# Generative Models for Long Time Series: Approximately Equivariant Recurrent Network Structures for an Adjusted Training Scheme

**Anonymous authors**
**Paper under double-blind review**

## Abstract

We apply a novel training scheme to a specific implementation of a Variational Autoencoder (VAE), which, in combination, we refer to as the Recurrent Variational Autoencoder Subsequent Train (RVAE-ST). This method progressively increases the sequence length during training, leveraging the sequence-length-independent parameterization of the model to address the challenge recurrent layers face when handling long sequences, particularly for datasets exhibiting approximate stationarity. Our experiments demonstrate that this approach significantly improves the model's performance, especially for datasets with periodic behavior. Compared to other recurrent and convolutional-based generative models, our method excels in generating synthetic data for long sequences of $l = 1000$, with notable improvements in both sample quality and the distribution of the generated datasets. We evaluate the effectiveness of our approach using multiple metrics, including the discriminative score, evidence lower bound (ELBO), and visualizations of embeddings generated by t-SNE and PCA.

## 1 Introduction

Time series data, particularly sensor data, plays a crucial role in science, industry, energy, and health. With the increasing digitization of companies and other institutions, the demand for advanced methods to handle and analyze time series sensor data continues to grow. Sensor data often exhibits distinct characteristics: it is frequently multivariate, capturing several measurements simultaneously, and may involve high temporal resolutions, where certain anomalies or patterns of interest only become detectable in sufficiently long sequences. Furthermore, such data commonly displays approximate stationarity or quasi-periodic behavior, reflecting repetitive patterns influenced by the underlying processes. These unique properties present both opportunities and challenges in the development of methods for efficient data synthesis and analysis, which are essential for a wide range of applications. Time series data analysis spans tasks such as forecasting (Siami-Namini et al., 2019), imputation (Tashiro et al., 2021; Luo et al., 2018), anomaly detection (Hammerbacher et al., 2021), and data generation. Of these, data generation stands out as the most general task, as advances in generative methods often yield improvements across the entire spectrum of time series applications (Murphy, 2022).

Recurrent neural networks, particularly Long Short-Term Memory (LSTM) networks (Hochreiter & Schmidhuber, 1997), are well-known for their ability to model temporal dynamics and capture dependencies in sequential data. However, their effectiveness tends to diminish with increasing sequence length, as maintaining long-term dependencies can become challenging (Zhu et al., 2023) where in contrast, convolutional neural networks (CNNs) (LeCun et al., 1998) demonstrate superior scalability for longer sequences (Bai et al., 2018). For instance, TimeGAN (Yoon et al., 2019) represents a state-of-the-art approach for generating synthetic time series data, particularly effective for short sequence lengths. In its original paper, TimeGAN demonstrates its capabilities on samples with sequence lengths of $l = 24$, showcasing limitations of LSTM-based architectures. By contrast, a model like WaveGAN (Donahue et al., 2019), which is built on a convolutional architecture, is trained on significantly longer sequence lengths, with $l = 16384$ at minimum.

This contrast highlights the fundamental differences and capabilities between recurrent and convolutional networks.

The limitations of LSTMs in modeling long-term dependencies are not restricted to time series data but also impact their performance in other domains, such as natural language processing (NLP). Early applications of Attention mechanisms that were integrated with recurrent neural networks such as LSTMs (Bahdanau, 2014) have nowadays been replaced by Transformer architectures (Vaswani et al., 2017), which especially excel in data-rich tasks due to their attention mechanisms and parallel processing capabilities. While Transformer architectures have shown remarkable results in NLP (Radford et al., 2019), their application to time series data is often limited by the comparatively smaller dataset sizes typically available and their lack of a strong inductive bias for sequential structures, which is crucial for time series where the arrangement of data points inherently defines the underlying temporal dependencies.

Among the primary approaches for generative modeling of time series, three dominant frameworks have emerged: Generative Adversarial Networks (GANs) (Goodfellow et al., 2020), Variational Autoencoders (VAEs) (Kingma & Welling, 2014), and, more recently, Diffusion Models (Ho et al., 2020). Diffusion Models have demonstrated impressive capabilities in modeling complex data distributions, but their significant computational demands, high latency, and complexity make them less practical for many applications (Yang et al., 2024). Moreover, in terms of practical applications, there are often constraints in both time and computational resources, which limit the feasibility of performing extensive fine-tuning for each individual dataset. A general, well-performing approach that is both simple and efficient is therefore more desirable. In this context, VAEs stand out for their simplicity and direct approach to probabilistic modeling. In our work, we focus on VAEs and propose a novel method for training VAEs with recurrent layers to handle longer sequence lengths. We argue that VAEs are particularly suited for generation of time series data, as they explicitly learn the underlying data distribution, making them robust, interpretable, and straightforward to implement. While our approach is demonstrated with VAEs, the underlying method is general and could be adapted for other generative frameworks using recurrent layers.

Our major contributions are:

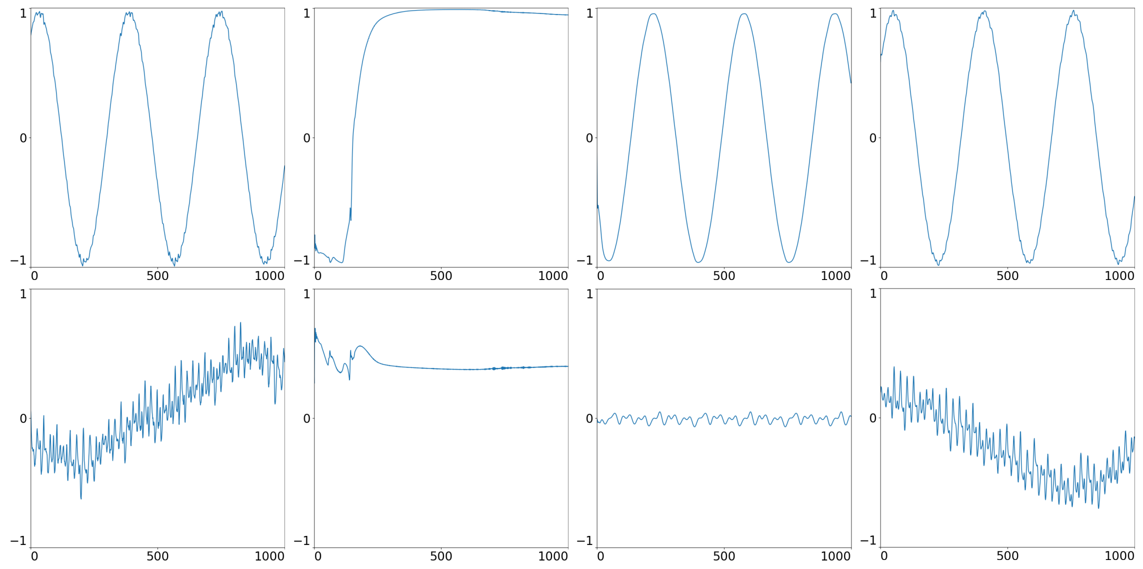
- We present a network architecture for Variational Autoencoders (VAEs) that integrates an approximately equivariant structure with recurrent layers. This architecture introduces an inductive bias toward stationary datasets, enhancing its suitability for modeling time series with such characteristics. A key feature of this model is its parameter count, which remains independent of the sequence length.
- We propose a novel method, combining the described network architecture with a tailored training scheme in which the sequence length is gradually increased during training. This method, which we term the Recurrent Variational Autoencoder Subsequent Train (RVAE-ST), leverages the model's sequence-length-independent parameterization. By progressively adapting the sequence length, it addresses a major limitation of recurrent layers: their difficulty in effectively handling long sequences. The method is particularly well-suited to datasets exhibiting approximate stationarity.
- We demonstrate the superiority of our approach through multiple experiments, comparing it to both recurrent and convolutional-based generative models. The validation metrics used to evaluate performance include the discriminative score, evidence lower bound (ELBO), and visualizations of embeddings generated by t-SNE and PCA.

Our implementation, including preprocessing and model training scripts, is available at `https://github.com/ruwenflk/rvae-st`.

## 2 Prerequisites

### 2.1 Variational Autoencoder

Given the input dataset X, the goal is to find a probability density $p_\theta$ with high marginal likelihood (or evidence) $p_\theta(x)$ for $x \in X$. By introducing $z = z_{1:m}$ latent variables and assuming the joint density



(a) Original data (b) TimeGAN (c) RVAE single train (d) RVAE-ST (ours)

Figure 1: This figure shows three excerpts from samples of the electric motor dataset (4.1), each with a sequence length of $l = 1000$. Sample (a) is taken from the original dataset. Sample (b) is generated using TimeGAN (Yoon et al., 2019), a state-of-the-art approach in time series generation, particularly leveraging recurrent layers. Sample (c) is generated using our model trained in the conventional way, while sample (d) is generated using our model trained in our suggested subsequent training schema. The first row in the figure displays the voltage of one of the phases. Notably, in the original samples (a), the extremities of the voltage waveforms show pronounced volatility, particularly at the peak and trough points. In the subsequently trained model (d), this behavior is still clearly visible in contrast to the conventionally trained model (c). The second row shows the effective motor current in the fixed coordinates of the stator. This channel exhibits both a high-frequency component that makes the graph appear very noisy, as well as a low-frequency oscillation that characterizes the long-term behavior. The conventionally trained model (c) has only learned the high-frequency component, with significant smoothing applied, while the sample from the subsequently trained model (d) closely resembles the original dataset (a). TimeGAN was unable to learn this dataset at the given sequence length.

$p_\theta(x, z) = p_\theta(z)p_\theta(x|z)$, we get the intractable integral $p_\theta(x) = \int p_\theta(z)p_\theta(x|z)dz$. The evidence $p_\theta(x)$ can be approximated by computing the evidence lower bound or ELBO

$$\mathcal{L}_{\theta,\phi}(x) = \mathbb{E}_{q_\phi(z|x)}[\log\ p_\theta(x|z)] + D_{\mathbb{KL}}(q_\phi(z|x)\,||\,p_\theta(z)) = \mathcal{L}_E + \mathcal{L}_R \leq \log p_\theta(x).$$

$\mathcal{L}_E$ is the reconstruction loss (negative log likelihood) and $\mathcal{L}_R$ is the KL-Divergence loss (Murphy, 2022). $p_\theta(z)$ is the prior distribution which is usually set to $\mathcal{N}(0, I)$. The ELBO $\mathcal{L}_{\theta,\phi}(x) \leq \log p_\theta(x)$ is a lower bound to the log marginal likelihood and is maximized by training a VAE.

A VAE is a generative model that maps a sample $x \in X$ into a probability distribution in latent space $z$ by using an inference network $q_\phi(z|x) = \mathcal{N}(z|\mu, \text{diag}(\exp(\log(\sigma))))$ with $(\mu, \log(\sigma)) = e_\phi(x)$ where $e_\phi(x)$ is the encoder. The latent variable $z = \mu + \sigma \odot \epsilon$ where $\odot$ is the entrywise multiplication with $\epsilon = \mathcal{N}(0, I)$ is then passed through the generator model or decoder network $p_\theta(x|z)$ giving us a reconstructed sample $\widetilde{x}$. Generating new timeseries samples works by just taking a sample from the known prior distribution $p_\theta(z)$ and feeding it through the generative model $p_\theta(x|z)$.

## 3 Methods

### 3.1 Equivariant Vec2Seq generator

The objective of our model development was to create a generator that incorporates an inductive bias toward time-shift invariance in its learned distribution $p_\theta(x)$ , while being capable of generating sequences of variable lengths from fixed-length inputs, as seen in vector-to-sequence (Vec2Seq) models. This design choice was motivated by the idea that such properties would improve model performance, particularly on (semi-)stationary and quasi-periodic time series.

Assuming that all variables represent probabilities rather than samples, we define an approximately time-shift invariant $p_\theta(x)$ as follows:

$$\int p_\theta(x|z)p_\theta(z)\,dz = p_\theta(x) \approx p_\theta(\tau(x)) = \int p_\theta(\tau(x)|z)p_\theta(z)\,dz,$$

where $\tau$ is the time-shift operator. This holds when $p_\theta(x|z) \approx p_\theta(\tau(x)|z)$, which implies that a generator model with an inductive bias toward time-shift invariance can be achieved using a equivariant network topology.

Assuming two key properties: $p_\theta(x)$ is time-shift invariant, and we aim to design an equivariant network structure, we now examine common types of layers: Convolutional layers are commonly employed for building equivariant network structures due to their ability to recognize patterns in data regardless of their position within the input. This property makes them highly effective especially in image processing. However, in the context of time series, convolutional layers alone are insufficient for generating sequences of variable lengths from fixed-length inputs. Recurrent neural networks (RNNs), on the other hand, excel in handling variable-length sequences, making them ideal for constructing Vec2Seq models. However, because they rely on the order of input data and continuously update their states, they are only approximately equivariant. We further note that transformers are not translation equivariant but permutation equivariant. They can handle inputs in any order due to their self-attention mechanism, which allows them to consider all elements of the input sequence simultaneously. Transformers lack the ability to capture temporal dynamics as effectively as recurrent layers. While positional encodings are used in transformers to provide some sense of order, they do not model the sequential dependencies and temporal evolution of data as naturally as RNNs do. Dense layers are not equivariant at all. They treat each input independently without considering the order or position.

### 3.2 RVAE-ST

The inference network $q_\phi(z|x)$ is implemented using stacked LSTM layers. Given the final point in time of a sequence, the output of the last LSTM layer is passed through two linear layers to determine $\mu$ and $\log(\sigma)$, which are then used to sample the latent variable $z$. Next, the generative network $p_\theta(x|z)$ reconstructs the data from the latent variable $z$. The network begins by repeating $z$ for each time step, followed by stacked LSTM layers. Finally, a time-distributed linear layer is applied in the output, which ensures that the temporal structure is preserved and that the number of trainable parameters remains independent of the sequence length. This independence enables the model to handle longer sequences more effectively and facilitates the use of an adapted training scheme with increasing sequence lengths. Moreover, this design enhances the inductive bias towards stationary time series, making the model particularly well-suited for such data. The network topology of RVAE-ST is shown in figure 2.

Although the combination of RNNs with time-distributed layers is not entirely novel, it has not been widely emphasized in the literature. We offer a fresh perspective by focusing on its application to stationary data, alongside an adapted training scheme, demonstrating how this architecture can deliver high performance in specific contexts. The details and hyperparameters are parovided in the appendix A.1.

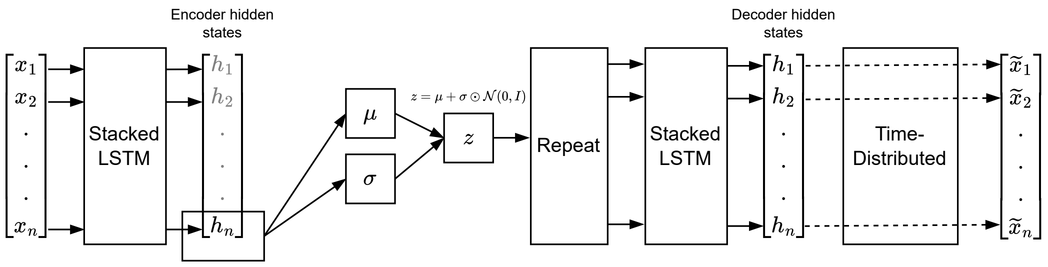


Figure 2: This figure shows the network topology of our model. The encoder and the decoder are built off stacked LSTM-layers. The output of the inference network, $\mu$ and $\sigma$, is determined by the encoder's final hidden states $h_n$. The latent variable $z$ is sampled a single time and then gets repeated (copied) for every point in time as the decoder input. Each individual output of the decoder is passed through a time-distributed linear layer, which generates the sequence by applying the same transformation at each time step. Equivariance is maintained throughout the network. The amount of trainable parameters is the same for varying sequence lengths.

### 3.3 Training scheme for sequence lengths

Training a recurrent neural network such as an LSTM to produce consistent long time sequences is challenging, as recurrent layers have a limited capacity to preserve information over extended temporal ranges. Here, we suggest a training scheme that allows training over longer time sequences for our RVAE-ST model, which is visualized in figure 2.

We begin by splitting the dataset into smaller chunks, as is commonly done. The model is initially trained on short sequences, which ensures stable training and facilitates faster progress. After training on short sequences, we progressively increase the sequence length, rebuild the dataset by splitting it into new, longer chunks, and continue training on the longer sequences. Each training phase is completed when no improvement is observed in the validation loss for a predefined number of epochs. This process is repeated as the sequence length increases. This method stabilizes the training process for long time sequences and improves the final results, as demonstrated in our experiments.

We motivate this probabilistically for a time series $x$ of length $l$, hidden features distributed over time $h$ again of length $l$, and a fixed latent vector $z$, where we have a recurrent structure over the $h$ via

$$p(x, h, z) = p(z) \prod_{i=1}^{l} p(h_i | z, h_{i-1}, \ldots, h_1) \cdot p(x_i | h_i).$$

The generative model $p(x, h|z)$ can now be approximated by only looking $t$ steps back in time.

$$\begin{aligned} p(x, h|z) &= \prod_{i=1}^{l} p(h_i | z, h_{i-1}, \ldots, h_1) \cdot p(x_i | h_i) \\ &\approx \prod_{i=1}^{l} p(h_i | z, h_{i-1}, \ldots, h_{\max(1, i-t)}) \cdot p(x_i | h_i) \end{aligned}$$

Hence, training to generate shorter sequences yields an approximation of generating long sequences. We do not provide general recommendations for which initial sequence length or increment values work best, as the sequence lengths chosen in our experiments are somewhat arbitrary and may vary depending on the dataset.

## 4 Experiments

In our experiments, we compare the performance of RVAE-ST to comparison models. We emphasize that, to ensure better comparability, we did not perform extensive hyperparameter tuning in our experiments.

In all experiments, including different datasets and varying sequence lengths, we used exactly the same hyperparameters on the model. For the training procedure, we started with a sequence length of 100 and progressively increased it by 100 in each subsequent training phase, until reaching a maximum sequence length of 1000. In our experiments, we compare the performance of the models at sequence lengths of 100, 300, 500, and 1000. We employ the following methods to evaluate performance: short-term consistency measurements based on independently generated ELBOs, the discriminative score, and visual comparisons between the training data distribution and the sampled distribution, specifically using PCA and t-SNE.

## 4.1 Data Sets

For our experiments we use three multivariate sensor datasets with it's typical semi-stationary behavior. We specifically selected datasets with a minimum size, as this is necessary for training generative models effectively, while still ensuring adequate diversity and the ability to robustly capture underlying patterns and structures.

**Electric motor (EM)(Wißbrock & Müller, 2025; Mueller, 2024):** This dataset was collected from a three-phase motor operating at a constant speed and load, resulting in a periodic behavior with similar, repeating patterns. The data was recorded at a sampling rate of 16 kHz. Out of the twelve channels initially available, four were removed as they exhibited discrete behavior or abrupt changes, rather than continuous, smooth patterns suitable for learning. It has ~500,000 datapoints and it is the most stationary dataset we use in our experiments.

**Ecg data (ECG)[1] Goldberger et al. (2000):** This dataset contains a two-channel electrocardiogram recording from the MIT-BIH Long-Term ECG Database. It has nearly 10 million time steps of which we use the first 500,000 for training. The ECG signals are nearly periodic but exhibits variations in frequency and occasional irregular arrhythmias, making the dataset slightly less stationary than the EM dataset. While ECG data serves as a suitable example in generating long sequences, our objective is not to produce medically usable data. We acknowledge that specialized models are likely more appropriate for medical applications, e.g. (Neifar et al., 2023).

**ETTm2 (ETT)[2]:** The ETTm2 dataset, collected between 2016 and 2018, consist of sensor measurements such as load and oil temperature from electricity transformers. These datasets contain short-term periodical patterns, long-term periodical patterns, long-term trends, and various irregular patterns. Out of the three datasets, it is the smallest, containing 69680 datapoints. It was published in the context of time series forecasting Zhou et al. (2021a) and is a widespread used dataset (Zhou et al., 2021b; Zhang et al., 2024; Zhu et al., 2023). Out of the three in this paper, is the least stationary dataset.

## 4.2 Comparison Models

In this subsection, we describe the baseline models selected for comparison in our experiments. These models are chosen for their relevance to time series generation and their established use in similar contexts.

**TimeGAN**(Yoon et al., 2019)**:** A GAN-based model that is considered state-of-the-art in generation of times series data. TimeGAN's generator has a recurrent structure like RVAE-ST. A key difference is that it's latent dimension is equal to the sequence length. Notably, equivariance on this model is lost on the output layer of the generator which maps all hidden states at once through a linear layer to a sequence. On its initial paper release, TimeGAN was tested and compared to other models on a small sequence length of $l = 24$.

**WaveGAN** (Donahue et al., 2019)**:** A GAN-based model developed for generation of raw audio waveforms. WaveGAN's generator is based on convolutional layers. It doesn't rely on typical audio processing techniques like spectrogram representations and is instead directly working in the time domain, making it also suitable for learning time series data. It is designed to exclusively support sequence lengths in powers of 2, specifically $2^{14}$ to $2^{16}$. Notably, WaveGAN loses it's equivariance on a dense layer between the latent dimension and the

[1]https://physionet.org/content/ltdb/1.0.0/14157.dat
[2]https://github.com/zhouhaoyi/ETDataset

generator, however the generator itself completely maintains equivariance with its upscaling approach. In our experiments, it was trained with the lowest possible sequence length of $2^{14}$, and the generated samples were subsequently split to match the required sequence length. In (Yoon et al., 2019), WaveGAN was outperformed by TimeGAN on low sequence length.

**TimeVAE** (Desai et al., 2021b)**:** A VAE-based model designed for time series generation using convolutional layers. Analogous to WaveGAN, it loses equivariance between the latent dimension and the decoder and additionally it loses equivariance on the output layer where a flattened convolutional output is passed through a linare layer. It has demonstrated performance comparable to that of TimeGAN.

**RCGAN** (Esteban et al., 2017)**:** A GAN-based model with a recurrent architecture designed for the generation of time series data. It was specifically developed with a focus on medical applications, making it particularly suitable for our experiments involving an ECG dataset. Analogous to TimeGAN, it's latent dimension is equal to the sequence length.

### 4.3 Evaluation by Average ELBO

First, we evaluate the average Evidence Lower Bound (ELBO) on a synthetic dataset $\tilde{X} \in \mathbb{R}^{n_s \times l \times c}$ where $n_s$ represents the numbers of samples, $l$ denotes the sequence length, and $c$ the number of channels. We refer to this metric as $\mathcal{E}_{\text{avg}}(\tilde{X})$. In detail, we first train a VAE model on shorter sequence lengths $\ell \ll l$, which facilitates easier training. We denote it as the *ELBO model* $\tilde{\mathcal{L}}_{\theta,\phi} : \mathbb{R}^{\ell \times c} \to \mathbb{R}$.

We then calculate the *average ELBO*:

$$\mathcal{E}_{\text{avg}}(\tilde{X}) = \frac{1}{n_s(l-\ell)} \sum_{i=0}^{n_s-1} \sum_{t=0}^{l-\ell-1} \text{ELBO}_{\text{norm}} \left( \tilde{\mathcal{L}}_{\theta,\phi}(\tilde{X}_{i,t:t+\ell,\cdot}) \right), \tag{1}$$

where $\text{ELBO}_{\text{norm}} = \text{ELBO} \cdot (ct)^{-1}$ is a normalized ELBO, as explained in Appendix A.2. By normalizing the ELBO, we get a fairer comparison of datasets with different dimensionalities and varying sequence lengths.

$\mathcal{E}_{\text{avg}}(\tilde{X})$ gives us information about short term consistency over the whole synthetic dataset. We chose $\ell = 50$ which is half of the lowest sequence length in the experiments. A well trained *ELBO model* (An & Cho, 2015) allows us to evaluate the (relative) short term consistency of synthetic data in high accuracy and low variance. To ensure reliable assessment of sample quality, we prevented overfitting of the *ELBO model* by applying early stopping after 50 epochs without improvement and restoring the best weights. In our experiments, we employed two distinct *ELBO models* for calculating $\mathcal{E}_{\text{avg}}(\tilde{X})$. The first model is based on the RVAE-ST architecture, while the second utilizes the TimeVAE framework (Desai et al., 2021a). The use of a TimeVAE-based *ELBO model* provides an additional evaluation to ensure that the RVAE-ST-based model is not biased toward our own generated samples. As detailed in Appendix A.5, the results obtained using TimeVAE are highly similar to those produced by the RVAE-ST-based model.

As shown in Table 1, $\mathcal{E}_{\text{avg}}(\tilde{X})$ indicates that RVAE-ST consistently generates the best samples across all datasets and sequence lengths. $\mathcal{E}_{\text{avg}}(\tilde{X})$ remains consistent not only across sequence lengths but also between datasets. WaveGAN behaves similarly, but the sample quality is consistently lower, especially on the ECG dataset. TimeGAN and TimeVAE exhibit less consistent performance, with significant drops occurring at different sequence lengths depending on the dataset. RCGAN consistently underperformed compared to all other models and faced substantial stability issues during training, which may have contributed to its poor overall performance. For all scores, the Wilcoxon rank test(Wilcoxon, 1992) indicates statistical significance with $p < 0.0002$, except for the ETT dataset at $l = 100$, where TimeVAE performs significantly better. We suspect that this is due to TimeVAE containing some poor outliers in its samples, which are not reflected in the ranking.

### 4.4 Evaluations by Discriminative Score

The discriminative score $\mathcal{D}$ was introduced by (Yoon et al., 2019) as a metric for quality evaluation of synthetic time series data. For the discriminative score a simple 2-layer RNN for binary classification is

Table 1: Average *ELBO* score $\mathcal{E}_{\text{avg}}(\tilde{X})$ of synthetic time series for five models (see 4.2), computed on the three datasets (see 4.1) at sequence lengths of $l = 100$, $l = 300$, $l = 500$, and $l = 1000$. A higher score indicates better performance. For each score, 1500 generated samples were evaluated by an *ELBO model* (based on the RVAE-ST architecture) and the results are reported with 1-sigma confidence intervals. Our model consistently achieves the best performance across all sequence lengths and all datasets.

| | | **Sequence lengths** | | | |
|---|---|---|---|---|---|
| **Dataset** | **Model** | **100** | **300** | **500** | **1000** |
| Electric Motor | **RVAE-ST(ours)** | **1.62±0.69** | **1.65±0.60** | **1.66±0.03** | **1.65±0.03** |
| | TimeGAN | 1.20±0.59 | 1.33±0.48 | 1.13±0.56 | -4.05±2.41 |
| | WaveGAN | 1.54±0.11 | 1.54±0.16 | 1.54±0.14 | 1.53±0.37 |
| | TimeVAE | 1.49±0.88 | 1.38±1.34 | 1.09±2.21 | 0.31±3.24 |
| | RCGAN | 0.33±0.56 | -61.0±10.4 | -26.1±3.94 | -66.0±1.26 |
| ECG | **RVAE-ST(ours)** | **1.64±0.13** | **1.64±0.18** | **1.63±0.20** | **1.59±0.27** |
| | TimeGAN | -14.6±1.87 | -14.6±1.41 | -13.7±6.67 | -15.3±2.57 |
| | WaveGAN | 1.12±0.81 | 1.11±0.87 | 1.10±0.86 | 1.10±0.83 |
| | TimeVAE | 1.55±0.37 | 1.37±0.65 | 1.26±0.70 | 0.87±0.92 |
| | RCGAN | -6.20±18.0 | -0.16±0.80 | 0.29±1.46 | -69.4±5.78 |
| ETT | **RVAE-ST(ours)** | **1.49±0.52** | **1.50±0.40** | **1.52±0.35** | **1.53±0.63** |
| | TimeGAN | 1.39±0.70 | 0.85±3.36 | -4.29±9.66 | -0.38±0.65 |
| | WaveGAN | 1.40±0.53 | 1.39±0.70 | 1.42±0.51 | 1.42±0.48 |
| | TimeVAE | 1.47±0.94 | 1.20±1.54 | 0.89±1.99 | 0.42±2.45 |
| | RCGAN | -10.4±24.7 | -0.59±1.84 | -108±0.36 | -41.7±2.39 |

trained to distinguish between original and synthetic data. Implementation details are in the appendix A.4. It is defined as $\mathcal{D} = |0.5 - a|$, where $a$ represents the classification accuracy between the original test dataset and the synthetic test dataset that were not used during training. The best possible score of 0 means that the classification network cannot distinguish original from synthetic data, whereas the worst score of 0.5 means that the network can easily do so.

The discriminative score provides particularly meaningful insights when it allows for clear distinctions between models, which is best achieved by avoiding scenarios where the score consistently reaches its best or worst possible values across different models. To ensure consistency, we used the same fixed number of samples for training the discriminator across all experiments, regardless of sequence length. This fixed sample size was found to be suitable for our experimental setup.

As shown in table 2, RVAE-ST consistently outperforms the other models on the EM and ECG datasets across all sequence lengths, demonstrating its ability to generate realistic time series, particularly on relatively stationary datasets. Statistical significance was assessed using Wilcoxon rank tests(Wilcoxon, 1992). For the EM and ECG datasets, most results were found to be statistically significant with $p \leq 0.02$, except for the ECG dataset at $l = 100$, where $p = 0.22$ when compared to TimeVAE. In contrast, on the ETT dataset, RVAE-ST does not outperform the comparison models. Specifically, TimeGAN achieves better performance at a sequence length of $l = 100$, while TimeVAE outperforms at sequence lengths of $l = 300$, $l = 500$, and $l = 1000$. These differences are statistically significant.

### 4.5 Evaluation by PCA and t-SNE

In this section, we evaluate the quality of the generated time series using dimensionality reduction techniques such as PCA(Hotelling, 1933) and t-SNE(Hinton & Van Der Maaten, 2008). The idea is to first train these methods on the original data, project the data into a lower-dimensional space, and visualize the resulting

Table 2: Discriminative score of synthetic time series for five models (see 4.2), computed on the three datasets (see 4.1) at sequence lengths of $l = 100$, $l = 300$, $l = 500$, and $l = 1000$. A lower discriminative score indicates better performance. For each score, 15 experiments were conducted by training the discriminator 15 times independently, and the results are reported with 1-sigma confidence intervals. Our model consistently achieves the best performance across all sequence lengths for the EM and ECG datasets. However, for the ETT dataset, TimeGAN performs best at a sequence length of 100, while TimeVAE outperforms other models at sequence lengths of 300, 500, and 1000.

| | | **Sequence lengths** | | | |
|---|---|---|---|---|---|
| **Dataset** | **Model** | **100** | **300** | **500** | **1000** |
| Electric Motor (EM) | **RVAE-ST (ours)** | **.121±.021** | **.032±.018** | **.038±.018** | **.085±.015** |
| | TimeGAN | .338±.030 | .477±.018 | .486±.013 | .500±.000 |
| | WaveGAN | .352±.009 | .416±.009 | .425±.011 | .444±.011 |
| | TimeVAE | .268±.214 | .226±.176 | .185±.083 | .152±.047 |
| | RCGAN | .499±.001 | .500±000 | .500±.000 | .500±000 |
| ECG | **RVAE-ST (ours)** | **.012±.011** | **.009±.008** | **.016±.014** | **.009±.010** |
| | TimeGAN | .466±.125 | .500±000 | .500±.000 | .500±000 |
| | WaveGAN | .306±.155 | .300±.201 | .402±.153 | .298±.217 |
| | TimeVAE | .034±.066 | .058±.120 | .131±.181 | .153±.177 |
| | RCGAN | .375±.131 | .467±.031 | .473±.012 | .500±000 |
| ETT | RVAE-ST (ours) | .179±.034 | .172±.105 | .189±.049 | .132±.147 |
| | TimeGAN | **.107±.075** | .160±.113 | .270±.106 | .320±.120 |
| | WaveGAN | .362±.080 | .345±.113 | .377±.099 | .385±.060 |
| | **TimeVAE** | .118±.110 | **.140±.053** | **.167±.040** | **.068±.051** |
| | RCGAN | .500±000 | .500±.001 | .500±000 | .499±.004 |

patterns. Subsequently, the same transformations are applied to the synthetic data to assess how well they align with the distribution of the original data.

These common techniques complement earlier methods that primarily assessed the sample quality of the models. For brevity, we present the results of four selected experiments in the main paper, as all experiments consistently yield the same findings. These four experiments include PCA plots on the EM dataset and t-SNE plots on the ETT dataset, each with sequence lengths of $l = 100$ and $l = 1000$. The full set of experiments is provided in Appendix A.7.

The visual inspection of the PCA plots for the EM dataset with a sequence length of $l = 100$ reveals no significant differences in the distributions of the models, with the exception of worse performance in RCGAN. However, as the sequence length increases to $l = 1000$, the performance differences between the models become clearly visible. Interestingly, the PCA at this length exhibits a circular pattern, indicating the periodic characteristics of the dataset. Among the models, RVAE-ST demonstrates the highest degree of overlap between the original and synthetic data, fitting the circular pattern without outliers. WaveGAN shows only a few outliers near the circular pattern. TimeVAE synthetic points further fill the circle, leading to greater deviation from the original data distribution. TimeGAN shows minimal overlap with the original points, yet the synthetic data still conforms to a circular arrangement. RCGAN encounters significant difficulties at a sequence length of 1000, with the training process affected by numerical stability issues.

The t-SNE plots for l=100 on the ETT dataset indicate that RVAE-ST and TimeVAE exhibit the most overlap between real and synthetic data, while the GAN-based models show significantly less overlap. For l=1000, RVAE-ST maintains the highest overlap, with no visible difference compared to the l=100 plot. TimeGAN experienced a significant performance drop, whereas WaveGAN as expected, showed no visual

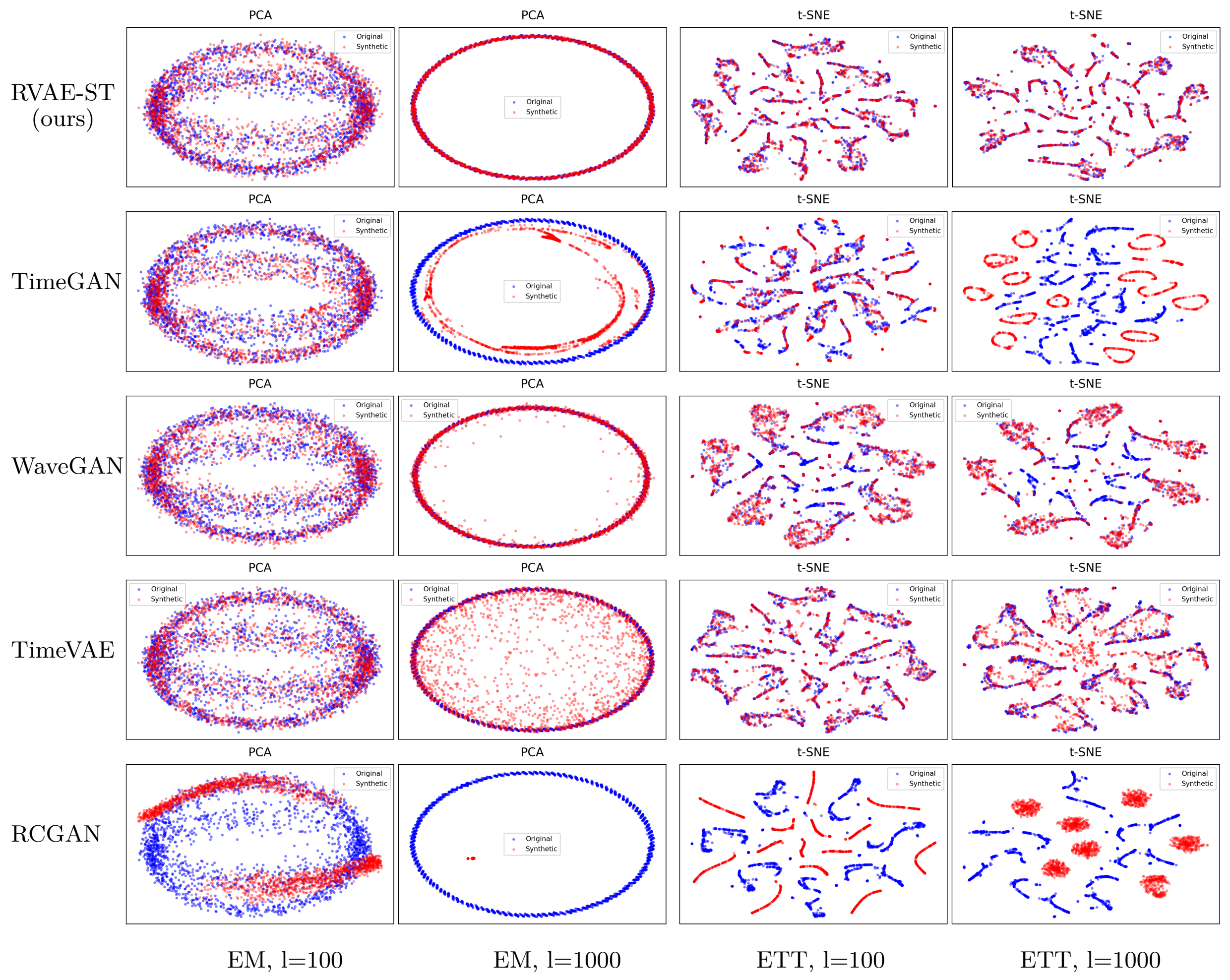


Figure 3: PCA and t-SNE plots comparing original (blue) and synthetic (red) samples across five models (see 4.2). The results are computed on the EM and ETT datasets (see 4.1) at sequence lengths of $l = 100$ and $l = 1000$. For the EM dataset (first and second columns), the PCA plots at $l = 100$ show no significant differences among the models, except for RCGAN, which generally performs weaker on all datasets. At $l = 1000$, RVAE-ST (our model) achieves the best results, demonstrating the highest alignment with the original distribution and no outliers. For the ETT dataset (third and fourth columns), the t-SNE plots indicate that VAE-based models generally perform better, with no visible difference between RVAE-ST and TimeGAN at $l = 100$. However, at $l = 1000$, RVAE-ST provides the most accurate representation of the original distribution.

difference. TimeVAE on the other hand, displayed more outliers for l=1000 than for l=100, but still followed the data distribution pattern.

### 4.6 Training scheme ablations

In this experiment, we compare the effectiveness of our proposed training approach against the conventional training method on the same network topology. Our comparison metric is the Evidence Lower Bound (ELBO), calculated for the original dataset $X \in \mathbb{R}^{n_s \times l \times c}$ where $n_s$ represents the numbers of samples, $l$ denotes the sequence length, and $c$ the number of channels. It is analogous to (1), but we get

$$\mathcal{E}(X) = \frac{1}{n_s} \sum_{i=0}^{n_s - 1} \mathrm{ELBO}_{\mathrm{norm}} \left( \tilde{\mathcal{L}}_{\theta,\phi}(X_i) \right), \tag{2}$$

since we use the trained model $\tilde{\mathcal{L}}_{\theta,\phi}$ itself for evaluation. Simply speaking, it is the typical model evaluation on a dataset, but converted to $\mathrm{ELBO}_{\mathrm{norm}}$. We run this comparison on all datasets with a sequence length of 1000, which is particularly long and challenging. It is the maximum sequence length used in any of the previous experiments. For each of the following training schemes, we do 10 repetitions:

(i) **Conventional train**: One trains the model for a predefined sequence length of $l = 1000$

(ii) **Subsequent train**: The training procedure begins with a sequence length of $l = 100$ and continues until the stopping criteria are met. Afterward, we increase the sequence length by 100 and retrain the model, repeating this process until we complete training with a sequence length of $l = 1000$.

Table 3: Comparison of the effectiveness of our proposed training approach versus the conventional method. The performance metric is the $\mathrm{ELBO}_{\mathrm{norm}}$ as described in the Appendix A.2. On each dataset and model we repeated the experiments $n = 10$ times. The 1-sigma confidence intervals describe the results between the independently trained models.

| **Train methods** | **EM** | **ECG** | **ETTm2** |
|---|---|---|---|
| conventional train | 0.094±0.004 | 0.103±0.000 | 0.174±0.016 |
| subsequent train | **0.218±0.004** | **0.201±0.004** | **0.217±0.012** |

As shown in Table 3, training scheme (ii) results in a significant improvement over training scheme (i) across all datasets. The Wilcoxon rank-sum test confirms these differences as statistically significant, with $p < 0.002$ on all three datasets. The performance gain was comparatively lower on the ETT dataset, which aligns with expectations, as this dataset is the least stationary among the three.

For the EM dataset, a comparison between the synthetic samples generated by a conventionally trained model and a subsequently trained model can be observed in Figure 1. The conventionally trained model has only learned the high-frequency component of the channel, with significant smoothing applied, while the subsequently trained model closely resembles the original dataset. Notably, the same applies to the ECG dataset where new training schedule enabled the model to learn the data correctly in the first place, resulting in the difference between a flat line and a proper ECG signal.

In summary, the RVAE-ST-model initially faced challenges when training on long time series, as expected, the adjusted training approach proposed in this paper effectively enabled the model to learn from extended sequences.

## 5 Discussion

In our experiments, we compared our proposed model with other state-of-the-art models for time series data synthesis. The focus was on generating realistic sensor data, i.e. long multivariate time series in the context of approximately stationary datasets or datasets with periodic characteristics.

We demonstrated that our model outperforms others in terms of sample quality as well as the distribution of synthetic datasets. The longer the sequence length, the better our model performed in comparison. To achieve this, we adopted a recurrent network topology with an approximately equivariant structure, where the number of trainable parameters is independent of the sequence length. This allowed us to repeat the training process with increasing sequence lengths. The performance gains from this method are particularly pronounced for datasets with strong periodicity. We further hypothesize that our approach could also improve the performance of recurrence-based GAN models.

Our model represents a fully generic approach that does not require any contextual information. Moreover, we emphasize that no individual hyperparameter tuning was performed for any of the datasets, ensuring a fair comparison. The experiments revealed that RNN-based approaches in other models (TimeGAN, RCGAN) underperformed compared to CNN-based models (WaveGAN, TimeVAE). Notably, RCGAN was unable to learn the datasets even at the shortest sequence length, $l = 100$. Surprisingly, on the ETT dataset, TimeVAE outperformed our model in the discriminative score, despite performing worse on the *average ELBO score* and the PCA/t-SNE plots.

Additionally, it was found that the *average ELBO score* is significantly more meaningful and less variable than the discriminative score. This becomes particularly apparent in the case of the RCGAN model. While the discriminative score, hovering around 0.5 in the experiments, provides little insight, the high negative values of the ELBO score are far more meaningful. This is further supported by the PCA and t-SNE plots.

The pca and t-SNE plots show that the synthetic datasets generated by VAE-based models better reflect the data distribution, even though the sample quality may be lower. This is unsurprising, as VAEs are explicitly designed to learn the data distribution. As a result, a much stronger correlation can be observed between the *ELBO score* and the PCA/t-SNE plots, compared to the discriminative score.

Furthermore, we point out that the increments between sequence lengths in our experiments were somewhat arbitrarily chosen. It is possible that this methodology could yield favorable results with a different scaling.

## References

Jinwon An and Sungzoon Cho. Variational autoencoder based anomaly detection using reconstruction probability. *Special lecture on IE*, 2(1):1–18, 2015.

Dzmitry Bahdanau. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

Christopher P Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. Understanding disentangling in $\beta$-vae. *arXiv preprint arXiv:1804.03599*, 2018.

Abhyuday Desai, Cynthia Freeman, Zuhui Wang, and Ian Beaver. Timevae: A variational auto-encoder for multivariate time series generation. *arXiv preprint arXiv:2111.08095*, 2021a.

Abhyuday Desai, Cynthia Freeman, Zuhui Wang, and Ian Beaver. Timevae: A variational auto-encoder for multivariate time series generation. `https://github.com/abudesai/timeVAE`, 2021b.

Chris Donahue, Julian McAuley, and Miller Puckette. Adversarial audio synthesis, 2019.

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

A. Goldberger, L. Amaral, L. Glass, J. Hausdorff, P. C. Ivanov, R. Mark, and H. E. Stanley. Physiobank, physiotoolkit, and physionet: Components of a new research resource for complex physiologic signals. *Circulation*, 101(23):e215–e220, 2000. doi: 10.1161/01.CIR.101.23.e215. Online.

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

Tom Hammerbacher, Markus Lange-Hegermann, and Gorden Platz. Including sparse production knowledge into variational autoencoders to increase anomaly detection reliability. In *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, pp. 1262–1267. IEEE, 2021.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-VAE: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2017.

G Hinton and L Van Der Maaten. Visualizing data using t-sne journal of machine learning research. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997. doi: 10.1162/neco.1997.9.8.1735.

Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417, 1933.

Diederik Kingma and Max Welling. Auto-encoding variational bayes. 12 2014.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Yonghong Luo, Xiangrui Cai, Ying Zhang, Jun Xu, et al. Multivariate time series imputation with generative adversarial networks. *Advances in neural information processing systems*, 31, 2018.

Philipp N. Mueller. Attention-enhanced conditional-diffusion-based data synthesis for data augmentation in machine fault diagnosis. *Engineering Applications of Artificial Intelligence*, 131:107696, 2024. doi: https://doi.org/10.1016/j.engappai.2023.107696.

Kevin P. Murphy. *Probabilistic Machine Learning: An introduction*. MIT Press, 2022. URL `probml.ai`.

Nour Neifar, Achraf Ben-Hamadou, Afef Mdhaffar, and Mohamed Jmaiel. Diffecg: A versatile probabilistic diffusion model for ecg signals synthesis. *arXiv preprint arXiv:2306.01875*, 2023.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Sima Siami-Namini, Neda Tavakoli, and Akbar Siami Namin. The performance of lstm and bilstm in forecasting time series. In *2019 IEEE International conference on big data (Big Data)*, pp. 3285–3292. IEEE, 2019.

Yusuke Tashiro, Jiaming Song, Yang Song, and Stefano Ermon. Csdi: Conditional score-based diffusion models for probabilistic time series imputation. *Advances in Neural Information Processing Systems*, 34: 24804–24816, 2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

Frank Wilcoxon. Individual comparisons by ranking methods. In *Breakthroughs in statistics: Methodology and distribution*, pp. 196–202. Springer, 1992.

P. Wißbrock and P. N. Müller. Lenze motor bearing fault dataset (lenze-mb), 2025. URL `https://doi.org/10.5281/zenodo.14762423`.

Yiyuan Yang, Ming Jin, Haomin Wen, Chaoli Zhang, Yuxuan Liang, Lintao Ma, Yi Wang, Chenghao Liu, Bin Yang, Zenglin Xu, et al. A survey on diffusion models for time series and spatio-temporal data. *arXiv preprint arXiv:2404.18886*, 2024.

Jinsung Yoon, Daniel Jarrett, and Mihaela Van der Schaar. Time-series generative adversarial networks. *Advances in neural information processing systems*, 32, 2019.

Yitian Zhang, Liheng Ma, Soumyasundar Pal, Yingxue Zhang, and Mark Coates. Multi-resolution time-series transformer for long-term forecasting. In *International Conference on Artificial Intelligence and Statistics*, pp. 4222–4230. PMLR, 2024.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *The Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Virtual Conference*, volume 35, pp. 11106–11115. AAAI Press, 2021a.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 11106–11115, 2021b.

Yuzhen Zhu, Shaojie Luo, Di Huang, Weiyan Zheng, Fang Su, and Beiping Hou. Drcnn: decomposing residual convolutional neural networks for time series forecasting. *Scientific Reports*, 13(1):15901, 2023.

## A Appendix

### A.1 Hyperparameters and Loss Function

In all experiments, for the encoder aswell as the decoder, we stack 4 LSTM-layers each with 256 hidden units. The latent dimension is 20. We use Adam optimizer with learning rate $\alpha = 10^{-4}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-7}$. We perform min-max scaling with $(-1, 1)$. After scaling we do a train/validation split with a ratio of 9:1. The creation of chunks is done with a step size of $\frac{1}{10} \cdot T$ for the EM and ECG dataset and with $\frac{1}{25} \cdot T$ for the ETTm2 dataset, where $T$ is the sequence length.

We use the loss function

$$\mathcal{L}_{\theta,\phi} = \alpha \cdot \mathrm{SSE} + \beta \cdot \mathrm{D_{KL}}, \tag{3}$$

where the reconstruction loss, SSE, represents the sum of squared errors, computed for each individual sample within a batch:

$$\mathrm{SSE} = \sum_T \sum_C (y_{tc} - \hat{y}_{tc})^2, \tag{4}$$

where $T$ is the sequence length and $C$ is the number of channels. We then average the SSE over the entire batch. In our experiments we set $\alpha = \frac{500}{\mathrm{T}}$ and $\beta = 0.1$.

The parameter $\beta$ was introduced with the $\beta$-VAE (Higgins et al., 2017). For $0 < \beta < 1$ the VAE stores more bits about each input and the reconstructed sample is less smoothed out. If $\beta > 1$ the VAE is encouraged to learn a latent representation that is disentangled (Burgess et al., 2018). We adjust $\alpha$ antiproportional to the sequence length to retain the ratio between the reconstruction loss and the KL-Divergence.

### A.2 Loss to ELBO conversion

Transforming the VAE loss function into the Evidence Lower Bound (ELBO) is essential to connect the optimization process to a well-established probabilistic framework. The ELBO arises from the variational inference approach, which allows us to approximate the intractable posterior distribution of latent variables by optimizing a lower bound to the marginal likelihood of the observed data. By expressing the VAE loss as the ELBO, we clarify that the model's objective is twofold: maximizing the likelihood of the data through reconstruction and simultaneously regularizing the latent space by minimizing the divergence between the approximate posterior and the prior distribution. This dual objective ensures that the learned latent space reflects meaningful, structured representations while maintaining the ability to reconstruct the input data. Using the ELBO as the loss function thus ties the VAE training to a coherent probabilistic theory, enhancing both its interpretability and its ability to generate diverse and realistic data.

Given the likelihood,

$$\text{likelihood} = \prod_T \prod_C \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\frac{(y_{tc} - \hat{y}_{tc})^2}{\sigma^2}\right), \tag{5}$$

we compute the log-likelihood, which can then be reformulated in terms of the SSE:

$$\begin{aligned}
\text{log-likelihood} &= \log\left(\prod_T \prod_C \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\frac{(y_{tc} - \hat{y}_{tc})^2}{\sigma^2}\right)\right) \\
&= \sum_T \sum_C \log\left(\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\frac{(y_{tc} - \hat{y}_{tc})^2}{\sigma^2}\right)\right) \\
&= \sum_T \sum_C \left(\log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) + \log\left(\exp\left(-\frac{1}{2}\frac{(y_{tc} - \hat{y}_{tc})^2}{\sigma^2}\right)\right)\right) \\
&= \sum_T \sum_C \left(\log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) - \frac{1}{2}\frac{(y_{tc} - \hat{y}_{tc})^2}{\sigma^2}\right) \\
&= \sum_T \sum_C \left(-\frac{1}{2}\log(2\pi\sigma^2) - \frac{1}{2}\frac{(y_{tc} - \hat{y}_{tc})^2}{\sigma^2}\right) \\
&= -\frac{1}{2}\log\left(2\pi\sigma^2\right) \cdot T \cdot C - \frac{1}{2\sigma^2}\sum_T \sum_C \left(y_{tc} - \hat{y}_{tc}\right)^2 \\
&= -\frac{1}{2}\log\left(2\pi\sigma^2\right) \cdot T \cdot C - \frac{1}{2\sigma^2}\text{SSE} \\
\iff -\frac{1}{2\sigma^2}\text{SSE} &= \text{log-likelihood} + \frac{1}{2}\log\left(2\pi\sigma^2\right) \cdot T \cdot C \\
\iff \text{SSE} &= -2\sigma^2 \cdot \text{log-likelihood} - \sigma^2 \log\left(2\pi\sigma^2\right) \cdot T \cdot C.
\end{aligned} \tag{6}$$

The ELBO is defined as the log-likelihood minus the kl-divergence (Murphy, 2022):

$$\text{ELBO} = \text{log-likelihood} - \text{D}_{\text{KL}}. \tag{7}$$

Given (3), (6) and $\sigma^2 = 0.5 \cdot \frac{\beta}{\alpha}$, we can derive the conversion to the ELBO:

$$\begin{aligned}
\frac{\mathcal{L}_{\theta,\phi}}{\beta} &= \frac{\alpha}{\beta} \cdot \text{SSE} + \text{D}_{\text{KL}} \\
&= \frac{\alpha}{\beta}\left(-2\sigma^2 \cdot \text{log-likelihood} - \sigma^2 \cdot \log\left(2\pi\sigma^2\right) \cdot T \cdot C\right) + \text{D}_{\text{KL}} \\
&= -2\sigma^2 \cdot \frac{\alpha}{\beta} \cdot \text{log-likelihood} - \sigma^2 \cdot \frac{\alpha}{\beta} \cdot \log\left(2\pi\sigma^2\right) \cdot T \cdot C + \text{D}_{\text{KL}} \\
&= -2 \cdot 0.5 \cdot \frac{\beta}{\alpha} \cdot \frac{\alpha}{\beta} \cdot \text{log-likelihood} - 0.5 \cdot \frac{\beta}{\alpha} \cdot \frac{\alpha}{\beta} \cdot \log\left(2\pi \cdot 0.5 \cdot \frac{\beta}{\alpha}\right) \cdot T \cdot C + \text{D}_{\text{KL}} \\
&= -\text{log-likelihood} - 0.5 \cdot \log\left(\pi \cdot \frac{\beta}{\alpha}\right) \cdot T \cdot C + \text{D}_{\text{KL}} \\
\iff \text{log-likelihood} - \text{D}_{\text{KL}} &= -\frac{\mathcal{L}_{\theta,\phi}}{\beta} - 0.5 \cdot \log\left(\pi \cdot \frac{\beta}{\alpha}\right) \cdot T \cdot C \\
\implies \text{ELBO}(\mathcal{L}_{\theta,\phi}, \alpha, \beta, T, C) &= -\frac{\mathcal{L}_{\theta,\phi}}{\beta} - 0.5 \cdot \log\left(\pi \cdot \frac{\beta}{\alpha}\right) \cdot T \cdot C.
\end{aligned} \tag{8}$$

In our experiments, we normalize the ELBO by dividing it by the product of the number of channels and the sequence length. This normalization allows for a fairer comparison of model performance across datasets with different dimensionalities, such as varying sequence lengths or numbers of channels. Without this adjustment, the ELBO would scale with the size of the data, potentially biasing the evaluation in favor of datasets with larger sequences or more channels. By normalizing, we make the ELBO more independent of the specific data structure, enabling a more consistent comparison of the underlying model's ability to capture data patterns.

Although this normalization provides a useful heuristic for comparing different datasets, it should be noted that it does not guarantee perfect comparability in all cases. In some situations, larger datasets with more channels or longer sequences may introduce additional complexity, which could influence the model's performance. Therefore, while the normalized ELBO serves as a practical and interpretable metric.

We denote the normalized version of the ELBO as

$$\mathrm{ELBO}_{\mathrm{norm}}(\mathcal{L}_{\theta,\phi}, \alpha, \beta, T, C) = \frac{\mathrm{ELBO}(\mathcal{L}_{\theta,\phi}, \alpha, \beta, T, C)}{T \cdot C}. \tag{9}$$

### A.3 Implementation details of comparison models

#### A.3.1 TimeGAN

We did all experiments with the same hyperparameters. Num layers=3, hidden dim=100, num iterations = 25000. The clockwise computation time on these hyperparameters were the highest of all models. We use the authors original implementation[3] on a Nvidia DGX server in the 19.12-tf1-py3 container[4]. On sequence length $l = 1000$ , the training took about 3 weeks wall-clock time.

#### A.3.2 WaveGAN

For WaveGan needed special preperation to be usable for training. First we min maxed scaled the dataset file, split it into training and validation parts and then converted each into a n-dimensional *.wav* file. WaveGan is limited in configurability. In terms of sequence length the user can decide between $2^{14}$, $2^{15}$ and $2^{16}$. We chose $2^{14} = 16384$ because it is the smallest possible length. When we generate samples, we cut them into equal parts which correspond to the desired sequence length $l$. The rest of the hyperparameters were set to default. We used the ported pytorch implementation[5].

#### A.3.3 TimeVAE

We use TimeVAE with default parameters. We integrated components of the original TimeVAE implementation[6], such as the encoder, decoder, and loss function, into our own program framework. The reconstruction loss of TimeVAE is

$$\sum_{T}\sum_{C}(y_{tc} - \hat{y}_{tc})^2 + \frac{1}{C}\sum_{C}(\bar{y}_c - \bar{\hat{y}}_c)^2. \tag{10}$$

TimeVAEincludes a hyperparameter a, which acts as a weighting factor for the reconstruction loss. The authors of the original paper recommend using a value for a in the range of 0.5 to 3.5 to balance the trade-off between reconstruction accuracy and latent space regularization. In all of our experiments, we set $a = 3$.

#### A.3.4 RCGAN

We used the original implementation[7] on a Nvidia DGX server in the 19.12-tf1-py3 container[8]. We trained RCGANfor 500 epochs and afterswards used the weights with the lowest $\hat{t}$ for sampling the dataset. On the ETT dataset, training was numerically instable for $l \geq 300$.

[3]https://github.com/jsyoon0823/TimeGAN
[4]https://docs.nvidia.com/deeplearning/frameworks/tensorflow-release-notes/rel_19.12.html
[5]https://github.com/mostafaelaraby/wavegan-pytorch
[6]https://github.com/abudesai/timeVAE
[7]https://github.com/ratschlab/RGAN
[8]https://docs.nvidia.com/deeplearning/frameworks/tensorflow-release-notes/rel_19.12.html

### A.4 Discriminative Score

The 2-layer RNN for binary classification consists of a GRU layer, where the hidden dimension is set to $\lfloor n_c/2 \rfloor$, where $n_c$ is the number of channels. This is followed by a linear layer with an output dimension of one. To prevent overfitting, early stopping with a patience of 50 is applied. We each discriminative score we repeated 15 training procedures. On each procedure, 2000 random samples were used as the train dataset and 500 samples were used as the validation dataset for early stopping monitoring. The discriminative score is then determined by validating further independent 500 samples.

### A.5 Average Elbo with TimeVAE Elbo-Model

Table 4 shows the results for the average *ELBO* score $\mathcal{E}(\tilde{X})$ using the base of TimeVAE as the *ELBO model*. However, instead of using the original loss function of TimeVAE, we utilized the loss function of RVAE-ST as it simplifies the conversion to the *ELBO* score as shown in (8). Analogous to Table 1, our model is outperforming all other models. The Wilcoxon rank test indicates statistical significance with p $<$0.0001, except for the ETT dataset at $l = 100$ and $l = 300$, where TimeVAEis statistically better.

Table 4: Average *ELBO* score $\mathcal{E}(\tilde{X})$ of synthetic time series for five models (see 4.2), computed on the three datasets (see 4.1) at sequence lengths of $l = 100$, $l = 300$, $l = 500$, and $l = 1000$. A higher score indicates better performance. For each score, 1500 generated samples were evaluated by an *ELBO model* (based on the TimeVAE architecture) and the results are reported with 1-sigma confidence intervals. Our model consistently achieves the best performance across all sequence lengths and all datasets.

| | | **Sequence lengths** | | | |
|---|---|---|---|---|---|
| **Dataset** | **Model** | **100** | **300** | **500** | **1000** |
| Electric Motor | **RVAE-ST (ours)** | **1.61±0.69** | **1.64±0.12** | **1.64±0.01** | **1.64±0.02** |
| | TimeGAN | 1.29±0.39 | 1.33±0.17 | 1.21±0.10 | -2.14±0.82 |
| | WaveGAN | 1.52±0.14 | 1.47±1.05 | 1.52±0.22 | 1.52±0.15 |
| | TimeVAE | 1.52±0.87 | 1.44±1.28 | 1.01±2.35 | 0.10±3.58 |
| | RCGAN | 0.53±0.36 | -2e9 | -134±49.9 | -89.1±1.28 |
| ECG | **RVAE-ST (ours)** | **1.62±0.07** | **1.62±0.07** | **1.62±0.06** | **1.59±0.06** |
| | TimeGAN | -2.57±0.22 | -2.26±0.22 | -2.67±1.92 | -2.58±0.49 |
| | WaveGAN | 1.32±0.29 | 1.33±0.18 | 1.32±0.16 | 1.32±0.15 |
| | TimeVAE | 1.57±0.15 | 1.46±0.16 | 1.39±0.15 | 1.08±0.28 |
| | RCGAN | -7.12±21.8 | 0.04±0.44 | 0.92±0.22 | -1e21 |
| ETT | **RVAE-ST (ours)** | **1.56±0.24** | **1.57±0.09** | **1.59±0.05** | **1.60±0.13** |
| | TimeGAN | 1.49±0.17 | 1.20±1.49 | 0.83±0.91 | -0.00±0.28 |
| | WaveGAN | 1.50±0.50 | 1.50±0.41 | 1.47±0.64 | 1.49±0.43 |
| | TimeVAE | **1.56±0.45** | 1.41±0.81 | 1.15±1.05 | 0.40±2.06 |
| | RCGAN | 0.50±0.87 | -0.73±2.96 | -39.4±0.01 | -27.0±0.06 |

### A.6 PyTorch vs TensorFlow

The experiments were conducted using a TensorFlow implementation of our model. Additionally, we performed tests with a PyTorch reimplementation (which is not part of this paper). In these tests, we found that the performance in PyTorch was significantly worse compared to the TensorFlow implementation.

Upon investigation, we identified the cause of the performance difference. The weight initialization in both the LSTM and Dense layers differs between TensorFlow and PyTorch. Specifically, TensorFlow uses a uniform distribution for the initialization of both LSTM and Dense weights, while PyTorch employs different initialization methods by default. To align the behavior between both frameworks, we modified the PyTorch implementation to use the same uniform weight initialization for both LSTM and Dense layers as in TensorFlow. After making these adjustments, we were able to achieve consistent results across both frameworks.

## A.7 PCA and t-SNE Results

The following section presents the PCA and t-SNE plots for all experiments, including each dataset, model, and sequence length. Since RCGAN consistently underperforms, and the performance of TimeGAN and WaveGAN remains unchanged across sequence lengths within a given dataset, these points will not be explicitly mentioned in each figure to maintain clarity and readability.

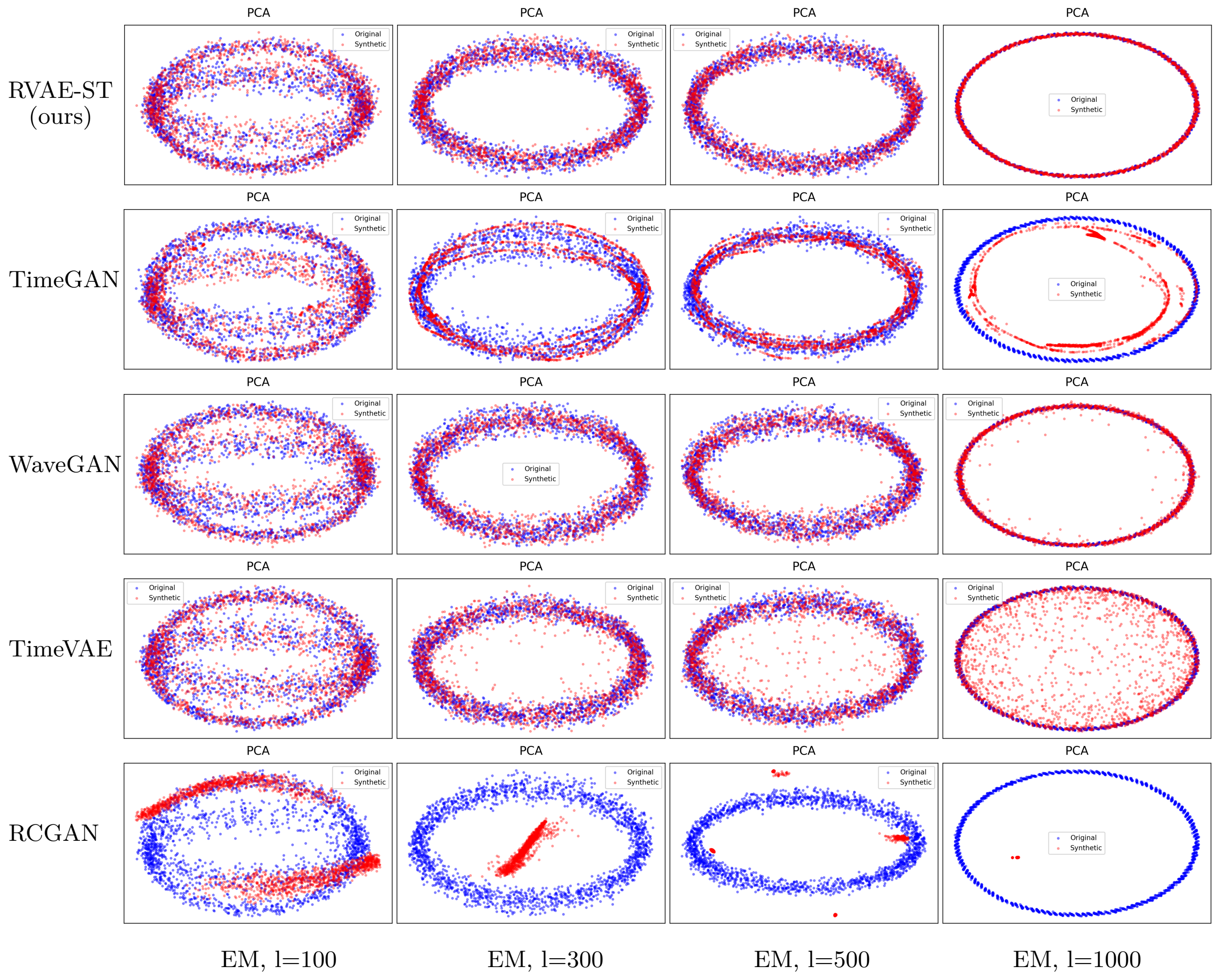


Figure 4: PCA plots for all sequence lengths on the Electric Motor dataset. At $l = 100$, no visible differences between the models are observed. However, at $l = 300$ and $l = 500$, TimeGAN shows a noticeable deterioration in performance. TimeVAE consistently worsens with increasing sequence length. Starting from $l = 300$, RVAE-ST performs visibly the best, followed by WaveGAN with only a slight decrease in performance, while TimeVAE exhibits significant scatter in the data.

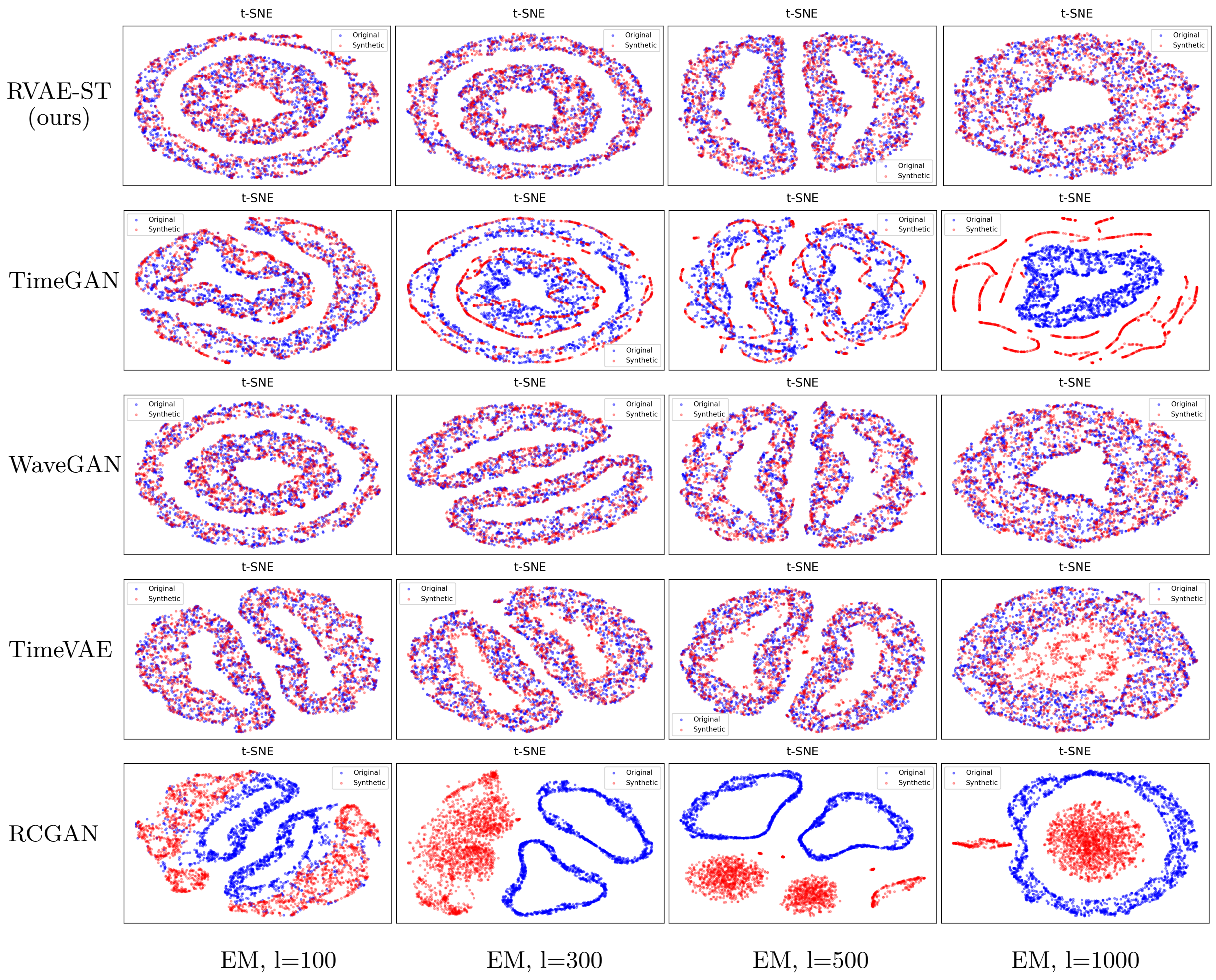


Figure 5: t-SNE plots for all sequence lengths on the Electric Motor dataset. At $l = 100$, no visible differences between the models are observed. However, at $l = 300$ and $l = 500$, TimeGAN shows a noticeable deterioration in performance. TimeVAE consistently worsens with increasing sequence length. Starting from $l = 300$, RVAE-ST performs visibly the best, followed by WaveGAN with only a slight decrease in performance, while TimeVAE exhibits significant scatter in the data.

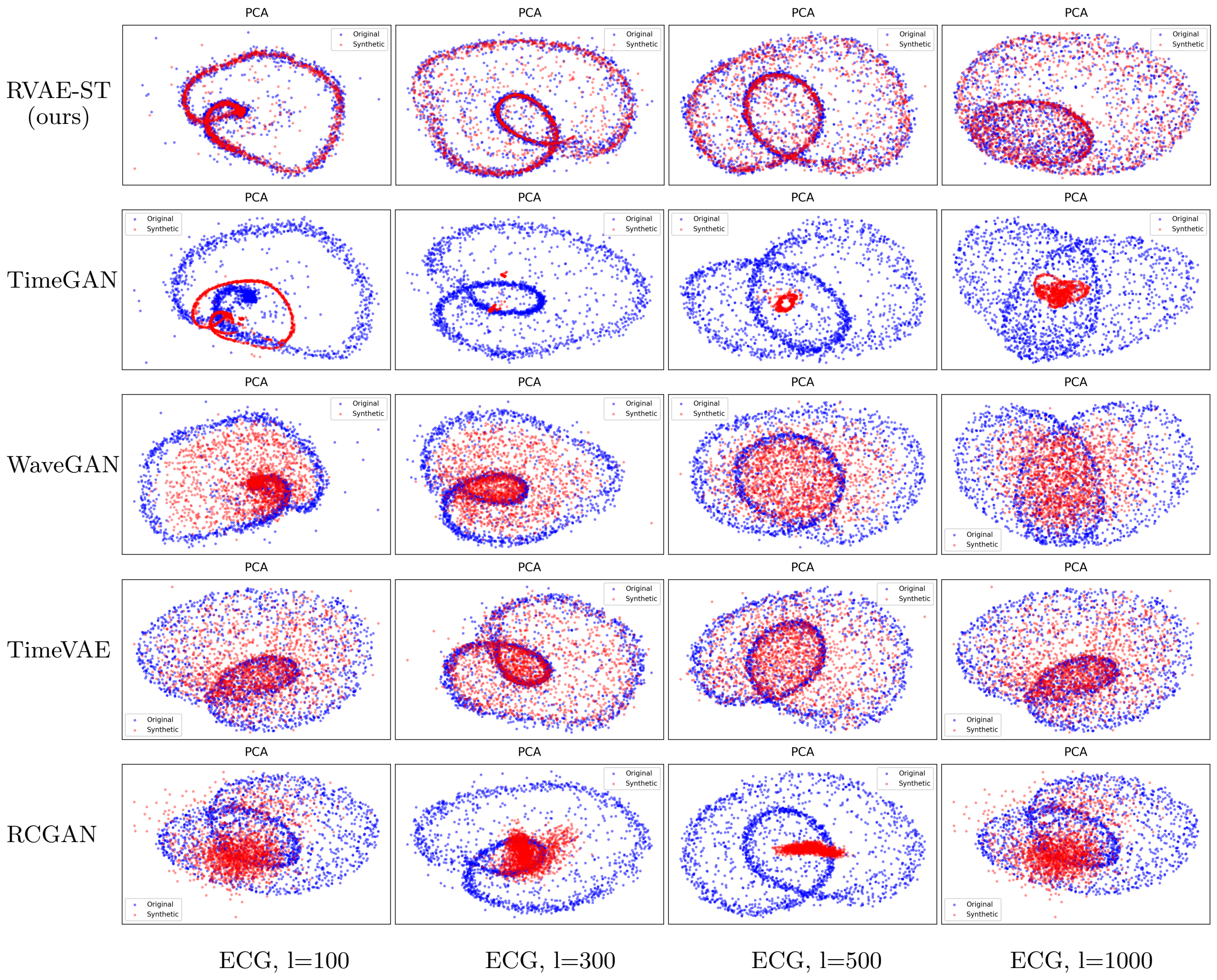


Figure 6: PCA plots for various all lengths on the ECG dataset. TimeGAN was unable to properly learn the dataset. RVAE-ST (our model) performs notably better than WaveGAN and TimeVAE across all sequence lengths.

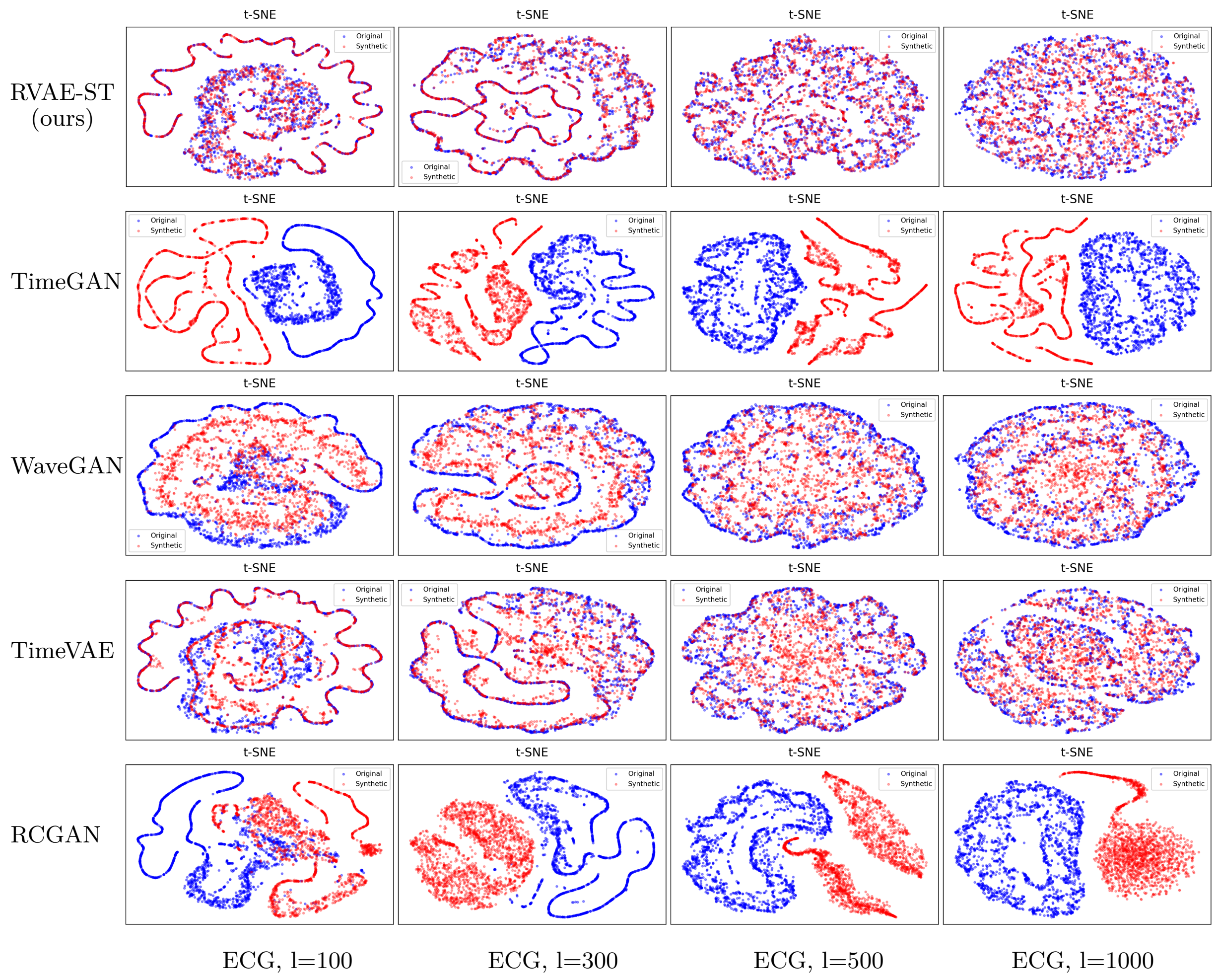


Figure 7: t-SNE plots for various all lengths on the ECG dataset. TimeGAN was unable to properly learn the dataset.RVAE-ST (our model) visibly performs best. TimeVAE notably performs better than WaveGAN at $l = 100$ and $l = 300$, but shows performance drops at $l = 500$ and $l = 1000$.

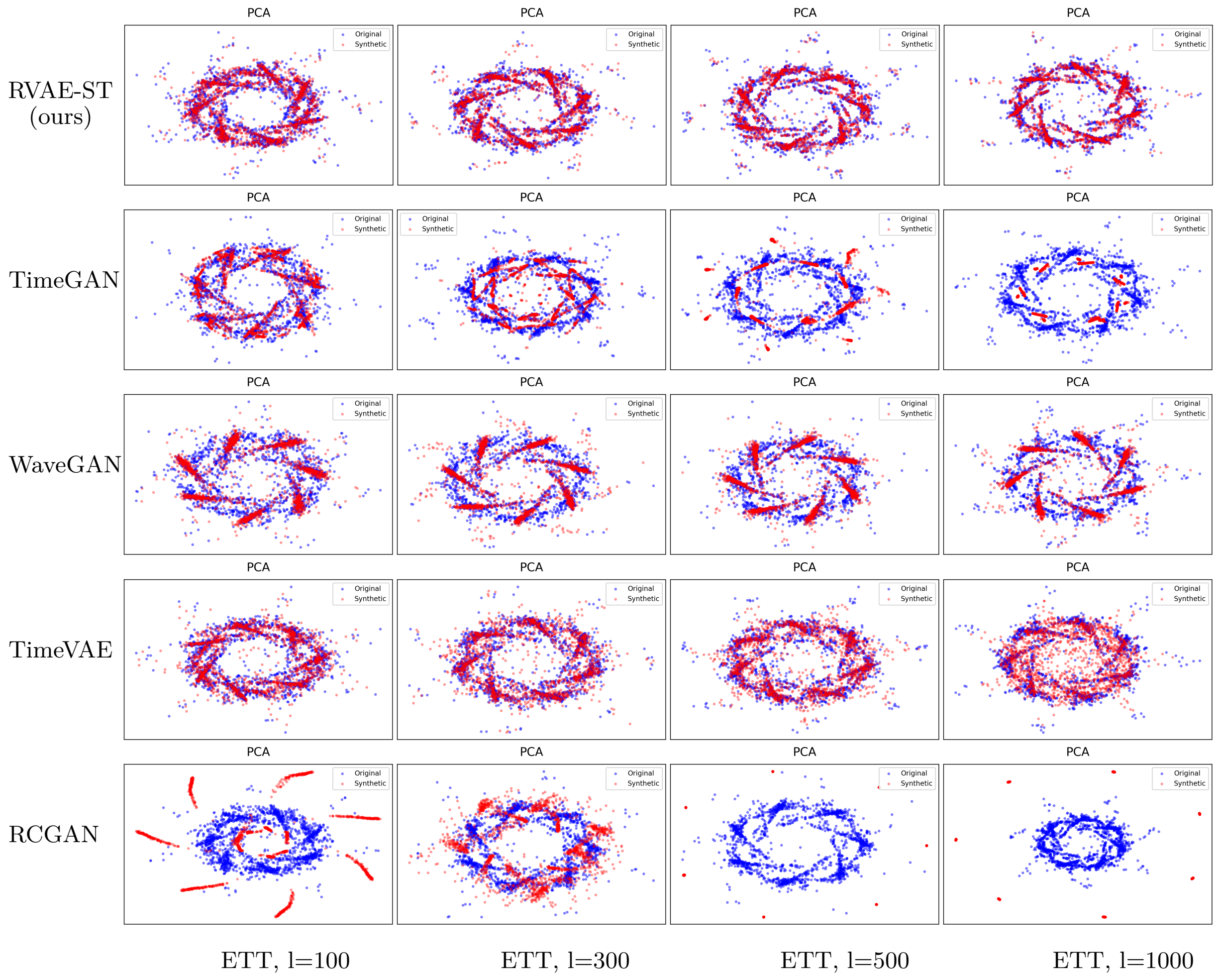


Figure 8: PCA plots for all lengths on the ETT dataset. The VAE-based models, RVAE-ST and TimeVAE, er visibly performing better than the other models. On $l = 100$ and $l = 300$, there is no visible difference between RVAE-ST and TimeVAE. On $l = 500$ and especially on $l = 1000$, visible outliers occur for TimeVAE.

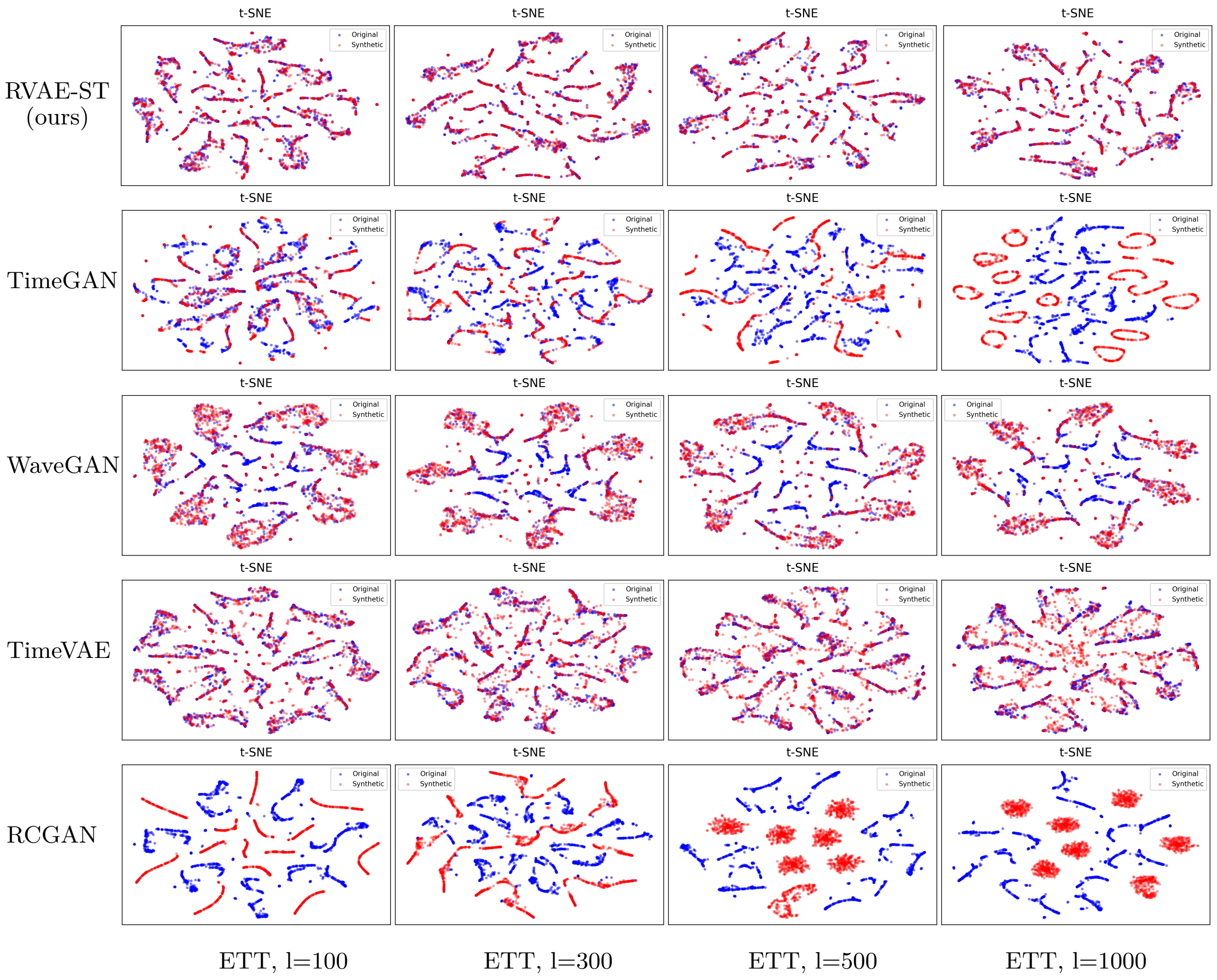


Figure 9: t-SNE plots for all sequence lengths on the ETT dataset. The VAE-based models, RVAE-ST and TimeVAE, are visibly performing better than the other models. At $l = 100$, there is no noticeable difference between RVAE-ST and TimeVAE. However, starting from $l = 300$, visible outliers begin to occur for TimeVAE, with the number of outliers increasing as the sequence length grows.