

Time-Transformer: Integrating Local and Global Features for Better Time Series Generation

Yuansan Liu * Sudanthi Wijewickrema † Ang Li ‡ Christofer Bester †
 Stephen O’Leary † James Bailey †

Abstract

Generating time series data is a promising approach to address data deficiency problems. However, it is also challenging due to the complex temporal properties of time series data, including local correlations as well as global dependencies. Most existing generative models have failed to effectively learn both the local and global properties of time series data. To address this open problem, we propose a novel time series generative model named ‘Time-Transformer AAE’, which consists of an adversarial autoencoder (AAE) and a newly designed architecture named ‘Time-Transformer’ within the decoder. The Time-Transformer first simultaneously learns local and global features in a layer-wise parallel design, combining the abilities of Temporal Convolutional Networks and Transformer in extracting local features and global dependencies respectively. Second, a bidirectional cross attention is proposed to provide complementary guidance across the two branches and achieve proper fusion between local and global features. Experimental results demonstrate that our model can outperform existing state-of-the-art models in 5 out of 6 datasets, specifically on those with data containing both global and local properties. Furthermore, we highlight our model’s ability to handle this kind of data via an artificial dataset. Finally, we show how our model performs when applied to a real-world problem: data augmentation to support learning with small datasets and imbalanced datasets.

Keywords: time series generation, temporal convolutional networks, transformer, bidirectional cross-attention

1 Introduction

Automatically synthesizing realistic data assists in solving real-world problems when access to real data is limited and manual generation is impractical. Deep generative models have shown great success in dealing with this problem of data deficiency in computer vision

and natural language processing during the last decade. Numerous models have been introduced to address downstream tasks such as image in-painting [43], text to image translation [61], and automated captioning [19].

Although data deficiency is similarly crucial in the time series domain, there exist relatively few generative models to tackle this problem. This is due to the fact that the generated data is required to share a similar global distribution with the original time series data and also preserve its unique temporal properties. Some existing works utilize Generative Adversarial Networks (GANs) [18] for time series generation and most of them address the temporal challenges using Recurrent Neural Networks (RNNs) such as Long Short Term Memory (LSTM) [15, 58, 44]. There are also approaches that use Variational Autoencoder (VAE) [28] as the basic framework to generate time series data [16, 13]. However, none of these works have fully succeeded in efficiently learning both local correlation and global interaction, which is critical for time series processing.

Recently, Transformer based models have been successful in learning global features for different types of data including time series [45, 14, 60, 8, 7]. On the other hand, models based on Convolutional Neural Networks (CNNs) have shown their strength in extracting local patterns with their filters [23, 54, 37]. Temporal Convolutional Networks (TCNs), consisting of dilated convolutional layers [41], preserve the original local processing capability of CNNs, but also have an enhanced ability to learn temporal dependencies in sequential data [4]. This makes them appropriate for time series modeling. Therefore, it is natural to combine Transformer and TCN together to learn better time series features. Some previous works use sequential combinations of these for time series tasks [35, 6]. However, such sequential designs do not consider the interaction between local and global features inherent in these datasets. Thus, it is critical to learn different levels of features separately to preserve their independence, and then, integrate them with an appropriate method.

Motivated by the above observations and analysis, we propose a novel time series generative model named

*University of Melbourne, Melbourne, VIC, Australia
 yuansanl@student.unimelb.edu.au

†University of Melbourne, Melbourne, VIC, Australia
 {sudanthy.wijewickrema, christofer.bester, sjoleary, baileyj}@unimelb.edu.au

‡Academy of Military Sciences, Beijing, China
 angli.cs@outlook.com

‘Time-Transformer AAE’. Specifically, we first use the Adversarial Autoencoder (AAE) [39] as the generative framework. The Time-Transformer is designed as part of the decoder to efficiently and effectively learn and integrate both local and global features. In each Time-Transformer block, the temporal properties are learnt by both a TCN layer and a Transformer block. They are then connected through a bidirectional cross attention block to fuse local and global features. This layer-wise parallelization, along with bidirectional interaction, combines the advantages of the TCN and Transformer models: the efficiency of the TCN to extract local features, as well as the Transformer’s ability in building global dependency. We evaluate our proposed Time-Transformer AAE on different types of time series data from both artificial and real-world datasets. Experiments show that the proposed model surpasses the existing state-of-the-art (SOTA) models in addressing the time series generation task. Furthermore, we also show our model’s effectiveness on downstream tasks including small dataset augmentation and imbalanced classification. To summarize, our contributions are as follows:

- We introduce the Time-Transformer module that simultaneously learns local and global features in a layer-wise parallel design, and facilitates interaction between these two types of features by performing feature fusion in a bidirectional manner.
- By inserting the proposed module into the decoder of the Adversarial Autoencoder, we present a novel time series generative model, Time-Transformer AAE, which efficiently and effectively generates high quality synthetic time series data containing both global and local features.
- We show empirically that the proposed Time-Transformer AAE can generate better synthetic time series data with respect to different benchmarks, when compared to SOTA methods.
- Through comprehensive experiments, we demonstrate the efficiency and effectiveness of our model in handling time series data containing both global and local features, and subsequently, addressing data deficiency issues.

2 Related Works

2.1 Time Series Generation: Deep generative models (DGMs) have gained increasing attention since their introduction. The variational autoencoder (VAE) uses the Bayesian method to learn latent representations and turn the classic autoencoder into a generative model [28]. Generative adversarial networks (GANs) introduce an

adversarial approach to shape the output distribution in order to generate realistic fake data [18]. The adversarial autoencoder (AAE) combines the previous two models together [39]. It uses the adversarial training procedure to perform variational inference in the VAE. Numerous models have been designed based on these basic generative frameworks that have shown superior performance in image and text processing [42, 43, 61, 27, 3, 21, 1].

Successes in processing graphs and text have led to the application of DGMs’ in time series problems. Most of them use the GAN framework with modifications to incorporate temporal properties. The first of these, called C-RNN-GAN [40], directly uses the GAN structure with LSTM to generate music data. Recurrent Conditional GAN (RCGAN) [15] uses a basic RNN as its backbone, and auxiliary label information as the condition to generate medical time series. Since its introduction, a number of works have utilized similar designs to generate time series data in various fields including finance, medicine and the internet [20, 30, 52, 49, 36]. TimeGAN [58] introduces additional embedding function and supervised loss to generate universal time series. Real-world Time Series GAN (RTSGAN) [44] combines WGAN [3] and autoencoder, and achieves good performance on real-world data generation. PSA-GAN [25] utilizes the progressively growing architecture [27] to generate long time series. To generate both regular and irregular sampled time series, GT-GAN [26] combines Neural Differential Equations and DGMs together for the first time. COSCI-GAN [48] focuses on multivariate time series, and uses a central discriminator to learn better inter-channel dynamics. On the other hand, VAE has also been used for this task. [16] designed a recurrent VAE to synthesize time series data. TimeVAE [13] implements an interpretable temporal structure together with VAE, and achieves SOTA results on universal time series generation. CR-VAE [33] incorporates the causal mechanism into time series generation by learning a Granger causal graph. Recently, the diffusion model has also been utilized together with TCNs to generate audio data [29]. In comparison to traditional DGMs, TimeGCI uses the contrastive learning framework to generate time series and achieves good results [24]. These works learn the overall distribution of the training set and generate synthetic time series within the same distribution.

On the other hand, another line of works focus on step-wise generation: They build generative models to infer each time step on the previous ones. Some early works modify the basic RNN using variational inference and achieve good results with regard to log-likelihood [9, 17]. Some later works combine this idea with TCN [32, 2]. In recent years, diffusion models are used to achieve SOTA results in this area [46, 34]. Though

different to our task, these works provide an instructive direction for applying generative models on the time series forecasting task.

2.2 Temporal Convolutional Networks and Transformer: Temporal Convolutional Networks (TCNs) [4] use dilated causal convolutions in WaveNet [41] to encode temporal patterns and avoid information leakage from future to past. TCN models have been used for sequential data such as time series and are able to successfully extract local features [47, 62, 22, 12]. Transformer [51] implements a self-attention mechanism on the entire data to learn global interactions between each point, which enhances ability to learn long range dependence and improves its performance on machine translation tasks. Its variants also achieve success on language, vision and time series tasks [45, 14, 60, 7]. Some works combine these two types of models sequentially (place a Transformer either before or after a TCN) to take advantage of both their capabilities [35, 6]. However, this design assumes dependencies exist between the models. However, a parallel structure like our model is necessary if different levels of features need to be learnt separately to preserve their independence.

3 Problem Formulation

Generally, multivariate time series generation via GAN-like models involves training a model to approximate a function $\mathcal{F} : \mathcal{P} \rightarrow \mathcal{P}_d$ which maps an arbitrary prior distribution \mathcal{P} to the real data distribution \mathcal{P}_d , so that the model can produce realistic synthetic data $X' \sim \mathcal{P}_d$ based on any samples s drawn from \mathcal{P} . For time series data $X \in \mathbb{R}^{T \times c}$, where T is maximum observable time and c is number of channels, we define the local patterns and the global dependencies as:

$$(3.1) \quad \mathcal{L} = p(X_{k:t-1})$$

$$(3.2) \quad \mathcal{G} = p(X_t | X_{1:t-1})$$

where $1 < k < t$ and p is the density. They are derived from previous researches [58, 56] with adaptations to align with our requirements. Thus, X contains both \mathcal{L} and \mathcal{G} and function \mathcal{F} must learn them simultaneously:

$$(3.3) \quad \mathcal{F}(p) = \mathcal{I}(\mathcal{L}, \mathcal{G}), 1 < k < t$$

where \mathcal{I} is an integration function that combines local and global features together.

The goal of our work is to design a model to approximate \mathcal{F} . It should extract both local and global features and properly combine them to generate realistic and useful synthetic time series data that can be used for downstream real-world tasks.

4 Time-Transformer AAE

4.1 Overview: We use Adversarial Autoencoder (AAE) as our generation framework due to the potential of extending it to supervised and semi-supervised learning settings [39]. To apply AAE for time series generation, we first use Convolutional Neural Networks (CNNs) as the basis of the AAE, and then insert the Time-Transformer (see Figure 1b) into the decoder of the AAE. The overall structure is shown in Figure 1a.

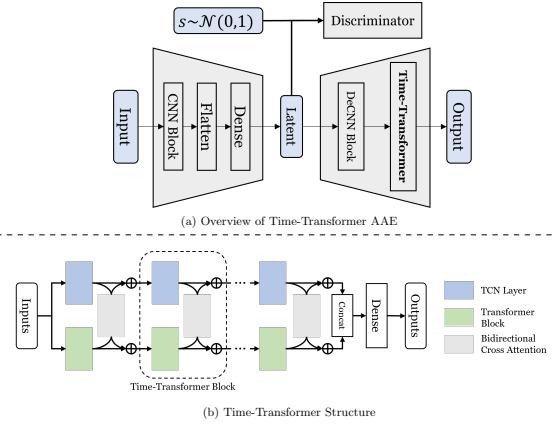


Figure 1: Time-Transformer AAE

Here, we use Gaussian as the prior distribution $\mathcal{P} = \mathcal{N}(0, 1)$ and insert the Time-Transformer after a De-Convolutional block to approximate the \mathcal{F} in (3.3) (This design is to achieve an optimal equilibrium between efficiency and effectiveness. We provide details in [38]). The De-Convolutional block will first reconstruct a prototype of the time series, and then the Time-Transformer aims at learning the detailed local/global features through feedback from reconstruction loss during training. Finally, the model will be able to synthesize realistic data in the generation step.

Within the Time-Transformer, we use a layer-wise parallelization to combine the Temporal Convolutional Networks (TCNs) and the Transformer. The learnt prototype from the De-Convolutional block is passed into a TCN layer and a Transformer block simultaneously. The TCN learns the local features of the time series, and the Transformer finds the global patterns of the data. The learnt results then go through a cross-attention block to fuse with each other bidirectionally. At the end of this parallel structure we concatenate the outputs from both sides, and use a fully-connected layer to map them into the expected dimension ($t \times c$) and reshape into the original time series dimensions.

4.2 Time-Transformer: The Time-Transformer consists of several Time-Transformer blocks, which have two

key differences to the standard TCN layers and Transformer blocks: 1) a layer-wise parallel design to combine local-window self-attention and depth-wise convolution and 2) bidirectional cross attention over the two branches. Figure 2 shows the details of the Time-Transformer block.

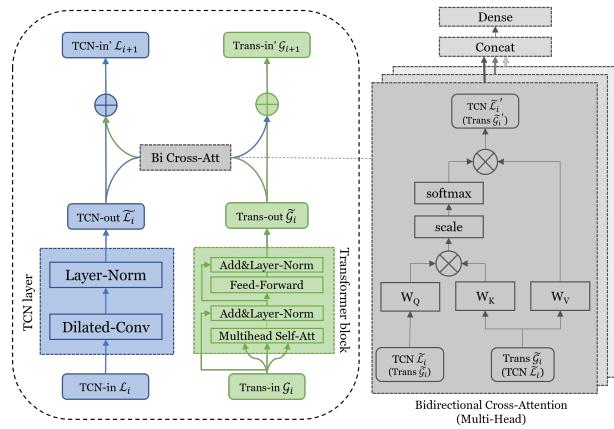


Figure 2: Detailed design of the Time-Transformer block

Layer-wise Parallelization: The Mobile-Former [8] combines MobileNet (a light-weight CNN) and Transformer in parallel, and achieve better performance than sequentially combined models on image classification and object detection tasks. Inspired by this, we also combine TCN and Transformer in a parallel manner to learn \mathcal{L} and \mathcal{G} in (3.3) simultaneously. However, instead of simply combining the entire TCN and Transformer, we only use one layer from the TCN and one block from the Transformer in each Time-Transformer block.

As shown in Figure 2, each Time-Transformer block has TCN and Transformer (Trans) streams parallel to each other. Inside the TCN stream we have a dilated convolutional layer with post layer normalization, which make up one hidden layer of the TCN model. The inputs and outputs of each block (TCN-in and TCN-in' respectively in Figure 2) are local features including those learnt directly from the last/current TCN layer (TCN-out in Figure 2) and the fusion of the local and global features. The Transformer block has a Multi-head Self Attention layer and a Feed-forward Network, both with post layer normalization. Similarly, the inputs/outputs (Trans-in and Trans-in' respectively in Figure 2) of this side are global features from last/current Transformer block together with a feature fusion.

Bidirectional Cross Attention: The bidirectional cross attention block contains two cross attention mechanisms, aims to build interactions between two parallel branches, and thus fuses local and global features to function as the \mathcal{I} in (3.3). As illustrated on the right

hand side of Figure 2, the output of the TCN layer $\mathcal{L}_i \in \mathbb{R}^{t \times c}$ and the output of the Transformer block $\tilde{\mathcal{G}}_i \in \mathbb{R}^{t \times c}$ interact in a mutual manner within the cross attention block, where it bidirectionally fuses the local feature \mathcal{L}_i and the global feature $\tilde{\mathcal{G}}_i$. Specifically, the TCN features are updated through a residual connection with the attention matrix to obtain \mathcal{L}_{i+1} :

$$(4.4) \quad \mathcal{L}_{i+1} = \tilde{\mathcal{L}}_i + \mathcal{A}_{\tilde{\mathcal{G}}_i \rightarrow \tilde{\mathcal{L}}_i} \cdot \mathcal{G}_i W_{ev}$$

where W_{ev} is the learnable parameter for the value embedding layer, and $\mathcal{A}_{\tilde{\mathcal{G}}_i \rightarrow \tilde{\mathcal{L}}_i}$ is the affinity matrix from Trans to TCN which can be calculated with matrix multiplication and a softmax function:

$$(4.5) \quad \mathcal{A}_{\tilde{\mathcal{G}}_i \rightarrow \tilde{\mathcal{L}}_i} = \text{softmax}\left(\frac{\tilde{\mathcal{L}}_i W_{eq} \cdot (\tilde{\mathcal{G}}_i W_{ek})^T}{\sqrt{c}}\right)$$

where W_{eq} and W_{ek} are learnable parameters of two linear projection layers. Similarly, the updated Trans feature map \mathcal{G}_{i+1} is defined as:

$$(4.6) \quad \mathcal{G}_{i+1} = \tilde{\mathcal{G}}_i + \mathcal{A}_{\tilde{\mathcal{L}}_i \rightarrow \tilde{\mathcal{G}}_i} \cdot \mathcal{L}_i W_{dv}$$

$$(4.7) \quad \mathcal{A}_{\tilde{\mathcal{L}}_i \rightarrow \tilde{\mathcal{G}}_i} = \text{softmax}\left(\frac{\tilde{\mathcal{G}}_i W_{dq} \cdot (\tilde{\mathcal{L}}_i W_{dk})^T}{\sqrt{c}}\right)$$

where W_{dq} , W_{dk} , and W_{dv} are learnable parameters of three linear projection layers. $\mathcal{A}_{\tilde{\mathcal{L}}_i \rightarrow \tilde{\mathcal{G}}_i}$ is the affinity matrix from TCN to Trans

5 Experiments

5.1 Datasets: We use six datasets to evaluate our model. The first three datasets have been used in several previous works [58, 44, 13]. Hence, they are referred to as *preliminary datasets*. The next three datasets contain time series with both local and global patterns. They are used to evaluate the model’s ability to learn both types of features, and are termed *local-global datasets*:

Sine_Sim: 5000 samples of 5-dimensional sinusoidal sequences: $x_t^{(d)} = \alpha \sin(2\pi f t + \varphi)$ where $d \in [1, 5]$, $\alpha \sim \mathcal{U}[1, 3]$, $f \sim \mathcal{U}[0.1, 0.15]$ and $\varphi \sim \mathcal{U}[0, 2\pi]$

Stock: 3686 samples of stock-price from historical Google daily stocks. Each sample has 6 features correlated with each other.

Energy: The UCI Appliances energy prediction dataset contains 19735 time series data with high dimensionality (28) and correlated features.

Sine_Cpx: Another 5000 samples of more complex 5-dimensional sinusoidal sequences. They are simulated using the sum of standard sinusoidal waves $x_t^{(d)} = \alpha_1 \sin(2\pi f_1 t + \varphi_1) + \alpha_2 \sin(2\pi f_2 t + \varphi_2) + \alpha_3 \sin(2\pi f_3 t + \varphi_3)$ where $\alpha_i \sim \mathcal{U}[1, 3]$, $f_i \sim \mathcal{U}[0.1, 0.15]$, $\varphi_i \sim \mathcal{U}[0, 2\pi]$, $i = 1, 2, 3$ and $\alpha_i \neq \alpha_j$, $f_i \neq f_j$, $\varphi_i \neq \varphi_j$, when $i \neq j$. Thus,

the sequences demonstrate global sine waves and local patterns in each dimension.

Music: A 2-dimensional time series waveform audio file of classical music ‘Canon in D’ converted from MP3, sampled at 8000Hz frequency with double channels, resulting in 2845466 time-steps. The music data exhibits local patterns and global seasonality. A 10000-step segment (2'05” to 2'06”) is selected for our experiments.

ECochG: A medical dataset of historical patient data from cochlear implant surgeries [53]. Each instance contains a univariate time series that represents a patient’s inner ear response during the surgery. In general, each time series is a stochastic trend globally, with some local drops. More details of this dataset can be found in [38].

5.2 Baseline Models & Benchmarks: We select previous models, TimeVAE, RTSGAN, and TimeGAN [13, 44, 58], as baselines based on their relevance, accessibility of the code, executability of the code, and performance of the model¹. We use their original settings as they have already been optimised by their authors. Details of our model² (hyper-parameters and data pre-processing) can be found in [38]. To evaluate these generative models, we use a range of metrics and consider several desiderata for the generated time series: They should (1) be distributed close to the original data and different from each other, (2) be indistinguishable from the original data, and (3) preserve the original temporal properties in order to be used for downstream tasks.

Visualisation is an indicator of (1) showing the distributions over the original and generated data. We flatten the temporal dimension and use t-SNE [50] plots for 2-dimensional space for visualisation.

Fréchet Inception Distance (FID) evaluates quality of generated data for GAN type models. Following[20, 49, 25], we replace the Inception model of the original FID with a recent SOTA time series representation learning method TS2Vec [59]. FID score provides a quantitative assessment of (1).

Discriminative Score measures indistinguishability of the generated data. We follow protocol from [58] when defining this and next benchmarks. We first train a post-hoc sequential classifier (a 2-layer LSTM) to distinguish original and generated data. Then we get test accuracy acc_{te} from a held-out test set. Good generated time series data should be indistinguishable from real data (accuracy close to 0.5). Thus, we use $|acc_{te} - 0.5|$ as the Discriminative Score to measure (2).

Predictive Score reflects how well the generated data inherits the predictive properties of the original data. We train a post-hoc sequential predictor (a 2-layer LSTM) to do one-more-step prediction, and calculate the mean absolute error (MAE). Here, we use the ‘Train on Synthetic, Test on Real’ (TSTR) technique introduced by [15]. The predictor should be able to learn from the generation and predict the real data. The Predictive Score provides a quantitative assessment of (3).

5.3 Experimental Results

5.3.1 Visualisation & FID: Figure 3 shows visualisation results of three local-global datasets (t-SNE plots for preliminary datasets can be found in [38]). Each row contains plots of one dataset and each column represents a model. The blue and red dots represent original and generated data points respectively. It is obvious that our proposed Time-Transformer AAE (denoted as **Time-Trans** hereafter) consistently produces synthetic data closely distributed to the original data. TimeVAE and RTSGAN also have good distributions for most datasets. Table 1 shows the FID scores of each model on each dataset. Our proposed Time-Trans obtains the best scores for all the datasets except ‘Stock’ where RTSGAN is slightly better. For those datasets where the models show similar visualisation results (e.g. Time-Trans and TimeVAE on ‘Sine_Cpx’), we can use the FID scores to quantitatively evaluate them.

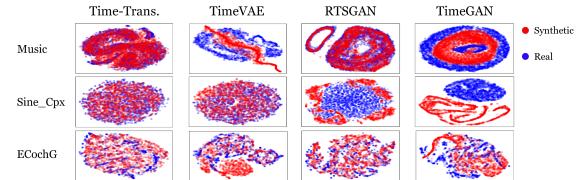


Figure 3: t-SNE visualisations of models (columns) on 3 datasets (rows). The blue dots represent original data and the red dots represent synthetic data.

Table 1: FID score: lower scores are better

Models	Time-Trans	TimeVAE	RTSGAN	TimeGAN
Sine_Sim	0.283±0.023	0.766±0.027	3.175±0.142	4.572±0.153
Stock	0.273±0.035	0.303±0.037	0.254±0.031	0.646±0.072
Energy	0.983±0.039	1.598±0.067	6.292±0.265	2.506±0.126
Music	0.395±0.013	13.415±0.937	0.957±0.095	1.894±0.121
Sine_Cpx	1.502±0.062	3.486±0.170	7.696±0.251	14.255±0.491
ECochG	0.348±0.024	5.197±0.237	0.527±0.062	0.674±0.098

¹Models aligned with our task but failed any of these criteria are excluded. E.g., PSA-GAN’s released codes are not executable

²The code is available at <https://github.com/Lysarthas/Time-Transformer>

5.3.2 Discriminative Score & Predictive Score: As per the Discriminative Scores shown in Table 2, our Time-Trans performs well for most datasets, and

Table 2: Discriminative score: $|acc_{te} - 0.5|$, lower scores are better

Models	Time-Trans	TimeVAE	RTSGAN	TimeGAN
Sine_Sim	0.131±0.021	0.217±0.015	0.489±0.007	0.485±0.008
Stock	0.463±0.023	0.476±0.050	0.374±0.022	0.481±0.033
Energy	0.496±0.005	0.499±0.002	0.499±0.001	0.499±0.001
Music	0.160±0.054	0.495±0.008	0.489±0.009	0.494±0.006
Sine_Cpx	0.168±0.041	0.471±0.016	0.489±0.031	0.500±0.000
ECochG	0.103±0.012	0.474±0.016	0.405±0.004	0.424±0.015

Table 3: Predictive score: $10 \times MAE$, lower scores are better

Model	Oracle	Time-Trans	TimeVAE	RTSGAN	TimeGAN
Sine_Sim	0.038±0.012	0.051±0.015	0.108±0.031	0.789±0.069	0.465±0.055
Stock	0.015±0.003	0.048±0.004	0.080±0.003	0.036±0.003	0.095±0.007
Energy	0.044±0.004	0.077±0.007	0.072±0.006	0.228±0.022	0.115±0.013
Music	0.014±0.008	0.017±0.009	0.077±0.051	0.021±0.009	0.027±0.007
Sine_Cpx	0.029±0.008	0.032±0.006	0.055±0.015	0.070±0.006	0.527±0.025
ECochG	0.011±0.009	0.013±0.006	0.087±0.008	0.176±0.021	0.042±0.006

particularly excels in performance for the local-global datasets where the generations from our model show better indistinguishability than the other methods.

We report ten times MAE as Predictive Score to avoid long decimal numbers while highlighting significant differences. An additional column ‘Oracle’ shows results from the ‘Train on Real, Test on Real’, which represent the theoretically best performances. Table 3 shows a similar trend: our model performs well in most datasets, particularly, in local-global datasets where our model’s scores are close to Oracle. This indicates that our model has learnt most of the predictive properties of these datasets. Note that scores in our work differ to those published in related works because we calculate them in a slightly different way. We provide a detailed justification in [38]. Apart from these six datasets, we also evaluated our model on longer time series data. The results show that our model consistently out-performs baselines with these longer time series (details can be found in [38]).

5.3.3 Local & Global Features Learning: We artificially simulate datasets to study the performance of the models with time series containing varying portions of global and local features. Figure 4 shows the underlying idea. The green and orange series represent

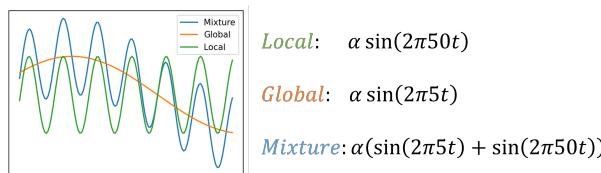


Figure 4: Mixture, Global and Local time series

local and global features respectively. They are generated with high (50Hz) and low (5Hz) frequencies with respect to the Fast Fourier Transform (FFT). By adding them together we get the blue series which, contains both global and local features (named ‘Mixture’). The formulae to generate local, global and mixture time series are alongside, where $\alpha \sim \mathcal{U}[1, 3]$.

For each of the three cases, we generate 5000 time series, each consisting of 128-step time steps (we use 128 to ensure global features are non-trivial). We thus obtain three different datasets, each containing 5000 time series.

We vary the portion of each part in Mixture to get two other series that contain 75% local and 75% global respectively. We train the generative models on these datasets and compare performance. Figure 5 shows the above scores (mean score with standard deviation) of the three best models (RTSGAN, TimeVAE, and Time-Trans). When the data only contains one feature type,

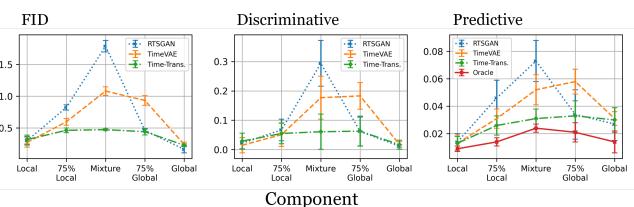


Figure 5: Results of Time-Trans(green), RTSGAN(blue), and TimeVAE(orange). Lower are better.

all models can generate comparatively good time series, but their performances decrease with mixed features. Among these models, ours shows the most consistent performance in all cases and has the best performance in mixture datasets with respect to all metrics, indicating

its effectiveness in learning both local and global features. We provide detailed results in [38].

5.3.4 Model Application - Data Augmentation:

We evaluate our model on imbalanced classification using the ECochG dataset and three real-world datasets from the UCR archive [11]: Wafer, Herring, and SwedishLeaf (details in [38]). Previous studies argued oversampling approaches (via replication or simple modifications like jittering) can cause overfitting or create out-of-distribution synthesis [57, 55]. Our model is expected to address these issues. Table 4 lists the classification results on the ECochG dataset using an off-the-shelf Multi-layer Perceptron (MLP) with (1) no augmentations, and augmenting the minority class via (2) repeating (3) jittering, (4) RTSGAN, (5) TimeVAE, and (6) our model (augmentations are only in training sets). We do not list TimeGAN because training data here are

Table 4: Imbalanced classification evaluation (ROC is receiver operating characteristic curve and PRC is precision-recall curve. Higher scores are better)

Components	Accuracy	Recall	Precision	AUC_ROC	AUC_PRC
No Aug	0.9751	0.7062	0.8706	0.9824	0.8521
Replication	0.9719	0.9552	0.7072	0.9824	0.8636
Jittering	0.9757	0.7074	0.8864	0.9866	0.8870
RTSGAN	0.9702	0.6791	0.8235	0.9767	0.7352
TimeVAE	0.9738	0.7388	0.8319	0.9807	0.7906
Time-Trans	0.9802	0.8178	0.9163	0.9945	0.9345

insufficient to train this model. The results on the other three datasets can be found in [38]. One crucial problem of imbalanced classification is low recall rates, caused by an under-represented positive class in the training set. This can result in negative prediction dominance. Repeating the minority class can improve the recall at the cost of precision, which reflects the aforementioned overfitting problem. However, augmenting the training set with synthetic data from our model shows better results with respect to all metrics suggesting that it can aid in solving this kind of problem.

Using the balanced datasets (< 2000 time series), we investigate how 3 generative models can improve the deep learning training for small datasets. We extend these datasets with 25%, 50%, 75%, and 100% of their size and use them to train the same MLP. Figure 6 shows the change of accuracy and area under ROC and PRC curves. The classifier performs better with more synthetic training data, and best with those from our model. Detailed results are provided in [38].

5.3.5 Ablation Study & Comparison with Sequential Design: Using the ‘Music’ dataset, we investigate how each component of the model contributes to the final result. We (1) use only a De-convolutional

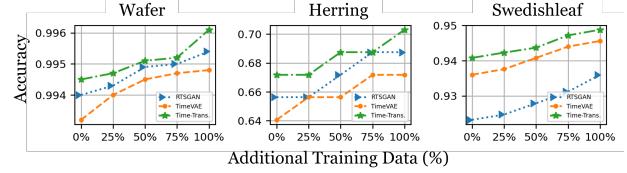


Figure 6: Classification results on augmented datasets with: Time-Trans(green), RTSGAN(blue), and TimeVAE(orange). Higher results are better.

block, (2) add only a TCN block after (1), (3) add only a Transformer block after (1), and (4) add our proposed Time-Trans after (1). Additionally, to test our claim that a parallel structure is better than a sequential combination, we run an extra experiment where we add a sequential combination of TCN and Transformer (Trans) after (1). Results are shown in Table 5(results of the other two local-global datasets are in [38]). From the FID column we see a clear improvement when adding either a TCN or a Transformer. The sequential combination also improves the performance. Finally, our proposed parallel structure Time-Trans achieves an approximate 40% improvement compared to the sequential one. The other two columns also reflect similar situations. These results indicate the strong ability of our proposed model to generate time series.

Table 5: Ablation study. ‘Seq.’ stands for Sequential. Lower results are better

Components	FID	Discriminative	Predictive
De-Convolution	2.298 ± 0.284	0.490 ± 0.006	0.037 ± 0.002
TCN	0.672 ± 0.097	0.471 ± 0.013	0.026 ± 0.007
Trans	0.815 ± 0.039	0.479 ± 0.009	0.027 ± 0.006
TCN+Trans (Seq.)	0.627 ± 0.031	0.455 ± 0.021	0.022 ± 0.008
Time-Trans	0.395 ± 0.013	0.160 ± 0.054	0.017 ± 0.009

6 Limitations & Border Impacts

Although we showed the strength of our model from multiple perspectives, we acknowledge there are some limitations and propose future work to mitigate these. First, as our model handles unsupervised generation, changes would need to be made for a conditional generation task, to produce user-defined synthetic time series data. Second, aiming at data augmentation, our model requires access to complete time series data. Expanding it to work with incomplete data and address time series imputation challenges would offer significant benefits. Apart from the positive impacts showed in 5.3.4, our model may have possible negative social impact if used unethically (E.g. realistic fake time series data in news may be used to mislead people). It is important to follow ethical practices when using the model.

7 Conclusion

We introduced a novel time series generative model (Time-Transformer AAE), which contains an adversarial autoencoder (AAE) and a key component (Time-Transformer) in the decoder. Via a layer-wise parallelization and a bidirectional cross attention, Time-Transformer AAE exploits the learning ability of Temporal Convolutional Networks and Transformer in extracting local features and building global interaction respectively. We experimentally showed its effectiveness in generating time series data. Additionally, by training on different sizes of data, we showed the proposed model can achieve competitive performance to SOTA methods, even with less training data. Furthermore, we showed the model's utility in imbalanced classification and small dataset augmentation problems.

Acknowledgements: This work is partially supported by NHMRC Development Grant Project 2000173 and The University of Melbourne's Research Computing Services and the Petascale Campus Initiative.

References

- [1] A. AHAMAD, *Generating text through adversarial training using skip-thought vectors*, in Proceedings of the 2019 Conference of the North, Association for Computational Linguistics, 2019.
- [2] E. AKSAN AND O. HILLIGES, *Stcn: Stochastic temporal convolutional networks*, 2019.
- [3] M. ARJOVSKY, S. CHINTALA, AND L. BOTTOU, *Wasserstein gan*, 2017.
- [4] S. BAI, J. Z. KOLTER, AND V. KOLTUN, *An empirical evaluation of generic convolutional and recurrent networks for sequence modeling*, 2018.
- [5] L. CAMPBELL, A. KAICER, D. SLY, C. ISELI, B. WEI, R. BRIGGS, AND S. O'LEARY, *Intraoperative real-time cochlear response telemetry predicts hearing preservation in cochlear implantation*, Otology & Neurotology, 37 (2016).
- [6] Y. CAO, Y. DING, M. JIA, AND R. TIAN, *A novel temporal convolutional network with residual self-attention mechanism for remaining useful life prediction of rolling bearings*, Reliability Engineering & System Safety, (2021).
- [7] W. CHEN, F. WANG, AND H. SUN, *S2tnet: Spatio-temporal transformer networks for trajectory prediction in autonomous driving*, 2022.
- [8] Y. CHEN, X. DAI, D. CHEN, M. LIU, X. DONG, L. YUAN, AND Z. LIU, *Mobile-former: Bridging mobilenet and transformer*, 2021.
- [9] J. CHUNG, K. KASTNER, L. DINH, K. GOEL, A. C. COURVILLE, AND Y. BENGIO, *A recurrent latent variable model for sequential data*, Advances in neural information processing systems, 28 (2015).
- [10] A. DALBERT, A. HUBER, D. VERAGUTH, C. ROOSLI, AND F. PFIFFNER, *Assessment of cochlear trauma during cochlear implantation using electrocochleography and cone beam computed tomography*, Otology & Neurotology, (2016).
- [11] H. A. DAU, E. KEOGH, K. KAMGAR, C.-C. M. YEH, Y. ZHU, S. GHARGHABI, C. A. RATANAMAHATANA, YANPING, B. HU, N. BEGUM, A. BAGNALL, A. MUEEN, G. BATISTA, AND HEXAGON-ML, *The ucr time series classification archive*, October 2018.
- [12] S. DELDARI, D. V. SMITH, H. XUE, AND F. D. SALIM, *Time series change point detection with self-supervised contrastive predictive coding*, in WWW '21, 2021.
- [13] A. DESAI, C. FREEMAN, Z. WANG, AND I. BEAVER, *Timevae: A variational auto-encoder for multivariate time series generation*, 2021.
- [14] A. DOSOVITSKIY, L. BEYER, A. KOLESNIKOV, D. WEISSENBORN, X. ZHAI, T. UNTERTHINER, M. DEHGHANI, M. MINDERER, G. HEIGOLD, S. GELLY, J. USZKOREIT, AND N. HOULSBY, *An image is worth 16x16 words: Transformers for image recognition at scale*, 2020.
- [15] C. ESTEBAN, S. L. HYLAND, AND G. RÄTSCH, *Real-valued (medical) time series generation with recurrent conditional gans*, 2017.
- [16] O. FABIUS AND J. R. VAN AMERSFOORT, *Variational recurrent auto-encoders*, 2014.
- [17] M. FRACCARO, S. K. SØNDERBY, U. PAQUET, AND O. WINTHER, *Sequential neural models with stochastic layers*, NeurIPS, 29 (2016).
- [18] I. J. GOODFELLOW, J. POUGET-ABADIE, M. MIRZA, B. XU, D. WARDE-FARLEY, S. OZAIR, A. COURVILLE, AND Y. BENGIO, *Generative adversarial networks*, 2014.
- [19] J. GUO, S. LU, H. CAI, W. ZHANG, Y. YU, AND J. WANG, *Long text generation via adversarial training with leaked information*, 2017.
- [20] K. G. HARTMANN, R. T. SCHIRRMEISTER, AND T. BALL, *Eeg-gan: Generative adversarial networks for electroencephalographic (eeg) brain signals*, 2018.
- [21] J. HE, D. SPOKOONY, G. NEUBIG, AND T. BERG-KIRKPATRICK, *Lagging inference networks and posterior collapse in variational autoencoders*, in International Conference on Learning Representations, 2019.
- [22] P. HEWAGE, A. BEHERA, M. TROVATI, E. PEREIRA, M. GHAHREMANI, F. PALMIERI, AND Y. LIU, *Temporal convolutional neural (tcn) network for an effective weather forecasting using time-series data from the local weather station*, Soft Comput., 24 (2020).
- [23] A. G. HOWARD, M. ZHU, B. CHEN, D. KALENICHENKO, W. WANG, T. WEYAND, M. ANDREETTO, AND H. ADAM, *Mobilenets: Efficient convolutional neural networks for mobile vision applications*, 2017.
- [24] D. JARRETT, I. BICA, AND M. VAN DER SCHAAR, *Time-series generation by contrastive imitation*, Advances in Neural Information Processing Systems, 34 (2021).
- [25] P. JEHA, M. BOHLKE-SCHNEIDER, P. MERCADO, S. KAPOOR, R. S. NIRWAN, V. FLUNKERT, J. GASTHAUS, AND T. JANUSCHOWSKI, *PSA-GAN: Progressive self attention GANs for synthetic time series*,

- in ICLR, 2022.
- [26] J. JEON, J. KIM, H. SONG, S. CHO, AND N. PARK, *GT-GAN: General purpose time series synthesis with generative adversarial networks*, in Advances in Neural Information Processing Systems, A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, eds., 2022.
- [27] T. KARRAS, T. AILA, S. LAINE, AND J. LEHTINEN, *Progressive growing of gans for improved quality, stability, and variation*, 2017.
- [28] D. P. KINGMA AND M. WELLING, *Auto-encoding variational bayes*, 2013.
- [29] Z. KONG, W. PING, J. HUANG, K. ZHAO, AND B. CATANZARO, *Diffwave: A versatile diffusion model for audio synthesis*, in ICLR, 2021.
- [30] A. KOOCHALI, P. SCHICHTEL, A. DENGEL, AND S. AHMED, *Probabilistic forecasting of sensory data with generative adversarial networks – ForGAN*, IEEE Access, 7 (2019), pp. 63868–63880.
- [31] G. LAI, W.-C. CHANG, Y. YANG, AND H. LIU, *Modeling long- and short-term temporal patterns with deep neural networks*, 2017.
- [32] G. LAI, B. LI, G. ZHENG, AND Y. YANG, *Stochastic wavenet: A generative latent variable model for sequential data*, 2018.
- [33] H. LI, S. YU, AND J. PRINCIPE, *Causal recurrent variational autoencoder for medical time series generation*, AAAI, (2023).
- [34] Y. LI, X. LU, Y. WANG, AND D. DOU, *Generative time series forecasting with diffusion, denoise, and disentanglement*, in Advances in Neural Information Processing Systems, A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, eds., 2022.
- [35] L. LIN, B. XU, W. WU, T. W. RICHARDSON, AND E. A. BERNAL, *Medical time series classification with hierarchical attention-based temporal convolutional networks: A case study of myotonic dystrophy diagnosis*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, 2019.
- [36] Z. LIN, A. JAIN, C. WANG, G. FANTI, AND V. SEKAR, *Using GANs for sharing networked time series data*, in Proceedings of the ACM Internet Measurement Conference, ACM, oct 2020.
- [37] B. LIU, R. TANG, Y. CHEN, J. YU, H. GUO, AND Y. ZHANG, *Feature generation by convolutional neural network for click-through rate prediction*, in WWW'19, 2019.
- [38] Y. LIU, S. WIJEWICKREMA, A. LI, C. BESTER, S. O'LEARY, AND J. BAILEY, *Time-transformer: Integrating local and global features for better time series generation*, 2023. Extended Version.
- [39] A. MAKHZANI, J. SHLENS, N. JAITLEY, I. GOODFELLOW, AND B. FREY, *Adversarial autoencoders*, 2015.
- [40] O. MOGREN, *C-rnn-gan: Continuous recurrent neural networks with adversarial training*, 2016.
- [41] A. v. d. OORD, S. DIELEMAN, H. ZEN, K. SIMONYAN, O. VINYALS, A. GRAVES, N. KALCHBRENNER, A. SENIOR, AND K. KAVUKCUOGLU, *Wavenet: A generative model for raw audio*, 2016.
- [42] A. v. d. OORD, N. KALCHBRENNER, AND K. KAVUKCUOGLU, *Pixel recurrent neural networks*, 2016.
- [43] D. PATHAK, P. KRAHENBUHL, J. DONAHUE, T. DARRELL, AND A. A. EFROS, *Context encoders: Feature learning by inpainting*, 2016.
- [44] H. PEI, K. REN, Y. YANG, C. LIU, T. QIN, AND D. LI, *Towards generating real-world time series data*, 2021.
- [45] C. RAFFEL, N. SHAZER, A. ROBERTS, K. LEE, S. NARANG, M. MATENA, Y. ZHOU, W. LI, AND P. J. LIU, *Exploring the limits of transfer learning with a unified text-to-text transformer*, 2019.
- [46] K. RASUL, C. SEWARD, I. SCHUSTER, AND R. VOLLMGRAF, *Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting*, in ICML, 2021.
- [47] R. SEN, H.-F. YU, AND I. S. DHILLON, *Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting*, in Advances in Neural Information Processing Systems, vol. 32, 2019.
- [48] A. SEYFI, J.-F. RAJOTTE, AND R. T. NG, *Generating multivariate time series with COmmon source coordinated GAN (COSCI-GAN)*, in NeurIPS, 2022.
- [49] K. E. SMITH AND A. O. SMITH, *Conditional gan for timeseries generation*, 2020.
- [50] L. VAN DER MAATEN AND G. HINTON, *Visualizing data using t-sne*, JMLR, 9 (2008), pp. 2579–2605.
- [51] A. VASWANI, N. SHAZER, N. PARMAR, J. USZKOREIT, L. JONES, A. N. GOMEZ, L. KAISER, AND I. POLO-SUKHIN, *Attention is all you need*, 2017.
- [52] M. WIESE, R. KNOBLOCH, R. KORN, AND P. KRETSCHMER, *Quant GANs: deep generation of financial time series*, Quantitative Finance, 20 (2020).
- [53] S. WIJEWICKREMA, C. BESTER, J.-M. GERARD, A. COLLINS, AND S. O'LEARY, *Automatic analysis of cochlear response using electrocochleography signals during cochlear implant surgery*, Plos one, 17 (2022).
- [54] R. YAMASHITA, M. NISHIO, R. K. G. DO, AND K. TOGASHI, *Convolutional neural networks: an overview and application in radiology*, Insights into Imaging, (2018).
- [55] Y. YAN, M. TAN, Y. XU, J. CAO, M. NG, H. MIN, AND Q. WU, *Oversampling for imbalanced data via optimal transport*, in the Thirty-Third AAAI Conference on Artificial Intelligence, 2019.
- [56] C. YANG, J. ZHOU, E. LI, M. WANG, AND T. JIN, *Local-feature and global-dependency based tool wear prediction using deep learning*, Scientific Reports, (2022).
- [57] B. W. YAP, K. A. RANI, H. A. A. RAHMAN, S. FONG, Z. KHAIRUDIN, AND N. N. ABDULLAH, *An application of oversampling, undersampling, bagging and boosting in handling imbalanced datasets*, in Proceedings of the First International Conference on Advanced Data and Information Engineering, Springer Singapore, 2014.
- [58] J. YOON, D. JARRETT, AND M. VAN DER SCHAAR, *Time-series generative adversarial networks*, in NeurIPS, vol. 32, 2019.
- [59] Z. YUE, Y. WANG, J. DUAN, T. YANG, C. HUANG, Y. TONG, AND B. XU, *Ts2vec: Towards universal*

- representation of time series*, 2021.
- [60] G. ZERVEAS, S. JAYARAMAN, D. PATEL, A. BHAMIDI-PATY, AND C. EICKHOFF, *A transformer-based framework for multivariate time series representation learning*, in Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, KDD, 2021.
 - [61] H. ZHANG, T. XU, H. LI, S. ZHANG, X. WANG, X. HUANG, AND D. METAXAS, *Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks*, 2016.
 - [62] W. ZHAO, Y. GAO, T. JI, X. WAN, F. YE, AND G. BAI, *Deep temporal convolutional networks for short-term traffic flow forecasting*, IEEE Access, (2019).

A ECochG dataset

Intra-operative electrocochleography (ECochG) is used to monitor the response of the cochlea to sound during a Cochlear Implant surgery. This surgery is a cost-effective solution for hearing impairment. However, a barrier to patients choosing to receive a cochlear implant is the high probability of losing their natural hearing due to trauma caused during the surgery. Previous studies show that the changes of some ECochG components can reflect such trauma [5, 10]. Specifically, [5] demonstrate that a 30% drop in amplitude of the ‘Cochlear Microphonic’ (CM - one component of the ECochG signal) leads to poorer natural hearing preservation, and thus can be used to predict trauma. This has motivated researchers to develop machine learning models to automatically detect these drops and assist the surgeon in preventing trauma during the surgery [53].

B Details of Imbalanced Datasets

The original ECochG dataset contains data from 77 patients. When sampled and pre-processed as previously mentioned, this results in 13982 time series including two classes. However, this dataset is extremely imbalanced: only 874 instances are positive, which makes it hard to train a machine learning model. One possible solution is augmenting the minority class to make it balanced. As to the UCR datasets, the first two are imbalanced binary datasets while the last one contains 15 classes, we create an imbalanced situation by assigning class 1 as the positive class and all remaining classes as negative (labeled as 0). Here, we provide statistical details of the datasets used in **Model Application** of Section 4. Table 6 shows this information including: number of training data (#Train), number of testing data (#Test), length of each time series (Len), positive rate of training set (Pos_Rate, equals to #Positive/#Train), and number of training data for generative model training (#G-Training).

Table 6: Details of Imbalanced Datasets

Dataset	# Train	# Test	Len	Pos_Rate	# G-Training
ECochG	9787	4195	128	0.063	874
Wafer	1000	6164	152	0.097	97
Herring	64	64	512	0.391	25
SwedishLeaf	500	625	128	0.058	29

C Experimental Setup

C.1 Data Pre-processing For the preliminary datasets, we follow the settings in previous work [58, 13]. We use a sliding window of size 24 to sample the data resulting in time series of length 24, and re-scale all data to $[0, 1]$ using the min-max normalization formula: $x' = \frac{x - \min(x)}{\max(x) - \min(x)}$, where x is the original data, and $\min(x)$, $\max(x)$ are minimum and maximum of the data respectively.

We choose a longer length (128) for time series in local-global datasets, to preserve more local and global patterns, and to evaluate the model’s ability to process longer data. We use min-max normalization to re-scale all the data to $[0, 1]$ as mentioned previously.

C.2 Model Settings We search the hyper-parameters of the autoencoder based on the average reconstruction performance on the preliminary datasets. We divide the datasets into training and validation sets, and search the hyper-parameters including layer number, filter number, kernel size, stride number, head number, head size, and dilation rate using the training set. The value range of each parameter is shown in Table 7. As a result, the

Table 7: Value range of Hyper-parameters

Hyper-parameters	layer number	filter number	kernel size	stride number	head number	head size	dilation rate
Value range	2,3	64-512	4,6,8	1,2	2,3,4	64-256	1,2,4,16

encoder is a 3-layer 1-dimensional CNN with 64, 128, and 256 filters and ‘relu’ activation. All filters have kernels of size 4, and move 2 unit-step each time (stride of 2). After this convolutional process, the results are flattened and go through a fully connected layer which maps them into the latent code with pre-defined dimensions. Here we use

8-dimensional latent code for preliminary datasets (original: 24-step) and 16-dimensional vectors for local-global datasets (original: 128-step).

In the decoder, the de-convolutional part has 2 transposed convolutional layers with 128 and 64 filters. All filters have kernels of size 4 and strides of 2 units. Then, the outputs are reshaped and mapped into the original data dimensions (length L and channels C) via reshape and fully connected layers, which results in prototypes of the time series. The following Time-Transformer has two blocks, which represents a TCN that has two hidden dilated convolutional layers. The hidden layers have C filters with kernel size of 4 and dilation rate of 1 and 4 respectively. The Transformer blocks combined with each dilated convolutional layer consist of a 3-head self-attention layer with head size of 64 and a feed-forward convolutional layer. The cross attention is also a 3-head attention module with a head size of 64. It takes inputs from different sides which makes it different from the self-attention module ahead. The discriminator, that accomplishes the adversarial process, is a 2-layer fully connected layer with 32 hidden units in both layers and ‘relu’ activation.

We follow the original AAE training procedure to train the model: We first update the encoder and decoder using the reconstruction loss (Mean Squared Error). Then, the encoder and discriminator are updated with respect to the adversarial loss consisting of a discriminative loss and a generative loss (both are cross-entropy). We use the Adam optimiser to update all the losses. The learning rate for reconstruction loss is 0.005 initially, and reduces to 0.0001 via a polynomial decay function (we directly implemented it using the tensorflow platform, the power of the polynomial is 0.5). Both the discriminative loss and generative loss have initial learning rates of 0.001, which also reduces to 0.0001 via a polynomial decay function (same as previous). The default training epoch is set to 1000. We provide the source code and a tutorial notebook here.

D Additional Experimental Results

We provide more results from the experiments on visualization, training size, ablation study and model application as mentioned in the paper.

D.1 Visualization t-SNE plots of the preliminary datasets are shown in Figure 7.

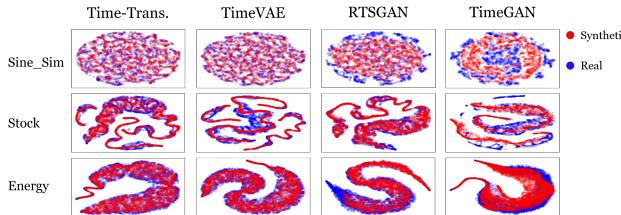


Figure 7: t-SNE visualizations on preliminary datasets

D.2 Longer time series data Most existing related works [58, 13, 44] for synthetic time series generation have used 24-length time series as their experimental datasets. A recent work (PSA-GAN [25]) has claimed to generate long time series and the longest time series used in their work are 256-length. They apply a sampling method from LSTNet [31] to the electricity datasets, to produce these 256-length time series. Hence, we use this dataset, with the same sampling method as [31], to evaluate our model. In addition, we further evaluate our model using another two real-world time series datasets from the UCR Archive [11]: Trace (cropped to 256-length) and ShapesAll (512-length). The results are shown in Table 8 below. We can see our model still performs well with these longer (256, 512-length) time series data, as it has the best performance against the baselines. (we only use the two best baseline models: RTSGAN and TimeVAE).

D.3 Local & Global Feature Learning details Table 9 lists the quantitative evaluations results of each model, which also shows the effectiveness of our model in extracting both global and local features simultaneously, as it gets the best scores with respect to all metrics for the mixture dataset. RTSGAN can capture the global trends but fails to extract local properties as it performs well when data contains more global features. On the other hand, TimeVAE is good at local features learning according to the results. TimeGAN and RCGAN seem to

Table 8: Long time series evaluation (lower is better for all metrics)

Model	Dataset	FID	Disc.	Pred.
Oracle	Electricity	N/A		0.010±0.004
TimeGAN		9.501±1.654	0.488±0.016	0.105±0.014
TimeVAE		1.655±0.621	0.415±0.037	0.034±0.005
RTSGAN		0.896±0.147	0.397±0.007	0.021±0.007
Time-Trans.		0.338±0.058	0.057±0.027	0.014±0.006
Oracle	Trace	N/A		0.062±0.001
TimgGAN		17.011±1.859	0.499±0.003	0.229±0.014
TimeVAE		7.024±0.228	0.482±0.015	0.116±0.004
RTSGAN		7.959±0.217	0.497±0.006	0.175±0.002
Time-Trans.		4.014±0.321	0.261±0.059	0.095±0.007
Oracle	ShapesAll	N/A		0.051±0.000
TimeGAN		2.795±0.098	0.326±0.076	0.188±0.003
TimeVAE		0.481±0.042	0.122±0.071	0.112±0.002
RTSGAN		3.648±0.107	0.296±0.073	0.248±0.001
Time-Trans.		0.445±0.029	0.034±0.045	0.094±0.005

be able to learn a single feature to some extent, but they fail to capture the mixed features.

Table 9: Evaluation results for global & local features learning (lower scores are better)

Model	Benchmark	Local	25%G	50%G	75%G	Global
TimeGAN	FID	4.736±0.224	5.274±0.561	5.195±0.316	6.139±0.738	1.597±0.294
RTSGAN		0.299±0.068	0.823±0.042	1.788±0.091	0.462±0.018	0.163±0.055
TimeVAE		0.271±0.074	0.597±0.054	1.081±0.068	0.934±0.076	0.241±0.037
Time-Trans.		0.317±0.071	0.463±0.038	0.474±0.014	0.441±0.052	0.229±0.019
TimeGAN	Disc.	0.178±0.062	0.476±0.028	0.485±0.011	0.495±0.006	0.048±0.007
RTSGAN		0.022±0.019	0.067±0.037	0.294±0.078	0.062±0.049	0.012±0.004
TimeVAE		0.015±0.026	0.051±0.040	0.177±0.073	0.183±0.045	0.018±0.009
Time-Trans.		0.029±0.027	0.055±0.035	0.061±0.060	0.063±0.051	0.017±0.015
Oracle	Pred.	0.009±0.002	0.014±0.003	0.024±0.003	0.021±0.007	0.014±0.008
TimeGAN		0.077±0.018	0.098±0.027	0.103±0.043	0.106±0.011	0.052±0.013
RTSGAN		0.014±0.006	0.046±0.013	0.073±0.015	0.034±0.018	0.027±0.012
TimeVAE		0.013±0.006	0.031±0.007	0.052±0.011	0.058±0.009	0.031±0.005
Time-Trans.		0.013±0.005	0.026±0.007	0.031±0.007	0.033±0.011	0.030±0.009

D.4 Imbalanced Classification Table 10 lists imbalanced classification results on UCR datasets: Wafer, Herring, and SwedishLeaf.

D.5 Small Dataset Augmentation We provide the detailed results on small datasets augmentation in Table 11.

D.6 Choice of Encoder Table 12 lists different performance outcomes from our model with two different encoders, namely, CNN and Time-Transformer. The first is the simple design which we used in our proposed model, and the second is the one with a Time-Transformer module inserted in the encoder. We test their performance on the Sine_Sim dataset to briefly investigate how much improvement a complex encoder can bring to the model. As shown in Table 12, using a Time-Transformer encoder improves the results a little, but it requires much more time to train. This indicates that a simple encoder can generate relatively good synthetic data with high efficiency.

D.7 Additional ablation study Table 13 shows the results of ablation study with respect to ‘Sine_Cpx’ and ‘ECochG’ datasets. Both show performance improvements when we add each part.

D.8 Computation Details We ran all the models on the three preliminary datasets and recorded their running time. The results are provided in Table 14 (all models were run on High Performance Computer with Intel(R)

Table 10: Imbalanced Classification Evaluation for UCR datasets (higher results are better)

Datasets	Components	Accuracy	Recall	Precision	AUC_ROC	AUC_PRC
Wafer	No Aug	0.9919	0.9639	0.9610	0.9981	0.9826
	Replication	0.9924	0.9835	0.9478	0.9979	0.9855
	Jittering	0.9933	0.9774	0.9643	0.9985	0.9892
	RTSGAN	0.9940	0.9759	0.9686	0.9973	0.9847
	TimeVAE	0.9932	0.9789	0.9587	0.9978	0.9868
	Time-Trans.	0.9945	0.9849	0.9618	0.9992	0.9928
Herring	No Aug	0.5469	0.3461	0.5625	0.6549	0.5762
	Replication	0.6563	0.4615	0.6000	0.7287	0.6429
	Jittering	0.6250	0.3846	0.5556	0.6781	0.5455
	RTSGAN	0.6563	0.4231	0.6111	0.7146	0.6307
	TimeVAE	0.6406	0.3846	0.5882	0.6817	0.5891
	Time-Trans.	0.6875	0.4615	0.6667	0.7712	0.6455
SwedishLeaf	No Aug	0.9264	0.0000	0.0000	0.9383	0.4296
	Replication	0.8848	0.8043	0.3700	0.9487	0.5337
	Jittering	0.9279	0.3043	0.5185	0.9426	0.5230
	RTSGAN	0.9232	0.1956	0.4500	0.9418	0.4656
	TimeVAE	0.9360	0.6304	0.5577	0.9439	0.5433
	Time-Trans.	0.9408	0.8478	0.5652	0.9536	0.6341

Table 11: Data Augmentation Evaluation (higher results are better)

Dataset	Wafer			Herring			Swedisleaf			
	Model	RTSGAN	TimeVAE	Time-Trans.	RTSGAN	TimeVAE	Time-Trans.	RTSGAN	TimeVAE	Time-Trans.
Accuracy	+0%	0.9940	0.9932	0.9945	0.6563	0.6406	0.6718	0.9232	0.9360	0.9408
	+25%	0.9943	0.9940	0.9947	0.6563	0.6563	0.6718	0.9247	0.9376	0.9423
	+50%	0.9949	0.9945	0.9951	0.6718	0.6563	0.6875	0.9279	0.9408	0.9437
	+75%	0.9950	0.9947	0.9952	0.6875	0.6718	0.6875	0.9312	0.9440	0.9472
	+100%	0.9954	0.9948	0.9961	0.6875	0.6718	0.7031	0.9360	0.9456	0.9488
	+0%	0.9973	0.9978	0.9987	0.7146	0.6817	0.7712	0.9418	0.9439	0.9536
AUC_ROC	+25%	0.9980	0.9979	0.9989	0.7256	0.7086	0.7758	0.9415	0.9526	0.9528
	+50%	0.9985	0.9982	0.9990	0.7316	0.7138	0.7828	0.9464	0.9532	0.9578
	+75%	0.9988	0.9985	0.9991	0.7398	0.7378	0.7891	0.9553	0.9582	0.9609
	+100%	0.9990	0.9986	0.9993	0.7537	0.7419	0.7936	0.9569	0.9603	0.9628
	+0%	0.9847	0.9868	0.9928	0.6307	0.5891	0.6455	0.4656	0.5933	0.6341
AUC_PRC	+25%	0.9856	0.9869	0.9929	0.6367	0.5936	0.6479	0.4827	0.6536	0.6435
	+50%	0.9866	0.9869	0.9932	0.6420	0.5982	0.6505	0.5324	0.6548	0.6642
	+75%	0.9876	0.9870	0.9935	0.6497	0.6001	0.6628	0.5922	0.6934	0.6993
	+100%	0.9896	0.9873	0.9937	0.6603	0.6129	0.6718	0.6126	0.7330	0.7460

Table 12: Comparison between different encoders (lower results are better)

Encoder	Training Time (s)	FID	Discriminative	Predictive
CNN	530.24	0.283±0.023	0.131±0.021	0.051±0.015
Time-Transformer	1651.33	0.256±0.014	0.120±0.011	0.036±0.010

Xeon(R) Gold 6132 CPU and v100 GPU, values are formatted in seconds):

For the memory comparison, we list the trainable parameters of each model in the Table 15 (the parameters were calculated with input size of (24, 1)).

From the two tables, we can see our model and TimeVAE are much faster than other models though they have more parameters. It is because most of the parameters of these two models are based on CNNs where parameters can be processed in parallel, while other models are mostly using RNNs which are much slower than CNNs because they can only process sequentially. (please see <https://github.com/baidu-research/DeepBench> for a detailed comparison between these two architectures). Thus, even though we designed a complex model containing more parameters than RNN-based models, it can still be faster.

Table 13: Extra Ablation Study

Dataset	Components	FID	Discriminator	Predictor
Sine_Cpx	De-Conv	6.976±0.157	0.490±0.021	1.832±0.526
	TCN	2.570±0.067	0.417±0.270	0.529±0.057
	Transformer	2.472±0.079	0.397±0.253	0.484±0.025
	TCN+Trans (Sequential)	1.721±0.055	0.193±0.181	0.037±0.013
	Time-Transformer	1.502±0.062	0.168±0.041	0.032±0.006
ECochG	De-Conv	0.781±0.099	0.461±0.048	0.092±0.011
	TCN	0.502±0.041	0.288±0.192	0.027±0.013
	Transformer	0.513±0.045	0.256±0.164	0.027±0.007
	TCN+Trans (Sequential)	0.402±0.031	0.162±0.059	0.017±0.005
	Time-Transformer	0.348±0.024	0.104±0.012	0.013±0.006

Table 14: Running Time Comparison

Models	Sine_Sim	Stock	Energy
TimeGAN	2589.16	2870.19	4776.35
TimeVAE	183.26	286.52	1301.81
RTSGAN	810.49	1298.74	2433.31
Time-Trans.	176.69	299.36	1313.47

Table 15: Trainable Parameters Comparison

Models	TimeGAN	TimeVAE	RTSGAN	Time-Trans.
Parameters	70712	266893	64530	217942

D.9 Justification of Baseline Models As mentioned in the main part of the paper, we calculate the Predictive Score in a different way to the related works. We used the one-step-ahead forecast. Assume we have a time series $x = [t_1, \dots, t_n]$, where $t_i = [t_i^{(1)}, \dots, t_i^{(C)}]^T$. The inputs are the first $n - 1$ time steps of all dimensions ($x_{train} = [t_1, \dots, t_{n-1}]$), and the model learns to predict the last time step ($y_{train} = [t_n]$). The loss is the mean absolute error between the predictions and the ground truth ($|y_{train} - y_{pred}|$). However, most related works directly use the method from TimeGAN. There, the inputs to the predictor are the first $n - 1$ steps of the first $C - 1$ dimensions ($x_{train} = [t'_1, \dots, t'_{n-1}]$, where $t'_i = [t_i^{(1)}, \dots, t_i^{(C-1)}]^T$). The model learns to predict the last dimension with one more step ($y_{train} = [t_2^{(C)}, \dots, t_n^{(C)}]$). The loss is the mean absolute error with respect to the new outputs. Furthermore, TimeGAN used a sigmoid function as the last activation (while we don't use any activation in the last layer).

These methodological differences led to the different results of the Predictive Score between our work and that of related works. Now, we have also used their method to get a new set of Predictive Scores. The results are shown in Table 16 below. These results are similar to the related works on the corresponding datasets, taking

Table 16: Predictive Score using the same method in TimeGAN (lower scores are better)

Model	Sine_sim	Stock	Energy	Music	Sine_Cpx	ECochG
Oracle	0.215±0.001	0.037±0.000	0.249±0.002	0.080±0.001	0.168±0.003	0.008±0.000
TimeGAN	0.257±0.004	0.040±0.001	0.278±0.005	0.106±0.003	0.234±0.005	0.013±0.002
TimeVAE	0.218±0.001	0.040±0.000	0.254±0.001	0.102±0.002	0.189±0.001	0.011±0.003
RTSGAN	0.242±0.002	0.038±0.000	0.253±0.005	0.097±0.001	0.227±0.006	0.014±0.002
Time-Trans.	0.223±0.002	0.039±0.001	0.261±0.001	0.086±0.001	0.187±0.001	0.009±0.000

into account acceptable variances caused by random seeds and hardware environments. This provides additional evidence that our baseline models were implemented properly. On the other hand, we can see although the values have changed, our claim in the paper still stands: our model performs the best for the three local-global datasets (Music, Sine_Cpx, ECochG), which indicates its effectiveness in extracting global and local features simultaneously.