Long-Term Multivariate Time Series Generation with the Capture of Intervariate Dependencies and Variatewise Characteristics

Kasumi Ohno, **Kohei Makino**, **Makoto Miwa** and **Yutaka Sasaki**
Toyota Technological Institute, Nagoya, Japan
{makoto-miwa, yutaka.sasaki}@toyota-ti.ac.jp

Abstract

This paper proposes a novel *Time Series Generation (TSG)* model named *Attended Variate-Conditioned GAN (AVC-GAN)* designed for generating multivariate long-term time series data. Recently, generative approaches to TSG have been explored for applications such as privacy protection, anomaly detection, and time series classification/forecasting. Conventional TSG methods, particularly *Generative Adversarial Networks (GANs)*-based methods, have evolved in modeling long-term time series and capturing intervariate dependencies; however, in addition to these points, a further avenue for exploration lies in the adequate capture of the characteristics of each variate. To address these issues on TSG simultaneously, we propose a novel generative model, AVC-GAN. The key components of AVC-GAN are as follows: 1) a nonautoregressive architecture for modeling long-term time series, 2) variate conditioning for capturing variatewise characteristics, and 3) a multihead attention mechanism for capturing intervariate dependencies. Our experiments on multiple benchmark datasets for *Long-Term Time Series Forecasting (LTSF)* demonstrate that overall, AVC-GAN outperforms state-of-the-art GAN-based generative models across a range of evaluation metrics. In terms of distance-based metrics, calculated as the MSE between real and generated data, our proposed model achieved improvements of 29.8% and 33.7% over state-of-the-art methods in assessing overall intervariate dependencies and variatewise characteristics, respectively.

1 Introduction

TSG involves the creation of synthetic time series data that exhibit the characteristics of real-world time series data. Synthetic data generated by TSG are used in various applications, such as in privacy protection as substitute data for privacy-sensitive data [Qian *et al.*, 2024], in time series anomaly detection as compensating data for anomaly data [Allam *et al.*, 2024], and in time series classification and forecasting as augmenting data for training data [Semenoglou *et al.*, 2023]. Since such applications require high-quality synthetic data, TSG models must accurately imitate the characteristics of real-world time series data.

Recently, deep learning-based generative models, such as *Variational AutoEncoder (VAE)* [Kingma and Welling, 2022]-based models [Desai *et al.*, 2021], GAN [Goodfellow *et al.*, 2014]-based models [Yoon *et al.*, 2019; Pei *et al.*, 2021; Seyfi *et al.*, 2022; Wang *et al.*, 2023], and diffusion [Ho *et al.*, 2020]-based models [Rasul *et al.*, 2021; Alcaraz and Strodthoff, 2023], have been actively studied. Among them, GAN-based models have been the most actively studied because they have empirically demonstrated superior performance in generating realistic time series data [Ang *et al.*, 2024].

Addressing the challenges of *Long-Term Time Series Generation (LTSG)* has been a focus of recent research on GAN-based generative models, particularly for applications that require long-term synthetic data. However, LTSG suffers from the problem of large cumulative errors in generating long-term time series because of the autoregressive architecture of the generation model [Wang *et al.*, 2023]. A key limitation of autoregressive models is their susceptibility to error accumulation during long-term generation, where small errors at each step can propagate and amplify over time. While *Adversarial Error Correction GAN (AEC-GAN)* [Wang *et al.*, 2023] addressed this issue by incorporating a dynamic error correction mechanism within its generator, nonautoregressive approaches have also been explored in the context of LTSF [Zhou *et al.*, 2021; Liu *et al.*, 2024] to alleviate this problem.

Another crucial aspect of multivariate time series generation is the accurate modeling of intervariate dependencies. *COmmon Source CoordInated GAN (COSCI-GAN)* [Seyfi *et al.*, 2022] attempts to capture these correlations by using a generator for each variate and a single discriminator for all variates. This architecture facilitates the modeling of dependencies between variates, thereby enabling the generation of multivariate data that capture intervariate dependencies.

The contributions of this study are as follows:

- We propose a novel TSG model, AVC-GAN, which captures intervariate dependencies via a multihead attention mechanism, preserves variatewise characteristics through variate conditioning, and models long-term time series via a nonautoregressive autoencoder.

- The results of our experiments show that AVC-GAN can generate long-term multivariate time series that are more similar to real data than those generated by existing state-of-the-art methods without compromising the overall performance.

2 Related Work

2.1 Time Series Generation

We let $\mathcal{D} = \{\boldsymbol{X}^{(1)}, \boldsymbol{X}^{(2)}, \ldots, \boldsymbol{X}^{(M)}\}$ be the observed dataset of multivariate time series. Each time series $\boldsymbol{X}^{(i)} \in \mathbb{R}^{N \times L}$ is a matrix where N is the number of variates and L is the timestep length. $\boldsymbol{X}^{(i)} = [\boldsymbol{x}_1^{(i)}, \ldots, \boldsymbol{x}_L^{(i)}]$, where $\boldsymbol{x}_t^{(i)} \in \mathbb{R}^{N \times 1}$ denotes the N-dimensional observation vector at time step t. Unless otherwise stated, we use $\boldsymbol{X}$ to denote a generic time series $\boldsymbol{X}^{(i)}$ throughout this paper. The goal of TSG is to create a generative model that can produce a new dataset $\hat{\mathcal{D}}$ of synthetic time series similar to $\mathcal{D}$.

Whereas traditional statistical models for TSG often fall short in capturing the complex, nonlinear dependencies present in real-world data [Hu *et al.*, 2023], deep learning-based generative models have shown significant promise in addressing this challenge. These models, which include VAEs, GANs, and diffusion models, offer the ability to learn complex data distributions and generate diverse, high-quality synthetic time series. One approach which adopted VAEs for TSG is TimeVAE [Desai *et al.*, 2021], which introduces a recurrent decoder to capture temporal dynamics and learn a more structured latent representation. TimeGrad [Rasul *et al.*, 2021] is a diffusion-based model for TSG that follows the standard diffusion model framework.

2.2 GAN-Based Time Series Generation

GANs models are a class of deep learning models designed to generate new data samples that follow the same distribution as a given training dataset. GANs models involve training two neural networks, namely, a generator G and a discriminator D, in an adversarial manner. The generator takes a random noise vector $\boldsymbol{z}$ (typically sampled from a Gaussian distribution) as input and transforms it into a data sample $G(\boldsymbol{z})$. The generator's objective is to generate samples that are indistinguishable from the original data samples. To train the generator, the discriminator is trained to classify the original and generated samples correctly. The discriminator takes a data sample $\boldsymbol{x}$ as input (either an original sample from the training data or a generated sample from G) and outputs a scalar value $D(\boldsymbol{x})$ that represents the probability that $\boldsymbol{x}$ is real. However, training GANs can be challenging because of instability and mode collapse. *Wasserstein GAN with Gradient Penalty (WGAN-GP)* [Gulrajani *et al.*, 2017] addresses these issues by using a *critic* instead of a discriminator and enforcing a Lipschitz constraint on the critic via a gradient penalty, thus stabilizing the training process.

The application of GANs models to TSG presents unique challenges, including the capture of intervariate dependencies in multivariate time series and the generation of long-term time series that preserve temporal dependencies. Several GAN-based approaches have been proposed to address these issues.

For example, to accurately represent a multivariate time series, it is necessary to correctly capture the correlation between variates [Snow, 2020; Li *et al.*, 2022; Seyfi *et al.*, 2022]. COSCI-GAN employs a separate generator and discriminator for each variate and a central discriminator. The individual generator-discriminator pairs focus on generating univariate time series, whereas the central discriminator encourages the intervariate dependencies in the generated data to match those observed in the original data. However, the cumulative error due to the autoregressive mechanism in COSCI-GAN makes generating a long-term time series challenging.

Another challenge in TSG is the generation of long-term time series [Ni *et al.*, 2022; Wiese *et al.*, 2020; Jeha *et al.*, 2022; Li *et al.*, 2022; Wang *et al.*, 2023]. To address this, AEC-GAN uses a dynamic error correction mechanism to alleviate the cumulative error problem. Although it is based on an autoregressive mechanism, this error correction mechanism updates the conditions dynamically to alleviate the problem of drift from the true data distribution during long-term generation.

Another recent approach is to employ a GANs to generate time series not directly but rather latent representations $\hat{\boldsymbol{r}}$ from which time series data are generated by decoding [Yoon *et al.*, 2019; Pei *et al.*, 2021]. We let $G(\boldsymbol{z})$ denote the latent representation generator with input vector $\boldsymbol{z}$ sampled from the prior, and we let $\mathrm{Decoder}(\boldsymbol{r})$ denote the model for decoding. The series $\boldsymbol{X}^{\mathrm{gen}}$ is generated as follows:

$$\hat{\boldsymbol{r}} = G(\boldsymbol{z}) \tag{1}$$

$$\boldsymbol{X}^{\mathrm{gen}} = \mathrm{Decoder}(\hat{\boldsymbol{r}}) \tag{2}$$

Time-series Generative Adversarial Networks (TimeGAN) generates synthetic time-series data by leveraging a pre-trained latent embedding space that is optimized through both adversarial training and supervised temporal prediction. Temporal dependencies are explicitly preserved by jointly learning the encoding, generation, and time-step transitions in the latent space. This method generates sequences from sequences of random vectors, and therefore, the dimensionality of the embedding space is proportional to the sequence length. In contrast to TimeGAN, *Real-World Time Series GAN (RTSGAN)* facilitates the imitation of latent representations by training generators and discriminators in a latent space independent of the sequence length, which is constructed via an autoencoder. However, generating long-term time series remains difficult because the imitated representations are decoded in an autoregressive manner. Moreover, a common limitation of these approaches is the difficulty in explicitly modeling variatewise characteristics.

2.3 Time-Series Forecasting

Advancements in *Time Series Forecasting (TSF)*, which is the task of predicting future time series from past and current time series values, have contributed significantly to the techniques used in TSG to model time series. Given the past and current time series $\boldsymbol{X}_{1:L} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_L]$, the TSF task is to predict a future time series $\boldsymbol{X}_{L+1:L+H} = [\boldsymbol{x}_{L+1}, \ldots, \boldsymbol{x}_{L+H}]$. When $N = 1$, the task is called univariate forecasting, whereas when $N \geq 2$, the task is called multivariate forecasting. LTSF refers to the case where the length of the prediction series $\boldsymbol{X}_{L+1:L+H}$ is long, typically $H \geq 96$ [Zhou *et al.*, 2021].

Recurrent Neural Networks (RNNs) have been widely used for TSF due to their ability to capture sequential dependencies via hidden state transitions. To overcome the vanishing gradient problem of vanilla RNNs, advanced architectures such as *Long Short-Term Memory (LSTM)* [Hochreiter and Schmidhuber, 1997] and *Gated Recurrent Unit (GRU)* [Cho *et al.*, 2014] were introduced, enabling more effective modeling of long-term dependencies [Qin *et al.*, 2017; Liu *et al.*, 2020]. However, these models still suffer from challenges in long-term forecasting due to their sequential processing nature, which limits parallelization and leads to cumulative error propagation across time steps.

Recently, Transformer [Vaswani *et al.*, 2017] has attracted attention in TSF due to its ability to model long-range dependencies by enabling the model to refer to any time point regardless of distance. Nevertheless, the quadratic computational complexity of the original Transformer hinders its direct application to long-term time series data. To address this, the Informer [Zhou *et al.*, 2021] utilizes a ProbSparse self-attention mechanism to reduce the computational cost significantly. Additionally, it introduces a generative-style decoder, which avoids the accumulation of errors by generating the entire predicted sequence in a single forward pass, achieving state-of-the-art performance in long-term series forecasting (LTSF) at the time of its proposal. In recent research, time series modeling using the *multilayer perceptron (MLP)* has become a popular approach due to its superior performance [Zeng *et al.*, 2023; Chen *et al.*, 2023; Liu *et al.*, 2024].

Inverted Transformer (iTransformer) [Liu *et al.*, 2024] was proposed as an effective method for applying Transformer to multivariate LTSF by identifying the time series length as a feature dimension. With an embedding for each variate, which is represented as $\boldsymbol{H}_{\text{inv}} \in \mathbb{R}^{N \times D_H}$, where D_H is the embedding dimension, iTransformer enables the self-attention mechanism to compute an attention map of size $\mathbb{R}^{N \times N}$ that represents the relationships between variates directly. This enables iTransformer to explicitly model these intervariate dependencies. As a result, iTransformer can effectively capture intervariate dependencies while preserving the temporal dynamics within each individual variate, thereby leading to improved performance in multivariate LTSF. In contrast, the original Transformer typically embeds the input sequence $\boldsymbol{X} \in \mathbb{R}^{N \times L}$ along the time dimension, which results in a representation $\boldsymbol{H} \in \mathbb{R}^{D_H \times L}$. The self-attention mechanism then computes attention weights between these time-step embeddings, thereby capturing temporal dependencies.

3 Proposed Model

This section describes the proposed AVC-GAN[1], which addresses the challenges of modeling long-term time series,

[1]We will make the source code publicly available upon acceptance. Note that it is not IJCAI's requirement to release the source code at the submission.

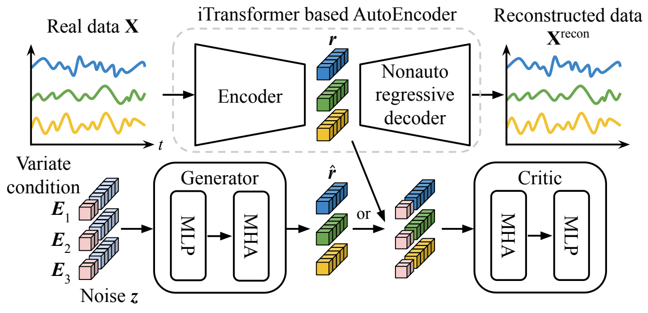

Figure 1: Overview of AVC-GAN. MLP and MHA denote multilayer perceptron and multihead attention, respectively.

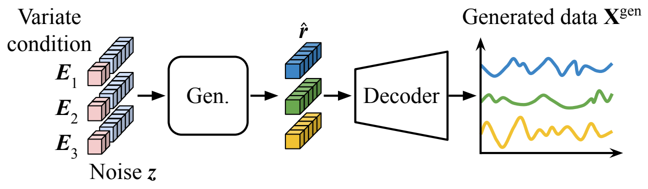

Figure 2: Synthetic data generation with AVC-GAN.

capturing variatewise characteristics, and capturing intervariate dependencies simultaneously. As shown in Figure 1, the main structure of AVC-GAN consists of an iTransformer-based autoencoder and a GAN for generating latent representations on the space of the autoencoder. When generating a series, as shown in Figure 2, the generator produces a latent representation, and the decoder generates a series by converting the latent representation. We introduce an iTransformer-based autoencoder for modeling long-term time series in Section 3.1, variate conditions for capturing variatewise characteristics in Section 3.2, and multihead attention for capturing intervariate dependencies in Section 3.2.

3.1 Autoencoder for long-term time series modeling

Autoregressive generation hinders the generation of long-term series with low error. Therefore, AVC-GAN employs a time series length-agnostic GAN that operates on a latent space and is constructed by a nonautoregressive autoencoder based on the structure of iTransformer [Liu *et al.*, 2024]. Specifically, the autoencoder encodes the input time series into latent representations, and the GAN generates new latent representations that correspond to synthetic time series in this latent space.

The autoencoder uses an encoder–decoder architecture based on the iTransformer model architecture, as shown in Equations (3) and (4). To make the latent space such that the creation of a latent representation that reflects the characteristics of each variate is possible, the latent representation is designed with D dimensions for each variate, namely, $\boldsymbol{r} \in \mathbb{R}^{N \times D}$.

$$\boldsymbol{r} = \text{Encoder}(\boldsymbol{X}) \tag{3}$$
$$\boldsymbol{X}^{\text{recon}} = \text{Decoder}(\boldsymbol{r}) \tag{4}$$

An objective of the autoencoder is the reconstruction loss in

Table 1: Dataset description. The number of timesteps refers to the number of time points for which data were recorded.

Dataset	Dim.	Generation Length	Timesteps	Frequency	Domain
ETTh1, ETTh2	7	{192, 288, 432, 816}	17,420	Hourly	Electricity
ETTm1, ETTm2	7	{192, 288, 432, 816}	69,680	15 min	Electricity
Exchange	8	{192, 288, 432, 816}	7,588	Daily	Economy
Weather	21	{192, 288, 432, 816}	52,696	10 min	Weather

Equation (5).

$$\mathbb{E}_{\boldsymbol{X}\in\mathcal{D}}\|\boldsymbol{X}-\boldsymbol{X}^{\text{recon}}\|^2 \tag{5}$$

After autoencoder training, an encoder that converts time series into features and a decoder that converts features into time series are obtained. Thus, new time series become available if new latent representations are available.

3.2 GAN for latent representation generation

To generate a new series via the autoencoder after training with frozen weights, this section establishes a GAN for generating latent representations. That is, a generator, such as Equation (1), outputs a representation that imitates that of the autoencoder. The GAN needs to be a model that imitates the characteristics of each variate and generates latent representations that consider correlations among the variates simultaneously. Therefore, we use a variate conditional GAN to capture the characteristics of the variates and introduce multihead attention into the model structure to consider the correlations among the representations of each variate.

Variatewise characteristics

The generator for a latent representation of the trained autoencoder enables the generation of a representation for each variate. Therefore, we realize latent representation generation in which the variatewise characteristics are considered by conditioning the generator input for each variate and training to generate a representation that corresponds to each variate.

The input is conditioned on the variates by concatenating the embedding prepared for each variate with a vector $\boldsymbol{z}\in\mathbb{R}^{N\times D}$ sampled from a prior distribution P_z, which is a Gaussian distribution with a mean of 0 and a variance of 1. Specifically, we let $\boldsymbol{E}\in\mathbb{R}^{N\times D_E}$ be trainable parameters that indicate each variate; we replace the input vector of the generator with a conditioned input $\boldsymbol{z}'\in\mathbb{R}^{N\times(D+D_E)}$, where the n-th element of $\boldsymbol{z}'$ is $\boldsymbol{z}'_n=\boldsymbol{E}_n\|\boldsymbol{z}_n$, in which $\|$ is an operator that concatenates two vectors or matrices.

To train the generator with the variate-conditioned input, we employ WGAN-GP, which promotes stable training with a critic C conditioned by the variates. Equation (6) shows the objective of WGAN-GP.

$$\begin{aligned}&\min_G\max_C \mathop{\mathbb{E}}_{\boldsymbol{X}\in\mathcal{D}}[C(\text{Encoder}(\boldsymbol{X})\|\boldsymbol{E})]\\ &-\mathop{\mathbb{E}}_{\boldsymbol{z}\sim P_z}[C(G(\boldsymbol{z}')\|\boldsymbol{E})]+\lambda\mathop{\mathbb{E}}_{\boldsymbol{r}\sim P_{\boldsymbol{r}}}[(\|\nabla_{\boldsymbol{r}}C(\boldsymbol{r}\|\boldsymbol{E})\|_2-1)^2],\end{aligned} \tag{6}$$

where $P_{\boldsymbol{r}}$ denotes a uniform distribution of $\boldsymbol{r}$ for computing the gradient penalty term of the third expectation term. Specifically, we use a linearly interpolated representation $\alpha\boldsymbol{r}+(1-\alpha)\hat{\boldsymbol{r}}$ between the encoded original representation $\boldsymbol{r}=\text{Encoder}(\boldsymbol{X})$ and the generated representation $\hat{\boldsymbol{r}}=G(\boldsymbol{z}')$, where α is a weight sampled from the uniform distribution between 0 and 1. Training with this objective function requires the generator to capture the characteristics of each variate and generate a representation similar to that of the autoencoder.

Intervariate dependencies

To capture the correlations between variates, we incorporate multihead attention into the model architectures of the generator and critic. The multihead attention mechanism computes attention weights between the latent representations of different variates, thereby enabling the model to capture their intervariate dependencies. Specifically, the model architectures of both the generator and the critic are Transformer-like model structures that combine an MLP, which processes the representation of each variate, and multihead attention, which considers correlations among variates.

4 Experiments

4.1 Experimental Setup

Datasets

We conducted experiments on six real-world multivariate time series datasets commonly used for LTSF: ETTh1, ETTh2, ETTm1, and ETTm2 of the *Electricity Transformer Temperature (ETT)* datasets, which contain load information of electricity transformers and are helpful for power load forecasting [Zhou *et al.*, 2021]; the Exchange dataset, which is a collection of exchange rates between several countries; and the Weather dataset, which contains meteorological data such as temperature and humidity [Wu *et al.*, 2021]. Following the standard evaluation protocols for these datasets, we split these datasets chronologically into training, validation, and test sets. The ETT datasets were split at a 6:2:2 ratio, whereas the other datasets were split at a 7:1:2 ratio. Additionally, we used several *Short-Term Time Series Generation (STSG)* datasets: the Beijing, Guangzhou, Shenzhen, and Tianjin datasets from the Air Quality Dataset [Zheng *et al.*, 2015], the Energy dataset, and the Stock dataset [Yoon *et al.*, 2019]. We split the Energy and Stock datasets into training, validation, and test sets at a 7:1:2 ratio. For the Air datasets, we used the preexisting training/test split and further divided the original training set into training and validation sets at a 7:1 ratio. To ensure a fair evaluation of these datasets, we used only the training set to train the generative models, and we used the validation set to calculate scores. The results for the short-term generation are in A.

We varied the length of the generated series {192, 288, 432, 816} to evaluate the model's performance on sequences of different lengths. Detailed information on the datasets for LTSG, including the number of variates (dimensionality), the lengths of the generated series, the timesteps, the frequency, and a brief description of the data domain, i.e., electricity or weather, is summarized in Appendix Table 1.

Compared models

We compared AVC-GAN with several state-of-the-art GAN-based TSG models[2]: RTSGAN [Pei *et al.*, 2021], which has a similar architecture to AVC-GAN and trains the generator and discriminator in a latent space learned by an autoencoder; COSCI-GAN [Seyfi *et al.*, 2022], which focuses on capturing intervariate dependencies in multivariate time series; and AEC-GAN [Wang *et al.*, 2023], which addresses long-term time series generation. For all compared models, we used the hyperparameters reported in their respective original papers (or the official implementations). For AVC-GAN, the batch size was set to 1,024, the dimension of the variate condition embedding D_E was set to 32, the dimension of the latent representation D was set to 128, 40,000 training iterations were performed for the generator, and 10 critic updates were performed per generator iteration for LTSG. For STSG, D_E was set to 16, and D was set to 64, whereas the other parameters were set as in the case of LTSG. Regarding computational cost, each experimental setting run required approximately one day for AVC-GAN and COSCI-GAN, a few hours for AEC-GAN, and several days for RTSGAN.

Evaluation metrics

We evaluated the quality of the generated data via two categories of metrics: distance-based metrics, which compute the distance between the original and generated data by aligning them; and visualization of the time series. Distance-based metrics assess the fidelity and diversity of the generated data by observing the alignment between the original and generated data. The computation of distance-based metrics for evaluation focuses on intervariate dependencies and variatewise characteristics. We evaluated the long-term generation performance by evaluating these metrics at different series lengths. We calculated distance-based metrics between the original validation data and the generated data. We also have conducted conventional statistical evaluations; however, due to the space limitation, the results are described in Appendix B.

Distance-based metrics. Distance-based metrics are calculated using distances between aligned time series in the original and generated data. We used distances measured via the *Incoming Nearest Neighbor Distance (INND)* and *Outgoing Nearest Neighbor Distance (ONND)* metrics [Arnout *et al.*, 2019][3], where INND is the average distance from the generated data to the nearest original data points and ONND is the average distance from the original data to the nearest generated data points.

$$\mathrm{INND} = \frac{1}{|\hat{\mathcal{D}}|} \sum_{\boldsymbol{X}^{\mathrm{gen}} \in \hat{\mathcal{D}}} \min_{\boldsymbol{X} \in \mathcal{D}} \mathrm{MSE}(\boldsymbol{X}, \boldsymbol{X}^{\mathrm{gen}}) \tag{7}$$

$$\mathrm{ONND} = \frac{1}{|\mathcal{D}|} \sum_{\boldsymbol{X} \in \mathcal{D}} \min_{\boldsymbol{X}^{\mathrm{gen}} \in \hat{\mathcal{D}}} \mathrm{MSE}(\boldsymbol{X}, \boldsymbol{X}^{\mathrm{gen}}) \tag{8}$$

INND reflects the fidelity of the generated data by measuring the average distance from each generated sample to its nearest neighbor in the original data. Conversely, ONND reflects the diversity of the generated data by measuring the average distance from each original data point to its nearest neighbor in the generated samples. We also report *Balanced Nearest Neighbor Distance (BNND)*, which is the mean of INND and ONND, such that $\mathrm{BNND} = (\mathrm{INND} + \mathrm{ONND})/2$, thus providing a balanced measure of fidelity and diversity.

Since these metrics are calculated across all variates simultaneously, they are inadequate for evaluating the per-variate generation quality. To assess the generation quality for each variate separately, we calculate a distance-based metric on a per-variate basis by reporting *Variatewise Balanced Nearest Neighbor Distance (VBNND)*, which is calculated by averaging the BNND scores for each variate. We let $\mathrm{bnnd_i}$ denote BNND of the i-th variate; VBNND is expressed by Equation (9).

$$\mathrm{VBNND} = \frac{1}{N} \sum_{i=1}^{N} \mathrm{bnnd}_i \tag{9}$$

Visualization-based evaluation. To visually assess the similarity between the original and generated data, we plotted the data with time on the x-axis and values on the y-axis. To compare the generated data visually across different models, we selected representative variates from the ETTh1, ETTm2, and Weather datasets that exhibit distinct and easily discernible periodic patterns. Specifically, we chose "High Useful Load (HUFL)" from ETTh1, "Oil Temperature (OT)" from ETTm2, and "rh (%)" from the Weather dataset because of their clear cyclical behavior, which facilitated visual assessment of the ability of the model to capture temporal dynamics for each variate. We generated 10 synthetic time series for each model and selected the most visually similar sample to the original data.

4.2 Results

Distance-based metrics

Table 2 shows the results of evaluating the generated long-term time series via distance-based metrics. 1stCount indicates the number of times each model achieved the best score across all datasets and metrics, i.e., the number of values in bold font. Table 2 shows that AVC-GAN consistently achieved lower INND, ONND, BNND, and VBNND values than the compared models did, except on the Exchange dataset. These results indicate that AVC-GAN can generate long-term synthetic time series that accurately capture both variatewise characteristics, as measured by VBNND, and intervariate dependencies, as measured by BNND. Although

[2]We have downloaded and used the original implementations of the models for the experiments.

[3]We did not use the Euclidean distance or dynamic time warping defined in TSGBench, each of which measures the distance between the original data and the generated data without aligning time series from two sources, because they could be random without explicit alignment.

Table 2: Evaluation of long-term time series generation using distance-based metrics. I, O, B, and V denote the INND, ONND, BNND, and VBNND measures, respectively. Lower values indicate better performance.

Model		AVC-GAN (ours)				AEC-GAN				COSCI-GAN				RTSGAN			
Metric		I	O	B	V	I	O	B	V	I	O	B	V	I	O	B	V
ETTh1	192	**0.614**	**0.776**	**0.695**	**0.353**	0.725	1.064	0.894	0.429	1.509	1.417	1.463	0.791	1.222	2.820	2.021	1.809
	288	**0.673**	**0.876**	**0.775**	**0.433**	0.819	1.139	0.979	0.525	2.579	1.897	2.238	1.473	1.377	2.895	2.136	1.915
	432	**0.799**	**0.899**	**0.849**	**0.496**	0.935	1.220	1.078	0.620	2.105	2.184	2.144	1.738	1.452	2.875	2.164	1.935
	816	**0.980**	**1.127**	**1.053**	**0.599**	1.170	1.335	1.253	0.741	4.409	3.685	4.047	3.510	1.621	3.008	2.315	2.117
ETTh2	192	**0.724**	**0.403**	**0.563**	**0.251**	0.845	1.155	1.000	0.517	0.779	1.124	0.952	0.538	2.946	6.115	4.530	4.126
	288	**0.754**	**0.477**	**0.615**	**0.282**	0.872	1.184	1.028	0.592	1.013	1.285	1.149	0.715	3.076	6.099	4.587	4.225
	432	**0.871**	**0.514**	**0.693**	**0.315**	1.149	1.269	1.209	0.796	1.381	1.363	1.372	0.809	3.201	6.136	4.669	4.362
	816	**1.008**	**0.680**	**0.844**	**0.427**	1.107	1.226	1.166	0.828	1.785	2.229	2.007	1.578	3.348	6.391	4.870	4.617
ETTm1	192	0.527	**0.635**	**0.581**	**0.197**	0.641	0.816	0.729	0.255	**0.474**	0.903	0.688	0.276	0.770	2.662	1.716	1.570
	288	**0.569**	**0.635**	**0.602**	**0.225**	0.715	0.886	0.800	0.324	0.632	0.997	0.814	0.369	0.889	2.701	1.795	1.638
	432	**0.628**	**0.728**	**0.678**	**0.272**	0.769	0.954	0.861	0.387	2.078	1.456	1.767	1.009	1.122	2.823	1.972	1.718
	816	**0.658**	**0.807**	**0.733**	**0.353**	0.887	1.084	0.986	0.483	2.884	2.132	2.508	1.610	1.218	2.731	1.975	1.749
ETTm2	192	0.615	**0.204**	**0.410**	**0.163**	1.250	0.617	0.934	0.345	**0.540**	0.860	0.700	0.299	1.907	5.908	3.907	3.516
	288	0.660	**0.247**	**0.454**	**0.183**	3.269	0.702	1.986	1.370	**0.639**	0.959	0.799	0.346	2.220	5.961	4.091	3.709
	432	**0.688**	**0.408**	**0.548**	**0.211**	3.012	0.720	1.866	1.268	0.794	1.002	0.898	0.451	2.595	5.993	4.294	3.893
	816	**0.766**	**0.453**	**0.610**	**0.271**	2.714	0.791	1.753	1.185	1.106	1.202	1.154	0.567	2.954	6.012	4.483	4.097
Exchange	192	2.868	**0.760**	**1.814**	1.627	3.015	0.786	1.900	1.748	**1.413**	2.256	1.834	**1.575**	31,973	588.1	16,281	16,223
	288	3.108	**0.753**	1.930	1.773	3.019	0.891	1.955	1.799	**1.421**	2.186	**1.803**	**1.591**	19,782	597.6	10,190	10,103
	432	2.468	**0.653**	**1.561**	1.413	2.511	0.820	1.666	1.541	**1.174**	2.105	1.640	**1.366**	20,472	251.0	10,362	10,272
	816	2.729	1.195	1.962	1.587	2.423	**0.587**	1.505	1.312	**1.171**	1.558	**1.365**	**1.141**	20,855	348.5	10,602	10,576
Weather	192	**0.356**	**0.343**	**0.350**	**0.161**	0.569	0.484	0.526	0.304	0.460	0.545	0.502	0.302	709.3	734.5	721.9	718.0
	288	**0.372**	**0.385**	**0.378**	**0.184**	0.608	0.489	0.549	0.340	0.739	0.785	0.762	0.517	710.8	734.3	722.6	719.1
	432	**0.370**	**0.406**	**0.388**	**0.203**	0.613	0.529	0.571	0.353	0.870	0.846	0.858	0.590	713.7	734.4	724.0	720.8
	816	**0.530**	**0.477**	**0.503**	**0.265**	0.785	0.604	0.694	0.401	1.404	0.934	1.169	0.854	717.9	734.5	726.2	723.6
1stCount		82				1				13				0			

AEC-GAN exhibited relatively stable errors across different sequence lengths, its errors were significantly greater on the ETTm2 dataset. Overall, compared to AEC-GAN, AVC-GAN achieved score improvements of 29.8% and 33.7% in terms of the average scores of BNND and VBNND, respectively. The performance of COSCI-GAN varied across the datasets. While it achieved lower errors than the other methods did on the Exchange dataset, its errors tended to increase with sequence length on several other datasets, which potentially indicates a limitation in its ability to model long-range time series. RTSGAN presented the greatest number of errors overall, which suggests a limitation in effectively modeling multivariate long-term time series data.

In summary, compared with the other models, AVC-GAN demonstrated superior performance in generating high-quality, long-term time series data that capture both variate-wise characteristics and intervariate dependencies.

Visualization-based evaluation

As shown in Figure 3, AVC-GAN generated time series that closely resemble the original data in terms of both amplitude and periodicity. In contrast, the samples generated by AEC-GAN exhibit a shift in periodicity for the "OT" variate. The data generated by COSCI-GAN appear to contain more high-frequency noise. Furthermore, for the "rh" variate from the Weather dataset with a longer sequence length (816), RTSGAN showed a tendency for the generated waveform to decay in amplitude, which may be attributable to the accumulation of errors in its autoregressive generation process.

Overall, the visualization results support the quantitative results, namely, that AVC-GAN generated realistic time series that preserve key features of the original data, such as the amplitude, periodicity, and shape of the waveforms for individual variates.

5 Ablation study

To investigate the effectiveness of the multihead attention mechanism and variate conditioning in AVC-GAN, we conducted an ablation study. We generated synthetic data via modified versions of AVC-GAN, namely, a version without multihead attention (AVC-GAN w/o MHA) and a version without variate conditioning (AVC-GAN w/o VC), and evaluated the generated data via distance-based metrics. The experimental setup for this ablation study followed the LTSG protocol described in Section 4.1, with the generated series length fixed to 816.

The results in Table 3 demonstrate that both multihead attention and variate conditioning contributed to the performance. The reduction in VBNND for "AVC-GAN" compared with "AVC-GAN w/o VC" indicates that variate conditioning enables the model to capture the individual char-

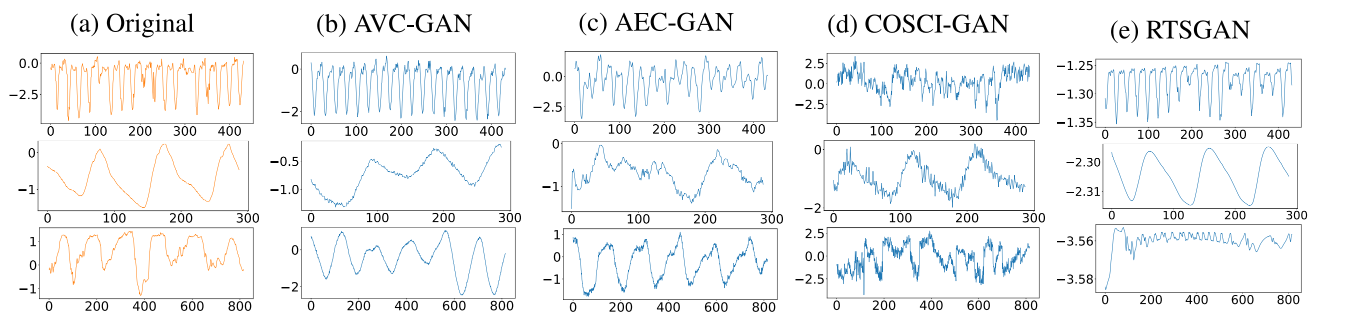

Figure 3: Examples of generated data. The top row shows waveforms for HUFL of ETTh1, the middle row shows those for OT of ETTm2, and the bottom row shows those for rh (%) of Weather. Each displayed series is the closest to the original among ten randomly sampled data generated by each model.

Table 3: Ablation study results for AVC-GAN, comparing the performance with and without variate conditions and multihead attention. Lower values indicate better performance. The first row corresponds to our proposed model. The second row corresponds to AVC-GAN with a GAN without the variate conditioning. The third row corresponds to AVC-GAN with a GAN without multihead attention.

Model	AVC-GAN				AVC-GAN w/o VC				AVC-GAN w/o MHA			
Metric	INND	ONND	BNND	VBNND	INND	ONND	BNND	VBNND	INND	ONND	BNND	VBNND
ETTh1	**0.332**	**0.323**	**0.327**	0.275	0.682	0.511	0.596	0.337	0.788	0.542	0.665	**0.272**
ETTh2	**0.272**	**0.273**	**0.272**	**0.219**	0.558	0.517	0.538	0.278	0.603	0.544	0.573	0.222
ETTm1	**0.301**	**0.276**	**0.289**	**0.183**	0.646	0.458	0.552	0.237	0.762	0.499	0.630	0.188
ETTm2	**0.193**	**0.181**	**0.187**	**0.123**	0.557	0.359	0.458	0.170	0.974	0.495	0.735	0.210
Exchange	**0.630**	0.109	0.369	0.294	0.652	**0.057**	**0.355**	**0.281**	0.775	0.245	0.510	0.306
Weather	**0.301**	**0.442**	**0.372**	**0.217**	0.818	0.821	0.819	0.326	0.892	0.864	0.878	**0.217**

acteristics of each variate better. The lower BNND for "AVC-GAN" than for "AVC-GAN w/o MHA" suggests that multihead attention effectively models the intervariate dependencies. On the Exchange dataset, "AVC-GAN w/o VC" exhibited a high INND and an extremely low ONND. This suggests that the model without variate conditioning may overfit limited patterns in the training data.

These results confirm that both the multihead attention mechanism and variate conditioning are crucial components of AVC-GAN and contribute to its ability to generate realistic long-term time series data that capture both intervariate dependencies and variatewise characteristics.

6 Conclusions and Remarks

This paper proposed a LTSG method called AVC-GAN. AVC-GAN integrates an iTransformer-based nonautoregressive autoencoder for modeling long-term time series, multihead attention for capturing intervariate dependencies, and variate conditioning for capturing variatewise characteristics. Our experiments, which included distance-based evaluations and visualization, demonstrated that AVC-GAN outperforms existing state-of-the-art methods. Ablation studies confirmed the effectiveness of both the multihead attention mechanism and variate conditioning in capturing variatewise characteristics and intervariate dependencies.

Our future work includes investigating the effectiveness of using AVC-GAN-generated synthetic data for downstream tasks such as privacy protection and TSF; this will include exploring methods for effectively incorporating synthetic data into training pipelines for these tasks and evaluating their effects on performance.

Ethical Statement

There are no ethical issues.

Acknowledgments

This study is partly supported by the Subsidy to Private Institutions of Higher Education for Current Expenditures from the Japanese Ministry of Education.

References

[Alcaraz and Strodthoff, 2023] Juan Lopez Alcaraz and Nils Strodthoff. Diffusion-based Time Series Imputation and Forecasting with Structured State Space Models. *Transactions on Machine Learning Research*, 2023.

[Allam *et al.*, 2024] Mohamed Allam, Noureddine Boujnah, Noel E. O'Connor, and Mingming Liu. Synthetic Time Series for Anomaly Detection in Cloud Microservices, 2024.

[Ang *et al.*, 2024] Yihao Ang, Qiang Huang, Yifan Bao, Anthony Tung, and Zhiyong Huang. Tsgbench: Time series generation benchmark. *Proceedings of the VLDB Endowment*, 17:305–318, 01 2024.

[Arnout *et al.*, 2019] Hiba Arnout, Johannes Kehrer, Johanna Bronner, and Thomas Runkler. Visual evaluation of generative adversarial networks for time series data, 2019.

[Chen *et al.*, 2023] Si-An Chen, Chun-Liang Li, Sercan O Arik, Nathanael Christian Yoder, and Tomas Pfister. TSMixer: An all-MLP architecture for time series forecast-ing. *Transactions on Machine Learning Research*, 2023.

[Cho *et al.*, 2014] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics.

[Desai *et al.*, 2021] Abhyuday Desai, Cynthia Freeman, Zuhui Wang, and Ian Beaver. TimeVAE: A variational auto-encoder for multivariate time series generation. *CoRR*, abs/2111.08095, 2021.

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

[Gulrajani *et al.*, 2017] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved Training of Wasserstein GANs. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[Ho *et al.*, 2020] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc., 2020.

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

[Hu *et al.*, 2023] Chaochen Hu, Zihan Sun, Chao Li, Yong Zhang, and Chunxiao Xing. Survey of time series data generation in iot. *Sensors*, 23(15), 2023.

[Jeha *et al.*, 2022] Paul Jeha, Michael Bohlke-Schneider, Pedro Mercado, Shubham Kapoor, Rajbir Singh Nirwan, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. PSA-GAN: Progressive self attention GANs for synthetic time series. In *International Conference on Learning Representations*, 2022.

[Kingma and Welling, 2022] Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes, 2022.

[Li *et al.*, 2022] Xiaomin Li, Vangelis Metsis, Huangyingrui Wang, and Anne Hee Hiong Ngu. TTS-GAN: A Transformer-Based Time-Series Generative Adversarial Network. In Martin Michalowski, Syed Sibte Raza Abidi, and Samina Abidi, editors, *Artificial Intelligence in Medicine*, pages 133–143, Cham, 2022. Springer International Publishing.

[Liu *et al.*, 2020] Yeqi Liu, Chuanyang Gong, Ling Yang, and Yingyi Chen. DSTP-RNN: A dual-stage two-phase attention-based recurrent neural network for long-term and multivariate time series prediction. *Expert Systems with Applications*, 143:113082, 2020.

[Liu *et al.*, 2024] Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. iTransformer: Inverted transformers are effective for time series forecasting. In *The Twelfth International Conference on Learning Representations*, 2024.

[Ni *et al.*, 2022] Hao Ni, Lukasz Szpruch, Marc Sabate-Vidales, Baoren Xiao, Magnus Wiese, and Shujian Liao. Sig-wasserstein GANs for time series generation. In *Proceedings of the Second ACM International Conference on AI in Finance*, ICAIF '21, New York, NY, USA, 2022. Association for Computing Machinery.

[Pei *et al.*, 2021] Hengzhi Pei, Kan Ren, Yuqing Yang, Chang Liu, Tao Qin, and Dongsheng Li. Towards Generating Real-World Time Series Data. In *2021 IEEE International Conference on Data Mining (ICDM)*, pages 469–478, 2021.

[Qian *et al.*, 2024] Zhaozhi Qian, Thomas Callender, Bogdan Cebere, Sam M. Janes, Neal Navani, and Mihaela van der Schaar. Synthetic data for privacy-preserving clinical risk prediction. *Scientific Reports*, 14(1):25676, 2024.

[Qin *et al.*, 2017] Yao Qin, Dongjin Song, Haifeng Chen, Wei Cheng, Guofei Jiang, and Garrison W. Cottrell. A Dual-Stage Attention-Based Recurrent Neural Network for Time Series Prediction. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 2627–2633, 2017.

[Rasul *et al.*, 2021] Kashif Rasul, Calvin Seward, Ingmar Schuster, and Roland Vollgraf. Autoregressive Denoising Diffusion Models for Multivariate Probabilistic Time Series Forecasting. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine*

Learning Research, pages 8857–8868. PMLR, 18–24 Jul 2021.

[Semenoglou *et al.*, 2023] Artemios-Anargyros Semenoglou, Evangelos Spiliotis, and Vassilios Assimakopoulos. Data augmentation for univariate time series forecasting with neural networks. *Pattern Recognition*, 134:109132, 2023.

[Seyfi *et al.*, 2022] Ali Seyfi, Jean-Francois Rajotte, and Raymond T. Ng. Generating multivariate time series with COmmon source coordinated GAN (COSCI-GAN). In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[Snow, 2020] Derek Snow. MTSS-GAN: Multivariate Time Series Simulation Generative Adversarial Networks. SSRN Electronic Journal, June 2020. Available at SSRN: https://ssrn.com/abstract=3616557 or http://dx.doi.org/10.2139/ssrn.3616557.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[Wang *et al.*, 2023] Lei Wang, Liang Zeng, and Jian Li. AEC-GAN: Adversarial Error Correction GANs for auto-regressive long time-series generation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(8):10140–10148, Jun. 2023.

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

[Wu *et al.*, 2021] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[Yoon *et al.*, 2019] Jinsung Yoon, Daniel Jarrett, and Mihaela van der Schaar. Time-series Generative Adversarial Networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[Zeng *et al.*, 2023] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(9):11121–11128, Jun. 2023.

[Zheng *et al.*, 2015] Yu Zheng, Xiuwen Yi, Ming Li, Ruiyuan Li, Zhangqing Shan, Eric Chang, and Tianrui Li. Forecasting Fine-Grained Air Quality Based on Big Data. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, page 2267–2276, New York, NY, USA, 2015. Association for Computing Machinery.

[Zhou *et al.*, 2021] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(12):11106–11115, May 2021.

Table 4: Datasets for short-term generation. The number of timesteps refers to the number of time points for which data were recorded.

Dataset	Dim.	Generation Length	Timesteps	Frequency	Domain
Air (Beijing)	12	{12, 24, 48, 96}	8,759	Hourly	Sensor
Air (Guangzhou)	12	{12, 24, 48, 96}	6,559	Hourly	Sensor
Air (Shenzhen)	12	{12, 24, 48, 96}	4,404	Hourly	Sensor
Air (Tianjin)	12	{12, 24, 48, 96}	8,759	Hourly	Sensor
Energy	28	{12, 24, 48, 96}	19,735	10 min	Appliances
Stock	6	{12, 24, 48, 96}	3,685	Daily	Economy

A Results on Short-Term Time Series Generation

Table 4 provides the statistics of the datasets used in the STSG. Subsequently, Table 5 reports the performance of each method in generating short-term time series, evaluated with distance-based metrics. The results demonstrate that the superior performance of AVC-GAN extends to STSG tasks, for which it achieved lower errors than the compared models on most datasets. AEC-GAN designed for LTSG generally underperformed compared with COSCI-GAN on STSG tasks. COSCI-GAN showed improved performance on STSG compared with LTSG. RTSGAN continued to exhibit high errors on all of the STSG datasets except Air (Guangzhou). The results on the STSG datasets further confirmed the effectiveness of AVC-GAN in generating realistic time series data.

B Evaluation using statistical metrics

Description of metrics

To assess the overall data quality for each variate, we employed four statistical metrics defined by TSGBench [Ang *et al.*, 2024]:

- *Marginal Distribution Difference (MDD)*
- *Autocorrelation Difference (ACD)*
- *Skewness Difference (SD)*
- *Kurtosis Difference (KD)*

These metrics compute the similarity of the statistical properties of the waveforms between the original and generated data. Lower values for these metrics indicate better performance, namely, greater similarity between the generated and original data.

MDD computes the histograms of values in the original and generated series for each variate and evaluates how closely the two distributions match based on the average of the absolute values of their differences.

$$\mathrm{MDD} = \frac{1}{NL}\sum_{i=1}^{N}\sum_{t=1}^{L}\int \left|p_{i,t}^{\mathrm{ori}}(x) - p_{i,t}^{\mathrm{gen}}(x)\right| dx \tag{10}$$

where N is the number of variates, L is the length of the time series, $p_{i,t}^{\mathrm{ori}}(x)$ is the probability density function of the i-th variate at time t in the original data $\boldsymbol{X}^{\mathrm{ori}}$, and $p_{i,t}^{\mathrm{gen}}(x)$ is that in the generated data $\boldsymbol{X}^{\mathrm{gen}}$.

ACD evaluates how similar the periodicities or patterns of two time series data are by calculating the difference in autocorrelation, which represents the relationship between the values of the time series data and their past values. In particular, for nonstationary data, the difference is evaluated based on the direct comparison of autocorrelation functions (ACFs) across all lags.

$$\mathrm{ACF}_{i,t,\tau} = \frac{x_{i,t}\cdot x_{i,\tau}}{|x_{i,t}|\cdot|x_{i,\tau}|} \tag{11}$$

$$\mathrm{ACD} = \sqrt{\sum_{i=1}^{N}\sum_{t=1}^{L}\sum_{\tau=1}^{L}\left(\mathrm{ACF}_{i,t,\tau}^{\mathrm{ori}} - \mathrm{ACF}_{i,t,\tau}^{\mathrm{gen}}\right)^2} \tag{12}$$

where t is the time step, τ is the lag, n is the variate index, and L is the time series length.

SD compares the skewness, which indicates symmetry or asymmetry, whereas KD compares the kurtosis, which indicates the peakedness of the distribution.

$$\mathrm{SD} = \left|\frac{\mathbb{E}[(\boldsymbol{X}^{\mathrm{gen}} - \boldsymbol{\mu}^{\mathrm{gen}})^3]}{\boldsymbol{\sigma}^{\mathrm{gen}3}} - \frac{\mathbb{E}[(\boldsymbol{X} - \boldsymbol{\mu})^3]}{\boldsymbol{\sigma}^3}\right| \tag{13}$$

$$\mathrm{KD} = \left|\frac{\mathbb{E}[(\boldsymbol{X}^{\mathrm{gen}} - \boldsymbol{\mu}^{\mathrm{gen}})^4]}{\boldsymbol{\sigma}^{\mathrm{gen}4}} - \frac{\mathbb{E}[(\boldsymbol{X} - \boldsymbol{\mu})^4]}{\boldsymbol{\sigma}^4}\right| \tag{14}$$

where $\boldsymbol{\mu}$, $\boldsymbol{\sigma}$, $\boldsymbol{\mu}^{\mathrm{gen}}$ and $\boldsymbol{\sigma}^{\mathrm{gen}}$ are the mean and standard deviation of the original data $\boldsymbol{X}$ and the generated data $\boldsymbol{X}^{\mathrm{gen}}$, respectively.

We have downloaded the authors' original implementations and conducted our-own experiments for the models on the same experimental settings because the experimental results on each publication are averages of several experiments. The scores we calculated are not exactly the same as the originals but are well aligned to the published scores.

Evaluation results

Table 6 shows the evaluation results for the feature-based metrics. As shown in Table 6, AVC-GAN had the highest 1stCount, which demonstrates its strong overall performance. The low MDD for AVC-GAN indicates its ability to capture the marginal distributions of the original data accurately. While AEC-GAN showed slightly better performance in terms of ACD, which suggests a stronger ability to capture the autocorrelations of the original data, AVC-GAN still achieved competitive results. COSCI-GAN exhibited relatively low SD and KD, which indicates a good fit to the shape of the original data distribution, particularly in terms of its asymmetry and peakedness/tailedness. RTSGAN generally showed higher errors across all the metrics.

In summary, the results in Table 6 demonstrate that AVC-GAN consistently achieved better performance across

Table 5: Evaluation of short-term time series generation using distance-based metrics. I, O, B, and V denote the INND, ONND, BNND, and VBNND measures, respectively. Lower values indicate better performance. Air (B) represents Beijing, Air (G) represents Guangzhou, Air (S) represents Shenzhen, and Air (T) represents Tianjin.

Model		AVC-GAN (ours)				AEC-GAN				COSCI-GAN				RTSGAN			
Metric		I	O	B	V	I	O	B	V	I	O	B	V	I	O	B	V
Air (B)	12	**0.793**	**0.460**	**0.627**	**0.166**	0.847	0.562	0.704	0.195	0.917	0.533	0.725	0.210	1.580	1.356	1.468	0.808
	24	**0.868**	**0.580**	**0.724**	**0.246**	0.944	0.676	0.810	0.284	0.922	0.694	0.808	0.315	1.661	1.344	1.503	0.843
	48	**0.946**	**0.659**	**0.803**	**0.354**	1.057	0.795	0.926	0.420	0.986	0.826	0.906	0.441	2.214	1.408	1.811	1.077
	96	1.145	**0.743**	**0.944**	**0.492**	1.309	0.879	1.094	0.579	**1.119**	1.019	1.069	0.604	2.639	1.448	2.044	1.351
Air (G)	12	**0.627**	**0.311**	**0.469**	**0.118**	0.685	0.349	0.517	0.140	0.766	0.344	0.555	0.160	2.099	1.496	1.797	0.731
	24	**0.705**	**0.348**	**0.527**	**0.189**	0.777	0.407	0.592	0.223	0.839	0.406	0.622	0.243	2.829	0.977	1.903	1.013
	48	**0.806**	**0.393**	**0.599**	**0.280**	0.886	0.461	0.674	0.332	0.910	0.491	0.701	0.351	4.329	1.047	2.688	1.807
	96	**0.888**	**0.444**	**0.666**	**0.392**	0.985	0.516	0.750	0.451	1.026	0.647	0.837	0.501	4.058	0.896	2.477	1.796
Air (S)	12	**0.871**	**0.436**	**0.653**	**0.128**	0.975	0.458	0.716	0.147	1.077	0.480	0.779	0.176	108.9	16.73	62.80	52.02
	24	**0.946**	0.537	**0.741**	**0.199**	1.072	**0.534**	0.803	0.234	1.165	0.581	0.873	0.275	112.0	28.15	70.07	57.94
	48	**0.981**	**0.541**	**0.761**	**0.294**	1.175	0.625	0.900	0.358	1.065	0.656	0.861	0.366	116.0	29.40	72.72	62.52
	96	1.124	**0.634**	**0.879**	**0.436**	1.357	0.776	1.067	0.532	**1.103**	0.778	0.941	0.520	121.9	38.90	80.40	64.84
Air (T)	12	**0.902**	**0.399**	**0.651**	**0.190**	0.917	0.496	0.706	0.206	0.975	0.431	0.703	0.213	0.727	1.047	0.887	0.484
	24	**0.968**	**0.496**	**0.732**	**0.267**	1.039	0.601	0.820	0.295	1.062	0.531	0.796	0.311	1.159	0.962	1.060	0.593
	48	**1.066**	**0.589**	**0.827**	**0.367**	1.143	0.691	0.917	0.409	1.198	0.643	0.921	0.443	1.425	1.018	1.221	0.750
	96	**1.202**	**0.671**	**0.937**	**0.519**	1.281	0.784	1.032	0.569	1.421	0.805	1.113	0.645	1.711	1.062	1.386	0.974
Energy	12	0.772	**0.344**	**0.558**	**0.106**	0.872	0.409	0.641	0.137	**0.734**	0.442	0.588	0.125	388.9	400.9	394.9	386.7
	24	0.825	**0.397**	0.611	0.145	0.901	0.441	0.671	0.168	**0.520**	0.484	**0.502**	**0.129**	389.1	399.6	394.3	386.7
	48	0.866	**0.419**	**0.642**	**0.179**	0.932	0.473	0.703	0.209	**0.718**	0.797	0.757	0.393	389.4	400.4	394.9	387.4
	96	0.864	**0.432**	**0.648**	**0.214**	0.990	0.504	0.747	0.264	**0.806**	0.898	0.852	0.507	390.2	399.8	395.0	388.4
Stock	12	**3.508**	**0.233**	**1.871**	**1.843**	3.699	0.487	2.093	2.051	3.741	0.246	1.993	1.955	13.58	21.16	17.37	17.36
	24	**3.801**	0.399	**2.100**	**2.080**	3.943	0.448	2.195	2.165	4.165	**0.165**	2.165	2.140	14.07	20.93	17.50	17.49
	48	**4.014**	0.352	2.183	2.172	4.107	0.367	2.237	2.217	4.190	**0.094**	**2.142**	**2.127**	14.48	20.51	17.50	17.49
	96	4.307	0.263	2.285	2.277	**4.214**	0.409	2.312	2.300	4.321	**0.140**	**2.230**	**2.214**	15.06	19.84	17.45	17.45
1^{st}Count		79				2				15				0			

the different datasets in terms of feature-based metrics, thus indicating its ability to generate high-quality synthetic time series data overall.

Table 6: Evaluation of synthetic data by statistical metrics. MDD: marginal distribution difference, ACD: autocorrelation difference, SD: skewness difference, and KD: kurtosis difference. Lower values indicate better performance.

Model		AVC-GAN (ours)				AEC-GAN				COSCI-GAN				RTSGAN			
Metric		MDD	ACD	SD	KD	MDD	ACD	SD	KD	MDD	ACD	SD	KD	MDD	ACD	SD	KD
ETTh1	192	**0.084**	2.763	0.544	1.244	0.086	**2.390**	0.686	2.204	0.138	2.838	1.025	**0.772**	0.259	2.960	**0.468**	1.732
	288	**0.088**	3.845	**0.516**	1.205	0.091	**3.292**	0.726	2.226	0.188	4.907	**0.877**	0.878	0.261	3.713	0.720	2.402
	432	0.097	5.348	0.488	**1.109**	**0.091**	**4.123**	0.822	2.548	0.147	5.950	**0.449**	1.533	0.261	5.290	0.636	1.760
	816	0.109	8.551	**0.490**	1.160	**0.102**	**5.961**	0.832	2.695	0.196	7.761	0.779	**0.812**	0.267	9.663	0.635	1.348
ETTh2	192	**0.400**	2.826	1.156	2.159	0.430	**2.389**	1.310	5.087	0.438	2.844	**0.569**	**1.058**	0.560	2.856	1.032	2.856
	288	**0.412**	3.887	1.235	2.255	0.445	**3.039**	1.278	5.220	0.448	4.260	**0.386**	**0.820**	0.572	3.667	1.062	3.326
	432	**0.391**	5.034	1.201	2.154	0.424	**3.682**	1.891	9.065	0.445	5.976	**0.343**	**0.815**	0.561	4.652	1.138	3.302
	816	**0.418**	7.692	1.088	2.915	0.432	**4.998**	2.005	10.03	0.442	8.266	**0.817**	**2.030**	0.569	7.862	0.981	3.538
ETTm1	192	**0.072**	2.103	0.628	1.920	0.077	**2.080**	0.676	2.216	0.100	2.419	0.834	2.154	0.268	2.325	**0.473**	**1.759**
	288	**0.069**	**2.711**	0.629	1.861	0.086	2.755	**0.550**	2.279	0.090	3.065	0.629	**1.648**	0.269	3.069	0.615	2.273
	432	**0.072**	**3.552**	0.648	1.911	0.090	3.700	**0.556**	2.251	0.152	5.400	1.053	**1.065**	0.269	4.012	0.619	1.839
	816	**0.077**	**5.541**	0.694	1.908	0.097	6.041	0.590	2.251	0.166	7.757	0.879	**1.113**	0.267	6.854	**0.579**	2.384
ETTm2	192	0.384	**1.868**	1.189	2.528	**0.347**	2.160	1.483	10.27	0.443	2.107	**0.510**	**1.355**	0.543	1.899	1.325	3.922
	288	0.385	2.601	1.164	2.499	**0.319**	4.031	2.351	19.35	0.435	3.014	**0.604**	**0.874**	0.545	**2.371**	1.220	3.476
	432	0.390	3.620	1.181	2.855	**0.322**	4.279	2.155	19.78	0.425	4.146	**0.481**	**0.829**	0.540	**3.417**	1.214	4.683
	816	0.382	6.022	1.239	**3.037**	**0.326**	**4.767**	1.502	23.95	0.430	7.664	**0.700**	3.407	0.537	5.024	1.270	3.121
Exchange	192	**0.609**	2.073	**0.665**	**0.688**	0.614	1.846	0.785	0.707	0.679	2.146	1.087	1.942	0.644	**1.392**	0.708	0.854
	288	**0.607**	2.890	0.632	**0.658**	0.614	2.508	0.693	0.740	0.682	3.414	0.986	2.759	0.654	**2.370**	**0.630**	0.857
	432	**0.620**	3.806	**0.583**	**1.022**	0.626	3.198	0.701	1.108	0.709	4.948	0.972	2.424	0.658	**3.094**	0.600	1.061
	816	0.665	8.593	0.885	1.834	**0.650**	**7.459**	**0.845**	**1.787**	0.825	8.396	1.138	2.700	0.675	8.635	1.066	2.245
Weather	192	0.378	**1.547**	**1.205**	**28.66**	0.342	2.444	15.11	3,586	**0.326**	1.970	2.318	80.21	0.842	1.921	4.640	182.0
	288	0.377	**2.219**	**1.566**	**39.65**	**0.353**	3.325	13.06	3,688	0.371	4.263	2.010	55.72	0.844	2.872	4.966	345.2
	432	0.425	**3.113**	1.579	38.65	**0.359**	4.458	13.33	4,495	0.417	5.205	**1.143**	**19.01**	0.842	4.196	5.277	238.0
	816	0.408	**5.533**	1.206	35.59	**0.371**	7.129	12.87	5,914	0.466	6.804	**1.134**	**29.99**	0.838	7.894	2.170	38.41
1st Count		34				25				27				10			