Time-series Transformer Generative Adversarial Networks

Padmanaba Srinivasan
Department of Computing
Imperial College London
London, SW7 2AZ, UK
ps3416@ic.ac.uk

William J. Knottenbelt
Department of Computing
Imperial College London
London, SW7 2AZ, UK
wjk@ic.ac.uk

Abstract

Many real-world tasks are plagued by limitations on data: in some instances very little data is available and in others, data is protected by privacy enforcing regulations (e.g. GDPR). We consider limitations posed specifically on time-series data and present a model that can generate synthetic time-series which can be used in place of real data. A model that generates synthetic time-series data has two objectives: 1) to capture the stepwise conditional distribution of real sequences, and 2) to faithfully model the joint distribution of entire real sequences. Autoregressive models trained via maximum likelihood estimation can be used in a system where previous predictions are fed back in and used to predict future ones; in such models, errors can accrue over time. Furthermore, a plausible initial value is required making MLE based models not really *generative*. Many downstream tasks learn to model conditional distributions of the time-series, hence, synthetic data drawn from a generative model must satisfy 1) in addition to performing 2). We present TsT-GAN, a framework that capitalises on the Transformer architecture to satisfy the desiderata and compare its performance against five state-of-the-art models on five datasets and show that TsT-GAN achieves higher predictive performance on **all** datasets.

1 Introduction

Many real world applications are reliant on time-series data, however, not all these applications necessarily have available the data they need. For some tasks, data is available only in small quantities – often too small to base detailed analysis on – and for others, data is protected by regulatory or ethical concerns . Medicine is one field that is notorious for suffering from such problems and the ability to generate synthetic data is an avenue through which analysis can continue [32, 49, 13, 5]. Another field with similar issues is data from human-internet interactions; as world governments have increasingly adopted GDPR-like legislation, the availability of such data is limited and methods that can generate synthetic data as a stand-in for user data will increase in importance. Although we do not expressly use privacy-preserving methods in training, this property can be achieved using methods such as differential privacy-preserving stochastic gradient descent [15, 1].

Good quality synthetic time-series data should respect the conditional distribution of a time-series $\prod_{t=1}^{T} p(\mathbf{x}_t \mid \mathbf{x}_1, ..., \mathbf{x}_t)$, so as to maximise utility for downstream models. Generated synthetic data should also capture well the joint distribution $p(\mathbf{x}_1, ..., \mathbf{x}_T)$ such that the synthetic data is indistinguishable from real data.

A straightforward, if naïve approach to generating synthetic data is to use autoregressive models trained using teacher forcing [22, 42] by repeatedly feeding it past predictions. Conditioning on

Preprint. Under review.

previous outputs is prone to errors adding up over the course of a sequence and techniques to overcome this [6, 19, 30] have not entirely solved the problem [51].

Generative models are statistical models that learn from a set of data instances, X, and their corresponding labels, Y to capture the joint probability distribution $p(X, Y)$. Learning the joint distribution allows generative models to generate new data instances. Generative Adversarial Networks (GANs) are one method of training generative models [21] that typically map some noise, z, to a posterior distribution. Specifically, for time-series, GANs incorporate an additional temporal dimension to model the joint distribution of all elements of the time-series. Synthetically generated sequences have a wide array of applications [11, 23, 4], yet in many existing approaches the true time-series dynamics are not explicitly learned and the quality of how well these dynamics are learned are not explored in detail.

Contributions We use the two characteristics of good generative models to guide the development of a new architecture that that contains a generator that can model the full joint distribution while also respecting the need for accurate conditional distributions. We develop a training framework for our model that can be applied to any time-series dataset and benchmark our method quantitatively using the *train on synthetic test on real* (TS-TR) [17] approach, and qualitatively using t-SNE [43]. We compare our model against five state-of-the-art baselines on five datasets and show that TsT-GAN consistently achieves superior performance, achieving the best predictive scores across the board while also demonstrating best discriminative performance on three out of five datasets. We also perform ablation experiments on TsT-GAN to identify sources of performance gain.

2 Related Work

Generative Adversarial Networks A GAN model consists of two components: firstly, a generator that maps noise to a posterior distribution; and a discriminator whose job it is to distinguish between samples produced by the generator and samples drawn from the dataset. Some popular applications of GANs focus on generating high-dimensional data, such as images [2, 31, 38] and videos [36, 10, 44]. Developments have also been made to make the notoriously unstable process of training GANs more stable [2, 27, 33, 37, 41].

The output of a generator can be controlled by conditioning on additional information. This class of GANs are called conditional GANs (cGANS) [34] where the generator takes as an additional input some information to direct the generative process. cGANs have been applied to generate sequential data in a number of fields, such as, video clips [10, 36, 44], time-series data [17, 35, 51], natural language tasks [39, 50, 53] and generating tabular data [9, 18, 28, 47]. Generating new data is a task well suited to GANs – the generator is trained to generate data by learning the underlying generating distribution and guided by the discriminator. Once trained, a generator can be used to generate synthetic data samples for cases where limited real data is available.

Time-Series Generative Models Generating synthetic time-series data extends the generation of synthetic tabular data by incorporating an additional temporal dependency. This tasks generative models with learning features of the data within each time step and also relating features across time.

One approach to designing generative models for time-series generation is Professor Forcing [30] which combines a GAN framework with a supervised learning approach where a Recurrent Neural Network (RNN) based generator [25, 8] alternates between teacher forcing [22, 42] and generative training and the discriminator distinguishes between hidden states produced in teacher forcing mode and free running mode. This encourages the generator to match the conditional distributions between the two modes.

The C-RNN-GAN framework [35] directly applies Long short-term memory (LSTM) [22] neural network in both generator and discriminator to generate sequential data. The generator receives noise inputs at each time step and generates some data, conditioned on previous outputs. RCGAN [17] extends this approach by allowing conditioning on additional information while also removing the dependence on previous outputs.

TimeGAN [51] modifies the standard GAN framework and adopts aspects of Professor Forcing. The framework uses four components: an embedder and decoder (trained via teacher forcing as a joint autoencoder network), a generator and a discriminator. The generator receives noise input and produces hidden states which are passed to the discriminator along with the hidden states of

the embedder, which discriminates between the embedder and generator latent distributions. An additional supervised loss penalises differences between the two distributions. COT-GAN [48] presents a flexible GAN architecture trained using a novel adversarial loss that builds on the Sinkhorn divergence [20].

3 Method

3.1 Problem Formulation

We denote a multivariate sequence of length T as $\mathbf{x}_{1:T} = \mathbf{x}_1, ..., \mathbf{x}_T$ with N such sequences form the training dataset $\mathcal{D} = \{\mathbf{x}_{1:T}\}_{n=1}^{N}$. The aim of the generator is to learn a distribution $\hat{p}(\mathbf{x}_{1:T})$ as an approximation to the generating distribution $p(\mathbf{x}_{1:T})$. Learning the conditional distribution $\prod_t p(\mathbf{x}_t \mid \mathbf{x}_1, ..., \mathbf{x}_{t-1})$ is a far simpler objective learned by single time-step autoregressive models.

Any synthetic data generated using a trained generator is likely to be used downstream to train autoregressive models. As a result, in addition to learning the joint distribution $p(\mathbf{x}_{1:T})$ the generator must also learn $\prod_t p(\mathbf{x}_t \mid \mathbf{x}_1, ..., \mathbf{x}_{t-1})$. The joint distribution suggests that the generative model can be bidirectional, whereas an autoregressive constraint is explicit in the conditional distribution. To this end, we propose two objectives. The first is to learn the joint distribution:

$$\min_{\hat{p}} \; D(p(\mathbf{x}_{1:T}) \;||\; \hat{p}(\mathbf{x}_{1:T})) \tag{1}$$

Which we interpret as a global objective where the learned joint distribution must match the true joint distribution. We also incorporate the conditional distribution:

$$\min_{\hat{p}} \; D(p(\mathbf{x}_t \mid \mathbf{x}_{1:t-1}) \;||\; \hat{p}(\mathbf{x}_t \mid \mathbf{x}_{1:t-1})) \tag{2}$$

where D represents suitable distance metrics for the two objectives. This represents a local (autoregressive) objective where each item is conditioned on previous ones.

3.2 Proposed Model

We present our TsT-GAN model which consists of four components, each designed with respect to the objectives in Section 3.1.

3.2.1 Embedder–Predictor

The embedder–predictor network consists of a transformer network that takes as input real multivariate sequences $\mathbf{x}_{1:T}$ and predicts the next item in the sequence at each position. This network consists of a linear projection of the input vector into the model dimension. The projected sequence is passed through the embedder network E_θ to produce the set of final embeddings $\hat{\mathbf{h}}_1, ..., \hat{\mathbf{h}}_T$. The predictor network P_θ takes the embeddings back into the original input dimension $P_\theta : \mathbb{R}^d \to \mathbb{R}^m$ and is implemented by a separate neural network. Applying the predictor network to all the embeddings produced by the embedding network generated one step ahead predictions for all positions in the input sequence $\hat{\mathbf{x}}_1, ..., \hat{\mathbf{x}}_T$.

The embedding network is parameterised by a transformer decoder network, which uses an autoregressive mask and we realise the predictor network as a simple linear layer that performs a linear projection of the embedding back into the original data input dimension.

The embedder–predictor network is trained using a supervised loss that penalises one step ahead prediction error:

$$\mathcal{L}_S(\mathbf{x}_{1:T}, \hat{\mathbf{x}}_{1:T}) = \frac{1}{T-1} \sum_{t=1}^{T-1} ||\mathbf{x}_{t+1} - \hat{\mathbf{x}}_t||_2 \tag{3}$$

where $\hat{\mathbf{x}}_t$ denotes the prediction by the embedder–predictor network at position t and that corresponds to the predicted value of the time-series at time $t+1$. This allows learning of the true conditional

distribution $\prod_t p(\mathbf{x}_t \mid \mathbf{x}_{1:t-1})$ with the embedder modelling the latent conditional distribution $\prod_t p(\hat{\mathbf{h}}_t \mid \hat{\mathbf{h}}_{1:t-1})$.

3.2.2 Generator and Discriminator

The generator model G_θ takes a sequence of random vectors $\mathbf{z}_1, ..., \mathbf{z}_T$ and projects these into the model dimension. The projected noise vector is passed through G_θ which outputs a set of latent embeddings $\tilde{\mathbf{h}}_1, ..., \tilde{\mathbf{h}}_T$. Each latent embedding $\tilde{\mathbf{h}}_t \in \mathbb{R}^d$ is then transformed back into the original input space by way of the predictor network from Section 3.2.1 to produce a synthetic sequence $\tilde{\mathbf{x}}_{1:T} = \tilde{\mathbf{x}}_1, ..., \tilde{\mathbf{x}}_T$. Parameters of the predictor network are shared between the generator and embedder.

We construct G_θ in a similar way to the embedding network; it consists of a transformer encoder that makes use of bidirectional attention. To enforce the autoregressive property, we allow the parameters of the predictor network to be updated only when performing backpropagation through the embedder–predictor network. When backpropagating through the generator-predictor network, gradients are calculated but the parameters of the predictor network are frozen. This forces the generator to learn the latent conditional distributions of the embedder to produce valid synthetic data while also allowing full treatment of the joint probability. The random vectors fed into the generator can be drawn from any distribution, we draw random vectors from a standard Gaussian distribution.

The discriminator model D_θ is constructed in a similar way to BERT [14] as a transformer encoder with bidirectional attention. A linear projection is used to map input sequences to the model dimension following which a `[CLS]` embedding is prepended to the beginning of the sequence. This sequence is passed through the discriminator and the embedding corresponding to the `[CLS]` position is projected into $\mathbb{R}^1$ for classification. The discriminator receives as input real sequences drawn from the dataset, which it is tasked with classifying as *true*, and synthetic sequences from the generator, which it must classify as *false*. Our discriminator design focuses on a global classification of the quality of a sequence, which differs from previous RNN based approaches which classify on a per time step basis. By performing global sequence classification with the discriminator, we address our first objective in Equation 1, while the stepwise objective in Equation 2 is handled indirectly via the embedder–predictor system. We apply the LS-GAN adversarial loss [33], which uses separate objectives for the discriminator and generator:

$$\begin{aligned} \mathcal{L}_{GAN}(D_\theta) &= \min_{D_\theta} \ \mathbb{E}_{\mathbf{x}_{1:T}\sim p}[((D_\theta(\mathbf{x}_{1:T}) - 1)^2] + \mathbb{E}_{\tilde{\mathbf{x}}_{1:T}\sim \hat{p}}[(D_\theta(\tilde{\mathbf{x}}_{1:T})^2] \\ \mathcal{L}_{GAN}(G_\theta) &= \min_{G_\theta} \ \frac{1}{2}\mathbb{E}_{\tilde{\mathbf{x}}_{1:T}\sim \hat{p}}[(D_\theta(\tilde{\mathbf{x}}_{1:T}) - 1)^2] \end{aligned} \tag{4}$$

The GAN objective is likely to be insufficient to fully capture the temporal dependencies across long time periods [26]. Applying a second, supervised autoregressive loss on the generator is not possible as the generator is bidirectional, so we turn to unsupervised masked training. Following a similar method to masked language modelling (MLM) in BERT [14], we randomly mask out items in a sequence with probability p_{mask}. Masked out positions are replaced with a learnable `[MASK]` embedding and the generator is tasked with predicting the true values at the masked positions. Transformers have been applied to autoregressive time-series tasks [52, 45, 46] and have been shown to benefit from masked modelling pre-training [52]. We add the following masked modelling objective:

$$\mathcal{L}_{MM}(\mathbf{x}_{1:T}, \overline{\mathbf{x}}_{1:T}) = \frac{1}{|M|}\sum_{t\in M} ||\mathbf{x}_t - \overline{\mathbf{x}}_t||_2 \tag{5}$$

where M denotes the masked positions and $\overline{\mathbf{x}}_t$ is the output of the generator at position t when performing masked modelling. We perform full generation and masked modelling in an alternating fashion using separate learnable linear projections for both tasks.

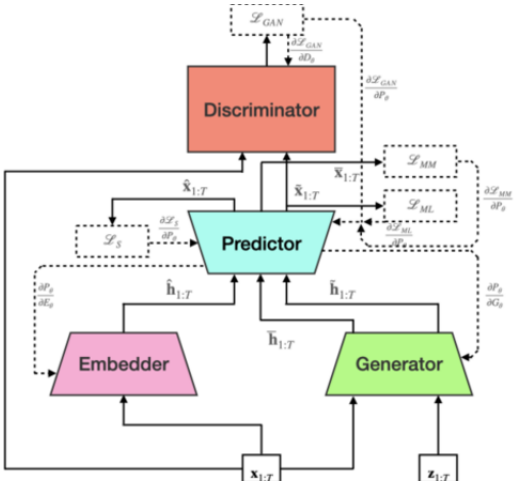


Figure 1: Block diagram of TsT-GAN showing data and gradient flows. Gradients that propagate back to the generator will pass through the predictor network, but, will not be allowed to change the parameters of the predictor. Predictor parameters are only updated with respect to gradient $\frac{\partial \mathcal{L}_S}{\partial P_\theta}$.

3.3 Architecture Overview

Given that $\mathbf{x}_{1:T} = \mathbf{x}_1, ..., \mathbf{x}_T$ denotes a true time-series of length T and $\mathbf{z}_{1:T} = \mathbf{z}_1, ..., \mathbf{z}_T$ denotes a sequence of random vectors drawn from an isotropic Gaussian, we provide formal motivation for our model, with an accompanying block diagram in Figure 1. The embedder, trained using maximum likelihood estimation (MLE) takes as input $\mathbf{x}_{1:T}$ and produces conditional latent embeddings $\hat{\mathbf{h}}_{1:T} = \hat{\mathbf{h}}_1, ..., \hat{\mathbf{h}}_T$ which the predictor maps to values $\hat{\mathbf{x}}_{1:T}$.

The generator takes as input a sequence of random vectors $\mathbf{z}_{1:T}$ and produces a corresponding set of latent embeddings $\tilde{\mathbf{h}}_{1:T} = \tilde{\mathbf{h}}_1, ..., \tilde{\mathbf{h}}_T$. To maximise the downstream utility of synthetic data, the generator aims to learn the conditional latent distribution produced by the embedder such that $p(\tilde{\mathbf{h}}_t) \approx p(\hat{\mathbf{h}}_t)\ \forall t \in T$. We achieve this by sharing predictor parameters between the generator and discriminator, but allowing the predictor's parameters to be changed only when updating with respect to the gradient $\frac{\partial \mathcal{L}_S}{\partial P_\theta}$, thereby forcing the condition.

The discriminator is tasked with differentiating between real sequences $\mathbf{x}_{1:T}$ and synthetic sequences $\tilde{\mathbf{x}}_{1:T}$. The discriminator operates over entire sequences, producing only one true/false classification and so, inspects synthetic sequences on a global scale and encourages the generator to learn the joint distribution of entire sequences. The objective $\mathcal{L}_{MM}$ reinforces this by while also exposing the generator to real samples and further encouraging bidirectional learning of the joint distribution.

Some approaches to time-series generation [26] have shown that explicit moment matching can improve the quality of synthetic data. We introduce an auxiliary moment loss to promote matching of first and second moments:

$$\mathcal{L}_{ML}(\mathbf{x}_{1:T}, \tilde{\mathbf{x}}_{1:T}) = |f_\mu(\mathbf{x}_{1:T}) - f_\mu(\tilde{\mathbf{x}}_{1:T})| + |f_\sigma(\mathbf{x}_{1:T}) - f_\sigma(\tilde{\mathbf{x}}_{1:T})| \tag{6}$$

where f_μ and f_σ are functions that compute the mean and standard deviation of a time-series.

3.4 Optimisation

We train our model in three stages. We begin by training the embedder–predictor components independently using the objective $\mathcal{L}_S$, followed by training the generator using only the masked modelling objective $\mathcal{L}_{MM}$. The final stage consists of joint training using all objectives.

All of our transformer model components are feed forward architectures that are insensitive to ordering of sequence. Hence, after the initial projection to the model dimension we add sinusoidal position embeddings to projected embeddings. For all transformer components, we use a model dimension

of $d = 32$, with $H = 8$ attention heads and a hidden layer dimension of $D = 4 \times d = 128$ with 3 encoder layers for each component. We use the GELU activation function [24] for the non-linearity with LayerNorm [3] normalisation.

For optimisation, we use the Adam optimiser [29], with a learning rate of 0.001 in the first two training stages for the embedder, predictor and generator, followed by a learning rate of 0.00002 for all components during joint training, with betas of $(0.5, 0.999)$. For masked modelling, we use $p_{mask} = 0.3$ for all datasets and a mini-batch size of 128.

4 Experiments

4.1 Evaluation Methodology

We compare the performance of our model on four different datasets against several baselines, including TimeGAN [51], RCGAN [17], C-RNN-GAN [35], COT-GAN [48] and Professor Forcing (P-Forcing) [30] models. We also perform ablation experiments, removing components of our TsT-GAN to identify sources of performance gain.

We consider the following evaluation metrics:

1. **Predictive Score** We generate a synthetic dataset using trained a generative model and train a post-hoc network on the synthetic data, after which we evaluate the post-hoc regression network on real data. If a generative model has captured the conditional distribution correctly, then we expect test mean absolute error (MAE) to be low and similar to when post-hoc network is trained on real data. The predictive score follows the TS-TR evaluation methodology [17]. An ideal generator will produce synthetic samples that, under the TS-TR framework, will produce a predictive score no worse than when the post-hoc model is trained on real data.

2. **Discriminative Score** A post-hoc network is trained to distinguish between real and synthetic data. The training set has an equal number of real and synthetic samples. We report the classification error on the held out test set. In the case of the ideal generator, synthetic samples are indistinguishable from real samples, resulting in a discriminative score of 0.

3. **Visualisation** We use t-SNE [43] to reduce the dimensionality of real and synthetic datasets, flattening across the temporal dimension. This allows comparison of how well the synthetic data distribution matches the original, indicating any areas of the original distribution not captured as well as out of distribution samples in the synthetic data. In addition, to evaluate how well the joint distribution is captured, we calculate the first difference for real and synthetic time-series and plot t-SNEs.

The code for each of the aforementioned models is available in public repositories published by the corresponding authors. We parameterise these models with all RNN-based components consisting of three layers of size 32. We use a sequence length of 24 for all datasets. Code associated with each of the models is used to train and generate synthetic data and evaluated using code similar to [51]. For both the predictive and discriminative evaluations, we use a two layer Gated Recurrent Unit (GRU) [8] with hidden size equal to the input dimension.

4.2 Datasets

We evaluate TsT-GAN across a range of datasets with different properties. All datasets we use are available online or can be generated.

1. **Sines** The quality of sine waves can be evaluated easily by inspection and this dataset consists of a number of sine waves with random shifts in phase and frequency. Phase and frequency shifts are random variables: phase shift in $\phi \sim$ *Uniform*$[-\pi, \pi]$ and frequency shift $\lambda \sim$ *Uniform*$[0, 1]$. We generate a multivariate Sines dataset consisting of 5 sine waves per sample. This synthetic dataset provides continuous valued, periodic functions with no correlations between features. **(5 features and 10 000 rows.)**

2. **Stocks** The Stocks dataset consists of daily data collected between 2004 and 2019 for the Google ticker.[1] **(5 features and 3685 rows.)**
3. **Energy** The UCI Appliances Energy Prediction dataset [7, 16] is high-dimensional and consists of features such as energy consumption, humidity and temperature collected by sensors. The data is complex with samples logged every 10 minutes for around four and a half months. **(28 features and 19 735 rows.)**
4. **Chickenpox** The UCI Hungarian Chickenpox Cases dataset [40, 16] consists of records of chickenpox cases weekly in 20 counties in Hungary. This dataset represents a realistic situation where generative models be trained on small amounts of data and then generate synthetic samples to train other models. **(20 features and 521 rows.)**
5. **Air** The UCI Air Quality dataset [12, 16] consists of levels of different gases recorded hourly in an Italian city. We remove the date and time columns as part of preprocessing. **(13 features and 9358 rows.)**

4.3 Visualisation, Predictive and Discriminative Scores

Table 1 shows that TsT-GAN consistently creates more *useful* data than the baseline models, achieving a lower predictive score across all datasets. Our predictive score for all datasets is remarkably close to the original score and the scores on the synthetic Sines and Stocks datasets outperform the original. TsT-GAN outperforms the next best performing baseline on the Sines, Energy and Stock datasets by 32%, 22% and 19%, respectively. TsT-GAN performs consistently in the discriminative tests, achieving best performance across two datasets. COT-GAN achieves an incredible 0.6% discriminative score on Chickenpox while exhibiting competitive predictive score performance with TsT-GAN.

We visualise t-SNEs in Figure 2 and see that samples from TsT-GAN overlap real data samples extremely well for all datasets. The Chickenpox dataset is especially difficult to model due to the small number of samples, nevertheless TsT-GAN is able to achieve significant coverage. TimeGAN seems to learn specific modes, while also producing some out of distribution samples. RCGAN, C-RNN-GAN, COT-GAN and P-Forcing produce several out of distribution samples. COT-GAN particular produces impressive looking graphs, being commensurate with predictive scores.

From Figure 3 we see that TsT-GAN captured the first differences well in all datasets. Of particular interest is the Sines row. Sines is a toy dataset with known generating distributions where first differences follow specific patterns (the differences are themselves sinusoidal); had TsT-GAN learned fully the generating distribution, we would expect to see distinct regions of high and low density for synthetic samples overlapping the true samples.

4.4 Ablation Experiments

We perform ablation experiments to evaluate the importance of each component of TsT-GAN. Our experiments are as follows:

- **- ML** Removes only the auxiliary moment matching objective. All subsequent ablations remove the moment matching objective as well.
- **- MM + Auto** Makes the generator autoregressive and removes the masked modelling objective, replacing it with a one step ahead prediction objective.
- **- Embedding** Removes the embedding network resulting in a generator that is trained with the LS-GAN objective and MM loss, with the parameters of the predictor network being updated jointly with the generator.
- **- MM** Removes only the masked modelling objective but retains the bidirectional generator.
- **Base** Is a standard transformer GAN made by removing MM and the embedding network.

From the latter two sections in Table 1 we see the TsT-GAN outperforming all ablations. The autoregressive generator outperforms TsT-GAN in the discriminative score for Stocks and Chickenpox, although the difference is small. Removing the embedding network in particular has a a significant

[1] Obtained from: `https://github.com/jsyoon0823/TimeGAN`

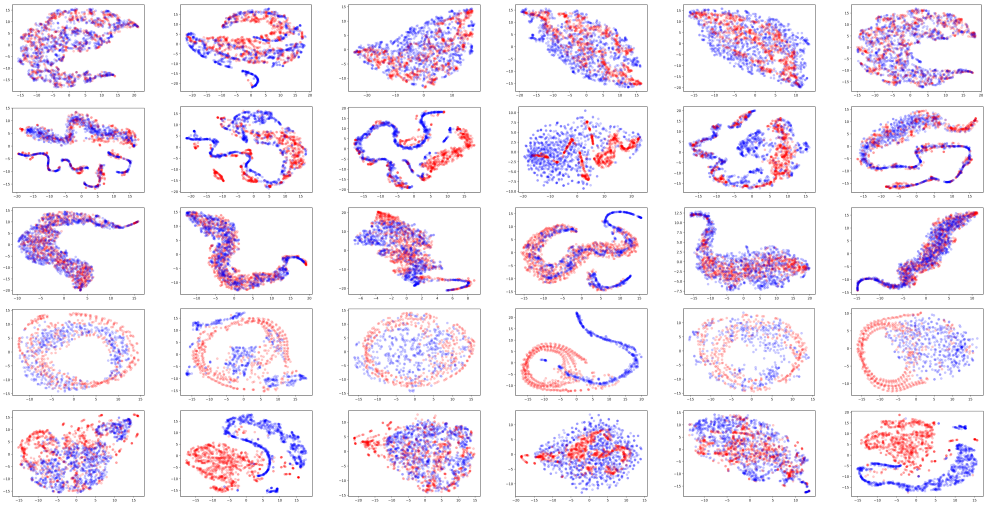
Table 1: Predictive and Discriminative scores with standard deviations for both comparison with baselines and ablations. Lower scores are better and best performance is indicated in bold. Predictive scores include score on the original data for comparison.

Predictive Score (MAE)					
Model	Sines	Stocks	Energy	Chickenpox	Air
Original	.009 ± .000	.010 ± .001	.032 ± .001	.089 ± .002	.034 ± .001
TsT-GAN	**.008 ± .000**	**.009 ± .000**	**.039 ± .001**	**.091 ± .001**	**.042 ± .002**
TimeGAN	.024 ± .004	.011 ± .001	.050 ± .001	.101 ± .002	.114 ± .005
RCGAN	.012 ± .000	.021 ± .001	.068 ± .001	.106 ± .002	.072 ± .001
C-RNN-GAN	.017 ± .000	.027 ± .001	.069 ± .001	.207 ± .002	.095 ± .002
COT-GAN	.016 ± .000	.012 ± .000	.056 ± .001	.094 ± .003	.044 ± .000
P-Forcing	.014 ± .001	.018 ± .000	.059 ± .003	.319 ± .002	.190 ± .041
Discriminative Score (proportion classified as synthetic)					
TsT-GAN	**.026 ± .018**	.122 ± .020	**.442 ± .007**	.053 ± .044	**.243 ± .009**
TimeGAN	.231 ± .117	.191 ± .029	.496 ± .004	.046 ± .044	.479 ± .025
RCGAN	.174 ± .033	.260 ± .010	.500 ± .000	.050 ± .003	.478 ± .004
C-RNN-GAN	.274 ± .040	.290 ± .032	.547 ± .004	.209 ± .001	.473 ± .024
COT-GAN	.302 ± .089	.260 ± .068	.500 ± .000	**.006 ± .053**	.441 ± .052
P-Forcing	.253 ± .036	**.088 ± .007**	.553 ± .044	.587 ± .009	.484 ± .018
Ablations Predictive Score (MAE)					
TsT-GAN	**.008 ± .000**	**.009 ± .000**	**.039 ± .001**	**.091 ± .001**	**.042 ± .002**
- ML	**.008 ± .007**	.011 ± .001	.047 ± .006	.101 ± .002	.045 ± .002
- MM + Auto	**.008 ± .001**	.012 ± .000	.054 ± .001	.111 ± .002	.046 ± .001
- Embedding	.009 ± .000	.016 ± .000	.081 ± .001	.145 ± .004	.056 ± .003
- MM	.009 ± .001	.013 ± .001	.057 ± .001	.095 ± .002	.051 ± .002
Base	.010 ± .001	.020 ± .000	.089 ± .001	.196 ± .006	.068 ± .003
Ablations Discriminative Score (proportion classified as synthetic)					
TsT-GAN	**.026 ± .018**	.122 ± .020	**.442 ± .007**	.053 ± .044	**.243 ± .009**
- ML	.028 ± .010	.140 ± .092	.465 ± .003	.069 ± .031	.302 ± .003
- MM + Auto	.029 ± .011	**.118 ± .089**	.488 ± .004	**.048 ± .017**	.452 ± .004
- Embedding	.145 ± .079	.254 ± .032	.497 ± .003	.342 ± .030	.514 ± .005
- MM	.166 ± .047	.171 ± .095	.498 ± .001	.480 ± .069	.477 ± .002
Base	.113 ± .042	.200 ± .035	.529 ± .0120	.462 ± .126	.613 ± .015

detrimental effect on predictive performance performance on all but the Sines dataset, suggesting that our enforcement of the conditional distribution plays an important role in capturing useful temporal correlations across time.

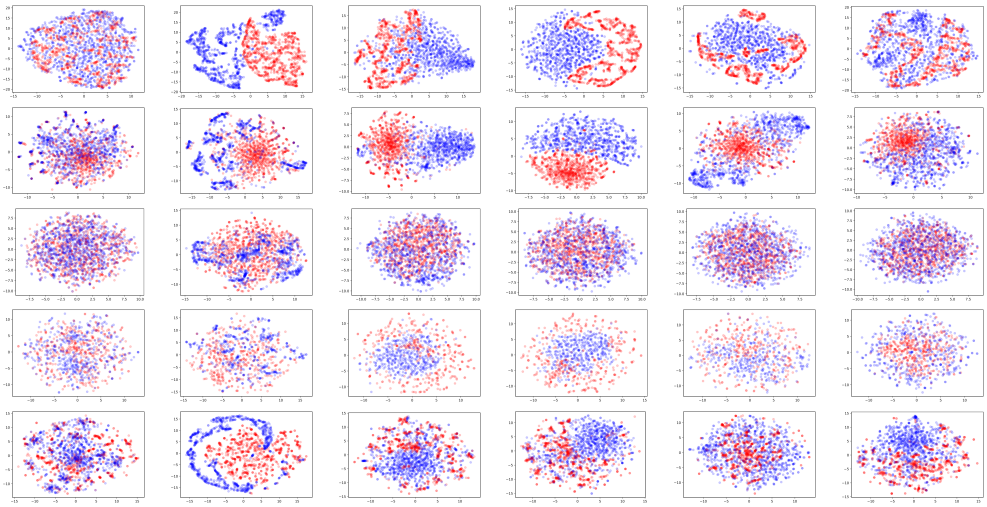
5 Conclusion

We have presented TsT-GAN, a new framework for training time-series generative models. The unconditional generator network in TsT-GAN is guided by unsupervised masked modelling to produce high quality synthetic sequences that capture both the global distribution as well as conditional time-series dynamics. We evaluate and benchmark our model using the TS-TR framework and show that TsT-GAN consistently outperforms existing methods. Future work could explore how to better incorporate moment matching in a unified framework, rather than as an auxiliary loss. Furthermore, TsT-GAN's discriminative scores still show scope for improvement suggesting that there still exists some discrepancy between true and learned distributions. A major limitation of our model is architecturally rooted: the Transformer architecture's self-attention mechanism has a computational complexity of $O(N^2)$ for a sequence of length N. As three out four components of TsT-GAN consist of Transformers, this results in a significant computational cost when training and performing inference with longer time-series.



(a) TsT-GAN (b) TimeGAN (c) RCGAN (d) C-RNN-GAN (e) COT-GAN (f) P-Forcing

Figure 2: t-SNE plots of Sines on the first row, Stocks on the second row, Energy on the third row, Chickenpox on the fourth row and Air on the fifth. Red indicates real data and blue indicates synthetic data. Best viewed in colour.



(a) TsT-GAN (b) TimeGAN (c) RCGAN (d) C-RNN-GAN (e) COT-GAN (f) P-Forcing

Figure 3: t-SNE plots of first differences with Sines on the first row, Stocks on the second row, Energy on the third row, Chickenpox on the fourth row and Air on the fifth. Red indicates real data and blue indicates synthetic data. Best viewed in colour.

Our model can also contribute to data compression. As data increases in resolution and demand for data increases, it is crucial to ensure that data remains accessible. We have shown that our model is able to learn meaningful representations of several time-series datasets as well as its utility in downstream tasks. In future applications, a trained model could be disseminated instead of a much larger dataset.

References

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318, 2016.

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

[3] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[4] Gaby Baasch, Guillaume Rousseau, and Ralph Evins. A conditional generative adversarial network for energy use in multiple buildings using scarce data. *Energy and AI*, 5:100087, 2021.

[5] Brett K Beaulieu-Jones, Zhiwei Steven Wu, Chris Williams, Ran Lee, Sanjeev P Bhavnani, James Brian Byrd, and Casey S Greene. Privacy-preserving generative deep neural networks support clinical data sharing. *Circulation: Cardiovascular Quality and Outcomes*, 12(7):e005122, 2019.

[6] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009.

[7] Luis M Candanedo, Véronique Feldheim, and Dominique Deramaix. Data driven prediction models of energy use of appliances in a low-energy house. *Energy and buildings*, 140:81–97, 2017.

[8] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[9] Edward Choi, Siddharth Biswal, Bradley Malin, Jon Duke, Walter F Stewart, and Jimeng Sun. Generating multi-label discrete patient records using generative adversarial networks. In *Machine learning for healthcare conference*, pages 286–305. PMLR, 2017.

[10] Aidan Clark, Jeff Donahue, and Karen Simonyan. Adversarial video generation on complex datasets. *arXiv preprint arXiv:1907.06571*, 2019.

[11] Saloni Dash, Andrew Yale, Isabelle Guyon, and Kristin P Bennett. Medical time-series data generation using generative adversarial networks. In *International Conference on Artificial Intelligence in Medicine*, pages 382–391. Springer, 2020.

[12] Saverio De Vito, Ettore Massera, Marco Piga, Luca Martinotto, and Girolamo Di Francia. On field calibration of an electronic nose for benzene estimation in an urban pollution monitoring scenario. *Sensors and Actuators B: Chemical*, 129(2):750–757, 2008.

[13] E. Debie, N. Moustafa, and M. T. Whitty. A privacy-preserving generative adversarial network method for securing eeg brain signals. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2020.

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[15] Jian Du, Song Li, Moran Feng, and Siheng Chen. Dynamic differential-privacy preserving sgd. *arXiv preprint arXiv:2111.00173*, 2021.

[16] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[17] Cristóbal Esteban, Stephanie L Hyland, and Gunnar Rätsch. Real-valued (medical) time series generation with recurrent conditional gans. *arXiv preprint arXiv:1706.02633*, 2017.

[18] Rasool Fakoor, Jonas W Mueller, Nick Erickson, Pratik Chaudhari, and Alexander J Smola. Fast, accurate, and simple models for tabular data via augmented distillation. *Advances in Neural Information Processing Systems*, 33:8671–8681, 2020.

[19] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030, 2016.

[20] Aude Genevay, Gabriel Peyré, and Marco Cuturi. Learning generative models with sinkhorn divergences. In *International Conference on Artificial Intelligence and Statistics*, pages 1608–1617. PMLR, 2018.

[21] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

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

[23] S. Haradal, H. Hayashi, and S. Uchida. Biosignal data augmentation based on generative adversarial networks. In *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 368–371, 2018.

[24] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

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

[26] Daniel Jarrett, Ioana Bica, and Mihaela van der Schaar. Time-series generation by contrastive imitation. *Advances in Neural Information Processing Systems*, 34, 2021.

[27] Alexia Jolicoeur-Martineau. The relativistic discriminator: a key element missing from standard gan. *arXiv preprint arXiv:1807.00734*, 2018.

[28] James Jordon, Jinsung Yoon, and Mihaela Van Der Schaar. Pate-gan: Generating synthetic data with differential privacy guarantees. In *International conference on learning representations*, 2018.

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

[30] Alex M Lamb, Anirudh Goyal ALIAS PARTH GOYAL, Ying Zhang, Saizheng Zhang, Aaron C Courville, and Yoshua Bengio. Professor forcing: A new algorithm for training recurrent networks. *Advances in neural information processing systems*, 29, 2016.

[31] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, and Zehan Wang. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690, 2017.

[32] Yi Liu, Jialiang Peng, JQ James, and Yi Wu. Ppgan: Privacy-preserving generative adversarial network. In *2019 IEEE 25Th international conference on parallel and distributed systems (ICPADS)*, pages 985–989. IEEE, 2019.

[33] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2794–2802, 2017.

[34] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

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

[36] Andres Munoz, Mohammadreza Zolfaghari, Max Argus, and Thomas Brox. Temporal shift gan for large scale video generation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3179–3188, 2021.

[37] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. *Advances in neural information processing systems*, 29, 2016.

[38] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[39] Sai Rajeswar, Sandeep Subramanian, Francis Dutil, Christopher Pal, and Aaron Courville. Adversarial generation of natural language. *arXiv preprint arXiv:1705.10929*, 2017.

[40] Benedek Rozemberczki, Paul Scherer, Oliver Kiss, Rik Sarkar, and Tamas Ferenci. Chickenpox cases in hungary: a benchmark dataset for spatiotemporal signal processing with graph neural networks. *arXiv preprint arXiv:2102.08100*, 2021.

[41] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29, 2016.

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

[43] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[44] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics. *Advances in neural information processing systems*, 29, 2016.

[45] Neo Wu, Bradley Green, Xue Ben, and Shawn O'Banion. Deep transformer models for time series forecasting: The influenza prevalence case. *arXiv preprint arXiv:2001.08317*, 2020.

[46] Sifan Wu, Xi Xiao, Qianggang Ding, Peilin Zhao, Ying Wei, and Junzhou Huang. Adversarial sparse transformer for time series forecasting. *Advances in Neural Information Processing Systems*, 33:17105–17115, 2020.

[47] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Modeling tabular data using conditional gan. *Advances in Neural Information Processing Systems*, 32, 2019.

[48] Tianlin Xu, Li Kevin Wenliang, Michael Munn, and Beatrice Acciaio. Cot-gan: Generating sequential data via causal optimal transport. *Advances in Neural Information Processing Systems*, 33:8798–8809, 2020.

[49] Ruikang Yang, Jianfeng Ma, Yinbin Miao, and Xindi Ma. Privacy-preserving generative framework against membership inference attacks. *arXiv preprint arXiv:2202.05469*, 2022.

[50] Zhen Yang, Wei Chen, Feng Wang, and Bo Xu. Improving neural machine translation with conditional sequence generative adversarial nets. *arXiv preprint arXiv:1703.04887*, 2017.

[51] Jinsung Yoon, Daniel Jarrett, and Mihaela Van der Schaar. Time-series generative adversarial networks. *Advances in Neural Information Processing Systems*, 32, 2019.

[52] George Zerveas, Srideepika Jayaraman, Dhaval Patel, Anuradha Bhamidipaty, and Carsten Eickhoff. A transformer-based framework for multivariate time series representation learning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 2114–2124, 2021.

[53] Yabing Zhu, Yanfeng Zhang, Huili Yang, and Fangjing Wang. Gancoder: an automatic natural language-to-programming language translation approach based on gan. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 529–539. Springer, 2019.