Time Weaver: A Conditional Time Series Generation Model

Sai Shankar Narasimhan [1] Shubhankar Agarwal [1] Oguzhan Akcin [1] Sujay Sanghavi [1] Sandeep Chinchali [1]

Abstract

Imagine generating a city's electricity demand pattern based on weather, the presence of an electric vehicle, and location, which could be used for capacity planning during a winter freeze. Such real-world time series are often enriched with paired heterogeneous contextual metadata (e.g., weather and location). Current approaches to time series generation often ignore this paired metadata. Additionally, the heterogeneity in metadata poses several practical challenges in adapting existing conditional generation approaches from the image, audio, and video domains to the time series domain. To address this gap, we introduce TIME WEAVER, a novel diffusion-based model that leverages the heterogeneous metadata in the form of categorical, continuous, and even time-variant variables to significantly improve time series generation. Additionally, we show that naive extensions of standard evaluation metrics from the image to the time series domain are insufficient. These metrics do not penalize conditional generation approaches for their poor specificity in reproducing the metadata-specific features in the generated time series. Thus, we innovate a novel evaluation metric that accurately captures the specificity of conditional generation and the realism of the generated time series. We show that TIME WEAVER outperforms state-of-the-art benchmarks, such as Generative Adversarial Networks (GANs), by up to 30% in downstream classification tasks on real-world energy, medical, air quality, and traffic datasets.

1. Introduction

Generating synthetic time series data is useful for creating realistic variants of private data (Yoon et al., 2020), stress-

[1]Department of Electrical and Computer Engineering, The University of Texas at Austin, Austin, USA. Correspondence to: Sai Shankar Narasimhan <nsaishankar@utexas.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

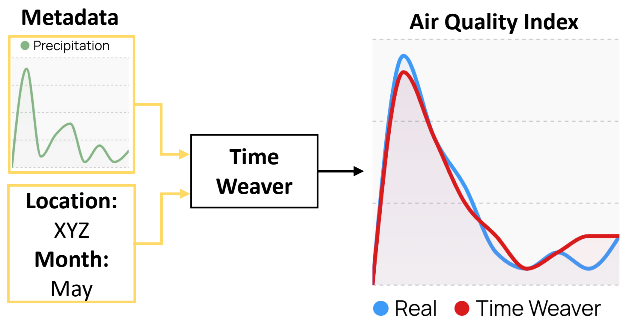

Figure 1. **TIME WEAVER generates realistic metadata-specific time series.** Consider generating the air quality index of a particular location (XYZ) given the expected precipitation (green) for a specific month (May). TIME WEAVER uses these metadata features to generate samples (red) that closely match reality (blue).

testing production systems with new scenarios (Rizzato et al., 2022; Agarwal & Chinchali, 2022), asking "what-if" questions, and even augmenting imbalanced datasets (Gowal et al., 2021). Imagine generating a realistic medical electrocardiogram (ECG) pattern based on a patient's age, gender, weight, medical record, and even the presence of a pacemaker. This generated data could be used to train medical residents, sell realistic data to third parties (anonymization), or even stress-test a pacemaker's ability to detect diseases on rare variations of ECG data.

Despite potential advantages, current time series generation methods (Yoon et al., 2019; Jeha et al., 2021; Donahue et al., 2019) ignore the rich contextual metadata and are incapable of generating time series for specific real-world conditions. This is not due to a lack of data, as standard time series datasets have long come with paired metadata conditions. Instead, it is because today's methods cannot handle diverse metadata conditions.

At first glance, generating realistic time series based on rich metadata conditions might seem like a straightforward extension of conditional image, video, or audio generation (Rombach et al., 2021; Ramesh et al., 2022; Kong et al., 2021). However, we argue that there are practical differences that make conditional time series generation and evaluation challenging, which are:

1. **Rich Metadata:** Metadata can be categorical (e.g., whether a patient has a pacemaker), quantitative (e.g.,

age), or even a time series, such as anticipated precipitation. Any conditional generative model for time series should incorporate such a diverse mix of metadata conditions (Table 1). In contrast, image, video, and audio generation often deal with static text prompts.

2. **Visual Inspection of Synthetic Data Quality:** Visual inspection is a key aspect in evaluating image generation approaches as evaluation metrics like the Inception Score (IS) are widely adopted due to their alignment with human judgment. On the contrary, it is non-trivial to glance at a time series and tell if it retains key features, such as statistical moments or frequency spectra.

3. **Architectural Differences:** In the image and audio domains, we have powerful feature extractors trained on internet-scale data (Radford et al., 2021; Wu* et al., 2023). These are vital building blocks for encoding conditions in image generation (Rombach et al., 2021). However, these models are non-existent in the time series domain due to the irregular nature of the time series datasets with respect to horizon lengths, number of channels, and the heterogeneity of the metadata.

4. **Evaluation Metrics:** Evaluating conditional generation approaches requires a metric that captures the specificity of the generated samples with respect to their paired metadata. In Fig. 4, we show how the existing metrics, such as the time series equivalent of the standard Frechet Inception Distance (FID) score (Jeha et al., 2021), fail to capture this specificity and only measure how close the real and generated data distributions are. This is because these metrics completely ignore the paired metadata in their evaluation.

Given the above differences and insufficiencies in metrics, **our contributions** are:

1. We present Time Weaver (Fig. 1), a novel diffusion model for generating realistic multivariate time series conditioned on metadata. We specifically innovate on the standard diffusion model architecture to process categorical and continuous metadata conditions.

2. We propose the Joint Frechet Time Series Distance (J-FTSD), specifically designed to evaluate conditional time series data generation models. J-FTSD incorporates time series and metadata conditions with feature extractors trained using a contrastive learning framework. In Sec. 6, we showcase J-FTSD's ability to accurately rank approaches based on their ability to model conditional time series data distributions.

3. We show that our approach significantly outperforms the state-of-the-art GAN models in generating high-quality, metadata-specific time series on real-world energy, healthcare, pollution, and traffic datasets (Fig. 2).

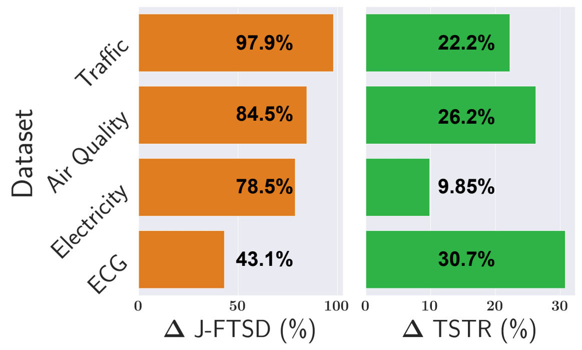

Figure 2. **Time Weaver beats GANs on all datasets on the Joint Frechet Time Series Distance (J-FTSD) and Train on Synthetic Test on Real (TSTR) metrics.** J-FTSD indicates the distributional similarity between the generated and real time series datasets. Lower values of J-FTSD indicate that both generated and real time series distributions are closer. TSTR indicates the performance of a downstream task model trained on generated time series data and evaluated on real time series data. Higher values of TSTR indicate higher quality of the generated time series data. We show the percentage improvement of Time Weaver over state-of-the-art GAN models on four diverse datasets.

2. Background and Related Works

Generative Models in Time Series: Recently, Generative Adversarial Networks (GANs) (Donahue et al., 2019; Yoon et al., 2019; Li et al., 2022; Thambawita et al., 2021) have emerged as popular methods for time series data generation. However, these GAN-based approaches often struggle with unstable training and mode collapse (Chen, 2021). In response, Diffusion Models (DMs) (Sohl-Dickstein et al., 2015) have been introduced in the time series domain (Alcaraz & Strodthoff, 2023; Tashiro et al., 2021), offering more realistic data generation. DMs are a class of generative models that are state-of-the-art in a variety of domains, including image (Dhariwal & Nichol, 2021; Ho et al., 2020), speech (Chen et al., 2020; Kong et al., 2021), and video generation (Ho et al., 2022). DMs operate by defining a Markovian forward process q. The forward process gradually adds noise to a clean data sample $x_0 \sim \mathcal{X}$, where $\mathcal{X}$ is the data distribution to be learned. The forward process is predetermined by fixing a noise variance schedule $\{\beta_1, \ldots, \beta_T\}$, where $\beta_t \in [0, 1]$ and T is the total number of diffusion steps. The following equations describe the forward process:

$$q(x_1, \ldots, x_T \mid x_0) = \prod_{t=1}^{T} q(x_t \mid x_{t-1}), \qquad (1)$$

$$q(x_t \mid x_{t-1}) = \mathcal{N}(\sqrt{1-\beta_t} x_{t-1}, \beta_t \mathbf{I}). \qquad (2)$$

Here, $\mathcal{N}(\mu, \Sigma)$ represents a Gaussian distribution with mean μ and covariance matrix Σ. During training, a clean sample x_0 is transformed into x_t using Eq. (2). Then, a neural network, $\theta_{\text{denoiser}}(x_t, t)$, is trained to estimate the amount of noise added between x_{t-1} and x_t with the following loss function:

$$\mathcal{L}_{\text{DM}} = \mathbb{E}_{x\sim\mathcal{X},\epsilon\sim\mathcal{N}(\mathbf{0},\mathbf{I}),t\sim\mathcal{U}(1,T)}\left[\|\epsilon - \theta_{\text{denoiser}}(x_t, t)\|_2^2\right]. \tag{3}$$

Here, $t \sim \mathcal{U}(1, T)$ indicates that t is sampled from a uniform distribution between 1 and T, and ϵ is the noise added to x_{t-1} to obtain x_t. In inference, we start from $x_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, where $\mathcal{N}(\mathbf{0}, \mathbf{I})$ represents a zero mean, unit variance Gaussian distribution, and iteratively denoise using θ_{denoiser} to obtain a clean sample from the data distribution $\mathcal{X}$, *i.e.*, $x_T \to x_{T-1}, \ldots, x_0$. A detailed explanation of DMs is provided in App. A.1.

For conditional DMs, the most commonly used approach is to keep the forward process the same as in Eq. (2), and add additional conditions c to the reverse process. Minimizing $\|\epsilon - \theta_{\text{denoiser}}(x_t, t, c)\|_2^2$ in the loss function provided in Eq. (3) facilitates learning the conditional distribution. Conditional DMs are used in image, video (Saharia et al., 2021; Lugmayr et al., 2022; Rombach et al., 2021; Ramesh et al., 2022), and speech (Kong et al., 2021) generation. These models allow for diverse conditioning inputs, like text, image, or even segmentation maps. However, these methods rely on image-focused tools like Convolutional Neural Networks (CNNs), which struggle to maintain essential time series characteristics such as long-range dependencies, as noted in (Gu et al., 2022). For time series data, models such as CSDI (Tashiro et al., 2021) and SSSD (Alcaraz & Strodthoff, 2022) exist but are mainly limited to imputation tasks without substantial conditioning capabilities. Closest to our work, Alcaraz & Strodthoff (2023) attempt to incorporate ECG statements as metadata (only categorical) for ECG generation. However, this approach falls short as it does not consider heterogeneous metadata. Our method surpasses these limitations by effectively handling a broader range of metadata modalities, thus enabling more realistic time series data generation under varied heterogeneous conditions.

Metrics for Conditional Time Series Generation: Various metrics have been developed in the time series domain, focusing on the practical utility of the generated time series data. To this end, the Train on Synthetic Test on Real (TSTR) metric (Jordon et al., 2018; Esteban et al., 2017) is used to assess the ability of synthetic data to capture key features of the real dataset. TSTR metrics have been widely used to evaluate unconditional time series generation. Yoon et al. (2019) proposed the predictive score where synthetic time series data is used to train a forecaster, and the forecaster's performance is evaluated on real time series data. More traditional approaches include average cosine similarity, Jensen distance (Li et al., 2022), and autocorrelation comparisons (Lin et al., 2020; Bahrpeyma et al., 2021). However, these heuristics often fail to fully capture the nuanced performance of conditional generative models.

A more popular method to evaluate generative models is to use distance metrics between the generated and real data samples. One of the most commonly used distance metrics is the Frechet Distance (FD) (Fréchet, 1957). The FD between two multivariate Gaussian distributions $\mathcal{D}_1 \sim \mathcal{N}(\mu_1, \Sigma_1)$ and $\mathcal{D}_2 \sim \mathcal{N}(\mu_2, \Sigma_2)$ is:

$$FD(\mathcal{D}_1, \mathcal{D}_2) = \|\mu_1 - \mu_2\|^2 + \text{Tr}(\Sigma_1 + \Sigma_2 - 2(\Sigma_1\Sigma_2)^{\frac{1}{2}}). \tag{4}$$

To evaluate image generation models, the FD is adjusted to the Frechet Inception Distance (FID) (Heusel et al., 2017). FID uses a feature extractor, the Inception-v3 model (Szegedy et al., 2015), to transform images into embeddings upon which the FD is calculated. Similar adaptations such as the Frechet Video Distance (Unterthiner et al., 2018), Frechet ChemNet Distance (Preuer et al., 2018), and Context-FID (Jeha et al., 2021) exist for other domains, employing domain-specific feature extractors. However, these metrics are designed only to evaluate unconditional data generation since they only match the true data distribution marginalizing over all the conditions.

To evaluate conditional generation models, many metrics are proposed for categorical conditions (Murray, 2019; Huang et al., 2018; Benny et al., 2020; Liu et al., 2018; Miyato & Koyama, 2018). To create a more general metric, Soloveitchik et al. (2022) proposed the conditional FID (CFID) metric that works with continuous conditionals and calculates the conditional distributions of the generated and real data given the condition. In particular, DeVries et al. (2019) propose the Frechet Joint Distance (FJD), where the embeddings of the image and condition are obtained with different embedding functions and concatenated to create a joint embedding space. DeVries et al. (2019) consider conditions that are classes (image category), text descriptions (image captions), or images (for tasks like style transfer). However, in our case, the metadata could be any arbitrary combination of categorical, continuous, and time-varying conditions. Additionally, like other metrics considered in the literature, FJD is defined for image generation and does not consider the unique characteristics of time series data. In contrast, our proposed J-FTSD metric is specifically designed to evaluate time series data generation models conditioned on heterogeneous metadata.

3. Problem Formulation

Consider a multivariate time series sample $x \in \mathbb{R}^{L\times F}$, where L denotes the time series horizon, and F denotes the number of channels. Each sample x is associated with metadata c, comprising categorical features $c_{\text{cat}} \in \mathbb{N}^{L\times K_{\text{cat}}}$ and continuous features $c_{\text{cont}} \in \mathbb{R}^{L\times K_{\text{cont}}}$. Here, K_{cat} and K_{cont} indicate the total numbers of categorical and contin-

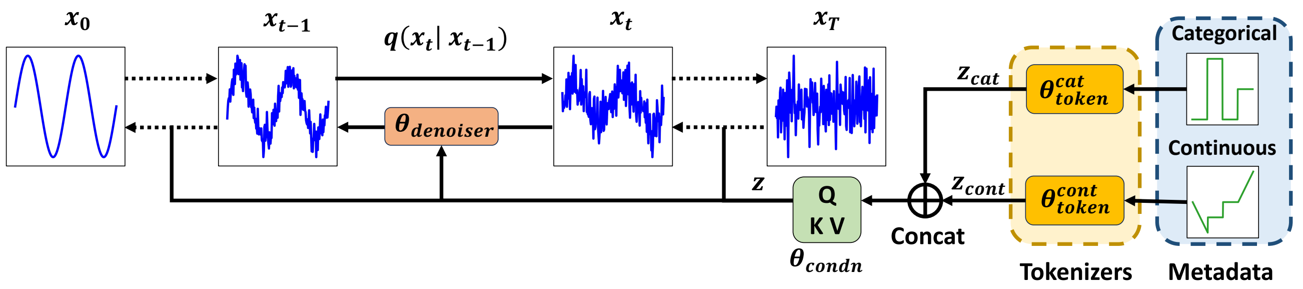

Figure 3. **Time Weaver architecture for incorporating metadata in the diffusion process:** This figure shows the training and inference processes of Time Weaver. For training, we start from the original sample x_0 (on the left) and gradually add noise through a forward process $q(x_t \mid x_{t-1})$, resulting in noisy samples x_t. The denoiser, θ_{denoiser}, is trained to estimate the amount of noise added to obtain x_t from x_{t-1}. During inference, the categorical and continuous metadata are first preprocessed with tokenizers $\theta_{\text{token}}^{\text{cat}}$ and $\theta_{\text{token}}^{\text{cont}}$, respectively. Then, we concatenate their output and process it through a self-attention layer θ_{condn} to create the metadata embedding z. This embedding is fed into θ_{denoiser} with the noisy sample x_t to obtain x_{t-1}. The denoising process is repeated for T diffusion steps to obtain a clean sample similar to x_0.

uous metadata features, respectively. These features are concatenated as $c = c_{\text{cat}} \oplus c_{\text{cont}}$, where $\oplus$ represents the vector concatenation operation. Thus, the metadata domain is defined as $\mathbb{N}^{L\times K_{\text{cat}}} \times \mathbb{R}^{L\times K_{\text{cont}}}$. Note that the domains of c_{cat} and c_{cont} allow time-varying metadata features.

Example: Consider generating time series data representing traffic volume on a highway ($F = 1$) over 96 hours ($L = 96$) using paired metadata. This metadata includes seven time-varying categorical features like holidays (12 unique labels) and weather descriptions (11 unique labels), denoted by $K_{\text{cat}} = 7$. It also includes four time-varying continuous features like expected temperature and rain forecast, represented by $K_{\text{cont}} = 4$.

We denote the dataset $D_{x,c} = \{(x_i, c_i)\}_{i=1}^{n}$ consisting of n independent and identically distributed (i.i.d) samples of time series data x and paired metadata c, sampled from a joint distribution $p(x, c)$. **Our objective is to** develop a conditional generation model G such that the samples generated by $G(c)$ distributionally match $p(x|c)$.

4. Conditional Time Series Generation using Time Weaver

Our approach, Time Weaver, is a diffusion-based conditional generation model. We choose DMs over GANs as we consider heterogeneous metadata, *i.e.*, the metadata can contain categorical, continuous, or even time-varying features. Previous works show that the conditional variants of GANs suffer from mode collapse when dealing with continuous conditions (Ding et al., 2020). Additionally, the proposed alternatives have not been tested in the time series domain for heterogeneous metadata. Our Time Weaver model consists of two parts - a denoiser backbone that generates data and a preprocessing module that processes the time-varying categorical and continuous metadata features.

Metadata Preprocessing: The preprocessing step involves handling the metadata $c = c_{\text{cat}} \oplus c_{\text{cont}}$. Here, $c_{\text{cat}} \in \mathbb{N}^{L\times K_{\text{cat}}}$ and $c_{\text{cont}} \in \mathbb{R}^{L\times K_{\text{cont}}}$ represent time-varying categorical and continuous metadata features, respectively (see Sec. 3). To better incorporate these features from different modalities, we process them separately and then combine them with a self-attention layer.

- The categorical tokenizer $\theta_{\text{token}}^{\text{cat}}$ first converts each category in c_{cat} into one-hot encoding and then processes with fully connected (FC) layers to create the categorical embedding $z_{\text{cat}} \in \mathbb{R}^{L\times d_{\text{cat}}}$. Similarly, the continuous tokenizer $\theta_{\text{token}}^{\text{cont}}$ also uses FC layers to encode continuous metadata c_{cont} into the continuous embedding $z_{\text{cont}} \in \mathbb{R}^{L\times d_{\text{cont}}}$. These FC layers learn the correlation between metadata features within the categorical and continuous domains. Using FC layers is just a design choice; more sophisticated layers can also be used.
- z_{cat} and z_{cont} are then concatenated and passed through a self-attention layer θ_{condn} to generate the metadata embedding $z \in \mathbb{R}^{L\times d_{\text{meta}}}$. The self-attention layer equips the generative model to capture the temporal relationship between different metadata features.

Here, d_{cat}, d_{cont}, and d_{meta} are design choices, and we refer the reader to App. A.4 for further details.

Denoiser: As the denoiser backbone for Time Weaver, we rely on two state-of-the-art architectures - CSDI (Tashiro et al., 2021) and SSSD (Alcaraz & Strodthoff, 2022). The CSDI model uses feature and temporal self-attention layers to process sequential time series data, while SSSD uses structured state-space layers. Note that these denoisers are designed for imputation and forecasting tasks. So, they are designed to take unimputed and historical time series as respective inputs. We modify these denoisers into more flexible metadata-conditioned time series generators by augmenting them with preprocessing layers ($\theta_{\text{token}}^{\text{cat}}$, $\theta_{\text{token}}^{\text{cont}}$, and θ_{condn}). We refer the reader to App. A.4 for details regarding architectural changes. We train the preprocessing layers and the denoiser θ_{denoiser} jointly with the following loss:

$$\mathcal{L}_{(\theta_{\text{denoiser}},\theta_{\text{condn}},\theta^{\text{cont}}_{\text{token}},\theta^{\text{cat}}_{\text{token}})} = \mathbb{E}_{x,c\sim D_{x,c},\epsilon\sim\mathcal{N}(\mathbf{0},\mathbf{I}),t\sim\mathcal{U}(1,T)} [\|\epsilon - \theta_{\text{denoiser}}(x_t,t,z)\|_2^2], \quad (5)$$

where $z = \theta_{\text{condn}}(\theta^{\text{cat}}_{\text{token}}(c_{\text{cat}}) \oplus \theta^{\text{cont}}_{\text{token}}(c_{\text{cont}}))$, $D_{x,c}$ represents the dataset of time series and paired metadata sampled from the joint distribution $p(x,c)$, and T is the total number of diffusion steps. As explained in Sec. 2, minimizing the loss in Eq. (5) allows TIME WEAVER to learn how to generate samples from the conditional distribution $p(x|c)$. During inference, we start from $x_T \sim \mathcal{N}(\mathbf{0},\mathbf{I})$ and iteratively denoise (with metadata c as input) for T steps to generate $x_0 \sim p(x|c)$. This process is depicted in Fig. 3.

5. Joint Frechet Time Series Distance

A good distance metric should penalize the conditional generation approach (provide higher values) if the real and generated joint distributions of the time series and the paired metadata do not match. Existing metrics such as Context-FID (Jeha et al., 2021) rely only on the time series feature extractor, and the metric computation does not involve the paired metadata. This prevents these metrics from penalizing conditional generation approaches for their inability to reproduce metadata-specific features in the generated time series. Therefore, we propose a new metric to evaluate metadata-conditioned time series generation, the Joint Frechet Time Series Distance (J-FTSD).

In J-FTSD, we compute the FD between the real and generated joint distributions of time series and paired metadata. Consider samples from a real data distribution indicated by $D_r = \{(x^r_1, c_1), \ldots, (x^r_n, c_n)\}$, where $x^r_i \in \mathbb{R}^{L\times F}$ indicates the time series, and $c_i \in \mathbb{R}^{L\times K}$ indicates the paired metadata as explained in Sec. 3. We denote the dataset of generated time series and the corresponding metadata as $D_g = \{(x^g_1, c_1), \ldots, (x^g_n, c_n)\}$, where $x^g_i = G(c_i)\ \forall\ i \in [1,n]$, and G denotes any arbitrary conditional generation model, as defined in Sec. 3. First, similar to the FID and FJD computations, we project the time series and the paired metadata into a lower-dimensional embedding space using $\phi_{\text{time}}(\cdot) : \mathbb{R}^{L\times F} \to \mathbb{R}^{d_{\text{emb}}}$ and $\phi_{\text{meta}}(\cdot) : \mathbb{R}^{L\times K} \to \mathbb{R}^{d_{\text{emb}}}$ as the respective feature extractors, where d_{emb} is the size of the embedding. We concatenate these time series and metadata embeddings to create a joint embedding space. We then calculate the FD over the joint embedding space. As such, the J-FTSD metric is formally defined as:

$$\text{J-FTSD}(D_g, D_r) = \|\mu_{z^r} - \mu_{z^g}\|^2 + \text{Tr}(\Sigma_{z^r} + \Sigma_{z^g} - 2(\Sigma_{z^r}\Sigma_{z^g})^{\frac{1}{2}}). \quad (6)$$

Here, μ_{z^d} and Σ_{z^d} for $d \in \{g, r\}$ are calculated as:

$$z^d_i = \phi_{\text{time}}(x^d_i) \oplus \phi_{\text{meta}}(c_i) \qquad \forall i : (x^d_i, c_i) \in D_d,$$

$$\mu_{z^d} = \frac{1}{n}\sum_{i=1}^{n} z^d_i,\ \Sigma_{z^d} = \frac{1}{n-1}\sum_{i=1}^{n}(z^d_i - \mu_{z^d})(z^d_i - \mu_{z^d})^\top.$$

In essence, J-FTSD computes the FD between the Gaussian approximations of the real and generated joint embedding datasets. In Eq. (6), μ_{z^r}, μ_{z^g} and Σ_{z^r}, Σ_{z^g} are the mean and the variance of the Gaussian approximations of the real and generated joint embedding datasets, respectively.

Training Feature Extractors: Now, we describe our approach to obtain the feature extractors ϕ_{time} and ϕ_{meta}. As explained in Sec. 2, DeVries et al. (2019) suggest using separate encoders for data samples and conditions. However, they only deal with a specific type of condition, and this naturally poses a problem for a straightforward extension of their approach to our case, where the metadata could be any arbitrary combination of categorical, continuous, and time-varying features. As such, we propose a novel approach to train the feature extractors ϕ_{meta} and ϕ_{time} specific to the time series domain. We jointly train ϕ_{time} and ϕ_{meta} with contrastive learning to better capture the joint distribution of the time series and paired metadata as contrastive learning is a commonly used method to map data from various modalities into a shared latent space (Yuan et al., 2021; Zhang et al., 2022; Ramesh et al., 2022).

Algorithm 1 One iteration for training time series ϕ_{time} and metadata ϕ_{meta} feature extractors.

```
1: Input: Time series feature extractor $\phi_{\text{time}}$, Metadata feature extractor $\phi_{\text{meta}}$, Time series batch $X_{\text{batch}}$, Paired Metadata batch $C_{\text{batch}}$, Number of patches $N_{\text{patch}}$, Patch length $L_{\text{patch}}$, Batch size $N_{\text{batch}}$.
2: Randomly select $N_{\text{patch}}$ patches of length $L_{\text{patch}}$ from each sample in $X_{\text{batch}}$ and $C_{\text{batch}}$ to generate $X^{\text{patch}}_{\text{batch}}$ and $C^{\text{patch}}_{\text{batch}}$.
3: Obtain the time series and metadata embedding - $\phi_{\text{time}}(X^{\text{patch}}_{\text{batch}})$ and $\phi_{\text{meta}}(C^{\text{patch}}_{\text{batch}})$ respectively.
4: Obtain the logits - $\phi_{\text{time}}(X^{\text{patch}}_{\text{batch}})^T \phi_{\text{meta}}(C^{\text{patch}}_{\text{batch}})$.
5: Define the labels - $[0, 1, 2, \ldots, N_{\text{batch}} \times N_{\text{patch}} - 1]$.
6: Compute $\mathcal{L}_{\text{time}} = \mathcal{L}_{\text{CE}}(\texttt{logits}, \texttt{labels})$.
7: Compute $\mathcal{L}_{\text{meta}} = \mathcal{L}_{\text{CE}}(\texttt{logits.T}, \texttt{labels})$.
8: Compute $\mathcal{L}_{\text{total}} = (\mathcal{L}_{\text{time}} + \mathcal{L}_{\text{meta}})/2$.
9: Update parameters of $\phi_{\text{time}}$ and $\phi_{\text{meta}}$ to minimize $\mathcal{L}_{\text{total}}$.
```

Algorithm 1 summarizes one training iteration of our feature extractors ϕ_{time} and ϕ_{meta}. This is visually depicted in App. A.3. Given the batch of time series X_{batch} and metadata C_{batch}, we randomly pick N_{patch} patches with horizon L_{patch} from each time series and metadata sample in batches X_{batch} and C_{batch} (line 1). Then, we obtain the time series and metadata embeddings for all patches through their respective feature extractors, ϕ_{time} for time series, and ϕ_{meta}

Dataset	Horizon	# Channels	Categorical features	Continuous features
Air Quality (Chen, 2019)	96	6	12 Stations, 5 years, 12 months, 31 dates, 24 hours, 17 wind directions	Temperature, Pressure, Dew point temperature, Rain levels, Wind speed
Traffic (Hogue, 2019)	96	1	12 holidays, 7 years, 12 months, 31 dates, 24 hours, 11 broad weather descriptions, 38 fine weather descriptions	Temperature, Rain levels, Snow fall levels, Cloud conditions
Electricity (Trindade, 2015)	96	1	370 users, 4 years, 12 months, 31 dates	N.A.
ECG (Wagner et al., 2020)	1000	8	71 heart disease statements	N.A.

Table 1. **Dataset overview for experiments with Time Weaver.** This table outlines the key characteristics of the datasets employed in our experiments. These datasets, encompassing Air Quality, Traffic, Electricity, and ECG, have been selected to demonstrate Time Weaver's versatility across different time horizons (*col 1*), number of channels (*col 2*), and a wide range of metadata types (*col 3,4*).

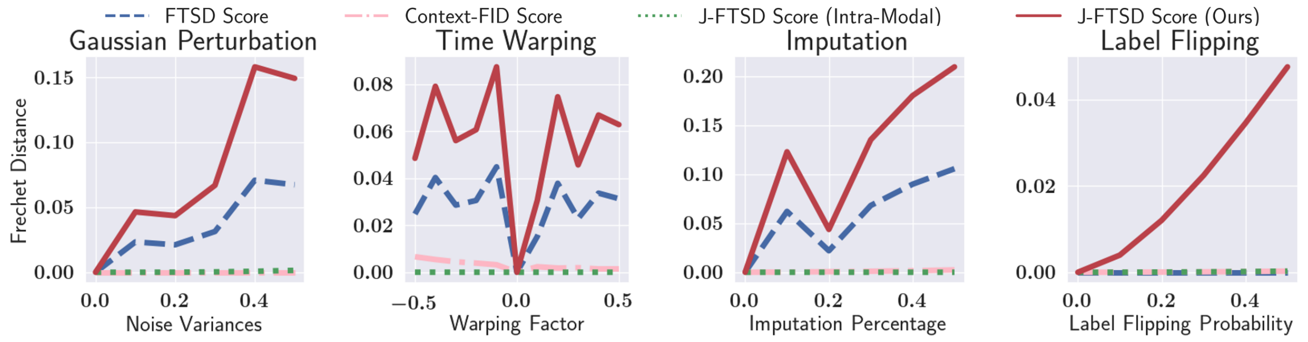

Figure 4. **J-FTSD correctly penalizes the conditional time series data distribution.** A good metric should penalize the conditional generation approaches for not being specific to the metadata and deviating from real time series data distribution. As such, we compare the sensitivity of different distance metrics under various synthetic disturbances on the Air Quality dataset (starting from the left); we add Gaussian noise, warp, impute, and randomly change the metadata of the time series samples. We clearly show that as the amount of perturbation increases, our J-FTSD metric (in red) shows the highest sensitivity, correctly capturing the dissimilarities between the perturbed and the original datasets. In contrast, the other metrics remain unchanged or show lower sensitivity.

for metadata (line 2). Finally, we compute the dot product of time series and metadata embeddings (line 3) and obtain the symmetric cross-entropy loss (lines 4 - 7), which is used to jointly update the parameters of ϕ_{time} and ϕ_{meta} (line 8).

In essence, we learn a joint embedding space for time series and metadata by jointly training ϕ_{time} and ϕ_{meta}. This is achieved by adjusting the feature extractors' parameters to maximize the cosine similarity of the time series embeddings and the metadata embeddings of $N_{\text{batch}} \times N_{\text{patch}}$ pairs of time series and paired metadata in the batch. In our experiments, we used the Informer encoder architecture (Zhou et al., 2021) for ϕ_{time} and ϕ_{meta}. We choose L_{patch} based on the length of the smallest chunk of the time series that contains metadata-specific features. We refer the readers to App. A.3 for further details on the choices of N_{patch}, L_{patch}, and the encoder architecture.

Why is J-FTSD a good metric for evaluating conditional generation models? One aspect of the J-FTSD computation involves estimating the covariance between the time series and the metadata embeddings. Additionally, jointly training the feature extractors with contrastive learning aids in effectively capturing the correlation between the time series and the metadata embeddings. Therefore, the covariance term decreases if the generated time series does not contain metadata-specific features. This allows J-FTSD to accurately penalize for the differences between the real and generated joint distributions, which directly translates to penalizing conditional generation approaches for their poor specificity in reproducing metadata-specific features.

6. Experiments

We evaluated the performance of Time Weaver across datasets featuring a diverse mix of seasonalities, discrete and continuous metadata conditions, a wide range of horizons, and multivariate correlated channels. The list of datasets and their metadata features are provided in Table 1. All models are trained on the train split, while all metrics are reported on the test split, further detailed in App. A.2.

Baselines: We represent the results for the CSDI and SSSD backbones for Time Weaver as Time Weaver-CSDI and Time Weaver-SSSD, respectively. Since there are no existing approaches for metadata-conditioned time series generation with categorical, continuous, and time-variant metadata features, we modify the existing state-of-the-art GAN approaches to incorporate metadata conditions, similar to Time Weaver. The GAN baselines include CNN-based approaches like WaveGAN (Donahue et al., 2019), an audio-focused GAN model, and Pulse2Pulse (Thambawita et al.,

Approach	Air Quality		ECG		Traffic		Electricity	
	J-FTSD ↓	TSTR ↑	J-FTSD ↓	TSTR ↑	J-FTSD ↓	TSTR ↑	J-FTSD ↓	TSTR ↑
WaveGAN (Donahue et al., 2019)	14.25±0.79	0.61±0.01	9.55±0.01	0.65±0.001	25.69±0.01	0.54±0.01	7.82±0.002	0.57±0.007
Pulse2Pulse (Thambawita et al., 2021)	22.07±0.02	0.60±0.002	13.49±0.04	0.63±0.03	17.70±0.002	0.52±0.03	2.8±0.01	0.71±0.004
Time Weaver-CSDI	**2.2±0.07**	**0.77±0.01**	7.25±0.09	0.83±0.001	0.53±0.01	**0.66±0.06**	**0.6±0.003**	**0.78±0.001**
Time Weaver-SSSD	8.61±0.18	0.63±0.02	**5.43±0.1**	**0.85±0.007**	**0.36±0.03**	0.65±0.07	1.19±0.008	0.77±0.001

Table 2. **Time Weaver outperforms GAN-based approaches on the J-FTSD and TSTR metrics**. The table shows the performance of all the models (rows) on specified datasets (columns). The Time Weaver variants significantly outperform GANs on the J-FTSD and TSTR metrics. Our experimental findings also confirm that lower J-FTSD scores correspond to higher AUC (TSTR) scores when tested on the original test dataset, showcasing the utility of our proposed J-FTSD metric in evaluating the quality of the generated data distribution. We report the mean and standard deviation of both metrics, averaged over three seeds.

2021), a model specializing in DeepFake generation. Additionally, we compare the Time Weaver variants against a diffusion model with a U-Net 1D (Ronneberger et al., 2015) denoiser backbone. We chose the 1D variant of U-Net as the basic U-Net architecture is the most preferred denoiser backbone used in pixel-space image diffusion. The exact training details are provided in App. A.4 and A.5. We additionally tried comparing with TimeGAN (Yoon et al., 2019), a Recurrent Neural Network (RNN) based approach, and TTS-GAN (Li et al., 2022), a Transformer-based approach. However, both of these GAN models did not converge on any of the datasets. We show their training results in App. A.5.4.

Evaluation Metrics: We evaluate our approaches and the GAN baselines using the J-FTSD metric, as detailed in Sec. 5. To validate the correctness of J-FTSD's evaluation, we also report the area under the curve (AUC) scores of a classifier trained only using synthetic data. The classifier is trained to predict the metadata given the corresponding synthetic time series. We then test this classifier on the real unseen test dataset. High accuracy indicates that our synthetic data faithfully retains critical features of the paired metadata. We use a standard ResNet-1D (He et al., 2016) model for the classifier. We denote this metric as TSTR in Table 2. For each dataset, the categories for which we train the classifier are listed as follows: Electricity - Months (12), Air Quality - Stations (12), Traffic - Weather Descriptions (11), and ECG - Heart Conditions (71). The exact training steps of the classifiers and a detailed description of the metrics are outlined in App. A.6.

Experimental Results and Analysis: Our experiments demonstrate that the Time Weaver models significantly outperform baseline models in synthesizing time series data across all evaluated benchmarks. Our experiments address the following key questions:

Does the J-FTSD metric correctly penalize when the generated time series samples are not specific to the paired metadata? In Fig. 4, we assess the sensitivity of our J-FTSD metric against previous FD-based metrics. This assessment involves introducing controlled perturbations into the time series to test the sensitivity of the metric. These perturbations include *Gaussian noise* - which introduces Gaussian noise of increasing variance; *time warping*, involving scaling adjustments; *imputation* - imputing the time series with local mean and *label flipping* - where metadata conditions are randomly changed, decoupling them from the time series. An effective metric should demonstrate an increased sensitivity when the real and generated joint distributions of time series and metadata diverge. We compare J-FTSD against three FD-based metrics: 1) the FTSD score, which calculates the FD using only time series embeddings (derived from the ϕ_{time} feature extractor); 2) the Context-FID score (Jeha et al., 2021) where ϕ_{time} is trained to maximize similarity of similar time series; 3) the J-FTSD (Intra-Modal) score, which differs from J-FTSD in that the time series and metadata feature extractors are trained individually to maximize the embedding similarity for similar samples. Our J-FTSD metric is the most sensitive when compared to other metrics under synthetic disturbances. The key benefit of our metric can be observed in the label-flipping experiment, where only our metric increases as we increase the label-flipping probability in the paired metadata conditions. Other metrics remain unchanged and lack sufficient sensitivity because they overlook paired metadata in their distance calculations, a critical factor that J-FTSD adeptly incorporates. Additionally, the J-FTSD (Intra-Modal) score remains mostly unchanged under these perturbations, highlighting the advantage of the joint training of the time series and metadata feature extractors in our metric. Experiments in Fig. 4 underscore the importance of our J-FTSD metric in assessing the quality of the generated time series data.

How does Time Weaver compare against other approaches on real-world datasets? Across all the datasets, the Time Weaver variants consistently outperform GAN models in terms of J-FTSD scores, as shown in Table 2. Specifically, we beat the best GAN model by roughly $6\times$ on the Air Quality dataset, $1.75\times$ on the ECG dataset, $4\times$ on the Electricity dataset, and more than $40\times$ on the Traffic dataset. To understand this performance gain, we conduct an ablation study where we compare the Time Weaver-CSDI model and WaveGAN (Donahue et al., 2019) in the presence and absence of metadata on the Air Quality and Electricity datasets. We use the FTSD metric, which is the Frechet Distance without considering the metadata embed-

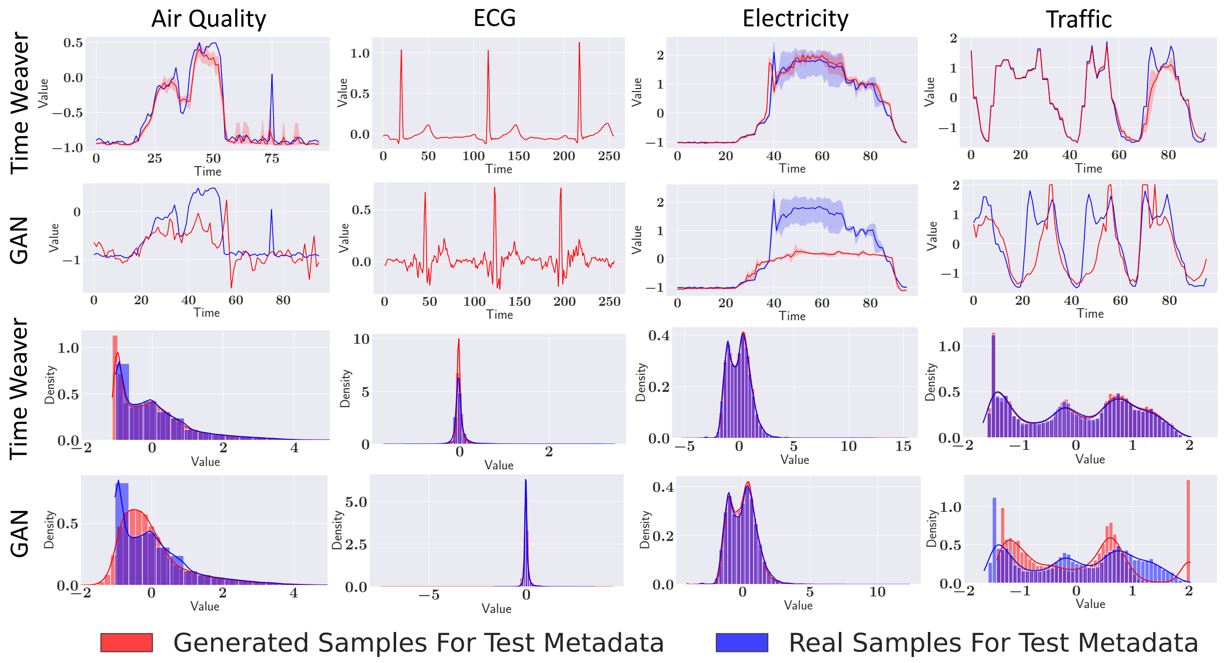

Figure 5. **TIME WEAVER generated time series distributions match the real time series distributions.** Each column represents a different dataset. The real time series is in blue, while the generated time series is in red. The first & third rows correspond to the TIME WEAVER model, and the second & fourth rows correspond to the best-performing GAN model. The top two rows have the real and generated time series for unseen test metadata conditions. The bottom two rows compare the histograms of the real and generated time series values aggregated over their respective datasets, also for unseen test metadata conditions. Both results indicate that our TIME WEAVER model can generate realistic time series samples that are specific to the corresponding metadata conditions, beating the previous state-of-the-art GAN model. In both scenarios, the GAN model fails to match the real time series data distribution, while our TIME WEAVER model has learned the correct conditional distribution for specific metadata conditions, specifically for the Air Quality and Traffic datasets.

dings, as our evaluation metric. The quantitative results of this study are summarized in Table 8. We find that TIME WEAVER provides lower FTSD for the Electricity dataset, and WaveGAN provides lower FTSD for the Air Quality dataset in the absence of metadata. However, in the presence of metadata, we note that TIME WEAVER outperforms WaveGAN on both datasets. The key insight is that TIME WEAVER processes metadata more efficiently than GAN-based approaches to generate metadata-specific time series. This observation aligns with the recent work (Ding et al., 2020) that shows GANs perform poorly with continuous conditions. The metadata conditions in the time series domain are so complex (a mix of categorical, continuous, and time-varying conditions) that the GAN models fail to learn a good conditional time series distribution.

Additionally, we compare TIME WEAVER against a diffusion model baseline with U-Net 1D (Ronneberger et al., 2015) as the denoiser backbone. The quantitative results for this comparison are provided in Table 9. We note that the best-performing TIME WEAVER model outperforms the diffusion model baseline with the U-Net 1D denoiser, with an average improvement of 9% in TSTR and 17% in J-FTSD metrics. This superior performance highlights the importance of time series-specific innovations in TIME WEAVER for complex metadata-specific time series generation.

Does the synthetic data generated by TIME WEAVER capture metadata-specific features to train an accurate classifier? When training with the generated synthetic time series data, the classifier's accuracy in classifying metadata hinges on the presence of distinct metadata-specific features in the time series. The high TSTR scores in Table 2 strongly suggest that the data generated by TIME WEAVER retain the essential characteristics necessary to train classifiers that exhibit high AUC on real unseen test data. The marked improvement in TSTR scores with TIME WEAVER when compared to GAN models demonstrates both the practical value and the superior quality of the synthetic data generated by our model.

Does lower J-FTSD correlate with higher TSTR performance? The experimental data, as outlined in Table 2, exhibit a clear correlation: lower J-FTSD scores are consistently associated with higher TSTR scores on the original, unseen test dataset. This correlation is expected as both metrics evaluate the precision of the generated time series relative to the corresponding metadata, as well as the closeness of the real and generated joint distributions. This further underscores the effectiveness of the J-FTSD metric as a reliable indicator to assess the quality of generated data. We further solidify the validity of J-FTSD by showing that lower values of J-FTSD correspond to lower values of the Dynamic Time Warping (DTW) metric, where DTW is computed between the real and generated time series samples. Lower DTW indicates higher similarity between the real and generated time series. We refer the reader to App. A.7.3 for further details and experimental results.

Does the synthetic data generated by TIME WEAVER qualitatively match the real data? Figure 5 (top two rows) displays the quality and realism of the time series data generated by the best-performing TIME WEAVER model. This figure contrasts generated time series samples with real ones under identical metadata conditions. The comparison demonstrates that the TIME WEAVER model produces time series samples that are highly similar to real samples, effectively mapping metadata to the corresponding time series. In contrast, GAN baseline models face challenges in generating realistic time series and accurately mapping metadata. A notable example is their performance with ECG signals (2nd column): GAN models only learn to generate a noisy version of the ECG samples while our TIME WEAVER model generates a pristine, realistic sample. We provide additional qualitative samples in App. A.11.

Does the synthetic data generated by TIME WEAVER match the real data in terms of density and spread of time series values? In Fig. 5 (bottom two rows), we extend our analysis to compare real and generated data distributions across all datasets. This is achieved by transforming the real and generated time series datasets into histograms of their respective values. Take, for instance, the traffic dataset: we aggregate all samples to form a histogram over the raw time series values for both real and generated datasets. The TIME WEAVER model provides a significantly more accurate representation of the real time series distribution than the best-performing GAN model. GAN models consistently fail to learn the complex underlying distributions of real data, particularly evident in the Air Quality and Traffic datasets.

Does TIME WEAVER retain the causal relationship between the input metadata and the corresponding generated time series? We analyze TIME WEAVER's ability to retain causal relationships using the Air Quality dataset. We note that there exist physical models that indicate the effect on the particulate matter level (one of the time series channels) whenever there is rainfall (metadata input). We show through qualitative examples in Fig. 11 that the generated particulate matter levels adhere to this effect. However, we note that this is not a rigorous proof of causality. We aim to build upon this work to develop time series generative models that can maintain causal relationships between the input metadata and the generated time series.

Training and Inference Complexity: Here, we discuss the training and inference complexity of our approach. For the inference complexity, we make the following observations:

- The TIME WEAVER-CSDI denoiser's inference time complexity reflects that of any transformer architecture, scaling quadratically with the time series length L and linearly with respect to the hidden dimension size H and the number of diffusion steps T.
- The TIME WEAVER-SSSD denoiser's inference time complexity reflects that of the S4 model (Alcaraz & Strodthoff, 2022), scaling quadratically with the hidden dimension size H and the number of diffusion steps T.
- Our metadata encoder's inference time complexity is the same as that of the transformer architecture, $\mathcal{O}(L^2K + K^2L)$, where K is the number of metadata conditions. As the metadata encoder only runs once for a single sample generation, its time complexity is not affected by the number of diffusion steps.

Additionally, we tabulate the exact time taken to generate a sample for all datasets in Table 11. For training complexity, we show the rate of change of J-FTSD for TIME WEAVER and GAN models. Specifically, we considered the Air Quality dataset, and the results are shown in Fig. 10. As expected, we observe that the rate of decrease of J-FTSD is quicker for TIME WEAVER when compared to GAN models.

7. Conclusion

This paper addresses a critical gap in synthetic time series data generation by introducing TIME WEAVER, a novel diffusion-based generative model. TIME WEAVER leverages heterogeneous paired metadata, encompassing categorical, continuous, and time-variant variables, to significantly improve the quality of generated time series. Moreover, we introduce a new evaluation metric, J-FTSD, to assess conditional time series generation models. This metric offers a refined approach to evaluate the specificity of generated time series relative to paired metadata conditions. Through TIME WEAVER, we demonstrate state-of-the-art results for generated sample quality and diversity across four diverse real-world datasets.

Limitations: Despite its superior performance in generating realistic time series data, TIME WEAVER encounters challenges typical of DMs, including slower inference and prolonged training durations compared to GAN-based models. Future work will focus on overcoming these limitations, potentially through techniques such as progressive distillation (Salimans & Ho, 2022) for accelerated inference. We also aim to explore the application of heterogeneous paired metadata conditions to enhance forecasting and anomaly detection within the time series domain.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

References

Agarwal, S. and Chinchali, S. P. Synthesizing adversarial visual scenarios for model-based robotic control. In *6th Annual Conference on Robot Learning*, 2022.

Alcaraz, J. L. and Strodthoff, N. Diffusion-based time series imputation and forecasting with structured state space models. *Transactions on Machine Learning Research*, 2022. ISSN 2835-8856. URL https://openreview.net/forum?id=hHiIbk7ApW.

Alcaraz, J. M. L. and Strodthoff, N. Diffusion-based conditional ecg generation with structured state space models. *Computers in Biology and Medicine*, pp. 107115, 2023.

Bahrpeyma, F., Roantree, M., Cappellari, P., Scriney, M., and McCarren, A. A methodology for validating diversity in synthetic time series generation. *MethodsX*, 8:101459, 2021. ISSN 2215-0161. doi: https://doi.org/10.1016/j.mex.2021.101459. URL https://www.sciencedirect.com/science/article/pii/S2215016121002521.

Benny, Y., Galanti, T., Benaim, S., and Wolf, L. Evaluation metrics for conditional image generation. *International Journal of Computer Vision*, 129:1712 – 1731, 2020. URL https://api.semanticscholar.org/CorpusID:216553817.

Chen, H. Challenges and corresponding solutions of generative adversarial networks (gans): A survey study. *Journal of Physics: Conference Series*, 1827(1): 012066, mar 2021. doi: 10.1088/1742-6596/1827/1/012066. URL https://dx.doi.org/10.1088/1742-6596/1827/1/012066.

Chen, N., Zhang, Y., Zen, H., Weiss, R. J., Norouzi, M., and Chan, W. Wavegrad: Estimating gradients for waveform generation. In *International Conference on Learning Representations*, 2020.

Chen, S. Beijing Multi-Site Air-Quality Data. UCI Machine Learning Repository, 2019. DOI: https://doi.org/10.24432/C5RK5G.

DeVries, T., Romero, A., Pineda, L., Taylor, G. W., and Drozdzal, M. On the evaluation of conditional GANs. *arXiv preprint arXiv:1907.08175*, 2019.

Dhariwal, P. and Nichol, A. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

Ding, X., Wang, Y., Xu, Z., Welch, W. J., and Wang, Z. J. Ccgan: Continuous conditional generative adversarial networks for image generation. In *International conference on learning representations*, 2020.

Donahue, C., McAuley, J., and Puckette, M. Adversarial audio synthesis. In *ICLR*, 2019.

Du, W., Côté, D., and Liu, Y. Saits: Self-attention-based imputation for time series. *Expert Systems with Applications*, 219:119619, June 2023. ISSN 0957-4174. doi: 10.1016/j.eswa.2023.119619. URL http://dx.doi.org/10.1016/j.eswa.2023.119619.

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

Fréchet, M. Sur la distance de deux lois de probabilité. *Annales de l'ISUP*, VI(3):183–198, 1957. URL https://hal.science/hal-04093677.

Gowal, S., Rebuffi, S.-A., Wiles, O., Stimberg, F., Calian, D. A., and Mann, T. A. Improving robustness using generated data. *Advances in Neural Information Processing Systems*, 34:4218–4233, 2021.

Gu, A., Goel, K., and Re, C. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=uYLFoz1vlAC.

He, K., Zhang, X., Ren, S., and Sun, J. Identity mappings in deep residual networks. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pp. 630–645. Springer, 2016.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 6629–6640, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Ho, J., Salimans, T., Gritsenko, A., Chan, W., Norouzi, M., and Fleet, D. J. Video diffusion models. *arXiv:2204.03458*, 2022.

Hogue, J. Metro Interstate Traffic Volume. UCI Machine Learning Repository, 2019. DOI: https://doi.org/10.24432/C5X60B.

Huang, X., Liu, M.-Y., Belongie, S., and Kautz, J. Multimodal unsupervised image-to-image translation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 172–189, 2018.

Itakura, F. Minimum prediction residual principle applied to speech recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 23(1):67–72, 1975. doi: 10.1109/TASSP.1975.1162641.

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

Jordon, J., Yoon, J., and van der Schaar, M. Pate-gan: Generating synthetic data with differential privacy guarantees. In *International Conference on Learning Representations*, 2018. URL https://api.semanticscholar.org/CorpusID:53342261.

Kong, Z., Ping, W., Huang, J., Zhao, K., and Catanzaro, B. Diffwave: A versatile diffusion model for audio synthesis. In *9th International Conference on Learning Representations, ICLR 2021*, 2021.

Li, X., Metsis, V., Wang, H., and Ngu, A. H. H. Tts-gan: A transformer-based time-series generative adversarial network. In *Conference on Artificial Intelligence in Medicine in Europe*, 2022. URL https://api.semanticscholar.org/CorpusID:246634793.

Lin, Z., Jain, A., Wang, C., Fanti, G., and Sekar, V. Using gans for sharing networked time series data: Challenges, initial promise, and open questions. In *Proceedings of the ACM Internet Measurement Conference*, IMC '20, pp. 464–483, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450381383. doi: 10.1145/3419394.3423643. URL https://doi.org/10.1145/3419394.3423643.

Liu, S., Wei, Y., Lu, J., and Zhou, J. An improved evaluation framework for generative adversarial networks. *arXiv preprint arXiv:1803.07474*, 2018.

Lugmayr, A., Danelljan, M., Romero, A., Yu, F., Timofte, R., and Gool, L. V. Repaint: Inpainting using denoising diffusion probabilistic models. In *CVPR*, pp. 11451–11461. IEEE, 2022.

Miyato, T. and Koyama, M. cgans with projection discriminator. *arXiv preprint arXiv:1802.05637*, 2018.

Murray, N. PfaGAN: An aesthetics-conditional GAN for generating photographic fine art. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pp. 0–0, 2019.

Odena, A., Olah, C., and Shlens, J. Conditional image synthesis with auxiliary classifier gans. In *International conference on machine learning*, pp. 2642–2651. PMLR, 2017.

Preuer, K., Renz, P., Unterthiner, T., Hochreiter, S., and Klambauer, G. Fréchet chemnet distance: a metric for generative models for molecules in drug discovery. *Journal of chemical information and modeling*, 58(9):1736–1741, 2018.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., and Sutskever, I. Learning transferable visual models from natural language supervision, 2021.

Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., and Chen, M. Hierarchical text-conditional image generation with clip latents, 2022.

Rizzato, M., Morizet, N., Maréchal, W., and Geissler, C. Stress testing electrical grids: Generative adversarial networks for load scenario generation. *Energy and AI*, 9:100177, 2022. ISSN 2666-5468. doi: https://doi.org/10.1016/j.egyai.2022.100177. URL https://www.sciencedirect.com/science/article/pii/S2666546822000295.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models, 2021.

Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pp. 234–241. Springer, 2015.

Saharia, C., Chan, W., Chang, H., Lee, C. A., Ho, J., Salimans, T., Fleet, D. J., and Norouzi, M. Palette: Image-to-image diffusion models. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021.

Salimans, T. and Ho, J. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=TIdIXIpzhoI.

Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pp. 2256–2265. PMLR, 2015.

Soloveitchik, M., Diskin, T., Morin, E., and Wiesel, A. Conditional frechet inception distance, 2022.

Song, Y. and Ermon, S. Generative modeling by estimating gradients of the data distribution. *Advances in Neural Information Processing Systems*, 32, 2019.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the inception architecture for computer vision. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2818–2826, 2015.

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

Thambawita, V., Isaksen, J. L., Hicks, S. A., Ghouse, J., Ahlberg, G., Linneberg, A., Grarup, N., Ellervik, C., Olesen, M. S., Hansen, T., Graff, C., Holstein-Rathlou, N.-H., Strümke, I., Hammer, H. L., Maleckar, M. M., Halvorsen, P., Riegler, M. A., and Kanters, J. K. Deepfake electrocardiograms using generative adversarial networks are the beginning of the end for privacy issues in medicine. *Scientific Reports*, 11(1):21896, 2021. doi: 10.1038/s41598-021-01295-2. URL https://doi.org/10.1038/s41598-021-01295-2.

Trindade, A. Electricity Load Diagrams. UCI Machine Learning Repository, 2015. DOI: https://doi.org/10.24432/C58C86.

Unterthiner, T., van Steenkiste, S., Kurach, K., Marinier, R., Michalski, M., and Gelly, S. Towards accurate generative models of video: A new metric & challenges. *arXiv preprint arXiv:1812.01717*, 2018.

Wagner, P., Strodthoff, N., Bousseljot, R.-D., Kreiseler, D., Lunze, F. I., Samek, W., and Schaeffter, T. Ptb-xl, a large publicly available electrocardiography dataset. *Scientific data*, 7(1):154, 2020.

Wu*, Y., Chen*, K., Zhang*, T., Hui*, Y., Berg-Kirkpatrick, T., and Dubnov, S. Large-scale contrastive language-audio pretraining with feature fusion and keyword-to-caption augmentation. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*, 2023.

Yoon, J., Jarrett, D., and van der Schaar, M. Time-series generative adversarial networks. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/c9efe5f26cd17ba6216bbe2a7d26d490-Paper.pdf.

Yoon, J., Drumright, L., and Schaar, M. Anonymization through data synthesis using generative adversarial networks (ads-gan). *IEEE Journal of Biomedical and Health Informatics*, PP:1–1, 03 2020. doi: 10.1109/JBHI.2020.2980262.

Yuan, X., Lin, Z. L., Kuen, J., Zhang, J., Wang, Y., Maire, M., Kale, A., and Faieta, B. Multimodal contrastive training for visual representation learning. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6991–7000, 2021. URL https://api.semanticscholar.org/CorpusID:233407906.

Zhang, Y., Jiang, H., Miura, Y., Manning, C. D., and Langlotz, C. P. Contrastive learning of medical visual representations from paired images and text. In Lipton, Z., Ranganath, R., Sendak, M., Sjoding, M., and Yeung, S. (eds.), *Proceedings of the 7th Machine Learning for Healthcare Conference*, volume 182 of *Proceedings of Machine Learning Research*, pp. 2–25. PMLR, 05–06 Aug 2022. URL https://proceedings.mlr.press/v182/zhang22a.html.

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., and Zhang, W. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 11106–11115, 2021.

A. Appendix

A.1. Diffusion Process

Diffusion Models (DMs) are trained to denoise a noisy sample, referred to as the reverse or the backward process p_θ. The noisy samples are generated by a Markovian forward process q, where we gradually corrupt a clean sample from the dataset by adding noise for T diffusion steps. The forward process is predetermined by specifying a noise schedule $\{\beta_1, \ldots, \beta_T\}$, where $\beta_t \in [0, 1]$. The following equations parameterize the forward process:

$$q(x_1, \ldots, x_T \mid x_0) = \prod_{t=1}^{T} q(x_t \mid x_{t-1}), \tag{7}$$

$$q(x_t \mid x_{t-1}) = \mathcal{N}(\sqrt{1-\beta_t} x_{t-1}, \beta_t \mathbf{I}), \tag{8}$$

where $\mathcal{X}$ is the data distribution that we want to learn, $x_0 \sim \mathcal{X}$, $\mathcal{N}(\mu, \Sigma)$ represents a Gaussian distribution with mean μ and covariance matrix Σ, and T is the number of diffusion steps. The noise schedule $\{\beta_1, \ldots, \beta_T\}$ and T are chosen such that x_T resembles samples from a Gaussian distribution with zero mean and unit variance, *i.e.*, $q(x_T) \simeq \mathcal{N}(\mathbf{0}, \mathbf{I})$. This allows us to start the backward process from $x_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and iteratively denoise for T steps to obtain a sample from $\mathcal{X}$. The reverse process is parameterized as follows:

$$p_\theta(x_0, \ldots, x_{T-1} \mid x_T) = p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1} \mid x_t). \tag{9}$$

Here, $p(x_T) = \mathcal{N}(\mathbf{0}, \mathbf{I})$. Essentially, the reverse process is learnable, and $p_\theta(x_{t-1} \mid x_t)$ approximates $q(x_{t-1} \mid x_t, x_0)$. $p_\theta(x_{t-1} \mid x_t)$ is parameterized using a neural network, θ_{denoiser}. Ho et al. (2020) show that through simple reparametrization tricks, we can convert the learning objective from approximating $q(x_{t-1} \mid x_t, x_0)$ to estimating the amount of noise added to go from x_{t-1} to x_t. Thus, the diffusion objective is stated as minimizing the following loss function:

$$\mathcal{L}_{\text{DM}} = \mathbb{E}_{x \sim \mathcal{X}, \epsilon \sim \mathcal{N}(\mathbf{0},\mathbf{I}), t \sim \mathcal{U}(1,T)} \left[\|\epsilon - \theta_{\text{denoiser}}(x_t, t)\|_2^2 \right], \tag{10}$$

where $t \sim \mathcal{U}(1, T)$ indicates that t is sampled from a uniform distribution between 1 and T, ϵ is the noise added to x_{t-1} to obtain x_t, and θ_{denoiser} takes the noisy sample x_t and the diffusion step t as input to estimate ϵ. This is equivalent to the score-matching techniques (Song & Ermon, 2019; Song et al., 2021).

A.2. Dataset Description

In this section, we describe in detail the various datasets used in our experiments, their training, validation, and testing splits, and the normalization procedures.

A.2.1. ELECTRICITY DATASET

The Electricity dataset consists of power consumption recordings for 370 users over four years from 2011 to 2015. We frame the following task for this dataset - "Generate the electricity demand pattern for user 257, for the 3rd of August 2011," which is a univariate time series. We consider the following features as the metadata: 370 users, four years, 12 months, and 31 dates (Table 3). The power consumption is recorded every 15 minutes, so the time series is 96 time steps long. The total number of samples without any preprocessing is 540200. We remove samples with values of 0 for the entire time series, and the resulting total number of samples is 434781. We establish a data split comprising training, validation, and test sets distributed in an 80-10-10 ratio. To obtain the split, we randomly pick 80% of the 434781 samples and assign them to the training set. The same is repeated for the validation and the test sets. We avoid using the traditional splits proposed by Du et al. (2023), as their split creates certain year metadata features in the test set that do not exist in the training set. For example, no month from 2011 exists in the training set split proposed by Du et al. (2023).

A.2.2. TRAFFIC DATASET

For traffic volume synthesis, we use the metro interstate traffic volume dataset. The dataset has hourly traffic volume recordings from 2012 to 2018, along with metadata annotations like holidays, textual weather descriptions, and weather forecasts (Table 3). Here, we want to generate time series samples for prompts such as "Given the following weather

Dataset	Horizon	# Channels	Categorical features	Continuous features
Air Quality	96	6	12 Stations, 5 years, 12 months, 31 dates, 24 hours, 17 wind directions	Temperature, Pressure, Dew point temperature, Rain levels, Wind speed
Traffic	96	1	12 holidays, 7 years, 12 months, 31 dates, 24 hours, 11 broad weather descriptions, 38 fine weather descriptions	Temperature, Rain levels, Snowfall levels, Cloud conditions
Electricity	96	1	370 users, 4 years, 12 months, 31 dates	N.A.
ECG	1000	8	71 heart disease statements	N.A.

Table 3. **Dataset overview for experiments with Time Weaver.** This table outlines the key characteristics of the datasets employed in our experiments. These datasets, encompassing Air Quality, Traffic, Electricity, and ECG, have been carefully selected to demonstrate Time Weaver's versatility across different time horizons, number of channels, and a wide range of metadata types.

forecast, synthesize a traffic volume pattern for New Year's Day.", which is a univariate time series. The dataset CSV file has 48204 rows containing the traffic volume. We synthesize the traffic volume for a 96-hour window. So, to create a dataset from the CSV file, we slide a window of length 96 with a stride of 24. This gives a total of 2001 time series samples, which we randomly divide into train, validation, and test sets with an 80-10-10 ratio.

A.2.3. Air Quality Dataset

This dataset contains hourly air pollutants data from 12 air quality monitoring stations in Beijing. The meteorological data in each air quality site are paired with the weather data from the nearest weather station (please refer to Table 3 for more details regarding the metadata conditions). Here, the task is to synthesize a multivariate (6 channels) time series given the weather forecast metadata. The dataset has missing values, which we replace with the mean for both continuous metadata and the time series. For categorical metadata, missing values exist only in the wind direction metadata feature, which we fill using an "unknown" label. The dataset is split into train, validation, and test sets based on months. The recordings are available from 2013 to 2017, and we have a total of 576 months, of which we randomly picked 460 as train data, 58 as validation data, and 58 as test data. For each month, we slide a window of length 96 with a stride of 24, and this provides a total of 12166 train time series samples, 1537 validation time series samples, and 1525 test time series samples.

A.2.4. ECG Dataset

The PTB-XL ECG dataset is a 12-channel (1000 time steps long) time series dataset with 17651 train, 2203 validation, and 2167 test samples. The dataset has annotated heart disease statements for each ECG time series. Here, the goal is to generate ECG time series samples for a specific heart disease statement, which is our metadata. In this work, we use 8 channels instead of 12, as shown in (Alcaraz & Strodthoff, 2023).

A.3. Architecture Description Of The Feature Extractors Used In J-FTSD

To compute our proposed J-FTSD metric, we relied on the Informer encoder architecture proposed by Zhou et al. (2021). Specifically, we used two encoders, one for the time series and one for the metadata features, represented as ϕ_{time} and ϕ_{meta}, respectively. We made the following modifications to the Informer encoder architecture:

- The raw time series was first processed using 1D convolution layers. We added positional encoding to the processed time series before providing it as input to the self-attention layers in ϕ_{time}. We used the same positional encoding as suggested by Zhou et al. (2021).

- The raw metadata was processed in the same way as the metadata for the time series generation process, which is highlighted in Sec. 4. We individually processed or tokenized the categorical and continuous metadata using linear layers and 1D convolution layers to obtain the metadata embedding z. We added positional encoding to z before providing z as input to the self-attention layers in ϕ_{meta}.

- We used 1D convolution layers at the end of every self-attention layer. We used striding after every three self-attention layers, *i.e.*, 1D convolution layers with a stride of 2 were applied after the 3^{rd}, 6^{th}, . . ., self-attention layers.

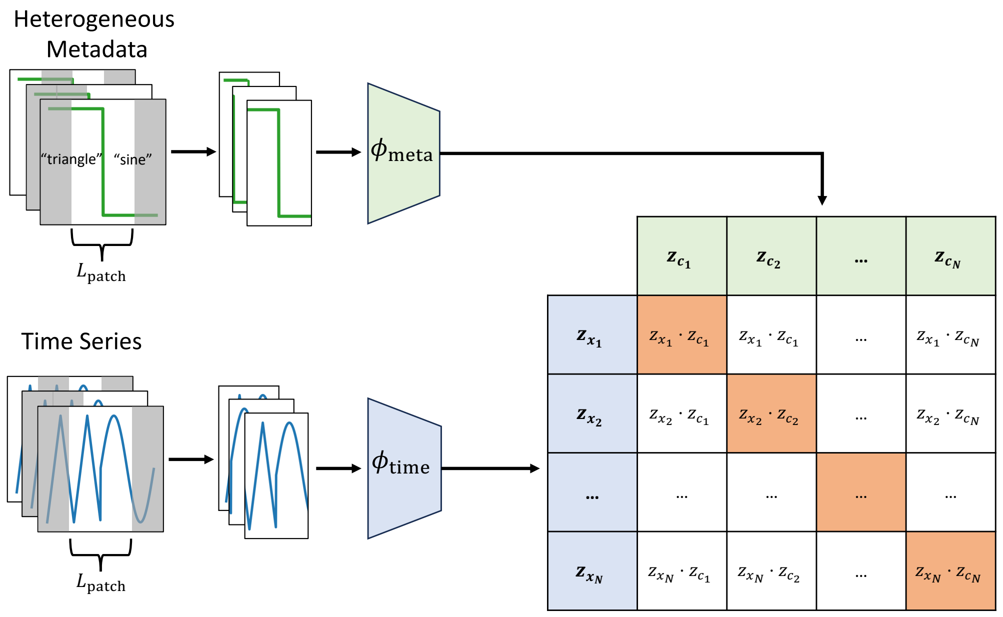

Figure 6. **Contrastive Training of J-FTSD Feature Extractors Inspired By CLIP (Radford et al., 2021):** This figure depicts the contrastive learning-based training approach for the J-FTSD feature extractors ϕ_{time} and ϕ_{meta}, akin to the methodology used in CLIP. Here, we consider a time series where the first half is a triangle wave and the second half is a sine wave. The categorical metadata corresponding to this time series has the first half labeled as 1 ("triangle") and the second half as 0 ("sine"). Patches of length L_{patch} are extracted from time series and metadata and processed through their respective feature extractors. The embeddings, z_c from the metadata encoder and z_x from the time series encoder, are compared using their dot products to identify correct pairings, highlighted along the matrix diagonal (in orange). The feature extractors are trained through contrastive learning, employing cross-entropy loss to enhance the accuracy of matching time series data with its relevant metadata, effectively capturing the nuanced relationship between the two.

- At the end of the self-attention layers of both ϕ_{time} and ϕ_{meta}, we flattened the outputs and projected the outputs to a lower-dimensional space using linear layers. We used the Gaussian Error Linear Unit (GELU) activation, similar to the Informer architecture.

Now, we describe the choice of L_{patch} for each dataset. As explained in Sec. 5, we chose L_{patch} based on the minimum horizon required for a patch to contain metadata-specific features. For each dataset, the values of L_{patch} and the embedding size (the output dimension of the feature extractors) are tabulated in Table 4.

DATASET	L_{patch}	EMBEDDING SIZE
AIR QUALITY	64	128
ECG	256	256
ELECTRICITY	64	48
TRAFFIC	64	48

Table 4. **Choice of the patch and embedding sizes.** In this table, we list the choices of the patch size L_{patch} and the embedding size used for each dataset in our experiments.

Specifically, we chose the embedding size such that given a time series sample $x \in \mathbb{R}^{L \times F}$, where L is the horizon, and F is the number of channels in the time series, the embedding size should be smaller than $F \times L_{\text{patch}}$. This is to ensure that we are reducing the dimensionality of the time series patch.

Now, we list the hyperparameter choices used for training the feature extractors in Table 5. These include the number of patches from a single time series sample N_{patch}, learning rate, etc, and the design choices in terms of the number of self-attention layers, number of transformer heads, etc.

Fig. 6 provides a pictorial representation of the contrastive learning framework used to train the feature extractors ϕ_{time} and

Design Parameter	Value
Embedding size (dmodel)	128
Attention heads (nheads)	8
Self-attention layers	8
dropout	0.05
Activation	GELU
N_{patch}	2
Learning Rate	10^{-4}

Table 5. **Hyperparameters for the feature extractors.**

ϕ_{meta}. The training process ensures that the time series samples with similar paired metadata have similar projections in the latent space.

A.4. Time Weaver Architecture Design

As mentioned in Sec. 4, Time Weaver has two denoiser backbones - CSDI (Tashiro et al., 2021) and SSSD (Alcaraz & Strodthoff, 2023). In this section, we describe the architecture changes we introduced to the CSDI and the SSSD backbones to extend their capabilities to metadata-specific time series generation.

A.4.1. Time Weaver-CSDI

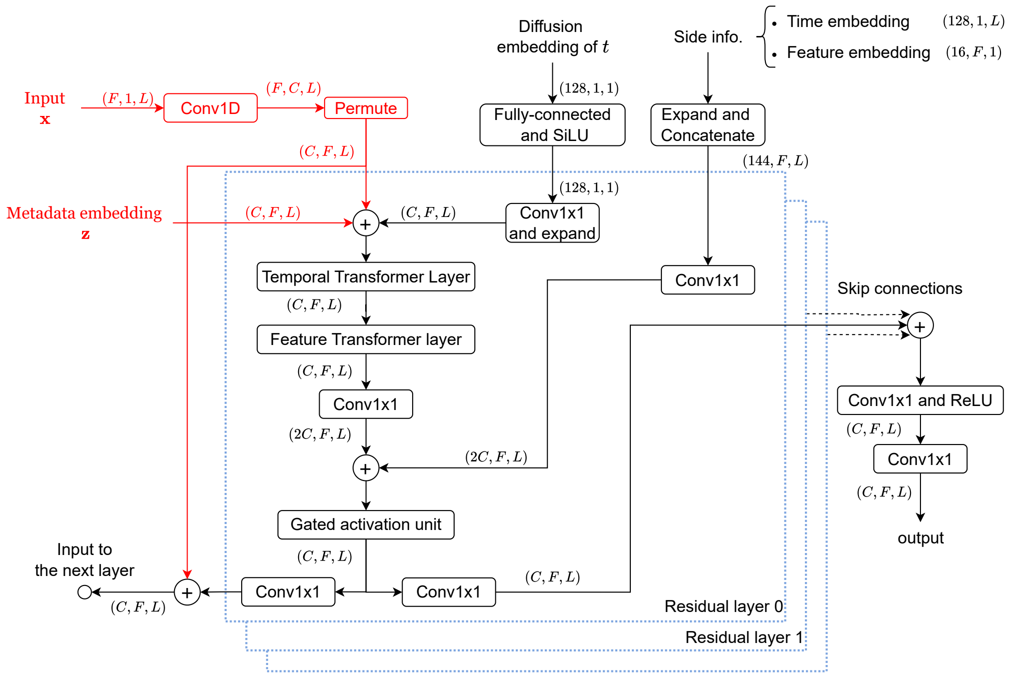

Figure 7. **Time Weaver-CSDI architecture:** This figure shows our changes to the original conditional CSDI model (Tashiro et al., 2021). We use this model as our denoiser (θ_{denoiser}) in our architecture, with metadata preprocessing fixed as in Fig. 3. Changes to the original architecture are colored in red.

Consider a batch of time series samples of size (N_{batch}, F, L), where N_{batch} represents the number of samples per batch, F represents the number of channels in the time series, and L represents the horizon. The paired metadata is represented as $c_{\text{cat}} \oplus c_{\text{cont}}$, where the shape of c_{cat} is $(N_{\text{batch}}, L, K_{\text{cat}})$ and the shape of c_{cont} is $(N_{\text{batch}}, L, K_{\text{cont}})$.

- *Input time series projection:* We first transformed the input time series batch to $(N_{\text{batch}} \times F, 1, L)$ and applied 1D

Design Parameter	Value
Position embedding	128
Feature or Channel embedding	16
Diffusion step embedding	256
Embedding size (d_{meta})	256
Attention heads (`nheads`)	16
Metadata encoder (θ_{condn}) embedding size	256
Metadata encoder (θ_{condn}) attention heads	8
Metadata encoder (θ_{condn}) Self-attention layers	2
Learning rate	10^{-4}

Table 6. **Hyperparameters for Time Weaver-CSDI architecture.**

convolution layers with d_{meta} filters to obtain a projection of shape $(N_{\text{batch}} \times F, d_{\text{meta}}, L)$. We then reshaped the projection from $(N_{\text{batch}} \times F, d_{\text{meta}}, L)$ to $(N_{\text{batch}}, d_{\text{meta}}, F, L)$.

- *Metadata projection:* Simultaneously, we converted each categorical metadata feature in c_{cat} into one-hot encoding and further processed using $\theta^{\text{cat}}_{\text{token}}$. Similarly, we processed the continuous metadata, c_{cont}, using $\theta^{\text{cont}}_{\text{token}}$. Both c_{cat} and c_{cont} were projected to latent representations of shape $(N_{\text{batch}}, L, d_{\text{cat}})$ and $(N_{\text{batch}}, L, d_{\text{cont}})$, respectively. These latent representations were concatenated along the final axis and processed using the self-attention layer θ_{condn}. At the end of this preprocessing, the categorical and continuous metadata were projected to a latent representation of shape $(N_{\text{batch}}, L, d_{\text{meta}})$. We then broadcasted and reshaped the projected metadata to $(N_{\text{batch}}, d_{\text{meta}}, F, L)$.

- *Diffusion step representation:* The CSDI architecture represents the diffusion step using a 128-dimensional representation, which is projected to d_{meta}. We broadcasted and reshaped the diffusion step representation to $(N_{\text{batch}}, d_{\text{meta}}, F, L)$.

- Then, we added the input time series projection, metadata projection, and diffusion step representation. This sum was passed to the temporal and feature transformer layers in the first residual layer.

- We provided the projected metadata as input to all the residual layers in the same manner.

These modifications are highlighted in red in Fig. 7. For the diffusion process, our experiments with Time Weaver-CSDI used 200 diffusion steps with the noise variance schedule values of $\beta_1 = 0.0001$ and $\beta_T = 0.1$.

Now, we list the architectural choices and the corresponding hyperparameter choices in Table 6. The number of residual layers used varies for each dataset. For the Air Quality dataset, we used 10 residual layers. Similarly, for the Traffic, Electricity, and ECG datasets, we used 8, 6, and 10 residual layers respectively.

A.4.2. Time Weaver-SSSD

The Time Weaver-SSSD model is based on the structured state-space diffusion (SSSD) model (Alcaraz & Strodthoff, 2022) which was originally designed for imputation tasks. The SSSD model is built on the DiffWave (Kong et al., 2021) architecture. Unlike the DiffWave model, SSSD utilizes the structured state-space model (SSM) (Gu et al., 2022), which connects the input sequence $u(t)$ to the output sequence $y(t)$ via the hidden state $x(t)$. This relation can be explicitly given as:

$$x'(t) = Ax(t) + Bu(t) \quad \text{and} \quad y(t) = Cx(t) + Du(t).$$

Here, A, B, C, and D are learnable transition matrices. Gu et al. (2022) propose stacking several SSM blocks together to create a Structured State Space sequence model (S4). These SSM blocks are connected with normalization layers and point-wise FC layers in a way that resembles the transformer architecture. This architectural change is done to capture long-term dependencies in time series data. Alcaraz & Strodthoff (2023) adjust this architecture to take label input, a binary vector of length 71. As shown in Fig. 8, we replaced this label input with the metadata embeddings obtained from our metadata preprocessing block to incorporate more complex metadata modalities.

For the diffusion process, our experiments with Time Weaver-SSSD used 200 diffusion steps with the noise variance schedule values of $\beta_1 = 0.0001$ and $\beta_T = 0.02$.

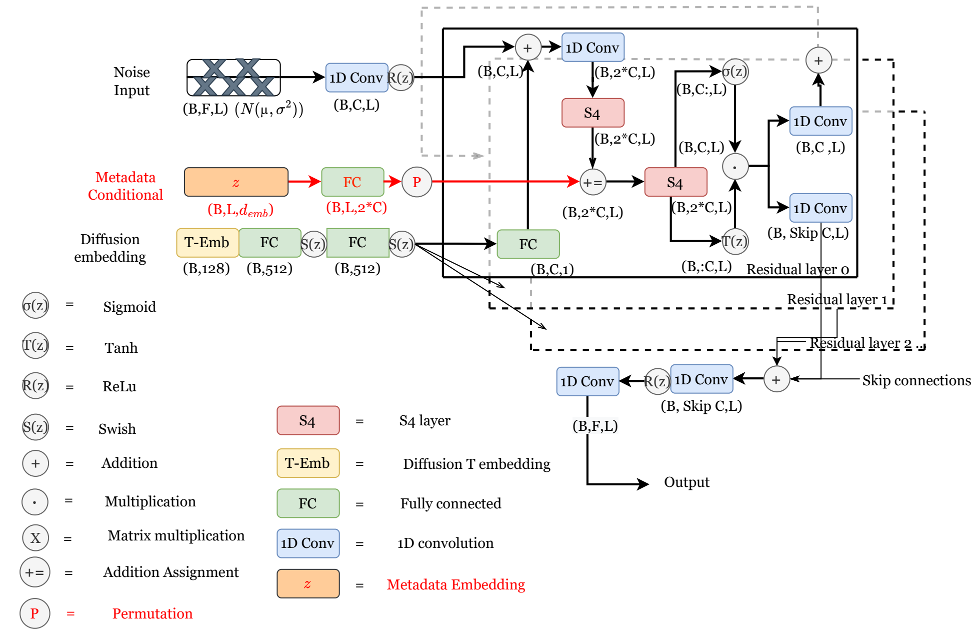

Figure 8. **TIME WEAVER-SSSD architecture:** This figure shows our changes to the original conditional SSSD model (Alcaraz & Strodthoff, 2023). We use this model as the denoiser (θ_{denoiser}) in our architecture, with metadata preprocessing being fixed as in Fig. 3. Changes to the original architecture are highlighted in red.

Now, we provide the list of design choices and hyperparameters used in the TIME WEAVER-SSSD model in Table 7.

DESIGN PARAMETER	VALUE
RESIDUAL LAYER CHANNELS	256
SKIP CHANNELS	16
DIFFUSION STEP EMBEDDING INPUT CHANNELS	128
DIFFUSION STEP EMBEDDING MID CHANNELS	512
DIFFUSION STEP EMBEDDING OUTPUT CHANNELS	512
S4 LAYER STATE DIMENSION	64
S4 LAYER DROPOUT	0.0
IS S4 LAYER BIDIRECTIONAL	TRUE
USE LAYER NORMALIZATION	TRUE
METADATA ENCODER (θ_{condn}) EMBEDDING SIZE	256
METADATA ENCODER (θ_{condn}) ATTENTION HEADS	8
METADATA ENCODER (θ_{condn}) SELF-ATTENTION LAYERS	2
LEARNING RATE	10^{-4}

Table 7. **Hyperparameters for the TIME WEAVER-SSSD architecture.**

A.5. GAN Baselines

A.5.1. MAIN GAN BASELINES

For our main GAN baselines, we use Pulse2Pulse GAN (Thambawita et al., 2021) and WaveGAN (Donahue et al., 2019). Since these approaches are not fundamentally conditional, we added additional layers to enable conditional generation.

- For the Electricity and ECG datasets, we used the implementation provided by (Thambawita et al., 2021) and (Alcaraz

& Strodthoff, 2023). Since these datasets only have categorical metadata, we represented each categorical label by a fixed embedding. This fixed embedding is added to the output of each layer in the generator after the batch normalization layers. Similarly, we add the fixed embedding to the output of each layer in the discriminator. To learn the conditional distribution, along with predicting whether a sample is real or fake, we also predict the logit of each categorical metadata, similar to (Odena et al., 2017). In our training experiments, we noticed that predicting the metadata category for the fake sample results in poor-quality samples. Hence, during training, we only predict the category for the real samples.

- For the Air Quality and Traffic datasets, we appended the inputs to the generator and discriminator with the metadata conditions.

For all the datasets except the Air Quality dataset, we used min-max normalization to transform the time series samples to lie between -1 and 1. For the Air Quality dataset, we used the standard zero mean, unit variance normalization.

A.5.2. WaveGAN Implementation Details

We train the WaveGAN model for all the datasets with a learning rate of 10^{-4} and store the checkpoints after every 100 epochs. We sample a 48-dimensional random vector for the Electricity, Air Quality, and Traffic datasets and a 100-dimensional random vector for the ECG dataset. This vector serves as the noise input to the generator. We used the `PyTorch` implementation (Link to the repo) and the code from (Alcaraz & Strodthoff, 2023) to implement WaveGAN. We adjusted the number of parameters in the generator and discriminator to roughly match the size of our Time Weaver models.

- For the Air Quality dataset, the total number of trainable parameters in the GAN model is 15.2 million, and the generator has 8.51 million trainable parameters.
- For the Traffic dataset, the total number of trainable parameters in the GAN model is 13.7 million, and the generator has 7.017 million trainable parameters.
- For the Electricity dataset, the total number of trainable parameters in the GAN model is 13.3 million, and the generator had 7.17 million parameters.
- For the ECG dataset, the total number of trainable parameters in the GAN model is 40.9 million, and the generator has 21.36 million parameters.

We chose the checkpoints that provided lower values of J-FTSD on the test data set.

A.5.3. Pulse2Pulse GAN Implementation Details

We train the Pulse2Pulse GAN model in the same manner as WaveGAN for all the datasets, with a learning rate of 10^{-4}, and store the checkpoints after every 100 epochs. Here, the noise input to the generator has the same dimensions as the time series sample that we want to generate. We adjusted the number of parameters in the generator and discriminator to roughly match the size of our Time Weaver models.

- For the Air Quality and Traffic datasets, the total number of trainable parameters in the GAN model is 14.1 million, and the generator has 7.45 million trainable parameters.
- For the Electricity dataset, the total number of trainable parameters in the GAN model is 16.9 million, and the generator has 8.4 million parameters.
- For the ECG dataset, the total number of trainable parameters in the GAN model is 43 million, and the generator has 23.47 million parameters.

A.5.4. Additional GAN Baselines

In addition to WaveGAN (Donahue et al., 2019) and Pulse2Pulse GAN (Thambawita et al., 2021) models, we experimented with TTS-GAN (Li et al., 2022) and the well-established TimeGAN (Yoon et al., 2019) models. Unfortunately, we were unable to train these models to generate quality samples. These models were likely challenged by higher input time series lengths than their original implementation. For example, TimeGAN and TTS-GAN consider time steps up to 24 and 188, respectively, while we consider time steps up to 1000. A similar problem was also faced by Alcaraz & Strodthoff (2023). We include our training examples after 10000 epochs for the Air Quality and Traffic datasets in Fig. 9.

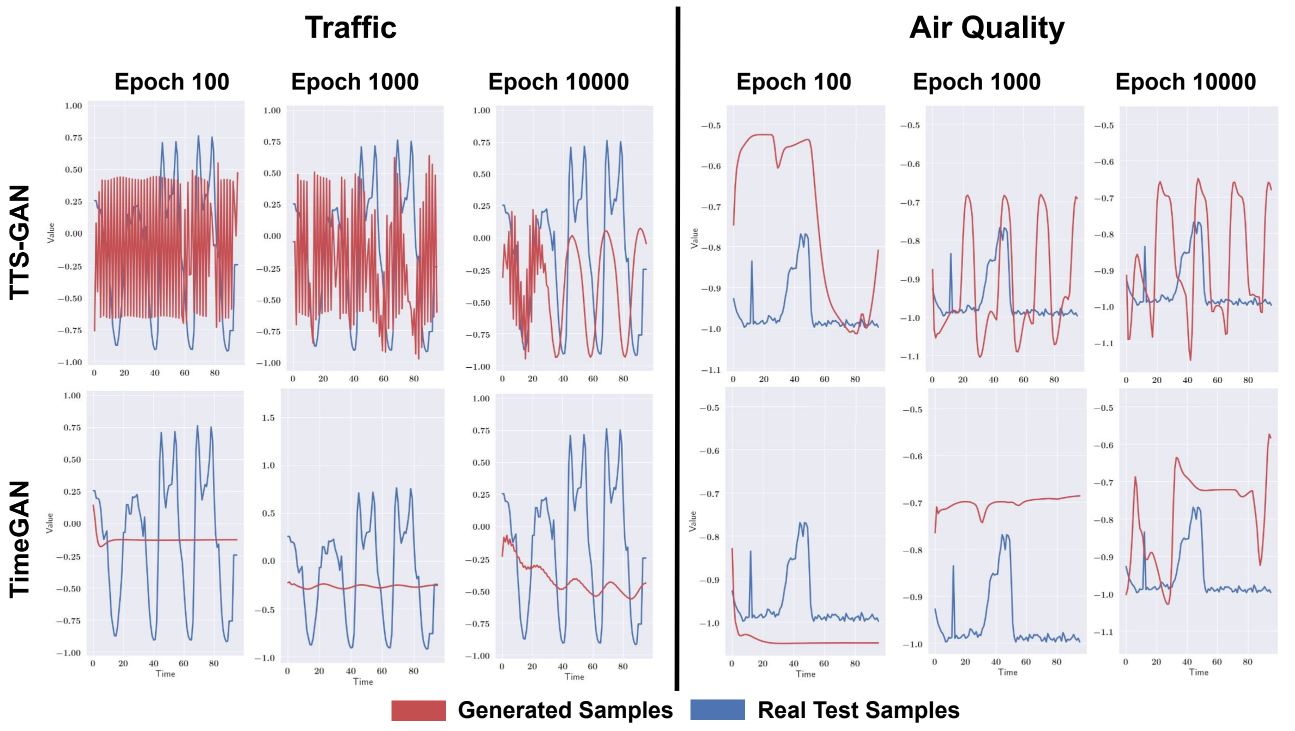

Figure 9. **TimeGAN and TTS-GAN failed to generate realistic samples after 10000 epochs.** This figure shows the samples generated from the test dataset after 100, 1000, and 10000 training epochs, where *row 1* and *row 2* correspond to TTS-GAN and TimeGAN, respectively. We can see that both models fail to generate high-quality realistic samples.

A.6. Evaluation Metrics

In this section, we briefly describe the details regarding the evaluation metrics, i.e., TSTR (train on synthetic test on real) and J-FTSD.

A.6.1. J-FTSD Details

For the Electricity, Air Quality, and Traffic datasets, the horizon is 96, *i.e.*, $L = 96$. So, we consider time series and metadata patches of length $L_{\text{patch}} = 64$. Consequently, we obtain the time series and the metadata embeddings for these patches using ϕ_{time} and ϕ_{meta}, respectively. We compute the J-FTSD metric from these embeddings using Eq. (6). For the ECG dataset, since the horizon is 1000, L_{patch} is set to 256.

One important detail is that the feature extractors, ϕ_{time}, and ϕ_{meta}, are trained on the entire dataset. This was done to ensure that the feature extractors could effectively obtain the metadata-specific features, improving the evaluation process. For example, if the feature extractors were trained only on the train and validation sets, they would not be able to detect unique trends in the test set.

A.6.2. Train on Synthetic Test on Real (TSTR) Metric Description

For TSTR, we use a standard ResNet 1D (He et al., 2016) architecture. We pick the most physically relevant categorical metadata category for the classification task for each dataset. To this end, we performed the following classification tasks:

- Classification over 12 months in the Electricity dataset. We trained the classifier with cross-entropy loss for 30 epochs with a learning rate of 10^{-4}.
- Classification over 71 heart disease statements in the ECG dataset. We note that for a given time series sample, more than one class could be active. So, we trained a classifier with binary cross-entropy loss for 200 epochs with a learning rate of 10^{-4}.
- Classification over 11 coarse weather descriptions in the Traffic dataset. We treat the classification task here as a multi-class, multi-label classification problem. So, we trained a classifier with binary cross-entropy loss for 250 epochs with a learning rate of 10^{-4}.
- Classification over 12 weather stations for the Air Quality dataset. We used the cross-entropy loss for 500 epochs with a learning rate of 10^{-4}.

Here, we note that with the pre-trained diffusion model, we generated the synthetic train, validation, and test datasets. The classifier is trained on the synthetic train dataset, and the checkpoints are stored with the synthetic validation dataset. We finally evaluated the model on the real test dataset. We use the Area Under the Receiver Operating Characteristic Curve (ROC AUC) as the TSTR metric.

A.7. Additional Quantitative Results

This section provides additional quantitative results and ablation studies for TIME WEAVER and J-FTSD.

A.7.1. PERFORMANCE COMPARISON BETWEEN TIME WEAVER AND GAN MODELS IN THE PRESENCE AND ABSENCE OF METADATA

We perform an ablation study to compare the performance of TIME WEAVER and GAN in the presence and absence of metadata for the Air Quality and Electricity datasets. We observe the FTSD metric (Frechet Distance without the metadata embedding) for TIME WEAVER-CSDI and WaveGAN (Donahue et al., 2019). The FTSD metric measures the real and generated time series datasets' distributional similarity. First, we note that in the absence of metadata, WaveGAN provides lower FTSD for the Air Quality dataset, and TIME WEAVER-CSDI provides lower FTSD for the Electricity dataset. However, in the presence of metadata, TIME WEAVER-CSDI outperforms WaveGAN in both datasets. The quantitative results are tabulated in Table 8. This result aligns with our key insight that GANs perform poorly with continuous conditions. However, diffusion-based approaches such as TIME WEAVER can handle any arbitrary combination of categorical, continuous, and time-varying metadata conditions.

APPROACH	AIR QUALITY		ELECTRICITY	
	WITHOUT METADATA	WITH METADATA	WITHOUT METADATA	WITH METADATA
WAVEGAN (DONAHUE ET AL., 2019)	**1.40±0.045**	2.62±0.001	5.26±0.025	0.92±0.005
TIME WEAVER-CSDI	2.98±0.097	**0.51±0.016**	**0.57±0.019**	**0.29±0.001**

Table 8. **TIME WEAVER outperforms GANs for metadata-specific time series generation for complex metadata.** Here, we show a quantitative comparison between the performance of TIME WEAVER and GAN models in the presence and absence of metadata. We use the FTSD metric (Frechet Distance without metadata embeddings) for this comparison. We evaluate TIME WEAVER-CSDI and WaveGAN (Donahue et al., 2019) on the Air Quality and Electricity datasets. The key observation is that in the presence of metadata, TIME WEAVER-CSDI significantly outperforms WaveGAN (Donahue et al., 2019) on the FTSD metric. This indicates TIME WEAVER's ability to handle any arbitrary combination of categorical, continuous, and time-varying metadata.

A.7.2. PERFORMANCE COMPARISON BETWEEN TIME WEAVER AND DIFFUSION MODEL WITH U-NET 1D DENOISER

In addition to GANs, we compare the performance of TIME WEAVER against diffusion models with U-Net 1D (Ronneberger et al., 2015) as the denoiser backbone. We choose U-Net 1D as U-Net is the most commonly used denoiser in image and video synthesis. This comparison aims to highlight the requirement of time series-specific modifications to the denoiser to obtain high-quality samples. Table 9 shows the quantitative comparison between U-Net 1D and the best-performing TIME WEAVER model for all datasets. We choose the best-performing TIME WEAVER model using the TSTR metric in Table 2.

		TSTR (↑)		
APPROACH	AIR QUALITY	ECG	TRAFFIC	ELECTRICITY
U-NET 1D (RONNEBERGER ET AL., 2015)	0.66±0.01	0.71±0.01	0.65±0.02	**0.78±0.003**
TIME WEAVER	**0.77±0.01**	**0.85±0.007**	**0.66±0.06**	**0.78±0.001**
		J-FTSD (↓)		
	AIR QUALITY	ECG	TRAFFIC	ELECTRICITY
U-NET 1D (RONNEBERGER ET AL., 2015)	7.32±0.04	12.47±0.07	**0.19±0.01**	0.64±0.003
TIME WEAVER	**2.2±0.07**	**5.43±0.1**	0.53±0.01	**0.6±0.003**

Table 9. **TIME WEAVER outperforms the diffusion model with a U-Net 1D denoiser backbone.** Here, we show quantitative results for the comparison between the best-performing TIME WEAVER model and the diffusion model baseline with U-Net 1D as the denoiser backbone. Overall, we observe that TIME WEAVER significantly outperforms the U-Net 1D baseline by around 9% on the TSTR metric and around 17% on the J-FTSD metric on average. This superior performance highlights the necessity for time series-specific changes to the denoiser backbone in diffusion models. For example, TIME WEAVER-CSDI contains feature and temporal transformer layers that can learn the required correlation between the input metadata and the generated time series channels, as well as the correlation between different time stamps. Meanwhile, U-Net, a popular denoiser used for image and video generation, cannot effectively capture the temporal and channel-wise correlation between the metadata and time series.

APPROACH	AIR QUALITY	TRAFFIC	ELECTRICITY
WAVEGAN (DONAHUE ET AL., 2019)	4.56±1.64	4.00±0.66	3.79±4.55
PULSE2PULSE (THAMBAWITA ET AL., 2021)	5.45±2.14	4.86±1.16	3.54±3.63
TIME WEAVER-CSDI	**2.50±1.12**	1.34±0.99	**2.05±1.64**
TIME WEAVER-SSSD	3.88±2.20	**1.10±1.00**	2.61±2.36

Table 10. **Our proposed J-FTSD metric correlates with the Dynamic Time Warping metric (Itakura, 1975).** This table shows the Dynamic Time Warping (DTW) metric computed between the real and generated time series samples for the Air Quality, Electricity, and Traffic datasets. Note that for the DTW metric computed between real and generated time series samples, lower values indicate higher similarity between the two samples. Comparing with the J-FTSD values in Table 2, we note that lower values of J-FTSD correspond to lower values of DTW. This further affirms the validity of our proposed J-FTSD metric. Additionally, note that our proposed TIME WEAVER models provide the lowest DTW for all the datasets.

Overall, we observe that TIME WEAVER significantly outperforms the U-Net 1D baseline by around 9% on the TSTR metric on average. Similarly, TIME WEAVER outperforms the U-Net 1D baseline by around 17% on the J-FTSD metric.

A.7.3. CORRELATION BETWEEN OUR PROPOSED J-FTSD METRIC AND THE DYNAMIC TIME WARPING METRIC

Dynamic Time Warping (DTW) (Itakura, 1975) is a similarity metric between two different time series samples. Dynamic Time Warping looks for the temporal alignment that minimizes the Euclidean distance between two aligned time series samples. Lower values of DTW indicate high similarity between the two time series samples. As a metric to evaluate metadata-specific time series generation models, DTW works effectively when there is only one time series per metadata condition. In this case, we can compare the single real time series with the generated time series for the same metadata to check if the metadata-specific features are retained. If there are multiple time series per metadata condition (which is the case in the ECG dataset), DTW might not be the most effective solution. In our experiments, we note that there is only one time series sample per metadata condition in the Air Quality, Electricity, and Traffic datasets. For these datasets, we observe that, among different approaches used in our experiments, lower values of J-FTSD correspond to lower values of DTW. The quantitative results are provided in Table 10. This further affirms the validity of our proposed J-FTSD metric.

A.8. Training and Inference Complexity

In this section, we provide a quantitative analysis of the training and inference complexities of our proposed TIME WEAVER models. For training complexity, we show how the J-FTSD metric varies over epochs during training in Fig. 10. We note that TIME WEAVER provides a faster rate of decrease of J-FTSD when compared to GANs.

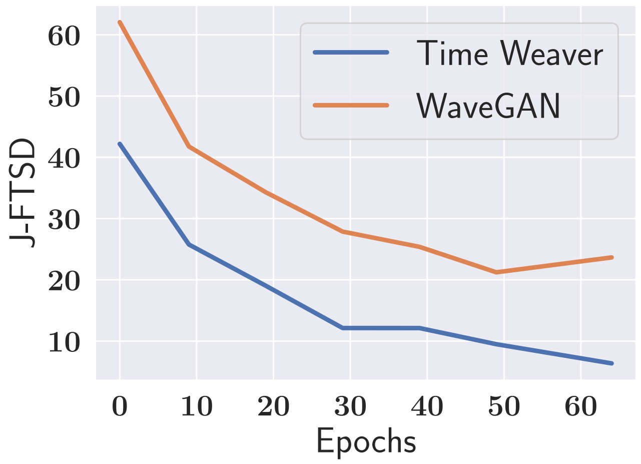

Figure 10. **TIME WEAVER provides a better rate of decrease of J-FTSD during training than GANs.** Here, we show how J-FTSD changes over epochs during the training phase for the Air Quality dataset. For this experiment, we compare TIME WEAVER-CSDI against WaveGAN. We observe that TIME WEAVER-CSDI provides a much higher rate of decrease of J-FTSD when compared to WaveGAN, indicating better training.

For the inference complexity, we provide a theoretical discussion on the time complexity of TIME WEAVER in Sec. 6. Additionally, we provide the measured inference latency for TIME WEAVER variants in Table 11. From Table 11, it can be seen that the inference latency for TIME WEAVER-CSDI is much lower than the inference latency for TIME WEAVER-SSSD for generating a single sample. However, we note that TIME WEAVER-SSSD is more suitable for batched generation as TIME WEAVER-CSDI drastically limits the maximum allowable batch size due to the forward pass through its feature transformer layers.

	INFERENCE LATENCY (S)			
	AIR QUALITY	ECG	TRAFFIC	ELECTRICITY
TIME WEAVER-CSDI	3.42±0.303	10.90±0.236	2.63±0.288	1.68±0.271
TIME WEAVER-SSSD	6.15±0.302	64.30±0.085	5.07±0.299	3.93±0.299

Table 11. **Inference latency of TIME WEAVER models for generating one sample.** In this table, we show the inference latency in seconds for our proposed TIME WEAVER models for all datasets. We compute the mean inference latency and the standard deviation for generating one sample over ten runs. The inference experiments were performed on a single NVIDIA RTX A5000 GPU.

A.9. Discussion On The Causal Relationship Between Time Series And Metadata

In this section, we present an interesting observation relevant to TIME WEAVER's ability to retain the causal effects of the input metadata on the generated time series. We consider the Air Quality dataset, where physical models exist that indicate the effect of rainfall (metadata) on the particulate matter levels (one of the time series channels). We show that TIME WEAVER generates particulate matter levels that adhere to this effect through qualitative examples in Fig. 11. However, we note that this is not rigorous proof of causality.

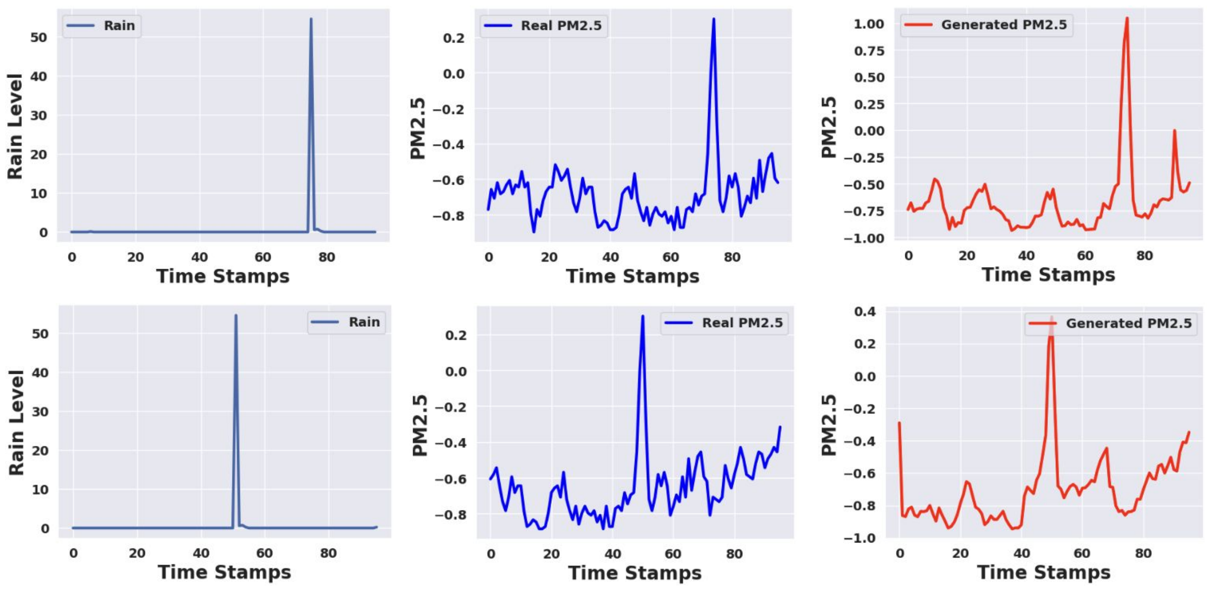

Figure 11. **TIME WEAVER retains causal relationships between the input metadata and the generated time series.** In this figure, we showcase TIME WEAVER's ability to retain essential causal relationships between metadata and generated time series for the Air Quality dataset. In particular, we consider the effects of rainfall (metadata) on the particulate matter levels (PM2.5, one of the channels in the generated time series). We show two instances from the dataset with a sudden spike in rainfall. In both rows, the left image shows the rainfall level metadata and the center image shows the PM2.5 time series corresponding to the rainfall metadata from the dataset. The right image shows the generated PM2.5 time series channel for the rainfall metadata input. Observe that the sudden spike in the particulate matter level is faithfully replicated in the generated samples.

A.10. Discussion On The Feature Extractors Used In The J-FTSD Metric

In this section, we analyze the embeddings from the feature extractors used to compute J-FTSD. Note that these feature extractors (ϕ_{time} and ϕ_{meta}) are learned using a contrastive learning-based training approach similar to CLIP (Radford et al., 2021). The feature extractors are jointly trained (check Alg. 1) to maximize the similarity between a time series embedding and its paired metadata embedding. This results in the clustering of time series embeddings for time series samples that have similar paired metadata. We test this observation on the ECG and the Air Quality datasets and show low-dimensional visualization of the time series embeddings in Fig. 12.

We use t-SNE on the time series embeddings for dimensionality reduction. For the ECG dataset, we observe that time series samples corresponding to the most common heart disease statement get clustered together. This is shown in Fig. 12. Similarly, for the Air Quality dataset, we observe that the time series samples get clustered into 4 clusters. However, the dataset has a combination of categorical metadata conditions such as stations, years, months, etc, and continuous metadata conditions such as temperature, pressure, rain levels, etc. This complexity in metadata conditions inhibits us from interpreting the clusters, as in the case of the ECG dataset. Therefore, our critical insight is that the contrastive learning approach results in the clustering of the time series embeddings corresponding to similar paired metadata. However, the interpretability of the clusters is non-trivial for arbitrary combinations of categorical, continuous, and time-varying metadata conditions.

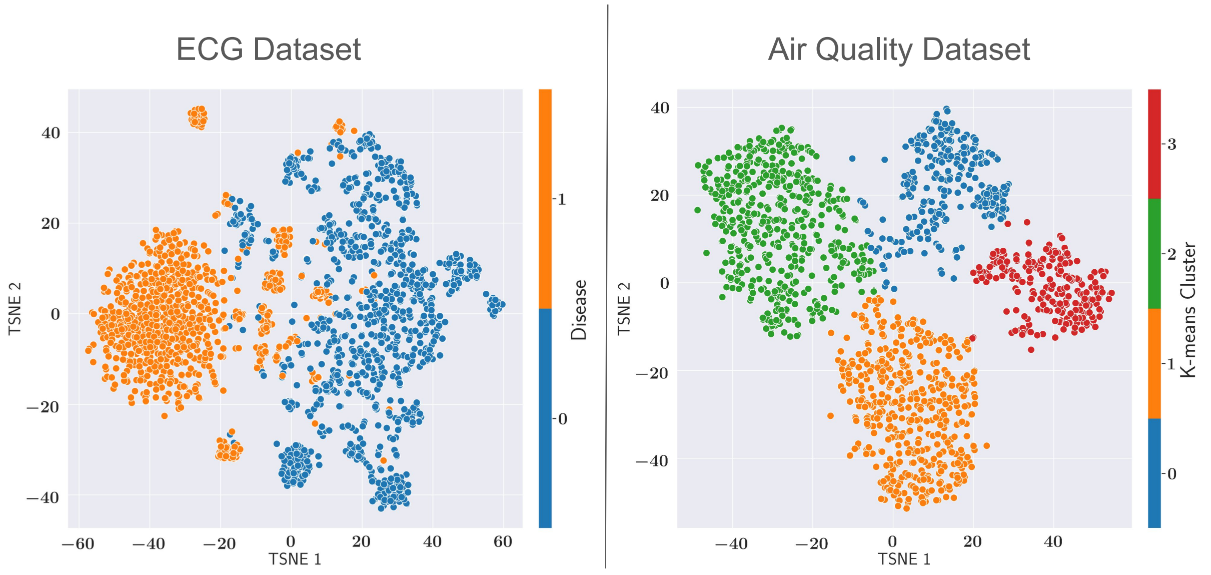

Figure 12. **Interpretability of the time series embeddings obtained from J-FTSD's time series encoder.** In this figure, we show the clustering of time series embeddings obtained from the time series feature extractor ϕ_{time}. We use t-SNE on the time series embeddings for dimensionality reduction. Note that the feature extractors (ϕ_{time} and ϕ_{meta}) are trained jointly using a contrastive learning-based approach (check Alg. 1). We show results for clustering on two datasets - ECG and Air Quality. For the ECG dataset, we showcase that the samples corresponding to the most common disease get clustered together. For the Air Quality dataset, we show that the time series embeddings get clustered into 4 clusters. However, due to the complexity of the metadata conditions (a combination of categorical and time-varying continuous conditions), we cannot interpret the clusters as in the case of the ECG dataset.

A.11. Additional Qualitative Results

In this section, we provide additional qualitative results generated using TIME WEAVER.

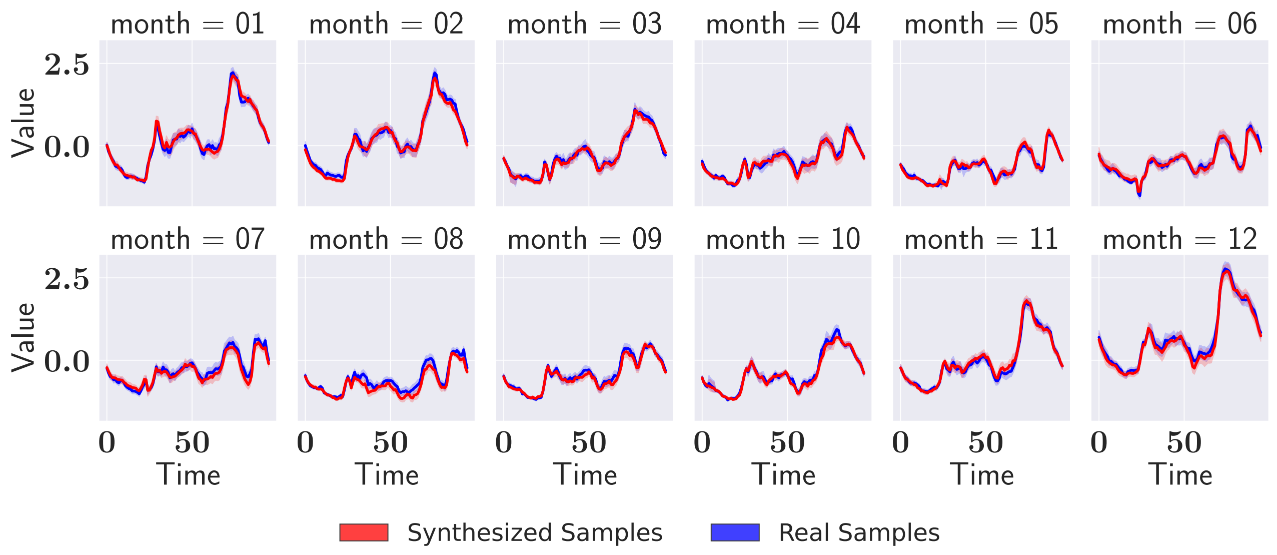

Figure 13. **Generated time series samples from the Electricity dataset**

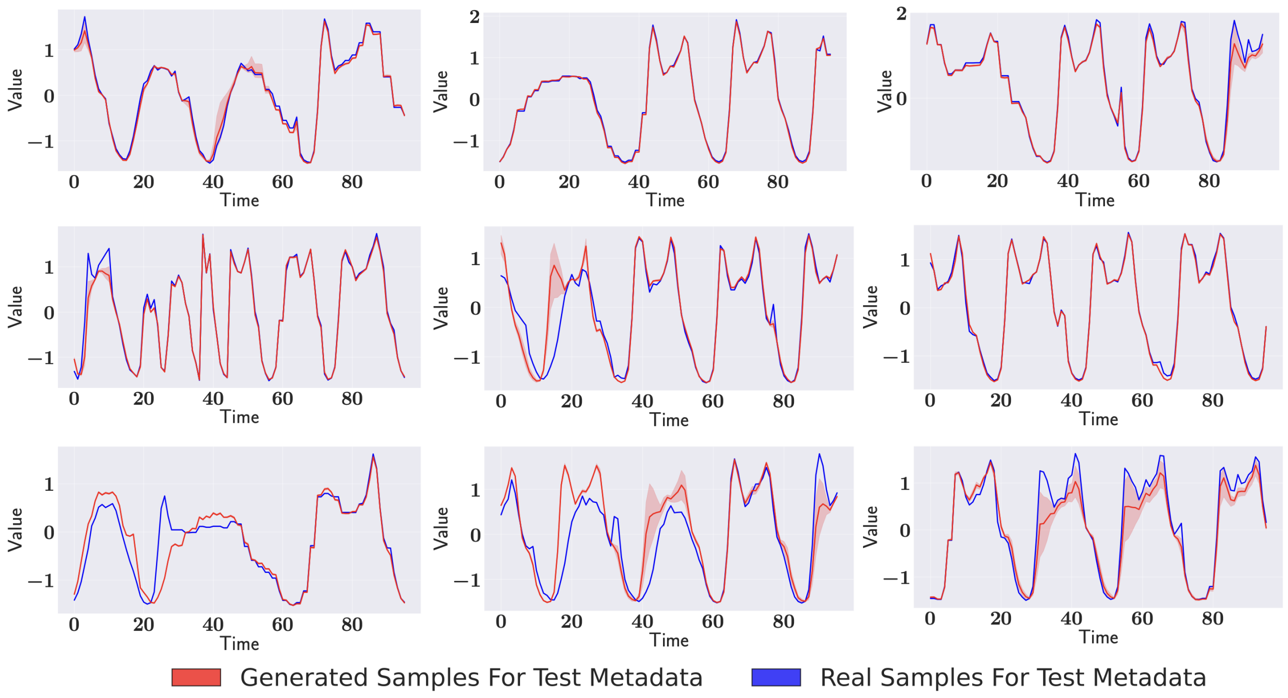

Figure 14. **Qualitative Results from the TIME WEAVER-CSDI model for the Traffic dataset**

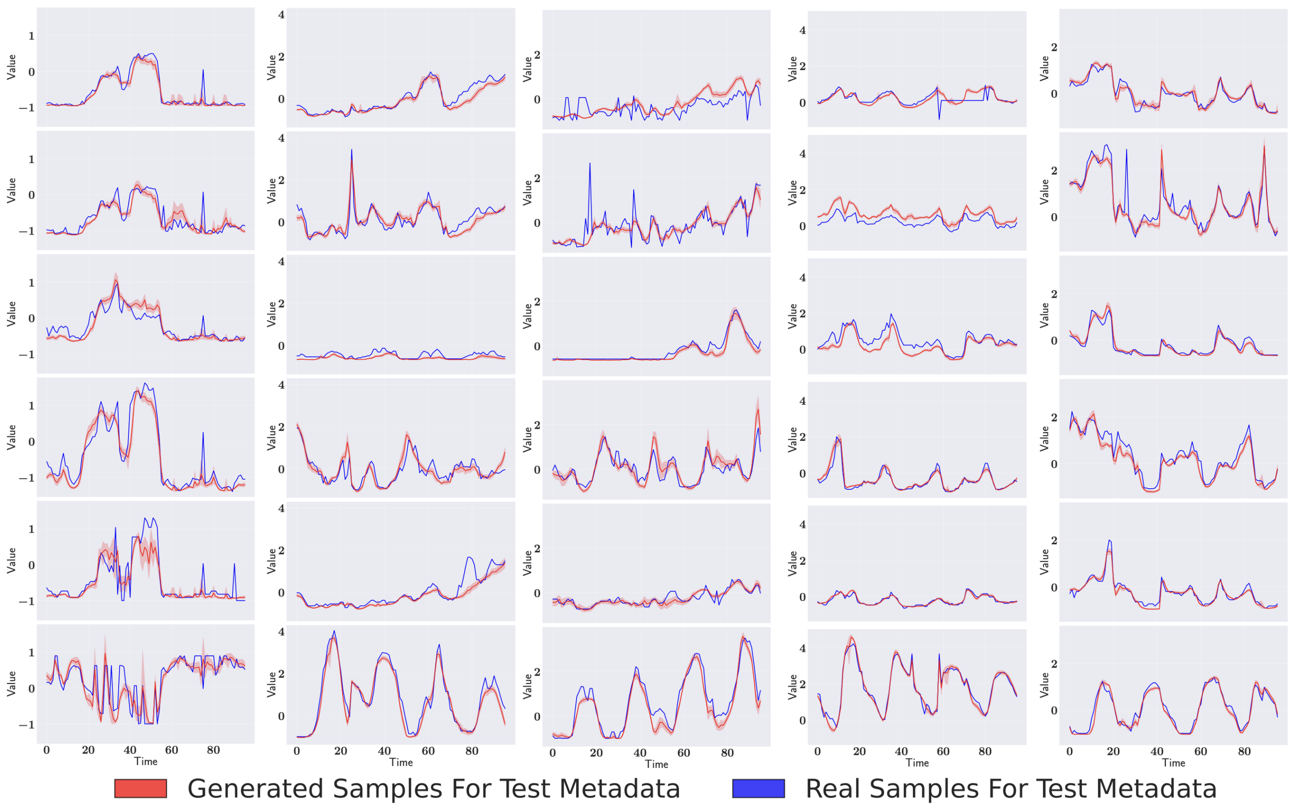

Figure 15. **Qualitative results from the TIME WEAVER-CSDI model for the Air Quality dataset**

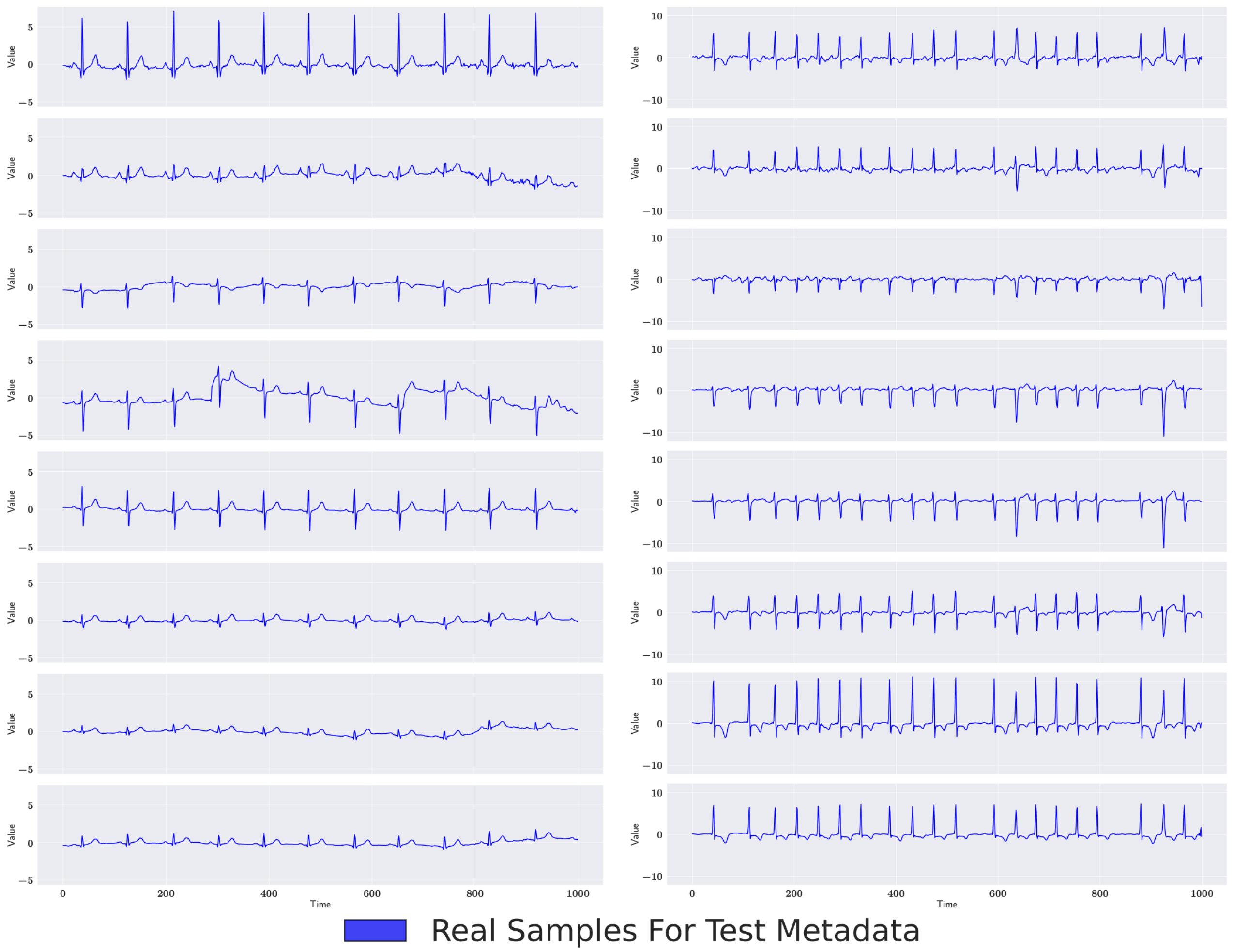

Figure 16. **Real time series samples from the ECG dataset**

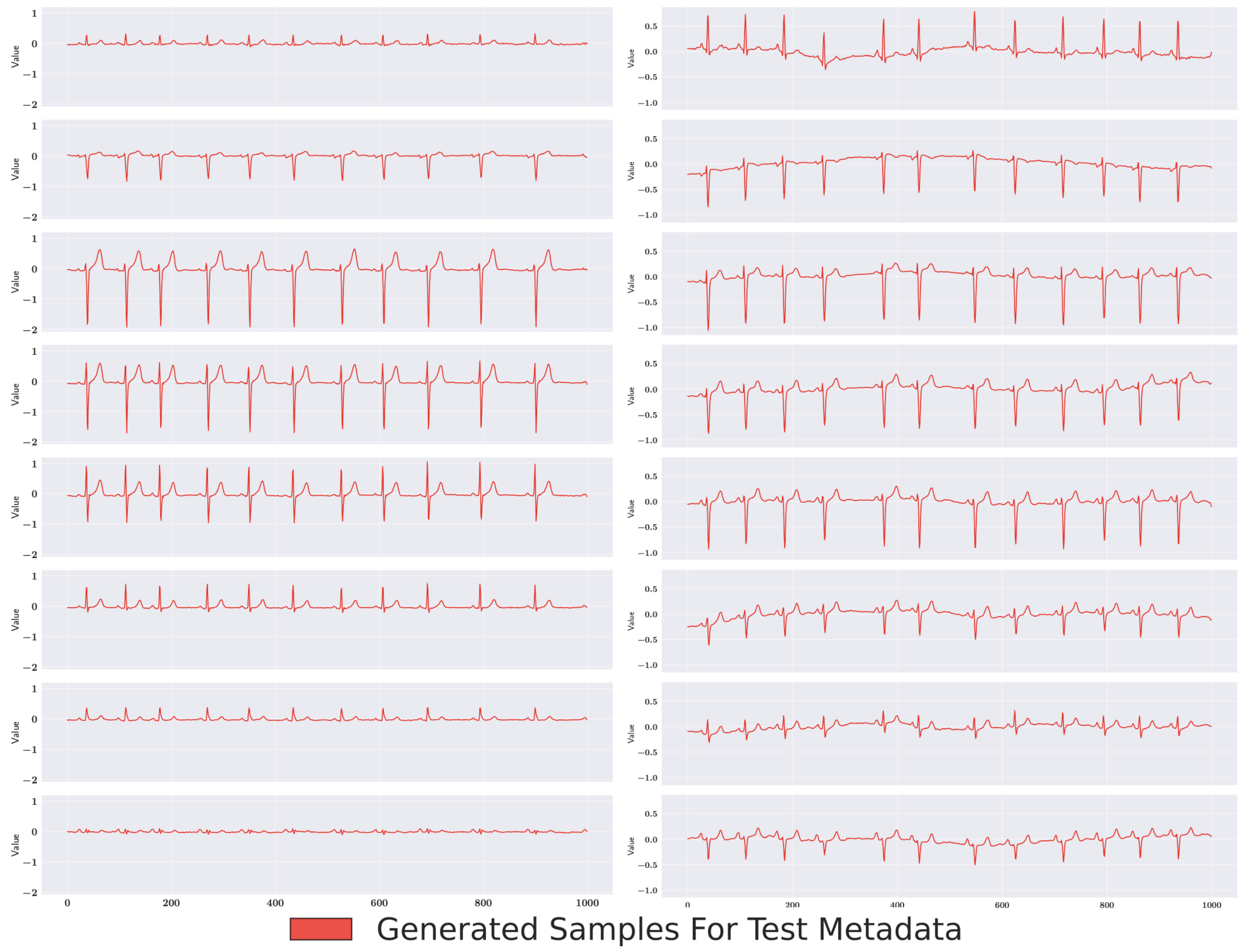

Figure 17. **Generated time series samples from the ECG dataset**