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)).

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

Our goal is to generate high-quality time-series data with arbitrary lengths given a short context \(c_t\) (context length is \(p\)) sampled from history data. We develop a conditional GAN (modeling \(P(x|c)\)), which contains a generator \(G_0\) and a discriminator \(D_0\) parameterized by \(\theta\) and \(\phi\) respectively. In the training phase, given the context \(c_t = x_{t-p+1:t}\), the GAN model is optimized to generate a time series of the target length \(q\), i.e., \(x_{t+1:t+q}\) (we use \(^\top\) to denote the generated data). Specifically, the one-step generator \(G_0\)
first transforms a Gaussian noise vector \( z_{t+1} \in \mathbb{R}^d \) (drawn from \( \mathcal{N}(0, I_{d \times d}) \)) into the a generated data point \( \hat{x}_{t+1} \) given the condition \( c_t \), i.e., \( \hat{x}_{t+1} = \hat{G}_\theta(z_{t+1}|c_t) \in \mathbb{R}^d \). Then \( G_\theta \) generate \( \hat{x}_{t+2} = \hat{G}_\theta(z_{t+2}|\hat{c}_{t+1}) \) based on the updated condition \( \hat{c}_{t+1} \), where \( \hat{c}_{t+1} = [x_{t-p+2:t}, \hat{x}_{t+1}] \). Auto-regressively, we obtain the target sequence \( \hat{x}_{t+1:t+q} = [\hat{G}_\theta(z_{t+1}|c_t), ..., \hat{G}_\theta(z_{t+q}|\hat{c}_{t+q-1})] \). For brevity, we overload the notation \( \hat{G}_\theta \) to denote the generations \( \hat{x}_{t+1:t+q} \) as \( \hat{G}_\theta(z_{t+1:t+q}|c_t) \). The discriminator \( D_\phi \) is optimized to distinguish generated sequence \( \hat{x}_{t+1:t+q} \) from real sequence \( x_{t+1:t+q} \), given the condition \( c_t \). For more realistic data, \( D_\phi(c_t, \hat{c}_t) : \mathbb{R}^{d \times x} \times \mathbb{R}^{d \times p} \rightarrow \mathbb{R} \) is desired to give a higher discriminant score. In the rest of the paper, we omit the subscripts of \( z \) unless required. The GAN’s objective function (Mirza and Osindero 2014) is shown in function (1). After training, \( G_\theta \) can generate sequences step by step after feeding in a context \( c_t \), without generation length limitation.

\[
\min_{\theta} \max_{\phi} \mathbb{E}_{x_{t+1:t+q} \sim \mathcal{P}(x|c_t)} \left[ \log D_\phi(x_{t+1:t+q}, c_t) \right] + \mathbb{E}_{x_{t+1:t+q} \sim \mathcal{P}(x|c_t)} \left[ \log \left(1 - D_\phi(\hat{G}_\theta(z_{t+1}|c_t), c_t) \right) \right].
\]  

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), the KL-divergence between the \( L \)th step generation and the underlying distribution is \( \Omega(\delta L) \), where \( L \) is generation length and \( \delta \) 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, \( x_t = \Phi x_{t-1} + \alpha_t \), where \( \Phi \in \mathbb{R}^{d \times d} \) is symmetric and \( \alpha_t \sim \mathcal{N}(0, I) \). We denote \( \hat{x}_t = \hat{\Phi} \hat{x}_{t-1} + \hat{\alpha}_t \) as the estimated generation process, where \( \hat{\Phi} \in \mathbb{R}^{d \times d} \) is also symmetric and \( \hat{\alpha}_t \sim \mathcal{N}(0, I) \). The max eigenvalues of \( \Phi \) and \( \hat{\Phi} \) are denoted as \( \lambda \) and \( \hat{\lambda} \), and they differ by a small quantity \( \delta \), i.e., \( |1 - \hat{\lambda}/\lambda| = \delta \). Generating \( L \) steps time-series data from \( x_0 \), if \( \lambda \geq 1 \), then

\[
D_{K\!L}(p(x_L|x_0)||p(\hat{x}_L|x_0)) = \Omega(\delta L),
\]

where \( D_{K\!L}(\cdot || \cdot) \) 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 ameliorate this problem.

4.2 Error Correction Module

In this section, we introduce an error correction module \( M_\omega(\cdot) : \mathbb{R}^{d \times p} \rightarrow \mathbb{R}^{d \times p} \), parameterized by \( \omega \), to mitigate the bias in the past generated sequences. Given the condition \( c_t \in \mathbb{R}^{d \times p} \sim \mathcal{P}(c) \) to generate time-series data beginning from the time \( t \), the generator \( G_\theta \) first generates a \( p \) step sequence \( \hat{x}_{t+1:t+p} \) in a rolling way, which is regarded as the next condition \( \hat{c}_{t+p} \) for the following generation. \( M_\omega \) modifies the condition \( \hat{c}_{t+p} \) with \( c_t + M_\omega(\hat{c}_{t+p}) \). Then the generator generates the following \( p \) step sequence \( \hat{x}_{t+1:t+p+1|t+2} \) given the refined condition \( \hat{c}_{t+p} + M_\omega(\hat{c}_{t+p}) \). After this, \( M_\omega \) modifies the new condition \( \hat{c}_{t+p} \) (i.e., \( \hat{x}_{t+p+1|t+2} \)) 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, \( M_w \) modifies the generated condition \( \hat{x}_{t+1:q} \) with \( \epsilon_{t+1:q} + M_w(\hat{x}_{t+1:q}) \) 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 \( \hat{x}_{t+1:q} + M_w(\hat{x}_{t+1:q}) \), where \( k \in \mathbb{Z}^+ \). In such a way, \( M_w \) mitigates the bias adaptively. Finally, we gather the generated \( \hat{x}_{t+1:1+q} \) as the final output denoted by \( G^M_\omega(z|c_t) \). The generation process with correction is detailed in Algorithm 1 and illustrated in Fig. 2.

To optimize the module \( M_w \), we perturb the realistic condition \( c_t \) with a slight perturbation \( \delta \) and expect \( M_w(c_t + \delta) \) to remove the perturbation. Specifically, \( \delta \) is created from the PCD-attack (Madry et al. 2017) within an \( l_2 \) radius \( \delta_{\text{max}} \), as shown in Eq. (3). More discussions about the kinds of perturbations can be found in the appendix.

\[
\delta = \arg \min_{||\delta||_2 \leq \delta_{\text{max}}} \log \left[ 1 - D_\theta(G_\theta(z|c_t), c_t + \delta) \right]. \tag{3}
\]

Besides modifying the adversarial example (i.e., \( c_t + \delta \)), \( M_w(c_t) \) is also supposed to output \( 0 \) for realistic \( c_t \). Hence, the objective function \( \min_{\omega} \mathcal{L}_M(\omega) \) for \( M_w \) is as follows:

\[
\min_{\omega} \mathbb{E}_{c_t \sim P(c_t)} \mathcal{L}_2(M_w(c_t + \delta), -\delta) + \mathcal{L}_2(M_w(c_t), 0), \tag{4}
\]

where \( \mathcal{L}_2(\cdot, \cdot) \) denotes the MSE loss. With \( M_w \) 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, \( G_\theta \) only generates sequences with a target length \( q \), as shown in the objective function (1). In the testing phase, \( G_\theta \) 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., \( G_\theta(z|c_t) \in \mathbb{R}^{d\times p} \). In the testing phase, suppose we are required to generate sequences with length \( Kp \), given \( c_{kp} \sim P(c_{kp}) \) (i.e., \( x_{1:kp} \)). Concretely, the generator \( G_\theta \) first generates a p-step sequence \( c_{2p} \) (i.e., \( \hat{x}_{p+1:2p} \)) based on \( c_p \) and then auto-regressively generates \( c_{kp+1} \) (i.e., \( \hat{x}_{kp+1:kp+2p} \)) based on the refined generated data \( \epsilon_{kp} + M_w(\epsilon_{kp}) \), where \( 2 \leq k \leq K \). To assess the generation quality, we first assume that the realistic sequences are \( c_{kp+p} \sim P(x_{kp+1:kp+p+1}) \) (note that \( c_p \sim P(c_p) \)). Then we calculate the generator’s loss \( \mathcal{L}_{\text{test}}(\theta, \phi, \omega) \) for the generated sequence, i.e., the loss of the discriminator’s judgement about whether the generated sequence \( x_{kp+1:kp+p} \sim G_\theta(z|\epsilon_{kp} + M_w(\epsilon_{kp})) \) is drawn from the ground-truth \( P(x_{kp+1:kp+p+1}|c_p) \), as follows:

\[
\mathbb{E}_{z \sim N(0, I)} \sum_{k=2}^K \log \left[ 1 - D_\phi(G_\theta(z|\epsilon_{kp} + M_w(\epsilon_{kp})), c_{kp}) \right]. \tag{5}
\]

We denote \( \delta_k = \epsilon_{kp} - c_{kp} \) (related to \( z \) and \( c_{kp} \)) as the generation bias. Then, the generation part of the equation (5) can be rewritten as follows:

\[
G_\theta(z|\epsilon_{kp} + \delta_k + M_w(\epsilon_{kp} + \delta_k)) \tag{6}
\]

**Algorithm 1: Auto-Regressive Generation**

**Input:** Condition \( c_t \in \mathbb{R}^{d\times q} \), generation length \( q \), generator \( G_\theta \), error-correction module \( M_w \).

**Output:** \( \hat{x}_{t+1:1+q} \in \mathbb{R}^{d\times q} \).

1. **Init**: \( \hat{x}_t = c_t \).
2. **for** \( i = 0,...,q-1 \) **do**
   3. **if** \( i + 1 \equiv q \) **then**
   4. \( \hat{x}_{t+i} \leftarrow \hat{x}_{t+i} + M_w(\hat{x}_{t+i}) \)
   5. \( z_{t+i+1} \leftarrow N(0, I) \in \mathbb{R}^d \)
   6. \( \hat{x}_{t+i+1} \leftarrow \text{pop}(\hat{x}_{t+i}) \cup \hat{x}_{t+i+1} \cdot \text{pop}(\cdot) \) **means**
   7. **remove** the earliest step.
3. **Return**: \( \hat{x}_{t+1:1+q} = [\hat{x}_{t+1}, \ldots, \hat{x}_{t+q}] \).

For brevity, we let \( \epsilon_{kp} = \delta_k + M_w(\epsilon_{kp} + \delta_k) \). With a powerful \( M_w \) to mitigate the bias, \( \epsilon_{kp} \) is small, then with Taylor expansion, we get the following upper bound:

\[
\mathcal{L}_{\text{test}} = \mathbb{E}_{z \sim N(0, I)} \sum_{k=2}^K \log \left[ 1 - D_\phi(G_\theta(z|\epsilon_{kp} + \epsilon_{kp}), c_{kp}) \right] \leq \mathbb{E}_{z \sim N(0, I)} \sum_{k=2}^K \left\{ \log \left[ 1 - D_\phi(G_\theta(z|\epsilon_{kp}), c_{kp}) \right] + \right. \frac{\left| \nabla_\theta \mathcal{L}_{\phi}(\theta, \phi, \omega) \nabla_\phi \mathcal{L}_{\phi}(\theta, \phi, \omega) \right|}{\nabla_\theta \mathcal{L}_{\phi}(\theta, \phi, \omega)} + O(||\epsilon_{kp}||^2) \}\right\}. \tag{7}
\]

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

\[
\mathcal{L}_{\text{test}} \leq \mathbb{E}_{z \sim N(0, I)} \sum_{k=2}^K \left\{ \mathcal{L}_G(z, c_{kp}; \theta, \phi) + \frac{\|\nabla_\theta \mathcal{L}_G\| \cdot \|\nabla_\phi \mathcal{L}_G\| \cdot ||\epsilon_{kp}|| + O(||\epsilon_{kp}||^2) \right\}. \tag{8}
\]

From above, \( \mathcal{L}_{\text{test}} \) is bounded by the training loss and some terms related to \( ||\epsilon_{kp}|| \), and we explicitly minimize the \( ||\epsilon_{kp}|| \) in the objective function (4). With the module \( M_w \), we can optimize the upper bound of the loss \( \mathcal{L}_{\text{test}} \) to generate longer sequences with high quality, rather than being limited to the length specified in training. In addition, the gradients \( \|\nabla_\phi \mathcal{L}_G\| \) and \( \|\nabla_\theta \mathcal{L}_G\| \) 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 \( \hat{x} \) = \( G_\theta(z|c_t) \) to mislead \( D_\phi \), where \( c_t \) 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., \( c_t + \delta \)) and \( M_w \), detailed as in Eq. (9).

\[
\begin{align*}
\hat{x}_1^1 &= G_\theta(z_t|c_t) \quad \hat{x}_r^1 &= G_\theta(z_t|c_t + \delta) \\
\hat{x}_3^1 &= c_{\theta}^1(z_t|c_t) \quad \hat{x}_r^3 &= c_{\theta}^1(z_t|c_t + \delta).
\end{align*} \tag{9}
\]
Among these data, $\hat{x}^1_t$ denotes the generated data optimized in inequality (8), $\hat{x}^2_t$ and $\hat{x}^3_t$ encourage the $G^i_t$ to resist perturbation (i.e., $\delta$) in the condition, and $\hat{x}^3_t$ denotes the generation in the testing phase. The objective function for $G^\theta_t$, i.e., $\min_\theta L_{G^A}^{AUG} (\theta, \phi, \omega)$, consists of four parts as follows:

$$
\begin{align*}
\mathbb{E}_{x_t^r \sim \mathcal{P}(x^r)} \left[ \log \left(1 - D^\phi (\hat{x}^1_t, c_t)\right) + \log \left(1 - D^\phi (\hat{x}^3_t, c_t)\right) \right]
\end{align*}
$$

$$
\begin{align*}
\mathbb{E}_{x_t^r \sim \mathcal{P}(x^r)} \left[ \log \left(1 - D^\phi (\hat{x}^2_t + \delta, c_t + \delta)\right) + \log \left(1 - D^\phi (\hat{x}^3_t, c_t + \delta)\right) \right].
\end{align*}
$$

Augmented with these generations, the objective function $\max_\phi L_D^{AUG} (\theta, \phi, \omega)$ for $D^\phi$ can be formulated as follows:

$$
\begin{align*}
\max_\phi \mathbb{E}_{x_t^r \sim \mathcal{P}(x^r)} \left[ \log D^\phi (x_t^r, c_t) \right] + \mathcal{L}_{G^{AUG}}^\phi (\theta, \phi, \omega). \quad (11)
\end{align*}
$$

It is worth noting that the slight perturbation $\delta$ 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 $c_t + \delta$ acts as a regularizer to prevent overfitting, as shown in Eq. (11).

In the training phase, we jointly optimize $G^\theta_t$, $D^\phi$ and $M_\omega$. After training, $G^M_t$ can generate sequences with arbitrary lengths by feeding in a context $c_t$ 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 $G^\theta_t$ and discriminator $D^\phi$, which use the past sequences as conditioning variables. For error correction module $M_\omega$, 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: ETT1h, ETT2h, 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 ETT* 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:

$$
\text{ACF}(k) = \frac{\text{Cov}(x_t, x_{t-k})}{\text{Var}(x_t)}, 1 \leq k \leq 100.
$$

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

$$
\text{Skew} = \mathbb{E}\left[\frac{(x_t - \mu_x)^3}{\sigma_x}\right], \quad \text{Kurt} = \mathbb{E}\left[\frac{(x_t - \mu_x)^4}{\sigma_x^4}\right]. \quad (13)
$$

where $\mu_x = \mathbb{E}[x_t]$, $\sigma_x = \sqrt{\text{Var}(x_t)}$.

- **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., $(M_r, C_r)$ and $(M_s, C_s)$, FD score is obtained as follows:

$$
\text{FD} = \|M_r - M_s\|^2 + \text{Tr}(C_r + C_s - 2(C_r C_s)^{1/2}). \quad (14)
$$

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.

<table>
<thead>
<tr>
<th></th>
<th>ETT1h</th>
<th>ETT2h</th>
<th>US Births</th>
<th>ILI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Methods</td>
<td>ACF</td>
<td>Skew</td>
<td>Kurt</td>
<td>ACF</td>
</tr>
<tr>
<td>QuantGAN</td>
<td>0.403</td>
<td>4.370</td>
<td>8.323</td>
<td>0.742</td>
</tr>
<tr>
<td>TimeGAN</td>
<td>0.636</td>
<td>2.596</td>
<td>2.103</td>
<td>0.169</td>
</tr>
<tr>
<td>Cot-GAN</td>
<td>0.153</td>
<td>2.082</td>
<td>7.451</td>
<td>0.157</td>
</tr>
<tr>
<td>SigCWGAN</td>
<td>0.155</td>
<td>3.021</td>
<td>7.081</td>
<td>0.142</td>
</tr>
<tr>
<td>AEC-GAN</td>
<td><strong>0.055</strong></td>
<td><strong>1.911</strong></td>
<td><strong>0.714</strong></td>
<td><strong>0.115</strong></td>
</tr>
</tbody>
</table>
Table 2: FD scores with different generation lengths.

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

Table 3: FD scores.

5.2 Experiment 1: Generation Quality

For each GAN, 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.

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-
Table 3: MSE and MAE errors of SCIConv trained on the generated data 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).

<table>
<thead>
<tr>
<th>Models</th>
<th>AEC-GAN</th>
<th>SigCWGAN</th>
<th>Cot-GAN</th>
<th>TimeGAN</th>
<th>QuantGAN</th>
<th>Original</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metric</td>
<td>MSE</td>
<td>MAE</td>
<td>MSE</td>
<td>MAE</td>
<td>MSE</td>
<td>MAE</td>
</tr>
<tr>
<td>ETTh1</td>
<td>168</td>
<td>0.403</td>
<td>0.342</td>
<td>0.519</td>
<td>0.497</td>
<td>0.465</td>
</tr>
<tr>
<td></td>
<td>336</td>
<td>0.498</td>
<td>0.496</td>
<td>0.627</td>
<td>0.650</td>
<td>0.556</td>
</tr>
<tr>
<td></td>
<td>720</td>
<td>0.501</td>
<td>0.504</td>
<td>0.710</td>
<td>0.713</td>
<td>0.599</td>
</tr>
<tr>
<td>ETTh2</td>
<td>168</td>
<td>0.393</td>
<td>0.420</td>
<td>0.483</td>
<td>0.387</td>
<td>0.423</td>
</tr>
<tr>
<td></td>
<td>336</td>
<td>0.390</td>
<td>0.428</td>
<td>0.516</td>
<td>0.377</td>
<td>0.428</td>
</tr>
<tr>
<td></td>
<td>720</td>
<td>0.495</td>
<td>0.510</td>
<td>0.509</td>
<td>0.537</td>
<td>0.539</td>
</tr>
<tr>
<td>ETTh3</td>
<td>168</td>
<td>0.190</td>
<td>0.279</td>
<td>0.316</td>
<td>0.369</td>
<td>0.398</td>
</tr>
<tr>
<td></td>
<td>336</td>
<td>0.306</td>
<td>0.354</td>
<td>0.369</td>
<td>0.354</td>
<td>0.392</td>
</tr>
<tr>
<td></td>
<td>720</td>
<td>0.376</td>
<td>0.395</td>
<td>0.369</td>
<td>0.354</td>
<td>0.392</td>
</tr>
<tr>
<td>Births</td>
<td>168</td>
<td>0.315</td>
<td>0.345</td>
<td>0.489</td>
<td>0.484</td>
<td>0.438</td>
</tr>
<tr>
<td></td>
<td>336</td>
<td>0.416</td>
<td>0.402</td>
<td>0.547</td>
<td>0.607</td>
<td>0.501</td>
</tr>
<tr>
<td></td>
<td>720</td>
<td>0.490</td>
<td>0.455</td>
<td>0.893</td>
<td>0.677</td>
<td>0.525</td>
</tr>
<tr>
<td>ILL</td>
<td>36</td>
<td>3.498</td>
<td>1.278</td>
<td>3.826</td>
<td>4.915</td>
<td>1.572</td>
</tr>
<tr>
<td></td>
<td>48</td>
<td>3.531</td>
<td>1.297</td>
<td>3.878</td>
<td>4.939</td>
<td>1.586</td>
</tr>
</tbody>
</table>

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. The performance 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., $\hat{x}_t$, $\hat{x}_{t+1}$, $\hat{x}_{t+2}$, denoted as AEC-GAN w/o Aug. We denote the vanilla 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 measures (ACF, Skewness and Kurtosis).

Furthermore, we study how the augmentation method improves generation quality. In inequality (8), smaller $||\nabla_cG||$ and $||\nabla_LC||$ can reduce the gap between the training and testing phases. In Fig. 6, we record these two gradients during the training of ILL. $||\cdot||$ denotes the $l_2$ norm.

Figure 6: Values of $||\nabla_cG||$ and $||\nabla_LC||$ during the training of ILL. $||\cdot||$ denotes the $l_2$ norm.

6 Conclusion

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


