Cumulative Difference Learning VAE for Time-series with Temporally Correlated Inflow-Outflow

Tianchun Li [1] **Chengxiang Wu** [2] **Pengyi Shi** [3] **Xiaoqian Wang** [1]

Abstract

We provide the first generative framework that is tailored to the healthcare domain (hospital census generation) and develop corresponding domain-specific learning algorithms, *DT-VAE*, for the application of census generation, which has great practical importance, particularly after the outbreak of COVID-19. Our development specifically incorporates the underlying structures of the application problem (domain knowledge) and is mathematically justified. The DT-VAE method uses the latent variables to capture the complex temporal correlations while being able to generate the entire sequence of the census in a period of time in one shot, removing the reliance on recursive sampling or generation as in typical time-series works that focus on step-wise generation. We also provide new, open datasets to supplement existing datasets, which can be utilized by more researchers to develop domain-specific models for this important application domain.

1. Introduction

Generating synthetic time-series data has gained increasing interest. Time-series generation provides augmented datasets that can be used for testing and validating models, improving the accuracy of predictive models, avoiding model overfitting based on limited observations, as well as for generating scenarios for decision-making. It is widely used in fields such as finance, economics, and engineering to analyze and predict trends and patterns. Recently, more attention was brought to the healthcare and medicine fields, which can benefit significantly from the generated synthetic data. Because concerns about data privacy and restricted access limit the sharing of original data and hinder model comparisons and the development of new models.

This work is primarily motivated by the burgeoning needs of synthesizing hospital census (number of hospitalized patients in different units/hospitals/regions) since the outbreak of the COVID-19 pandemic. Predicting the *distribution* of hospital census, instead of point estimates, for a period of days in the future and generating future scenarios of census time-series is of critical importance in decision making such as surge bed capacity planning, nurse staffing, and allocation of medical equipment (e.g., ventilators, PPE) via sample-average based stochastic programming. However, the time-series generation in this domain has unique challenges that existing methods are not sufficient to address.

To elaborate, existing methods of time-series generation fall into three major categories: (i) Monte Carlo simulation that samples from prior defined distributions; (ii) deep-learning-based methods to train generators to learn the distribution from real-world data; and (iii) discrete-event simulation, which simulates the flow of patients through different units in a hospital. (Desai et al., 2021) provided a nice summary of the pros and cons of approaches in (i)-(ii) and called for a combination of both approaches for various downstream tasks such as prediction, classification, and testing. Challenges specifically to the hospital census generation and other similar practical applications are as follows. First, the census time-series are driven by temporally correlated inflow (arrivals) and outflow (departure) variables, where these variables are not directly observed for learning. Direct methods such as ARIMA or RNN, which ignore the dependence of the census on other variables, cannot capture the underlying complex temporal corrections. In particular, during the COVID-19 pandemic, the arrivals to and departures from hospitals were often driven by the disease spread situation in the local region and are correlated among days. Generative models are more suitable to address the first challenge and provide more flexibility. However, the majority of generative models such as GAN- or VAE-based models learn the step-wise, conditional distribution, e.g. the census distribution of the next day based on past history. The second challenge is that when generating future time-series, the errors accumulate due to the need of recursive generating and may not be able to capture the temporal patterns that

[1]School of Electrical and Computer Engineering, Purdue University, West Lafayette, Indiana, USA [2]Department of Computer Science, Purdue University, West Lafayette, Indiana, USA [3]Krannert School of Management, Purdue University, West Lafayette, Indiana, USA. Correspondence to: Pengyi Shi <shi178@purdue.edu>.

In Submission to *the 40^{th} International Conference on Machine Learning*, Hawaii, USA, 2023.

are critical for decision-making. Our numerical study will further demonstrate the insufficiency of existing methods when dealing with the first two challenges.

The third issue is that general-purposed (non-domain specific) generative methods tend to ignore the structure of the domain problems and often lack theoretical justification or interpretability. In our setting, the structure that the census is driven by daily arrivals and departures is important for designing theoretically-justified models to capture the underlying dependence and temporal correlations. It becomes a consensus that it is unlikely that a single general-purpose deep generative model works universally well; different domains require different architectures and tailored algorithms that incorporate the domain knowledge.

In this paper, we develop a novel, mathematically-justified and domain-aware time-series generation framework. We consider time series that are driven by temporally correlated inflow and outflow variables, where these variables are not directly observed for learning. As such, we set up a generative modeling framework, where a sequence of latent variables drives the inflow and outflow variables and their temporal correlations, which then drive the observed time series. We then develop a VAE-based method, the *cumulative Difference Temporal VAE* (DT-VAE) method, to perform learning within this generative framework.

Our development specifically incorporates the underlying structures of the application problem (domain knowledge) via a theoretical modeling framework, and addresses the aforementioned issues in a holistic manner. (i) The model structure provides the basis for us to mathematically derive the learning objective and the DT-VAE design, in contrast to using pure black-box models or heuristic design. (ii) To deal with the error accumulation problem from step-wise learning, we transform the original learning task to a mathematically equivalent task to learn the cumulative difference between the census on any given day t and the initial value (the *DT* part). This allows us to use the latent variables to capture the complex temporal correlations while being able to generate the entire sequence of the census in a period of time in one shot, removing the reliance on recursive sampling or generation. (iii) The encoder and decoder of the *VAE* part provide a flexible and general framework to learn the underlying correlations and mappings from the sequence of latent variables to the future census. Meanwhile, the designs are mathematically justified via our theoretical generative model setup and the transformation of the learning task.

We demonstrate the superior performance of our DT-VAE method over several benchmark methods (including the state-of-art, general-purpose method TimeGAN) on two domain-specific datasets: the first one is a semi-synthetic dataset derived based on a collaborating partner hospital; the other is a new dataset, created by the authors and made open, based on published COVID-19 hospitalization records. Our method is portable and very easy to train. Meanwhile, it is very flexible to connect with subsequent learning tasks. For example, we show that by connecting with a GAN component, the resulting DT-VAE-GAN method can generate more accurate time series for classification tasks. We also design several performance metrics specifically for subsequent tasks in our domain application, e.g., the temporal trends need to be preserved well after the data generation since this is critical for scenario generation in decision support. The metrics are both numerical scores and visualized, where the visualized metrics can help interpretation.

To sum, we provide the first generative framework that is tailored to the healthcare domain (hospital census generation) and develop corresponding domain-specific learning algorithms and performance metrics, where the application has great practical importance, particularly after the outbreak of COVID-19 (and becoming an endemic). We also provide new, open datasets to supplement existing datasets, which can be utilized by more researchers to develop domain-specific models for this important application domain. For broader impact, our method can be generalized to other application areas with similar problem structures, for example, traffic flow prediction (with input and output traffic flows), and demand prediction (with customers joining/leaving the waitlist).

2. Related Work

Related Work

- **GAN based framework**

  The first work. C-RNN-GAN (Mogren, 2016), combines LSTM and GAN.

  Recurrent Conditional GAN (RCGAN) (Esteban et al., 2017) combines conditional GAN and Recurrent Neural Networks.

  Time-series GAN (TimeGAN) (Yoon et al., 2019) used Recurrent Neural Networks to align the latent representations of real and generated data.

  Progressive Self Attention GAN (PSA-GAN) (Jeha et al., 2021) used attention structures to progressively learn the generative functions. This works focuses more on filling in missing values in the time-series data.

  Time-series Transformer Generative Adversarial Networks (TsT-GAN) (Srinivasan & Knottenbelt, 2022) combines the transformers with GAN to generate a time-series sequence.

- **VAE based framework**

Variational Recurrent AutoEncoder (Fabius & Van Amersfoort, 2014) is one of the very first works that combines the VAE and RNN. However, it doesn't have any derivations. I highly suspect that it corresponds to our T-VAE baseline. Need more time to check.

Stochastic WaveNet (Lai et al., 2018) borrowed insights from VAE and proposed a general generative framework for sequential data. This is more similar to our framework as it also generates the joint distribution with recursive construction. If we have time, we can use this as one of the baseline

Stochastic Temporal Convolutional Networks (S-TCN) (Aksan & Hilliges, 2019) also borrowed VAE's idea and combined it with Temporal Convolutional Networks to generate sequential data. This is not that related as it generates the marginal distribution.

TimeVAE (Desai et al., 2021) combined VAE and some interpretable components to generate time series. We could also compare this as a baseline. Although it's refused, the comments from the reviewers are very useful. We can review those when we polish the draft.

- **Other related framework**

  Generative models for Time-series data with Fourier Flows (Alaa et al., 2020) used the Fourier flows to formulate the problem.

  Time-series Generation by Contrastive Imitation (Jarrett et al., 2021) [Tianchun: Still need time to review this paper]

3. Preliminaries

In this section, we will briefly review the related methods.

Variational Autoencoder As a generative model, the goal of Variational Autoencoder (VAE) (Kingma & Welling, 2013) is generating new data similar to the training data. To achieve this, the VAE has an Encoder-Decoder design that encodes training data to a latent space and generates new data from this latent space.

For the given training dataset $\mathcal{D} = \{x^{(i)}\}_{i=1}^{N}$, where consists of N i.i.d. samples x, VAE assumes that each $x^{(i)}$ can be generated by some unobserved latent variable $z^{(i)}$. In the context of time-series data, each $x^{(i)}$ and $z^{(i)}$ can be vectors that contain observations over T time steps. The VAE comprises three components that utilize this assumption for data generation: an encoder, a decoder, and a prior distribution. The encoder, which has parameters ϕ, approximates the posterior distribution $p(z|x)$ via $q_\phi(z|x)$ and enforces this distribution to resemble a known prior distribution $p(z)$ through a KL distance penalty. One of the common assumptions on prior distribution $p(z)$ in VAE is that it follows a standard Gaussian distribution with mean 0 and identity covariance matrix $\big(p(z) \sim N(0, I)\big)$. The decoder, which has parameters θ, is to reconstruct the data $x^{(i)}$ by approximating the conditional distribution $p_\theta(x|z)$ from all sampled latent variables $z_{post}^{(i)} \sim q_\phi(z|x)$ from the posterior distribution. Note that VAE utilizes the so-called reparameterization trick to enable the differential sampling process. During the generation stage, we can directly sample latent variables $z_{prior} \sim p(z)$ from the prior distribution and let the decoder generate new data.

To learn the encoder and decoder, VAE optimizes the Evidence Lowerbound (ELBO) (Jordan et al., 1999). The ELBO serves as a surrogate objective function for the log-likelihood function (which is often intractable) and follows

$$\min_{\theta,\phi} -\mathbb{E}_{z\sim q_\phi}\big(\log(p_\theta(x|z))\big) + D_{KL}\big(q_\phi(z|x)||p(z)\big) \quad (1)$$

The first term in (1) is the negative log-likelihood of the conditional distribution $p_\theta(x|z)$ when z is sampled from the variational distribution $q_\phi(z|x)$, often referred to as the reconstruction loss. Minimizing the reconstruction loss helps the decoder learn how to reconstruct x from the latent variable sampled from the posterior distribution during the training stage. The second term in (1) is the KL-divergence between the posterior distribution $q_\phi(z|x)$ and the prior distribution $p(z)$. Minimizing this KL-divergence introduces a regularization to ensure the decoder does not lose the variance so that we can generate new data from the latent variable drawn from the prior distribution during the generation stage.

Combination of Variational Autoencoder and Generative adversarial network (VAE-GAN) Despite its huge success, VAE still has its limitations. One such example is that VAE tends to generate blur images in the field of Computer Vision. In contrast, Generative adversarial networks (Goodfellow et al., 2020) (GAN) are known for generating more vivid and diverse images than VAE but can be more challenging to train. One way to benefit from the strengths of both GAN and VAE is to combine the VAE and GAN (Larsen et al., 2016), often referred to as VAE-GAN.

VAE-GAN combines VAE and GAN by treating the decoder of the VAE as the generator network in GAN and utilizing the GAN's discriminator network. Additionally, to fully utilize the rich similarity metric learned by the discriminator network f_d in distinguishing between real and generated data, VAE-GAN replaces the traditional VAE reconstruction loss with a reconstruction loss expressed in the GAN discriminator. This is achieved by assuming a Gaussian observation on the $l-th$ layer embeddings in the discriminator $f_{d,l}(x)$ with mean $f_{d,l}(\tilde{x})$ and identity covariance $p(f_{d,l}(x)|z) = \mathcal{N}(f_{d,l}(\tilde{x}), I)$, where $\tilde{x} \sim p_\theta(x|z)$ are the samples from the VAE's decoder. Then, VAE-GAN replaces

the reconstruction loss as follows:

$$\mathcal{L}^{Dis}_{Reconstruction} = -\mathbb{E}_{z\sim q_\phi}\big(\log(p(f_{d,l}(x)|z))\big) \quad (2)$$

Therefore, the training of VAE-GAN is adversarial and composes triple criterion as follows:

$$\min_{\theta,\phi}\big(\mathcal{L}^{Dis}_{Reconstruction} + D_{KL} + \max_d \mathcal{L}_{Dis}\big) \quad (3)$$

where D_{KL} denotes the KL-divergence term in 1, $\mathcal{L}_{Dis}$ denotes the binary classification loss in GAN, and θ and ϕ are parameters of the encoder and decoder in the VAE as previously mentioned.

4. Method

4.1. Problem Formulation

Consider a collection of random variables that form a time-series sequence $\{X_t, t = 0, 1, \ldots, T\}$ with the length of $T + 1$. We denote this sequence as $X_{0:T}$. The training dataset $\mathcal{D}$ consists with N observed sequences, denoted as $\mathcal{D} = \{x^{(1)}_{0:T}, \ldots, x^{(N)}_{0:T}\}$. For a given sequence, the observation at time t, x_t, is a vector in $\mathbb{R}^k$, where k is the feature space size. Going forward, N and k will be omitted unless explicitly mentioned. Our goal is to learn the joint distribution $p(X_{0:T})$ from the training data.

Generative Models. We adopt the generative modeling framework for the time series. The assumptions for the dependency structure between the latent variables and observations are summarized in Figure 1 and are specified as follows. We assume that each random variable X_t is driven by the previous random variable X_{t-1}, an "inflow" (arrival) variable A_t, and an "outflow" (discharge) variable D_t. The relationship of X_t, X_{t-1}, A_t, D_t is described as

$$\begin{aligned} X_t|X_{t-1}, A_t, D_t &= X_{t-1} + A_t - D_t + \epsilon \\ s.t. \quad t &= 0, \ldots, T \quad \epsilon \sim N(0, \tau) \end{aligned} \quad (4)$$

where τ denotes the covariance matrix in the Gaussian distribution. Note that we may also assume that the noise can change over time as $\epsilon_t \sim N(0, \tau_t)$, where τ_t denotes the covariance matrix in the Gaussian distribution but can change over time. Let us temporarily continue with the fixed covariance matrix τ and focus on the following assumptions and descriptions. Since the random variable X_t is dependent on other unobserved variables, we will refer to X_t as the dependent variable. Next, we elaborate on the motivation of this dependence structure in the healthcare setting and the assumptions on the arrival and discharge variables.

In managing hospital resources, X_t corresponds to the patient census on day t, the number of hospitalized patients in a hospital unit, a county, or a region. This census is driven by the daily number of arrivals A_t and daily discharges D_t, which are further driven by some underlying "envoriment factors" $\{Z_t\}$. Take the pandemic as an example: the latent variables correspond to the disease spread and recovery. Motivated by the stochastic SIR model (Allen, 2008; 2017) that is commonly used for modeling epidemic process, we assume that the arrivals A_t and discharges D_t follow

$$\begin{aligned} A_0 &= a_0; \quad D_0 = d_0; \\ A_t &= A_{t-1} + b_a(A_{t-1}) + \sigma_a Z^a_t, t = 1, \ldots, T \quad (5) \\ D_t &= D_{t-1} + b_d(D_{t-1}) + \sigma_d Z^d_t, t = 1, \ldots, T \quad (6) \end{aligned}$$

where the sequences of latent variables $Z^a_1, \ldots, Z^a_T \sim^{iid} \mathcal{N}(0, I_k)$ and $Z^d_1, \ldots, Z^d_T \sim^{iid} \mathcal{N}(0, I_k)$ are all i.i.d. standard Gaussian vectors in $\mathbb{R}^k$ and drive the arrival and discharge processes. Equations (5) and (6) can be seen as the discretized version of the original stochastic differential equations for the stochastic SIR model (more details are given in the appendix B), with $b_a(\cdot)$ and $b_d(\cdot)$ as the (unknown) drift functions and $\sigma_a Z^a_t$ and $\sigma_d Z^d_t$ as the (unknown) diffusion terms; see more details in the online supplement.

The parameters to be learned include those parametrizing the drift and the diffusion terms. This learning task is challenging in many healthcare settings because very often, only the hospitalized count (census) is tracked each day, not the daily arrivals and discharges. [Tianchun: TBD: cite website sources to show only daily hospitalization is tracked.] In other words, we cannot directly learn the parameters from $\{A_t\}$ and $\{D_t\}$ but need to learn from $\{X_t\}$ via the dependence structures given in Equations (4) and (5)-(6). Therefore, we adopt the generative methods for the learning task, which eventually provides the joint distribution of $\{X_t\}$ to fulfill subsequent tasks such as prediction or time-series generation. Compared to direct methods such as ARIMA, which model $\{X_t\}$ as a stochastic process and directly learn parameters that drive the stochastic process, our generative-model-based method can capture more complex temporal correlations yet avoid error accumulation (a problem faced by many of the direct methods) as we show in Section 4.2. We will demonstrate this advantage via numerical study.

4.2. Cumulative Difference Learning

A common way of learning via the time-series data $\{X_t\}$ is step-wise learning, i.e., learning the conditional distribution $X_t|X_{0:t-1}$, which faces an issue: the possible accumulation of errors; we elaborate on this point later. To overcome this issue, we use a novel cumulative difference learning, specified as follows. First, we denote $\Delta_t = A_t - D_t$. The variable Δ_t is the random variable that represents the difference between arrival and discharge variables A_t and D_t from (4), i.e., the net changes in X_t's. In our training dataset, we can directly observe the sample $\delta_t = x_t - x_{t-1}$. The variable X_t is also dependent on the difference variable Δ_t.

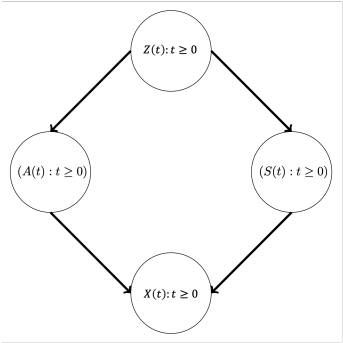


Figure 1. Geneative model structure.

Then, we can rewrite the relationship between X_t and X_{t-1} as

$$X_t|X_{t-1}, \Delta_t = X_{t-1} + \Delta_t + \epsilon \tag{7}$$

Then, from (5) and (6), the difference variable can be described as follows:

$$\begin{aligned} \Delta_0 &= a_0 - d_0; \\ \Delta_t &= b_\Delta(\Delta_{t-1}) + \sigma_\Delta Z_t^\Delta, \quad t = 1, \ldots, T, \end{aligned} \tag{8}$$

Directly learning on the difference variable Δ_t is essentially learning the unknown parameters in drift functions $b_\Delta(\cdot)$ and unknown diffusion matrix σ_Δ.

Directly learning on the latent difference variable is possible from (8), yet this approach imposes an additional challenge. Let $\widehat{\Delta}_l$ denote the learned difference variable to approximate true difference variable Δ_l at time step l where $l < T$. In practice, $\widehat{\Delta}_l$ is likely biased and inaccurate. This inaccurate estimation of the difference variable $\widehat{\Delta}_l$ will result in an inaccurate dependent variable $\widehat{X}_l$ from (7). Eventually, the generated observed variable $\widehat{X}_T$ must be biased and may even be catastrophically deviated compared to the true X_T because of the recursive relationship from (7), i.e., the accumulation of errors due to the step-wise learning. Therefore, a proper learning strategy is needed.

To avoid this problem, we define a new variable that captures the *cumulative* difference. Formally, we define

$$\Gamma_t = X_t - X_0 = \sum_{i=1}^{t} \Delta_i = \sum_{i=1}^{t} A_i - \sum_{i=1}^{t} D_i, \tag{9}$$

Here, in (9), Γ_t can be described as the cumulative difference. In our training dataset, this cumulative difference can be observed by $\gamma_t = x_t - x_0$. The new relationship between X_0, X_t, and Γ_t can be described as follows:

$$X_t|X_0, \Gamma_t = X_0 + \Gamma_t + \epsilon_t' \quad s.t. \quad \epsilon_t' \sim N(0, \tau_t') \tag{10}$$

It's important to note that, different from (4), the noise ϵ_t' must change over time as the covariance matrix τ_t' will change over time when considering the cumulative difference.

From (10), it appears that the current observed variable X_t only depends on the current information of Γ_t for any given starting census X_0. However, all previous latent variables drive the cumulative difference Γ_t. This can be seen from the following recursive equations on Γ_t, which are obtained from combining (8) and (9):

$$\begin{aligned} \Gamma_0 &= \Delta_0 = a_0 - d_0 \\ \Gamma_t &= \sum_{t=1}^{T} \Delta_t = \sum_{i=1}^{t-1} \Delta_i + \Delta_t \\ &= \Gamma_{t-1} + b_\Delta(\Delta_{t-1}) + \sigma_\Delta Z_t^\Delta, \quad t = 1, \ldots, T \end{aligned} \tag{11}$$

The reconstruction process from Γ_t to X_t in (9) prevents the error accumulation. Because Γ_t is a cumulative difference, it only requires the initial value X_0 to reconstruct X_t, unlike the recursive reconstruction method for learning differences in (7). Specifically, by expanding the recursion in (11), we can see that the variable Γ_t for each day is driven by the sequence of latent variables $Z_{1:t}^\Delta$. Therefore, we are essentially learning a mapping from the sequence of latent variables $Z_{1:t}^\Delta$ to the variable Γ_t. Any bias present in the learned $\widehat{\Gamma}_{t-1}$ will not affect the learning of Γ_t, as it is determined solely by the latent variables. Meanwhile, this dependency does not eliminate the temporal correlations within sequences. Because Γ_{t-1} depends on $Z_{1:t-1}^\Delta$, the dependency between Γ_t and $Z_{1:t}^\Delta$ automatically encodes the temporal correlations between Γ_{t-1} and Γ_t.

4.3. Cumulative Difference Temporal VAE (DT-VAE)

In this section, we specify how to use a VAE-based framework to learn the cumulative difference given by Equations (10) and (11). We refer to our framework as *DT-VAE*.

ELBO. Denote the observed sequence of cumulative difference variables $\Gamma_{1:T}$ as $\gamma_{1:T} = \{\gamma_1, \ldots, \gamma_T\}$. As previously mentioned from (11), Γ_t is driven by latent variables $Z_{1:t}^\Delta$ at each time step $t = \{1, \cdots, T\}$. Let $z_{1:T} = (z_1, \ldots, z_T)$ denote the sequence of prior variables from prior distribution $p(z_{1:T})$. Therefore, we assume that the $\gamma_{1:t}$ can be generated from prior variables $z_{1:t}$. The log-likelihood of joint distribution $p(\gamma_{1:T})$ can be described as

$$\log p_\theta(\gamma_{1:T}) = \log \int p_\theta(\gamma_{1:T} \mid z_{1:T}) p(z_{1:T}) dz_{1:T} \tag{12}$$

where θ represents model parameters. This marginal likelihood function is difficult to calculate. VAE evaluates the Evidence Lowerbound (ELBO) as the surrogate likelihood function. The ELBO in our temporal setting is derived as

below:

$$\begin{aligned}
\log p_\theta(\gamma_{1:T}) &= \log \int p_\theta(\gamma_{1:T}, z_{1:T}) dz_{1:T} \\
&= \log \int p_\theta(\gamma_{1:T}, z_{1:T}) \frac{q_\phi(z_{1:T} \mid \gamma_{1:T})}{q_\phi(z_{1:T} \mid \gamma_{1:T})} dz_{1:T} \\
&\geq \mathbb{E}_{z_{1:T}\sim q_\phi}\left[\log\left(\frac{p_\theta(\gamma_{1:T}, z_{1:T})}{q_\phi(z_{1:T}|\gamma_{1:T})}\right)\right] \\
&= \mathbb{E}_{z_{1:T}\sim q_\phi}\left[\log\left(\frac{\prod_{t=1}^T p(\gamma_t|z_{1:t}))p(z_t|z_{1:t-1})}{\prod_{t=1}^T q_\phi(z_t|z_{1:t-1},\gamma_{1:t})}\right)\right] \\
&= \sum_{t=1}^T \mathbb{E}_{z_{1:t}} \log p(\gamma_t|z_{1:t}) \\
&\quad - \mathbb{E}_{z_{1:t-1}} D_{KL}\Big(q_\phi(z_t|z_{1:t-1},\gamma_{1:t})||N(0,I)\Big) \\
&= \mathcal{L}(\gamma_{1:T})
\end{aligned} \tag{13}$$

The detailed derivation is given in Appendix A. Here, we describe the joint distribution $p_\theta(\gamma_{1:T}, z_{1:T})$ as the generative process for generating $\gamma_{1:T}$, and $q_\phi(z_{1:T}|\gamma_{1:T})$ denotes the variational distribution with parameter ϕ that approximates the true posterior distribution. Next, we will describe the generative process $p_\theta(\gamma_{1:T}, z_{1:T})$ and variational distribution $q_\phi(z_{1:T}|\gamma_{1:T})$ separately.

Decoder design. For generative process, DT-VAE is to learn a decoder $f_\theta(\cdot)$ with parameter θ to decode latent variables $z_{1:t} \sim P(z_{1:t})$ from the latent space to generate γ_t. In DT-VAE, the decoder $f_\theta(\cdot)$ essentially learns the conditional distribution $p_\theta(\gamma_t|z_{1:t})$. Going forward, we will use $f_\theta(z_{1:t})$ and $p_\theta(\gamma_t|z_{1:t})$ interchangeably.

A key step in deriving the ELBO in (13), particularly from line 3 to line 4, is via the following decomposition for $p_\theta(\gamma_{1:T}, z_{1:T})$

$$\begin{aligned}
p_\theta(\gamma_{1:T}, z_{1:T}) &= p_\theta(\gamma_{1:T}|z_{1:T})p(z_{1:T}) \\
&= \Big(\prod_{t=1}^T p_\theta(\gamma_t|z_{1:t})\Big)p(z_{1:T}) \\
&= \prod_{t=1}^T p_\theta(\gamma_t|z_{1:t}) \prod_{t=1}^T p(z_t|z_{1:t-1})
\end{aligned} \tag{14}$$

where $p_\theta(\gamma_t|z_{1:t})$ denotes the approximation of the true conditional distribution $p(\gamma_t|z_{1:t})$ and $p(z_t|z_{1:t-1})$ denotes the prior distribution for latent variables z_t.

From (14), we make an important assumption on the conditional distribution $p_\theta(\gamma_{1:T}|z_{1:T})$ and prior distribution $p(z_{1:T})$. As previously mentioned, for each γ_t, it solely depends on latent variables $z_{1:t}$ to avoid error accumulation. This essentially makes γ_t to be *conditionally independent* across different time steps given the latent variables $z_{1:t}$. That is, for any two time steps $w, v = 1, \cdots, T$ and $w \neq v$, the cumulative difference variable $(\gamma_w|z_{1:w}) \perp (\gamma_v|z_{1:v})$ are independent conditional on corresponding latent variables. This assumption is crital, allowing the transformation from $p_\theta(\gamma_{1:T}|z_{1:T})$ to the product form $\prod_{t=1}^T p_\theta(\gamma_t|z_{1:t})$.

Following the VAE literature, we assume the conditional distribution $p_\theta(\gamma_t|z_{1:t}) \sim N(\mu_{t,\theta}, \sigma_{t,\theta})$, i.e., a Gaussian distribution with a diagonal covariance matrix. Note that the $\sigma_{t,\theta}$ has to change over time as previously mentioned in (10). For the prior distribution, we assume they are independent Gaussian, namely, $p(z_t|z_{1:t}) \sim N(0, I)$. Though $z_t \sim p(z_t|z_{1:t-1})$ is independent of other variables $z_{1:t-1}$, the decoder f_θ still allows us to restore the relationships between z_t and $z_{1:t-1}$. In this way, we do not lose the correlation but are able to speed up the generation process significantly given the independence between z_t's. Because instead of sampling recursively, we can directly sample all z_t from the standard Gaussian distribution.

For implementation, we design the decoder via a recurrent network $_{\theta_1}$, enclosing all time steps information $z_{1:t}$ recursively, with a feedforward network $_{\theta_2}$, further transforming.

$$h_t = f_{\theta_1}(h_{t-1}, z_t) \qquad (\mu_{t,\theta}, \sigma_{t,\theta}) = f_{\theta_2}(h_t) \tag{15}$$

where h_t is the hidden state in the RNN structure f_{θ_1}. The fully connected layer f_{θ_2} further transforms h_t to the mean μ_t and σ_t for the Gaussian distribution.

Note that the decoder f_θ can be parameterized by any architecture, as long as the output γ_t depends on the latent variables $z_{1:t}$. For instance, it is possible to implement the decoder by self-attention-based structures (Vaswani et al., 2017) or temporal convolutions (Oord et al., 2016).

Posterior distribution and encoder design. Next, we will describe the posterior distribution, also known as the encoder. DT-VAE learns an encoder $f_\phi(\cdot)$ with parameter ϕ to encode observed $\gamma_{1:t}$ into the variational (posterior) distribution $q_\phi(z_{1:T}|\gamma_{1:T})$. Going forward, we will use $f_\phi(\gamma_{1:t})$ and $p_\phi(z_{1:t}|\gamma_{1:t})$ interchangeably.

We factor the posterior distribution $q_\phi(z_{1:T}|\Gamma_{1:T})$ as the following

$$q_\phi(z_{1:T}|\gamma_{1:T}) = \prod_{t=1}^T q_\phi(z_t|z_{1:t-1}, \gamma_{1:t}) \tag{16}$$

During the training stage, we will sample z_t from the posterior distribution $q_\phi(z_t|z_{1:t-1}, \gamma_{1:t})$ and let the decoder reconstruct the observed γ_t. The sampling procedure from posterior distribution can be described as follows. At each time step t, we sample the posterior variable z_t from the distribution conditioned on the historical posterior variables $z_{1:t-1}$ and all observed $\gamma_{1:t}$.

Following the VAE literature, we assume that variational distribution $q_\phi(z_t|z_{1:t-1}, \gamma_{1:t}) \sim N(\mu_{t,q}, \sigma_{t,q})$, i.e., a Gaussian distribution with a diagonal covariance matrix, where

the $\mu_{t,q}$ and $\sigma_{t,q}$ are learned using the encoder f_ϕ. To capture the both historical information in z's and γ's, we decompose f_ϕ into three functions with parameters ϕ_1, ϕ_2 and ϕ_3

$$\begin{aligned} h_{t,q} &= f_{\phi_1}(h_{t-1,q}, \gamma_t) \\ \mu_{t,q} &= f_{\phi_2}(h_{t,q}, \mu_{t-1,q}) \\ \sigma_{t,q} &= f_{\phi_3}(h_{t,q}, \sigma_{t-1,q}) \end{aligned} \tag{17}$$

where $h_{t,q}$ is the hidden state in RNN structure f_{ϕ_1}. For each time step, $h_{t,q}$ encodes the all observed $\gamma_{1:t}$. The RNN structure f_{ϕ_2} will output the mean of posterior distribution $\mu_{t,q}$ by utilizing the $h_{t,q}$ and previous $\mu_{t-1,q}$. Therefore, for each time step, the current mean $\mu_{t,q}$ contains information of previous means $\mu_{1:t-1,q}$, which corresponds the conditional distribution $q_\phi(z_t|z_{1:t-1})$ from (16). The RNN structure f_{ϕ_3} will output the variance of posterior distribution $\sigma_{t,q}$ by utilizing the $h_{t,q}$ and previous $\sigma_{t-1,q}$.

Generation overview. During the training stage, the encoder generates a sequence of latent variables $z_{1:t}$, where each z_t is drawn independently from a Gaussian distribution with a diagonal covariance matrix $N(\mu_{t,\phi}, \sigma_{t,\phi})$. These parameters $\mu_{t,\phi}$ and $\sigma_{t,\phi}$ are learned based on the previous latent variables $z_{1:t-1}$ and input data $\gamma_{1:t}$ with functions parameterized by ϕ. To ensure that gradients can be computed, the "parameterization trick" is used. In the meantime, the encoder also enforces the posterior distribution to resemble standard Gaussian distribution by the KL distance penalty in (13). The decoder, during training, takes the latent variables $z_{1:t}$ from the posterior distribution (output by the encoder) and tries to reconstruct the input data $\gamma_{1:t}$ by drawing each γ_t from a Gaussian distribution with mean $\mu_{t,\theta}$ and variance $\sigma_{t,\theta}$. This is controlled by the log-likelihood in (13). These parameters $\mu_{t,\theta}$ and $\sigma_{t,\theta}$ are learned based on some functions with parameters θ, which take the latent variables $z_{1:t}$ as input.

During the generation stage, we can directly sample all z_t from standard Gaussian and use the decoder to generate the data $\gamma'_{1:t}$. For any given initial x_0, the corresponding value of x_t can be generated by $x_t = x_0 + \gamma_t$.

Training ELBO loss. It is well known that the VAE-based framework might be difficult to train due to the KL term in the ELBO loss. Therefore, we provide an additional hyperparameter λ to further balance the log-likelihood and the KL term as

$$\begin{aligned} \mathcal{L}(\gamma_{1:T}) = &\sum_{t=1}^{T} \mathbb{E}_{z_{1:t}} \log p\big(\gamma_t|z_{1:t}\big) \\ &- \lambda \mathbb{E}_{z_{1:t-1}} D_{KL}\Big(q_\phi(z_t|z_{1:t-1}, \gamma_{1:t})||N(0, I_k)\Big) \end{aligned} \tag{18}$$

where λ is a constant and $\lambda > 0$.

4.4. DT-VAE-GAN

[Tianchun: Under construction]Downstream tasks like daily prediction or classification require an accurate generation from the generative model. To further improve DT-VAE, we connect DT-VAE with GAN component to let our method have a more accurate generation. To achieve this, we implement the discriminator network via RNN to distinguish, for each time step, whether the input $\hat{\gamma}_{1:t}$ is from generated data or real data:

$$h_{t,d} = f_d(h_{t-1,d}, \hat{y}_t) \qquad \hat{y}_t = c(h_{t,d}) \tag{19}$$

where $h_{t,d}$ is the hidden state of RNN f_{d_1} and contains information from all time steps $\hat{\gamma}_{1:t}$, and c is the output layer for classification. Furthermore, borrowing the idea from VAE-GAN, we further utilize the discriminator on samples from the prior distribution $z_{prior,1:t} \sim p(z_{1:t})$ and also samples from the posterior $z_{post,1:t} \sim q_\phi(z_{1:t}|\gamma_{1:t})$. In specific, the objective of the discriminator in DT-VAE-GAN is described as follows:

$$\begin{aligned} \mathcal{L}_{GAN} = &\log\big(f_d(\gamma_{1:t})\big) + \log\big(1 - f_d(f_\theta(z_{prior,1:t}))\big) \\ &+ \log\big(1 - f_d(f_\theta(z_{post,1:t}))\big) \end{aligned} \tag{20}$$

where f_θ denotes the decoder in the DT-VAE as previously mentioned.

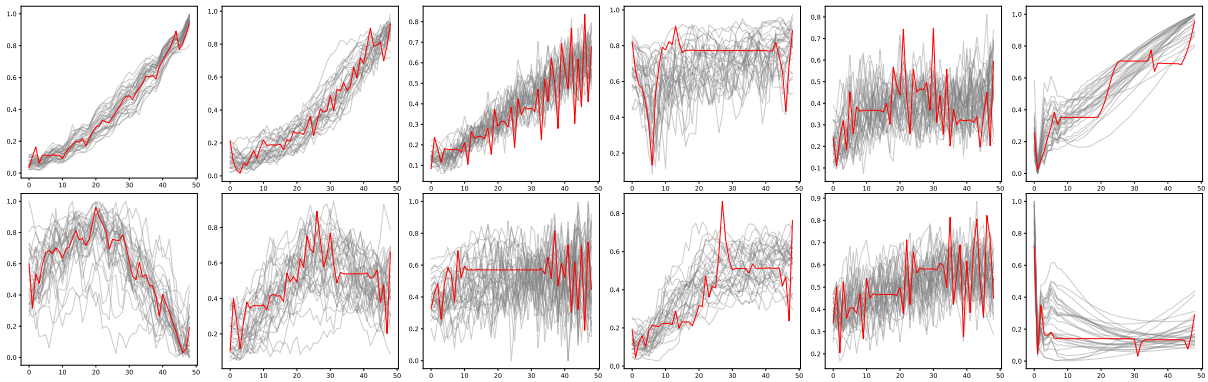
On the other hand, different from VAE-GAN, we cannot directly replace the reconstruction loss as in 2 since we assume that the conditional distribution $p_\theta(\gamma_t|z_{1:t}) \sim N(\mu_{t,\theta}, \sigma_{t,\theta})$ as previously mentioned. As an alternative, we formulate the reconstruction loss in 2 as an auxiliary loss to further control the DT-VAE-GAN with a constant weight η. In specific, the loss for DT-VAE-GAN is as follows:

$$\begin{aligned} \min_{\theta,\phi} \big(&\mathcal{L}_{Reconstruction} + \lambda D_{KL} \\ &+ \eta \mathcal{L}^{Dis}_{Reconstruction} + \max_d \mathcal{L}_{Dis}\big) \end{aligned} \tag{21}$$

where $\mathcal{L}_{Reconstruction}$ and D_{KL} are the log-likelihood and KL divergence in the (18), and $\eta\mathcal{L}^{Dis}_{Reconstruction}$ is the reconstruction loss associate with the discriminator's embedding in (2)

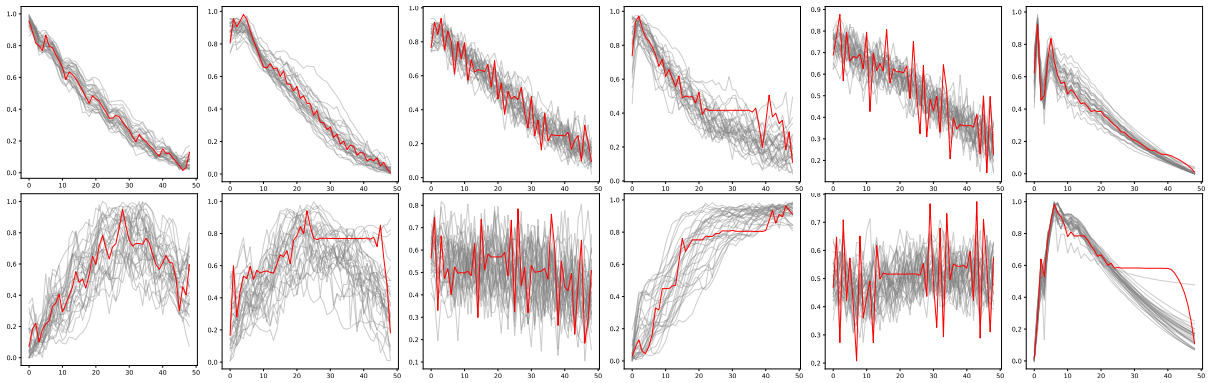
5. Experiment

We evaluate our method on two groups of datasets: the first is a synthetic dataset produced based on real data from an individual partner hospital; the second is a set of open data from public data sources on COVID-19 hospitalizations. We specify the datasets in Section 5.1. Given the healthcare context, we design our evaluation metrics on generated time-series data as commonly done in generative models, where evaluation metrics require to be domain-specific. For example, in chemical molecule generation, the novel fractions



(a) Real (I+E) (b) DT-VAE-GAN (c) DT-VAE (d) T-VAE-GAN (e) T-VAE (f) TimeGAN

Figure 2. First column provides the visualization for the groups of clusters (test set) in the state daily hospitalization dataset, including Increasing (INC) and Early Peak (ERP) trends. The next columns provide the visualization for each of the 5 benchmarks. The centroids of each cluster (red) can represent the general trend for the corresponding time-series data (grey).



(a) Real (D+L) (b) DT-VAE-GAN (c) DT-VAE (d) T-VAE-GAN (e) T-VAE (f) TimeGAN

Figure 3. First column provides the visualization for the groups of clusters (test set) in the state daily hospitalization dataset, including Decreasing (DEC) and Late Peak (LTP) trends. The next columns provide the visualization for each of the 5 benchmarks. The centroids of each cluster (red) can represent the general trend for the corresponding time-series data (grey).

in generated molecules are measured (Liu et al., 2018). In protein structure generation, protein perplexity is measured (Ingraham et al., 2019). In image generation, Inception score (Salimans et al., 2016) and FID score (Heusel et al., 2017) are common metrics used to evaluate the diversity and clarity of the generated images.

For our application domain, the generation of patient census data is often used for subsequent census prediction and decision support, e.g., generating future census scenarios for staffing planning via stochastic optimization (Shi et al., 2022). The nature of these tasks requires the generated data to preserve the temporal patterns (e.g., the day-of-week pattern) and allow the downstream tasks performed on generated data to be as close as possible to be on the real data. Therefore, we consider two main principles in designing the evaluation metrics: (i) measuring to which extent the generative model preserves trends and patterns in the real data; (ii) measuring the performance of the generated data on challenging downstream tasks. We specify the domain-specific evaluation metrics we designed for these two purposes in Section 5.2. We also evaluate the performance of our model using common metrics, i.e., we apply the t-SNE (Van der Maaten & Hinton, 2008), and PCA (Bryant & Yarnold, 1995) on both real and generated data, in which the temporal dimension is flattened. These visualizations show the similarity of distributions between the original and generated data. The detailed experiment setup is in Section 5.3, and all performance comparisons are in Section 5.4.

5.1. Datasets

Semi-Synthetic Data A primary novelty of our method is the generative modeling structure and the cumulative difference learning, which are designed to capture complex temporal correlations. To highlight this benefit with other conventional time-series prediction methods such as ARIMA, we simulate the hospital census $x(t)$ consider-

ing the complex impacts with respect to epidemics and actual patient flows in hospitals. In specific, we simulate the number of patient arrivals $a(t)$ according to the discretized Cox–Ingersoll–Ross(CIR) process (Cox et al., 2005) to imitate the spread of the epidemics. The number of patient discharges $d(t)$ is simulated by randomly assigning patients to units and length of stay with different probabilities. The corresponding parameters in the simulation are calculated empirically using the real data from our partnerIn particular, we calibrated the day-of-week pattern, which is a salient feature in hospital census. More details on the CIR model and the semi-synthetic generation is given in appendix C.

COVID-19 County Daily Hospitalization (County-DH) We gathered COVID-19 county hospitalization data from the government websites [Tianchun: add ref] of the three most populous states: California, New York, and Pennsylvania. The data covers the period from March 29, 2020, to June 6, 2022, and includes numbers of patients hospitalized in both Medical/Surgical units and ICUs. Missing values were dealt with by using interpolation and smoothing (see Appendix D.1). Due to the large variation among counties (large cities versus small rural counties), we develop a new normalization method, namely Week-of-Day Quantile-min normalization (see Appendix D.2).

COVID-19 States Daily Hospitalization (States-DH) In addition to the county-level data described above, we also gathered state-level data on the number of patients hospitalized from September 2, 2020, to November 2, 2022, provided by the US health department [Tianchun: add ref]. This data, which includes patients hospitalized in both Medical/Surgical units and ICUs, was aggregated at the state level. We pre-process the data using the same techniques for the County DH as described above.

5.2. Domain-specific Evaluation Metrics

Semi-Synthetic Dataset *(i) Parameter comparison.* Since the semi-synthetic data is simulated with the parameters for the arrival and discharge processes being known, we can directly compare the learned parameters and the true ones for evaluation purpose (i), i.e., evaluate whether the generated model is able to learn and preserve the temporal trend.

(ii) Evaluations on downstream tasks. For evaluation purpose (ii), we consider the "Long-term Prediction" downstream task, which involves using a period of time-series data to predict a succeeding period of data. These tasks are common in practice and require practitioners to forecast future values based on past observations. To reflect that the generated data preserves the predictive properties of real data, we use the 'Train on Synthetic, Test on Real' (TSTR) approach (Esteban et al., 2017). After creating the hold-out test dataset and training dataset for the generative models, which contain similar trends, we use the generated data to train a post-hoc sequence-prediction model (by optimizing 2-layer LSTM), which uses a period of data to predict a succeeding period of data. To evaluate, the post-hoc sequence-prediction model will perform same prediction task in the hold-out test dataset. We evaluate the prediction performance on the test dataset using the Mean Square Error (MSE) as our Predictive score. More details can be found in Appendix G.

Real Dataset For the real data, we use the same metrics on downstream tasks as for the semi-synthetic data for the evaluation principle (ii). However, for principle (i), we cannot directly compare the learned parameter values since we do not know the "ground-truth" parameters. To evaluate whether the generated data can preserve important temporal pattern information, we design the following clustering-then-separate framework for evaluating (i).

Trend Pattern Clustering. Multiple COVID-19 pandemic waves caused diverse trend patterns over time, e.g., in some months the number of hospitalizations increased while in other months it decreased. We first identify these trend patterns present in the real dataset, using a 7-week time window. Specifically, we use the K-means Dynamic Time Warping Barycenter Averaging (DBA) algorithm (Petitjean et al., 2011), which is based on the Dynamic Time Warping distance (Müller, 2007), to group similar patterns in the real dataset into clusters. Appendix TBD shows the clusters we identify, which clearly demonstrate the different temporal patterns in hospitalizations. Specifically, for the States-DH dataset, four different trends: (i) Increasing, (ii) Decreasing, (iii) Early Peak, and (iv) Late Peak; see Figure 6. For the County-DH dataset, also four different trends as shown in in the appendix: (i) Increasing, (ii) Decreasing, (iii) Flat Valley, and (iv) Late Peak; see Figure 5.

Diverse Trend Preservation (DTP) evaluation. With the identified clusters, we construct the training data and test data with time-series sequences sampled from each combination of two clusters in the States-DH or County-DH dataset, where the two clusters have distinct temporal patterns. In other words, the input is sampled from a mix of two types of patterns. To evaluate whether the generated time series (i.e., output) can adequately learn and preserve the two types of input patterns, we apply the K-means DBA algorithm again to the generated data (from the generative model) to see if two clusters can be identified and whether these two clusters have similar pattern with the original two input clusters. That is, whether the generative model can successfully separate the two types of input patterns *and* learn the corresponding pattern well. The DTP visualization displays the trends identified by the K-means DBA algorithm in both the hold-out test dataset and generated dataset. We then calculate the corresponding DTP score by mea-

suring the similarity between the centroids of the clusters identified in the test dataset and generated dataset. More precisely, we employ the Dynamic Time Wrapping (DTW) distance as the metric, which measures the alignment between time-series sequences. See more details about the K-means DBA algorithm in the appendix E.

5.3. Baselines and Setup

Baselines Since we focus on the healthcare domain, we select one benchmark which is a general framework for all domains. We design two other benchmarks as variations of our methods that misses the key component – cumulative difference learning. The first one is TimeGAN (Yoon et al., 2019), which is regarded as the state-of-the-art model in time-series generation and is frequently used as a baseline for comparison in the literature. TimeGAN is a general-purpose generative framework for all fields, while our method is more domain specific. Moreover, we specifically targeted our design to address the issue of error accumulation while TimeGAN still generates in an autoregressive way. The other two benchmarks are Temporal VAE (T-VAE) and Temporal VAE-GAN (T-VAE-GAN). These two benchmarks have the exact same structure as DT-VAE and DT-VAE-GAN but directly learn on the census instead of the cumulative difference.

Setup

5.4. Results Comparison

We evaluate our method using DTP scores and visualizations, Prediction scores, and t-SNE and PCA visualizations as mentioned previously.

DTP Visualization In Figures 2 and 3, the generated data from our method (DT-VAE and DT-VAE-GAN) demonstrates overall better performance in preserving the original trends in real data. In fact, the increasing trend (first row) in the generated data closely matches the trend in the real data. The cumulative learning framework further enhances the preservation of diverse trends (e.g. compare results from DT-VAE-GAN in column (b) and T-VAE-GAN in column (d)). On the other hand, TimeGAN tends to generate similar samples among diverse trends and cannot differentiate the clusters, which is infrequent for decision support. In contrast, our VAE-based methods effectively preserve patterns and generate diverse samples.

DTP Score As Table 3 illustrates the distance between the centroids identified in generated data and real data (lower the better), our method demonstrates significant improvements over other benchmarks, with $12\% \downarrow$ on average and up to $30\% \downarrow$. Additionally, as previously stated, the input data contains two distinct trends, and our method exhibits low scores for both trends in most cases. In the County DH, while TimeGAN outperforms our method on the Late Peak (LTP) trend by $5\% \downarrow$, our method outperforms TimeGAN on the Decreasing (DEC) trend by up to $60\% \downarrow$. These performance boosts demonstrate that our method effectively preserves diverse trends in the original data, which matches our observations in DTP visualization.

Prediction Scores In this experiment, we use the first 40 days to predict the next 9 days. As shown in table 1, our methods consistently generate better quality data on the basis of long-term prediction scores. Our methods demonstrate significant improvements over other benchmarks, with 10% lower on average and up to $\%$ [Tianchun: Eric: please help with the numbers] lower. On the other hand, we observe that TimeGAN achieves the second-best score for Training Set 2 in County DH. As previously mentioned the training sets contain similar trends, and TimeGAN tends to generate similar samples and lacks trend diversity. In this case, the samples generated by TimeGAN may achieve comparable performance when considering long-term prediction tasks.

t-SNE and PCA Visualization

Figure 4 shows t-SNE visualization on the State Daily Hospitalization test set. We observed that our methods show better overlap with the original data than other benchmarks. Particularly, without cumulative difference learning, other benchmarks have poor alignment between the generated data and real data. We also note that TimeGAN has a thin pattern in t-SNE, which is consistent with the observation that TimeGAN lacks trend diversity. More experiments and observations are given in appendix I

References

Aksan, E. and Hilliges, O. Stcn: Stochastic temporal convolutional networks. *arXiv preprint arXiv:1902.06568*, 2019.

Alaa, A., Chan, A. J., and van der Schaar, M. Generative time-series modeling with fourier flows. In *International Conference on Learning Representations*, 2020.

Allen, L. J. A primer on stochastic epidemic models: Formulation, numerical simulation, and analysis. *Infectious Disease Modelling*, 2(2):128–142, 2017. ISSN 2468-0427. doi: https://doi.org/10.1016/j.idm.2017.03.001. URL https://www.sciencedirect.com/science/article/pii/S2468042716300495.

Allen, L. J. S. *An Introduction to Stochastic Epidemic Models*, pp. 81–130. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-78911-6. doi: 10.1007/978-3-540-78911-6_3. URL https://doi.org/10.1007/978-3-540-78911-6_3.

Bryant, F. B. and Yarnold, P. R. Principal-components

	County DH		State DH	
	Training Set 1	Training Set 2	Training Set 1	Training Set 2
Original	0.0959±0.0144	0.2401±0.0951	0.0186±0.0129	0.1252±0.0544
TimeGan	1.8062±0.1364	0.9848±0.1055	0.2020±0.0293	0.5258±0.0175
T-VAE	3.7915±0.0526	5.5988±0.4505	1.3143±0.1046	1.2552±0.0506
T-VAE-GAN	2.5431±0.1062	6.2268±0.1393	0.7234±0.1095	2.2192±0.2093
DT-VAE	**0.5983±0.0546**	3.1878±1.1980	0.1642±0.0453	0.3223±0.0412
DT-VAE-GAN	0.6804±0.2742	**0.9396±0.5801**	**0.1091±0.0904**	**0.1666±0.0196**

Table 1. Results on Long-term downstream prediction task. Numbers reported are the MSE and corresponding confidence interval. Bold indicates the best performance, underlined indicates the next-best performance. Training set 1 in both County and State DH includes most of increasing trends. Training set 2 in both datasets includes most of decreasing trends. More details are given in the appendix E.6.

Metric	Dataset	Trends	TimeGAN	T-VAE	T-VAE-GAN	DT-VAE	DT-VAE-GAN
DTP Score	County-DH	INC	0.4942±0.00001	0.8730 ± 0.00066	0.4164±0.00054	0.7840±0.00356	**0.3717±0.00017**
		FLV	1.085±0.00000	0.7970±0.00028	0.8186±0.000362	0.8243±0.000061	**0.7636±0.00008**
		DEC	0.8640±0.00019	0.7932±0.00063	1.2546±0.00075	0.5760±0.00010	**0.3271±0.00003**
		LTP	**0.4954±0.00001**	1.0184±0.00065	1.0099±0.000031	0.9704±0.00031	0.5225±0.00002
	States-DH	INC	0.3252±0.00124	0.9341±0.01340	0.6570±0.00282	0.7435±0.00089	**0.3240±0.00029**
		ERP	0.8574±0.00011	0.9310±0.00065	0.8067±0.11438	0.9250±0.00762	**0.7564±0.00191**
		DEC	0.4392±0.00066	0.6822±0.00991	0.3359±0.00055	0.4421±0.00044	**0.3135±0.00003**
		LTP	0.7095±0.02287	1.2387±0.00224	0.9031±0.00304	1.1094±0.00089	**0.4566±0.00232**

Table 2. Results on DTP Scores using Dynamic Time Wrapping (DTW) distance (*the lower the better*) for Medical/Surgical units. The corresponding trends in the datasets are Increasing (INC), Flat Valley (FLV), Decreasing (DEC), and Late Peak (LTP). More details about the original trends and additional experiments can be found in Appendix [Tianchun: add ref].

analysis and exploratory and confirmatory factor analysis. 1995.

Cox, J. C., Ingersoll Jr, J. E., and Ross, S. A. A theory of the term structure of interest rates. In *Theory of valuation*, pp. 129–164. World Scientific, 2005.

Desai, A., Freeman, C., Wang, Z., and Beaver, I. Timevae: A variational auto-encoder for multivariate time series generation. *arXiv preprint arXiv:2111.08095*, 2021.

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

Fabius, O. and Van Amersfoort, J. R. Variational recurrent auto-encoders. *arXiv preprint arXiv:1412.6581*, 2014.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

Ingraham, J., Garg, V., Barzilay, R., and Jaakkola, T. Generative models for graph-based protein design. *Advances in neural information processing systems*, 32, 2019.

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

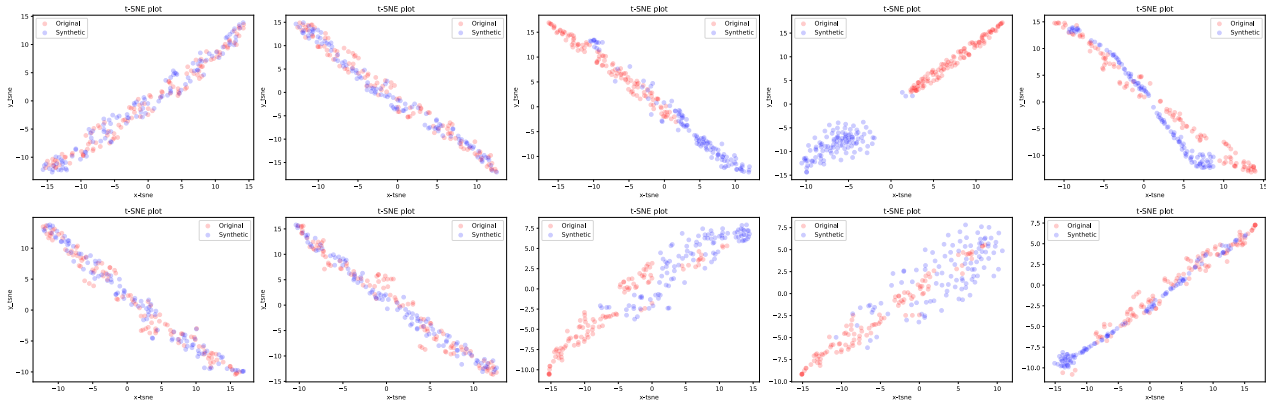
Jeha, P., Bohlke-Schneider, M., Mercado, P., Kapoor, S., Nirwan, R. S., Flunkert, V., Gasthaus, J., and Januschowski, T. Psa-gan: Progressive self attention gans for synthetic time series. In *International Conference on Learning Representations*, 2021.

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Lai, G., Li, B., Zheng, G., and Yang, Y. Stochastic wavenet: A generative latent variable model for sequential data. *arXiv preprint arXiv:1806.06116*, 2018.

Larsen, A. B. L., Sønderby, S. K., Larochelle, H., and Winther, O. Autoencoding beyond pixels using a learned similarity metric. In *International conference on machine learning*, pp. 1558–1566. PMLR, 2016.



(a) DT-VAE-GAN (b) DT-VAE (c) T-VAE-GAN (d) T-VAE (e) TimeGAN

Figure 4. PCA visualization on Decreasing and Early Peak trends in the State Daily Hospitalization test set (1st row) and Increasing and Late Peak in the State Daily Hospitalization test set (2nd row). Each column provides the visualization for each of the 5 benchmarks. Red denotes original data, and blue denotes generated data.

Liu, Q., Allamanis, M., Brockschmidt, M., and Gaunt, A. Constrained graph variational autoencoders for molecule design. *Advances in neural information processing systems*, 31, 2018.

Mogren, O. C-rnn-gan: Continuous recurrent neural networks with adversarial training. *arXiv preprint arXiv:1611.09904*, 2016.

Müller, M. Dynamic time warping. *Information retrieval for music and motion*, pp. 69–84, 2007.

Oord, A. v. d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

Petitjean, F., Ketterlin, A., and Gançarski, P. A global averaging method for dynamic time warping, with applications to clustering. *Pattern recognition*, 44(3):678–693, 2011.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. Improved techniques for training gans. *Advances in neural information processing systems*, 29, 2016.

Shi, P., Helm, J. E., Chen, C., Lim, J., Parker, R. P., Tinsley, T., and Cecil, J. Operations (management) warp speed: Rapid deployment of hospital-focused predictive/prescriptive analytics for the covid-19 pandemic. *Production and Operations Management*, 2022.

Srinivasan, P. and Knottenbelt, W. J. Time-series transformer generative adversarial networks. *arXiv preprint arXiv:2205.11164*, 2022.

Van der Maaten, L. and Hinton, G. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

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

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