
	1024	0.512	2.370	0.625	**0.113**
	4000	0.655	4.219	6.302	**0.447**
ILI	256	53.352	2.859	0.763	**0.545**
	512	96.474	4.459	1.347	**1.291**
	1024	156.660	5.564	1.687	**1.644**
	4000	307.928	8.245	2.157	**2.082**

Table 2: FD scores with different generation lengths.

5.2 Experiment 1: Generation Quality

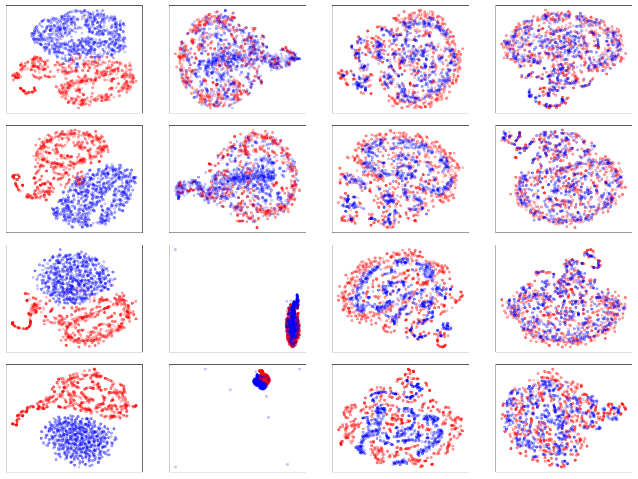
For each algorithm, we randomly generate 1000 sequences, each of which is auto-regressively generated with 4000 steps (much longer than that in training, e.g., GANs are trained to generate 36 steps on ILI). Under such a long generation, we measure the MSE error of the statistical indicators between the generated and realistic sequences (Ni et al. 2020), as shown in Table 1 (Results of ETTm1/ETTm2 are shown in the appendix due to the space limitation). It shows that AEC-GAN exhibits a low relative error in statistics, while other methods have a certain degree of distribution shift. Some generated examples are shown in Fig. 3, in which QuantGAN is ignored due to its poor performance. It shows that AEC-GAN's generation is closer to the real data both visually and statistically.

We evaluate the quality of generated data with different lengths. We respectively generate 1000 sequences with various lengths (256, 512, 1024, 4000) for each algorithm, and calculate the FD score to assess the generation quality of the generated sequences. Table 2 illustrates that AEC-GAN outperforms others in different lengths on the FD score. Furthermore, we visualize the generated data in the latent space via t-SNE (Van der Maaten and Hinton 2008), in Fig. 4. We observe that AEC-GAN matches the realistic distribution better than other benchmarks at different lengths. Especially when the generation length is 4000, there are clear boundaries in TimeGAN and SigCWGAN, and apparent anomalies in Cot-GAN. To sum up, AEC-GAN can mitigate the distribution shifts and generate high-quality long sequences.



(a) Realistic Data (b) TimeGAN (c) Cot-GAN (d) SigCWGAN (e) AEC-GAN

Figure 3: Generation examples of the variate *HUFL* in ETTh1. (a) reports the real data sample and (b-e) are generated samples.



(a) TimeGAN (b) Cot-GAN (c) SigCWGAN (d) AEC-GAN

Figure 4: T-SNE plots on ETTh1. Each row represents a generation length (256, 512, 1024, 4000). Red and blue points represent the realistic and generated data respectively.

5.3 Experiment 2: Downstream Performance

After generating the synthetic data, we quantify how much it can help the downstream forecasting task. In this experiment, we develop a fake dataset by GANs to replace the original training set for training the forecasting model and evaluate the model's performance on the original testing set (Yoon, Jarrett, and Van der Schaar 2019). Note that all the GANs are trained on the training set. For a fair comparison, we control the same amount of data between the fake dataset and the original training set (i.e., GANs generate sequences with the same amount as the sequences gained by consecutive slicing in the original training set).

We conduct experiments on SCINet, Informer and Autoformer, and all models are implemented with their suggested parameters. Table 3 shows our AEC-GAN achieves much lower errors than other methods with SCINet (results for two others can be found in the appendix). The improve-

Models		AEC-GAN		SigCWGAN		Cot-GAN		TimeGAN		QuantGAN		Original	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	168	**0.403**	**0.432**	0.519	0.495	0.497	0.465	0.923	0.696	1.398	0.943	0.450	0.453
	336	**0.498**	**0.496**	0.627	0.566	0.650	0.556	0.953	0.711	1.777	1.042	0.528	0.513
	720	**0.501**	**0.504**	0.710	0.616	0.713	0.599	0.881	0.678	1.239	0.811	0.597	0.571
ETTh2	168	0.393	**0.420**	0.483	0.450	**0.387**	0.423	0.466	0.459	2.125	1.064	0.554	0.517
	336	0.390	**0.428**	0.516	0.491	**0.377**	**0.428**	0.496	0.493	1.923	0.956	0.657	0.576
	720	**0.495**	**0.510**	0.509	0.512	0.537	0.539	0.538	0.535	1.504	0.879	1.118	0.776
ETTm1	96	**0.190**	**0.279**	0.316	0.352	0.266	0.339	0.248	0.351	0.550	0.532	0.197	0.294
	288	**0.306**	**0.354**	0.369	0.398	0.354	0.392	0.318	0.384	0.422	0.442	0.350	0.405
	672	**0.376**	**0.395**	0.538	0.486	0.412	0.428	0.419	0.445	0.454	0.455	1.214	0.836
ETTm2	96	**0.315**	**0.345**	0.489	0.436	0.484	0.438	0.909	0.609	0.861	0.639	0.330	0.377
	288	0.416	**0.402**	0.547	0.493	0.607	0.501	1.144	0.721	0.754	0.593	**0.383**	0.408
	672	**0.490**	**0.455**	0.893	0.643	0.677	0.525	0.743	0.541	0.780	0.590	0.501	0.490
Births	168	**0.231**	**0.345**	0.702	0.662	1.735	1.151	2.299	1.310	3.828	1.651	0.268	0.376
	336	**0.611**	**0.602**	1.126	0.896	2.835	1.379	5.784	1.957	6.077	2.002	0.865	0.737
	720	**0.567**	**0.581**	1.488	0.982	2.845	1.377	4.640	1.750	3.729	1.558	1.775	1.140
ILI	36	**3.498**	**1.278**	3.826	1.378	4.915	1.572	5.539	1.705	9.574	2.331	4.089	1.400
	48	**3.531**	**1.297**	3.878	1.395	4.939	1.586	5.071	1.621	8.809	2.242	4.076	1.401
	60	**3.598**	**1.329**	4.183	1.467	5.037	1.618	5.069	1.639	9.321	2.329	4.216	1.452

Table 3: MSE and MAE errors of SCINet trained on the generated data generated by GANs (lower is better, best results are in bold). Each dataset has experimented with three different forecasting horizons (as shown in the second column).

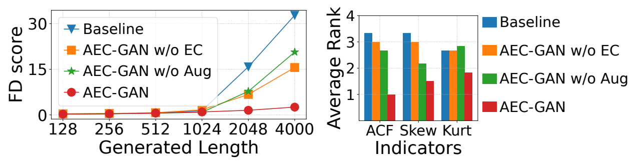


Figure 5: (Left) FD scores with different generation lengths on ETTh2. (Right) The performance rankings of these four models over three indicators averaged on six datasets.

ment gradually increases along with the longer forecasting horizons, indicating our excellent performance in the long sequence generation. Moreover, AEC-GAN obtains an average 17.9% promotion regarding the forecasting errors on three forecasting models. Furthermore, GANs can generate an arbitrary amount of data. With the increasing amount of data, the performance of the forecasting model can be further improved and the results are shown in the appendix.

5.4 Experiment 3: Ablation Study

As aforementioned, AEC-GAN mainly gains from two innovations: the error correction module and the augmentation method. Hence, we study the effects of these two components on the generation quality. We remove the error correction module in the testing phase, denoted as **AEC-GAN w/o EC**. We remove the augmented data in training, i.e., $\hat{x}^2_\tau, \hat{x}^3_\tau, \hat{x}^4_\tau$, denoted as **AEC-GAN w/o Aug**. We denote the vallina GAN as **Baseline**. Fig. 5 shows the performance of the above three models. The poor performance of AEC-GAN w/o EC in the long sequence generation (4000 steps) demonstrates the effectiveness of the error correction module. In addition, the augmentation also decreases the FD score and improves the generation quality on the statistical

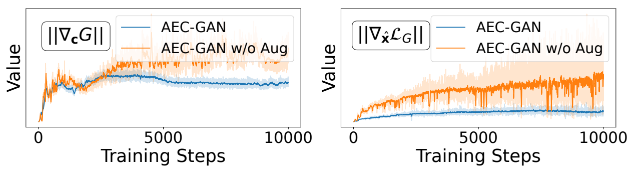


Figure 6: Values of $\|\nabla_c G\|$ and $\|\nabla_{\hat{x}} \mathcal{L}_G\|$ during the training of ILI. $\|\cdot\|$ denotes the l_2 norm.

measures (ACF, Skewness and Kurtosis).

Furthermore, we study how the augmentation method improves generation quality. In inequality (8), smaller $\|\nabla_c G\|$ and $\|\nabla_{\hat{x}} \mathcal{L}_G\|$ can reduce the gap between the training and testing phases. In Fig. 6, we record these two gradients during the training of ILI. It shows that the augmentation method significantly reduces both $\|\nabla_c G\|$ and $\|\nabla_{\hat{x}} \mathcal{L}_G\|$, stabilizes the GAN's training, and improves the long sequence generation.

6 Conlusion

In this paper, we present a novel method called AEC-GAN (Adversarial Error Correction GAN) to generate more realistic time-series data with arbitrary length. In AEC-GAN, we propose a generic error correction module to mitigate the distribution shift and an augmentation method to facilitate the GAN's training. Experimental results have demonstrated our superior performance over other deep generative models for time-series generation on six datasets and three forecasting models. AEC-GAN provides a promising approach to generating large-scale high-quality data, and how to employ these data to facilitate the downstream tasks still deserves to be explored.

Acknowledgments

The authors are supported in part by the National Natural Science Foundation of China Grant 62161146004, Turing AI Institute of Nanjing and Xi'an Institute for Interdisciplinary Information Core Technology.

References

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

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

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

Centers for Disease Control and Prevention. 2020. National, Regional, and State Level Outpatient Illness and Viral Surveillance. https://gis.cdc.gov/grasp/fluview/fluportaldashboard.html. Accessed: 2020-06-30.

Chakraborty, A.; Alam, M.; Dey, V.; Chattopadhyay, A.; and Mukhopadhyay, D. 2018. Adversarial attacks and defences: A survey. *arXiv preprint arXiv:1810.00069*.

Ding, Q.; Wu, S.; Sun, H.; Guo, J.; and Guo, J. 2020. Hierarchical Multi-Scale Gaussian Transformer for Stock Movement Prediction. In *IJCAI*, 4640–4646.

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

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

Franceschi, J.-Y.; Dieuleveut, A.; and Jaggi, M. 2019. Unsupervised scalable representation learning for multivariate time series. *Advances in neural information processing systems*, 32.

Fréchet, M. 1957. Sur la distance de deux lois de probabilité. *Comptes Rendus Hebdomadaires des Seances de L Academie des Sciences*, 244(6): 689–692.

Godahewa, R.; Bergmeir, C.; Webb, G. I.; Hyndman, R. J.; and Montero-Manso, P. 2021. Monash time series forecasting archive. *arXiv preprint arXiv:2105.06643*.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. *Advances in neural information processing systems*, 27.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Hamilton, J. D. 2020. *Time series analysis*. Princeton university press.

Hazra, D.; and Byun, Y.-C. 2020. SynSigGAN: Generative adversarial networks for synthetic biomedical signal generation. *Biology*, 9(12): 441.

Jarrett, D.; Bica, I.; and van der Schaar, M. 2021. Time-series generation by contrastive imitation. *Advances in Neural Information Processing Systems*, 34: 28968–28982.

Jiang, L.; Dai, B.; Wu, W.; and Loy, C. C. 2021. Deceive D: Adaptive pseudo augmentation for gan training with limited data. *Advances in Neural Information Processing Systems*, 34: 21655–21667.

Kapoor, A.; Ben, X.; Liu, L.; Perozzi, B.; Barnes, M.; Blais, M.; and O'Banion, S. 2020. Examining covid-19 forecasting using spatio-temporal graph neural networks. *arXiv preprint arXiv:2007.03113*.

Karras, T.; Aittala, M.; Hellsten, J.; Laine, S.; Lehtinen, J.; and Aila, T. 2020a. Training generative adversarial networks with limited data. *Advances in Neural Information Processing Systems*, 33: 12104–12114.

Karras, T.; Aittala, M.; Laine, S.; Härkönen, E.; Hellsten, J.; Lehtinen, J.; and Aila, T. 2021. Alias-free generative adversarial networks. *Advances in Neural Information Processing Systems*, 34: 852–863.

Karras, T.; Laine, S.; Aittala, M.; Hellsten, J.; Lehtinen, J.; and Aila, T. 2020b. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 8110–8119.

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

Lim, B.; and Zohren, S. 2021. Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A*, 379(2194): 20200209.

Liu, M.; Zeng, A.; Xu, Z.; Lai, Q.; and Xu, Q. 2021. Time series is a special sequence: Forecasting with sample convolution and interaction. *arXiv preprint arXiv:2106.09305*.

Liu, X.; and Hsieh, C.-J. 2019. Rob-gan: Generator, discriminator, and adversarial attacker. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11234–11243.

Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2017. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*.

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

Mirza, M.; and Osindero, S. 2014. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*.

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

Paul, J.; Michael, B.-S.; Pedro, M.; Rajbir, S. N.; Shubham, K.; Valentin, F.; Jan, G.; and Tim, J. 2021. PSA-GAN: Progressive Self Attention GANs for Synthetic Time Series. *arXiv preprint arXiv:2108.00981*.

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; and Fergus, R. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

Tran, N.-T.; Tran, V.-H.; Nguyen, N.-B.; Nguyen, T.-K.; and Cheung, N.-M. 2021. On data augmentation for gan training. *IEEE Transactions on Image Processing*, 30: 1882–1897.

Van der Maaten, L.; and Hinton, G. 2008. Visualizing data using t-SNE. *Journal of machine learning research*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

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

Wu, H.; Xu, J.; Wang, J.; and Long, M. 2021. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in Neural Information Processing Systems*, 34: 22419–22430.

Xiao, C.; Li, B.; Zhu, J.-Y.; He, W.; Liu, M.; and Song, D. 2018. Generating adversarial examples with adversarial networks. *arXiv preprint arXiv:1801.02610*.

Xu, T.; Wenliang, L. K.; Munn, M.; and Acciaio, B. 2020. Cot-gan: Generating sequential data via causal optimal transport. *Advances in Neural Information Processing Systems*, 33: 8798–8809.

Yoon, J.; Jarrett, D.; and Van der Schaar, M. 2019. Time-series generative adversarial networks. *Advances in neural information processing systems*, 32.

Zhou, H.; Zhang, S.; Peng, J.; Zhang, S.; Li, J.; Xiong, H.; and Zhang, W. 2021. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 11106–11115.