DYNAMIC-AWARE GANS: TIME-SERIES GENERATION WITH HANDY SELF-SUPERVISION

Anonymous authors
Paper under double-blind review

ABSTRACT

This paper presents Dynamic-Aware GAN (DAGAN) as a data-efficient self-supervised paradigm for time-series data generation. To support sequential generation with sufficient clues of temporal dynamics, we explicitly model the transition dynamics within the data sequence through differencing, thus refining the vanilla sequence into one with inter-correlated triplets to characterize each time-step. This localized triplet consistent structure contributes to a self-supervision mechanism, which can provide more aspects of supervision for the overall stepwise dependencies encoded within the training data. Such a handy self-supervision mechanism is simple but can be beneficial especially when a model is presented with limited training data. Based on the insight, we present DAGAN which generalizes the locally regularized triplet consistency to distributional-level via dynamic encoding and joint distribution matching. Experiments on various synthetic and real-world datasets verify that our model achieves superior generation results with better quality and diversity compared with the state-of-the-art benchmarks, especially when the training data is scarce. Moreover, benefited from the dynamic-conditional and dynamic-consistent design, our DAGAN is capable of performing dynamic-controllable generation *i.e.* generating data with specified dynamics.

1 INTRODUCTION

With the wide popularity of smart devices and online services, we are facing and also producing significant amounts of data every minute of the day. Such a situation poses a rapid explosion of sequential data which greatly expedites the demand for time-series data analysis. While performing effective time-series data analysis relies on full exploitation of historical data, there are often circumstances where we can hardly obtain adequate amounts of data for analysis, due to low-occurrences issues (*e.g.* monitoring of system crash) or privacy reasons (*e.g.* medical sequential data in the ICU). Time-series data generation provides an intuitive solution to tackle such data shortage dilemmas.

Due to its sequential property, the effectiveness of a sequential generation model highly relies on its ability to make full use of the dynamic information encoded within the training data. To model temporal dynamics, previous methods mainly resort to three strategies. Auto-regressive methods (Goyal et al., 2016; Bahdanau et al., 2017) factorize the distribution of sequences into a product of stepwise conditionals. These methods, fitting the observed data without extra conditioning, lack the flexibility to generate arbitrary sequences. TimeGAN (Yoon et al., 2019), among all the prevalent Generative Adversarial Networks (GANs)-based sequential generation models (Mogren, 2016; Esteban et al., 2017; Ramponi et al., 2018) is the only one that explicitly regularizes the preservation of real-data dynamics into generation. It captures the sequential stepwise dependencies through a supervised loss imposed on the latent feature of consecutive observations. But, since it only relies on the clue provided by the consecutive observations to supervise, it may lack the efficiency to well-capture and mimic the original dynamics when the observed training data is scarce. ExtraMAE (Zha, 2022), inspired by Masked-AutoEncoders (MAEs) (He et al., 2021), learns the temporal dynamics by recovering the masked patches of the original time-series. Although its AE-based structure enables self-supervised dynamics exploration through imputation, this method lacks generation flexibility compared with GAN-based methods which directly model data distribution from random noises.

In this paper, to incorporate the generation flexibility of GAN-based sequential generation and the data-efficiency benefits of self-supervised learning, we propose Dynamic-Aware GAN (DAGAN).

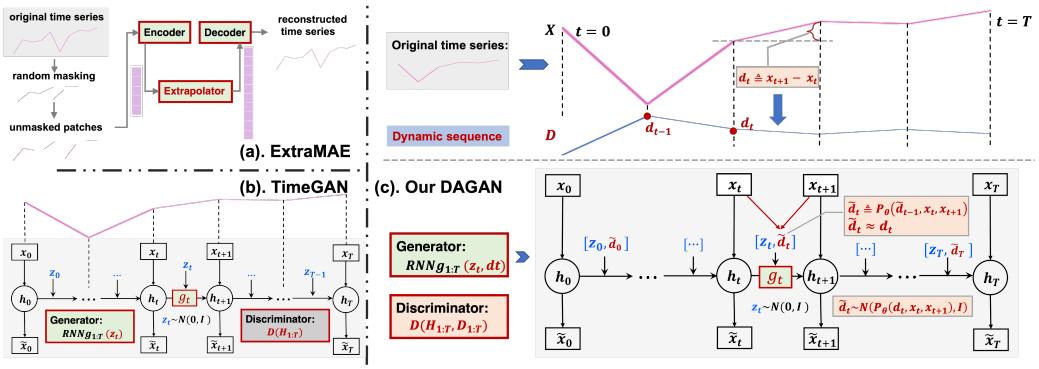


Figure 1: Model comparison of our DAGAN to ExtraMAE (AEs-based) and TimeGAN (GANs-based).

To exploit sufficient clues of the temporal dynamics, we explicitly model the stepwise transition dynamic with variable $\mathbf{D}_{1:T}$. This thus refines the traditional pairwise sequential modeling into one with inter-correlated triplets characterizing each time-step t, *i.e.* $(\mathbf{x}_t, \mathbf{x}_{t+1})$ versus $(\mathbf{x}_t, \mathbf{d}_t, \mathbf{x}_{t+1})$. Then, we establish consistency among these triplet variables at each time-step via differencing, a.k.a. $\mathbf{d}_t \triangleq \mathbf{x}_{t+1} - \mathbf{x}_t$. This contributes to a localized triplet consistent structure that provides several new aspects of clues to supervise the stepwise dependency for time-series modeling. Based on the insight, we present DAGAN, which performs dynamic consistent time-series generation on a latent feature space with our designed triplet consistency loss terms. It further benefits the capture of temporal dynamics by mimicking the second-order dynamic within data, a.k.a. the dynamic within dynamic, through recurrent transition encoding and generalizing the locally regularized triplet consistency to distribution-level via joint distribution matching on data features and dynamics. Such design also enables our DAGAN to facilitate dynamic consistent sequential generation tasks.

The contributions of this work can be summarized as follows.

- We provide a handy self-supervised mechanism for the generative modeling of time-series data. To the best of our knowledge, this is the first attempt to facilitate self-supervision by explicitly modeling the transitions, *i.e.* temporal dynamics within the sequential data.
- We propose DAGAN as a data-efficient time-series data generation model. Our model well respects the temporal dynamics implied within the original data, and more importantly, possess a speciality to perform dynamic controllable sequential generation.
- Extensive experiments on various datasets verify our model outperforms the SoTA benchmarks in generating high-quality and diverse data sequences. Our superiority is evident in both complete and limited data settings.
- We verify DAGAN's ability to perform dynamic controllable generation with several synthetic datasets. Our results demonstrate that we can get roughly consistent dynamics in the generated data sequence by controlling the stepwise conditional variable.

2 Related Work

Time-series generative modeling has witnessed great developments, due to the fast development and deployment of Deep Generative Models (DGMs) (Goodfellow et al., 2020; Hazra & Byun, 2020; Sutskever et al., 2011). Determined by the sequential property of time-series data, the key factor for successful generative modeling is whether the model can well-capture the temporal dynamics implied within the original data. To tackle the problem, existing methods mainly resort to three generation strategies. 1). *Auto-Regressive Networks (ARNs)-based* methods, *e.g.* Teacher-forcing (T-forcing) (Graves, 2013) and Professor Forcing (P-forcing) (Goyal et al., 2016), explicitly decompose the distribution of sequences into a product of conditional probabilities, and form its model under the maximum likelihood principle (Williams & Zipser, 1989; Bahdanau et al., 2017). To fit the transition dynamics, these models guide the stepwise forecasting with additional conditional signals (e.g. multi-steps of previous guesses and ground-truth outputs (Bengio et al., 2015)). Since this category of methods heavily relies on the autoregressive prior to control the generation, it lacks the flexibility to generate arbitrary sequences. 2). *AutoEncoders (AEs)-based* methods perform sequential generation via a traditional encoder-decoder structure (Wu et al., 2021). ExtraMAE (Zha, 2022)

is a recent and the most representative work of this category. This method performs explicit learning of temporal dynamics, reformulates the sequential generation problem into a masked dynamic imputation task, and trains the model to recover the original training data without extra conditioning. Such a solution would restrict the diversity of its synthesized sequences for mainly two reasons. First, under an imputation formulation, the model needs to rely on an existing pattern to synthesize new sequences (see Figure 2 in its paper). This not only reduces its usability, but also makes the model prone to memorizing the training data. Second, without extra conditioning, the model lacks the flexibility to perform arbitrary generation. 3). *Generative Adversarial Networks (GANs)-based* methods (Wang et al., 2019; Sumiya et al., 2019; Ni et al., 2020; Sun et al., 2020), pose to directly map the distribution of the original data sequences to random noise. since we can flexibly sampling new sequences by mapping from arbitrary noise samples, this category of methods has inherent advantage in performing arbitrary generation. Therefore, we make it the focus of our discussion.

To capture temporal dynamics, this category of methods seeks to guide their modules, *i.e.* the generator and discriminator, with more information, so that the model can learn a good joint probability of the sequences. CRNN-GAN (Mogren, 2016) explicitly incorporates clues of stepwise dynamic into its generator, by making the output of the previous cell as additional input. RCGAN (Esteban et al., 2017) guide its generation with more clues on data features. It introduces extra knowledge of the static feature of each sequence as an additional input of both the generator and discriminator. Inspired by these previous efforts, Yoon et al. (2019) present a general formulation for the GANs-based time-series generation problem. Consider a general setting where each sequence is characterized with two sets of features: static features (which remains unchanged over time), and temporal features (which occurs over time). Given a training dataset $\mathcal{D} = \{(\mathbf{s}_n, \mathbf{x}_{n,1:T_n})\}_{n=1}^{N}$, where $\mathbf{s}_n$ and $\mathbf{x}_n$ denote the *static feature* and *temporal feature* of the instance indexed by n, respectively. (For clarity, we omit the subscripts n unless required.) Let $\mathbf{S}$ and $\mathbf{X}$ be the random variable for the static feature and the temporal features, respectively. The tuples of the form $(\mathbf{S}, \mathbf{X}_{1:T})$ define an unknown joint distribution p. To equip a model with the ability to capture both the feature distribution and the dynamics across time, the GAN-based generation objective is modeled in two aspects.

• *Global sequence-level objective*: to learn a density $\hat{p}(\mathbf{S}, \mathbf{X}_{1:T})$ that best approximates the joint distribution $p(\mathbf{S}, \mathbf{X}_{1:T})$ that operates on the entire sequence. This objective can be denoted as

$$\min_{\hat{p}} D(p(\mathbf{S}, \mathbf{X}_{1:T}) \parallel \hat{p}(\mathbf{S}, \mathbf{X}_{1:T}))$$

• *Local stepwise objective:* to capture stepwise dynamics across time. Inspired by autoregressive decomposition of the joint distribution $p(\mathbf{S}, \mathbf{X}_{1:T}) = p(\mathbf{S})\prod_t p(\mathbf{X}_t|\mathbf{S}, \mathbf{X}_{1:t-1})$, a model also needs to encourage stepwise conditional distribution matching for each time-step t. That is,

$$\min_{\hat{p}} D(p(\mathbf{X}_t|\mathbf{S}, \mathbf{X}_{1:t-1}) \parallel \hat{p}(\mathbf{X}_t|\mathbf{S}, \mathbf{X}_{1:t-1}),$$

where D is some appropriate measure of distance between distributions modeled via discrimination.

TimeGAN, depicted in Figure 1, instantiates the second objective by explicitly supervising the generation with the stepwise dependency of the training data in a latent space of data features. Although this method is verified to be effective in generating high-quality samples, it only relies on the clue provided by the consecutive observations to supervise (pairwise at each step). Therefore, in situations where we have scarce training data (*e.g.* system crash signals), such supervision clues would be quite limited, making it hard for the model to learn and mimic the original data dynamics.

This consequently motivates us to incorporate self-supervised learning (SSL) into the exploitation of temporal dynamics within data sequences. We seek to explicitly parameterize the transition dynamic in each time-step with variable $\mathbf{d}_t \triangleq f(\mathbf{x}_{t+1}, \mathbf{x}_t)$, and exploit additional clues of stepwise dependency within such localized triplet consistent structure. In this way, we can upgrade the above GAN-based generative modeling by performing better conditional distribution matching with more supervision clues on the stepwise dynamic, thus equipping a model with a data-efficiency property, *i.e.* the ability to effectively learn and mimic temporal dynamics with a relatively small amount of data.

Compared with ExtraMAE, which enables SSL on temporal dynamics via masked dynamic imputation, our work is different and poses extra advantages. First, our self-supervision is handily exploited from the data in a stepwise manner, while ExtraMAE should learn it through a complex imputation task. This increases its computational cost. Second, our insight is formulated within the GAN-based framework, which enjoys the flexibility to perform arbitrary generation, compared with imputation-based AE models. Third, using the explicit dynamic variable for extra conditioning, we can enable controllable generation for time-series data. We experimentally verify our advantages in section 5.

3 PROBLEM FORMULATION

We explicitly parameterize the transition dynamic within data sequences with variable $\mathbf{D}$, and characterize it with the difference between consecutive observations. Therefore, the sequential observations are formed as $(\mathbf{S}, \mathbf{X}_{1:T}, \mathbf{D}_{1:T})$ with $\mathbf{d}_t \triangleq \mathbf{x}_{t+1} - \mathbf{x}_t$ for $\forall t$. Our strategy of constructing dynamic sequence as supervision is similar to the differencing technique adopted in traditional time-series forecasting models, *e.g.* Autoregressive Integrated Moving Average (ARIMA) (Gilbert, 2005).

Under the reformulation, given training data $\mathcal{D} = \{(\mathbf{s}_n, \mathbf{x}_{n,1:T_n}, \mathbf{d}_{n,1:T_n})\}_{n=1}^N$, the global and local consistent objectives of GAN-based sequential generation are defined as follows.

• ***Global sequence-level objective*:** to perform joint distribution matching on both the transition dynamic and observed features, *i.e.* $p(\mathbf{S}, \mathbf{X}_{1:T}, \mathbf{D}_{1:T})$. That is,

$$\min_{\hat{p}} D(p(\mathbf{S}, \mathbf{X}_{1:T}, \mathbf{D}_{1:T}) \parallel \hat{p}(\mathbf{S}, \mathbf{X}_{1:T}, \mathbf{D}_{1:T})). \tag{1}$$

• ***Local stepwise objective*:** to perform stepwise conditional distribution matching for each time-step t, with extra conditioning on the transition dynamic variable, denoted as

$$\min_{\hat{p}} D(p(\mathbf{X}_t|\mathbf{S}, \mathbf{X}_{1:t-1}, \mathbf{D}_{1:t-1}) \parallel \hat{p}(\mathbf{X}_t|\mathbf{S}, \mathbf{X}_{1:t-1}, \mathbf{D}_{1:t-1})) \tag{2}$$

4 DYNAMIC-AWARE GAN (DAGAN)

We construct a localized triplet consistent structure for each time-step, *i.e.* $(\mathbf{x}_t, \mathbf{d}_t, \mathbf{x}_{t+1})$ to benefit the second objective: it helps to construct new clues of stepwise dependencies and thus benefits a model with more supervision on temporal dynamics. Based on this insight, we present Dynamic-Aware GAN (DAGAN), which similar to TimeGAN, learns the data temporal dynamics via a lower-dimensional feature space, but specifically benefits the capture of temporal dynamics by mimicking the second-order dynamic within data through recurrent transition encoding and generalizing an overall locally regularized triplet consistency to distributional-level via joint distribution matching.

Our DAGAN consists of five network components: dynamic predictor P, embedding network E, recovery network R, conditional generator G and a sequence discriminator D.

4.1 DYNAMIC PREDICTION FOR LOCALIZED TRIPLET CONSISTENCY

Suppose the dynamic observations $\{\mathbf{d}_{n,1:T_n}\}_{n=1}^N$ are drawn from an unknown distribution $p_{\mathbf{D}}$, *i.e.* $\mathbf{d}_{1:T} \sim p_{\mathbf{D}}(\mathbf{d}_{1:T})$, We first build a stochastic predictor P to serve two purposes: 1). predict the transition between consecutive observations, *i.e.* $P_\theta(\mathbf{x}_t, \mathbf{x}_{t+1}) \rightarrow \tilde{\mathbf{d}}_t$, for $\forall t$; 2). learn a distribution for the dynamic variable $\mathbf{D}_{1:T}$ with q functions, *i.e.* $q_\phi(\mathbf{d}_t|\mathbf{x}_t, \mathbf{x}_{t+1}) \triangleq P_\phi(\mathbf{x}_t, \mathbf{x}_{t+1})$, where ϕ denotes the parameters of this stochastic encoding module (Kingma & Welling, 2013). We denote our learned distribution as $q_\phi(\mathbf{d}_{1:T})$ for short. We implement P with a recurrent neural network, so that

$$\tilde{\mathbf{d}}_t = P(\tilde{\mathbf{d}}_{t-1}, \mathbf{x}_t, \mathbf{x}_{t+1}), \tag{3}$$
$$\tilde{\mathbf{d}}_t \sim q_\phi(\mathbf{d}_t|\tilde{\mathbf{d}}_{t-1}, \mathbf{x}_t, \mathbf{x}_{t+1}). \tag{4}$$

This recurrent transition encoding module plays a key role in our DAGAN. It facilitates better capture of temporal dynamics in two aspects: 1). it mimics the second-order dynamic within data, *i.e.* the dynamic within dynamic sequence, through a recurrent design; 2). it bridges original data and its step-wise dynamics, enabling additional clues to supervise temporal dynamics. We define the prior over dynamic variable at each step to be centered isotropic multivariate Gaussian, *i.e.* $\mathbf{d}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.

4.2 LATENT REPRESENTATION LEARNING THROUGH AUTO-ENCODING

We perform latent representation learning for static feature $\mathbf{S}$ and temporal feature $\mathbf{X}$, with embedding network $E_\mathcal{S}$ and recurrent embedding network $E_\mathcal{X}$, respectively. Their obtained low-dimensional features are denoted as $\mathbf{h}_\mathcal{S}$ and $\mathbf{h}_{1:T} = E(\mathbf{s}, \mathbf{x}_{1:T})$ respectively, where

$$\mathbf{h}_\mathcal{S} = E_\mathcal{S}(\mathbf{s}), \qquad \mathbf{h}_t = E_\mathcal{X}(\mathbf{h}_\mathcal{S}, \mathbf{h}_{t-1}, \mathbf{x}_t), \qquad \mathbf{h}_{t+1} = E_\mathcal{X}(\mathbf{h}_\mathcal{S}, \mathbf{h}_t, \mathbf{x}_{t+1}).$$

In the opposite direction, we model the recovery network R with a feed-forward network to reconstruct the latent codes back to their feature representations *i.e.* $\hat{\mathbf{s}}, \hat{\mathbf{x}}_{1:T} = R(\mathbf{h}_\mathcal{S}, \mathbf{h}_{1:T})$, where

$$\tilde{\mathbf{s}} = R_\mathcal{S}(\mathbf{h}_s), \qquad \tilde{\mathbf{x}}_t = R_\mathcal{X}(\mathbf{h}_t), \qquad \tilde{\mathbf{x}}_{t+1} = R_\mathcal{X}(\mathbf{h}_{t+1}).$$

Note that, the implementation of E and R are flexible, given that they are autoregressive and obey causal ordering, *i.e.* output(s) at each step only depend on the preceding information.

4.3 CONDITIONAL GENERATION OF LATENT REPRESENTATIONS

We implement conditional generator G with a recurrent network, with the inputs of each cell augmented with the dynamic variable as the conditional information. Therefore, G takes the concatenation $[z_t; \mathbf{d}_t]$ at each step as input and synthesizes latent codes $\tilde{\mathbf{h}}_{\mathcal{S}}, \tilde{\mathbf{h}}_{1:T} = G(\mathbf{z}_s, \mathbf{z}_{1:T}, \mathbf{d}_{1:T})$, where

$$\hat{\mathbf{h}}_{\mathcal{S}} = G_{\mathcal{S}}(\mathbf{z}_s), \qquad \hat{\mathbf{h}}_t = G_{\mathcal{X}}(\mathbf{h}_{\mathcal{S}}, \mathbf{h}_{t-1}, \mathbf{z}_{t-1}, \mathbf{d}_{t-1}), \qquad \hat{\mathbf{h}}_{t+1} = G_{\mathcal{X}}(\mathbf{h}_{\mathcal{S}}, \mathbf{h}_t, \mathbf{z}_t, \mathbf{d}_t).$$

With these generated latent codes, we can obtain newly generated data features by recovering them back to the original feature space through the recovery network R, *i.e.* $\hat{\mathbf{s}}, \hat{\mathbf{x}}_{1:T} = R(\hat{\mathbf{h}}_{\mathcal{S}}, \hat{\mathbf{h}}_{1:T})$, where

$$\hat{\mathbf{s}} = R_{\mathcal{S}}(\hat{\mathbf{h}}_s), \qquad \hat{\mathbf{x}}_t = R_{\mathcal{X}}(\hat{\mathbf{h}}_t) \qquad \hat{\mathbf{x}}_{t+1} = R_{\mathcal{X}}(\hat{\mathbf{h}}_{t+1}).$$

The stepwise dynamic predicted for the generated data is obtained through

$$\hat{\mathbf{d}}_t = P(\hat{\mathbf{d}}_{t-1}, \mathbf{x}_t, \hat{\mathbf{x}}_{t+1}).$$

The discriminator D is also designed in a conditional manner. It takes the static codes, temporal codes and the dynamic condition as input, and returns classification $\tilde{y}_{\mathcal{S}}, \tilde{y}_{1:T} = D(\tilde{\mathbf{h}}_{\mathcal{S}}, \tilde{\mathbf{h}}_{1:T}, \tilde{\mathbf{d}}_{1:T})$, indicating whether $\tilde{\mathbf{h}}_{1:T}$ is a real latent sequence that truly reflects the dynamic implied with $\mathbf{d}_{1:T}$. The $\tilde{\mathbf{h}}_*$ notation denotes either real ($\mathbf{h}_*$) or synthetic $\hat{\mathbf{h}}_*$ latent codes; $\tilde{\mathbf{d}}_*$ denotes either real ($\mathbf{d}_*$) or the predicted $\hat{\mathbf{d}}_*$ dynamic, similarly, the $\tilde{y}_*$ notation denotes classifications of either real (y_*) or synthetic ($\hat{y}_*$) data. We implement D via a bidirectional recurrent network with a feedforward output layer,

$$\tilde{y}_{\mathcal{S}} = D_{\mathcal{S}}(\tilde{\mathbf{h}}_{\mathcal{S}}), \qquad \tilde{y}_t = D_{\mathcal{X}}(\overleftarrow{\mathcal{U}}_t, \overrightarrow{\mathcal{U}}_t),$$

where $\overleftarrow{u}_t = \overleftarrow{c}_{\mathcal{X}}(\tilde{\mathbf{h}}_{\mathcal{S}}, \tilde{\mathbf{h}}_t, \tilde{\mathbf{u}}_{t-1})$ and $\overrightarrow{u}_t = \overrightarrow{c}_{\mathcal{X}}(\tilde{\mathbf{h}}_{\mathcal{S}}, \tilde{\mathbf{h}}_t, \tilde{\mathbf{u}}_{t+1})$ denote the sequences of forward and backward hidden states, $\overleftarrow{c}_{\mathcal{X}}$; $\overrightarrow{c}_{\mathcal{X}}$ are recurrent functions; $\mathbf{d}_{\mathcal{S}}$, $\mathbf{d}_{\mathcal{X}}$ are the output of classification.

Note that, there are two synthetic paired sequences treated as "*fake*" in our DAGAN: $(\hat{\mathbf{h}}_{1:T}, \mathbf{d}_{1:T})$ and $(\mathbf{h}_{1:T}, \hat{\mathbf{d}}_{1:T})$. To distinguish them, we mark their classification result with $\hat{y}_t^h$ and $\hat{y}_t^d$, respectively.

4.4 JOINTLY LEARNING THROUGH LOCALIZED CONSISTENCY AND DISCRIMINATION

We describe the objectives to jointly train each of these aforementioned schemes in turn.

For ***dynamic prediction***, we enforce accurate prediction on stepwise dynamics via

$$\mathcal{L}_P = \mathbb{E}_{\mathbf{s},\mathbf{x}_{1:T},\mathbf{d}_{1:T}\sim p}[\textstyle\sum_t \parallel \mathbf{d}_t - P(\mathbf{d}_{t-1}, \mathbf{x}_t, \mathbf{x}_{t+1}) \parallel_2] \tag{5}$$

For ***dynamic distribution learning***, we adopt a KL-divergence (Federici et al., 2020) loss term,

$$\mathcal{L}_P^{dist} = \mathbb{E}_{\mathbf{d}_{1:T}\sim p_{\mathbf{D}}}[\mathrm{KL}(q_\phi(\mathbf{d}_t) \parallel p_D(\mathbf{d}_t))]. \tag{6}$$

For ***feature representation learning***, we regularize data feature reconstruction with

$$\mathcal{L}_R^{\mathbf{x}} = \mathbb{E}_{\mathbf{s},\mathbf{x}_{1:T},\mathbf{d}_{1:T}\sim p}[\parallel \mathbf{s} - \tilde{\mathbf{s}} \parallel_2 + \textstyle\sum_t \parallel \mathbf{x}_t - \tilde{\mathbf{x}}_t \parallel_2], \tag{7}$$

$$\mathcal{L}_R^{\mathbf{d}} = \textstyle\sum_t \parallel \tilde{\mathbf{x}}_{t+1} - \tilde{\mathbf{x}}_t - \mathbf{d}_t \parallel_2], \tag{8}$$

where $\mathcal{L}_R^{\mathbf{d}}$ regularizes the feed-forwardly reconstructed data feature to preserve the dynamic implied with the original $\mathbf{d}_t$ at each time-step.

For ***dynamic-conditional latent feature generation***, we have

$$\mathcal{L}_S^{\mathbf{h}} = \mathbb{E}_{\mathbf{s},\mathbf{x}_{1:T},\mathbf{d}_{1:T}\sim p}[\textstyle\sum_t \parallel \mathbf{h}_t - G(\mathbf{h}_{\mathcal{S}}, \mathbf{h}_{t-1}, \mathbf{z}_t, \mathbf{d}_t) \parallel_2] \tag{9}$$

For ***joint learning*** to encode data features, predict dynamics and iteratively generate, we have

$$\mathcal{L}_S^{\mathbf{d}} = \mathbb{E}_{\mathbf{s},\mathbf{x}_{1:T},\mathbf{d}_{1:T}\sim p}[\textstyle\sum_t \parallel \mathbf{d}_t - P(\mathbf{d}_{t-1}, \mathbf{x}_t, \hat{\mathbf{x}}_{t+1}) \parallel_2]. \tag{10}$$

The ***discrimination*** loss term designed for training D is given as

$$\begin{aligned} \mathcal{L}_D \quad = \quad & \mathbb{E}_{\mathbf{s},\mathbf{x}_{1:T},\mathbf{d}_{1:T}\sim p}[\log y_{\mathcal{S}} + \textstyle\sum_t \log y_t^{\mathbf{h}} + \sum_t \log y_t^{\mathbf{d}}] \\ & + \mathbb{E}_{\mathbf{s},\mathbf{x}_{1:T},\mathbf{d}_{1:T}\sim \hat{p}}[\log(1 - \hat{y}_{\mathcal{S}}) + \textstyle\sum_t \log(1 - \hat{y}_t^{\mathbf{h}}) + \sum_t \log(1 - \hat{y}_t^{\mathbf{d}})] \end{aligned} \tag{11}$$

Note that, among all these loss terms, Eq. 5, Eq. 8, Eq. 10, are supervision of temporal dynamics induced by our explicitly modeling of transition dynamics, as stated in Eq. 3 and Eq. 4.

Optimization: Let θ_P, θ_E, θ_R, θ_G and θ_D represent the parameters of the dynamic predictor P, embedding network E, recovery network R, latent code generator G and sequence discriminator D. P is trained with loss terms of dynamic prediction (Eq. 5), dynamic distribution learning (Eq. 6), dynamic-conditional latent feature generation (Eq. 10), and joint training (Eq. 9), *i.e.*

$$\min_{\theta_P}(\lambda\mathcal{L}_S^{\mathbf{h}} + (\mathcal{L}_P + \mathcal{L}_P^{dist} + \mathcal{L}_S^{\mathbf{d}})) \tag{12}$$

where λ is a hyperparameter that balances the loss terms related to the stochastic dynamic encoding and dynamic-conditioned generation. We then train E and R are trained with the loss terms of feature representation learning (Eq. 7, Eq. 8) and joint training (Eq. 9). The objective function is

$$\min_{\theta_E,\theta_R}(\eta\mathcal{L}_S^{\mathbf{h}} + (\mathcal{L}_R^{\mathbf{d}} + \mathcal{L}_R^{\mathbf{x}})) \tag{13}$$

where η is the hyperparameter that balances the two sets of loss terms.

The sequence generator G discriminator D are adversarially train through

$$\min_{\theta_G}(\gamma\mathcal{L}_S^{\mathbf{h}} + \max_{\theta_D}\mathcal{L}_D) \tag{14}$$

where γ is another hyperparameter that balances the two loss terms for G and D.

Training details: DAGAN is trained by the Adam Optimizer (Kingma & Ba, 2014). The coefficients of the loss terms are set as: $\lambda = \eta = 0.1$, and $\gamma = 10$ to highlight the importance of the supervised loss. The other parameters, such as batch size and number of iterations, require tuning for different datasets. We provide more details of our DAGAN in the Supplementary.

5 EXPERIMENTS

To evaluate the quality of the generated data, we focus on three main criteria.

Fidelity: meaning that the generated data should retain the characteristics of the original data, thus making them indistinguishable from the latter. We can quantitatively evaluate this via the **discriminative score**. We define the score to be the classification accuracy of a classifier (a 2-layer LSTM model) that discriminates between original data labelled *'real'*, and synthetic data labelled *'fake'*. When presented with synthetic data of high fidelity, the discriminator would likely be fooled and thus have low accuracy. Thus, a *low* discriminative score is indicative of faithful synthetic data.

Usefulness: Useful or practical characteristics of the original data, most notably their predictive properties, should be preserved in the synthetic data. In this manner, synthetic data could be employed in practical applications to augment limited data for predictive purposes. As a quantitative measure, the **predictive score** is calculated. Under the *Train on Synthetic, Test on Real* (TSTR) framework (Esteban et al., 2017), a predictor (a 2-layer LSTM) is trained on the synthetic dataset to predict the values of the sequence at the next time-step. The predictive score is defined to be the mean absolute error of the prediction model when tested on the original dataset. A *low* predictive score suggests that the generated data retains predictive properties from the original data.

Diversity: Finally, good generated data should be sufficiently diverse to cover the distribution of the real data. PCA (Bryant & Yarnold, 1995) and t-SNE (Van der Maaten & Hinton, 2008) plots containing the original and generated data, provide a qualitative assessment of diversity. Diverse synthetic data should show significant overlap with the real data.

With these criteria as the basis of evaluation, DAGAN is tested on a range of datasets with varying properties, *e.g.* different periods, temporal and feature correlations, number of feature dimensions and noise level. We also vary the size of training data to simulate limited data settings.

5.1 AUTOREGRESSIVE GAUSSIAN MODELS WITH LINEAR TEMPORAL DYNAMICS

To underscore our advantage of capturing complex temporal dynamics via conditioning on the additional dynamic variable, we implement DAGAN on sequences simulated from multivariate

autoregressive (AR) Gaussian models of the forms $\mathbf{x}_t = \phi\mathbf{x}_{t-1} + \mathbf{n}$, where $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \sigma\mathbf{1} + (1-\sigma)\mathbf{I}$, and $\phi \in [0, 1]$ is the temporal correlation coefficient and $\sigma \in [-1, 1]$ is the feature correlation coefficient. This particular model is appropriate for closer scrutiny because we can study the performance of DAGAN on datasets of different temporal dynamics by varying ϕ. The results in Table 1 show that DAGAN performance improves when there is a stronger temporal connection between the time-steps, as supported by the falling discriminative scores for higher values of ϕ. Compared to TimeGAN, which similarly claims to leverage supervision to model transition dynamics, our proposed model shows a considerable decrease in discriminative score in most settings. This supports our incorporation of $\mathbf{d}_t$ as additional self-supervision information to benefit time-series generation tasks.

Table 1: Comparison of DAGAN (1st row) and TimeGAN (2st row) regarding discriminative scores on AR toy datasets of different ϕ and σ.

Correlation	$\phi = 0.2$	$\phi = 0.5$	$\phi = 0.8$
$\sigma = 0.2$	**.379 ± .058**	**.364 ± .028**	**.156 ± .010**
	.499 ± .001	.494 ± .003	.378 ± .033
$\sigma = 0.5$	**.466 ± .015**	**.349 ± .045**	**.226 ± .020**
	.500 ± .000	.489 ± .010	.314 ± .034
$\sigma = 0.8$	.497 ± .004	**.485 ± .009**	.340 ± .024
	.497 ± .001	.493 ± .493	**.306 ± .056**

5.2 Results Comparison of Sequence Generation

On top of some toy datasets, we focus mainly on the following three datasets:

Stocks: Google's historical stocks data from 2004 to 2019 are extracted to obtain continuous-valued sequences (sto, 2021). The dataset included are volume and high, low opening, closing, and adjusted closing prices. This data is aperiodic and the five features relating to price, are highly correlated.

Sine: We simulate five-dimensional sinusoidal sequences from the sine function, *i.e.* for $i \in \{1, \ldots, 5\}$, we obtain $x_i(t) = \sin(2\pi ft + \phi)$, where $f \sim \mathcal{U}[0, 1]$ denotes frequency, and $\phi \sim \mathcal{U}[-\pi, \pi]$ denotes phase. The sine dataset exhibits fixed periods and amplitudes for each sequence, with the features independent of each other.

Energy: For a more challenging task with high-dimensional and noisy data, the energy dataset is used (ene, 2017). This dataset monitors energy consumption and various related variables over time, including temperature and humidity within the building. The data displays some seasonality but has a high degree of noise compared to the sine data.

5.2.1 Complete Data Setting

From Table 2, we observe that DAGAN consistently outperforms TimeGAN across all the datasets in terms of discriminative and predictive scores. In comparison to ExtraMAE, DAGAN shows better results for the sine dataset and is comparable on the stocks data. In particular, the performance of DAGAN shows the most improvement on the sine dataset with the discriminative score 60% and 93%, and the predictive score 5% and 52% lower than TimeGAN and ExtraMAE respectively. Although ExtraMAE's performance is on par with DAGAN on the stocks dataset, the former is more computationally intensive, taking on average approximately 50% longer to train on the same dataset.

Table 2: Discriminative and predictive scores of DAGAN and baseline methods.

Metric	Method	Sine	Stocks	Energy
Discriminative Score	DAGAN	.006 ± .004	.085 ± .044	.439 ± .013
	TimeGAN	.015 ± .010	.192 ± .023	.489 ± .005
	ExtraMAE	.090 ± .063	.074 ± .118	.500 ± .000
Predictive Score	DAGAN	.093 ± .000	.038 ± .000	.328 ± .006
	TimeGAN	.098 ± .002	.039 ± .000	.314 ± .003
	ExtraMAE	.193 ± .006	.037 ± .000	.289 ± .006
	Original	.094 ± .001	.036 ± .001	.250 ± .003

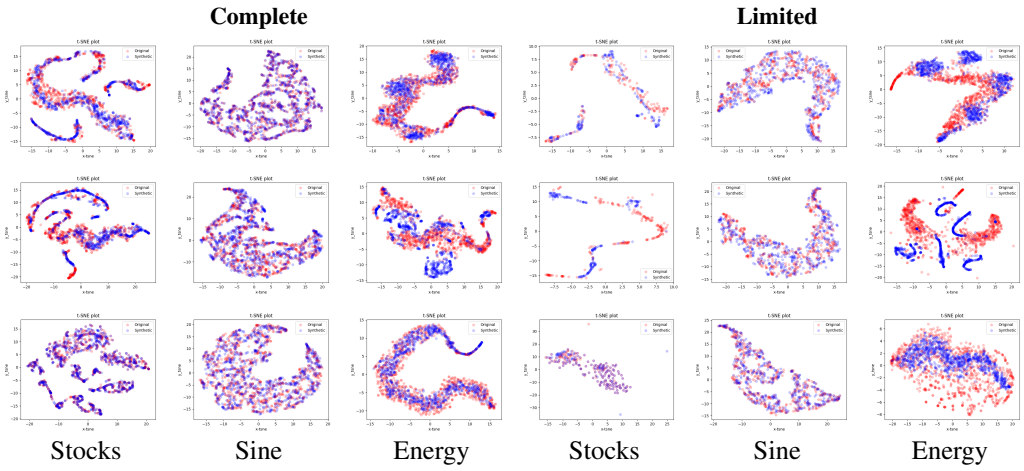


Figure 2: t-SNE visualization by DAGAN (1st row), TimeGAN (2nd row) and ExtraMAE (3rd row) for the complete (left) and limited (right) datasets. Each column is the results for Stock, Sine, Energy.

5.2.2 Limited Data Setting

We testify the benefits of leveraging self-supervised information on transition dynamics when training data is scarce. In this section, we demonstrate the performance of DAGAN under a limited data setting, where the datasets were decreased in size by 20 times, and compare it with the SOTA methods. For example, the stocks dataset was randomly reduced from more than 3600 to under 200 observations. The discriminative and predictive scores are shown in Table 3.

Metric	Method	Sine	Stocks	Energy
Discriminative Score	DAGAN	.036 ± .029	.119 ± .059	.479 ± .015
	TimeGAN	.051 ± .049	.349 ± .064	.494 ± .004
	ExtraMAE	.205 ± .110	.138 ± .124	.498 ± .003
Predictive Score	DAGAN	.096 ± .001	.044 ± .000	.291 ± .005
	TimeGAN	.097 ± .002	.043 ± .001	.294 ± .006
	ExtraMAE	.188 ± .001	.033 ± .000	.299 ± .010
	Original	.094 ± .000	.037 ± .001	.253 ± .001

Table 3: Discriminative and predictive scores of DAGAN and baseline methods when trained on limited data.

Method	**MMD**(X, Y)	**MMD**(X, Z)
DAGAN	.002689	.002503
TimeGAN	.001302	.006861
ExtraMAE	.000021	.002467

Table 4: MMD scores of (limited) stock dataset between X, Y, and Z.

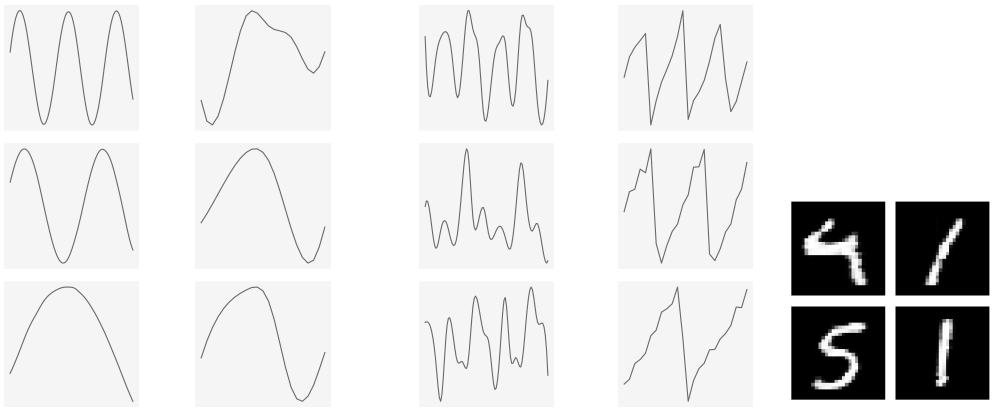
In both settings, and especially the limited data case, DAGAN displays good overlap in the t-SNE plots (see Figure 2). Although ExtraMAE seems to exhibit even more preferable overlap, for example, on the energy data, we argue that this can be caused by its proneness to memorize the training data, since it obtains the generated data through imputation. We assess this by evaluating the Maximum Mean Discrepancy (MMD) scores (Gretton et al., 2006) between the training data (X) and synthetic data (Y), and between the training data (X) and a test set (Z). These scores give an indication of whether the synthetic data is too similar to the training data, *i.e.* simply memorized them. Table 4 shows the MMD scores on the limited stocks dataset. We found that for ExtraMAE, MMD(X, Z) was much higher than MMD(X,Y) while the two scores were more comparable for our DAGAN. This means that ExtraMAE suffers low generation diversity compared with DAGAN.

5.3 Dynamic Conditional Sequence Generation

Since DAGAN is constructed with a dynamic-consistent design, we expect, to some extent, it can conditionally generate sequences displaying characteristics of the supplied dynamics. Here, we test conditional generation on models trained on several toy datasets by varying the conditions $\mathbf{d}_t$. Furthermore, as the conditions are a function of the differences between time-steps, they are interpretable. Thus, we can evaluate the quality of the synthetic datasets by visual inspection to determine whether the sequences are faithful to the original, and whether the conditions are well-captured.

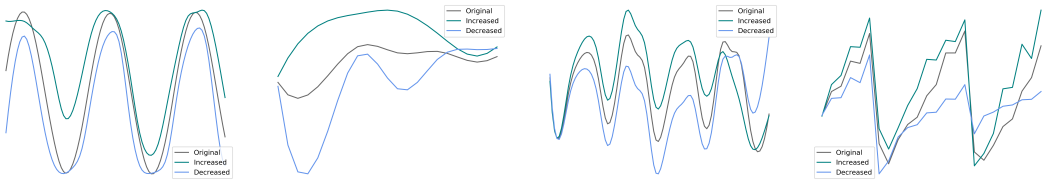
Figure 3 presents generated samples from various toy datasets of increasing complexity: (a) simple sine curves with fixed periods and amplitudes; (b) Modified sine curves with changing periods and amplitudes; (c) Smooth signals exhibiting some seasonality; (d) Sawtooth-shaped Fourier series; and lastly (e) MNIST dataset by treating each $28x28$ image as 28-dimensional sequences of length 28. The generated sequences resemble the original data, demonstrating that DAGAN is capable of performing faithful reconstruction of data, conditioning on the original dynamic.

Subsequently, to investigate the effects of the conditional dynamic variable on the generated data, we vary $\mathbf{d}_t$ by increasing or decreasing it with a fixed constant. The results, shown in Figure 4, indicate our generated sequences maintain the expected characteristics, and also capture the conditions supplied. For instance, the sequences conditioned on the increased $\mathbf{d}_t$, signifying an increased (more positive) difference at each time-step, tend to lie above the sequence generated based on the original condition. This lends credence to our insight of utilizing $\mathbf{d}_t$ for conditional sequence generation.



(a) Sine (b) Modified sine (c) Smooth signals (d) Fourier (e) MNIST

Figure 3: Samples generated by DAGAN trained on multiple toy datasets.



(a) Sine (b) Modified sine (c) Smooth signals (d) Fourier

Figure 4: Generated samples conditioned on the original, increased and decreased $\mathbf{d}_t$. The increased and decreased $\mathbf{d}_t$ were obtained by adding and subtracting a constant value from the original.

5.4 Ablation Study

We conduct ablation studies to analyze the importance of performing joint distribution matching in DAGAN. Our result suggests that data-dynamic joint distribution matching plays a crucial role in improving the quality of the synthetic data of DAGAN. We present comparison results between DAGAN and its variant without the joint distribution matching design in the Supplementary.

6 Conclusion

In this paper, we facilitate data-efficient GAN-based time-series generation with handy self-supervised information provided by the differencing sequence. Modeling a distribution for dynamic and enforcing a generalized local triplet consistency, our DAGAN mimic the second-order information, *i.e.* the dynamic of dynamic, encoded within the data sequence to capture temporal dynamic. This makes our model superior to capture and mimic original data dynamics with limited training data. Moreover, DAGAN's speciality in performing dynamic-controllable generation makes it of great utility to generate sequences with specified dynamics in real applications.

REFERENCES

Energy dataset, 2017. URL https://archive.ics.uci.edu/ml/datasets/Appliances+energy+prediction.

Stocks dataset, 2021. URL https://github.com/jsyoon0823/TimeGAN/blob/master/data/stock_data.csv.

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron C. Courville, and Yoshua Bengio. An actor-critic algorithm for sequence prediction. In *ICLR*. OpenReview.net, 2017.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *NeurIPS*, pp. 1171–1179, 2015.

Fred B Bryant and Paul R Yarnold. Principal-components analysis and exploratory and confirmatory factor analysis. 1995.

Cristóbal Esteban, Stephanie L. Hyland, and Gunnar Rätsch. Real-valued (medical) time series generation with recurrent conditional gans. *CoRR*, abs/1706.02633, 2017.

Marco Federici, Anjan Dutta, Patrick Forré, Nate Kushman, and Zeynep Akata. Learning robust representations via multi-view information bottleneck. In *ICLR*. OpenReview.net, 2020.

Kenneth Gilbert. An arima supply chain model. *Management Science*, 51(2):305–310, 2005.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

Anirudh Goyal, Alex Lamb, Ying Zhang, Saizheng Zhang, Aaron C. Courville, and Yoshua Bengio. Professor forcing: A new algorithm for training recurrent networks. In *NeurIPS*, pp. 4601–4609, 2016.

Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

Arthur Gretton, Karsten Borgwardt, Malte Rasch, Bernhard Schölkopf, and Alex Smola. A kernel method for the two-sample-problem. *Advances in neural information processing systems*, 19, 2006.

Debapriya Hazra and Yung-Cheol Byun. Synsiggan: Generative adversarial networks for synthetic biomedical signal generation. *Biology*, 9(12):441, 2020.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross B. Girshick. Masked autoencoders are scalable vision learners. *CoRR*, abs/2111.06377, 2021.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

Olof Mogren. C-RNN-GAN: continuous recurrent neural networks with adversarial training. *CoRR*, abs/1611.09904, 2016.

Hao Ni, Lukasz Szpruch, Magnus Wiese, Shujian Liao, and Baoren Xiao. Conditional sig-wasserstein gans for time series generation. *CoRR*, abs/2006.05421, 2020.

Giorgia Ramponi, Pavlos Protopapas, Marco Brambilla, and Ryan Janssen. T-CGAN: conditional generative adversarial network for data augmentation in noisy time series with irregular sampling. *CoRR*, abs/1811.08295, 2018.

Yuki Sumiya, Kazumasa Horie, Hiroaki Shiokawa, and Hiroyuki Kitagawa. Nr-gan: Noise reduction gan for mice electroencephalogram signals. In *CBSP*, pp. 94–101, 2019.

He Sun, Zhun Deng, Hui Chen, and David C Parkes. Decision-aware conditional gans for time series data. *arXiv preprint arXiv:2009.12682*, 2020.

Ilya Sutskever, James Martens, and Geoffrey E Hinton. Generating text with recurrent neural networks. In *ICML*, 2011.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *JMLR*, 9(11), 2008.

Lu Wang, Wei Zhang, and Xiaofeng He. Continuous patient-centric sequence generation via sequentially coupled adversarial learning. In *DASFAA*, volume 11447 of *Lecture Notes in Computer Science*, pp. 36–52. Springer, 2019.

Ronald J. Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Comput.*, 1(2):270–280, 1989.

Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *NeurlPS*, 34:22419–22430, 2021.

Jinsung Yoon, Daniel Jarrett, and Mihaela Van der Schaar. Time-series generative adversarial networks. *NeurlPS*, 32, 2019.

Mengyue Zha. Time series generation with masked autoencoder. *CoRR*, abs/2201.07006, 2022.